Applied Abstract Algebra

D. Joyner, R. Kreminski, J. Turisco

2-1-2003

Contents

Preface

This text is based on notes which have been used in teaching at Texas A& M University - Commerce, and at the US Naval Academy, for several semesters. The notes were used to to teach two different courses. One, a first course in modern algebra, was taught to students who have only had a "fundmentals in mathematics" course and a matrix theory course. (So, they knew, for example, mathematical induction and how to compute eigenvalues of a matrix). Applications were emphasized and it had a once-a-week computer lab. This course covered some of chapters 1,2,3,4 and 5. A second course was a course on error-correcting codes. This course covered chapters 2 and 3 in more detail, and parts of chapter 6. It also had a once-a-week computer lab.

So, on one hand this book aims to teach some undergraduate modern algebra with an emphasis on applications. These applications are mostly focused on error-correcting codes, the Rubik's cube and cryptography. Special projects and applications sections in the text develop some of these. The special projects are suggestions for the student taking a course which requires a term paper. Topics include: Latin squares, continued fractions, Diffue-Hellman key exchange, RSA for polynomials, tiling problems, campanology (bell-ringng), and special constructions involving the Rubik's cube. One does need a Rubik's cube to follow the sections on the Rubik's cube, any more than one needs bells to follow the section on campanology. We do not discuss solution strategies or go into the mathematics of the Rubik's cube, as in [J1] for example, but we occasionally use it as a teaching tool to illustrate an idea.

Each chapter has exercises in GAP (a free computer algebra system, [Gap]) and MAGMA (a non-commercial but not free computer algebra system, [MAGMA]). Instructions for using GAP and MAGMA may be found in the appendices at the end of the book. The web site [Gap] contains detailed instructions for installing GAP on windows, mac, and unix/linux computers. Both GAP and MAGMA have extensive capabilities to do modern algebra. Since many of the applications are oriented to communications, it is natural to use computers to understand these applications. In fact, for coding theory and cryptography, many systems use such large parameters that working practical exercises "by hand" is virtually impossible and computers are not just useful but required. Moreover, most chapters end with some "special projects", which might be useful for a student class project. Some related

MAPLE and MATHEMATICA code can be found on the author's websites, though for various reasons we decided not to include them in this book.

On the other hand, this book can be used to teach an undergraduate course in error-correcting codes. The GAP and MAGMA exercises are especially helpful for giving the student a grasp of examples. After such a course, the student should be prepared to move onto a more serious treatment, such as [MS].

The text begins with elementary number theory and algebraic properties of the integers. Modular arithmetic especially is covered. Here the reader finds applications to the RSA cryptosystem, linear feedback shift registers, and the Diffie-Hellman key exchange, for example.

The next chapter builds on the first by analogy: The polynomials are analogous to the integers and quotient rings are analogous to modular arithmetic. In fact, some proofs in this chapter are almost exactly the same as the analogous theorems in the first chapter. Properties of quotient rings are covered, with an eye towards applications to cyclic error-correcting codes and linear feedback shift registers. There is a short section on Gröbner bases, which have shown to be extremely useful in applications to a variety of topics. We restrict ourselves to a few very simple applications. "Nimbers" and factorization techniques for polynomials are covered as special topics.

The third chapter introduces error-correcting codes. Basic topics, such as check matrices, generator matrices, and syndrome decoding are covered. Special codes such as the Hamming codes and cyclic codes are also covered. Cyclic codes are a very broad class of codes which includes, for example Reed-Solomon codes, the codes used on todays CDs. Reed-Solomon codes are briefly discussed as well. More codes are covered in the special topics section and in the computer exercises.

The fourth chapter presents some material on permutations needed for group theory. Some applications to campanology and the Rubik's cube are discussed. Another application: the Nokia 7160 cell phone has a game called "rotation", §4.7 gives some useful moves to help solve this game. (You don't need the cell phone to play rotation - numbered pieces of paper on a table work fine.)

The next chapter, on group theory, discusses the theory of groups in more detail. The "standard treatment", which emphasizes the classification of groups, is put aside in favor of a treatment which emphsizes examples and computations. Cyclic groups, the general linear group $GL(n, \mathbb{R})$ and the symmetric group are examples which are emphasized. The Cayley graph of a

group, homomorphisms, and quotient groups are introduced. The automorphism group of a linear error-correcting code and more campanology and the theory of the Rubik's cube are discussed as applications. The computer exercises for both chapter 4 and chapter 5 are put together at the end of this chapter.

In chapter 6 more advanced topics in error-correcting codes are presented. Since some codes are best introduced after knowing some group theory, they are defined here. Other codes, such as the Goppa codes and the low density parity check codes, do not need group theory but were too advanced to fit naturally into chapter 3. The computer exercises explore these interesting codes in more detail.

Some of chapters 4 and 5 borrow heavily from one of the author's previous book, [J1]. As with [J1], all of that author's royalties go directly to charity.

Nokia and Rubik's cube are registered trademarks (TM). We shall omit the symbol (TM) for readability.

We thank Gene Berg, Don Newhart, and Amin Shokrolahi for answering lots of questions on coding theory. Of course, any errors remaining are entirely the authors' responsibility.

Chapter 1

Some elementary number theory

1.1 Introduction

Our ultimate goal is to understand applications of mathematics to computers and electronic communications. Since computers speak a language of 0's and 1's, we want to develop so-called "Boolean algebra" or "mod 2 arithmetic". It turns out to be not much harder to develop "mod m arithmetic" and this is done in this chapter.

A lot of abstract algebra either requires properties of the integers, or is based by analogy on the integers, or is inspired by a property of the integers. This is why we begin our story there. In this chapter, we shall study divisibilty properties of the integers, modular arithmetic, and properties of the primes. The general idea is that we wish to learn how to solve some simple types of equations in the integers and also "modulo m". This will help us introduce finite fields later. Along the way, we will develop some interesting techniques, see some useful applications to cryptography, and have some fun with more "recreational" topics such as calendar calculations and the game of Nim.

The key computational methods are

- the Euclidean algorithm,
- the "repeated squaring" algorithm,

and the key results are

- the Fundamental Theorem of Arithmetic,
- Euler's theorem (and, the special case of Fermat's last theorem),

1.2 Divisibility

We will use the following notation throughout this book. The **integers** are

$$\mathbb{Z} = \{..., -2, -1, 0, 1, 2, ...\}$$

and the **natural numbers** are

$$\mathbb{N} = \{1, 2, ...\}.$$

The rational numbers will be denoted by $\mathbb{Q}$. They are the number which can be written as fractions:

$$\mathbb{Q} = \{\frac{a}{b} \mid a, b \in \mathbb{Z}, b \neq 0\}.$$

The real numbers will be denoted by $\mathbb{R}$ and the complex numbers will be denoted by $\mathbb{C}$.

Definition 1.2.1. *Given integers a and b we say a* **divides** *b (or a is a* **factor of** *b, or a is a* **divisor of** *b, or b is a* **multiple of** *a) if there is an integer k such that $b = ak$. In this case, we write*

$$a|b \quad \text{or} \quad b \equiv 0 (\text{mod } a).$$

If a does not divide b then we write

$$a \not| b \quad \text{or} \quad b \not\equiv 0 (\text{mod } a).$$

We call an expression $a = bd$ (a, b, d integers) a **factorization** *of a.*

Since 1 divides all integers, we sometimes call 1 a "trival factor" and the factors greater than 1 the "non-trivial factors".

Definition 1.2.2. *An integer $n > 1$ which has no positive factors other than itself and 1 is called* **prime**. *A number $n > 1$ which is not prime is called a* **composite**.

Example 1.2.3. $12 \equiv 0 (\text{mod } 3)$, $-18|0$, $1|7$, $3 \not| 5$, *the positive factors of* 12 *are* $1, 2, 3, 4, 6, 12$.

Lemma 1.2.4. *If* $d \mid a$ *and* $d \mid b$ *then* $d \mid (ax+by)$, *for any integers* x *and* y.

proof: $d \mid a$ implies that $a = dc$ and $d \mid b$ implies that $b = df$. Therefore:

$$ax + by = dcx + dfy = d(cx + fy)$$

i.e. $d|(ax + by)$. □

A problem on cryptography which we will encounter later in this chapter involves factoring large numbers. In particular, consider the following question: What is the fastest general procedure to find the smallest non-trivial positive factor of an integer $n > 0$? This question may seem simple but if n is very large the answer is far from obvious. For example, consider the following 309-digit number

```
13506641086599522334960321627880596993888147560566702752448514385152
65106048595338339402871505719094417982072821644715513736804197039641
91743046496589274256239341020864383202110372958725762358509643110564
07350150818751067659462920556368552947521350085287941637732853390610
97505443349998111500569772368909275 63
```

This number is called RSA-1024, since it has 1024 binary digits. The sum of its digits is 1369. If you can factor this number then RSA Security Inc will give you one hundred thousand dollars! See

`http://www.rsasecurity.com/rsalabs/challenges/factoring/index.html`

for more details.

However, if n is relatively small then there are some easy strategies to follow to factor it.

Lemma 1.2.5. *(Basic factoring strategy) If a positive integer* n *has a factorization* $n = ab$ *(*a, b *integers) then either* $a \leq \sqrt{n}$ *or* $b \leq \sqrt{n}$.

proof: Suppose that this statement were false, so $a > \sqrt{n}$ and $b > \sqrt{n}$. Then $n = ab > \sqrt{n}\sqrt{n} = n$. This is a contradiction, so either $a \leq \sqrt{n}$ or $b \leq \sqrt{n}$. □

The result above implies that,

- if n is composite then the smallest non-trivial factor d of n must satisfy $1 < d \leq \sqrt{n}$,

- if n has no factor d where $1 < d < \sqrt{n}$ then n must be a prime.

Example 1.2.6. *Let* $n = 1233$. *By the above fact, to find a positive factor of* 1233, *we need only check if the integers* $1, 2, 3, ..., 35$ *divide* n *since* $\sqrt{1233} = 35.11....$ *Moreover, any such divisor must be odd is* 1233 *is odd. Our first try,* $1233/3$ *turns out to be an integer and we get* $1233 = 3 \cdot 411$.

What about the factors of 411*? By the above fact, to find a positive factor of* 411, *we need only check if the integers* $1, 2, 3, ..., 20$ *divide* 441 *since* $\sqrt{411} = 20.273....$ *Again, it must be odd and our first try,* $411/3$ *turns out to be an integer and we get* $411 = 3 \cdot 137$.

To factor 137 *we check if* $1, 2, 3, ..., 11$ *divide* 137 *since* $\sqrt{137} < 12$. *Trying all the odd numbers* $3, 5, 7, 9, 11$, *we see that none of them are divisors of* 137.

The complete factorization is $1233 = 3 \cdot 3 \cdot 137$.

The following notation is quite useful, as we will see in the next section.

The set of all multiples of b is denoted

$$b\mathbb{Z} = \{..., -2b, -b, 0, b, 2b, ...\} = \{m \in \mathbb{Z} \mid m = bk, \text{ for some integer } k\}.$$

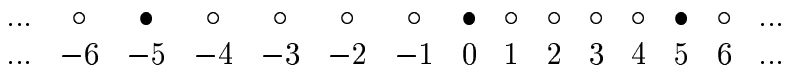
This may be visualized as the first column of the array which you get by listing all the integers in strings of b-at-a-time,

⋮				⋮
-b	-b+1	...	-2	-1
0	1	...	b-2	b-1
b	b+1	...	2b-2	2b-1
⋮				⋮

(1.1)

or as a real number line with a "notch" every b units.

The multiples of 5



which we may also visualize as the first column of the array

⋮				⋮
-5	-4	-3	-2	-1
0	1	2	3	4
5	6	7	8	9
⋮				⋮

(1.2)

1.2.1 Division algorithm

The next result basically says that, for $b > 0$ every integer a occurs in one of the columns in the array (1.1) above. It also goes by the name "long division".

Theorem 1.2.7. *(***Division algorithm***) Let a and $b > 0$ be integers. There are integers q (the* **quotient***) and r (the* **remainder***) such that*

$$a = bq + r, \qquad 0 \leq r < b.$$

Furthermore, q and r are unique.

Our proof of the division algorithm depends on the following axiom.

Axiom 1.2.8. *(Well-ordering principle) Each non-empty set of natural numbers contains a least element.*

In particular, each set S of integers which contains at least one non-negative element must contain a smallest non-negative element. Furthermore, this smallest element is unique.

We begin the proof of the division algorithm by showing that q and r are unique. Suppose $a = bq + r = bq_1 + r_1$, where $0 \leq r < b$, $0 \leq r_1 < b$. Then $0 = b(q - q_1) + (r - r_1)$. Since b divides the right side and the first term of the left side of the above equation, b must divide $r - r_1$. But $-b < r - r_1 < b$, so $r - r_1 = 0$. Therefore r is unique. Since $bq + r = bq_1 + r_1$, this in turn implies q is also unique.

We present two proofs of the existence of q and r in the division algorithm.

First proof of Division algorithm: First, we *claim* that a must lie in between some two consecutive elements of

$$b\mathbb{Z} = \{..., -2b, -b, 0, b, 2b, ...\},$$

say $qb \leq a < (q+1)b$ (see (1.1) for a visual representation of this idea). Then $0 \leq a - qb < b$. Therefore, $a = qb + r$, where $r = a - qb$. This proves the division algorithm, assuming the truth of the claim (which is left as an **exercise**).

Second proof of Division algorithm: Consider the set

$$S = a + b\mathbb{Z} = \{a + nb \mid n \in \mathbb{Z}\} = \{..., a - 2b, a - b, a, a + b, a + 2b, ...\}.$$

This contains at least one non-negative element, for example, $a + |a|b \geq a + |a| \geq 0$. By the well-ordering principle, S contains a least non-negative element, say r. By definition of S, there is an integer q such that $r = a - qb$. It remains to show $r < b$. Suppose not (to get a contradiction), i.e., suppose $r \geq b$. Then $0 \leq r - b = a + (-q - 1)b \in S$, and $r - b < r$, so r was not the smallest non-negative element of S. This contradiction shows that the hypothesis $r \geq b$ is false. Therefore $r < b$ and the proof is complete. □

About 200 years ago in Germany, at the age of 23 C. F. Gauss [1] published **Disquisitiones Arithmeticae**, essentially an expanded version of his PhD thesis, that revolutionized the study of number theory. One piece of new notation which Gauss introduced is the **congruence** or **modulus** notation. Let a, b, m be integers. We say that a **is congruent to** b **modulo** m, written

$$a \equiv b \pmod{m},$$

if $m|(a - b)$. In this case, m is called the **modulus** and b is a **residue** of a modulo m.

The division algorithm may be restated in this new notation as follows.

Theorem 1.2.9. *If a and $m > 0$ are integers then there is an integer $r \in \{0, 1, ..., m - 1\}$ satisfying*

$$a \equiv r \pmod{m} .$$

In other words, each integer has a residue mod m (called the **least residue**)) which is in the range $0, 1, ..., m-1$. Furthermore, this residue can be computed using the division algorithm.

[1] An often told story (the exact details of which depend on the source) of Gauss as a boy: As a 10 year old student in school, Gauss astonished his teacher who, wanting a break, told his class to add up all the numbers from 1 to 100. Gauss figured out the answer in a matter of seconds. He had noticed that by pairing 1 and 100, 2 and 99, ..., 50 and 51, all the paired numbers added to 101 and that there were 50 such pairs, so the answer was 5050.

Example 1.2.10. *We have*

- $71 \equiv 1 \pmod 7$,
- $147 \equiv 3 \pmod{12}$,
- $km \equiv 0 \pmod m$ *for any integers* k *and* $m > 0$, *and*
- $-1 \equiv 10 \pmod{11}$.

1.2.2 The greatest common divisor and least common multiple

In this section, we introduce two commonly used and useful notations in number theory, the greatest common divisor and least common multiple.

Definition 1.2.11. *Let* $a > 0$ *and* $b > 0$ *be integers. The* **greatest common divisor** *of* a, b *is the largest integer* $d > 0$ *satisfying both* $d|a$ *and* $d|b$. *The greatest common divisor is denoted by either* $gcd(a, b)$ *or simply by* (a, b) *when there is no ambiguity.*

The greatest common divisor of a and b is simply the largest positive integer which divides both a and b.

Definition 1.2.12. *When* $gcd(a, b) = 1$ *then we say that* a, b *are* **relatively prime**.

Example 1.2.13. $gcd(12, 15) = 3$, $gcd(3, 5) = 1$, $gcd(100, 46) = 2$.

To return to the question above, we now know that the set of all possible sums and differences of two integers $a > 0$ and $b > 0$ is of the form $c\mathbb{Z}$, for some $c > 0$. What is c? This question is answered in the section on the Euclidean algorithm later in this chapter

As we saw above, the gcd is the largest integers which divides both a and b. Somewhat analogous to this is the least integer which both a and b divide.

Definition 1.2.14. *Let* $a > 0$ *and* $b > 0$ *be integers. The* **least common multiple** *of* a, b *is the smallest integer* $m > 0$ *satisfying both* $a|m$ *and* $b|m$. *The least common multiple is denoted by either* $lcm(a, b)$ *or sometimes simply by* $[a, b]$ *(also,* $\{a, b\}$ *is sometimes used).*

Example 1.2.15. $lcm(12,15) = 60$, $lcm(3,5) = 15$, $lcm(100,46) = 2300$.

The lcm can be computed from the gcd (which can be computed using the Euclidean algorithm) using the following fact.

Proposition 1.2.16. *Let a, b, c be integers.*

(1) $gcd(a,b)lcm(a,b) = ab$.

(2) If $a|bc$ *then* $a|gcd(a,b)c$.

(3) If $a|bc$ *and* $gcd(a,b) = 1$ *then* $a|c$.

(4) If $a|c$ *and* $b|c$ *and* $gcd(a,b) = 1$ *then* $ab|c$.

(5) $gcd(ab,c) = 1$ *if and only if* $gcd(a,c) = 1$ *and* $gcd(b,c) = 1$.

(6) If $c|a$ *and* $c|b$ *then* $c|gcd(a,b)$.

(7) If $a|c$ *and* $b|c$ *then* $lcm(a,b)|c$.

(8) If $d = gcd(a,b)$ *then* $gcd(a/d, b/d) = 1$.

(9) For any integers m, n *we have* $gcd(a,b)|(ma+nb)$.

The reader can use the examples above to verify (1) in the cases $(a,b) = (12,15), (3,5), (100,46)$.

proof of (1): We will show that $\frac{ab}{(a,b)} = lcm(a,b)$. Note that $\frac{b}{(a,b)} \in \mathbb{Z}$, since (a,b) divides b. Therefore, $a \cdot \frac{b}{(a,b)} \in \mathbb{Z}$.

Since $\frac{\frac{ab}{(a,b)}}{a} = \frac{b}{(a,b)} \in \mathbb{Z}$, we know that $\frac{ab}{(a,b)}$ is a multiple of a. Similarly, $\frac{\frac{ab}{(a,b)}}{b} = \frac{a}{(a,b)} \in \mathbb{Z}$ (why?) implies that $\frac{ab}{(a,b)} \geq lcm(a,b)$.

Now $\frac{ab}{lcm(a,b)} \in \mathbb{Z}$ (why?) and $\frac{a}{\frac{ab}{lcm(a,b)}} = \frac{lcm(a,b)}{b} \in \mathbb{Z}$ since b divides $lcm(a,b)$. Therefore, $\frac{ab}{lcm(a,b)}$ divides a. Similarly, $\frac{ab}{lcm(a,b)}$ divides b. Thus:

$$\frac{ab}{lcm(a,b)} \leq (a,b) \quad \text{i.e.,} \quad \frac{ab}{(a,b)} \leq lcm(a,b).$$

Since we have shown the inequality in both directions, we must have equality. □

Later, part (1) of this proposition shall be proven again, but as a consequence of the Fundamental Theorem of Arithmetic.

We prove (2) below since it will be used in the proof of the Fundamental Theorem of Arithmetic.

proof of (2): For the proof of this part, we shall use a result which will not be proven until later in this chapter. Assume $a|bc$. Let $d = gcd(a, b)$. By the Euclidean algorithm (Theorem 1.4.3 below), there are integers x, y such that $ax + by = d$, so $acx + bcy = dc$. Since a divides acx and bcy (by assumption), it divides $acx + bcy$, hence $a|gcd(a, b)c$. □

Part (4) shall also be proven later as a consequence of the Fundamental Theorem of Arithmetic. Parts (3), (6), (8), (9) are left as exercises.

proof of (7): We will prove this by contradiction. Suppose that $lcm(a, b) \nmid c$. Then $\frac{c}{lcm(a,b)}$ is not an integer, and we may write it as:

$$\frac{c}{lcm(a, b)} = q + \alpha, \quad \text{where} \quad 0 < \alpha < 1.$$

Multiplying both sides of the above inequality by the positive integer $lcm(a, b)$ gives: $c = q \cdot lcm(a, b) + \alpha \cdot lcm(a, b)$. Let $r = c - q \cdot lcm(a, b)$. Then:

$$c = q \cdot lcm(a, b) + r, \quad \text{with} \quad 0 < r < lcm(a, b).$$

But $a \mid r$ and $b \mid r$, so r is a common multiple of a and b which is less than the least common multiple $lcm(a, b)$, by the above inequality. Thus we have reached a contradiction, and our original assumption was false. We conclude that $lcm(a, b) \mid c$. □

Exercise 1.2.17. *Find* $gcd(24, 78)$ *and* $lcm(24, 78)$ *and verify that their product is* $24 \cdot 78$.

Exercise 1.2.18. *Find all integers* $d > 0$ *such that* $12 \mid d$ *and* $d \mid 260$.

Exercise 1.2.19. *Find* q *and* r *guaranteed by the division algorithm for* $a = 28$ *and* $b = 160$.

Exercise 1.2.20. *Show that if* $a = bq + r$ *then* $gcd(a, b) = gcd(r, b)$.

Exercise 1.2.21. *Prove the claim in the first proof of the division algorithm, theorem 1.2.7.*

Exercise 1.2.22. *For any integer n, prove that:*
(a) 2 divides $n(n+1)$,
(b) 3 divides $n(n+1)(n+2)$.

Exercise 1.2.23. *Prove that if $d = gcd(a,b)$ then $gcd(\frac{a}{d}, \frac{b}{d}) = 1$. (In other words, $\frac{a}{d}$ and $\frac{b}{d}$ are relatively prime.)*

Exercise 1.2.24. *Show that for any integers m and n, $gcd(a,b)$ divides $ma+nb$.*

Exercise 1.2.25. *Prove that every square integer is of the form $4k$ or $4k+1$, where k is an integer.*

Exercise 1.2.26. *Prove that 4 does not divide n^2+2, for any integer n.*

Exercise 1.2.27. *Prove that if 3 does not divide an odd integer n, then 24 divides n^2-1.*

Exercise 1.2.28. *Find*
(a) $lcm(6,21)$,
(b) $lcm(10,15)$,
(c) $gcd(99,100)$.
(d) $gcd(24,78)$,
(e) $lcm(24,78)$.

Exercise 1.2.29. *Suppose $a > 1$ is an odd integer. Find a formula for $lcm(a, a+2)$.*
Suppose $a > 1$ is an even integer. Find a formula for $lcm(a, a+2)$.

Exercise 1.2.30. *Verify that (6) in Proposition 1.2.16 is true.*

Exercise 1.2.31. *Verify that (8) in Proposition 1.2.16 is true.*

Exercise 1.2.32. *Verify that (9) in Proposition 1.2.16 is true.*

Exercise 1.2.33. *Suppose $a > 1$ is an integer. Find $gcd(a, a+1)$.*

Exercise 1.2.34. *Suppose $a > 1$ is an odd integer. Find $gcd(a, a+2)$.*
Suppose $a > 1$ is an even integer. Find $gcd(a, a+2)$.

Exercise 1.2.35. *(a) Find $gcd(51,15)$.*
(b) Find $gcd(51,105)$.
(c) Find $gcd(501,105)$.

Exercise 1.2.36. *Let* $\left(\begin{array}{c} n \\ k \end{array} \right) = \frac{n!}{k!(n-k)!}$ *denote the combinatorial symbol* n **choose** k *(also called a* **binomial coefficient***), where* $m! = 1 \cdot 2 \cdot ... \cdot m$ *is* m **factorial**. *This is the coefficient of* $x^k y^{n-k}$ *in the binomial expansion of* $(x+y)^n$,

$$(x+y)^n = x^n + \left(\begin{array}{c} n \\ 1 \end{array} \right) xy^{n-1} + ... \left(\begin{array}{c} n \\ k \end{array} \right) x^k y^{n-k} + ... + \left(\begin{array}{c} n \\ n-1 \end{array} \right) x^{n-1} y + y^n.$$

These can be computed by hand using **Pascal's triangle**,

$$\begin{array}{ccccccccccc} & & & & & & 1 & & & & \\ & & & & & 1 & & 1 & & & \\ & & & & 1 & & 2 & & 1 & & \\ & & & 1 & & 3 & & 3 & & 1 & \\ & & 1 & & 4 & & 6 & & 6 & & 4 & & 1 \\ & 1 & & 5 & & 10 & & 10 & & 5 & & 1 & \\ 1 & & 6 & & 15 & & 20 & & 15 & & 6 & & 1 \\ & & & & & & ... & & & & \end{array}$$

Check $5 | \left(\begin{array}{c} 5 \\ k \end{array} \right)$, *for* $k = 1, 2, 3, 4$.

Exercise 1.2.37. *Factor all the integers* $100, 101, 102, 103, 104, 105$.

Exercise 1.2.38. *Show that* $7 | \left(\begin{array}{c} 7 \\ k \end{array} \right)$, *for* $k = 1, 2, 3, 4, 5, 6$. *Show that* $11 | \left(\begin{array}{c} 11 \\ k \end{array} \right)$, *for* $k = 1, 2, ..., 10$. *Is there some* $k = 1, 2, ..., 8$ *such that* $9 | \left(\begin{array}{c} 9 \\ k \end{array} \right)$ *?*

1.3 Binary and m-ary notation

Each natural number is most commonly written in **decimal form** (or **base** 10),

$$a = a_k 10^k + ...a_1 10 + a_0,$$

where $0 \leq a_i \leq 9$ are the **digits**. (Without loss of generality we may assume that the leading digit a_k is non-zero.) This representation is unique. (In spite of the fact that the decimal representation of a *real number* is not unique - $1.0 = .9999.....$) Similarly, each natural number can be written (uniqely) in a **binary expansion** (or **base** 2),

$$a = a_k 2^k + ... + a_1 2 + a_0,$$

where $0 \leq a_i \leq 1$ are the **bits**. The **binary representation** of a is written as $a \sim a_k...a_1 a_0$. Clearly, a is even if and only if $a_0 = 0$. To find the binary expansion of a natural number, perform the following steps.

(1) Find the "leading bit" a_k by determining the largest power of 2 less than or equal to a. Call this power k and let $a_k = 1$.

(2) Subtract this power from a and replace a by this difference.

(3) If the result is non-zero, go to step 1; otherwise, stop.

This determines all the non-zero bits in the binary representation of a. The other bits are 0.

Example 1.3.1. *Find the binary representation of* 130. *The largest power of* 2 *less than or equal to* 130 *is* $128 = 2^7$, *so* $a_7 = 1$. *The largest power of* 2 *less than or equal to* $2 = 130 - 128$ *is* $2 = 2^1$, *so* $a_1 = 1$. *The other bits are zero:* $a_6 = a_5 = a_4 = a_3 = a_2 = a_0 = 0$, *so*

$$130 = 128 + 2 \sim 10000010.$$

More generally, let us fix an integer $m > 1$. Each natural number can be written in an m**-ary expansion**

$$a = a_k m^k + ... + a_1 m + a_0,$$

where $0 \leq a_i \leq m - 1$ are the m**-ary digits**. Again, the m**-ary representation** of a is written as $a \sim a_k...a_1 a_0$. Clearly, $m|a$ if and only if $a_0 = 0$. To find the m-ary expansion of a natural number, perform the following steps.

(1) Find a_k by determining the largest power of m less than or equal to a. Call this power k.

(2) Find the largest positive integer multiple of this power which is less than or equal to a. This multiple will be the k-th digit a_k.

(3) Subtract $a_k m^k$ from a and replace a by this difference.

(4) If the result is non-zero, go to step 1; otherwise, stop.

Example 1.3.2. *Find the* 3*-ary representation of* 211. *The largest power of* 3 *less than or equal to* 211 *is* $81 = 3^4$, *and* $211 > 2 \cdot 81 = 162$ *so* $a_4 = 2$. *The largest power of* 3 *less than or equal to* $49 = 211 - 162$ *is* $27 = 3^3$, *and* $49 < 2 \cdot 27$ *so* $a_3 = 1$. *The largest power of* 3 *less than or equal to* $22 = 49 - 27$ *is* $9 = 3^2$, *and* $22 > 2 \cdot 9$ *so* $a_2 = 2$. *The largest power of* 3 *less than or equal to* $4 = 22 - 18$ *is* $3 = 3^1$, *and* $4 < 2 \cdot 3$ *so* $a_1 = 1$. *The last "bit" is 1:* $a_4 = 2, a_3 = 1, a_2 = 2, a_1 = 1, a_0 = 1$, *so*

$$211 = 2 \cdot 3^4 + 1 \cdot 3^3 + 2 \cdot 3^2 + 1 \cdot 3 + 1 \sim 21211.$$

Example 1.3.3. *Convert* 100101 *from binary to* 5*-ary. In decimal (or "*10*-ary"),* 100101 *is*

$$1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 32 + 4 + 1 = 37.$$

In 5*-ary,*

$$37 = 1 \cdot 5^2 + 2 \cdot 5^1 + 2 \cdot 5^0 \sim 122.$$

1.3.1 Application: Divisibility criteria

In grade school, you probably learned the rule that an integer is divisible by 3 if and only if the number of it's digits is divisible by 3. There is a similar criteria for divisibility by 9 which sometimes goes by the name "casting out nines".

From the m-ary expansion of an integer, one can deduce other useful divisibility criteria. In this section, we outline a few of the better-known divisibility tests based on the decimal expansion of an integer, and defer the proofs of these tests to section 1.7 (although the reader is invited to prove as many of these tests as she/he can).

Let

$$a = a_k 10^k + ... a_1 10 + a_0,$$

where $0 \leq a_i \leq 9$ are the digits.

By 2 : $2|a$ if and only if $2|a_0$.

By 3 : $3|a$ if and only if $3|(a_0 + a_1 + ... + a_k)$.

By 4 : $4|a$ if and only if $4|(a_0 + a_1 10)$.

By 5 : $5|a$ if and only if $5|a_0$.

By 6 : $6|a$ if and only if $2|a$ and $3|a$.

By 7 : $7|a$ if and only if

$$7|(\quad a_0 + 10a_1 + 100a_2 + a_6 + 10a_7 + 100a_8 + ... \\ -a_3 - 10a_4 - 100a_5 - a_9 - 10a_{10} - 100a_{11} - ...).$$

For example, $7|100100$.

For a proof of this criterion, see Example 1.7.6.

By 8 : $8|a$ if and only if $8|(a_0 + 10a_1 + 100a_2)$.

For a proof of this criterion, see Example 1.7.6.

By 9 : $9|a$ if and only if $9|(a_0 + a_1 + ... + a_k)$.

For example, $9|11115$.

By 10 : $10|a$ if and only if $a_0 = 0$.

By 11 : $11|a$ if and only if $11|(a_0 + a_2 + a_4 + ... - a_1 - a_3 - a_5...)$.

For example, $11|511115$.

In fact, this divisibility rule is the basis for the "ISBN code" (an error-detecting code used internationally in labeling books) whch we shall study later.

By 12 : $12|a$ if and only if $3|a$ and $4|a$.

By 13 : $13|a$ if and only if

$$13|(\quad a_0 + 10a_1 + 100a_2 + a_6 + 10a_7 + 100a_8 + ... \\ -a_3 - 10a_4 - 100a_5 - a_9 - 10a_{10} - 100a_{11} - ...).$$

For example, $13|123123$.

1.3.2 Application: Winning the game of Nim

The game of nim is very simple to play and has an interesting mathematical structure. Nim is a game for two players, who alternate taking turns to play. Initially the players are given a "state" or "position" consisting of several piles of tokens. On each turn a player picks a pile and then removes at least one token from that pile. The player who picks up the last token wins [2]. Assuming your opponent plays optimally, some positions cannot be won. Such positions are called **losing** (for the player whose turn it is to move). The positions which are not losing are called **winning**. (In [BCG], a winning position is defined for the player who just moved.)

The following question has a mathematical answer.

Question: Which positions are losing for the player whose turn it is to move?

The answer is determined by the "nim sum" of the piles. The "nim sum" of r piles with m_1 elements in the 1^{st} pile, m_2 elements in the 2^{nd} pile, ..., m_r elements in the last (r^{th}) pile, is obtained as follows:

1. Write each of these numbers in binary and represent this binary number as a vector of 0's and 1's with 0's filled in on the end to make them all the same length:

   $$v_1 = [a_{10}, a_{11}, ..., a_{1n}], v_2 = [a_{20}, a_{21}, ..., a_{2n}], ..., v_r = [a_{r0}, a_{r1}, ..., a_{rn}],$$

   so that $m_i = a_{i1} + a_{i2}2 + ... + a_{in}2^n$ $(i = 1, ..., r)$,

2. Define the **nim sum vector** of the position vector to be the vector sum $v_1 + v_2 + ... + v_r$ (componentwise addition).

3. Define the **nim sum** $G(m_1, m_2, ..., m_r)$ of the position to be the number whose binary digits are the entries in the nim sum vector.

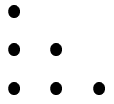
If each of the sums $a_{1j} + a_{2j} + ... + a_{rj}$ (for $j = 0, .., n$) is even then the nim sum is 0 and we call the position **even**. Otherwise, we call the position **odd**. In the book by Ball and Coxeter [BC], one finds the following result.

Theorem 1.3.4. *Odd positions are winning and even positions cannot be won if your opponent plays optimally.*

[2] This is a two-person game in the sense of §2.9 below.

If an initial position is not a losing position, what strategy should one use to win? Make a move which leaves your opponent an even position. In the special case when you only have two piles, a simpler version of this strategy is simply to make the two piles equal by taking tokens from the larger pile.

Example 1.3.5. *Suppose Laura plays Robert and that the initial position is*



which is an even position (the nim sum vector is $[2, 2, 0, 0]$*). Laura has graciously allowed Robert to play first and he removes one token from the first pile. Laura plays by removing the one (and only) token from the third pile. The position is now*

which is again an even position (the nim sum vector is $[0, 2, 0, 0]$*). Robert plays by removing another token from the first pile. Laura plays by removing one token from the second pile. The position is now*

(the nim sum vector is $[2, 0, 0, 0]$*). Robert removes one token from one pile and Laura wins by taking the last remaining token.*

For more detailed answers to these questions, see the references [BC] or [HW].

Exercise 1.3.6. *Show* $a = 14$ *is* 1110 *in binary, is* 112 *in* 3*-ary, is* 24 *in* 5*-ary, and is* 14 *in* 10*-ary.*

Exercise 1.3.7. *(Bachet's measuring problem) An assayer owns a balance scale for ore smaples. He places samples on the left scale, weights on the right scale until balance is achieved. He wants to weight all possible samples of grams 1 to n. What is the smallest number of (integral) weights he will need?*

Exercise 1.3.8. *Use the above rules to answer the following questions. Is 12345678887654321 divisible by 3? 7? 9? 11? 13?*

Exercise 1.3.9. *Use the above rules to answer the following questions. Is 12345678 divisible by 3? 4? 7? 8? 9? 12?*

Exercise 1.3.10. *Use the above rules to answer the following questions. Is 1002003004005006 divisible by 3? 4? 7? 8? 9? 11? 12? 13?*

Exercise 1.3.11. *One quick way to tell if a number was divisible by 7, is as follows. Take the number (expressed in base 10, as are all numbers in the fifth grade), and temporarily put aside the two rightmost digits. Double the number that remains, and add this result to the two-digit number that was just removed. This new result will leave the same remainder as the original number, when divided by 7.*

Example: The remainder when 1234 is divided by 7 is the same as $2 \cdot 12 + 34 = 58$. *Clearly the remainder should therefore be 2.*

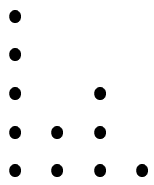
(a) *Express this "divisibility test" in the same form as those above, namely fill in the blank: "*$7|a$ *if and only if* _____*"*

(b) *Prove that this test indeed works.*

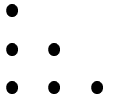
(c) *Invent a related divisibility test for 17. [HINT: Here, consider subtracting rather than adding.]*

Exercise 1.3.12. *Show that if the starting position in a game of nim is* $[2, 2]$, *then the second player can always win.*

Exercise 1.3.13. *Find the nim sum vector for the position*



Exercise 1.3.14. *By enumerating all possible games starting from the position*



show that the second player can always win.

1.4 The Euclidean algorithm

In this section, we will develope a method for computing $gcd(a, b)$. This of practical importance.

Another motivation for this section arises from the following fact. We have seen that all the multiples of a single integer b may be visualized in several ways: as a series of "bumps" spaced b units apart on the real number line, or as the first column of the array of integers when arranged in a series of increasing rows of b numbers each (see (1.1)). The following question asks: How do we generalze this?

Question: Let a and b be any two non-zero integers. Can the set of all possible sums of multiples of a and b be described in a "simple" way? If so, what is it?

1.4.1 Ideals

Think of an ideal as certain types of set of integers which is closed under addition and multiplication. The concept of an ideal will pay big dividends for us in the next chapter.

Definition 1.4.1. *Let $I \subset \mathbb{Z}$ be a subset closed under addition and subtraction, that is to say, for any $r, s \in I$, $r + s \in I$ and $r - s \in I$. Such a subset is called an* **ideal** *of $\mathbb{Z}$.*

Note you can multiply any element of I by any integer and the product will still be in I. You can add two elements (but they *both* must belong to I) and the sum will still be in I.

Thanks to the following result, the answer to the first question asked above is "yes".

Lemma 1.4.2. *Let $I \subset \mathbb{Z}$ be a subset closed under addition and subtraction Then either*

- *I is empty,*
- *$I = \{0\}$,*
- *there is an integer a such that $I = a\mathbb{Z}$, i.e., I is the set of all multiples of some integer a (which is constructed in the proof below).*

The proof below is worthwhile trying to understand well since it contains a basic idea which occurs in other parts of mathematics (in fact, we shall see the idea again in the next chapter).

proof: If I is not empty then it must contain 0. Suppose that I is non-zero. Let $r \in I$ be non-zero. Since $-r \in I$, and either $r > 0$ or $-r > 0$, I must contain a positive element. By the well-ordering principle, I contains a least element $a > 0$.

Claim: $I = a\mathbb{Z}$.

If $b \in I$ then the division algorithm (theorem 1.2.7) says that there is an r such that $0 \leq r < a$ with $b = qa + r$. Since I is closed under addition and subtraction, $r = b - qa$ belongs to I. But a is the smallest non-zero element of I, so r must be zero. This implies $b \in a\mathbb{Z}$. Since b was choosen arbitrarily, this imples $I \subset a\mathbb{Z}$. The reverse inclusion $a\mathbb{Z} \subset I$ follows from the assumption that I is closed under addition and subtraction. This proves the claim and the lemma. □

1.4.2 The Euclidean algorithm

The following result has been known for thousands of years. The method is named for the Greek mathematician Euclid who decribed it in book 7 of this "Elements", written about 2300 years ago. For more details on his life, see the award-winning web site [MacOR].

Theorem 1.4.3. *("Euclidean algorithm") Let $a > 0$ and $b > 0$ be integers. Then*

$$\{ma + nb \mid m, n \in \mathbb{Z}\} = gcd(a, b)\mathbb{Z}.$$

Furthermore, there are integers x, y such that $ax + by = gcd(a, b)$.

proof 1: We first show that there are integers x, y such that $ax + by = gcd(a, b)$.

Let S be the set of all positive integers of the form $sa + tb$, where s and t are integers. Since S is not empty, it has a smallest element by the well-ordering principle. Let $d = xa+yb$ be the smallest element of S. The division algorithm (theorem 1.2.7) tells us that there are integers q and r satisfying $a = dq + r$ and $0 \leq r < d$. But $r = a - dq = (1 - qx)a + (-qy)b$ is in S, contradicting our choice of d as the smallest element of S. Therefore, $r = 0$ and $a = dq$ is divisible by d. Similarly, $d \mid b$, and so it is a common divisor

of a and b. If c is another common divisor of a and b, then $c \mid d = xa + yb$, so $c < d$ and this implies $d = \gcd(a, b)$.

For the proof of the first statement of the theorem, see the proof below. □

proof 2: We can (and do) assume without loss of generality that $0 < b < a$. First we show that there are integers x, y such that $ax+by = gcd(a, b)$. The proof of this is constructive and follows an algorithm called the **Euclidean algorithm**.

Let $r_{-1} = a$ and $r_0 = b$. Let $i = 0$.

(1) Use the division algorithm (theorem 1.2.7) to compute integers q_{i+1} and $0 \leq r_{i+1} < r_i$ such that

$$r_{i-1} = r_i q_{i+1} + r_{i+1}.$$

(2) If $r_{i+1} \neq 0$ then increment i and go to step 1. If $r_{i+1} = 0$ then stop.

Since $0 \leq ... < r_1 < r_0 = b < r_{-1} = a$, at some point the above algorithm must terminate. Suppose that $r_k = 0$ and k is the smallest such integer.

Claim: $gcd(a, b) = r_{k-1}$.

To see that, we first show that $r_{k-1} = ax + by$ for some integers x, y. Indeed, we claim that *every* r_i $(-1 \leq i < k)$ is a integral linear combination of a, b. This may be proven by mathematical induction. (The details are left to the reader as an exercise. The cases $i = -1, 0, 1, 2$ follow from $a = bq_1 + r_1$, $b = r_1 q_2 + r_2$.) Thus $r_{k-1} = ax + by$. The fact $r_{k-1} = ax + by$ implies $gcd(a, b) | r_{k-1}$.

Next, to finish the proof of the above claim, we show that $r_{k-1} | a$ and $r_{k-1} | b$. Indeed, we claim that r_{k-1} divides *every* r_i $(-1 \leq i < k)$ (The details are left to the reader as an exercise. The cases $i = k-1, k-2, k-3$ follow from $r_{k-3} = r_{k-2}q_{k-1} + r_{k-1}$, $r_{k-2} = r_{k-1}q_k$.) The fact $r_{k-1} | a, r_{k-1} | b$ implies $r_{k-1} | gcd(a, b)$.

The claim follows.

It remains to prove

$$\{ma + nb \mid m, n \in \mathbb{Z}\} = gcd(a, b)\mathbb{Z}.$$

The previous paragraphs of this proof show that

$$gcd(a, b)\mathbb{Z} \subset \{ma + nb \mid m, n \in \mathbb{Z}\}.$$

For any integers m, n we note $gcd(a,b)|(ma+nb)$, so

$$\{ma+nb \mid m,n \in \mathbb{Z}\} \subset gcd(a,b)\mathbb{Z}.$$

This implies the equality above, and completes the proof of the theorem. □

1.4.3 Linear diophantine equations

Diophantus, a Greek mathematician who lived during the 4th century A.D., was one of the first people who attempted to find integral or rational solutions to a given system of equations. Often the system involves more unknowns than equations. We will consider a linear equation, $ax+by=c$, with two unknowns x, y.

Theorem 1.4.4. *The linear equation $ax+by=c$ has no solutions if $gcd(a,b)$ does not divide c. If $gcd(a,b)$ does divide c then there are infinitely many solutions given by:*

$$x = x_0 + \frac{b}{gcd(a,b)}\,t, \quad y = y_0 - \frac{a}{gcd(a,b)}\,t,$$

where $x=x_0$, $y=y_0$ is any solution and t is any integer.

proof: The first part of the theorem follows from lemma 1.2.4. Let $x=x_0$, $y=y_0$ be any solution, and let $x=r$, $y=s$ be any other solution. We want to show that $r = x_0 + \frac{b}{d}t$ and $s = y_0 - \frac{a}{d}\,t$, where $d = gcd(a,b)$. Substitute into the equation: $ax+by=c$:

$$0 = c-c = ar+bs-(ax_0+by_0) = a(r-x_0)+b(s-y_0).$$

Therefore, $a(r-x_0) = b(y_0-s)$. If $d>1$ we can divide both sides of this equation by d:

$$\frac{a}{d}(r-x_0) = \frac{b}{d}(y_0-s).$$

Since $(\frac{a}{d}, \frac{b}{d}) = 1$ (see the Exercise 1.2.23), it follows that $\frac{a}{d} \mid (y_0-s)$. Substituting $y_0 - s = \frac{a}{d}\,t$ into the above equation gives:

$$r - x_0 = \frac{d}{a}\,\frac{b}{d}\,(y_0-s) = \frac{b}{a}\,t\,\frac{a}{d} = \frac{b}{d}t,$$

and our proof is complete. □

Corollary 1.4.5. *If $a \mid bc$ then $a \mid gcd(a,b) \cdot c$.*

proof: Assume $a \mid bc$. Let $d = gcd(a,b)$. By the above theorem, there exist integers x, y such that $ax + by = d, soacx + bcy = dc$. Since a divides acx and a divides bcy, by the hypthothesis, it divides $acx + bcy$, by Lemma 1.2.4. Therefore $a \mid (a,b)c$. $\square$

The following corollary is an immediadiate consequence of the previous one.

Corollary 1.4.6. *If $a \mid bc$ and $gcd(a,b) = 1$ then $a \mid c$.*

1.4.4 The extended Euclidean algorithm

Extended Euclidean Algorithm for $d = \gcd(a,b) = xa + yb$ for $0 < b < a$.

(a) let $\overline{u} = \langle a, 1, 0 \rangle$

(b) let $\overline{v} = \langle b, 0, 1 \rangle$

(c) Repeat:

- let $\overline{w} = \overline{u} - \lfloor u_1/v_1 \rfloor \overline{v}$,
- let $\overline{u} \leftarrow \overline{v}$,
- let $\overline{v} = \overline{w}$,

while $v_1 \neq 0$

(d) let $d = u_1$, $x = u_2$, $y = u_3$

(e) return d, x, y.

Example 1.4.7. *Let $a = 331$, $b = 81$. We have*

$$u = [331, 1, 0], v = [81, 0, 1], w = [7, 1, -4],$$

since $[331/81] = 4$. *We have*

$$u = [81, 0, 1], v = [7, 1, -4], w = [4, -11, 45],$$

since $[81/7] = 11$. *We have*

$$u = [7, 1, -4], v = [4, -11, 45], w = [3, 12, -49].$$

We have

$$u = [4, -11, 45], v = [3, 12, -49], w = [1, -23, 94].$$

We have

$$u = [3, 12, -49], v = [1, -23, 94], w = [0, 81, -331].$$

We have

$$u = [1, -23, 94], v = [0, 81, -331],$$

and we must now stop since $v_1 = 0$. *We have* $gcd(331, 81) = 1$ *and* $331 \cdot (-23) + 81 \cdot 94 = 1$.

Example 1.4.8. *Lissa needed extra cash to pay for her college books and decided to sell some of her CD's. She divided them into two groups: she would charge \$29.00 for those in the A-group and only \$18.00 for those in the B-group. She collected a total of \$289.00. How many CD's in each group did she sell?*

Solution: The linear equation to be solved is: $29x + 18y = 289$, *where* x *is the number of CD's in the A-group and* y *is the number of CD's in the B-group. Since* $(29, 18) = 1$, *Theorem 1.4.4 guarantees us a solution.*

Let $u = [29, 1, 0]$, $v = [18, 0, 1]$, $w = u - [\frac{29}{18}]v = [11, 1, -1]$. *We proceed as above to obtain:*

$$u = [18, 0, 1], \quad v = [11, 1, -1], \quad , w = [7, -1, 2],$$

$$u = [11, -1, 1], \quad v = [7, -1, 2], \quad w = [4, 2, -3],$$

$$u = [7, -1, 2], \quad v = [4, 2, -3], \quad w = [3, -3, 5],$$

$$u = [4, 2, -3], \quad v = [3, -3, 5], \quad w = [1, 5, -8],$$

$$u = [3, -3, 5], \quad v = [1, 5, -8], \quad w = [0, -18, -13],$$

$$u = [1, 5, -8], \quad v = [0, -18, 29]$$

We stop here since $v_1 = 0$, *and note that this calculation has given us* $gcd(29, 18) = 1$, $x = 5$, $y = -8$. *Therefore,* $29(5) + 18(-8) = 1$ *and* $29(5 \cdot 289) + 18(-8 \cdot 289) = 289$. *That is, a particular solution to the linear*

equation $29x + 18y = 289$ *is* $x_0 = 5 \cdot 289 = 1445$ *and* $y_0 = -8 \cdot 289 = -2312$. *By Theorem 1.4.4, all solutions are given by:*

$$x = 1445 + 18t, \quad y = -2312 - 29t,$$

where t *is any integer. To solve the above problem, we must find an integer* t *which gives nonnegative values for* x *and* y. *In other words, we wish to find* t *such that* $x = 1445 + 18t \geq 0$, $y = -2312 - 29t \geq 0$. *It turns out that the only value of* t *which satisfies these inequalities is* $t = -80$, *which makes* $x = 5$ *and* $y = 8$.

1.4.5 Euler's phi function

Euler's totient- or ϕ-function is needed for the description of the RSA cryptosystem given later.

Definition 1.4.9. *The number of integers* $1 \leq a \leq b$ *which are relatively prime to* b *is denoted* $\phi(b)$, *called the* **Euler ϕ-function** *or the* **Euler totient function**.

Example 1.4.10. *(1)* $gcd(3, 5) = 1$, $gcd(7, 19) = 1$, *and more generally, any two distinct primes are always relatively prime.*

(2) If p *is any prime then each of the integers* $1, 2, ..., p - 1$ *is relatively prime to* p. *Therefore,*

$$\phi(p) = p - 1,$$

where p *is any prime number.*

(3) We have the following table of values of $\phi(n)$ *for small* n*:*

n	1	2	3	4	5	6	7	8	9	10
$\phi(n)$	1	1	2	2	4	2	6	4	6	4

Corollary 1.4.11. *Let* $a > 0$ *and* $b > 0$ *be integers.* a *is relatively prime to* b *if and only if there are integers* x, y *such that* $ax + by = 1$.

proof: (only if) This is a special case of the Euclidean algorithm (Theorem 1.4.3).

(if) If $d|a$ and $d|b$ then $d|(ax + by)$, so $d|1$. This forces $d = \pm 1$, so $gcd(a, b) = 1$. □

The fastest way to compute $\phi(n)$ is to use the formula $\phi(n) = \phi(a)\phi(b)$, assuming one can factor $n = ab$ into *relatively prime* factors $a > 0$ and $b > 0$. The proof of this fact and more details on the Euler ϕ-function will be given later in §1.6.

Exercise 1.4.12. *Find integers x, y, and $d = gcd(a, b)$ for which $ax + by = gcd(a, b)$, where*
(a) $a = 155$ and $b = 15$,
(b) $a = 105$ and $b = 100$,
(c) $a = 15$ and $b = 51$.

Exercise 1.4.13. *(a) Let*

$$I = \{4m + 10n \mid m, n \in \mathbb{Z}\}.$$

(b) Let

$$I = \{9m + 10n \mid m, n \in \mathbb{Z}\}.$$

(c) Let

$$I = \{15m + 51n \mid m, n \in \mathbb{Z}\}.$$

Use Theorem 1.4.3 to find an $a \in \mathbb{Z}$ such that $I = a\mathbb{Z}$.

Exercise 1.4.14. *Lisa paid \$ 133.98 for some \$ 1.29 pens and \$ 2.49 notebooks. She didn't buy 100 pens and 2 notebooks, even though this adds up to the right amount. How many of each did she buy?*

Exercise 1.4.15. *Laurel owes Hardy \$10. Neither has any cash but Laurel has 14 DVD players which he values at \$ 185 each. He suggests paying his debt with DVD players, Hardy making change by giving Laurel some CD players at \$ 110 each. Is this possible? If so, how?*

Exercise 1.4.16. *Laurel still owes Hardy \$10. Hardy says his CD players are worth \$ 111 dollars. Is the deal still possible? If so, how?*

Exercise 1.4.17. *Laurel still owes Hardy \$10. Hardy says his CD players are worth \$ 111 dollars. Laurel says he will value his DVD players at \$ 184 if Hardy will take \$ 110 for his CD players. Is the deal still possible? If so, how?*

Exercise 1.4.18. *Compute $\phi(25)$, $\phi(50)$, $\phi(100)$.*

Exercise 1.4.19. *(a) Write down the* 10 *elements of the set*

$$I = \{4m + 6n \mid m, n \in \mathbb{Z}\}$$

closest to (and including) 0.

(b) Show that the set I is closed under addition and multiplication [3].

(c) Use part (a) to find an $m \in \mathbb{Z}$ such that $I = m\mathbb{Z}$. (You know such an m exists by the above Lemma 1.4.2.)

Exercise 1.4.20. *(a) Let $I = 3\mathbb{Z} = \{3n \mid n \in \mathbb{Z}\}$.*

(b) Let $I = 5\mathbb{Z} = \{5n \mid n \in \mathbb{Z}\}$.

Show that I is an ideal.

Exercise 1.4.21. *Prove or disprove: For $n > 1$, n is a prime if and only if $\phi(n) = n - 1$.*

Exercise 1.4.22. *Show that $8x + 20y = 30$ has no integer solutions.*

Exercise 1.4.23. *Find all the integers x and y which satisfy both $2x + y = 4$ and $3x + 7y = 5$.*

Exercise 1.4.24. *Suppose that you have a collection of 144 coins made up of dimes amd pennies whose total worth is \$ 11.25. How many of each coin must you have?*

Exercise 1.4.25. *Show that $8x + 20y = 44$ has no positive integer solutions.*

1.5 Primes

Recall an integer p is prime if $p > 1$ and the only positive integers dividing p are 1 and p itself. The first few primes are

$$2, 3, 5, 7, 11, 13, 17, 19, \ldots .$$

The primes form "building blocks" for the integers in some sense (made more precise by the Fundamental Theorem of Arithmetic and by Goldbach's conjecture). We will later see how primes occur in the encryption of information passed over the internet.

It has been known since the times of the Greeks that there are infinitely many primes. The following result is one of the oldest and best known results in mathematics!

[3] This means that (i) if $a, b \in I$ then $a + b \in I$ and (ii) if $a, b \in I$ then $a \cdot b \in I$.

Theorem 1.5.1. *(Euclid's Second Theorem) There are infinitely many primes.*

proof: If $p_1, p_2, ..., p_k$ denote a sequence of primes then $n = p_1 p_2 ... p_k + 1$ is not divisible by any of these primes. But every integer is divisible by some prime, so there is a prime not included in this set. Therefore, the set $\{p_1, ..., p_k\}$ cannot be all the primes. If there is no finite list of primes then they must be infinite in number! □

In spite of their basic nature and importance, many questions about primes remain unknown.

Question: Given a "random" integer n is there a "fast" method of determining if n is a prime or not?

The answer depends on your definition of "fast". Basically, the answer is that there are several primality testing methods known and some work faster for certain types of integers (like those of the form $2^n \pm 1$) and for other types. We will return to this topic in §1.5.3 below.

In two letters to Euler dated 1742, Goldbach conjectured the following result.

Conjecture 1.5.2. *(Goldbach conjectures)*

(a) If $n > 5$ is any odd integer then n is the sum of three primes (i.e., there exist three prime numbers p_1, p_2, p_3 such that $n = p_1 + p_2 + p_3$).

(b) If $n > 2$ is any even integer then n is the sum of two primes (i.e., there exist two prime numbers p_1, p_2 such that $n = p_1 + p_2$).

Conjecture (a) was essentially solved by I. M. Vinogradov in 1937 [Va] but Conjecture (b), after over 250 years, is still unsolved.

Suppose the number of primes up to N is denoted by $\pi(N)$. By Theorem 1.5.1, we know that as $N \to \infty$, $\pi(N) \to \infty$. No *simple* exact formula for $\pi(N)$ is known. It was guessed in the 1800's that $\pi(N)$ is about $\frac{N}{\ln(N)}$.

More precisely, we have the following result.

Theorem 1.5.3. *(prime number theorem) For any choosen $\epsilon > 0$ there is an $N_0(\epsilon) > 1$ (which may be extremely large) such that, $(1 - \epsilon)\frac{N}{\ln(N)} < \pi(N) < (1 + \epsilon)\frac{N}{\ln(N)}$, for all $N > N_0(\epsilon)$.*

The proof of this theorem goes beyond the scope of this book (see for example, Hardy and Wright [HW]). Unfortunately, no one knows for sure what a good estimate for what $N_0(\epsilon)$ is! One can re-interpret the result as stating that, for x large, the probability an integer near x is prime is about $\frac{1}{\ln(x)}$.

In spite of the fact that there are infinitely many primes, the general problem of showing that there are infinitely many primes of a *certain type* is very hard. We give a couple of examples of this idea.

Conjecture 1.5.4. *(twin primes conjecture) There are infinitely many primes p for which p and $p+2$ are prime.*

Examples include 11 and 13, 17 and 19, and many others.

Conjecture 1.5.5. *There are infinitely many primes p of the form n^2+1, for some integer n.*

Examples include 5, 17 and many others.

Conjecture 1.5.6. *(Mersenne prime conjecture) There are infinitely many primes p of the form 2^n-1, for some integer n.*

Examples include 7, 31 and some others.

1.5.1 Pascal's triangle revisited

Recall $\left(\begin{array}{c} n \\ k \end{array} \right) = \frac{n!}{k!(n-k)!}$ denotes the combinatorial symbol "n choose k".

If p is any prime then $p | \left(\begin{array}{c} p \\ k \end{array} \right)$, for $k = 1, 2, ..., p-1$. (We saw cases of this in Example 1.2.36 above.) This is because p divides $p!$ but p does not divide $k!$ nor $(p-k)!$ $(k < p)$.

Let p^{b_k} be the largest power of p that divides $\left(\begin{array}{c} p \\ k \end{array} \right)$, where $b_k = b_k(p) \geq 0$ is an integer. The above reasoning tells us that $b_1 = 0$, $b_p = 0$ and $b_1 > 0$ for $0 < k < p$. The exact value of b_k has been known for over 150 years. In 1855, Kummer discovered that b_k is equal to the number of "carries" required when adding k and $p-k$ in base p. For a discussion of this and many other remarkable facts about binomial coefficients, see A. Granville's excellent paper [G].

1.5.2 The Fundamental Theorem of Arithmetic

We shall prove in this section the following result.

Theorem 1.5.7. *(Fundamental Theorem of Arithmetic) If $n > 1$ is an integer then*

$$n = p_1^{e_1} p_2^{e_2} ... p_m^{e_m},$$

where $p_1 < p_2 ... < p_m$ are primes and $e_1 > 0$, $e_2 > 0$, ..., $e_m > 0$ are integers. Furthermore, this representation of n is unique.

Before proving this result, we require a preliminary fact about primes.

Proposition 1.5.8. *(Euclid's First Theorem) An integer $p > 1$ is prime if and only if it satisfies the following property: for all integers a, b, $p|ab$ implies $p|a$ or $p|b$.*

proof of Euclid's First Theorem: (only if) We may assume that a, b are non-zero. In this case, we have either $gcd(a, p) = 1$ or $gcd(a, p) = p$ since p is a prime. In the latter case, $p|a$ and we are done. By Proposition 1.2.16(3), $p|gcd(a, p)b$, so if $gcd(a, p) = 1$ then $p|b$.

(if) Assume for all integers a, b, $p|ab$ implies $p|a$ or $p|b$. If p was composite then $p = ab$, for some $a < p$ and $b < p$. But $p|p = ab$ would then imply $p|a$ or $p|b$, both of which are impossible. □

In fact, rather than using Euclid's First Theorem directly in the proof of the Fundamental Theorem of Arithmetic, we use the following consequence of it.

Corollary 1.5.9. *(of Euclid's First Theorem) Let p be a prime. For any integers $a_1, a_2, ..., a_k$, $p|a_1...a_k$ implies $p|a_1$ or $p|a_2$ or ... or $p|a_k$.*

We also need the following basic result.

Lemma 1.5.10. *If $m > 1$ is an integer then the smallest $d > 1$ which divides m must be a prime number.*

proof of lemma: Let $d > 1$ be the smallest positive divisor of m. Suppose d has a factor b with $1 < b \leq d$ and $d = bc$. But $b|d$ and $d|m$ implies $b|m$. Since d is the smallest factor of m which is greater than 1, we must have $b = d$, so d is prime. □

proof of Fundamental Theorem of Arithmetic: First, we show that n is a product of primes.

If n is a prime then we are done, so we may assume n is not a prime.

Let $n_0 = n$ and let $i = 0$.

step 1: Let $d_i > 1$ denote the smallest divisor of n_i. (By the above lemma, d_i is a prime.) Let $n_{i+1} = n_i/d_i$. (Note: $n_{i+1} < n_i$.)

step 2: If n_{i+1} is prime then stop. If n_{i+1} is not a prime then replace i by $i+1$ and go to step 1.

Since $n_0 > n_1 > ...$, this process must eventually terminate. Say n_k is a prime. Then

$$n = n_0 = n_k d_{k-1} ... d_0,$$

so n is a product of primes.

By arranging the primes in increasing order and grouping identical primes in the product together, we may write n as in the statement of the theorem.

Now we prove uniqueness. Suppose

$$n = p_1^{e_1} p_2^{e_2} ... p_m^{e_m} = p_1^{f_1} p_2^{f_2} ... p_m^{f_m},$$

where the p_i's are distinct primes but *not* necessarily in any particular order and the $e_1, e_2, ..., e_m, f_1, f_2, ..., f_\ell$ are *non-negative* integers. (By allowing the exponents to be 0, we can insert "dummy prime factors" so that it looks as though both sides run over the same set of primes.)

We may suppose that $e_1 > 0$ (some exponent must be greater than 0 since $n > 1$). By the corollary to Euclid's First Theorem, we must have $p_1 | p_1^{f_1}$ or $p_1 | p_2^{f_2}$ or ... $p_1 | p_m^{f_m}$. Since the p_i's are distinct, the only possibility is if $p_1 | p_1^{f_1}$, so in particular $f_1 > 0$. If $f_1 > e_1$ then $p_2^{e_2} ... p_m^{e_m} = p_1^{f_1 - e_1} p_2^{f_2} ... p_m^{f_m}$, and the corollary to Euclid's First Theorem implies $p_1 | p_2^{e_2}$ or ... $p_1 | p_m^{e_m}$. This is impossible since the p_i's are distinct. Similarly, $e_1 > f_1$ leads to a contradiction. This leaves the only possibility $e_1 = f_2$, so

$$p_2^{e_2} ... p_m^{e_m} = p_2^{f_2} ... p_m^{f_m}.$$

Proceeding inductively, we find $e_2 = f_2$, ..., $e_m = f_m$. This proves uniqueness and the theorem. □

Many important results in number theory are proved using prime factorization. The following theorem is an immediate consequence of the Fundamental Theorem of Arithemetic. The proof is left as an exercise.

Theorem 1.5.11. *Let $a = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ and $b = p_1^{f_1} p_2^{f_2} \cdots p_k^{f_k}$ be any positive integers with the given prime factorizations ($e_i \geq 0$, $f_i \geq 0$, for all i, $1 \leq i \leq k$). Then the following is true:*

- *$a \mid b$ if and only if $e_i \leq f_i$, for all i,*

- $gcd(a,b) = p_1^{m_1} p_2^{m_2} \cdots p_k^{m_k}$, *where* $m_i = \min(e_i, f_i)$,
- $lcm(a,b) = p_1^{M_1} p_2^{M_2} \cdots p_k^{M_k}$, *where* $M_i = \max(e_i, f_i)$.

1.5.3 Primality testing

In this section, we examine some simple general methods for determining if a number is prime. There are much more efficient (and complicated) primality tests what those which we discuss here [4] - see for example [BS].

The first prime to be discovered having over 1 million digits was the Mersenne prime, $2^{6972593} - 1$. The primality of this integer was determined by Nayan Hajratwala in 1993, who received \$ 50000 from the Electronic Frontier Foundation as a reward for it's discovery. The first person who discovers a prime having over 1 billion digits may win \$ 250000.

Sieve of Eratosthenes

The key fact that is used for the method discussed in this section is the fact that if n is any composite than it must have a prime factor which is less than or equal to $\sqrt{n}$, by Lemma 1.2.5.

The method to produce all primes up to $N > 2$:

1. List all integers $2, ..., N$.
2. Let $a = 2$.
3. Cross out all multiples of a except for a itself.
4. If all integers between a and N are crossed out then stop. Otherwise, replace a by then next largest integer which has not been crossed out. If this new a is greater than $\sqrt{N}$ then stop.
5. Go to step 3.

This process must terminate after at most $\sqrt{N}$ steps.

[4] In fact, a "polynomial time" primality test has recently (August 2002) been announced by M. Agrawal, and two of his students, N. Kayal and N. Saxena, in the Dept. of Computer Science and Engineering at the Indian Inst.of Tech. in Kanpur. This is a very important result. However, at the time of this writing, it has not yet been published.

Example 1.5.12. *Let* $N = 20$. *step 1:*

$$2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20$$

step 2: Cross out multiples of 2*:*

$$2, 3, \not{4}, 5, \not{6}, 7, \not{8}, 9, \not{10}, 11, \not{12}, 13, \not{14}, 15, \not{16}, 17, \not{18}, 19, \not{20}$$

step 3: Cross out multiples of 3*:*

$$2, 3, \not{4}, 5, \not{6}, 7, \not{8}, \not{9}, \not{10}, 11, \not{12}, 13, \not{14}, \not{15}, \not{16}, 17, \not{18}, 19, \not{20}$$

All the remaining numbers are prime.

Exercise 1.5.13. *Using the Sieve of Eratosthenes, find all the primes from* 1 *to* 50.

Exercise 1.5.14. *How many digits does* $2^{6972593} - 1$ *have? (Hint: What is* $\log_{10}(2^{6972593})$*?)*

Exercise 1.5.15. *Check that* $n = 6$ *and* $n = 28$ *are perfect.*

Exercise 1.5.16. *Determine the prime decomposition of (a)* 111*, (b)* 1111*, (c)* 1234.

Exercise 1.5.17. *Show that if* $2^n - 1$ *is a prime, for some integer* n*, then* n *is also a prime.*

Exercise 1.5.18. *In the notation of §1.5.1, compute* $b_k(p)$ *for*
(a) $p = 7$, $1 \leq k \leq p$,
(b) $p = 11$, $1 \leq k \leq p$,
(c) $p = 13$, $1 \leq k \leq p$.

Exercise 1.5.19. *(a) Assume some encryption scheme requires a* 100 *digit prime. It is unlikely you will find a such a prime on your first guess. Approximately how many* 100 *digit integers would have to be randomly picked before a prime is found?*
(b) Estimate how many 100-digit prime numbers there are.

Exercise 1.5.20. *Assume z is a real number greater than 1. Let*

$$\zeta(z) = \sum_{n=1}^{\infty} n^{-z} = 1 + 2^{-z} + 3^{-z} + \dots ,$$

denote the Riemann zeta function. Here the sum runs over all integers $n \geq 1$. Let

$$P(z) = \prod_{p \text{ prime}} (1 - p^{-z})^{-1} = (1 - 2^{z})^{-1}(1 - 3^{-z})^{-1}(1 - 5^{-z})^{-1} \dots ,$$

denote the Euler product. *Here the product runs over all prime numbers $p \geq 2$. Without worrying about convergence issues (using calculus, one can show that these series considered here all converge absolutely), show that $\zeta(z) = P(z)$. In other words, show*

$$1+2^{-z}+3^{-z}+4^{-z}+5^{-z}+\dots = (1-2^{z})^{-1}(1-3^{-z})^{-1}(1-5^{-z})^{-1}(1-7^{-z})^{-1}\dots .$$

(Hint: Recall $1/(1-x) = 1 + x + x^2 + x^3 + \dots$ and use the Fundamental Theorem of Arithmetic.)

Exercise 1.5.21. *Using the fundamental theorem of arithmetic, prove (1), (2), (3), and (4) of Proposition 1.2.16.*

Exercise 1.5.22. *Show that if $2^p - 1$ is a prime then p must also be a prime. (Hint: $2^{ab} - 1 = (2^a)^b - 1$ is divisible by $2^a - 1$.)*

1.6 Multiplicative Functions

Recall integers a and b are relatively prime if $gcd(a, b) = 1$.

Definition 1.6.1. *Let $f : \mathbb{N} \to \mathbb{C}$ be a function. The function f is called* **multiplicative** *if $f(ab) = f(a)f(b)$, whenever a and b are relatively prime. If $f(ab) = f(a)f(b)$, for all $a, b \in \mathbb{N}$ then we call f* **completely multilpicative**.

In this section, we will introduce three important multiplicative functions. Note these functions are determined by their values on the prime powers.

You may have guessed the strategy for finding a formula for multiplicative functions. Since $gcd(p, q) = 1$, for any prime numbers p and q, in order to

find a formula for $f(n)$, where n is a positive integer and f is a multiplicative function, we factor n as a product of prime powers and our work is reduced to finding a formula for $f(p^e)$, where p^e is the highest power of the prime p dividing n.

Definition 1.6.2. *Let n be a positive integer. The number of positive divisors of n is denoted $d(n)$ and is called the* **number of divisors function**. *The sum of the positve divisors of n is denoted $\sigma(n)$ and is called the* **sum of divisors function**. *In other words,*

$$d(n) = \sum_{d|n} 1,$$

and

$$\sigma(n) = \sum_{d|n} d.$$

Example 1.6.3. *(a) $d(p) = 2$ and $\sigma(p) = 1 + p$, where p is any prime number.*

(b) $d(24) = 8$,

(c) $\sigma(24) = 60$.

Example 1.6.4. *We have $d(p^k) = k+1$, since the positive divisors of p^k are $1, p, p^2, p^3, \ldots, p^k$. We have*

$$\sigma(p^k) = 1 + p + p^2 + p^3 + \cdots + p^k = \frac{1 - p^{k+1}}{1 - p}.$$

Theorem 1.6.5. *Let n be any positive integer with prime factorization: $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$. Then*

$$d(n) = d(p_1^{e_1})\, d(p_2^{e_2}) \cdots d(p_k^{e_k}) = (e_1 + 1)(e_2 + 1) \cdots (e_k + 1).$$

proof: By Theorem 1.5.11 (1) all divisors of n are of the form: $a = p_1^{f_1} p_2^{f_2} \cdots p_k^{f_k}$, where $0 \leq f_i \leq e_i$. Therefore, there are $e_1 + 1$ choices for f_1, $e_2 + 1$ choices for f_2, etc, which gives us the above formula. □

Corollary 1.6.6. *The number of divisors function d is multiplicative.*

proof: Let m and n be relatively prime positive integers. We write the prime factorizations as follows:

$$m = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k} \quad \text{and} \quad n = q_1^{f_1} q_2^{f_2} \cdots q_l^{f_l},$$

where all the p_i and q_j are distinct primes.

We have $d(mn) = d(p_1^{e_1} \cdots p_k^{e_k} q_1^{f_1} \cdots q_l^{f_l})$, which, by Theorem 1.6.5, is equal to: $(e_1 + 1) \cdots (e_k + 1)(f_1 + 1) \cdots (f_l + 1) = d(m)\, d(n)$. □

Theorem 1.6.7. *The function $\sigma(n)$ is multiplicative.*

proof: Let m and n be relatively prime positive integers with positive divisors $a_1, a_2, \ldots, a_k$ and $b_1, b_2, \ldots, b_l$, respectively. Then:

$$\sigma(m)\,\sigma(n) = (a_1+a_2+\cdots+a_k)\,(b_1+b_2+\cdots+b_l) = a_1(b_1+\cdots+b_l)+\cdots+a_k(b_1+\cdots+b_l).$$

Since $gcd(m, n) = 1$, each term is distinct, i.e. $a_i\, b_j \neq a_s\, b_t$, if $i \neq s$ and $j \neq t$ (why?). Therefore, the numbers in the above sum represent all of the divisors of mn and we have proved that $\sigma(mn) = \sigma(m)\,\sigma(n)$. □

Let n be a positive integer. Recall the Euler phi function, $\phi(n)$, is the number of positive integers less than or equal to n which are relatively prime to n.

Lemma 1.6.8. *$\phi(p^n) = p^{n-1}(p-1)$ for all positive integers n.*

proof: The positive integers less than or equal to p^n, which are not relatively prime to p^n are all multiples of p: $1 \cdot p,\ 2 \cdot p,\ 3 \cdot p,\ \ldots, p^{n-1} \cdot p$. Thus, we can see that there are p^{n-1} such integers. Since there are p^n integers less than or equal to p^n, it follows that:

$$\phi(p^n) = p^n - p^{n-1} = p^{n-1}(p-1).$$

□

Our next goal is to show that ϕ is multiplicative. This result, the Fundamental Theorem of Artithmetic, and Lemma 1.6.8 will give us a formula for $\phi(n)$, for any positive integer n.

We start with a lemma. Recall that for integers a, b and a fixed integer $m > 1$, we define $a \equiv b \pmod{m}$ if and only if $m \mid (a - b)$.

Lemma 1.6.9. *If $gcd(a, m) = 1$ and $a \equiv b \pmod{m}$ then $gcd(b, m) = 1$*

proof: Write $b = a + c\,m$, for some integer c. The rest of the proof is left as an exercise. □

Theorem 1.6.10. *ϕ is multiplicative.*

proof: Let m and n be positive integers with $gcd(m,n) = 1$. Write the numbers 1 to mn as an array:

$$\begin{array}{ccccc} 1 & m+1 & 2m+1 & \dots & (n-1)m+1 \\ 2 & m+2 & 2m+2 & \dots & (n-1)m+2 \\ . & . & . & & . \\ . & . & . & & . \\ m & 2m & 3m & \dots & nm \end{array}$$

If $gcd(k,m) = d > 1$ then no element in the row which has k as the first element is relatively prime to mn. That is, all numbers which are relatively prime to mn must lie in a row whose first element is relatively prime to m. Since there are $\phi(m)$ such rows, our proof will be complete if we can show that there are $\phi(n)$ elements in each such row which are relatively prime to mn.

Consider the k^{th} row: $k \quad m+k \quad 2m+k \quad \dots \quad (n-1)m+k$, where $gcd(k,m) = 1$. Suppose $im + k \equiv jm + k \pmod{n}$, with $0 \le i, j < n$. Then n divides $im + k - (jm + k) = (i-j)m$. But, since $gcd(m,n) = 1$, it follows from Proposition 1.2.16(2) that n divides $i - j$, i.e. $i \equiv j \pmod{n}$. This, together with the above inequalities involving i and j show that $i = j$. Therefore, no two elements in the k^{th} row are congruent $(\mathrm{mod}\, n)$. By Lemma 1.6.9, every element in the k^{th} row is relatively prime to m. Since no two elements are congruent, their least residues are $0, 1, 2, \dots, n-1$, in some order. By definition, exactly $\phi(n)$ elements of $\{0, 1, ..., n-1\}$ are relatively prime to n. It then follows that there are exactly $\phi(n)$ elements in the k^{th} row which are relatively prime to mn which completes the proof. □

Theorem 1.6.11. *Let n be a positive integer with prime factorization: $n = p_1^{e_1}\, p_2^{e_2} \cdots p_k^{e_k}$. Then $\phi(n) = p^{e_1-1}\,(p_1-1)\,p_2^{e_2-1}\,(p_2-1) \cdots p_k^{e_k-1}\,(p_k-1)$.*

proof: Since $gcd(p_i^{e_i}, p_j^{e_j}) = 1$ for all $i \neq j$, the result follows from Lemma 1.6.8 and Theorem 1.6.10. □

Example 1.6.12. *Show that: $\phi(n) + \sigma(n) = n\,d(n)$ if and only if n is prime.*

Solution: We know that $\phi(n) \leq n-1$ *and* $\phi(n) = n-1$ *if and only if* n *is prime. In general,* $\sigma(n) = 1 + r_1 + r_2 + \cdots + r_k$, *where* $r_k = n$ *and* $r_i \mid n, \quad 1 < r_i < n$, *for* $1 \leq i \leq k-1$, $k = d(n) - 1$. *Therefore,*

$$\sigma(n) \leq 1 + kn,$$

$$\phi(n) + \sigma(n) \leq (n-1) + 1 + kn,$$

$$\phi(n) + \sigma(n) \leq n + (d(n) - 1)n = nd(n).$$

If n *is prime, then* $\phi(n) = n-1$, $d(n) = 2$, *and* $\sigma(n) = n+1$, *so:*

$$\phi(n) + \sigma(n) = (n-1) + (n+1) = 2\,n = d(n)\,n.$$

1.6.1 Perfect numbers and Mersenne primes

It is remarkable that even at this "elementary" level there are many problems which are still unsolved. In this section, we mention one of the oldest unsolved problems in mathematics.

Let $n > 0$ be an integer and let

$$\sigma(n) = \sum_{d|n} d.$$

For example, $\sigma(12) = 1 + 2 + 3 + 4 + 6 + 12 = 28$.

Definition 1.6.13. *A* **perfect number** *is an integer* $n > 1$ *such that* $\sigma(n) = 2n$, *in other words,* n *is the sum of its proper divisors.*

No odd perfect numbers are known. The following conjuecture may be the oldest unsolved problem in mathematics!

Conjecture 1.6.14. *Odd perfect numbers don't exist.*

It is known if an odd perfect number exists then it must be at least 10^{300}.

Lemma 1.6.15. *An integer* n *is an even perfect number if and only if* $n = 2^{p-1}(2^p - 1)$, *where* $2^p - 1$ *is a prime.*

Though this result is often quoted as being due to Euler, it may have been known to Euclid.

proof: We leave the "if" direction as an exercise.

"Only if": Since n is an even number, we can write $n = 2^s t$, where t is odd and $s \geq 1$. We know that $\sigma(t) > t$, so let $\sigma(t) = t + r$, with $r > 0$. Since $2\,n = \sigma(n)$, we have:

$$2^{s+1}\,t = (2^{s+1} - 1)\,(t + r) = 2^{s+1}\,t - t + (2^{s+1} - 1)r$$

which gives us:

$$t = (2^{s+1} - 1)\,r.$$

Therefore, r is a divisor of t and $r < t$. But $\sigma(t) = t + r$ implies that r is the sum of all of the divisors of t which are less than t (i.e. r is the sum of a group of numbers which include r). This is possible only if the group consists of one number, and that number must be one. Therefore, $\sigma(t) = t + 1 = 2^{s+1} - 1$, which implies that both $t = 2^{s+1} - 1$ and $s + 1$ are prime. Letting $p = s + 1$ gives the conclusion. □

A prime number of the form $2^p - 1$ is called a **Mersenne prime**. As we've seen already, it is unknown whether or not there are infinitely many Mersenne primes.

For further details on perfect numbers, see for example Ball and Coxeter [BC], page 66, or Hardy and Wright [HW], §16.8.

Exercise 1.6.16. *Find all solutions to* $\phi(n) = 4$.

Exercise 1.6.17. *Show that there are no solutions to* $\phi(n) = 14$.

Exercise 1.6.18. *Show that if* $f(n)$ *is multiplicative then so is* $f(n)/n$.

Exercise 1.6.19. *Show that if* $f(n)$, $g(n)$ *are multiplicative then so is* $f(n)g(n)$.

Exercise 1.6.20. *Find all* n *such that* $\phi(2n) = \phi(n)$.

Exercise 1.6.21. *Show that* $d(n)$ *is odd if and only if* n *is a square.*

Exercise 1.6.22. *Find all* n *such that* $\phi(n)$ *is odd.*

Exercise 1.6.23. *Show that no perfect number is a square.*

Exercise 1.6.24. *Show that if* $gcd(m, n) = 2$ *then* $\phi(mn) = 2\phi(m)\phi(n)$.

1.7 Congruences

Recall that for integers a, b and a fixed integer $m > 1$, we define $a \equiv b \pmod{m}$ if and only if $m|(a - b)$. In this case, we say that a is congruent to b modulo m.

1.7.1 Application: Divisibility criteria revisited

Let

$$a = a_k 10^k + ...a_1 10 + a_0,$$

where $0 \leq a_i \leq 9$ are the digits. The following congruence conditions generalize some of the criteria given in §1.7.1.

mod 2 : $a \equiv a_0 \pmod 2$.

mod 3 : $a \equiv (a_0 + a_1 + ... + a_k) \pmod 3$.

mod 4 : $a \equiv (a_0 + a_1 10) \pmod 4$.

mod 5 : $a \equiv a_0 \pmod 5$.

mod 8 : $a \equiv (a_0 + 10a_1 + 100a_2) \pmod 8$.

mod 9 : $a \equiv (a_0 + a_1 + ... + a_k) \pmod 9$.

For example, $11116 \equiv 1 \pmod 9$.

mod 10 : $a \equiv a_0 \pmod{10}$.

There are also conditions for 7, 11, and 13 but they are left as exercises.

1.7.2 Aside on equivalence relations

A **relation** on a set S is a subset R of $S \times S$. By a slight abuse of notation, let us write, for any $s, t \in S$, sRt if and only if $(s, t) \in R$. An **equivalence relation** is a relation satisfying (for equivalence relations, we write $s \sim t$ instead of sRt)

- for all $s \in S$, $s \sim s$ (reflexive),
- for all $s, t \in S$, $s \sim t$ implies $t \sim s$ (symmetric),
- for all $s, t, u \in S$, if $s \sim t$ and $t \sim u$ then $s \sim u$ (transitivity).

Example 1.7.1.

- *The most common example of an equivalence relation is equality $=$.*
- *Another example is from high school geometry: the similarity or congruence relation for triangles.*

- *If $m > 1$ is a fixed modulus then $\equiv$ modulo m is an equivalence relation. The verification of this is left as an exercise.*

The **equivalence class** of $s \in S$ is

$$[s] = \{t \in S \mid t \sim s\}.$$

If $[s]$ is an equivalence class of S then we call s (or any other element of $[s]$) a **representative** of $[s]$ in S.

1.7.3 Properties of $\equiv \pmod m$

If the equivalence relation is congruence modulo m, $\equiv \pmod m$, then equivalence classes are more often called **residue classes**. The element i of a residue class $[a]$ mod m with $0 \leq i < m$ is called the **remainder** of a mod m. Moreover, a residue class is sometimes denoted by a bar rather than by square brackets:

$$\overline{a} = \{b \in \mathbb{Z} \mid a \equiv b \pmod m\} = a + m\mathbb{Z}.$$

(Of course, $\overline{a}$ depends implicitly on m.) In this case, the total possible number of distinct residue classes is finite (in fact, there are exactly n such classes), $\overline{0}, \overline{1}, ..., \overline{m-1}$.

Example 1.7.2. *Let $m = 100$ and $a = 37$, $b = 63$. Note that $\overline{a} = \overline{37} = 37 + 100\mathbb{Z}$, $\overline{-a} = \overline{-37} = -37 + 100\mathbb{Z} = -37 + 100 + 100\mathbb{Z} = \overline{63} = \overline{b}$. Also, $\overline{37} = \overline{137} = \overline{237} = ...$ and $\overline{37} = \overline{-63} = \overline{-163} = ...$*

Lemma 1.7.3. *Fix an integral modulus $m > 1$ and let a, b, c, d be integers.*

- *If $a \equiv c \pmod m$ and $b \equiv d \pmod m$ then*

$$\begin{aligned} a + b &\equiv c + d \pmod m \\ a - b &\equiv c - d \pmod m \\ ab &\equiv cd \pmod m \\ a^2 &\equiv c^2 \pmod m. \end{aligned}$$

- *If $a + c \equiv b + c \pmod m$ then $a \equiv b \pmod m$.*
- *If $ac \equiv bc \pmod m$ and $gcd(c, m) = 1$ then $a \equiv b \pmod m$. ("cancellation mod m")*

proof: All of these are left as an exercise, except for the last one.

Assume $ac \equiv bc \pmod{m}$ and $gcd(c, m) = 1$. Then $m|(ac-bc) = c(a-b)$. By Proposition 1.2.16(3), $m|(a-b)$. $\square$

Example 1.7.4. *Which values of x satisfy $4x \equiv 12 \pmod{5}$? Solution: Since $gcd(4, 5) = 1$, we can use cancellation mod 5 to reduce the congruence to $x \equiv 3 \pmod{5}$. So the solutions are $x = 3 + 5k$, for any integer k.*

Example 1.7.5. *Which values of x satisfy $16x \equiv 30 \pmod{17}$? Solution: Since $gcd(2, 17) = 1$, we can use cancellation mod 17 to reduce the congruence to $8x \equiv 15 \pmod{17}$. Until we learn how to compute the inverse of 8 mod 17 (see below), we cannot divide by 8. However, the following strategy works just as easily: add multiples of 17 to 15 until we can divide by 2 again. For example, $8x \equiv 15 \equiv 32 \pmod{17}$. As above, by the cancellation law, $x \equiv 4 \pmod{17}$. So the solutions are $x = 4 + 17k$, for any integer k.*

Example 1.7.6. *We shall now verify the divisibilty criteria for the cases $7|a$ and $8|a$ that were stated in §1.3.1, and leave the others as exercises.*

We have $1001 = 7 \cdot 11 \cdot 13$, so $1000 \equiv -1 \pmod{7}$. Substituting, we obtain

$$\begin{aligned} & a_0 + a_1 10 + a_2 100 + a_3 1000 + a_4 10^4 + a_5 10^5 + a_6 10^6 + a_7 10^7 + ... \\ \equiv\ & a_0 + a_1 10 + a_2 100 - a_3 - a_4 10 - a_5 100 - a_6 10^3 - a_7 10^4 - ... \pmod{7} \\ \equiv\ & a_0 + a_1 10 + a_2 100 - a_3 - a_4 10 - a_5 100 + a_6 + a_7 10 + ... \pmod{7} \end{aligned}$$

from which the divisibilty result for 7 follows.

We have $1000 = 8 \cdot 125$, so $1000 \equiv 0 \pmod{8}$. Substituting, we obtain

$$\begin{aligned} & a_0 + a_1 10 + a_2 100 + a_3 1000 + a_4 10^4 + a_5 10^5 + a_6 10^6 + a_7 10^7 + ... \\ \equiv\ & a_0 + a_1 10 + a_2 100 \pmod{8} \end{aligned}$$

from which the divisibilty result for 8 follows.

The following result, which will be used later, is a consequence of the Euclidean algorithm.

Lemma 1.7.7. *Let $a > 0$ and $m > 1$ be integers. a is relatively prime to m if and only if there is an integer x such that $ax \equiv 1 \pmod{m}$.*

The integer x in the above lemma is called the **inverse of** a **modulo** m.

Example 1.7.8. 5 *is the inverse of itself modulo* 6 *since* $5 \cdot 5 = 25 \equiv 1 \pmod 6$.

proof: (only if) Assume $gcd(a, m) = 1$. There are integers x, y such that $ax + my = 1$, by the Euclidean algorithm (more precisely, by Corollary 1.4.11). Thus $m|(ax - 1)$.

(if) Assume $ax \equiv 1 \pmod m$. There is an integer y such that $ax-1 = my$. By Corollary 1.4.11 again, we must have $gcd(a, m) = 1$. □

More generally, we have the following result.

Proposition 1.7.9. *Let* $a > 0, b > 0$ *and* $m > 1$ *be integers.* $gcd(a, m)|b$ *if and only if there is an integer* x *such that* $ax \equiv b \pmod m$.

The result above tells us exactly when we can solve the "modulo m analogs" of the equation $ax = b$ studied in elementary school. The proof (which requires the previous lemma and Proposition 1.2.16) is left as a good exercise.

1.7.4 Repeated squaring algorithm

How hard do you think it would be to compute *by hand* $2^{128} \bmod 5$? If you guess too hard, you're wrong. The algorithm described below shows how such seemingly difficult calculations are actually quite easy. In preparation for computing with Fermat's little theorem and Euler's theorem, both of which will be stated later, we show how to efficiently compute the power of an integer modulo m.

One way to compute a^n modulo m is simply to compte a^n then to reduce modulo m. This is relatively hard to do if n is large since there are a lot of digits to keep track of. We describe an alternative method.

Repeated squaring algorithm

To compute a^n modulo m:

step 1: write n is binary,

$$n = a_k 2^k + ... + a_1 2 + a_0,$$

where each a_i is either 0 or 1.

step 2: Compute

$$a \pmod m,\ a^2 \pmod m,\ ..., a^{2^k} \pmod m,$$

(in that order).

step 3: Compute $a^n = a^{a_0} a^{a_1 2} ... a^{a_k 2^k} \pmod m$.

Example 1.7.10. *We compute* $10^{11} \pmod{21}$. *First,* $11 = 8 + 2 + 1$, *so we compute*

- $10 \pmod{21} = 10$,
- $10^2 \pmod{21} = 16$,
- $10^4 \pmod{21} = 16^2 = 4$,
- $10^8 \pmod{21} = 4^2 = 16$.

Thus $10^{11} \pmod{21} = 4 \cdot 16 \cdot 10 = 19$.

1.7.5 Fermat's little theorem

We have seen that if p is a prime then p divides all the binomial coefficients $\begin{pmatrix} p \\ k \end{pmatrix}$, $1 \leq k \leq p-1$. Because of this and the binomial theorem, we have

$$(a+b)^p \equiv a^p + b^p \pmod{p},$$

for any integers a, b. Using mathematical induction, we have

$$(a_1 + ...a_k)^p \equiv a_1^p + ... + a_k^p \pmod{p},$$

for any integers $a_1, ..., a_k$. In particular, we can take all the a_i's equal to 1. The proves the following result.

Theorem 1.7.11. *(Fermat's little theorem) If* k *is any integer then* $k^p \equiv k \pmod{p}$.

This fact was first discovered by Pierre Fermat [5], though he apparently gave no proof. This theorem was stated without proof by Fermat in a letter dated 1640. Leibnitz proved this in 1683 in an unpublished manuscript and Euler, in 1736, published the first proof.

[5] Fermat (1601-1665) was by profession a lawyer and judge in Toulouse, France. There were no real employment opportunities in mathematics at the time (at least not like there are now). Though perhaps most famous as a number theorist, he also worked on the foundations of calculus, and he is widely regarded as the best French mathematician of that century and one of the greatest number theorists of all time.

Example 1.7.12. *If* $p = 7$ *then* $2^7 \equiv 2 \pmod 7$. *Indeed,* $2^7 = 128$ *and* $7|126$ *since* $126 = 140 - 14 = 18 \cdot 7$.

Exercise 1.7.13. *Use the repeated squaring algorithm or your calculator to directly compute* $3^5 (\text{mod } 5)$, $2^7 (\text{mod } 7)$, $10^1 1 (\text{mod } 11)$. *Check that Fermat's little theorem holds in this case.*

Exercise 1.7.14. *(No computers allowed!)*

(a) Use the division algorithm (Theorem 1.2.7) to find the remainder of 1234567887654320 *modulo* m, *where* $m = 2, 3, 4, 5, 8, 9$.

(b) Use the division algorithm (Theorem 1.2.7) to find the remainder of 1002003004005006 *modulo* m, *where* $m = 2, 3, 4, 5, 8, 9$.

Exercise 1.7.15. *Use the algorithm above to compute*

(a) $10^{11} \pmod 7$,

(b) $2^{10} \pmod 3$,

(c) $5^{13} \pmod 8$.

Exercise 1.7.16. *Verify the divisibilty criteria for* $4|a$, $9|a$ *and for* $13|a$ *in* §*1.3.1.*

Exercise 1.7.17. *Show that if* $\sim$ *is any equivalence relation on* S *then*

(a) $s \in [s]$,

(b) $[s] = [t]$ *if and only if* $s \sim t$,

(c) $[s] \cap [t] = \emptyset$ *if and only if* s *is not equivalent to* t.

Exercise 1.7.18. *Prove Proposition 1.7.9.*

Exercise 1.7.19. *Solve* $3x \equiv 7 \pmod{100}$.

1.7.6 Linear recurrence equations

The general method for solving a recurrence equation of the form

$$s_{k+i} = a_0 s_i + a_1 s_{i+1} + ... + a_{k-1} s_{k+i-1}, \qquad i > 1,$$

with $s_1, s_2, ..., s_k$ given, is rather simple to describe (in principle - in practice it may be quite hard).

First, "guess" $s_n = cr^n$, where c and r are constants. Substituting into the recursion relation and simplifying, we find that c can be arbitrary but r must satisfy

$$a_0 + a_1 r + ... + a_{k-1} r^{k-1} - r^k = 0.$$

Let $r_1, ..., r_k$ be the roots of this polynomial. We shall *assume that these roots are distinct.* Under these conditions, let s_n be an arbitrary linear combination of all your "guesses",

$$s_n = c_1 r_1^n + c_2 r_2^n + ... c_k r_k^n.$$

Recall that $s_1, ..., s_k$ are known, so we have k equations in the k unknown $c_1, ..., c_k$. This completely determines s_n.

Example 1.7.20. *Let s_n satisfy $s_n = s_{n-1} + s_{n-2}$ and let $s_1 = 1, s_2 = 1$.*

We must solve $r^2 - r - 1 = 0$, whose roots are $r_1 = \frac{1+\sqrt{5}}{2}$ and $r_2 = \frac{1-\sqrt{5}}{2}$. Therefore,

$$s_n = c_1(\frac{1+\sqrt{5}}{2})^n + c_2(\frac{1-\sqrt{5}}{2})^n, \qquad n > 0.$$

Since $s_1 = 1$ and $s_2 = 1$, we have

$$s_n = 5^{-1/2} r_1^n - 5^{-1/2} r_2^n.$$

The recurrence equation

$$s_{k+n} = a_0 s_n + a_1 s_{n+1} + ... + a_{k-1} s_{k+n-1}, \qquad n > 1, \tag{1.3}$$

is equivalent to the matrix equation

$$\begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 1 \\ \vdots & & \dots & & \\ 0 & 0 & \dots & 0 & 1 \\ a_0 & a_1 & \dots & & a_{k-1} \end{pmatrix} \begin{pmatrix} s_n \\ s_{n+1} \\ \vdots \\ s_{n+k-1} \end{pmatrix} = \begin{pmatrix} s_{n+1} \\ s_{n+2} \\ \vdots \\ s_{n+k} \end{pmatrix},$$

where s_{n+k} is given as above. Let

$$A = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 1 \\ \vdots & & \dots & & \\ 0 & 0 & \dots & 0 & 1 \\ a_0 & a_1 & \dots & & a_{k-1} \end{pmatrix}, \quad \text{and} \quad \vec{s}_n = \begin{pmatrix} s_n \\ s_{n+1} \\ \vdots \\ s_{n+k-1} \end{pmatrix}.$$

Then the recursive equation is equivalent to $A\vec{s}_n = \vec{s}_{n+1}$. The sequence $s_1, s_2, ...$ is periodic with period N if and only if $A^N \vec{s}_1 = \vec{s}_{N+1}$. The sequence $s_1, s_2, ...$ is eventually periodic with period N if and only if $A^N \vec{s}_r = \vec{s}_{N+r}$, for some $r > 0$.

Example 1.7.21. *Define* $s_{i+2} = s_{i+1} + s_i \pmod{11}$, $s_1 = 1$, $s_2 = 1$. *The above method gives*

$$s_i = (3 \cdot 8^i + 8 \cdot 4^i \pmod{11}.$$

1.7.7 Application: Two ciphers

Let A be a finite set, which we call the **alphabet**, and let M be the set of all finite sequences of elements of A, which we call the **message space**. A **cipher** is a mapping

$$E : M \to E, \tag{1.4}$$

called **encryption**, and an inverse mapping

$$D : M \to M,$$

called a **decryption**, which satisfy $D(E(m)) = m$ for all $m \in M$. The messages in the range of E are called the **message text** and the domain of E is called the **cipher text**.

Caesar's cipher

In this cipher, the alphabet A is the usual alphabet of 26 Roman letters. Punctuation, spaces and capitalizations are ignored. It is reported that Caesar used an encryption $E : M \to M$ based on the substitution A $\longmapsto$ d, ... , Z $\longmapsto$ c, given by the table below.

A	B	C	D	E	F	G	H	I	J	K	L	M
d	e	f	g	h	i	j	k	l	m	n	o	p
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
q	r	s	t	u	v	w	x	y	z	a	b	c

Example 1.7.22. *"Go Tigers" is enciphered as "jr wljhuv".*

If we identify each letter with its corresponding number via the labeling A is 1, B is 2, ..., Z is 26, then Caesar's cipher is a cipher for which any letter Λ is related to its enciphered letter λ by

$$\Lambda \equiv \lambda \pmod{n},$$

where in the example above $n = 3$. More generally, a **substitution cipher** is a cipher for which the alphabet A has been rearranged according to some fixed permutation. The Caesar cipher is a special case of this where the permutation is as indicated above.

A stream cipher: linear feedback shift registers

Another type of cipher is the following. Suppose that your alphabet is $A = \{0, 1, 2, ..., n-1\}$ and that the message space M is as above. Let $r = (r_1, r_2, ...)$ be an infinite sequence of "random" elements of A. Define the encryption map $E : M \to E$ by $E(m) = m + r \pmod{n}$, where addition is componentwise modulo n. Since r is a random sequence, any eavesdropper would think the received message is random as well. Define the decryption map $D : M \to E$ by $E(m) = m - r \pmod{n}$, where subtraction is componentwise modulo n. This is called a **one time key pad cipher** and r is called the **key**.

How do we construct a random sequence?

Let us begin with an example. How does the sequence

$$12343577824506956...$$

arise and how do you predict the next term in the sequence? Start with the "seed" 1234. To get the 5^{th} digit, add the 1^{st} and 2^{nd} modulo 10. To get the 6^{th} digit, add the 2^{nd} and 3^{rd} modulo 10, and so on. In general, if s_i denotes the i^{th} digit then

$$s_{4+i} = s_i + s_{i+1} \pmod{10}, \qquad i > 0.$$

In particular, the next term in the series is a 5 since $6+9 = 15 \equiv 5 \pmod{10}$. This is a special case of a linear feedback shift register sequence (LFSR).

Definition 1.7.23. *A* **linear feedback shift register sequence** *modulo n of length $k > 0$ is a sequence $s_1, s_2, ...$ such that $s_1, ..., s_k$ are given and*

$$s_{k+i} \equiv a_1 s_i + a_2 s_{i+1} + ... + a_k s_{k+i-1} \pmod{n}, \qquad i > 0, \qquad (1.5)$$

where $a_1, ..., a_k$ are given integers. An equation such as this is called a **recursion equation** *of length k modulo n.*

For security applications, one would like the sequence $\{s_i\}$ to be as "random looking" as possible.

S. Golomb [Gol] gives a list of three statistical properties a sequence of numbers $\mathbf{a} = \{a_n\}_{n=1}^{\infty}$, $a_n \in \{0, 1\}$, should display to be considered "random". Define the **autocorrelation** of $\mathbf{a}$ to be

$$C(k) = C(k, \mathbf{a}) = \lim_{N \to \infty} \frac{1}{N} \sum_{n=1}^{N} (-1)^{a_n + a_{n+k}}.$$

In the case where **a** is periodic with period P then this reduces to

$$C(k) = \frac{1}{P}\sum_{n=1}^{P}(-1)^{a_n+a_{n+k}}.$$

Assume **a** is periodic with period P.

balance: $|\sum_{n=1}^{P}(-1)^{a_n}| \leq 1$.

low autocorrelation:

$$C(k) = \begin{cases} 1, & k = 0, \\ \epsilon, & k \neq 0. \end{cases}$$

(For sequences satisfying these first two properties, it is known that $\epsilon = -1/P$ must hold.)

proportional runs property: In each period, half the runs have length 1, one-fourth have length 2, etc. Moveover, there are as many runs of 1's as there are of 0's.

To make the sequence $\{s_i\}$ be as "random looking" as possible, we'd like its period to be as long as possible. How do we do that? The answer depends on the theory of polynomials over finite fields, discussed in the next chapter. In order to not keep the reader in suspense and to motivate the material in the next chapter, we present the precise answer in the theorem below.

Theorem 1.7.24. *Let $S = \{s_i\}$ be defined by (1.5), where $n = p$ is a prime. The period of S is at most $N = p^k - 1$. It's period is exactly $p^k = 1$ if and only if the characteristic polynomial of*

$$A = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 1 \\ \vdots & & \dots & & \\ 0 & 0 & \dots & 0 & 1 \\ a_0 & a_1 & \dots & & a_{k-1} \end{pmatrix},$$

is irreducible and primitive [6] *over $\mathbb{F}_p$.*

[6] See Definition 2.6.5 in the next chapter.

This proof follows from the definition of primitive, and the construction of the finite extension fields of $\mathbb{F}_p$, discussed in the next chapter.

A related result is the following fact.

Theorem 1.7.25. *If* $\mathbf{c} = \{c_n\}_{n=1}^{\infty}$ *are the coefficients of* $f(x)/g(x)$*, where* $f, g \in \mathbf{F}_2[x]$ *and* $g(x)$ *is an irreducible polynomial with* x *primitive (mod* $g(x)$*) (i.e., powers of* x *(mod* $g(x)$*) yield every element in* $(\mathbf{F}_2[x]/g(x)\mathbf{F}_2[x])^{\times}$*). Then* $\mathbf{c}$ *is periodic with period* $P = 2^d - 1$ *(where* d *is the degree of* $g(x)$*) and satisfies Golomb's randomness conditions.*

A proof of this may be found in [Gol].

1.7.8 Feedback with carry shift register ciphers

In 1991 G. Marsaglia and A. Zaman [MZ] introduced a new class of pseudo-random number sequences. The shift register analogs of these have been investigated by A. Klapper and M. Goresky in several papers (see [GK1], for example).

The general idea is very simple. Let p be any integer and let a, b be positive integers with no factors in common with p or each other. If you take any rational number a/b then the coefficients $\mathbf{c} = \mathbf{c}(a, b)$ in the p-adic expansion

$$\frac{a}{b} = c_{-N}p^{-N} + ... + \frac{c_{-1}}{p} + c_0 + c_1 p + ... \ ,$$

are eventually periodic. Like decimal expansions and unlike power series expansions, the coefficients are NOT in general unique (for example, in the case $p = 10$ we have $1 = .999999...$). There is an analog of the Berlekamp-Massey decryption method [GK2]. What is worst (depending on your point of view) is that if you add two such sequences then the sequence $\mathbf{c} + \mathbf{c}'$ (defined using the carry operation) can be decrypted using at most $S + S'$ terms, where $S = S(\mathbf{c}) = \log_2(\max\{|a|_2, |b|_2\})$ is the **2-adic span** of $\mathbf{c} = \mathbf{c}(a, b)$. In particular, the stream cipher, when viewed from the perspective of FCSR's is even less secure than previously thought.

Example 1.7.26. *Let* $a = 10, b = 3$*, so*

$$\frac{10}{3} = 2 + 2^2 + 2^3 + 2^5 + 2^7 + 2^9 + 2^{11} + ...$$

whose coefficients are $0, 1, 1, 1, 0, 1, 0, 1, 0, 1, ...$

The coefficients in the 2-adic expansion of a/b are very easy to determine from the following algorithm:

- Find the least k such that $2^k - 1$ is divisible by b (the is the **order** of $b \pmod 2$) and let $2^k - 1 = bd$.
- Write ad as a polynomial in powers of 2 with coefficients in $\{0, 1\}$.
- Using the geometric series expansion $\frac{1}{1-x} = 1 + x + x^2 + ...$ compute

$$\frac{a}{b} = \frac{ad}{bd} = \frac{ad}{2^k - 1}$$

  as a power series in 2 (this converges in the 2-adic norm and represents the 2-**adic expansion** of a/b).

Theorem 1.7.27. *When b is a power of a prime and when 2 is primitive mod b (i.e., 2 generates the cyclic group $(\mathbb{Z}/b\mathbb{Z})^\times$) then the coefficients of the 2-adic expansion of $1/b$ satisfy analogs of Golomb's randomness conditions.*

For a proof, see [GK2].

1.7.9 Application: calendar calculations

A **leap year** is a year y which satisfies either (a) $4|y$ and $100 \not| y$, or (b) $400|y$.

First method

Let Sunday be 1, Monday be 2, ..., Saturday be 7.

1. Divide the last 2 digits of the year by 4 and discard the fraction, call it a_1.
2. Add the date of the month to a_1, call the result a_2.
3. Add to a_2 the month's **key value** as given by the table below and call the result a_3.

Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1	4	4	0	2	5	0	3	6	1	4	6

4. Subtract 1 from a_3 if the month is a Jan or Feb of a leap year. Call the result a_4.

5. To a_4, add
$$\begin{cases} 0, & \text{if the year was in the 1900's,} \\ 6, & \text{if the year was in the 2000's,} \\ 4, & \text{if the year was in the 1700's,} \\ 2, & \text{if the year was in the 1800's,} \end{cases}$$
and if the year is not above then add a multiple of 400 to convert to one of the above cases. Call the result a_5.

6. Add last 2 digits of the year to a_5 and call the result a_6.

7. Let $1 \leq a_7 \leq 7$ satisfy $a_6 \equiv a_7 \pmod 7$.

This is the desired day of the week.

Notation: Let $[x]$ denote the integer part of a real number x.

Example 1.7.28. *Find the day of the week of* $9-14-99$ *using the above algorithm.*

(1) $[99/4] = 24$, *(2)* $24+14 = 38$, *(3)* $38+6 = 44$, *(4)* $44+0 = 44$, *(5)* $44+99 = 143$, *(6)* $143 \equiv 3 \pmod 7$.

The third day is Tuesday.

Second method (due to Conway)

The day of the week for the last day of February (either 2-28 or 2-29) is called **doomsday**.

Example 1.7.29. *Doomsday for 2001 is Wednesday, for 2000 it is Tuesday, for 1999 it is Sunday, and for 1900 it is Wednesday.*

A number of useful facts are known about the doomsday.

Fact 1.7.30.

- *April 4, June 6, October 10, and December 12 (i.e., 4-4, 6-6, 8-8, 10-10, and 12-12) are all doomsdays (easy to remember since they involve all even numbered months* > 3*).*

- *May 7, July 11, September 5, and November 7 (i.e., 5-9, 9-5, 7-11, 11-7) are all doomsdays (easy to remember since all odd numbered months* > 3 *are involved, along with the mneumonic "I work at the 7-11 from 9 to 5.").*

- *March 0 (last day of February) is a doomsday.*
- *If the year is not a leap year then January 31 (1-31) is a doomsday.*
- *If the year is a leap year then January 32 (1-32) is a doomsday.*

Fact 1.7.31. *The doomsday for* $19YY$ *is given by adding* $[YY/12]+(YY \bmod 12)+[\frac{YY \bmod 12}{4}] \pmod 7$ *to Wednesday. For* $20YY$*, the procedure is the same except add to Tuesday.*

Example 1.7.32. *Find the day of the week of* $9-14-99$ *and* $9-14-2001$ *using the above algorithm.*

Doomsday for 1999 is a Sunday. By the above fact, 9-5-99 is a doomsday (Sunday), so 9-12-99 is Sunday. Therefore, 9-14-99 is a Tuesday. Similary, 9-5-2001 is a doomsday (Wednesday), so 9-14-2001 must be a Friday.

1.7.10 The Chinese remainder theorem

In this section, we see how to solve simple simultaneous congruences modulo n. This will be applied to the study of the Euler ϕ-function.

Theorem 1.7.33. *(Chinese remainder theorem) Let* $n_1 > 1$ *and* $n_2 > 1$ *be relatively prime integers. Then*

$$x \equiv a_1 \pmod{n_1}, \qquad x \equiv a_2 \pmod{n_2}, \tag{1.6}$$

has a simultaneous solution $x \in \mathbb{Z}$*. Furthermore, if* x, x' *are two solutions to (1.6) then* $x' \equiv x \pmod{n_1 n_2}$*.*

Example 1.7.34. $23 \equiv 7 \pmod 8$ *and* $23 \equiv 3 \pmod 5$*.*

proof: $x \equiv a_1 \pmod{n_1}$ if and only if $x = a_1 + kn_1$, for some k. Therefore, the truth of the existence claim above is reduced to finding an integer k such that $a_1 + kn_1 \equiv a_2 \pmod{n_2}$. Since $gcd(n_1, n_2) = 1$, there are integers r, s such that $1 = rn_1 + sn_2$, so $a_2 - a_1 - (a_2 - a_1)rn_1 = (a_2 - a_1)sn_2$. This implies $kn_1 \equiv a_2 - a_1 \pmod{n_2}$, where $k = r(a_2 - a_1)$. Thus a solution exists.

To prove uniqueness mod $n_1 n_2$, let $x \equiv a_1 \pmod{n_1}, x \equiv a_2 \pmod{n_2}$ and $x' \equiv a_1 \pmod{n_1}, x' \equiv a_2 \pmod{n_2}$. Subtracting, we get $x' \equiv x \pmod{n_1}$ and $x' \equiv x \pmod{n_2}$. Since $gcd(n_1, n_2) = 1$, the result follows. □

General Chinese remainder theorem

Theorem 1.7.35. *(Chinese remainder theorem, general version) Let* $n_1 > 1, n_2 > 1, ..., n_k > 1$ *be pairwise relatively prime integers. Let* $n = n_1 n_2 ... n_k$. *Then*

$$x \equiv a_1 \pmod{n_1},\ x \equiv a_2 \pmod{n_2}, ..., x \equiv a_k \pmod{n_k} \qquad (1.7)$$

has a simultaneous solution $x \in \mathbb{Z}$. *Furthermore, if* x, x' *are two solutions to (1.7) then* $x' \equiv x \pmod{n}$.

This follows from the $k = 2$ case proven above using mathematical induction. The details are left as an exercise. We give a different proof below.

proof: As a runs over all n integers $0 \leq a < n$, the k-tuples

$$(a \pmod{n_1}, ..., a \pmod{n_k})$$

form a collection of n distinct k-tuples in $\{0, 1, ..., n-1\}^k$. (Exercise: show why they are distinct.) On the other hand, there are n distinct, k-tuples $(a_1, a_2, ..., a_k)$ with $0 \leq a_i < n_i$. Therefore, each k-tuple $(a_1, a_2, ..., a_k)$ must equal one of the $(a \pmod{n_1}, ..., a \pmod{n_k})$, for $0 \leq a < n$. □

1.7.11 An application to Euler's ϕ-function

Let

$$\Phi_n = \{1 \leq a \leq n \mid gcd(a, n) = 1\},$$

as in 1.4.5.

Proposition 1.7.36. *If* $gcd(m, n) = 1$ *then there is a 1-1, onto mapping between the sets*

$$f : \Phi_m \times \Phi_n \to \Phi_{mn}.$$

proof: If $a \in \Phi_m$ and $b \in \Phi_n$, then define $f(a, b) = x$, where x satisfies $x \equiv a \pmod{m}, x \equiv b \pmod{n}$. This x exists and is unique mod mn, by the Chinese remainder theorem, so f is well-defined - provided we show $x \in \Phi_{mn}$. To show this, we must show that $(x, mn) = 1$ if $(x, m) = (x, n) = 1$, and that if $(a, m) = 1$ and $x \equiv a \pmod{m}$, then $(x, m) = 1$. We leave these as exercises.

f is 1-1: If $x = f(a, b) = f(a', b')$ then $x \equiv a \pmod{m}, x \equiv b \pmod{n}$ and $x \equiv a' \pmod{m}, x \equiv b' \pmod{n}$. Subtracting, we get $a' \equiv a \pmod{m}, b' \equiv$

$b \pmod n$, so $a = a'$ and $b = b'$ by definition of Φ_m and Φ_n. Therefore, f is 1-1.

f is onto: Let $x \in \Phi_{mn}$. Define $1 \leq a \leq m$ and $1 \leq b \leq n$ by $x \equiv a \pmod m, x \equiv b \pmod n$. Then, by definition $x = f(a, b)$. Therefore, f is onto. □

The following result has been proven by more direct methods above. We given another proof as an application of the Chinese remainder theorem.

Proposition 1.7.37. *If $gcd(m, n) = 1$ then $\phi(m)\phi(n) = \phi(mn)$.*

proof: The number of elements in the range of f is $\phi(m)\phi(n)$ and the number of elements in the domain of f is $\phi(mn)$. Since f is a bijection, these two must be equal. □

Corollary 1.7.38. *If $n = p_1^{e_1}...p_k^{e_k}$ is the prime factorization of an integer n then $\phi(n) = n \cdot (1 - 1/p_1)(1 - 1/p_2)...(1 - 1/p_k)$.*

Example 1.7.39. $\phi(100) = 100 \cdot (1 - 1/2)(1 - 1/5) = 40$.

Exercise 1.7.40. *Verify Example 1.7.39.*

Exercise 1.7.41. *Define $s_{i+2} = s_{i+1} + s_i \pmod{19}$, $s_1 = 1$, $s_2 = 1$. Find a formula for s_i as in the above example.*

Exercise 1.7.42. *(***Fibonacci numbers***) Let s_n satisfy $s_n = s_{n-1} + s_{n-2}$ and let $s_1 = 1, s_2 = 0$. Solve for s_n.*

Exercise 1.7.43. *Let s_n satisfy $s_{n+3} = s_{n+1} + s_n$ and let $s_0 = 3$, $s_1 = 0$, $s_2 = 2$. Solve for s_n. (This is the* **Lucas-Perrin sequence***.)*

Exercise 1.7.44. *An internet pyramid scheme starts with a group of a friends in different states. They come up with an email which asks the recipient to forward the email to b others. Find the linear recurrence relation which describes this process after n iterations (assuming every email is sent to a different person). First, solve it in terms of a and b in general. Next, for specific values of a and b (say $a = 3$ and $b = 2$).*

Exercise 1.7.45. *An internet pyramid scheme starts with a group of a friends in different states. They come up with an email which asks the recipient to add their name onto the list (which originally contains the names of the a friends) and forward the email to N others, where N is the number*

of people on the list. Find the linear recurrence relation which describes this process after n iterations (assuming every email is sent to a different person). First, solve it in terms of a in general. Next, for specific values of a (say $a = 3$).

Exercise 1.7.46. *Use Caeser's cipher in §1.7.7 to decode "ehdw dupb". (Hint: a commonly uttered phrase at the U. S. Naval Academy.)*

Exercise 1.7.47. *Find the first 20 terms of the LFSR with seed* 12345 *and recursion equation* $s_{5+i} \equiv s_i + s_{i+1} \pmod{10}$, $i > 0$.

Exercise 1.7.48. *Find the first 20 terms of the LFSR with seed* 10101 *and recursion equation* $s_{5+i} \equiv s_i + s_{i+1} \pmod{2}$, $i > 0$.

Exercise 1.7.49. *A sequence* $\{a_n\}_{n=1,2,\ldots}$ *is* **eventually periodic** *if there are fixed* $P > 0$ *(called the* **period***) and* $n_0 > 0$ *such that* $a_n = a_{n+P}$*,for all* $n > n_0$*. (If* $n_0 = 0$ *then we call the sequence* **periodic***.) Are the sequences in the above exercises eventually periodic? If so, what are their periods?*

Exercise 1.7.50. *Using the algorithm in §1.7.9, solve for the day of the week of your birthday.*

Exercise 1.7.51. *Using the algorithm in §1.7.9, solve for the day of the week of today's date.*

Exercise 1.7.52. *Using the algorithm in §1.7.9, solve for the day of the week of your birthday.*

Exercise 1.7.53. *Using the algorithm in §1.7.9, solve for the day of the week of today's date.*

Exercise 1.7.54. *At the annual Mathematics Department Fun Run, when the joggers lined up 4 abreast there was 1 left over, when the joggers lined up 5 abreast there were 2 left over. How many were at the fun run? (Take the smallest solution to the congruences.)*

Exercise 1.7.55. *Solve the following problem: If eggs are removed from a basket* 2, 3, 4, 5, *and* 6 *at a time, there remain respectively* 1, 2, 3, 4, *and* 5 *eggs. But if the eggs are removed* 7 *at a time no eggs remain. What is the least number of eggs that could have been in the basket?*

Exercise 1.7.56. *A politician runs for office every 6 years and cheats on his taxes every 5 years. He was just elected and cheated on his taxes last year. When is the next time he does both in the same year?*

Exercise 1.7.57. *At a local triathalon, when everyone lined up 4 abreast there was 1 left over, when everyone lined up 5 abreast there were 2 left over, and when everyone lined up 7 abreast there were 3 left over. How many were at the triathalon? (Take the smallest solution to the congruences.)*

Exercise 1.7.58. *Show that* $\Phi_{10} = \{1, 3, 7, 9\}$*, in the notation of §1.7.11. Compute* $\{17 (\mathrm{mod}\ 10), 17 \cdot 3 (\mathrm{mod}\ 10),\ 17 \cdot 9 (\mathrm{mod}\ 10), 17 \cdot 9 (\mathrm{mod}\ 10)\}$.

1.8 Integral powers mod n

In calculus, you learn to solve (or at least find approximate solutions to) equations of the form $y = a^x$. Solving for x in terms of y incolves using the logarithm to base a.

We want to do something analogous here. One difference here is that, because the integers are discrete, there are no approximate solutions!

Definition 1.8.1. *Let* $a \geq 1, n > 1$ *be relatively prime integers. We say that* a *has* **order** k *modulo* n *is* k *if* k *is the smallest positive integer satisfying* $a^k \equiv 1 \pmod{n}$*. The order is denoted* $\mathrm{ord}_n(a)$.

For example, the order of $a = 2$ modulo $n = 7$ is $k = 3$ since $2^3 \equiv 1 \pmod{7}$.

One way to think about the material in this section is it is a study of properties of the order function. The order of an integer is not "easy" to find in the sense that if n is a large integer then there is no known "efficient" algorithm for determining $\mathrm{ord}_n(a)$ [BS].

1.8.1 Fermat's Little Theorem, revisited

We prove a special case of Euler's theorem, Fermat's little theorem, first. The proof given here, unlike the one in §1.7.5, can be generalized to the proof of Euler's theorem.

Theorem 1.8.2. *(Fermat's Little Theorem) If* p *is a prime then* $a^p \equiv a \pmod{p}$.

Before proving Fermat's Little Theorem, we need a key property concerning residue classes (or equivalence classes) mod n.

Lemma 1.8.3. *Let $n > 1$ be an integer. Let a be any integer relatively prime to n. Then each integer in the set $a\Phi_n = \{a \cdot r \mid gcd(r, n) = 1\}$ is relatively prime to n and they are mutually incongruent to each other.*

proof of lemma: The elements of $\{a \cdot r \mid gcd(r, n) = 1\}$ are all relatively prime to n since $gcd(a, n) = 1$. They are mutually incongruent to each other since if $ia \equiv aj \pmod{n}$ then by the cancellation law (the third part of Lemma 1.7.3), $i \equiv j \pmod{n}$, for any $i, j \in \Phi_n$. But if $1 \leq i, j \leq n - 1$ then we must have $i = j$. □

proof of theorem: First, assume $p \not| a$. The set $\{1, 2, ..., p - 1\}$ forms a complete set of representatives of residue classes modulo p. The set $\{a, 2a, ..., (p-1)a\}$ must as well, by the above lemma. Therefore, for each $1 \leq i \leq p - 1$ there is a unique $1 \leq k_i \leq p - 1$ such that $i \equiv ak_i \pmod{p}$. Multiply all these together:

$$(p-1)! \equiv 1 \cdot 2 \cdot ... \cdot (p-1) \equiv ak_1 \cdot ak_2 \cdot ... \cdot ak_{p-1} \equiv a \cdot 2a \cdot ... \cdot (p-1)a \equiv (p-1)! a^{p-1} \pmod{p}.$$

Since $p \not|(p-1)!$, the cancellation law gives $a^{p-1} \equiv 1 \pmod{p}$. Multiply both sides by a to get $a^p \equiv a \pmod{p}$.

If $p|a$ then $a \equiv 0 \pmod{p}$, so $a^p \equiv 0 \pmod{p}$, so by the transitivity of congruence, we have $a^p \equiv a \pmod{p}$, as desired. □

1.8.2 Euler's theorem

Now we can state Euler's generalization.

Let $n > 1$ be an integer. Recall that $\phi(n)$ is then number of positive integers relatively prime to n.

Theorem 1.8.4. *(Euler's Theorem) If $n > 1$ is an integer and $gcd(a, n) = 1$ then $a^{\phi(n)} \equiv 1 \pmod{n}$.*

proof: The proof of Fermat's Little Theorem above can be modified somewhat to work in this more general case as well. Exercise: Try to do this. □

We know that if a number p is prime then $a^p \equiv a$ (**mod** p). An integer n for which $a^n \equiv a$ (**mod** n) is called a **Fermat pseudoprime to base**

a. Roughly speaking, if n is a Fermat pseudoprime to many different bases then n is "probably" a prime [7].

Example 1.8.5. *We have* $\phi(10) = 4$ *(see Example 1.4.10 above), so by Euler's theorem,* $1001^4 \equiv 1 \pmod{10}$. *We can check this by hand without a computer: by the binomial expansion* $1001^4 = (1000+1)^4 = 1000^4 + 4 \cdot 1000^3 + 6 \cdot 1000^2 + 4 \cdot 1000 + 1 = 1004006004001 \equiv 1 \pmod{10}$.

1.8.3 Application: Decimal expansions of rational numbers

Let $a_1, a_2, ...$ be a sequence of integers. We call the sequence **eventually periodic** if there is an $i_0 \in \mathbb{N}$ and a $P \in \mathbb{N}$ such that $a_{i+P} = a_i$, for all $i > i_0$. The smallest P for which this holds if called the **period** of the sequence.

It is well-known that a real number is rational if and only if it is eventually periodic.

Lemma 1.8.6. *A real number* $r \in \mathbb{R}$ *has an eventually periodic decimal expansion if and only if it is a rational number,* $r \in \mathbb{Q}$.

proof: Let

$$r = b_0 b_1 ... b_r . a_0 a_1 ..., \qquad a_i, b_j \in \{0, 1, .., 9\},\ b_0 \neq 0,$$

denote the decimal expansion of $r \in \mathbb{R}$. We shall call each a_i or b_j a **digit** in the expansion. One direction is very easy. Using the geometric series expansion $1/(1-x) = 1 + x + x^2 + ...$ is is easy to see that if the expansion of r is eventually periodic then $r \in \mathbb{Q}$. The converse is less trivial. It suffices to consider the case $0 < r < 1$, since adding or subtracting an integer won't affect whether or not the expansion is eventually periodic. Let $r = a/b \in \mathbb{Q}$, where a, b are relatively prime integers. If $gcd(b, 10) \neq 1$ then $b = 2^r 5^s b'$, where $r \geq 0$, $s \geq 0$, and $gcd(b', 10) = 1$. Note that a/b is eventually periodic if and only if $10^t a/b$ is eventually periodic, where $t = max(r, s)$. Therefore, after possibly replacing a/b by $10^t a/b$, we may assume without loss of generality that $gcd(10, b) = 1$. By Euler's theorem, if $(10, b) = 1$ we know that there is a $k > 0$ such that $10^k \equiv 1 \pmod{b}$ (indeed, we may take

[7] However, it has been shown by Pomerance, Granville, and Alford that there are infinitely many composite integers which are Fermat pseudoprimes to all bases a.

$k = \phi(b)$ and the smallest possible such k must divide $\phi(b)$, where ϕ is Euler's totient function). If $bd = 10^k - 1$ then we may write

$$\frac{ad}{bd} = \frac{ad}{10^k - 1} = \frac{ad}{10^k}\frac{1}{1 - 10^{-k}} = \frac{ad}{10^k}(1 + 10^{-k} + 10^{-2k} + ...).$$

Since $ad < bd < 10^k$, the number of non-zero digits in the decimal expansion of $\frac{ad}{10^k}$ is at most k. Therefore r has a periodic decimal expansion. □

First, to illustrate the ideas in a simpler setting, let us consider the special case of the expansion of the number $r = 1/p$, p prime not equal to 2 or to 5. By the above proof, we know that if $k > 0$ is *any* integer satisfying $10^k \equiv 1 \pmod{p}$ then $1/p$ is periodic with period k. However, this period may not be the *smallest* possible period (remember, a sequence of integers which repeats every k times will also repeat every $2k$ times, for instance). We are naturally lead to the following concept (one which also occurs later in the study of group theory as well).

The above proof implies that the following result holds.

Lemma 1.8.7. *Let $k > 0$ be the order of 10 modulo p (in other words, suppose $p|99...9$, k 9's in a row, and k is smallest possible). Then $1/p$ is periodic with period k and there is no smaller period.*

Example 1.8.8. *(1) Fermat's little theorem gives $10^6 \equiv 1 \pmod{7}$. The order of 10 modulo 7 is 6, so $1/7 = .142857142857142...$ is period 6.*

(2) Fermat's little theorem gives $10^{10} \equiv 1 \pmod{11}$. The order of 10 modulo 11 is 2, so $1/11 = .090909...$ is period 2.

(3) Fermat's little theorem gives $10^{18} \equiv 1 \pmod{19}$. The order of 10 modulo 19 is 18, so $1/19 = .052631578947368421052631578947368421 05...$ is period 18.

(4) Fermat's little theorem gives $10^{36} \equiv 1 \pmod{37}$. The order of 10 modulo 37 is 3, so $1/37 = .02702702...$ is period 3.

This raises the question: When is the period of $1/p$ as long as possible (i.e., $p - 1$)? We shall call such a prime a **full period** prime. A curiosity about full-period primes: if the prime is p, the period is p-1, so look at the first $(p - 1)/2$ and last $(p - 1)/2$ digits in the expansion of $1/p$. Those two $(p - 1)/2$-digit numbers add up to 9999...9999.

Example 1.8.9. *(a) $1/7$ is .142 857 142 857 ... and $142 + 857 = 999$.*

(b) $1/23$ is 0.04347826086 95652173913 043... and
$04347826086 + 95652173913 = 99999999999$

We are naturally lead to the following concept (one which also occurs later under a different name in the study of cyclic groups).

Definition 1.8.10. *Let p be a prime. We say that a is a* **primitive root** *modulo p if the smallest $k > 0$ satisfying $a^k \equiv 1 \pmod{p}$ is $k = p - 1$.*

Thus the question "When is the period of $1/p$ as long as possible?" becomes "For which primes p is 10 a primitive root mod p?" To our knowledge, there does not seem to be a simple answer to this question. Artin conjectured in the 1920s the following statement.

Conjecture 1.8.11. *(Artin) Every positive integer $x > 1$ is the primitive root of p for infinitely many primes p.*

So that means that 10 *should* be the primitive root for infinitely many primes p, so there should be infinitely many full-period primes. Quantitatively, the conjecture boils down to 37% of all primes asymptotically have 10 as primitive root (the 37% is really an approximation to **Artin's constant**; it's $\prod_{p \text{ prime}}(1 - 1/(p(p-1)))$). Although many people have tried, Artin's conjecture is not yet proven.

1.8.4 Application: RSA encryption

Ron Rivest, Adi Shamir, and Leonard Adleman developed RSA in 1977 [RSA][8]. Two excellent expository accounts can be found in Cosgrave [Co] and Wardlaw [Wa].

In this section, we assume that you have somehow translated the message you wish to send into an integer (i.e., a finite sequence of digits which does not start with a 0).

RSA works as follows: Take two large primes, p and q, and compute their product $n = pq$. In practice, one chooses both p and q to have at least 100 digits and p, q are kept secret (the sender of the message). Because factoring very large integers seems to be such a very hard problem, it is widely believed that such a choice of p, q will make it practically impossible to determine p, q given n. Choose an integer e with $0 < e < n$ and $gcd(e, (p-1)(q-1)) = 1$. Find the multiplicative inverse of e modulo $(p-1)(q-1)$, call it d. The

[8] It has recently been declassified that a very similar idea was developed earlier by Cliff Cocks, a British cryptographer, though the Cocks paper was classified until 1997 [E].

values e and d are called the **public** and **private exponents**, respectively. The **public key** is the pair (n, e). The **private key** is (n, d).

Suppose Alice wants to send a message m to Bob. *We assume in this section that* $gcd(m, n) = 1$. We also assume that Alice and Bob have found a way to secretly share the private key (n, d).

step 1: Alice creates the enciphered message ("ciphertext") c by exponentiating the message: $c \equiv m^e \pmod{n}$, where e and n are Bob's public key. In other words, in the notation of (1.4) the encryption map is $E(m) = m^e \pmod{n}$.

step 2: She sends c to Bob.

step 3: To decrypt, Bob also exponentiates: $m \equiv c^d \pmod{n}$. The relationship between e and d (namely, $de \equiv 1 \pmod{(p-1)(q-1)}$) ensures that Bob correctly recovers m. Since d is kept secret, no one else can (easily) decrypt this message without knowing d (or p, q, from which one can determine d).

Example 1.8.12. *Let* $p = 3$, $q = 7$, *so* $\phi(n) = \phi(21) = 12$. *Let* $e = 11$ *(the public exponent), so* $d = 11$ *(the private exponent). Let* $m = 10$ *be the message. The "cipher text" is* $c = 19$ *since* $m^e = 10^{11} \equiv 19 \pmod{21}$.

Example 1.8.13. *Let* $p = 17$, $q = 19$, *so* $\phi(n) = \phi(323) = 288$. *Let* $e = 97$ *(the public exponent), so* $d = 193$ *(the private exponent). Let* $m = 100$ *be the message. The "cipher text" is* $c = 168$ *since* $m^e = 100^{97} \equiv 168 \pmod{323}$.

Small exponent attack on RSA

Is RSA secure if the private exponent is "small"? Not if you send the same message to many people, even if you use different public bases n for each message. This is called "Hastad's broadcast attack".

Suppose that the exponent d is small, for example, $d = 3$. Let the private keys be $(n_1, 3)$, $(n_2, 3)$, $(n_3, 3)$, where each n_i is a product of two large secret primes, and where the message m satisfies $m < n_i$. Assume that n_1, n_2, n_3 are pairwise relatively prime. Suppose that you send the same RSA encrypted message m to $d = 3$ people. The idea is very simple. Assume $m^3 \pmod{n_1}$ is intercepted, as is $m^3 \pmod{n_2}$ and also $m^3 \pmod{n_3}$. (This assumption is a standard hypothesis when constructing an attack, since the encrypted messages are presumably traveling in the open). The Chinese remainder theorem allows one to determine $m^3 \pmod{n_1 n_2 n_3}$. But since $m < n_1$,

$1 \leq i \leq 3$, we must have $m^3 < n_1 n_2 n_3$. Thus, $m^3 < n_1 n_2 n_3$ really is just m^3. So take the cube root, and you have the message.

1.8.5 Application: The discrete log problem

Let p be a prime number and let $1 < a < p$ be an integer. Let $(\mathbb{Z}/p\mathbb{Z})^\times$ denote all the non-zero elements of $\mathbb{Z}/p\mathbb{Z} = \{\overline{0}, \overline{1}, ..., \overline{p-1}\}$. Consider the map $e_a : \mathbb{Z} \to (\mathbb{Z}/p\mathbb{Z})^\times$,

$$e_a(n) = a^n \,(\mathrm{mod}\, p), \qquad n \geq 0.$$

When this "exponential map" is onto then a a primitive root mod p. If the order of $a \,(\mathrm{mod}\, p)$ is $p-1$ then a is a primitive root.

Though this map is analogous to the exponential map on the real numbers, there are some big differences between them! For example, if $y > 0$ is given and you you want to solve $\exp(x) = y$ for x (i.e., take the log of y) then you can use a pocket calculator to quickly find a "good" approximation for x (where by "good" we mean the approximate answer the calculator gives you is probably close enough for the application you have in mind). It is worth emphasizing that the calculator will return an answer almost instantly. This basically comes from the fact that, thanks to calculus, there is a Taylor series expansion for $\exp(x)$ which converges rapidly and which can be "hard wired" into the calculators circuitry.

Discrete log problem: Given a non-zero b in $\mathbb{Z}/p\mathbb{Z}$, if $e_a(n) = b$ holds for some n then find n. (The smallest such $n > 0$, if it exists, is called the **discrete log** of b to base a mod p.)

The situation of computing logs is different in the discrete setting. Given a non-zero b in $\mathbb{Z}/p\mathbb{Z}$, even if you know $e_a(n) = b$ holds for some n (we know this is true when a is a primitive root mod p), it doesn't make sense to ask for an "approximation" to n. More seriously though, there is no "fast" algorithm known to compute n given p and a. Moreover, it should be emphasized that the security of many cryptographic schemes depend on this "fact" that the discrete log problem is "hard".

1.8.6 Application: Diffie-Hellman key exchange

How do two people, Amelia and Ben, share a secret over an open channel of communications? It may seem incredible at first, but it is possible and is

the remarkable discovery of Whitfield Diffie, now at Sun Microsystems, and Martin Hellman, an electrical engineering professor at Stanford University, in 1976.

The idea is to first determine a shared secret key, which one could use (for example) to encrypt the messages between them. Its security depends on the difficulty of the discrete log problem, discussed in the previous section.

- Pick a "large" prime number p and a generator a of $(\mathbb{Z}/p\mathbb{Z})^\times$, $2 \leq a \leq p-2$. Publish them.
- Exchange data as follows:
  - Amelia picks a random secret x, $2 \leq x \leq p-2$, sends Ben, $a^x \pmod p$.
  - Ben picks a random secret y, $2 \leq y \leq p-2$, sends Amelia , $a^y \pmod p$.
- Compute keys as follows:
  - Amelia receives $a^y \pmod p$ and computes $K = (a^y)^x \pmod p$.
  - Ben receives $a^x \pmod p$ and computes $K = (a^x)^y \pmod p$.
- K is the shared secret key.

Example 1.8.14. *Pick* $p = 541$, $a = 2$. *Amelia chooses* $x = 137$ *and sends* $a^x = 2^{137} \equiv 208 \pmod{541}$ *to Ben. Ben picks* $y = 193$ *and sends* $a^y = 2^{193} \equiv 195 \pmod{541}$ *to Amelia. They compute the secret shared key*

$$K = (a^x)^y \equiv (2^{137})^{193} \equiv (208)^{193} \equiv 486 \pmod{541}.$$

More details are given in [MOV], §12.6.

1.8.7 Application: ElGamal encryption

ElGamal is another public key encryption idea. It is different from the RSA encryption scheme in that its security depends on the difficulty depends on the difficulty of the discrete log problem, as opposed to the factoring problem.

Anne's key

- Pick a "large" prime number p and a generator a of $(\mathbb{Z}/p\mathbb{Z})^\times$, $2 \leq a \leq p-2$. Publish them.

- Anne picks a random secret k, $2 \leq k \leq p-2$ and publishes $a^k \pmod{p}$.
- Anne's **public key** is $(p, a, a^k \pmod{p})$. Anne's **private key** is k.

ElGamal encryption
Ben sends message m to Anne as follows:

- Ben obtains Anne's public key $(p, a, a^k \pmod{p})$.
- Ben represents the message m as an integer $\leq m \leq p-1$ (by choosing p sufficiently large, this is always possible; alternatively, the message may be broken up into smaller pieces, each piece belonging to the range $0, 1, ..., p-1$).
- Ben selects a random integer ℓ, $1 \leq \ell \leq p-2$.
- Ben computes $d \equiv a^\ell \pmod{p})$ and $e \equiv m \cdot (a^k)^\ell \pmod{p})$, $1 \leq d, e \leq p-1$.
- Ben sends the "ciphertext" $c = (d, e)$ to Anne.

Anne decrypts m as follows:

- Anne uses her private key to compute $d^{p-1-k} \pmod{p})$ and $d^{p-1-k} \equiv (a^{-k\ell} \pmod{p})$.
- Anne computes $m \equiv (d^{p-1-k}) \cdot e \pmod{p})$.

Example 1.8.15. *Using the numerical position in the alphabet, "A" is the* 1^{st} *letter, which corresponds numerically to* 01, *"B" to* 02, ..., *"Z" to* 26. *A word corresponds to the associated string of numbers. Note, for example, "ab" is* $0102 = 102$ *but "ob" is* 1502.

Pick $p = 150001$, $a = 7$. *Pick* $k = 113$, *so that* $a^k = 7^{113} \equiv 66436 \pmod{p}$. *Anne's public key is* $(150001, 7, 66436)$. *Anne's private key is* 113. *Suppose Ben is Anne's son and wants to send the message "Hi Mom". He converts this to* $m = 080913 = 809$ *and* $m = 131513$. *Ben picks* $\ell = 1000$ *and computes* $7^\ell \equiv 90429 \pmod{150001})$, *so* $d = 90429$. *The encryption is* $809 \cdot (7^{113})^{1000} \equiv 15061 = e \pmod{150001})$ *and* $131513 \cdot (7^{113})^{1000} \equiv 57422 = e \pmod{150001})$, *so* $c = (90429, 15061)$ *and* $c = (90429, 57422)$ *are sent to Anne.*

Anne computes $d^{p-1-k} = 90427^{149887} \equiv 80802 \pmod{150001}$. *Then* $m \equiv 80802 \cdot e \pmod{150001}$ *returns the original messages* $m = 809$ *and* $m = 131513$. *Since* 809 *has an odd number of digits, the first digit must have been a* 0. *Anne can see that* 809 *must mean that the first letter comes from the* $8 = 08$, *the second letter comes from a* 09. *These spell out "hi". For* $m = 131513$, *the third letter must come from the* 13, *the fourth letter must come from the* 15, *and the last must be a* 13. *These spell out "hi" and "mom". Adding a space and proper capitalization, this is "Hi Mom"!*

More details are given in [MOV], §8.4.

Exercise 1.8.16. *(a) Let* $p = 541$, $a = 2$. *Pick* $k = 113$, $\ell = 101$. *Encrypt the messages* $m = 200$ *and* $m = 201$ *using ElGamal.*

(a) Let $p = 541$, $a = 2$, $k = 101$. *Decrypt the ElGamal ciphers* $c = (54, 300)$ *and* $c = (54, 301)$.

Exercise 1.8.17. *Let* $p = 541$, $a = 2$. *Amelia chooses* $x = 17$ *and sends* $a^x = 2^{17} \pmod{541}$ *to Ben. Ben picks* $y = 71$ *and sends* $a^y = 2^{71} \pmod{541}$ *to Amelia. Compute the secret shared (Diffie-Hellman) key.*

Exercise 1.8.18. *Show that there is no* k *satisfying* $2^k \equiv 1 \pmod{10}$. *Why doesn't this contradict Definition 1.8.1?*

Exercise 1.8.19. *Find*

(a) the order of 2 *mod* 3,

(b) the order of 2 *mod* 5,

(c) the order of 7 *mod* 10,

(d) the order of 49 *mod* 10.

Exercise 1.8.20. *Verify* $a^{\phi(n)} \equiv 1 \pmod{n}$ *when* $a = 3$ *and* $n = 100$.

Exercise 1.8.21. *Is* 2 *a primitive root mod* 11*? Find the discrete log, if it exists, of* 6 *to the base* 2, *mod* 11.*(Ans: yes,* 9*)*

Exercise 1.8.22. *Is* 2 *a primitive root mod* 541*? Find the discrete log, if it exists, of* 34 *to the base* 2, *mod* 541. *(Ans: yes,* 100*)*

Exercise 1.8.23. *Examine the first 100 digits in the decimal expansions of the following fractions, looking for any patterns. Try to predict the periods, then deduce them using the ideas from this section:*

$$x = 5000/4999,$$

$$y = 1000000000000/998999999001,$$

$$z = 1000099990001000000000000/999999970000000299999999.$$

Exercise 1.8.24. *Given p, q, e as in Example 1.8.12, decrypt $c = 20$.*

1.9 Arithmetic properties of $\mathbb{Z}/n\mathbb{Z}$: a summary

In this section, we give an abstract summary of the main results discussed in this chapter.

Fix an integer modulus $n > 1$. Recall that if a is any integer then $\overline{a} = a + n\mathbb{Z}$. Let

$$\mathbb{Z}/n\mathbb{Z} = \{\overline{0}, \overline{1}, ..., \overline{n-1}\},$$

denote the set of all residue classes of n. Define addition $+$ and multiplication $\cdot$ in $\mathbb{Z}/n\mathbb{Z}$ by the following rules:

$$(a + n\mathbb{Z}) + (b + n\mathbb{Z}) = a + b + n\mathbb{Z},$$

and

$$(a + n\mathbb{Z}) \cdot (b + n\mathbb{Z}) = a \cdot b + n\mathbb{Z}.$$

The properties of addition and multiplication are summarized in the following proposition.

Proposition 1.9.1. *Fix an integer modulus $n > 1$. For any integers a, b, c,*

1. $\overline{a} + \overline{b} = \overline{a+b} = \overline{b} + \overline{a}$, *("addition is commutative")*
2. $\overline{a}\overline{b} = \overline{ab} = \overline{b}\overline{a}$, *("multiplication is commutative")*
3. $(\overline{a} + \overline{b}) + \overline{c} = \overline{a} + (\overline{b} + \overline{c})$, *("addition is associative")*
4. $(\overline{a}\overline{b})\overline{c} = \overline{a}(\overline{b}\overline{c})$, *("multiplication is associative")*
5. $(\overline{a} + \overline{b})\overline{c} = \overline{ac} + \overline{b}\overline{c}$, *("distributive")*
6. $\overline{a}\overline{1} = \overline{a} = \overline{1}\overline{a}$, *("$\overline{1}$ is a multiplicative identity")*
7. $\overline{a} + \overline{-a} = \overline{0}$,

8. $\overline{a}\overline{0} = \overline{0}$,

9. $\overline{a} + \overline{0} = \overline{a}$ *("$\overline{0}$ is a additive identity")*,

10. *if* $\overline{ac} = \overline{b}\overline{c}$ *and* $gcd(a, c) = 1$ *then* $\overline{a} = \overline{b}$ *("cancellation law")*,

11. *if* $gcd(a, n) = 1$ *then* $\overline{a}^{\phi(n)} = \overline{1}$ *("Euler's theorem")*,

12. $gcd(a, n) = 1$ *if and only if there is an element, denoted* $\overline{a}^{-1}$, *such that* $\overline{a} \cdot \overline{a}^{-1} = \overline{1}$,

13. *there is a solution* $\overline{x}$ *to* $\overline{a} \cdot \overline{x} = \overline{b}$ *if and only if* $gcd(a, n)|b$.

All these are restatements, in different notation, of results proven above. The verification of all these are left as an exercise.

In the special case when n is a prime p, we have the following facts.

Proposition 1.9.2. *Fix a prime modulus* $p > 1$. *For any integers* a, b, c,

1. $\overline{a} + \overline{b} = \overline{a + b} = \overline{b} + \overline{a}$, *("addition is commutative")*

2. $\overline{a}\overline{b} = \overline{ab} = \overline{b}\overline{a}$, *("multiplication is commutative")*

3. $(\overline{a} + \overline{b}) + \overline{c} = \overline{a} + (\overline{b} + \overline{c})$, *("addition is associative")*

4. $(\overline{a}\overline{b})\overline{c} = \overline{a}(\overline{b}\overline{c})$, *("multiplication is associative")*

5. $(\overline{a} + \overline{b})\overline{c} = \overline{ac} + \overline{b}\overline{c}$, *("distributive")*

6. $\overline{a}\overline{1} = \overline{a} = \overline{1}\overline{a}$, *("$\overline{1}$ is a multiplicative identity")*

7. $\overline{a} + \overline{-a} = \overline{0}$,

8. $\overline{a}\overline{0} = \overline{0}$,

9. $\overline{a} + \overline{0} = \overline{a}$ *("$\overline{0}$ is a additive identity")*,

10. *if* $\overline{ac} = \overline{b}\overline{c}$ *and* $\overline{c} \neq \overline{0}$ *then* $\overline{a} = \overline{b}$ *("cancellation law")*,

11. $\overline{a}^p = \overline{a}$ *("Fermat's little theorem")*,

12. *if* $\overline{a} \neq \overline{0}$ *then there is an element, denoted* $\overline{a}^{-1}$, *such that* $\overline{a} \cdot \overline{a}^{-1} = \overline{1}$.

1.10 Special project: continued fractions (optional)

A **finite regular continued fraction** is an expression of the form

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\ldots + \frac{1}{a_N}}}}, \tag{1.8}$$

where the $a_0, ..., a_N$ are integers (called the **partial quotients** of the continued fraction). This is also denoted

$$a_0 + \frac{1}{a_1+} \; \frac{1}{a_2+} \; \ldots \frac{1}{a_N},$$

or

$$[a_0, a_1, ..., a_N].$$

Expressing a real number x by its decimal expansion may seem very natural, but of course upon reflection it is an artifact of humans having 10 fingers. The expansion is clearly base-dependent, so that the integers that combine together to yield x would be quite different if we switched from base 10 to base 7. By contrast, note that a regular continued fraction for a real number x supplies an expansion of x in integers that is not base dependent. However strange it may appear on first glance, the continued expansion of x is in this sense more natural than the decimal expansion of x.

A continued fraction as above is a rational number, say $\frac{p_n}{q_n} = [a_0, a_1, ..., a_n]$. If $a_0, a_1, ...$ is a sequence of integers such that $\lim_{n\to\infty} \frac{p_n}{q_n} = x$, where x is a real number, then we say x has **regular (or simple) continued fraction** $[a_0, a_1, ...]$ with n^{th} **convergent** $\frac{p_n}{q_n} = [a_0, a_1, ..., a_n]$. Continued fractions occur frequently in other parts of mathematics and even in the analysis of feedback shift register sequences[9].

Example 1.10.1.

$$\frac{8}{5} = 1 + \frac{3}{5} = 1 + \frac{1}{5/3} = 1 + \frac{1}{1+2/3} = 1 + \frac{1}{1+\frac{1}{3/2}} = 1 + \frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{2}}}},$$

so $1 + \frac{1}{1+} \; \frac{1}{1+}\frac{1}{2} = [1, 1, 1, 2] = 8/5$.

[9] Though we shall not prove it here, it is known that each real number x has a continued fraction expansion and that the n^{th} convergents provide, in some sense, the best rational approximation to x. See chapter 10 of [HW] for example.

Let $[x]$ denote the greatest integer less than or equal to x, called the **integer part of** x. Given a rational number $r > 0$, the following two methods will find the continued fraction expansion.

step 1: Let $r_0 = r$, $a_0 = [r_0]$, $r_1 = r_0 - a_0$, and let $i = 1$.

step 2: Let $a_i = [r_i^{-1}]$, replace r_i by $r_i^{-1} - a_i$.

step 3: If $r_i \neq 0$ then replace i by $i + 1$ and go to step 2. If $r_i = 0$ then stop.

The following theorem says that this algorithm, for any rational number r, will terminate.

Theorem 1.10.2. *Each rational number x has a continued fraction expansion.*

The proof uses the Euclidean algorithm.

proof: We assume r is not an integer. By adding or subtracting a suitable integer, we may assume that $0 < r < 1$. Let $r = a/b$, where a, b are relatively prime integers and $b > 1$. Recall the steps used in the Euclidean algorithm:

Let $r_{-1} = a$ and $r_0 = b$. Let $a_0 = 0$, $i = 0$.

step 1: Use the division algorithm to compute integers q_{i+1} and $0 \leq r_{i+1} < r_i$ such that

$$r_{i-1} = r_i q_{i+1} + r_{i+1}, \qquad 0 \leq \frac{r_{i+1}}{r_i} < 1,$$

Let $a_{i+1} = q_{i+1}$.

step 2: If $r_{i+1} \neq 0$ then increment i and go to step 1. If $r_{i+1} = 0$ then stop.

Since $r_{-1} = a > r_0 = b > r_1 > ... \geq 0$, at some point the above algorithm must terminate.

□

Example 1.10.3. *(a)* $8/11 = [0, 1, 2, 1, 2]$,

(b) $101/100 = [1, 100]$,

(c) $11/5 = [2, 5]$.

Exercise 1.10.4. *Check that the continued fractions in the above example are true.*

Find the continued fraction expansions of

(a) $21/13$,

(b) $21/11$.

Exercise 1.10.5. *Find the continued fraction expansions of the golden ratio* $(1 + sqrt[5])/2$, *and make a conjecture as to what the nth convergents are.*

1.11 Number theory exercises using GAP

The GAP command to compute all the divisors of an integer n is `DivisorsInt(n);` For example, `DivisorsInt(15);` returns `[1,3,5,15]`.

Exercise 1.11.1. *Find all the positive divisors of* 1234567.

The remainder and quotient of n divided by m are the commands `RemInt( n, m );` `QuoInt( n, m );`, respectively. For example, `RemInt(155, 15 );` returns `5`.

Exercise 1.11.2. *Compute the remainder and quotient of dividing* $m = 789$ *into* $n = 123456$.

GAP does not have a base m conversion directly. One must use it's p-adic numbers package to convert from decimal to base m and this only works when $m = p$ is a prime. The command `fam:=PurePadicNumberFamily(p,k);` creates the data needed to work with base m notation, when $m = p$ is a prime. The integer k indicates the number of m-ary digits which will be used. The command `PadicNumber(fam,a);` returns the expansion of a in base $m = p$ out to k places. For example, `fam2:=PurePadicNumberFamily(2,6);` and `PadicNumber(fam2,14);` returns `0.111(2)` This is read least significant to most significant digits, left to right: the 0 is the 1's place, those to the right of the decimal point are the 2's place, 4's places, and so on. The 2 in the parentheses is the base $m = 2$, so the output tells us $14 = 0{\cdot}1{+}1{\cdot}2{+}1{\cdot}4{+}1{\cdot}8$.

Exercise 1.11.3. *Convert (the decimal)* 123456789 *to binary. Also, convert* 10001001011 *from binary to decimal.*

The greatest common divisor of m and n is given by `GcdInt( m, n);` . For example, `GcdInt( 108,801);` returns `9`.

Exercise 1.11.4. *Compute the gcd of* 123456789 *and* 987654321.

To find x, y such that $mx + ny = gcd(m, n)$, use the GAP command `Gcdex( m,n);`. For example, `Gcdex( 108,801);` returns
`rec( gcd := 9, coeff1 := -37, coeff2 := 5, coeff3 := 89, coeff4 := -12 )`,
where $x =$`coeff1`, $y =$`coeff2` .

Exercise 1.11.5. *Compute the gcd* d *of* 123456789 *and* 987654321 *and find* x, y *such that* $123456789x + 987654321y = d$.

To find the least common divisor of m and n, use the GAP command `LcmInt( m, n);` . For example, `LcmInt( 155,15);` returns `255` .

Exercise 1.11.6. *Compute the gcd and lcm of* 12345 *and* 6789.

The Euler totient function $\phi(n)$ is given by the command `Phi(n);`. For example, `Phi(2^{13}-1);` returns `8190`.

Exercise 1.11.7. *Compute* $\phi(2^{15}-1)$ *and* $\phi(2^{17}-1)$*, then deduce whether or not* $2^{15}-1$ *and* $2^{17}-1$ *are prime.*

The next prime after n is given by the command `NextPrimeInt(n);`. The n^{th} prime is given by the command `Primes(n);`. For example, `NextPrimeInt(100);` `Primes(100);` returns `101` and `541`, respectively.

The prime factorization of n is given by the command `FactorsInt(n);` . For example, `FactorsInt( 2^63 - 1 );` returns `[ 7, 7, 73, 127, 337, 92737, 649657 ]` .

Exercise 1.11.8. *Find the* 150^{th} *prime and test if* 1234567 *is prime by finding its factorization.*

Program the Lucas-Lehmer sequence $v_1 = 4$, $v_2 = 14$, ... $v_n = v_{n-1}^2 - 2$, by typing

```
lucaslehmer :=function(n)
 if n = 1 then return 4; fi;
 if 1 < n then  return lucaslehmer(n - 1)^2 - 2; fi;
 return 0;
end;
```

Exercise 1.11.9. *(a) Compute* $v_{22} \bmod (2^{23}-1)$*,* $v_{30} \bmod (2^{31}-1)$*, and* $v_{46} \bmod (2^{47}-1)$*. Determine if* $2^{23}-1$*,* $2^{31}-1$*, and* $2^{47}-1$ *are prime (using the test in Lemma 1.12.10).*

(b) What is the largest prime you can find on your computer using this test?

(c) Rewrite the code so that the procedure computes $v_1 = 4 \pmod q$ *and* $v_n = v_{n-1}^2 - 2 \pmod q$.

(d) Find the smallest n for which your computer takes more than 1 minute (by your watch) to compute v_n.

The Lucas-Perrin sequence

$$3, 0, 2, 3, 2, 5, 5, 7, 10, 12, 17, 22, 29, 39, 51, 68, 90, 119, 158, 209, 277, ...,$$

is defined by the recurrence $a_{n+3} = a_{n+1} + a_n$, $a_0 = 3$, $a_1 = 0$, $a_2 = 2$. This is also called the **Perrin sequence**, named after R. Perrin who mentioned this sequence in a paper in 1899, though it was found earlier Edouard Lucas in 1878, who gave the following result.

Lemma 1.11.10. *(Lucas-Perrin) If p is a prime then $p|a_p$.*

Program this sequence into GAP by typing

```
lucasperrin:=function(n)
 if n = 1 then return 0; fi;
 if n = 2 then return 2; fi;
 if n = 3 then return 3; fi;
 if n > 3 then return lucasperrin(n-2)+lucasperrin(n-3); fi;
 return -1;
end;
```

Exercise 1.11.11. *Compute a_i, for $1 \leq i \leq 11$. Verify the Lucas-Perrin lemma for $p = 2, 3, 5, 7, 11$.*

Exercise 1.11.12. *(a) Compute a_{47}.*

(b) What is the largest "prime candidate" you can find on your computer [10] *?*

Question: Is a_n is divisible by n if and only if n is prime?

If the answer was yes, this would be a very fast primality test. On the other hand, it is "yes" for all n up to 15000, as can be checked using GAP. (Exercise: Do this!) However, the answer is "no". Indeed, the smallest counterexample (discovered by D. Shanks and W. Adams [AS]) seems to be $271441 = 521^2$.

The Chinese remainder theorem commands give us the minimal solution $x \geq 0$ of $x \equiv a_1 \pmod{n_1}$, $x \equiv a_2 \pmod{n_2}$, ..., $x \equiv a_k \pmod{n_k}$.

The command for the Chinese remainder theorem is `ChineseRem([n1,n2,...,nk],[a1,a2,...,ak]);` . For example, `ChineseRem([5,7],[1,2]);` returns `16`.

[10] Assuming the converse to the Lucas-Perrin Lemma holds, the "prime candidate" would definitely be a prime

Exercise 1.11.13. *Solve* $x \equiv 2 \pmod 3$, $x \equiv 1 \pmod 4$, $x \equiv 3 \pmod 5$.

The multiplicative order of a mod m in GAP is given by `OrderMod( a, m );` . For example, `OrderMod(2,7);` returns `3`.

Exercise 1.11.14. *Find*
(a) the order of 10 *mod* 17,
(b) the order of 10 *mod* 101.

`PowerMod( r, e, m );` returns the e^{th} power of r modulo m.

Exercise 1.11.15. *Verify the example: Let* $p = 17$, $q = 19$, *so* $\phi(n) = \phi(323) = 288$. *Let* $e = 97$ *(the public exponent), so* $d = 193$ *(the private exponent). Let* $m = 100$ *be the message. The "cipher text" is* $c = 168$ *since* $m^e = 100^{97} \equiv 168 \pmod{323}$.

The primitive root mod m is given by `PrimitiveRootMod(m);` . The discrete log is given by `LogMod( a, b,m);` . For example, `PrimitiveRootMod(7);` and `LogMod( 64, 5,97);` returns `3` and `12`, respectively.

Exercise 1.11.16. *(a) Find the discrete log of* 11 *to base* 5 *mod* 97, *if it exists.*
(b) Is 197 *a prime? Is* 2 *a primitive root mod* 197*? Find the discrete log of* 91 *to base* 2 *mod* 197, *if it exists (ans:* 44*),*

1.12 Number theory exercises using MAGMA

The MAGMA command to compute all the divisors of an integer n is `Divisors(n);` For example, `Divisors(15);` returns `[1,3,5,15]`.

Exercise 1.12.1. *Find all the positive divisors of* 1234567.

The remainder and quotient of n divided by m is the MAGMA command `Quotrem(n,m);`. For example, `Quotrem( 11, 2);` returns `5 1`.

Exercise 1.12.2. *compute the remainder and quotient of dividing* $m = 789$ *into* $n = 123456$.

Exercise 1.12.3. *Compute the remainder and quotient of dividing* $m = 789$ *into* $n = 123456$.

MAGMA does not have a base m conversion directly. One must use it's p-adic numbers package to convert from decimal to base m and this only works when $m = p$ is a prime. The command `Zp<p> := pAdicRing(m);` creates the data needed to work with base m notation, when $m = p$ is a prime. The integer n indicates the number of m-ary digits which will be used. The commands `r := Zp!a; Coefficients(r);` returns the expansion of a in base $m = p$ out to k places. For example,
`Zp<p> := pAdicRing(5);r := Zp!127;, Coefficients(r);` returns
`[ 2, 0, 0, 1 ]` This is read least significant to most significant digits, left to right: the 2 is the 1's place, those to the right of the decimal point are the 5's place, 25's places, and so on. The output tells us $127 = 2 \cdot 1 + 0 \cdot 5 + 0 \cdot 25 + 1 \cdot 125$.

Exercise 1.12.4. *Convert (the decimal)* 123456789 *to binary. Also, convert* 10001001011 *from binary to decimal.*

The greatest common divisor of m and n is given by
`GreatestCommonDivisor(m, n);` . For example, `GreatestCommonDivisor(10005,50001);` returns `3`.

Exercise 1.12.5. *Compute the gcd of* 123456789 *and* 987654321.

To find x, y such that $mx + ny = gcd(m, n)$, use the GAP command `XGCD([ m,n]);`. For example, `XGCD([108,801]);` returns

Exercise 1.12.6. *Compute the gcd d of* 123456789 *and* 987654321 *and find* x, y *such that* $123456789x + 987654321y = d$.

To find the least common divisor of m and n, use the GAP command `LCM([ m, n]);` . For example, `LCM([ 155,15]);` returns `255` .

Exercise 1.12.7. *Compute the gcd and lcm of* 12345 *and* 6789.

The Euler totient function $\phi(n)$ is given by the command `EulerPhi( n);`.
For example, `EulerPhi(2^{13}-1);` returns `8190`.

Exercise 1.12.8. *Compute* $\phi(2^{15} - 1)$ *and* $\phi(2^{17} - 1)$, *then deduce whether or not* $2^{15} - 1$ *and* $2^{17} - 1$ *are prime.*

The next prime after n is given by the command `NextPrime(n);`. For example, `NextPrimeInt(100);` returns `101`.

The prime factorization of n is given by the command `Factorization(n);` . For example, `Factorization(100);` returns `[ <2, 2>, <5, 2> ]`.

Exercise 1.12.9. *Find the* 150^{th} *prime and test if* 1234567 *is prime by finding its factorization.*

Program the Lucas-Lehmer sequence $v_1 = 4$, $v_2 = 14$, ... $v_n = v_{n-1}^2 - 2$, by typing

```
function lucaslehmer(n)
 if n eq 1 then return 4; end if;
 if 1 lt n then  return lucaslehmer(n - 1)^2 - 2; end if;
 return 0;
end function;
```

We turn to a method to test the primality of special numbers of the form $2^p - 1$, where p is a prime. The next result is included to illustrate the type of special methods known to test the primality of integers of special forms.

Lemma 1.12.10. *(Lucas-D.H. Lehmer) For* $p > 2$, $q = 2^p - 1$ *is prime if* $q | v_{p-1}$, *where* $v_1 = 4$ *and* $v_n = v_{n-1}^2 - 2$.

The proof is omitted. See [Br].

Note that we need not and should not compute v_n directly, which grows very fast (in fact, it grows faster than $2^{1.9^n}$). For example, when testing for the primality of $q = 2^31 - 1$, we do not compute v_30 and see if $q | v_30$; rather we compute $v_30 \pmod q$ and see if it is zero. The point is that the new sequence

$$\overline{v}_1 = 4 \pmod q, \qquad \overline{v}_n = \overline{v}_{n-1}^2 - 2 \pmod q,$$

can be computed rather quickly on a computer, since the terms are always less than or equal to q. On the other hand, the sequence v_n is rather slow to compute (due to their size), when n is large.

Lucas-Lehmer test: Let $q = 2^p - 1$, where p is a prime. For $p > 2$, if $\overline{v}_{p-1} = 0 \pmod q$, then q is a prime.

Exercise 1.12.11. *(a) Compute* $v_{22} \bmod (2^{23} - 1)$, $v_{30} \bmod (2^{31} - 1)$, *and* $v_{46} \bmod (2^{47} - 1)$. *Determine if* $2^{23} - 1$, $2^{31} - 1$, *and* $2^{47} - 1$ *are prime (using the test in Lemma 1.12.10).*

(b) What is the largest prime you can find on your computer using this test?

(c) Rewrite the code so that the procedure computes $v_1 = 4 \pmod q$ *and* $v_n = v_{n-1}^2 - 2 \pmod q$.

(d) Find the smallest n *for which your computer takes more than 1 minute (by your watch) to compute* v_n.

The Lucas-Perrin sequence

$$3, 0, 2, 3, 2, 5, 5, 7, 10, 12, 17, 22, 29, 39, 51, 68, 90, 119, 158, 209, 277, ...,$$

is defined by the recurrence $a_{n+3} = a_{n+1} + a_n$, $a_0 = 3$, $a_1 = 0$, $a_2 = 2$. This is also called the **Perrin sequence**, named after R. Perrin who mentioned this sequence in a paper in 1899, though it was found earlier Edouard Lucas in 1878, who gave the following result.

Lemma 1.12.12. *(Lucas-Perrin) If p is a prime then $p|a_p$.*

Program this sequence into MAGMA by typing

```
function lucasperrin(n)
 if n eq 1 then return 0; end if;
 if n eq 2 then return 2; end if;
 if n eq 3 then return 3; end if;
 if 1 lt n then  return lucasperrin(n - 2) + lucasperrin(n - 3); end if;
 return -1;
end function;
```

Exercise 1.12.13. *Compute a_i, for $1 \leq i \leq 11$. Verify the Lucas-Perrin lemma for $p = 2, 3, 5, 7, 11$.*

Exercise 1.12.14. *(a) Compute a_{47}.*
(b) What is the largest "prime candidate" you can find on your computer [11] *?*

Question: Is a_n is divisible by n if and only if n is prime?

If the answer was yes, this would be a very fast primality test. On the other hand, it is "yes" for all n up to 15000, as can be checked using MAGMA. (Exercise: Do this!) However, the answer is "no". Indeed, the smallest counterexample (discovered by D. Shanks and W. Adams [AS]) seems to be $271441 = 521^2$.

The Chinese remainder theorem commands give us the minimal solution $x \geq 0$ of $x \equiv a_1 \pmod{n_1}$, $x \equiv a_2 \pmod{n_2}$, ..., $x \equiv a_k \pmod{n_k}$.

The command for the Chinese remainder theorem is `CRT([a1,a2,...,ak],[n1,n2,...,nk]);`. For example, `CRT([1,2],[5,7]);` returns `16` .

[11] Assuming the converse to the Lucas-Perrin Lemma holds, the "prime candidate" would definitely be a prime

Exercise 1.12.15. *Solve* $x \equiv 2 \pmod 3$, $x \equiv 1 \pmod 4$, $x \equiv 3 \pmod 5$.

The multiplicative order of a mod m in MAGMA is given by the commands `R := ResidueClassRing(m); Order(R!a);` . For example, `R := ResidueClassRing(17); Order(R!11);` returns `16`.

Exercise 1.12.16. *Find*
(a) the order of 10 *mod* 17,
(b) the order of 10 *mod* 101.

`R := ResidueClassRing(m); r:=R!a; r^e;` returns the e^{th} power of r modulo m.

Exercise 1.12.17. *Verify the example: Let* $p = 17$, $q = 19$, *so* $\phi(n) = \phi(323) = 288$. *Let* $e = 97$ *(the public exponent), so* $d = 193$ *(the private exponent). Let* $m = 100$ *be the message. The "cipher text" is* $c = 168$ *since* $m^e = 100^{97} \equiv 168 \pmod{323}$.

The primitive root mod m is given by `PrimitiveRoot(m);`. The discrete log is given by `Log(GF(m)!a,GF(m)!b);` (m prime). For example, `PrimitiveRootMod(7);` and `LogMod( 64, 5,97);` returns `3` and `12`, respectively.

Exercise 1.12.18. *(a) Find the discrete log of* 11 *to base* 5 *mod* 97, *if it exists.*
(b) Is 197 *a prime? Is* 2 *a primitive root mod* 197*? Find the discrete log of* 91 *to base* 2 *mod* 197, *if it exists (ans:* 44*),*

In MAGMA, type `ContinuedFraction(r);`, where r is real, to get the continued fraction of r. For example, `ContinuedFraction(Sqrt(2));` returns `[ 1, 2, 2, ...]`.

Exercise 1.12.19. *Find the continued fraction expansions of the golden ratio* $(1+\sqrt{5})/2$, *and make a conjecture as to what the nth convergents are.*

Chapter 2

Polynomials, rings and fields

In this chapter, we introduce two of the most common of all algebraic structures in mathematics, a "ring" and a "field".

These terms will be defined precisely in the next section so here we just say a few words of motivation. Roughly speaking, a "ring" is a set of elements having two operations, usually called addition and multiplication, which behaves in many ways like the integers. You can add and multiply elements in a ring but you can't (usually) divide them and get another element in the ring. Roughly speaking, a "field" is a set of elements having two operations, usually called addition and multiplication, which behaves in many ways like the rational numbers. You can add and multiply elements in a field and you can divide one element by any non-zero element to get another element in the field.

However, there exist rings and fields which, unlike the rationals, have only finitely many elements. Because they have only finitely many elements, they are of particular interest for applications to computer science and communication theory. However, their finiteness also forces them to have many properties which are not true for the integers or rational numbers.

Related to this is the study of polynomials. Polynomials not only help us to construct new and useful examples of rings and fields but are of interest in themselves. The "ring" of polynomials in one variable [1] has two binary operations - addition and multiplication. These operations have many properties similar to the integers and we can, and will, study analogs of many of the properties considered in chapter 1. In particular, we will study factoriza-

[1] Polynomials in several variables can be treated similarly and will also be discussed below.

tions of polynomials, irreducible polynomials (which are the analogs of prime integers), and modular arithmetic of polynomials (the polynomial analog of addition and multiplication modulo an integer m). The modular arithmetic of polynomials leads to other rings of particular interest to coding theorists, studied further in the next chapter.

We let $\mathbb{C}$ denote the complex numbers, $\mathbb{R}$ denote the real numbers, and let $\mathbb{Q}$ denote the rational numbers. Let $\mathbb{Z}$ denote the integers and let $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$, if p is a prime.

2.1 Fields: basic examples

Here is the simplest example, called the **Boolean field**: $F = \{0, 1\}$, with two binary operations, $+$ (addition mod 2), and $\cdot$ (integer multiplication). This field is used is the mathematics of electrical circuits (1 being "on" and 0 being "off").

In general, a field is a set F which has two binary operations, denoted $+$ and $\cdot$, satisfying the usual algebraic properties which the reals $\mathbb{R}$ or complexes $\mathbb{C}$ satisfy.

Definition 2.1.1. *A* **field** *is a set F which has two binary operations, denoted $+$ and $\cdot$, satisfying the following properties. For all $a, b, c \in F$, we have*

1. $a + b = b + a$, *("addition is commutative")*
2. $a \cdot b = b \cdot a$, *("multiplication is commutative")*
3. $(a + b) + c = a + (b + c)$, *("addition is associative")*
4. $(a \cdot b)c = a(b \cdot c)$, *("multiplication is associative")*
5. $(a + b) \cdot c = a \cdot c + b \cdot c$, *("distributive")*
6. *there is an element* $1 \in F$ *such that* $a \cdot 1 = a$, *("1 is a multiplicative identity")*
7. *there is an element* $0 \in F$ *such that* $a + 0 = a$ *("0 is a additive identity"),*
8. *if* $a \neq 0$ *then there is an element, denoted* a^{-1}, *such that* $a \cdot a^{-1} = 1$ *("the inverse of any non-zero element exists").*

Example 2.1.2. *We have seen several examples of fields already:* $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$, *and* $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$, *where* p *is a prime.*

A word of advertisement. Since the set $\mathbb{F}_p$ *is finite, we call it (at least when* p *is a prime number) a* **finite field**. *This field is often also denoted* $GF(p)$. *There are other finite fields besides* $\mathbb{F}_p$. *For each prime power* $q = p^k$ *there is a finite field, usually denoted* $\mathbb{F}_q$ *or* $GF(q)$. *In later subsections, we shall introduce methods of constructing these fields. A word of caution:* $\mathbb{Z}/n\mathbb{Z}$ *is never a field unless* n *is a prime (the proof of this is left as an exercise).*

If F *satisfies (1)-(7) but not necessarily (8) then* F *is called a* **ring** [2] [3] . *The subset of elements of a ring* F *whose inverse do exist is denoted by* $F^\times$. *If* F *satisfies (1),(3)-(8) but not necessarily (2) then* F *is called a* **skew field**.

If there is an integer $n > 0$ *such that , for all* $x \in F$, $n \cdot x = 0$ *then the smallest such integer is called the* **characteristic** *of* F. *If there is no such integer then we say that* F *is* **characteristic** 0.

Example 2.1.3. *The fields* $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ *all have characteristic* 0.

If E *and* F *are fields and*
(a) as sets, $F \subset E$,
(b) the field operations for F *are the restrictions of the field operations for* E,
then F *is called a* **subfield** *of* E *and* E *is called an* **extension field** *of* F, *written* E/F.

In particular, every field must contain at least two elements (namely, 0 and 1). The field containing exactly two elements is called the **Boolean field** and denoted $\mathbb{F}_2$ (or $GF(2)$). It is of characteristic 2.

Example 2.1.4. *(1)* $\mathbb{Q}$ *is a field.* $\mathbb{R}$ *is a field.* $\mathbb{C}$ *is a field.*
(2) If p *is a prime then* $\mathbb{F}_p$ *is a field.*

[2] More precisely, such an algebraic structure is called a "commutative ring with unit". All "rings" in this book shall be "commutative rings with unit".

[3] The following joke (told to one of us by Bill Wardlaw) illustrates this definition in an amusing way: Two young mathematics students, Alice and Bob, were out enjoying a picnic on a nice day in a field of clovers. They were not yet engaged to be married, so Alice put Bob's amorous advances on check when she said "Not until I have a ring!" Bob, knowing Alice was taking an abstract algebra course, countered "But we're in a field. Can't we take this field as our ring?"

(3) $\mathbb{Z}/n\mathbb{Z}$ is a ring. (It is not a field unless n is a prime.) For example, $\overline{2} \neq \overline{0}$ in $\mathbb{Z}/4\mathbb{Z}$ but $\overline{2}^{-1}$ does not exist. (If $\overline{2}^{-1}$ did exist, then since $\overline{2} \cdot \overline{2} = \overline{0}$, by the cancellation law, $\overline{2} = \overline{2} \cdot \overline{2} \cdot \overline{2}^{-1} = \overline{0} \cdot \overline{2}^{-1} = \overline{0}$, a contradiction.)

Lemma 2.1.5. *If F is a field with characteristic $n > 0$ then n is a prime number.*

proof: If not, $n = ab$ and $nx = abx = 0$ for all x, where $a < n$ and $b < n$. If $ax \neq 0$ for some $x \neq 0$ in F then (by the cancellation property) $ax \neq 0$ for all x. Conversely, if $ax = 0$ for some $x \neq 0$ in F then (by the cancellation property) $ax = 0$ for all x. These two facts allow us to conclude that either $ax = 0$ for all x in F or $bx = 0$ for all x in F. Since n was defined to be the smallest such integer, this is contradiction. □

A field F and let $p(x) = a_0 + a_1x + ... + a_{n-1}x^{n-1} + x^n$ be a polynomial having coefficients in F (so $a_i \in F$). Some fields, such as $F = \mathbb{C}$ have the property that all the roots of $p(x) = 0$ belong to $\mathbb{C}$. This fact is called the fundamental theorem of algebra.

Theorem 2.1.6. fundamental theorem of algebra *Let $F = \mathbb{C}$. Let $p(x) = a_0 + a_1x + ... + a_{n-1}x^{n-1} + x^n$ be any polynomial with coefficients in F (so $a_i \in F$). Then $p(x) = 0$ has exactly n roots in $\mathbb{C}$ (counted with multiplicity).*

Some fields do not have the property that every polynomial has all its roots in the field. For example, if $F = \mathbb{Q}$ and $p(x) = x^2 + 1$ then $p(x) = 0$ has no roots in F (though it has, by the above theorem, 2 roots in $\mathbb{C}$).

Definition 2.1.7. *A field F satisfying the property that every polynomial of degree n with coefficients in F has all its coefficients in F is called* **algebraically closed**.

For example, $\mathbb{C}$ is algebraically closed but $\mathbb{Q}$ is not. Though we shall not prove it here, every field F is contained in an algebraically closed field K. If K is an algebraically closed field containing F and there is no algebraically closed subfield of K containing F then we call K an **algebraic closure** of F, written $K = \overline{F}$.

If F is not algebraically closed then it is possible to construct extension fields of F. These constructions are explored in the next several sections.

2.1.1 Quadratic number fields

Let a be any integer which is not the perfect square of another integer. Let us abbreviate $\alpha = \sqrt{a}$ and define

$$\mathbb{Q}(\alpha) = \{x + \alpha y \mid x, y \in \mathbb{Q}\}.$$

This is called the **quadratic extension** of the rationals associated to α. It is an extension since it contains the field $\mathbb{Q}$ as a subset. It is called an **imaginary quadratic field** if $a < 0$ and a **real quadratic field** if $a > 0$.

For brevity, let $K = \mathbb{Q}(\alpha)$. Multiplication in K is given by

$$(x + \alpha y)(x + \alpha y) = xx' + ayy' + \alpha(xy' + x' + y),$$

for $x, x', y, y' \in \mathbb{Q}$. Inverse is given by

$$(x + \alpha y)^{-1} = \frac{x - \alpha y}{x^2 + ay^2}.$$

Since a is not a perfect square, $x^2 + ay^2 \neq 0$ for rational x, y unless $x = y = 0$.

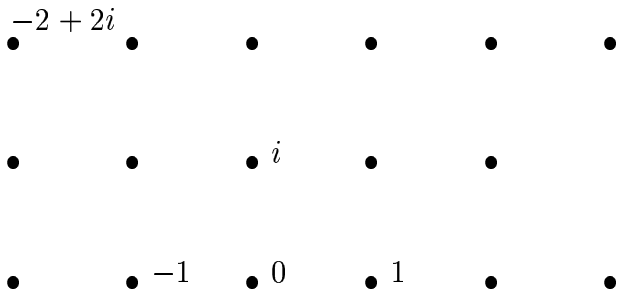
Gaussian integers

The field $\mathbb{Q}(\alpha)$ introduced above may be thought of as being similar to $\mathbb{Q}$ in many respects. What is the analog (if any) of the ring of integers, $\mathbb{Z} \subset \mathbb{Q}$? This subsection addresses this question.

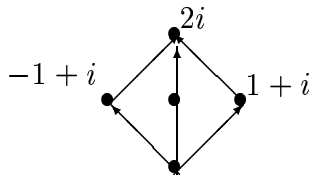
Let $i = \sqrt{-1}$. The **Gaussian number field** is the special quadratic number field $\mathbb{Q}(i)$. The **Gaussian integers** is the ring

$$\mathbb{Z}[i] = \{a + ib \mid a, b \in \mathbb{Z}\} = \mathbb{Z} + i\mathbb{Z}.$$

These "integers" can be pictured as an lattice in the complex plane:



Geometrically, addition of Gaussian integers may be visualized using the parallelogram law: if you represent the integer $a + ib$ by the vector $\vec{v}$ from $(0, 0)$ to (a, b) and the integer $c + id$ by the vector $\vec{w}$ from $(0, 0)$ to (c, d) then the sum $a + b + i(c + d)$ is represented by the vector from $(0, 0)$ to the other endpoint of the diagonal of the parallelogram spanned by $\vec{v}$ and $\vec{w}$. For example, to add $1 + i$ and $-1 + 1$, we have the following sketch



Definition 2.1.8. *An element $p = a + ib \neq 0$ is a* **prime (element of $\mathbb{Z}[i]$)** *if it has the following property: for all $r, s \in \mathbb{Z}[i]$, $p|rs$ then $p|r$ or $p|s$.*

The following characterization of Gausssian primes is known.

Theorem 2.1.9. *An element $p = a + ib \neq 0$ is a prime element of $\mathbb{Z}[i]$ if and only if either*

(a) $ab = 0$ and $|a + ib|$ is a prime integer which is $\equiv 3 \pmod 4$, or

(b) $ab \neq 0$ and $a^2 + b^2$ is a prime integer which is $\equiv 1 \pmod 4$.

The proof of this result goes beyond the scope of this book. See [HW] for a proof.

There is an analog of the fundamental theorem of arithmetic for the ring of Gaussian integers. In other words, each Gaussian integer can be uniquely (up to order and factors of the form ± 1 and $\pm i$) factored into a product of Gaussian primes.

Remark 2.1.10. *If you ask for a general analog of the fundamental theorem of arithmetic for the ring of integers of one of the number fields $\mathbb{Q}(\sqrt{d})$ then less is known. It was conjectured by Gauss (and still unproven today) that there are infinitely squarefree $d > 0$ for which ring of integers of $\mathbb{Q}(\sqrt{d})$ satisfies the unique factorization theorem. It was proven by H. Stark in 1967 [St] (and independently by A. Baker) that the only cases for which ring of integers of $\mathbb{Q}(\sqrt{d})$, $d < 0$ squarefree, satisfies the unique factorization theorem is when $d = -1, -2, -3, -7, -11, -19, -43, -67, -163$. See [HW], page 217, for further details.*

The integers of $\mathbb{Q}(\sqrt{2})$

Let $\alpha = \sqrt{2}$. The **quadratic number field** $\mathbb{Q}(\sqrt{2})$ is the field

$$\mathbb{Q}(\alpha) = \{x + \alpha y \mid x, y \in \mathbb{Q}\},$$

which we denote by K for short. The **integers of** K is the ring

$$\mathbb{Z}[\alpha] = \{a + \alpha b \mid a, b \in \mathbb{Z}\} = \mathbb{Z} + \alpha\mathbb{Z}.$$

Definition 2.1.11. *An element $p = a + \alpha b \neq 0$ is a* **prime (element of** $\mathbb{Z}[\alpha]$**)** *if it has the following property: for all $r, s \in \mathbb{Z}[\alpha]$, $p|rs$ then $p|r$ or $p|s$.*

Is there an analog of Theorem 2.1.9 for primes in this field?

Yes. Before describing them, we need to introduce the notion of an associate.

Definition 2.1.12. *If R is an commutative ring with unit 1 and $u \in R$ has the property that $uv = 1$, for some $v \in R$ then we call u a* **unit of** *R and call v the* **inverse of** *u. . The set of units of R is denoted $R^\times$.*

Let $x, y \in R$. We say x is an **associate** *of y if $x = uy$, for some $u \in R^\times$. The set of associates of x is denoted $xR^\times$.*

There is an analog of the fundmental theorem of arithmetic for $\mathbb{Z}[\alpha]$. In other words, each element in $\mathbb{Z}[\alpha]$ can be uniquely (up to order and "unit" factors of the form $\pm(1 + \sqrt{2})^k$, $k \in \mathbb{Z}$) factored into a product of prime elements. In other words, $\mathbb{Z}[\sqrt{2}]$ is a unique factorization domain.

Example 2.1.13. *(a) If $R = \mathbb{Z}$ then $R^\times = \{1, -1\}$.*

(b) If $R = \mathbb{Z}[i]$ then $R^\times = \{1, -1, i, -i\}$.

(c) if $R = \mathbb{Z}[\alpha]$ then

$$R^\times = \{\pm(1 + \sqrt{2})^k \mid k \in \mathbb{Z}\}.$$

The proof of this goes beyond the scope of this book. See for example, Hardy and Wright, [HW], §14.5. (See the exercises below for further examples.)

The following characterization of primes is known.

Theorem 2.1.14. *An element $a + \alpha b \neq 0$ is a prime element of $\mathbb{Z}[\alpha]$ if and only if either*

(a) $a = 0$ and $b = \pm 1$,

(b) $b = 0$ and $|a|$ is a prime integer which is $\equiv \pm 3 \pmod 8$, or

(c) $ab \neq 0$ and $a^2 - 2b^2$ is a prime integer which is $\equiv 1 \pmod 8$,

(d) $a + \alpha b$ is an associate of an element as in (a), (b) or (c).

Remark 2.1.15. *If we replace $\alpha = \sqrt{2}$ by the root of some higher degree polynomial in $\mathbb{Z}[x]$ then there is, in general, no analog of Theorem 2.1.14 for $\mathbb{Q}(\alpha)$.*

In fact, the search for an analog is one of the motivations for the very deep Langlands philosophy [Art], §1.

If we replace $\alpha = \sqrt{2}$ by the root of some higher degree polynomial in $\mathbb{Z}[x]$ then there is, in general, no analog of the fundamental theorem of arithmetic. The fundmental theorem of arithmetic fails for the integers in $\mathbb{Q}(\sqrt{-5})$ and for the integers in $\mathbb{Q}(\sqrt{10})$, for example. (See Hardy and Wright, §§14.6 for more details.)

2.1.2 A construction of finite fields

We next turn to the finite field analog of the above section. One advantage we had when constructing $\mathbb{Q}(\sqrt{2})$ (for example), was that "$\sqrt{2}$" made sense - it was an element of the larger field $\mathbb{R}$ which contains $\mathbb{Q}$.

Example 2.1.16. *Suppose we want to construct the analog of $\mathbb{Q}(\sqrt{2})$ but with $\mathbb{Q}$ replaced by $\mathbb{F}_3$. First, we need to check that 2 is not a perfect square in $\mathbb{F}_3$ (it isn't but the reader should check this). Second, and more important perhaps, we need to define $\sqrt{2}$.*

Let F be a vector space of dimension 2 over $\mathbb{F}_3$ with vector space basis

$$B = \{e_1 = 1, e_2 = \sqrt{2}\},$$

where $\sqrt{2}$ is a formal symbol for some element which satisfies $e_2^2 = 2 \in \mathbb{F}_3$ and commutes with all the elements of $\mathbb{F}_3$. (We shall see later, thanks to a Kronecker's Theorem 2.6.11, that this makes sense.) In other words, each element of F is of the form

$$x_1 e_1 + x_2 e_2, \qquad x_i \in \mathbb{F}_3.$$

Note

$$(x_1 e_1 + x_2 e_2)(y_1 e_1 + y_2 e_2) = (x_1 y_1 + 2x_2 y_2)e_1 + (x_2 y_1 + x_1 y_2)e_2.$$

This implies

$$(x_1 e_1 + x_2 e_2)^{-1} = (x_1 e_1 - x_2 e_2)/(x_1^2 - 2x_2^2).$$

The field axioms hold for F (the reader should check this), so F is a field of degree 2 over $\mathbb{F}_3$.

Example 2.1.17. *Let $p = 5$, so $\mathbb{F}_5 = \{0, 1, 2, 3, 4\}$ (with addition and multitplication mod 5). The set of squares is given by*

$$\{x^2 \mid x \in \mathbb{F}_5\} = \{0, 1, 4\}.$$

In particular, $2, 3$ are not squares in this field. Let $e_2 = \sqrt{2}$ be a formal symbol for some element which satisfies $e_2^2 = 2$. This is a root of the polynomial $x^2 - 2 = 0$.

The vector space F over $\mathbb{F}_5$ with basis $\{e_1 = 1, e_2\}$ is 2-dimensional over $\mathbb{F}_5$. Two elements $x_1e_1 + x_2e_2 = x_1 + x_2\sqrt{2}$ and $y_1e_1 + y_2e_2 = y_1 + y_2\sqrt{2}$ are multiplied by the rule

$$(x_1 + x_2\sqrt{2}) \cdot (y_1 + y_2\sqrt{2}) = x_1y_1 + 2x_2y_2 + (x_1y_2 + y_1x_2)\sqrt{2}.$$

It is a degree 2 field extension of $\mathbb{F}_5$.

The construction used in the above example may be summarized more generally as follows:

1. Pick an element $m \in \mathbb{F}_p$ which is not the square of another element, if such an element exists.

2. Let $e_1 = 1$ and $e_2 = \sqrt{m}$ be a formal symbol for some element which satisfies $e_2^2 = m$.

3. As a set, let $F = \{xe_1 + ye_2 \mid x, y \in \mathbb{F}_p\}$. To define F as a field, let $+$ be "componentwise addition" mod p and let $\cdot$ be defined by

$$(x_1 + x_2\sqrt{m}) \cdot (y_1 + y_2\sqrt{m}) = x_1y_1 + mx_2y_2 + (x_1y_2 + y_1x_2)\sqrt{m}.$$

A finite field F constructed in this way is called a **quadratic extension** of $\mathbb{F}_p$. A finite field F constructed in this way has p^2 elements.

Remark 2.1.18. *What if every element of $\mathbb{F}_p$ is the square of another element? (This can happen, for example, when $p = 2$.) In this case, the above construction does not apply. However, other constructions do work. Details will be given in the following section and in §2.6.*

2.1.3 Matrix constructions of finite fields

There are finite fields other than those fields $\mathbb{F}_p$, p prime, already introduced. This section will introduce some explicit examples. Later in this chapter, we will see how they are constructed more generally.

Example 2.1.19. *In this example, we construct a field having* 4 *elements. Let*

$$F = \{\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}\},$$

where each matrix entry is considered as an element of $\mathbb{F}_2$*, then* F *is a field under the operations of matrix addition and multiplication. It has characteristic* 2.

Example 2.1.20. *If we let*

$$F = \{\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix},$$
$$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix},$$
$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}\},$$

where each matrix entry is considered as an element of $\mathbb{F}_2$*, then* F *is a field. It has characteristic* 2.

Example 2.1.21. *Let*

$$F = \{\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}, \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix}, \begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix},$$
$$\begin{pmatrix} 0 & 4 \\ 1 & 4 \end{pmatrix}, \begin{pmatrix} 1 & 4 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 2 & 4 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 3 & 4 \\ 1 & 2 \end{pmatrix}, \begin{pmatrix} 4 & 4 \\ 1 & 3 \end{pmatrix},$$
$$\begin{pmatrix} 0 & 3 \\ 2 & 3 \end{pmatrix}, \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}, \begin{pmatrix} 2 & 3 \\ 2 & 0 \end{pmatrix}, \begin{pmatrix} 3 & 3 \\ 2 & 1 \end{pmatrix}, \begin{pmatrix} 4 & 3 \\ 2 & 2 \end{pmatrix},$$
$$\begin{pmatrix} 0 & 2 \\ 3 & 2 \end{pmatrix}, \begin{pmatrix} 1 & 2 \\ 3 & 3 \end{pmatrix}, \begin{pmatrix} 2 & 2 \\ 3 & 4 \end{pmatrix}, \begin{pmatrix} 3 & 2 \\ 3 & 0 \end{pmatrix}, \begin{pmatrix} 4 & 2 \\ 3 & 1 \end{pmatrix},$$
$$\begin{pmatrix} 0 & 1 \\ 4 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 4 & 2 \end{pmatrix}, \begin{pmatrix} 2 & 1 \\ 4 & 3 \end{pmatrix}, \begin{pmatrix} 3 & 1 \\ 4 & 4 \end{pmatrix}, \begin{pmatrix} 4 & 1 \\ 4 & 0 \end{pmatrix}\}.$$

This is a field with 25 *elements. It has characteristic* 5.

We shall later explain (see §2.7) how these come about in general.

Exercise 2.1.22. *Verify the following factorizations in* $\mathbb{Z}[i]$.
(a) $5 = (2+i)(2-i) = (1+2i)(1-2i)$,
(b) $13 = (2+3i)(2-3i) = (3+2i)(3-2i)$,
(c) $17 = (4+i)(4-i) = (1+4i)(1-4i)$.
Are there integers a, b, *not both equal to zero, such that* $7 = (a+bi)(a-bi)$*?*

Exercise 2.1.23. *Show that the field* $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ *has characteristic* p. *Here* p *is a prime.*

Exercise 2.1.24. *Write* $N(a+ib) = a^2 + b^2$. *If* $r, s \in \mathbb{Z}[i]$, *show* $N(rs) = N(r)N(s)$.

Exercise 2.1.25. *(a) Find all prime elements* $p = a+ib$ *of* $\mathbb{Z}[i]$ *with* $a^2+b^2 \leq 25$.
(b) Plot these primes $a+ib$ *as the points* (a, b) *on the* (x, y)*-plane.*

Exercise 2.1.26. *Write the addition and multiplication tables for the set* F *in Example 2.1.19. Using these tables, check that* F *is a field of characteristic* 2.

Exercise 2.1.27. *Check that* $\mathbb{Q}(\sqrt{2})$ *is a field.*

Exercise 2.1.28. *(Assumes linear algebra [4]) If E is, as an F-vector space, finite dimensional with dimension d then we say that the* **degree** *of E over F is d.*

Check that $\mathbb{Q}(\sqrt{2})$ is degree 2 over $\mathbb{Q}$.

Exercise 2.1.29. *Suppose $a = b \cdot c^2$, where $b, c \in \mathbb{Z}$. Show $\mathbb{Q}(\sqrt{a}) = \mathbb{Q}(\sqrt{b})$.*

Exercise 2.1.30. *Let F be the field constructed in Example 2.1.16, with addition and multiplication mod 3. Compute*

(a) $(1+\sqrt{2})^{-1}$ (in terms of the basis e_1, e_2),
(b) $(1-3\sqrt{2})/(1+\sqrt{2})$ (in terms of the basis e_1, e_2),
(c) $(1-3\sqrt{2})^{-1} \cdot (1+\sqrt{2})$ (in terms of the basis e_1, e_2).

Exercise 2.1.31. *Let F be the field constructed in Example 2.1.16, with addition and multiplication mod 3. Find the multiplication table for the group $(F^\times, \cdot)$ and the addition table for the group $(F, +)$. Check that F is a field.*

Exercise 2.1.32. *Check that the set F in Example 2.1.20 is a field of characteristic 2.*

Exercise 2.1.33. *Consider the field F in Example 2.1.21. Find a matrix $A \in F$ such that*

$$\{A^i \mid 0 \leq i \leq 5\}$$

is the set of all elements in F except for the 0 matrix.

Exercise 2.1.34. *Check that the set F in Example 2.1.21 is a field of characteristic 5.*

Exercise 2.1.35. *Consider the field F in Example 2.1.21. Let $A = \begin{pmatrix} 0 & 1 \\ 4 & 1 \end{pmatrix}$. Show that*

$$\{A^i \mid 0 \leq i \leq 23\}$$

is the set of all elements in F except for the 0 matrix.

Exercise 2.1.36. *Try to construct a field of characteristic 3 having 9 elements, by modifying Example 2.1.21.*

[4] For those who have not had a course in linear algebra, vectors spaces are defined in the next chapter.

Exercise 2.1.37. *Complete the the addition and multiplication tables for* $\mathbb{F}_2 = \{0, 1\}$*:*

$+$	0	1
0		
1		

$\cdot$	0	1
0		
1		

Check that $\mathbb{F}_2$ *is a field (check all the axioms).*

Exercise 2.1.38. *Verify the following in* $\mathbb{Z}[\sqrt{2}]$*.*
(a) $2 = (2 - \sqrt{2})(2 + \sqrt{2}) = \sqrt{2}\sqrt{2}$*,*
(b) $-1 = (1 - \sqrt{2})(1 + \sqrt{2})$*,*
(c) $(1 + \sqrt{2})^{-1} = -1 + \sqrt{2}$*.*

Exercise 2.1.39. *Write* $(1+\sqrt{2})^{-5}$ *in the form* $m+\sqrt{2}n$*, for some* $m, n \in \mathbb{Z}$*.*

Exercise 2.1.40. *(a) Write down the* 10 *elements in*

$$\{m + n\sqrt{2} \mid m, n \in \mathbb{Z}\}$$

closest to 0*. Plot them on the real number line.*
(b) Write down the 10 *elements in*

$$I = \{2m + n\sqrt{2} \mid m, n \in \mathbb{Z}\}$$

closest to 0*. Plot them on the real number line. Can you find an* $r \in \mathbb{Z}[\sqrt{2}]$ *such that* $I = r\mathbb{Z}[\sqrt{2}]$*?*

Exercise 2.1.41. *Write* $N(a + \sqrt{2}b) = a^2 - 2b^2$*. If* $r, s \in \mathbb{Z}[\sqrt{2}]$*, show* $N(rs) = N(r)N(s)$*.*

Exercise 2.1.42. *(a) Find all prime elements* $p = a + \sqrt{2}b$ *of* $\mathbb{Z}[\sqrt{2}]$ *with* $a^2 + b^2 \leq 25$*.*
(b) Plot these primes $a + \sqrt{2}b$ *as the points* (a, b) *on the* (x, y)*-plane.*

Exercise 2.1.43. *Let $F = \mathbb{Z}/3\mathbb{Z}$. (a) Write the addition table of F.*
(b) Write the multiplication table of $F^\times = F - \{0\}$.
Using these tables, show that
(c) F is a field,
(d) F has characteristic 3.

Exercise 2.1.44. *Let $F = \mathbb{Z}/5\mathbb{Z}$. (a) Write the addition table of F.*
(b) Write the multiplication table of $F^\times = F - \{0\}$.
Using these tables, show that
(c) F is a field,
(d) F has characteristic 5.

Exercise 2.1.45. *Show that $\mathbb{F}_p$ is a field, if p is a prime. (Hint: Use Proposition 1.9.2.)*

Exercise 2.1.46. *Show that $\mathbb{Z}/n\mathbb{Z}$ is not a field, if n is not a prime. (Hint: Use Proposition 1.9.1.)*

The following exercise refers to Exercise 2.1.28.

Exercise 2.1.47.
1. *Show that $\mathbb{C}$ is an extension field of $\mathbb{R}$. Is $\mathbb{C}$ a finite dimensional vector space over $\mathbb{R}$? (In other words, is the degree finite?) If so, find its degree.*

2. *Show that $\mathbb{R}$ is an extension field of $\mathbb{Q}$. Is the degree of $\mathbb{R}/\mathbb{Q}$ finite (i.e., is $\mathbb{R}$ a finite dimensional vector space over $\mathbb{Q}$)? If so, find its degree.*

Exercise 2.1.48. *Let S be a finite non-empty set. Let F denote the set of all subsets of S. Let set-theoretic intersection $\cap$ denote "multiplication" and let set-theoretic union $\cup$ denote "addition". Show that F is a ring with these operations.*

Exercise 2.1.49. *(a) Show that $a + (-a) = 0$, $a \cdot 0 = 0$, for any a belonging to a field F.*
(b) Show that the "cancellation law" holds: if $ac = bc$ and $c \neq 0$ then $a = b$.

2.2 Polynomials

A **polynomial** over a ring [5] F is an expression of the form

$$p(x) = a_0 + a_1x + ... + a_{n-1}x^{n-1} + a_nx^n,$$

where x is an independent variable and $a_0, a_1, ..., a_n \in F$. We sometimes say that p is a polynomial "over" F. If $a_n \neq 0$ then n is called the **degree** and we write $\deg(p) = n$ and call a_n the **leading coefficient**. A polynomial with leading coefficient equal to 1 is called a **monic** polynomial. The collection of all such polynomials is denoted by $F[x]$.

Let's slow down and think about this for a second. How is this expression for $p(x)$ rigorously defined? For example, we could assume that the possible values of x range over F, so the expression $p(x)$ is well-defined. Unfortunately, if F were a finite field (or finite ring) then the set of values of $p(x)$, for $x \in F$, does not uniquely determine the coefficients a_i of p. (See Exercise 2.2.20 below.) This is not what we we want, since we want to say that two polynomials are equal if and only if the coefficients are all equal. Since we might be in a situation where F is finite, we cannot define a polynomial in terms of its values.

Instead, we simply assume that a polynomial is a *formal* expression where x^i is a "place-marker" for its coefficients a_i.

A sequence $\{a_i\}_{i=0}^{\infty}$ of elements of F will be called **finite** if there are only finitely many non-zero terms in the sequence. Let $\mathcal{S}_F$ denote the set of all such finite sequences. Note that there is a map

$$C : F[x] \to \mathcal{S}_F$$

given by sending $p(x) = a_0 + a_1x + ... + a_{n-1}x^{n-1} + a_nx^n$ to the sequence $\{a_0, a_1, ..., a_n, 0, 0, ...\}$.

We define the addition and multiplication of two such formal polynomial expressions in the usual way [6]: addition is "coefficient-wise",

$$(a_0 + a_1x + ... + a_nx^n) + (b_0 + b_1x + ... + b_nx^n) = \\ a_0 + b_0 + (a_1 + b_1)x + ... + (a_n + b_n)x^n,$$

[5] Recall that a ring similar to a field except that a non-zero element need not have an inverse.

[6] With these binary operations on $F[x]$, it turns out that $F[x]$ is a ring in the sense of the definition above.

and multiplication is defined by

$$(a_0 + a_1x + ...) \cdot +(b_0 + b_1x + ...) = \\ c_0 + c_1x + ...,$$

where $c_n = \sum_{i=0}^{n} a_i b_{n-i}$. With this definition, two polynomials are equal if and only if the coefficients are all equal.

This contrast and distinction of formal expressions versus the idea of a polynomial being an F-valued function is a moot point for polynomials with coefficients in $\mathbb{R}$ or $\mathbb{C}$ (or in any infinite field, for that matter), but it is important for polynomials with coefficients in a finite field because of the phenomenon illustrated in the following example.

Let F be any field. First, we observe that any finite set of points in the "plane" F^2 may be "interpolated" using a polynomial (which depends on these points and is of of suitably high degree). The precise statement is the following result.

Theorem 2.2.1. *(Lagrange interpolation formula) Let $x_1, x_2, ..., x_n \in F$ be n distinct elements and let $f : F \to F$ be any function. Then there is a polynomial function $p(x)$ of degree n such that $p(x_i) = f(x_i)$, for $1 \leq i \leq n$. In fact,*

$$p(x) = \frac{\prod_{1 \leq i \leq n,\ i \neq 1}(x - x_i)}{\prod_{1 \leq i \leq n,\ i \neq 1}(x_1 - x_i)} f(x_1) + \frac{\prod_{1 \leq i \leq n,\ i \neq 2}(x - x_i)}{\prod_{1 \leq i \leq n,\ i \neq 2}(x_2 - x_i)} f(x_2) + ... \\ + \frac{\prod_{1 < i < n,\ i \neq n}(x - x_i)}{\prod_{1 \leq i \leq n,\ i \neq n}(x_n - x_i)} f(x_n).$$

proof: "By inspection." $\square$

Example 2.2.2. *We will see that if F is a finite field, every function $f : F \to F$ may be written down as a polynomial with coefficients in F using the Lagrange interpolation formula. Let F have elements $F = \{x_1, x_2, ..., x_n\}$ and let $f : F \to F$ be any function. Let $p(x)$ be as in the Lagrange interpolation formula. Then $f(x) = p(x)$ for all $x \in F$. In other words, over a finite field, any function many be represented as a polynomial function. Furthermore, this representation is not unique.*

This is why it is important not to assume that "x" is a "variable with values in F", as is commonly done in calculus. It is better to think of the powers x^i of a polynomial over a finite field as "placeholders" for the coefficients [7].

[7] Alternatively, one could think of polynomials $p(x)$ over a finite field F as really being functions on an "algebraic closure" $\overline{F}$ (a certain infinite field containing F) which has been restricted to F

2.2.1 $F[x]$ is a ring

The properties of addition and multiplication are summarized in the following proposition.

Proposition 2.2.3. *Let F be a field or one of the rings $\mathbb{Z}$ or $\mathbb{Z}/m\mathbb{Z}$. For any polynomials $a(x), b(x), c(x) \in F[x]$,*

1. $a(x) + b(x) = b(x) + a(x)$, *("addition is commutative")*
2. $a(x)b(x) = b(x)a(x)$, *("multiplication is commutative")*
3. $(a(x) + b(x)) + c(x) = a(x) + (b(x) + c(x))$, *("addition is associative")*
4. $(a(x)b(x))c(x) = a(x)(b(x)c(x))$, *("multiplication is associative")*
5. $(a(x) + b(x))c(x) = a(x)c(x) + b(x)c(x)$, *("distributive")*
6. $a(x) \cdot 1 = a(x)$, *("1 is a multiplicative identity")*
7. $a(x) + (-a(x)) = 0$,
8. $a(x) \cdot 0 = 0$,
9. $a(x) + 0 = a(x)$ *("0 is a additive identity"),*
10. *if $a(x)c(x) = b(x)c(x)$ and $c(x) \neq 0$ then $a(x) = b(x)$* *("cancellation law").*

In other words, $F[x]$ is a ring.

2.2.2 Roots

Let F be a field or even the ring $\mathbb{Z}/n\mathbb{Z}$. A **root** of a polynomial $p(x)$ in $F[x]$, is an element $r \in F$ such that $p(x) = 0$ in F.

As a "non-example", we have the following fact which goes back several thousand years ago/

Lemma 2.2.4. *$p(x) = x^2 - 2$ has no roots in $F = \mathbb{Q}$.*

proof: (Euclid) Suppose not. Let $r = a/b$ be a root with a, b integers satisfying $gcd(a, b) = 1$. Since $a^2/b^2 = 2$, we have $a^2 = 2b^2$. In particular, a must be even, say $a = 2c$. Then $4c^2 = 2b^2$, so $2c^2 = b^2$. This implies b is even, contradicting out assumption that $gcd(a, b) = 1$. $\square$

Two facts are in sharp contrast to what we are used to for real-valued and complex-valued polynomials:

Fact 1 If F is not a field then it is possible for a polynomial of degree n in $F[x]$ to have more than n roots.

Fact 2 If F is a finite field then it is possible for a (non-zero) polynomial of degree n in $F[x]$ to be zero for every $x \in F$ (i.e., be identically zero on F).

For further examples, see the exercises.

2.2.3 The division algorithm

Let $f(x)$ and $g(x)$ be polynomials in $F[x]$. We say that $g(x)$ **divides** $f(x)$, denoted $g(x)|f(x)$, if there is a polynomial $h(x)$ in $F[x]$ such that $f(x) = g(x)h(x)$. In this case, we call $g(x), h(x)$ **factors** of $f(x)$.

To determine if one polynomial divides another, one use the division algorithm for polynomials. The algorithm below is the analog of the division algorithm for the integers which was discussed in the previous chapter.

Lemma 2.2.5. *(Division algorithm, special case) Let F be $\mathbb{C}$ or $\mathbb{Z}/n\mathbb{Z}$. If $p(x)$ is a polynomial in $F[x]$ and $c \in F$ then then there is a polynomial $q(x)$ of degree $deg(p) - 1$ in $F[x]$ (depending on $p(x)$ and on c) such that*

$$p(x) = (x - c)q(x) + p(c).$$

proof: Suppose $p(x) = a_0 + a_1x + ... + a_{n-1}x^{n-1} + a_nx^n$. Note that

$$\begin{gathered} p(x) - p(c) \\ = (a_0 + a_1x + ... + a_{n-1}x^{n-1} + a_nx^n) - (a_0 + a_1c + ... + a_{n-1}c^{n-1} + a_nc^n) \\ = a_1(x - c) + a_2(x^2 - c^2) + ... + a_n(x^n - c^n) \\ = (x - c)(a_1 + a_2(x + c) + ... + a_n \textstyle\sum_{j=0}^{n-1} x^{n-1-j}c^j). \end{gathered}$$

$\square$

Example 2.2.6. *If $F = \mathbb{F}_p$ then for any non-zero $a \in F$, we have $a^p = a$ by Fermat's little theorem. Moreover, we have*

$$(x + a)^p = x^p + a^p = x^p + a,$$

since p divides all the binomial coefficients in the expansion of $(x+y)^p$ except those at the "ends" (see Example 1.5.1).

Theorem 2.2.7. *Let F be a field and $p \in F[x]$ be a polynomial of degree $n > 0$. Then there are at most n roots of $p(x) = 0$ in F, counted according to multiplicity.*

proof: The proof is by mathematical induction.

Induction hypothesis: Any polynomial $f \in F[x]$ is degree $k > 0$ has at most k roots.

Case $n = 1$: If $p(x) = a_1x + a_0$ is of degree 1 then (since F is a field) $r = -a_0/a_1$ is a root. This proves the theorem in this case.

Case $n > 1$: Assume the induction hypothesis holds for all $k < n$. If $p(x)$ is a polynomial of degree n then either $p(x)$ has no root (and we are done) or there is a $c \in F$ such that $p(c) = 0$. By the division algorithm, $p(x) = (x - c)q(x)$, where $q(x)$ is a polynomial of degree $n - 1$. By the induction hypothesis, $q(x)$ has at most $n - 1$ roots, so $p(x)$ has at most n roots. □

Example 2.2.8. *Let $p(x) = x^{p-1} - 1$. By Fermat's little theorem, the numbers $1, 2, ..., p - 1$ are distinct roots of $p(x)$. By the above theorem, there can be no others. By the division algorithm, we have*

$$x^{p-1} - 1 = (x - 1)...(x - p + 1). \tag{2.1}$$

Plugging $x = 0$ into (2.1), implies the following result.

Corollary 2.2.9. *(Wilson's theorem)* $(p - 1)! \equiv -1 (\mathrm{mod}\, p)$.

Example 2.2.10. *If $p = 7$ then $(p-1)! = 720$ and $(p-1)!+1 = 721 = 7 \cdot 103$, so $720 \equiv -1 (\mathrm{mod}\, 7)$.*

2.2.4 The Euclidean algorithm

Just as in the case of the integers, there is a Euclidean algorithm for polynomials over any ring.

Definition 2.2.11. *The* **greatest common divisor** *of any two polynomials $f(x), g(x) \in F[x]$ is the monic polynomial $p(x)$ of highest degree such that $p(x)|f(x)$ and $p(x)|g(x)$.*

Proposition 2.2.12. *(Division algorithm) Let F be a field. If $p(x)$, $d(x)$ are polynomials in $F[x]$ with $0 < deg(d(x)) < deg(p(x))$ then there are polynomials $q(x)$, $r(x)$ in $F[x]$ such that $deg(q(x)) < deg(p(x))$, $deg(r(x)) < deg(d(x))$ in $F[x]$ and*

$$p(x) = d(x)q(x) + r(x).$$

For p, d, q, r as in the above theorem, we call $q(x)$ the **quotient**, written $quo(p(x), d(x))$, and we call $r(x)$ the **remainder**. written $rem(p(x), d(x))$.

Example 2.2.13. *This does not hold for rings in the form stated. (See Knuth [Kn], §4.6.1, for a more general form of the division algorithm.) For example, If $F = \mathbb{Z}$, $d(x) = 2x$ and $p(x) = x^2$ then there is no quotient $q(x)$ in $\mathbb{Z}[x]$ such that $p(x) = d(x)q(x) + r(x)$ holds.*

Example 2.2.14. *Let $p(x) = 3x^6 + 5x^4 - 4x^2 - 9x + 21$, $d(x) = x^2 + 1$. We have*

	$3x^4$		$2x^2$		-6		
x^2+1	$3x^6$	$+0 \cdot x^5$	$5x^4$	$0 \cdot x^3$	$-4x^2$	$-9x$	$+21$
	$3x^6$	$+0 \cdot x^5$	$3x^4$				
			$2x^4$	$0 \cdot x^3$	$-4x^2$	$-9x$	$+21$
			$2x^4$	$0 \cdot x^3$	$2x^2$		
					$-6x^2$	$-9x$	$+21$
					$-6x^2$	$0 \cdot x$	-6
						$-9\ x$	$+27$

So $q(x) = 3x^4 + 2x^2 - 6$ and $r(x) = 27 - 9x$.

Theorem 2.2.15. *(Euclidean algorithm for polynomials) Let F be a field. If $a(x)$, $b(x)$ are polynomials in $F[x]$ then there are polynomials $p(x)$, $q(x)$ in $F[x]$ such that*

$$gcd(a(x), b(x)) = a(x)p(x) + b(x)q(x).$$

The proof of this, which is very similar to the proof for the case of the integers given in chapter 1, is omitted. However, here is a sketch of the **Euclidean algorithm for polynomials**.

Let $r_{-1} = a$ and $r_0 = b$. Let $i = 0$.

(1) Use the division algorithm to compute polynomials q_{i+1} and r_{i+1}, $0 \leq \deg(r_{i+1}) < deg(r_i)$ such that

$$r_{i-1} = r_i q_{i+1} + r_{i+1}.$$

(2) If $r_{i+1} \neq 0$ then increment i and go to step 1. If $r_{i+1} = 0$ then stop.

Since $0 \leq ... < deg(r_1) < deg(r_0) = deg(b) < deg(r_{-1}) = deg(a)$, at some point the above algorithm must terminate. Suppose that $r_k = 0$ and k is the smallest such integer. Fact: $gcd(a, b) = r_{k-1}$.

Corollary 2.2.16. *Let $a(x)$ and $b(x)$ be polynomials in $F[x]$, where F is a field. $a(x)$ is relatively prime to $b(x)$ if and only if there are $p(x), q(x) \in F[x]$ such that $a(x)p(x) + b(x)q(x) = 1$.*

proof: (only if) This is a special case of the Euclidean algorithm above.

(if) If $d(x) \in F[x]$ is a non-zero polynomial satisfying both $d(x)|a(x)$ and $d(x)|b(x)$ then $d(x)|(a(x)p(x) + b(x)q(x) = 1$, so $d|1$. This forces $d(x) \in F^{\times}$, so $gcd(a(x), b(x)) = 1$. □

Here is a reformulation of the Euclidean Algorithm.

Theorem 2.2.17. *Let $f(x), g(x) \in F[x]$ be given, where F is a field. Assume $deg(g(x)) \leq deg(f(x))$, and let $r_{-1}(x) = f(x)$, $r_0(x) = g(x)$. Repeated Euclidean Algorithm can be regarded as a sequence of polynomials $\{r_k(x)\}$, $\{a_k(x)\}$, and $\{b_k(x)\}$ satisfying the following properties.*

- *For $k > 0$,*

$$r_{k-2}(x) = q_k(x)r_{k-1}(x) + r_k(x), \qquad deg(r_k(x)) < deg(r_{k-1}(x)),$$

$$r_k(x) = a_k(x)f(x) + b_k(x)g(x).$$

- *For some ℓ,*

$$gcd(f(x), g(x)) = a(x)f(x) + b(x)g(x),$$

where $a(x) = a_\ell(x)$, $b(x) = b_\ell(x)$.

- $$deg(a_k(x)) = \sum_{j=2}^{k} deg(q_j(x)), \qquad deg(b_k(x)) = \sum_{j=1}^{k} deg(q_j(x)).$$

- $$\det \begin{pmatrix} a_k(x) & b_k(x) \\ a_{k+1}(x) & b_{k+1}(x) \end{pmatrix} = (-1)^{k+1}.$$
- $$\det \begin{pmatrix} r_k(x) & b_k(x) \\ r_{k+1}(x) & b_{k+1}(x) \end{pmatrix} = (-1)^{k+1} f(x).$$
- $$\det \begin{pmatrix} u_k(x) & r_k(x) \\ u_{k+1}(x) & r_{k+1}(x) \end{pmatrix} = (-1)^{k+1} g(x).$$

Remark 2.2.18. *This version is useful for decoding alternant codes using Sugiyama's algorithm. However, this goes beyond the scope of this book. For more details, see Roman [Rom], page 399.*

2.2.5 Extended Euclidean Algorithm

Extended Euclidean Algorithm for $d(x) = \gcd(a(x), b(x)) = a(x)p(x) + b(x)q(x)$ for $0 \leq deg(b(x)) < deg(a(x))$.

(a) let $\overline{u} = \langle a(x), 1, 0 \rangle$

(b) let $\overline{v} = \langle b(x), 0, 1 \rangle$

(c) Repeat:

- let $\overline{w} = \overline{u} - quo(u_1, v_1)\overline{v}$,
- let $\overline{u} = \overline{v}$,
- let $\overline{v} = \overline{w}$,

while $v_1 \neq 0$

(d) let $d(x) = u_1$, $p(x) = u_2$, $q(x) = u_3$

(e) return $d(x), p(x), q(x)$.

The repeat loop in step (c) means to continue every step until you see $v_1 = 0$. This algorithm does not always yield a monic polynomial for $d(x)$ (the gcd is always monic), so it doesn't always yield the correct answer. However, the polynomial $d(x)$ it produces is a constant times the correct gcd.

Example 2.2.19. *Let* $a(x) = (x+1)(3x^3+1)$, $b(x) = (x+1)(x^2+1)$, *so* $gcd(a(x), b(x)) = x+1$. *We have*

$$\begin{aligned} u &= [(x+1)\left(3\,x^3+1\right), 1, 0], \\ v &= [(x+1)\left(x^2+1\right), 0, 1], \\ w &= [1-2\,x-3\,x^2, 1, -3\,x], \end{aligned}$$

since $quo(a, b) = 3x$. *We have*

$$\begin{aligned} u &= [(x+1)\left(x^2+1\right), 0, 1], \\ v &= [1-2\,x-3\,x^2, 1, -3\,x], \\ w &= [\tfrac{10}{9}+\tfrac{10}{9}\,x, x/3\ +1/9, -x^2-x/3\ +1], \end{aligned}$$

since $quo((x+1)\left(x^2+1\right), 1-2\,x-3\,x^2) = -x/3-1/9$. *We have*

$$\begin{aligned} u &= [1-2\,x-3\,x^2, 1, -3\,x], \\ v &= [\tfrac{10}{9}+\tfrac{10}{9}\,x, x/3\ +1/9, -x^2-x/3\ +1], \\ w &= [0, \tfrac{9}{10}\,x^2+\tfrac{9}{10}, -\tfrac{27}{10}\,x^3-\tfrac{9}{10}], \end{aligned}$$

since $quo(1-2\,x-3\,x^2, \frac{10}{9}+\frac{10}{9}\,x) = (-27x/10+9/10)$. *We have*

$$\begin{aligned} u &= [\tfrac{10}{9}+\tfrac{10}{9}\,x, x/3\ +1/9, -x^2-x/3\ +1], \\ v &= [0, \tfrac{9}{10}\,x^2+\tfrac{9}{10}, -\tfrac{27}{10}\,x^3-\tfrac{9}{10}]. \end{aligned}$$

We can stop now since $v_1 = 0$. *We read off* $p(x) = x/3+1/9$, $q(x) = -x^2-x/3+1$, $d(x) = \frac{10}{9}+\frac{10}{9}\,x$ *(which must be multiplied by* $9/10$ *to make it monic).*

Exercise 2.2.20. *Let* $F = \mathbb{Z}/p\mathbb{Z}$ *denote the field having* p *elements and let* $f(x) = x^p - x \in F[x]$. *Show that for all* $x \in F$, $f(x) = 0$. *(Hint: Use Fermat's Little Theorem.)*

Exercise 2.2.21. *Let* F *be a field and let* $F(x)$ *denote the set of all rational functions (i.e., expressions of the form* $\frac{p(x)}{q(x)}$ *where* $p(x), q(x) \in F[x]$ *and* $q(x) \neq 0$*). Show that* $F(x)$ *is a field.*

Exercise 2.2.22. *Let* $F = \mathbb{F}_5$ *and let* $f(x) = 2^x$, *so* $f : F \to F$. *Use the interpolation formula to find a polynomial representation of* f.

Exercise 2.2.23. *Use the extended Euclidean algorithm to find the gcd* $d(x)$ *of* $a(x) = x^2 - 1$ *and* $b(x) = x^3 - 1$ *in* $\mathbb{Q}[x]$. *Also, find* $u(x), v(x)$ *such that*

$$u(x)(x^2-1) + v(x)(x^3-1) = d(x).$$

Exercise 2.2.24. *Use the extended Euclidean algorithm to find the gcd $d(x)$ of $a(x) = x^3 - 1$ and $b(x) = x^4 - 1$ in $\mathbb{F}_5[x]$. Also, find $u(x), v(x)$ such that*

$$u(x)(x^3 - 1) + v(x)(x^4 - 1) = d(x).$$

Exercise 2.2.25. *Use the extended Euclidean algorithm to find the gcd $d(x)$ of $a(x) = x^3 - 1$ and $b(x) = x^4 - 1$ in $\mathbb{F}_5[x]$. Also, find $u(x), v(x)$ such that*

$$u(x)(x^3 - 1) + v(x)(x^4 - 1) = d(x).$$

Exercise 2.2.26. *Show that $p(x) = x^2 - 2$ has no roots in $F = \mathbb{F}_3$.*

Exercise 2.2.27. *Show that $p(x) = x^2 + 1$ has no roots in $F = \mathbb{F}_3$.*

Exercise 2.2.28. *Show that $p(x) = x^2 + 1$ has two roots in $F = \mathbb{F}_5$ (namely, $\overline{2}$ and $\overline{3}$).*

Exercise 2.2.29. *Show that $p(x) = x^3 - x$ has 6 roots in $\mathbb{Z}/6\mathbb{Z}$.*

Exercise 2.2.30. *Show that $p(x) = x^3 - x$ has 3 roots in $\mathbb{F}_3$.*

Exercise 2.2.31. *Show that $p(x) = x^{p-1} - 1$ has $p - 1$ roots in $\mathbb{Z}/p\mathbb{Z}$. (This is a restatement of Fermat's little theorem.)*

Exercise 2.2.32. *Let $p(x) = x^2 + 1$ in $R[x]$, where $R = \mathbb{Z}/4\mathbb{Z}$.*
(a) Let $c = 1$. Find $q(x)$ satisfying Lemma 2.2.5.
(b) Find all roots of $p(x)$ in R.

Exercise 2.2.33. *Let $p(x) = x^2 - 1$ in $R[x]$, where $R = \mathbb{Z}/6\mathbb{Z}$.*
(a) Let $c = 1$. Find $q(x)$ satisfying Lemma 2.2.5.
(b) Find all roots of $p(x)$ in R.

Exercise 2.2.34. *Let $p(x) = x^4 + 4$ in $F[x]$, where $F = \mathbb{F}_5$.*
(a) Let $c = 1$. Find $q(x)$ satisfying Lemma 2.2.5.
(b) Find all roots of $p(x)$ in F.

Exercise 2.2.35. *Use the Euclidean algorithm to compute the gcd of $a(x) = x^2 - 1$ and $b(x) = x^3 - 1$ in $\mathbb{Q}[x]$.*

Exercise 2.2.36. *Use the Euclidean algorithm to compute the gcd of $a(x) = x^3 - 1$ and $b(x) = x^4 - 1$ in $\mathbb{F}_5[x]$.*

Exercise 2.2.37. *Use the Euclidean algorithm to compute the gcd of* $a(x) = x^3 - 1$ *and* $b(x) = x^4 - 1$ *in* $\mathbb{Q}[x]$.

Exercise 2.2.38. *Prove Theorem 2.2.15. (Hint: Modify the proof of 1.4.3.)*

Exercise 2.2.39. *Carry out each step of the "reformulated" Euclidean algorithm for* $a(x) = x^2 - 1$ *and* $b(x) = x^3 - 1$ *in* $\mathbb{Q}[x]$. *Verify*

$$\det \begin{pmatrix} u_k(x) & r_k(x) \\ u_{k+1}(x) & r_{k+1}(x) \end{pmatrix} = (-1)^{k+1} a(x).$$

Exercise 2.2.40. *Use the "reformulated" Euclidean algorithm to compute the gcd* $d(x)$ *of* $a(x) = x^2 - 1$ *and* $b(x) = x^3 - 1$ *in* $\mathbb{Q}[x]$ *and* $u(x), v(x)$ *such that*

$$u(x)(x^2 - 1) + v(x)(x^3 - 1) = d(x).$$

2.3 Polynomial ideals

We first motivate the material and give some definitions.

2.3.1 Motivation

Recall that in chapter 1, §1.4, we studied the following question:

Question: Let a and b be any two non-zero integers. Can the set of all possible sums of multiples of a and b be described in a "simple" way? If so, how?

Before we recall how we answered this question, let us introduce some terminology. The " the set of all possible sums of multiples" of a given set of elements occurs so often in abstract algebra that it has a special name.

Definition 2.3.1. *Let* R *be a ring and let* $a_1, ..., a_k$ *be elements of* R. *The set* I *of all possible sums of multiples of the* a_i *is called the* **ideal of** R **generated by** a_1**, ...,** a_k. *In other words,*

$$I = \{r_1 a_1 + ... r_k a_k \mid r_i \in R\}.$$

When there is no danger of confusion, it is often denoted $I = (a_1, ..., a_k)$.

In this terminology, the question above becomes: What is $I = (a, b) \subset \mathbb{Z}$?

We answered the question by proving that *any* subset of the integers which was closed under addition and (integer) multiplication must be of the form $c\mathbb{Z}$, for some integer $c \geq 1$ (Lemma 1.4.2). (In fact, $c = gcd(a, b)$.) In other words, we showed $I = (a, b) = (c)$, where $c = gcd(a, b)$. This may be reworded as saying, the ideal in $\mathbb{Z}$ generated by the 2 elements a, b is an ideal generated by only 1 element $c = gcd(a, b)$. Ideals generated by only one elements also have a special name.

Definition 2.3.2. *Let R be a ring and let a be an element of R. The set I of all possible multiples of a is called the* **principal ideal of R generated by** *a. In other words,*

$$I = \{ra \mid r_i \in R\}.$$

When there is no danger of confusion, it is often denoted $I = (a)$.

Using this terminology, we showed in chapter 1 that the ideal of $\mathbb{Z}$ generated by a, b is a principal ideal. One can use mathematical induction to show that any ideal in $\mathbb{Z}$ of the for $I = (a_1, ..., a_k)$ is a principal ideal. Then *every* ideal of a ring R is a principal ideal, then R is called a **principal ideal domain**. It is possible to show that the ring $\mathbb{Z}$ is a principal ideal domain.

2.3.2 The ring $F[x]$

In this section, we shall ask and answer the polynomial analog of the question mentioned in the previous section. We state a prove a polynomial analog of Lemma 1.4.2.

Proposition 2.3.3. *Let F be a field. Let $I \subset F[x]$ be a closed under addition and subtraction, and under multiplication by any polynomial having integer coefficients* [8]*. Then either*

- *I is empty,*
- *$I = \{0\}$.*
- *there is a polynomial $a(x)$ with coefficients in F such that $I = g(x)F[x]$, i.e., I is the set of all multiples of some polynomial $g(x)$ (which is constructed in the proof below).*

[8] This condition says that I is an **ideal** in $F[x]$.

The proof below is worthwhile trying to understand well since it contains a basic idea which occurs in other parts of mathematics.

proof: If I is not empty then it must contain 0. If it contains some non-zero element then it must contain a polynomial of degree > 0. By the well-ordering principle, I contains an element of least degree $g(x) \neq 0$.

Claim: $I = a(x)F[x]$.

If $b \in I$ then the division algorithm says that there is a polynomial $r(x)$ $0 \leq \deg(r(x)) < \deg(g(x))$ such that $b(x) = q(x)g(x) + r(x)$. Since I is closed under addition and subtraction, $r(x) = b(x) - q(x)g(x)$ belongs to I. But g is a non-zero element of I of lowest degree, so $r(x)$ must be zero. This implies $b(x) \in g(x)F[x]$. Since $b(x)$ was choosen arbitrarily, this imples $I \subset g(x)F[x]$. The reverse inclusion $g(x)F[x] \subset I$ follows from the assumption that I is closed under addition and subtraction. This proves the claim and the lemma. $\square$

In other words, any non-empty subset I of polynomials having coefficients in a field F which is closed under addition and polynomial multiplication must be the set of multiples of a fixed polynomial $g(x)$. Using mathematical induction, it is possible to show that an ideal of $F[x]$ is principal. This implies $F[x]$ is a principal ideal domain.

We will study later the "ring of polynomials modulo I",

$$F[x]/I = \{\overline{f(x)} \mid f(x) \in F[x]\},$$

where

$$\overline{f(x)} = f(x)+I = \{f(x)+g(x) \mid g(x) \in I\} = \{f(x)+g(x)p(x) \mid p(x) \in F[x]\}.$$

This may at first look rather mysterious but it is the polynomial analog of the ring of integers modulo m, $\mathbb{Z}/m\mathbb{Z}$. At this point, to understand $F[x]/I$ we need to understand how to factor polynomials better, which we turn to next.

Exercise 2.3.4. *Let I be the ideal of $\mathbb{Z}[x]$ generated by 2 and x. Show I is not a principal ideal.*

Exercise 2.3.5. *Let I be the ideal of $\mathbb{Q}[x]$ generated by $x^2 - 1$ and $x^3 - 1$. Show I is a principal ideal and find its generator.*

2.4 Factoring polynomials

Getting back to polynomials, we know that each polynomial of degree n with coefficients in a field can have at most n factors. We don't yet have any strategies

(a) for finding these factors, or

(b) for determining whether or not the polynomial has any factors.

Note that a polynomial may factor yet have no roots in the field, for example $x^4 + 3x^2 + 2 = (x^2 + 1)(x^2 + 2)$ has no real roots. However, if the polynomial has a root then it factors.

Definition 2.4.1. *Let F be a ring. A polynomial in $F[x]$ which does not factor into a product of two or more polynomials of smaller degree which also belong to $F[x]$ is called* **irreducible** *over F.*

Irreducible polynomials are analogous to prime numbers. One of the properties of a prime number was that if p was a prime and $p|ab$ then either $p|a$ or $p|b$ ("Euclid's First Theorem", Proposition 1.5.8). The analogous result also holds for irreducible polynomials.

Lemma 2.4.2. *Let $p(x)$ be an irreducible polynomial in $F[x]$, where F is a field. If $p(x)|f(x)g(x)$, where $f(x), g(x) \in F[x]$, then $p(x)|f(x)$ or $p(x)|g(x)$.*

The proof is analogous to the proof of Proposition 1.5.8, and so is omitted.

If a polynomial $p(x)$ has a root in F, say $p(c) = 0$, then it is not irreducible since, by the division algorithm, $x - c$ is a factor. One would like to be able to determine whether or not a polynomial $p(x)$ is irreducible simply by means of its roots (or lack of them) in F. This will not work in general, as we saw with the example $x^4 + 3x^2 + 2 = (x^2 + 1)(x^2 + 2)$ (which obviously factors but has no real roots). However, if the polynomial is either a quadratic or a cubic, then it's a different story.

Lemma 2.4.3. *If $p(x)$ is either a quadratic or a cubic in $F[x]$ then $p(x)$ is irreducible if and only if $p(x)$ has no roots.*

proof: If $p(x)$ has degree 2 then it can only factor into a product of two degree 1 polynomials, each of which will have a root. If $p(x)$ has degree 3 then it can only factor into either a product of three degree 1 polynomials or a degree 2 and a degree 1 polynomial, at least one of which will have a root. □

In the previous chapter, we found out one can always factor an integer into prime powers (the fundamental theorem of arithmetic) and we considered a basic technique for factoring integers which is practical for integers of "small" degree. Here we want to know

- if one can always factor a polynomial over a ring or field F into powers of irreducible polynomials, and
- an algorithm for factoring polynomials over F which is practical at least for those of "small" degree.

The first issue is addressed in the next section.

2.4.1 Unique factorization theorem

The polynomial analog of the fundamental theorem of arithmetic for the integers is the unique factorization theorem.

Theorem 2.4.4. *(Unique factorization theorem) Let F be a field and $p(x) \in F[x]$. Then there are monic irreducible polynomials $p_i(x) \in F[x]$, integers $e_i > 0$, $i = 1, 2, ..., k$, and a constant $c \in F^\times$ such that*

$$f(x) = cp_1(x)^{e_1}...p_k(x)^{e_k}.$$

Furthermore, this expression is unique up to order.

proof: The proof is analogous to the proof of the fundamental theorem of arithmetic. It is therefore omitted. □

2.4.2 General principles in factoring

A general principle in factoring polynomials is to determine whether or not the polynomial has multiple factors is the following one. First, for any polynomial $f(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n \in F[x]$, let

$$f'(x) = a_1 + 2a_2x + ... + na_nx^{n-1}$$

denote the **derivative polynomial**.

Example 2.4.5. *If the field is not characteristic zero then funny things can happen when you differentiate! For example, if* $f(x) = x^{25} + x^7 \in \mathbb{F}_5[x]$ *then* $f'(x) = 25x^{24} + 7x^6 = 2x^6$*, so* $deg(f(x)) = 25$ *while* $deg(f'(x)) = 6$*. Even stranger things than this can happen: if* $f(x) = x^5 + 1 \in \mathbb{F}_5[x]$ *then* $f'(x) = 5x^4 = 0$*. In other words, some non-constant polynomials have derivative equal to* 0*!*

Lemma 2.4.6. *If* F *is a ring,* $f(x)$ *is a polynomial in* $F[x]$*, and* $f'(x) \neq 0$ *(i.e.,* $f'(x)$ *is not the zero polynomial) and then* $f(x)$ *has a multiple factor if and only if* $gcd(f(x), f'(x)) \neq 1$*.*

proof: The proof uses the fact that if $f(x) = g(x)^k h(x)$ then we have the "product rule" $f'(x) = kg(x)^{k-1}g'(x)h(x) + g(x)^k h'(x)$. This implies $g(x)^{k-1} | gcd(f(x), f'(x))$, which proves the lemma. □

Example 2.4.7. *Let*

$$f(x) = (x+2)^3(x^2+1)^5,$$

so

$$\begin{aligned} f'(x) &= 3\,(x+2)^2\,(x^2+1)^5 + 10\,(x+2)^3\,(x^2+1)^4\,x \\ &= (3 + 20\,x + 13\,x^2)\,(x+2)^2\,(x^2+1)^4\,, \end{aligned}$$

so $gcd(f(x), f'(x)) = (x+2)^2(x^2+1)^4$*.*

Lemma 2.4.8. *If* F *is a field of characteristic zero and* $f(x)$ *is an irreducible polynomial in* $F[x]$*,* $f(x)$ *has cannot have a multiple root in any field extension* E/F*.*

proof: Since $f(x)$ is an irreducible polynomial in $F[x]$, by the previous lemma and the Euclidean algorithm, there are polynomials $p(x)$ and $q(x)$ such that $f(x)p(x) + f'(x)q(x) = 1$. This also holds over any field extension.

Suppose, to get a contradition, $f(x)$ had a multiple root in E. By the previous lemma, it would have a root in common with $f'(x)$, say $c \in E$. Plugging this into the equation $f(x)p(x) + f'(x)q(x) = 1$ would yield $0 = 1$, which is impossible. □

2.4.3 Factoring $x^n - 1$ over $\mathbb{F}_p$

As an example of polynomial factorization, we shall factor $x^n - 1$ over and $\mathbb{F}_p$ (over the fields $\mathbb{C}$ and $\mathbb{Q}$, see the Special Projects section towards the end of the chapter).

There are many polynomials having integral coefficients which are irreducible over $\mathbb{Q}$ yet, when regarded as polynomials over $\mathbb{F}_p$, factor into irreducibles having smaller degree. For example, x^2+1 is irreducible over $\mathbb{Q}$ but over $\mathbb{F}_2$, $x^2+1 = x^2-1 = (x+1)^2$, since $+1 = -1$ in $\mathbb{F}_2$.

This is not too bizarre. It simply warns us that even though we might have an irreducible polynomial having integral coefficients (i.e., irreducible in $\mathbb{Q}[x]$), after we reduce mod p, it need not be irreducible (i.e., might be reducible in $\mathbb{F}_p[x]$). What is even more remarkable is the following fact.

Theorem 2.4.9. *Let p be any prime and let $f(x)$ be any polynomial having coefficients in $\mathbb{F}_p$. There there is an $n > 1$ for which $f(x)$ divides $x^n - 1$.*

(This theorem will not be proven here - it follows from the polynomial analog of Fermat's little theorem.) In other words, factoring all the $x^n - 1$ (the title of this section!) is tantamount to determining all the irreducible polynomials over $\mathbb{F}_p$.

Here is the rough idea. To factor $x^n - 1$, we begin by using the same formula that we did over $\mathbb{Q}$:

$$x^n - 1 = \prod_{m|n} C_m(x).$$

It suffices to factor the $C_m(x)$'s. Let k be the smallest integer for which $p^k \equiv 1 \pmod{m}$ (this exists by Fermat's Little Theorem). It is known that the roots of $C_m(x)$ in $\mathbb{F}_{p^k}$ are precisely the elements of order m in $\mathbb{F}_{p^k}$. The minimal polynomial of any element of order m in $\mathbb{F}_{p^k}$ is an irreducible factor of $C_m(x)$ and all irreducible factors of $C_m(x)$ arise in this way.

2.4.4 Factoring over a finite field

Let $p(x)$ be a polynomial in $\mathbb{F}_p[x]$. The idea is to try to find a polynomial $q(x)$ such that $g(x) = gcd(p(x), q(x))$ is of $\deg(g(x)) > 0$. This is then a factor of $p(x)$.

A basic fact here is the following result.

Theorem 2.4.10. *If $p(x)$ is an irreducible polynomial of degree d in $\mathbb{F}_p[x]$ then there is an $n > 1$ such that $p(x)$ divides $x^n - 1$. In fact, one may take $n = p^d - 1$.*

The smallest n satisfying Theorem 2.4.10 is called the **order of x mod $p(x)$**.

This theorem suggests the following strategy to factor a polynomial $f(x)$ over the finite field $\mathbb{F}_p$, one may use the following procedure.

- Eliminate any multiple factors by dividing $f(x)$ by $gcd(f(x), f'(x))$, as in Lemma 2.4.6. Let $q(x)$ denote the resulting quotient.
- Let $d = 1$.
- Compute $gcd(q(x), x^{p^d} - x)$. If this gcd is not equal to 1 and if $d < deg(q(x))$, replace $q(x)$ by $q(x)/gcd(q(x), x^{p^d} - x)$, replace d by $d + 1$, and go to the previous step. If $d \geq deg(q(x))$, stop. If $d < deg(q(x))$ but the gcd equals 1 then replace d by $d + 1$, and go to the previous step.

Example 2.4.11. *Let $F = \mathbb{F}_3$ and $p(x) = (x+2)^3(x^2+1)^5 \in F[x]$, so $q(x) = (x+2)(x^2+1)$. We have $gcd(q(x), x^{3^i} - x) \equiv x + 2 \pmod 3$. Since $q(x)/(x+2) = x^2+1$ is irreducible, the method above has completely factored.*

Exercise 2.4.12. *Factor all monic polynomials of degree ≤ 4 in $\mathbb{F}_2[x]$.*

Exercise 2.4.13. *Factor all monic polynomials of degree ≤ 3 in $\mathbb{F}_3[x]$.*

Exercise 2.4.14. *Factor all monic polynomials of degree ≤ 2 in $\mathbb{F}_5[x]$.*

Exercise 2.4.15. *Prove Lemma 2.4.2.*

Exercise 2.4.16. *Find all the irreducible polynomials of degree ≤ 3 over $\mathbb{F}_2$.*

Exercise 2.4.17. *Find all the irreducible polynomials of degree ≤ 3 over $\mathbb{F}_3$.*

Exercise 2.4.18. *Find all the irreducible polynomials of degree ≤ 2 over $\mathbb{F}_5$.*

Exercise 2.4.19. *Factor $x^2 + 1$ in $\mathbb{F}_2[x]$.*

Exercise 2.4.20. *Factor $x^3 + 1$ in $\mathbb{F}_3[x]$.*

Exercise 2.4.21. *Factor $x^3 + 2$ in $\mathbb{F}_3[x]$.*

Exercise 2.4.22. *Let F be any field. Show that there are infinitely many irreducible polynomials in $F[x]$. (Hint: Think of the polynomial analog of Euclid's second theorem (Theorem 1.5.1).)*

Exercise 2.4.23. *Find if x^2+x+1 in $\mathbb{Q}[x]$ has a multiple root or not.*

Exercise 2.4.24. *Find if x^2+x+1 in $\mathbb{F}_3[x]$ has a multiple root or not.*

Exercise 2.4.25. *Find if $x^4+x^3+x^2+x+1$ in $\mathbb{Q}[x]$ has a multiple root or not.*

Exercise 2.4.26. *Find if $x^4+x^3+x^2+x+1$ in $\mathbb{F}_5[x]$ has a multiple root or not.*

Exercise 2.4.27. *Find an $n>1$ such that x^3+4x^2+3 divides x^n-1 in $\mathbb{F}_5[x]$. (This illustrates Theorem 2.4.10.)*

2.5 Modular arithmetic with polynomials

In this section, F will be a field, unless stated otherwise.

Modular arithmetic with polynomials is very similar to modular arithmetic with integers.

Definition 2.5.1. *If $a(x), b(x), m(x) \in F[x]$ then we say $a(x)$* **is congruent to** *$b(x)$* **modulo** *$m(x)$, written $a(x) \equiv b(x) \pmod{m(x)}$, if $m(x)$ divides the polynomial $a(x)-b(x)$.*

Lemma 2.5.2. *Let F be a ring. Fix a polynomial modulus $m(x) \neq 0$ and let $a(x), b(x), c(x), d(x) \in F[x]$.*

- *If $a(x) \equiv c(x) \pmod{m(x)}$ and $(x)b \equiv d(x) \pmod{m(x)}$ then*

$$\begin{aligned} a(x)+b(x) &\equiv c(x)+d(x) \pmod{m(x)} \\ a(x)-b(x) &\equiv c(x)-d(x) \pmod{m(x)} \\ a(x)b(x) &\equiv c(x)d(x) \pmod{m(x)} \\ a(x)^2 &\equiv c(x)^2 \pmod{m(x)}. \end{aligned}$$

- *If $a(x)+c(x) \equiv b(x)+c(x) \pmod{m(x)}$ then $a(x) \equiv b(x) \pmod{m(x)}$.*
- *If $a(x)c(x) \equiv b(x)c(x) \pmod{m(x)}$ and $gcd(c(x), m(x)) = 1$ then $a(x) \equiv b(x) \pmod{m(x)}$.*

proof: All of these are left as an exercise, except for the last one.

Assume $a(x)c(x) \equiv b(x)c(x) \pmod{m(x)}$ and $gcd(c(x), m(x)) = 1$. Then $m(x)|(a(x)c(x)-b(x)c(x)) = c(x)(a(x)-b(x))$. By the unique factorization theorem, $m(x)|(a(x)-b(x))$. $\square$

Example 2.5.3. $2x^2 \equiv 2 \pmod{x^2-1}$ *since* $(x^2-1)|(2x^2-2)$. *Similarly,* $x^4 \equiv 1 \pmod{x^2-1}$ *and* $x^3 \equiv x \pmod{x^2-1}$.

In general, when computing a polynomial modulo x^2-1, *we may always replace* x^2, x^4, x^6, ... *by* 1 *and* x^3, x^5, x^7, ... *by* x. *For example,* $7x^8+3x^7-5x^6+2x^5-2x^4+x^3-4x^2-x+9 \equiv (3+2+1-1)x+(7-5-2-4+9) \equiv 5x+5 (\mathrm{mod}\ x^2-1)$.

Example 2.5.4. *Let* $m(x) = x^2+1$. *By the division algorithm, we have* $x \equiv x \,(\mathrm{mod}\, x^2+1)$, $x^2 \equiv -1 \,(\mathrm{mod}\, x^2+1)$, $x^3 \equiv -x \,(\mathrm{mod}\, x^2+1)$, $x^4 \equiv -x^2 \equiv 1 \,(\mathrm{mod}\, x^2+1)$, $x^5 \equiv -x^3 \equiv x \,(\mathrm{mod}\, x^2+1)$. *In general, this pattern of congruences leads to* $x^{2n} \equiv (-1)^n \,(\mathrm{mod}\, x^2+1)$, $x^{2n+1} \equiv (-1)^n x \,(\mathrm{mod}\, x^2+1)$, *for* $n > 0$.

For each fixed non-zero $m(x) \in F[x]$, the congruence relation $\equiv$ defines an equivalence relation (in the sense of §1.7.2). Let

$$\overline{a(x)} = a(x) + m(x)F[x].$$

denote an equivalence class, which we call the **congruence class** (or **residue class**) of $a(x)$. Define addition and multiplication of congruence classes by

$$(a(x) + m(x)F[x]) + b(x) + m(x)F[x]) = a(x) + b(x) + m(x)F[x],$$

and

$$(a(x) + m(x)F[x])(b(x) + m(x)F[x]) = a(x)b(x) + m(x)F[x].$$

This collection of congruence classes is denoted $F[x]/(m(x))$.

As in the integral case treated in chapter 1, the following result, is a consequence of the Euclidean algorithm.

Lemma 2.5.5. *Let* $a(x), m(x) \in F[x]$ *be non-zero polynomials. Assume that* $m(x)$ *is not a constant.* $a(x)$ *is relatively prime to* $m(x)$ *if and only if there is a* $b(x) \in F[x]$ *such that* $a(x)b(x) \equiv 1 \pmod{m(x)}$.

The polynomial $b(x)$ in the above lemma is called the **inverse of** $a(x)$ **modulo** $m(x)$, written $b(x) = a(x)^{-1} \pmod{m(x)}$.

Example 2.5.6. $-x$ *is the inverse of* x *modulo* x^2+1 *in* $\mathbb{R}[x]$ *since* $(-x)\cdot x = -x^2 \equiv 1 \pmod{x^2+1}$.

proof: (only if) Assume $gcd(a(x), m(x)) = 1$. There are $p(x), q(x) \in F[x]$ such that $a(x)p(x) + m(x)q(x) = 1$, by the Euclidean algorithm (more precisely, by Corollary 2.2.16). Thus $m(x)|(a(x)p(x) - 1)$.

(if) Assume $a(x)p(x) \equiv 1 \pmod{m(x)}$. There is a polynomial $q(x)$ such that $a(x)p(x) - 1 = m(x)q(x)$. By Corollary 2.2.16 again, we must have $gcd(a(x), m(x)) = 1$. □

Exercise 2.5.7. *Let $m(x) = x^2+1 \in \mathbb{R}[x]$. Compute $(x+1)^{-1} \pmod{m(x)}$.*

Exercise 2.5.8. *Let $m(x) = x^2 + x + 1 \in \mathbb{F}_2[x]$. Compute $x^{-1} \pmod{m(x)}$.*

2.6 Arithmetic properties of $F[x]/(m(x))$

Let F be a field. Is $F[x]/(m(x))$ a ring? Is there an analog of Euler's theorem 1.8.4? The answer to both is yes.

Proposition 2.6.1. *Let F be a field. For any polynomials $a(x), b(x), c(x) \in F[x]$,*

1. $\overline{a(x)} + \overline{b(x)} = \overline{b(x)} + \overline{a(x)}$, *("addition is commutative")*
2. $\overline{a(x)b(x)} = \overline{b(x)a(x)}$, *("multiplication is commutative")*
3. $(\overline{a(x)} + \overline{b(x)}) + \overline{c(x)} = \overline{a(x)} + (\overline{b(x)} + \overline{c(x)})$, *("addition is associative")*
4. $(\overline{a(x)b(x)})\overline{c(x)} = \overline{a(x)}(\overline{b(x)c(x)})$, *("multiplication is associative")*
5. $(\overline{a(x)} + \overline{b(x)})\overline{c(x)} = \overline{a(x)c(x)} + \overline{b(x)c(x)}$, *("distributive")*
6. $\overline{a(x)}\overline{1} = \overline{a(x)}$, *("$\overline{1}$ is a multiplicative identity")*
7. $\overline{a(x)} + \overline{-a(x)} = \overline{0}$,
8. $\overline{a(x)}\overline{0} = \overline{0}$,
9. $\overline{a(x)} + \overline{0} = \overline{a(x)}$ *("$\overline{0}$ is an additive identity"),*
10. *if $\overline{a(x)c(x)} = \overline{b(x)c(x)}$ and $gcd(m(x), c(x)) = 1$ then $\overline{a(x)} = \overline{b(x)}$ ("cancellation law").*

Each of these properties of the congruence classes corresponds to a property of polynomials already proven in a previous section.

Example 2.6.2. *Let* $R = \mathbb{R}[x]/(x^2+1)$. *We have* $1 \equiv 1 \pmod{x^2+1}$, $x \equiv x \pmod{x^2+1}$, $x^2 \equiv -1 \pmod{x^2+1}$, ... $x^n \equiv (-1)^n \pmod{x^2+1}$, $x^n \equiv x(-1)^n \pmod{x^2+1}$*). Therefore, any element* $\overline{p(x)} \in R$ *may be represented by a polynomial of degree* ≤ 1. *As a set, we have*

$$R = \{a + xb \mid a, b \in \mathbb{R}\}.$$

Addition on R *corresponds to the usual addition of polynomials on* $\{a + xb \mid a, b \in \mathbb{R}\}$. *Multiplication on* R *corresponds to*

$$(a + xb)(c + xd) = ac + bd + (ad + bc)x,$$

since $x^2 = -1$. *In fact, with these definitions of addition and multiplication,* R *is a field. Moreover, the map* $\phi : R \to \mathbb{C}$, *defined by* $\phi(a+xb) = a+ib$, *is an isomorphism of fields. In other words, under* ϕ, *addition and multiplication of elements of* R *corresponds to addition and multiplication of elements of* $\mathbb{C}$ *in the sense that*

$$\phi(a)\phi(b) = \phi(ab), \qquad \phi(a) + \phi(b) = \phi(a + b),$$

for all $a, b \in \mathbb{R}$.

2.6.1 Constructing finite extensions of fields

In this section, we generalize the construction of §2.1.1.

Theorem 2.6.3. *Let* F *be a field. Then* $F[x]/(m(x))$ *is a field if* $m(x)$ *is an irreducible polynomial in* $F[x]$.

Remark 2.6.4. *The converse to this thorem also holds.*

proof: We have already seen that $F[x]/(m(x))$ is a commutative ring with unity. We need only show that each non-zero element in $F[x]/(m(x))$ has an inverse modulo $m(x)$, provided $m(x)$ is irreducible.

Let $a(x) \in F[x]/(m(x))$ be non-zero. Since $m(x)$ is irreducible, we have $gcd(m(x), a(x)) = 1$ (otherwise, it would be a non-trivial factor of $m(x)$). By Lemma 2.5.5, $a(x)$ has an inverse modulo $m(x)$. □

Definition 2.6.5. *If F is a finite field, if $m(x)$ is an irreducible polynomial in $F[x]$ of degree d, and if the powers of x (1, x, x^2, ..., x^{d-1}) yield all the non-zero elements of the field $F[x]/(m(x))$, then we call $m(x)$ a* **primitive polynomial**. *If $a \in F$ and if the powers of a (1, a, a^2, ..., a^{d-1}) yield all the non-zero elements of the field F then a is called a* **primitive element** *of F.*

Example 2.6.6. *(a) $m(x) = x^2 + x + 1$ is a primitive polynomial over $\mathbb{F}_2$.*

(b) $m(x) = x^4 + x + 1$ is a primitive polynomial over $\mathbb{F}_2$.

Example 2.6.7. *$\mathbb{R}[x]/(x^2+1)$ is a field. As a set, we may identify $\mathbb{R}[x]/(x^2+1)$ with a set of residue class representatives*

$$\mathbb{R}[x]/(x^2+1) = \{a + bx \mid a, b \in \mathbb{R}\}.$$

In fact, if $p(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n \in \mathbb{R}[x]$ then

$$\overline{p(x)} \equiv (a_0 - a_2 + -...) + (a_1 - a_3 + -...)x \,(\mathrm{mod}\, x^2+1),$$

by Example 2.5.4.

As long as you remember that in $\mathbb{R}[x]/(x^2+1)$, we have $x^2 = -1$ (since everything is modulo x^2+1), addition and multiplication in $\{a+bx \mid a, b \in \mathbb{R}\}$ is easy:

$$(a+bx)+(c+dx) = a+c+(b+d)x, \qquad (a+bx)\cdot(c+dx) = ad-bd+(ad+bc)x.$$

This defines addition $+$ and multiplication $\cdot$ on $\{a + bx \mid a, b \in \mathbb{R}\}$. With these binary operations, $\{a + bx \mid a, b \in \mathbb{R}\}$ is a field.

This looks just like complex multiplication, where $i = \sqrt{-1}$ has been replaced by x. In other words, there is a direct correspondence between the addition and multiplication formulas for $\mathbb{C}$ and for $\{a+bx \mid a, b \in \mathbb{R}\}$. More abstractly said, if we define $\phi : \{a+bx \mid a, b \in \mathbb{R}\} \to \mathbb{C}$ by $\phi(a+bx) = a+ib$ then

$$\phi(a)\phi(b) = \phi(ab), \qquad \phi(a) + \phi(b) = \phi(a+b),$$

for all $a, b \in \mathbb{R}$. This means that $\{a + bx \mid a, b \in \mathbb{R}\}$ and $\mathbb{C}$ are "isomorphic fields."

Example 2.6.8. *$\mathbb{F}_3[x]/(x^2+1)$ is a field. As a set, we may identify $\mathbb{F}_3[x]/(x^2+1)$ with a set of residue class representatives*

$$\mathbb{F}_3[x]/(x^2+1) = \{a + bx \mid a, b \in \mathbb{F}_3\}.$$

In fact, if $p(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n \in \mathbb{F}_3[x]$ *then*

$$\overline{p(x)} \equiv (a_0 - a_2 + -...) + (a_1 - a_3 + -...)x \pmod{x^2+1},$$

as in the above example.

Addition and multiplication in $\{a + bx \mid a, b \in \mathbb{F}_3\}$ *is easy:*

$$(a+bx)+(c+dx) = a+c+(b+d)x, \qquad (a+bx)\cdot(c+dx) = ac-bd+(ad+bc)x.$$

With these binary operations, $\{a + bx \mid a, b \in \mathbb{F}_3\}$ *is a field.*

This looks just like complex multiplication, where $i = \sqrt{-1}$ *has been replaced by* x*. In other words, there is a direct correspondence between the addition and multiplication formulas for* $\mathbb{F}_3(i) = \{a + bi \mid a, b \in \mathbb{F}_3\}$ *and* $\mathbb{F}_3[x]$*.*

In this example, $x^2 + x$ *is a primitive element in* $\mathbb{F}_3[x]$ *but* x *is not (so* x^2+1 *is not a primitive polynomial over* $\mathbb{F}_3$*). Also,* $i-1 \in \mathbb{F}_3(i)$ *is a primitive element but* i *is not.*

Motivated by the facts in the above examples, we make the following definition.

Definition 2.6.9. *Let* F, F' *be fields. Denote the binary oprations on both these fields by* $+$ *and* $\cdot$ *(even though they are probably different operations, it is assumed that the reader can distinguish which is which by the context). If* $\phi : F \to F'$ *is a map such that*

$$\phi(a)\phi(b) = \phi(ab), \qquad \phi(a) + \phi(b) = \phi(a + b),$$

for all $a, b \in F$ *then we call* ϕ *a* **field isomorphism**. *We also say that* F *and* F' *are* **isomorphic** *(as fields) and write* $F \cong F'$*.*

Example 2.6.10. *The field* $\mathbb{R}[x]/(x^2 + 1)$ *is isomorphic to* $\mathbb{C}$*. The field isomorphism is induced by sending* $x \in \mathbb{R}[x]/(x^2 + 1)$ *to* $i \in \mathbb{C}$ *and then extending it to all of* $\mathbb{R}[x]/(x^2+1)$*. In other words, define* $\phi : \mathbb{R}[x]/(x^2+1) \to \mathbb{C}$ *by* $\phi(1) = 1$*,* $\phi(x) = i$*, and then*

$$\phi(a)\phi(b) = \phi(ab), \qquad \phi(a) + \phi(b) = \phi(a + b),$$

for all $a, b \in \mathbb{R}[x]/(x^2 + 1)$*. This defines a field isomorphism.*

2.6.2 Kronecker's theorem

We apply the above results to proving that every polynomial has a root in some (extension) field.

Theorem 2.6.11. *(Kronecker's theorem) Let F be a field and let $p(x) \in F[x]$ be a non-constant polynomial. There is a field extension E of F and an element $c \in E$ such that $p(c) = 0$ (in E).*

proof: We may assume that $p(x)$ does not have a linear factor in $F[x]$ since otherwise that factor would yield the desired root (in F, which is an extension field of itself!). Let $f(x)$ denote a factor of $p(x)$ which is irreducible. Let $E = F[x]/(f(x))$. This is an extension field of F. The representative $x \in F[x]$ of the element $\overline{x} \in E$ satisfies $p(x) = f(x)g(x) \equiv 0 \pmod{f(x)}$, so $p(\overline{x}) = u \cdot f(\overline{x}) = \overline{0}$. $\square$

Example 2.6.12. *If $F = \mathbb{R}$ and $p(x) = x^2 + 1$ then $p(x)$ has a root i in $\mathbb{C} \cong \mathbb{R}[x]/(x^2 + 1)$ (which we see upon identifying i with x).*

Exercise 2.6.13. *Construct an extension field of $\mathbb{F}_2$ in which $x^2 + x + 1 \in \mathbb{F}_2[x]$ has a zero.*

Exercise 2.6.14. *Show that the smallest extension field of $\mathbb{F}_2$ in which $p_1(x) = x^4 + x + 1 \in \mathbb{F}_2[x]$ has a zero is isomorphic to the smallest extension field of $\mathbb{F}_2$ in which $p_2(x) = x^4 + x^3 + 1 \in \mathbb{F}_2[x]$ has a zero. (Hint: $p_1(x)$ is the "reciprocal" of $p_2(x)$: $x^4 p_1(x^{-1}) = p_2(x)$.)*

Exercise 2.6.15. *In Example 2.6.7, show that $(a + bx)^{-1} = \frac{a-bx}{a^2+b^2}$.*

Exercise 2.6.16. *In Example 2.6.8, show that $(a + bx)^{-1} = \frac{a-bx}{a^2+b^2}$.*

Exercise 2.6.17. *Let $m(x) = x^2 + x + 1 \in \mathbb{F}_2[x]$. Let $F = \mathbb{F}_2[x]/(m(x))$. Write down the addition and multiplication tables for F. Show that F is a field.*

Exercise 2.6.18. *Let $m(x) = x^2 + x + 1 \in \mathbb{F}_2[x]$. Let $F = \mathbb{F}_2[x]/(m(x))$. Show that F is isomorphic to the field in Example 2.1.19.*

Exercise 2.6.19. *State and prove a variant of Euler's Theorem 1.8.4. for polynomials in $F[x]/(m(x))$, where m is in $F[x]$ and F is a finite field. (Hint: What's the analog for Euler's ϕ function for polynomials?).*

Exercise 2.6.20. *Let $p_1(x) = x^3 + x + 1$ and $p_2(x) = x^3 + x + 2$ in $\mathbb{F}_5[x]$. Compute $(x+2)^{2114}$ in $\mathbb{F}_5[x]/(p_1(x))$ and in $\mathbb{F}_5[x]/(p_2(x))$. [Hint: one of p_1 and p_2 is irreducible, the other isn't. Use Exercise 2.6.19.]*

Exercise 2.6.21. *In high school algebra you learned how to "rationalize the denominator"; for instance, $\frac{1}{2+\sqrt{3}}$ is $\frac{2-\sqrt{3}}{4-3} = 2 - \sqrt{3}$. Express $\frac{1}{2 \cdot 4^{1/3} - 3}$ in terms of $2^{1/3}$. [Hint: What does this have to do with the inverse of $2x^2 - 3$ in $\mathbb{Q}[x]/(x^3 - 2)$? (and you know how to find this quickly, right?)]*

2.7 Companion matrices and extension fields

In this section, we finally explain how to contruct the fields mentioned in §2.1.3 above.

This section assumes some familiarity with linear algebra.

Definition 2.7.1. *If $p(x) = a_0 + a_1 x + ... + a_{n-1} x^{n-1} + x^n$ is a monic polynomial of degree n having coefficients in a ring F then the* **companion matrix** *of $p(x)$ is the $n \times n$ matrix*

$$C = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & -a_0 \\ 1 & 0 & 0 & 0 & 0 & -a_1 \\ 0 & 1 & 0 & 0 & 0 & -a_2 \\ \vdots & 0 & \ddots & 0 & 0 & \vdots \\ 0 & \dots & 0 & 1 & 0 & -a_{n-2} \\ 0 & \dots & 0 & 0 & 1 & -a_{n-1} \end{pmatrix}.$$

This matrix has the property that its characteristic polynomial is $p(x)$:

$$\det(C - xI_n) = p(x),$$

where I_n is the $n \times n$ identity matrix,

$$I_n = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ 0 & \dots & 1 & 0 \\ 0 & \dots & 0 & 1 \end{pmatrix}$$

Example 2.7.2. *Let $P(x) = x^2 + x + 1$. This is an irreducible polynomial over $\mathbb{F}_2$. The companion matrix of $p(x)$ is*

$$C = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}.$$

The set $\mathbb{F}_4 = \{0, C, C^2, I_2 = C^3\}$ is a field.

Example 2.7.3. *If $p(x) = x^4 + 9\,x^3 + 2\,x^2 + 17\,x + 5$ then its companion matrix is*

$$\begin{pmatrix} 0 & 0 & 0 & -5 \\ 1 & 0 & 0 & -17 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & -9 \end{pmatrix}$$

One of the main reasons for introducing the companion matrix at this stage is because of the following theorem.

Theorem 2.7.4. *Let F be a field and let $m(x) \in F[x]$ denote an irreducible monic polynomial in $F[x]$ of degree d. Let C denote the companion matrix of $m(x)$. Then under ordinary matrix addition and matrix multiplication, $F[C]$ is a field extension of F of degree d. Moreover, $F[x]/(m(x))$ is a field and the mapping $x^i \longmapsto C^i$ $(i = 0, 1, 2, ...)$ induces a field isomorphism $F[x]/(m(x)) \cong F[C]$.*

The proof of this fact uses the Cayley-Hamilton theorem from linear algebra (see for example [JN]) and therefore would take us a little too far afield to present here.

Example 2.7.5. *Let $F = \mathbb{F}_2$ and let $m(x) = x^3 + x + 1$. The multiplication table is*

$*$	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
0	0	0	0	0	0	0	0	0
1	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
x	0	x	x^2	x^2+x	$x+1$	1	x^2+x+1	x^2+1
$x+1$	0	$x+1$	x^2+x	x^2+1	x^2+x+1	x^2	1	x
x^2	0	x^2	$x+1$	x^2+x+1	x^2+x	x	x^2+1	1
x^2+1	0	x^2+1	1	x^2	x	x^2+x+1	$x+1$	x^2+x
x^2+x	0	x^2+x	x^2+x+1	1	x^2+1	$x+1$	x	x^2
x^2+x+1	0	x^2+x+1	x^2+1	x	1	x^2+x	x^2	$x+1$

The addition table is

$+$	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
0	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
1	1	0	$x+1$	x	x^2+1	x^2	x^2+x+1	x^2+x
x	x	$x+1$	0	1	x^2+x	x^2+x+1	x^2	x^2+1
$x+1$	$x+1$	x	1	0	x^2+x+1	x^2+x	x^2+1	x^2
x^2	x^2	x^2+1	x^2+x	x^2+x+1	0	1	x	$x+1$
x^2+1	x^2+1	x^2	x^2+x+1	x^2+x	1	0	$x+1$	x
x^2+x	x^2+x	x^2+x+1	x^2	x^2+1	x	$x+1$	0	1
x^2+x+1	x^2+x+1	x^2+x	x^2+1	x^2	$x+1$	x	1	0

Exercise 2.7.6. *In Example 2.7.5, show that* $\mathbb{F}_2[x]/(x^3+x+1)$ *is a field.*

Exercise 2.7.7. *If* A *is any* $n \times n$ *matrix with entries in a ring* F*, let* $F[A]$ *denote the set of all "polynomials in* A*", i.e., all matrices of the form* $a_0 I_n + a_1 A + a_2 A^2 + ... + a_k A^k$*, where* $k \geq 0$ *is any integer and* $a_i \in F$ *are arbitrary. Under ordinary matrix addition and matrix multiplication,* $F[A]$ *is a ring.*

Check this.

2.8 Polynomials in many variables

Some of the most important applications come from polynomials in several variables. Applications include robotics, error-correcting codes, cryptography, and geometry. We will only scratch the surface of this vast field in this book.

Think back to the long division algorithm for polynomials in one variable. When dividing $f(x)$ by $g(x)$, one of the first things to do is to identify the leading, or highest order, term of $f(x)$ and the leading term of $g(x)$. The expression "leading term of" has not yet been defined for polynomials of several variables. Motivated by the hope to generalize the division algorithm from one to several variables, we next discuss how to define the "leading term".

2.8.1 Monomials

First, what is a "term" of a polynomial of several variables? Let $x_1, ..., x_n$ be variables and let R be a ring. A **monomial** (or "term") is a polynomial of the form

$$x_1^{i_1} ... x_n^{i_n},$$

where the i_j's are non-negative integers. This expression is often abreviated using "multi-index notation" as $\underline{x}^{\underline{i}}$, where $\underline{i} = (i_1, ..., i_n)$ and $\underline{x} = (x_1, ..., x_n)$ and is a vector of variables. Note that the map

$$\begin{array}{rccc} \phi: & \mathbb{N}^n & \to & \{\text{monomials}\}, \\ & (i_1, ..., i_n) & \longmapsto & x_1^{i_1}...x_n^{i_n} \end{array}$$

defines a bijection from the set of n-tuples of non-negative integers to the set of monomials. In other words, there is a 1-1 correspondence between monomials and their vectors of exponents.

Though there may be some ambiguity as to how to divide one arbitrary polynomial by another, for the reasons discuessed above, when we reastrict our considerations to monomials, the ambiguities disappear. If R is a ring and

$$f(x_1, ..., x_n) = ax_1^{a_1}...x_n^{a_n} \in R[x_1, ..., x_n], \quad g(x_1, ..., x_n) = bx_1^{b_1}...x_n^{b_n} \in R[x_1, ..., x_n],$$

are monomials ($a_i \geq 0$, $b_j \geq 0$ are non-negative intgers) then f divides g, written $f|g$ if and only if $a|b$ in R and $a_i \leq b_i$ for all $1 \leq i \leq n$.

2.8.2 Leading terms

To say what a "leading term" is, we need to be able to order the monomials in some way. The amounts to the same thing as ordering the set $\mathbb{N}^n$. What is an "ordering"?

Definition 2.8.1. *Let S be a set and let $\leq: S \times S \to \{\text{true}, \text{false}\}$ be a function* [9]*. We call $\leq$ a* **total ordering** [10] *if*

- *$s \leq s$, ("reflective")*
- *if $s \leq t$ and $t \leq s$ then $s = t$, ("anti-symmetric")*
- *if $s \leq t$ and $t \leq u$ then $s \leq u$, ("transitive")*
- *for all $s, t \in S$, either $s \leq t$ or $t \leq s$.*

A function satisfying only the first three conditions above is called a **partial ordering**.

[9] Such a function is called a "relation on S".

[10] Some people also call this a **linear ordering**.

We shall only be interested in orderings on the monomials. In this case, there are some other conditions we shall want to add.

Definition 2.8.2. *Let M be the set of monomials $\underline{x}^{\underline{i}}$ in the variables $x_1, ..., x_n$. A relation $\leq$ on M is called a* **monomial ordering** *if, as an ordering on $\mathbb{N}^n$ it satisfies*

- *$\leq$ is a total ordering on $\mathbb{N}^n$,*
- *if $\underline{a} \leq \underline{b}$ and $\underline{c} \in \mathbb{N}^n$ then $\underline{a} + \underline{c} \leq \underline{b} + \underline{c}$,*
- *$\leq$ is a well-ordering [11] on $\mathbb{N}^n$.*

The following example is one of the most commonly used monomial orderings.

Example 2.8.3. *Define $(1, 0, ..., 0) \leq (0, 1, 0, ..., 0) \leq ... \leq (0, 0, ..., 0, 1)$. In general, we say $(i_1, i_2, ..., i_n) \leq (j_1, j_2, ..., j_n)$ if the first non-zero entry in $(j_1 - i_1, j_2 - i_2, ..., j_n - i_n)$ is positive.*

Another perspective: Think of $(1, 0, ..., 0) \in \mathbb{N}^n$ as the letter "a", $(2, 0, ..., 0) \in \mathbb{N}^n$ as the word "aa", ..., $(0, 1, 0, ..., 0) \in \mathbb{N}^n$ as the letter "b", $(1, 1, 0, ..., 0) \in \mathbb{N}^n$ as the word "ab", ...In general, we think of $(i_1, i_2, ..., i_n)$ as the word having i_1 "a"s, followed by i_2 "b"s, ... We say $(i_1, i_2, ..., i_n) \leq (j_1, j_2, ..., j_n)$ if the word for $(i_1, i_2, ..., i_n)$ occurs before the word for $(j_1, j_2, ..., j_n)$ in the dictionary.

This ordering is called the **lexicographical ordering**, *denoted $\leq_{lex}$ if there is an ambiguiuty.*

Lemma 2.8.4. *The lexicographical ordering is a monomial ordering.*

The proof is omitted (see Cox, Little, O'Shea [CLO], Proposition 4, Chapter 2, §2, for example).

The following example is another one of the most commonly used monomial orderings.

Example 2.8.5. *In general, we say $\underline{i} = (i_1, i_2, ..., i_n) \leq \underline{j} = (j_1, j_2, ..., j_n)$ if either*

$$|\underline{i}| < |\underline{j}|,$$

where $|\underline{i}| = i_1 + .. + i_n$, or, if $|\underline{i}| = |\underline{j}|$ and $(i_1, i_2, ..., i_n) \leq_{lex} (j_1, j_2, ..., j_n)$. This ordering is called the **graded lexicographical ordering** *or* **degree ordering**, *denoted $\leq_{grlex}$ if there is an ambiguiuty.*

[11] This means every non-empty subset has a smallest element with respect to $\leq$.

Lemma 2.8.6. *The graded lexicographical ordering is a monomial ordering.*

This proof is also omitted.

Let $<$ be a fixed monomial ordering on $F[x_1, ..., x_n]$. We define **leading term** of f (with respect to $<$) to be the largest of the monomial terms occurring in the expression for f with respect to $<$. Since the leading term is used so often, we introduce a short-hand notation for it. For $f \in F[x_1, ..., x_n]$, let $lt(f) = lt_<(f)$ denote the leading term of f (with respect to $<$).

2.8.3 Gröbner bases

Let $G = \{g_1, ..., g_k\} \subset F[x_1, ..., x_n]$ be a finite collection of polynomials with coefficients in some field F. Let $I = \langle G \rangle$ denote the ideal in $F[x_1, ..., x_n]$ generated by G.

Definition 2.8.7. *We say that G is a* **Gröbner basis** *for I if each $f \in I$ has a leading term (with respect to some fixed ordering on the set of monomials) which is divisible by the leading term of some element of G.*

A construction Gröbner bases (due to Buchburger) will be given later.

Definition 2.8.8. *Let $f, g \in F[x_1, ..., x_n]$ be non-zero polynomials. The* **S-polynomial** *of f, g is*

$$S(f,g) = \frac{lcm(lt(f), lt(g)}{lt(f)} \cdot f - \frac{lcm(lt(f), lt(g)}{lt(g)} \cdot g.$$

Reduction modulo a set

We have studied, back in chapter 1, the idea of reducing an integer modulo another integer. The concept which will be introduced here is similar, except the role of the integers is played by the ring of polynomials over a field F.

Definition 2.8.9. *Let $f, g \in F[x_1, ..., x_n]$ and fix a monomial ordering $<$. We say f* **reduced to** *h* **modulo** *g* **in one step** *if $lt(g)$ divides some term $ax_1^{a_1}...x_n^{a_n}$ of f and*

$$h = f - \frac{ax_1^{a_1}...x_n^{a_n}}{lt(g)} g.$$

In this case, we write $f \xrightarrow{g} h$.

Example 2.8.10. *Let $<$ be graded lexicographical ordering - terms are listed from highest to lowest degree and $x > y > z$.*

Let $f(x,y,z) = 2x^2z^4 + xz^3 + y^3z^3$ and $g(x,y,z) = xz^3 + y^2z^2$. Then $lt(f) = 2x^2z^4$ and $lt(g) = xz^3$. There are two terms of f which are divisible by $lt(g)$: $2x^2z^4$ and xz^3. In particular, f reduced to h modulo g in one step, where

$$\begin{aligned} h(x,y,z) &= f - \tfrac{lt(f)}{lt(g)}g \\ &= f - (2xz)g \\ &= 2x^2z^4 + xz^3 + y^3z^3 - 2x^2z^4 - 2xy^2z^3 \\ &= -2xy^2z^3 + y^3z^3 + xz^3. \end{aligned}$$

Definition 2.8.11. *Let $f, <$ be as in the previous definition. Let $G \subset F[x_1, ..., x_n]$ be a finite set. We say f* **reduced to** *h* **modulo** *G if there is a finite sequence of polynomials $h_0 = f, h_1, ..., h_k = h$ such that h_i reduces to h_{i+1} modulo some $g \in G$ (depending on h_i) in one step, $0 \leq i \leq k-1$. In this case, we write $f \xrightarrow{G} h$.*

We say that h is in **normal form** *modulo G if is no h such that $h \xrightarrow{G} h'$. We say h is the* **reduced normal form of** *f* **modulo** *G if f reduces to h modulo G and h is in normal form modulo G.*

The normal form of f modulo G, if it exists, is denoted $NF(f, G)$.

2.8.4 Buchburger's algorithm

The following algorithm was given by Buchburger in his 1965 PhD thesis (as a student of Gröbner at the Univ. of Innsbruck, Austria). Our discussion follows Buchburger [Bu] and Nakos-Glinos [NG].

Input: A finite set F of polynomials generating an ideal I in $R[x_1, ..., x_n]$.

Output: A Gröbner basis for I.

Let

$$P := \{\ \{f, g\} \mid f, g \in F,\ f \neq g\}.$$

(a) Pick a pair $\{f, g\}$ in P.

(b) Let $P = P - \{f, g\}$. Compute $NF(S(f, g), F)$.

(c) If $NF(S(f, g), F) = 0$ and if P is non-empty then go to (a). If $NF(S(f, g), F) = 0$ and there are no more elements in P then stop and output F. If $NF(S(f, g), F) \neq 0$, let $F = F \cup \{\text{NF(S(f,g),F)}\}$.

(d) Go to (a).

Example 2.8.12. *Let $<$ be the graded lexicographic ordering on $\mathbb{Q}[x,y]$. Let $f_1(x,y) = xy - 2x$, $f_2(x,y) = x^2 - y$, $G = \{f_1, f_2\}$. We have $S(f_1,f_2) = y^2 - 2x^2$ and $y^2 - 2x^2 \xrightarrow{G} y^2 - 2y$, so $NF(S(f_1,f_2)) = y^2 - 2y$. Let $f_3(x,y) = y^2 - 2y$ and $G = \{f_1, f_2, f_3\}$. We have $S(f_1,f_2) = 0$, so G is not increased. We have $S(f_1,f_3) \xrightarrow{G} 0$, so, again, no more is to be added to G. The Gröbner basis for the ideal $I = (f_1,f_2) \subset \mathbb{Q}[x,y]$ is $G = \{f_1,f_2,f_3\}$.*

2.8.5 Applications

Gröbner bases have applications to robotics, algebraic geometry, linear programming, chemistry, error-correcting codes, to name a few. However, each example is rather technical, so we shall stick to a few, very simple cases.

Algebraic curves

To find the equations of an algebraic curve parametrized by $x = f(t), y = g(t), z = h(t)$, where f, g, h are polynomials,

- Form the ideal $I \subset F[x,y,z,t]$ generated by $x - f(t)$, $y - g(t)$, $z - h(t)$.
- "Project this ideal onto $F[x,y,z]$": Compute the Gröbner basis of this ideal, using an ordering for which $x > y > z > t$. Choose only those terms in the basis whcih do *not* involve t.
- These terms will correspond to the equations for the curve in s dimensions.

Example 2.8.13. *The twisted cubic is parameterized by $x = t$, $y = t^2$, $z = t^3$. The object is to find a Gröbner basis for the ideal $I = (x - t, y - t^2, z - t^3)$ in $\mathbb{Q}[x,y,z,t]$. Let $<$ denote the lexicographic order defined by $t < z < y < x$. The Gröbner basis for I is $G = \{y - x^2, z - x^3, -x + t\}$.*

Therefore the algebraic equations for the curve are

$$y = x^2, \ z = x^3.$$

Example 2.8.14. *Let $x = t^2 + 2t - 5$, $y = t^4 + 4t^3 + 7t^2 + 6t - 4$, $z = 100 - 70t - 23t^2 + 12t^3 + 3t^4$ yields the ideal $I = (x - t^2 - 2t + 5, y - t^4 - 4t^3 - 7t^2 - 6t + 4, z - 100 + 70t + 23t^2 - 12t^3 - 3t^4)$ which has the Grobner basis $\{-32 - 13x - x^2 + y, 5x - 3x^2 + z, 5 - 2t - t^2 + x\}$. So the equations for the curve are $y = x^2 + 13x + 32$, and $z = 3x^2 - 5x$.*

Algorithm to find the GCD of $f, g \in F[x, y]$

Input: $f, g \in F[x, y]$.
Output: $gcd(f, g)$.

- Compute the Gröbner basis G of the ideal $I = \langle tf, (1-t)g \rangle$ with respect to the ordering $t < x < y$.
- Take p in G such that p involves only x and y. If there is no such p in G then
$$GCD(f, g) = 1.$$
Otherwise
$$GCD(f, g) = fg/p.$$

Example 2.8.15. *Let $f(x, y) = x^2 - y^2$ and $g(x, y) = x^3 - y^3$. The Gröbner basis of $I = (tf(x, y), (1 - t)g(x, y))$ using the lexicographic ordering defined by $t < x < y$ is:*

$$G = \{-yx^3 + y^4 - x^4 + xy^3, ytx^2 + x^3 - y^3 - tx^3, -tx^2 + ty^2\}.$$

Taking $p = -yx^3+y^4-x^4+xy^3$, we have $gcd(f, g) = f(x, y)g(x, y)/p = y-x$.

Exercise 2.8.16. *Consider a curve parameterized by $x = t$, $y = 2t^2$, $z = t^5$. Using a Gröbner basis, find the algebraic equations for the curve.*
(Ans: [$y = 2x^2$, $z = x^5$.)

Exercise 2.8.17. *Let $f(x, y) = x^2 - 4y^2$ and $g(x, y) = x^2 - 4xy + 4y^2$. Using a Gröbner basis, find the gcd of f and g.*
(Ans: $x - 2y$.)

Exercise 2.8.18. *Let f, g be monomials as above. Based on the analog with definitions on $\mathbb{N}$, define the greatest common divisor, $gcd(f, g)$ and the least common multiple, $lcm(f, g)$, of f, g.*

Exercise 2.8.19. *Let $f(x_1, x_2) = x_1^4 x_2^3$ and $g(x_1, x_2) = x_1^2 x_2^6$. Find $gcd(f, g)$ and $lcm(f, g)$.*

Exercise 2.8.20. *Let $f(x_1, x_2, x_3) = x_1 + x_2 + x_3 + x_2x_3$, $g(x_1, x_2, x_3) = x_1 + x_2$. Find $S(f, g)$, where*
(a) $<$ is the lexicographical ordering,
(b) $<$ is the graded lexicographic ordering.

Exercise 2.8.21. *Let* $f(x_1, x_2, x_3) = x_1 - 13x_2^2 - 12x_3^2$, $g(x_1, x_2, x_3) = x_1^2 - x_1x_2 + 92x_3$. *Find* $S(f, g)$, *where*
(a) $<$ *is the lexicographical ordering,*
(b) $<$ *is the graded lexicographic ordering.*

Exercise 2.8.22. *Let* $<$ *be the graded lexicographic ordering. Let* $f_1(x, y) = xy - 2x$, $f_2(x, y) = x^2 - y$, $f_3(x) = y^2 - 2y$. *Show* $S(f_1, f_2) = y^2 - 2x^2$, $S(f_1, f_3) = 0$.

Exercise 2.8.23. *Find* F *be the smallest field that contains* $\mathbb{Q}$, $\sqrt{3}$, *and* $i6^{1/3}$. *Prove that* F *can be generated by just the single element* ($\alpha = \sqrt{3} + i6^{1/3}$ *(so that the smallest field containing* $\mathbb{Q}$ *and* α *is exactly* F*), by expressing both* $\sqrt{3}$ *and* $i6^{1/3}$ *as polynomials in* α. *(Hint: What does the Groebner basis of* $(x^2 - 3, y^2 + 1, z^3 - 6, w - x - yz)$ *have to do with this problem?)*

Exercise 2.8.24. *Let* $<$ *be the graded lexicographic ordering. Let* $f_1(x, y) = xy - 2x$, $f_2(x, y) = x^2 - y$, $G = \{f_1, f_2\}$. *Show* $y^2 - 2x^2 \xrightarrow{G} y^2 - 2y$.

Exercise 2.8.25. *Let* $<$ *be graded lexicographical ordering - terms are listed from highest to lowest degree and* $x > y > z$. *Let* $f(x, y, z) = 2x^2z^4 + xz^3 + y^3z^3$. *Order the terms of* f *from highest to lowest.*

Exercise 2.8.26. *Let* $f(x_1, x_2, x_3) = x_1 + x_2 + x_3 + x_2x_3$. *Find* $lt_<(f)$, *where*
(a) $<$ *is the lexicographical ordering,*
(b) $<$ *is the graded lexicographic ordering.*

Exercise 2.8.27. *Find the polynomial of smallest degree with integer coefficients that has* $4 - \frac{2-5^{1/3}}{3-2i}$ *as a root. (Hint: What does the Grobner basis of* $(x^3 - 5, y^2 + 1, z(3 - 2y) - (2 - x))$ *have to do with this problem?)*

2.9 Special project: Nimbers

John Conway [C] discovered that there is a collection of objects derived from combinatorial games which forms a field, called here the field of **nimbers**. We shall construct the set of nimbers, giving some examples along the way, but we shall not prove that it is a field.

First, what do we mean precisely by a "game"? A **two person game** is a sequence of moves played alternately between two players following certain rules satisfying

- there are finitely many moves at each stage,
- there is a finite sequence of moves which yields a win for exactly one of the players and after each move the number of options for each player has strictly decreased,
- there are no chance or random moves,
- there is complete information about each move,
- each move depends only on the present position, not on the existence or non-existence of a certain previous move (such as chess, where castling is made illegal if the king has been moved previously),
- the player who has no legal moves when it is their turn loses.

Sometimes we slightly abuse language and identify a game with a position P of that game (it being implicitly assumed that it is known whose turn it is to move). Likewise, we may identify any move with the board position which occurs *after* the move is made. We shall call the two players **Left** (or Laura) and **Right** (or Robert). We identify a game position P with its collection of Left options (the legal moves Left can make if it was her move), denoted P^L, and Right options (the legal moves Right can make if it was his move), denoted P^R, and we write $P = \{P^L \mid P^R\}$.

If the move options for a position P is a disjoint union of move options for subpositions P_1, P_2, ..., P_k, then we say P is the **sum** of the games P_1, ..., P_k as we write

$$P = P_1 + P_2 + ... + P_k.$$

Sums of games which are not necessarily disjoint are defined later.

Example 2.9.1. *The game of* **Domineering**. *In this game, two players alternate by placing certain tiles on a certain board according to certain rules.*

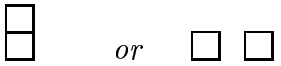
The **board**, *which may be regarded as the initial starting position of the game, is a subset of an $m \times n$ piece of graph paper. For example,*

is a possible starting position. A **tile** *is a* 1×2 *subgrid or a* 2×1 *subgrid of the board position.*

Left may place a 1×2 *tile (a horizonal domino) anywhere on the board provided it does not overlap any other square which has already been played on. Right may place a* 2×1 *tile (a vertical domino) anywhere on the board provided it does not overlap any other square which has already been played on.*

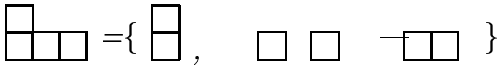
The only possible move Right has (from the starting position) is to remove the 2 vertical tiles, resulting in

The only possible moves Left has (from the starting position) is to remove 2 horizonal tiles tiles, resulting in



The latter move is best if Left wants to win, since Right would be left with no legal moves and therefore would loses the game.

Using the above notation, we may compactly describe this set up as



Numerical notation

Some further notation will help us describe the possibilities a little better.

(0) If neither player has a legal move in a given position, then we denote that position by

$$0 = \{ \mid \} = \{\emptyset|\emptyset\},$$

where $\emptyset$ denotes the empty set. In the game of Domineering, this occurs if there is only one square remaining:

In this position, the player to move loses.

(1) If Left has exactly one legal move in a given position, and Right has none, then we denote that position by

$$1 = \{0| \} = \{\{\emptyset|\emptyset\}| \}.$$

In the game of Domineering, this occurs if there are two or three consecutive squares in a horizonal row remaining (and any number of isolated squares):

In this position, Left wins no matter who plays first.

(-1) If Right has exactly one legal move in a given position, and Left has none, then we denote that position by

$$-1 = \{ |0\} = \{ \mid \{\emptyset|\emptyset\}\}.$$

In the game of Domineering, this occurs if there are two or three consecutive squares in a vertical row remaining (and any number of isolated squares):

In this position, Right wins no matter who plays first.

(*) If Left and Right each have exactly one legal move in a given position, then we denote that position by

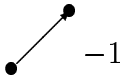
$$* = \{\ 0|0\} = \{\{\emptyset|\emptyset\}|\ \{\emptyset|\emptyset\}\}.$$

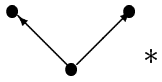
In the game of Domineering, this occurs if

In this position, whoever plays first wins.

Tree notation

If we denote the starting position P_0 by the root node in a tree graph, each Left option by a arrow pointing towards the left to nodes representing a new positions P^L, each Right option by an arrow pointing towards the right to nodes representing a new positions P^R, then the numerical notation can be drawn as follows.





We define

$$-\{\ |\ \} = 0,$$
$$-\{\ 0\ |\ \} = \{\ |\ 0\} = -1,$$
$$-\{\ |0\} = \{\ 0\ |\ \} = 1,$$
$$-\{0\ |0\} = \{0\ |0\} = *.$$

In general, we define

$$-P = -\{P^L \mid P^R\} = \{-P^R \mid -P^L\},$$

where we may assume (by induction) that $-P^L$ and $-P^R$ have already been defined (they are games with a smaller number of options).

We define

$$\{\ |\ \} + \{\ |\ \} = \{\ |\ \}, \qquad (0+0=0)$$
$$\{\ |\ \} + \{0\ |\ \} = \{0|\ \}, \qquad (0+1=1)$$
$$\{\ |\ \} + \{\ |0\} = \{\ |0\}, \qquad (0+(-1)=-1)$$
$$\{\ |\ \} + \{0\ |0\} = \{0\ |0\}, \qquad (0+*=0).$$

If $P = \{P^L \mid P^R\}$ and $Q = \{Q^L \mid Q^R\}$ are two games then we define

$$P + Q = \{P^L + Q, P + Q^L \mid P^R + Q, P + Q^R\},$$

where we assume (by induction) that $P^L + Q, ..., P + Q^R$ have already been defined. This is interpreted to mean that if, for example, P^L is empty then $P^L + Q$ is to be ignored. For example,

$$\{0|\ \}+\{\ |0\} = \{0+\{\ |0\}|\{0|\ \}+0\} = \{\{\ |\ \}+\{\ |0\}|\{0|\ \}+\{\ |\ \}\} = \{\ |\ \}, \qquad (1+(-1)=0).$$

It is clear from the definition that $P + Q = Q + P$.

We define

$$\begin{aligned}\{\ |\ \}\cdot\{\ |\ \} &= \{\ |\ \}, \quad (0\cdot 0=0)\\ \{\ |\ \}\cdot\{0\ |\ \} &= \{\ |\ \}, \quad (0\cdot 1=0)\\ \{\ |\ \}\cdot\{\ |0\} &= \{\ |\ \}, \quad (0\cdot(-1)=0)\\ \{\ |\ \}\cdot\{0\ |0\} &= \{\ |\ \}, \quad (0\cdot *=0).\end{aligned}$$

If $P=\{P^L\mid P^R\}$ and $Q=\{Q^L\mid Q^R\}$ are two games then we define

$$\begin{aligned}P\cdot Q=\{&P^L\cdot Q+P\cdot Q^L-P^L\cdot Q^L, P^R\cdot Q+P\cdot Q^R-P^R\cdot Q^R\\ &\mid P^L\cdot Q+P\cdot Q^R-P^L\cdot Q^R, P^R\cdot Q+P\cdot Q^L-P^R\cdot Q^L\},\end{aligned}$$

where we assume (by induction) that $P^L\cdot Q+P\cdot Q^L-P^L\cdot Q^L,...,P^R\cdot Q+P\cdot Q^L-P^R\cdot Q^L$ have already been defined. This is interpreted to mean that if, for example, P^L is empty then $P^R\cdot Q+P\cdot Q^L-P^R\cdot Q^L$ is to be ignored. For example, $1\cdot 1=1$ since

$$\{0\mid\}\cdot\{0\mid\}=\{0\cdot\{0\mid\}+\{0\mid\}\cdot 0-0\cdot 0\mid\}=\{0\mid\}.$$

If $P=\{P^L\mid P^R\}$ is non-zero then we define $Q=P^{-1}$ by

$$\begin{aligned}P^{-1}=\{0,&\tfrac{1+(P^L-P)\cdot Q^L}{P^R},\tfrac{1+(P^L-P)\cdot Q^R}{P^L}\\ &\mid \tfrac{1+(P^L-P)\cdot Q^L}{P^L},\tfrac{1+(P^R-P)\cdot Q^R}{P^R}\},\end{aligned}$$

where we assume (by induction) that $\frac{1+(P^L-P)\cdot Q^L}{P^R}$ have already been defined. This is interpreted to mean that if, for example, P^L is empty *or* $P^R=0$ then $\frac{1+(P^L-P)\cdot Q^L}{P^R}$ is to be ignored. For example, $1^{-1}=1$ since

$$\{0\mid\}^{-1}=\{0,\frac{1+(0-\{0\mid\})\cdot Q^L}{0}\mid\frac{1+(0-\{0\mid\})\cdot Q^L}{0}\}=\{0\mid\}.$$

2.10 Special project: factoring over $\mathbb{C}$

In the case where F is the complex numbers, let use $z=x+iy$ as our variable name.

The most important result about a polynomial p over the complex numbers is that it will have exactly $\deg(p)$ roots. In other words, we have the following result.

Theorem 2.10.1. *(Fundamental Theorem of Algebra) If $p(z)$ is a polynomial of degree n in $\mathbb{C}[z]$ then there are n complex roots $r_1,...,r_n$ (counted according to multiplicity) such that $p(z)=a(z-r_1)...(z-r_n)$, where a is the leading coefficient of $p(z)$.*

C. F. Gauss was the first to give a complete proof of this fact. It's simplest proof (as far as we know of) uses complex-analytic techniques which go beyond the scope of this book. A topological argument is given in [Ar], ch 13, §9.

Corollary 2.10.2. *If $p(z)$ is an irreducible polynomial in $\mathbb{C}[x]$ then $p(z) = az + b$, for some $a, b \in \mathbb{C}$.*

To factor a polynomial $p(z)$ over the complex numbers, one may use the following procedure.

- Eliminate any multiple factors by dividing $p(z)$ by $gcd(p(z), p'(z))$, as in Lemma 2.4.6.
- Find a root of the resulting polynomial.
- Use the division algorithm to factor $p(x)$ into a linear factor times a polynomial of lower degree.
- Repeat this process until $p(z)$ is completely factored.

Example 2.10.3. *If $p(z) = z^n - 1$ then all n roots of $p(z)$ lie equally distributed on the unit circle in the complex plane. In fact, and complex number $z = x + iy$ may be written in its* **polar decomposition** *$z = re^{i\theta}$, where $r = \sqrt{x^2 + y^2}$ and $\theta = arg(z)$ is the "argument" of z. If $z = re^{i\theta}$ is a root of $z^n = 1$ then $r^n e^{n\theta} = 1$, then $r = 1$ (which implies that such a root is on the unit circle) and $n\theta$ is a multiple of 2π. The n distinct numbers $z_1 = 1$, $z_2 = e^{2\pi i/n}$, $z_3 = e^{4\pi i/n}$, ..., $z_n = e^{2\pi i(n-1)/n}$ satisfy $z_i^n = 1$. By the fundamental theorem of algebra, they must be all the roots of $p(z)$.*

In case $n = 2$, this implies that the roots are $1, -1$.
In case $n = 3$, this implies that the roots are $1, \frac{-1+\sqrt{3}i}{2}, \frac{-1-\sqrt{3}i}{2}$.
In case $n = 4$, this implies that the roots are $1, -1, i, -i$.

2.10.1 Explicit formulas for the roots

There are explicit formulas for the roots of any polynomial of degree 4 or less in terms of radicals. It was proven by N. Abel in 1824 that no such formula exists for degrees 5 or above. Around the same time period, E. Galois classified which polynomials had the property that its roots could be given by radicals.

Quadratic formula

For example, the **quadratic formula** states that the roots to $ax^2+bx+c=0$ are given by

$$r = \frac{-b \pm \sqrt{b^2-4ac}}{2a}.$$

Cubic formula

The **cubic formula** states that the roots of $x^3+ax+b=0$ are of the form

$$r = -A-B, -A\omega - B\omega^2, -A\omega^2 - B\omega,$$

where $\omega = -\frac{1}{2} + \frac{\sqrt{3}i}{2}$, and

$$A = (\frac{b}{2} + (\frac{b^2}{4} + \frac{a^3}{27})^{1/2})^{1/3},$$

$$B = (\frac{b}{2} - (\frac{b^2}{4} + \frac{a^3}{27})^{1/2})^{1/3},$$

Any cubic polynomial y^3+py^2+qy+r can be reduced to the form above using the substitution $y = x - \frac{p}{3}$, where

$$a = q - \frac{p^2}{3}, \qquad b = \frac{2p^3 - 9pq + 27r}{27}.$$

There is a similar (though much more complicated) general formula for the roots of quartic polynomials.

Exercise 2.10.4. *Derive the above formula for the solution of the cubic. [Hint: One approach is the following. Make the substitution $x = m+n$, so $(m+n)^3 + a(m+n) + b = 0$. If $m^3+n^3+b=0$ and $3mn+a=0$ then $m^3 + 3mn(m+n) + n^3 + a(m+n) + b = 0$. Solve for m, n in terms of a, b.*

Quartic formula

We shall only sketch the rough idea, since the formula itself is rather complicated.

Let

$$p(x) = x^4 + a_3x^3 + a_2x^2 + a_1x + a_0.$$

Complete the square of the first few terms to get

$$\left(x^2 + 1/2\, a_3\, x\right)^2 = \left(1/4\, {a_3}^2 - a_2\right) x^2 - a_1\, x - a_0.$$

Adding $2(x^2 + (a_3/2)x)y + y^2$ to both sides and collecting terms, this is

$$(x^2 + (a_3/2)x + y)^2 = (a_3^2/4 - a_2 + 2y)x^2 + (a_3 y - a_1)x + y^2 - a_0.$$

Write the right-hand side in the form $ax^2 + bx + c$. We want to choose y so that $ax^2 + bx + c = a(x - r)^2$, for some r. To do this we want (by the quadratic formula) $b^2 - 4ac = 0$. This condition is

$$-8\, y^3 + 4\, a_2\, y^2 + (-2\, a_3\, a_1 + 8\, a_0)\, y - 4\, a_2\, a_0 + {a_1}^2 + {a_3}^2 a_0 = 0,$$

a cubic equation which can be solved using the cubic formula. In any case, we can now determine the root r above and write

$$x^2 + (a_3/2)x + y = \pm\sqrt{a}(x - r),$$

where r and y were solved for previously. There are 4 roots to these two equations (one for each sign of the $\pm$ sign).

The cubic and quartic formulas were discovered in the middle ages.

Higher degree case

There is no such general formula for the roots of polynomials of degree 5 or greater. This was proven by an astonishing display of ingenuity and creativity by E. Galois before he died at the age of 23. We will not prove this fact here, however, see §5.17.1 below. For the curious, the polynomial $x^5 - x + 1$ has roots which cannot be expressed in terms of radicals of rational numbers, whereas $x^5 - 5x + 12$ has roots which can so be expressed.

2.10.2 Factoring $x^n - 1$ over $\mathbb{C}$

In other words, we shall express $x^n - 1$ as a product of irreducible polynomials having coeficients in $\mathbb{C}$.

This is, in fact, very easy. The complex roots of $x^n - 1$ are precisely the n^{th} roots of unity: those numbers z for which

$$z^n = 1.$$

To determine these, recall that any complex number z may be written in its polar decomposition, $z = re^{i\theta}$, where $r \geq 0$ and $0 \geq \theta < 2\pi$ are it's "polar coordinates". (Moreover, if $r > 0$ then these coordinates are unique.) Then $z^n = 1$ implies $r^n e^{in\theta} = 1$, which in turn implies $r^n = 1$ (so $r = 1$ since $r \geq 0$) and $n\theta \in \{0, \pm 2\pi, \pm 4\pi, ...\}$. Substituting these into $z = re^{i\theta}$, we find that z must be equal to one of the following numbers:

$$\zeta_n^0 = 1, \quad \zeta_n = e^{2\pi i/n}, \quad \zeta_n^2 = e^{4\pi i/n}, \quad \zeta_n^3 = e^{6\pi i/n}, \ldots .$$

In fact, this sequence is periodic, so only the first n terms are distinct and after that the numbers start repeating themselves. We summarize this as the following result.

Lemma 2.10.5. *The complex roots of* $x^n - 1 = 0$ *are*

$$\zeta_n^0 = 1, \quad \zeta_n = e^{2\pi i/n}, \quad \zeta_n^2 = e^{4\pi i/n}, ..., \zeta_n^{n-1} = e^{2(n-1)\pi i/n}.$$

These may be visualized as follows. In general, plot the complex number

$$z = x + iy = re^{i\theta} = r\cos(\theta) + ir\sin(\theta)$$

at the point $(x, y) = (r\cos(\theta), r\sin(\theta))$ in the xy-plane. (Sometimes this way of plotting complex numbers is called the **Gaussian plane**.) The n roots of $x^n - 1 = 0$ are equally spaced points on the unit circle, starting at $z = 1$.

In fact,

$$x^n - 1 = (x - \zeta_n^0)(x - \zeta_n^1)...(x - \zeta_n^{n-1}) = \prod_{j=0}^{n-1}(x - \zeta_n^j)$$

is the factorization into irreducibles opver $\mathbb{C}$. In other words, factoring $x^n - 1$ over $\mathbb{C}$ amounts geometrically to subdividing the circle into n equal parts.

Remark 2.10.6. *The ancient Greeks asked for which n is it possible to subdivide the circle into n equal parts using only a ruler and compass. This, and its connection with factoring polynomials, is discussed in further detail in chapter 22 of Schroeder [Sch].*

2.11 Special project: Factoring over $\mathbb{R}$

The most important result about a polynomial p over the real numbers is that it will have at least $\deg(p)/2$ irreducible factors. In other words, we have the following result.

Theorem 2.11.1. *If $p(x)$ is a polynomial of degree n in $\mathbb{R}[x]$ then for each complex roots $r = \alpha + i\beta$ of $p(x)$ the conjugate $\overline{r} = \alpha - i\beta$ is also a root. Moreover, $p(x) = a(x - r_1)...(x - r_k)q_1(x)...q_\ell(x)$, where $0 \leq k \leq n$ (in which case we say that $p(x)$ splits or factors completely over $\mathbb{R}$), $0 \leq \ell \leq n/2$ and each $q_i(x)$ is an irreducible quadratic polynomial in $\mathbb{R}[x]$.*

This actually follows from the fundamental theorem of algebra, though we shall delay its proof until a later chapter.

Corollary 2.11.2. *If $p(x)$ is an irreducible polynomial in $\mathbb{R}[x]$ then $p(x)$ has either degree 1 (with one real root) or degree 2 (with no real roots).*

In fact, if $\alpha + i\beta$ is any complex root of a polynomial in $\mathbb{R}[x]$ then its complex conjugate $\alpha - i\beta$ must also be a root. Moreover, $x^2 - 2x\alpha + \alpha^2 + \beta^2$ is an irreducible factor of $p(x)$.

To factor a polynomial $p(x)$ over the real numbers, one may use the following procedure.

- Eliminate any multiple factors by dividing $p(x)$ by $gcd(p(x), p'(x))$, as in Lemma 2.4.6.
- Find a (possible complex) root of the resulting polynomial.
- If the root is complex, say $\alpha + i\beta$, then use the division algorithm to factor $p(x)$ into the quadratic factor $x^2 - 2x\alpha + \alpha^2 + \beta^2$ times a polynomial of lower degree. If the root is real use the division algorithm to factor $p(x)$ into a linear factor times a polynomial of lower degree.
- Repeat this process until $p(x)$ is completely factored.

2.12 Special Project: Factoring over $\mathbb{Q}$ or $\mathbb{Z}$

Factoring a polynomial in $\mathbb{Q}[x]$ is essentially equivalent to polynomial in $\mathbb{Z}[x]$, since one can always multiply by a suitable large integer to clear all the

denominators. However, the situation is even better than this. Remarkably enough, it is not possible to factor a polynomial with integer coefficients into polynomials with rational (and non-integral) coefficients. This fact is proven in the next subsection.

2.12.1 Primitive polynomials

If all the coefficients of a polynomial with integer coefficients are divisible by an integer $d \neq \pm 1$ then we say that the polynomial is **imprimitive**. Otherwise, the polynomial is called **primitive**. For example, $2x^{100} - 68x^{47} - 2$ is imprimitive but $x + 1$ is primitive.

Theorem 2.12.1. *(Gauss' lemma) The product of primitive polynomials is primitive.*

proof: Let p be a prime. Let $f(x) = a_0 + a_1 x + ... + a_m x^m$ and $g(x) = b_0 + b_1 x + ... + b_n x^n$ be primitive. Let a_s be the smallest coefficient of $f(x)$ not divisible by p and let b_r be the smallest coefficient of $g(x)$ not divisible by p. We have

$$f(x)g(x) = c_0 + c_1 x + ... + c_{r+s} x^{r+s} + ... + c_{m+n} x^{m+n},$$

where $c_{r+s} = b_0 a_{r+s} + ... + b_r a_s + ... + b_{r+s} a_0$. By our assumption, p does not divide c_{r+s}. Moreover, c_{r+s} is coefficient of $g(x)h(x)$ least degree which is not divisible by p. This implies that $f(x)g(x)$ is primitive. $\square$

This lemma is used to show the following theorem.

Theorem 2.12.2. *If $p(x) \in \mathbb{Z}[x]$ and $p(x)$ factors into a product of irreducible polynomials with coefficients in $\mathbb{Q}$ then $p(x)$ factors into a product of irreducible polynomials with coefficients in $\mathbb{Z}$.*

proof: Suppose $p(x) = f(x)g(x)$, where $f(x), g(x) \in \mathbb{Q}[x]$. We can factor out common factors of the denomiators and numerators of the coefficients and put this in the form $p(x) = \frac{a}{b} f^*(x) g^*(x)$, where $f^*(x), g^*(x) \in \mathbb{Z}[x]$ are primitive polynomials, and a, b are relatively prime integers. By Gauss' lemma, $f^*(x)g^*(x)$ is also primitive. Since $p(x)$ has integer coefficeints, this implies $b = 1$. $\square$

2.12.2 Factoring strategies

One useful idea is to factor $p(x)$ over a finite field $\mathbb{F}_p$, where p is a very large prime, and all the representatives of the finite field are choosen to lie between $-p/2$ and $p/2$. The prime p must be choosen so large that the coefficients of $p(x)$ and all its factors are in between $-p/2$ and $p/2$. Some techniques for factoring over a finite field are described in the next section.

Irreducibility criteria

One method of determining whether a given polynomial in $\mathbb{Z}[x]$ is irreducible over $\mathbb{Z}$ is the following test.

Theorem 2.12.3. *(Eisenstein's criterion) If all the coefficients (except the leading coefficient) are divisible by a prime p, and the constant term coefficient is not divisible by p^2, then the polynomial is irreducible over $\mathbb{Z}$.*

proof: Let $f(x) = a_n x^n + ... + a_1 x + a_0 \in \mathbb{Z}[x]$ satisfy the hypotheses of the theorem. By assumption, $p|a_i$ for $0 \leq i < n$, $p \not| a_n$, and $p^2 \not| a_0$.

Suppose, to get a contradiction, that $f(x) = g(x)h(x)$, where $g(x) = b_k x^k + ... + b_1 x + b_0 \in \mathbb{Z}[x]$, $h(x) = c_m x^m + ... + c_1 x + c_0 \in \mathbb{Z}[x]$. Let $b_i = 0$ if $i > k$ and $c_j = 0$ if $j > m$. In general, we have

$$a_t = b_t c_0 + b_{t-1} c_1 + ... + b_1 c_{t-1} + b_0 c_t. \tag{2.2}$$

In particular, $a_0 = b_0 c_0$. Since $p|a_0$ but $p^2 \not| a_0$, we may assume without loss of generality that $p|b_0$ and $p \not| c_0$. Let t be the smallest index such that p does not divide b_t. We know $t \neq 0$. But then p divides all but the first term of the right-hand side of (2.2) and p divides the left-hand side. This is a contradiction. □

Example 2.12.4. *Let $p(x) = 2x^3+3x^2+9x+12$. By the Eisenstein criterion, $p(x)$ is irreducible.*

Another method of determining whether a given polynomial in $\mathbb{Z}[x]$ is irreducible over $\mathbb{Z}$ is to use the following test.

Theorem 2.12.5. *Given is a polynomial $f(x)$ of degree n in $\mathbb{Z}[x]$. If there are $2n+1$ integers k such that $f(k)$ is a prime number then then $f(x)$ is irreducible over the integers.*

proof: If $f(x) = g(x)h(x)$, where $g, h \in \mathbb{Z}[x]$, then $f(k)$ prime implies $g(k) = \pm 1$ or $h(k) = \pm 1$. By the fundamental theorem of algebra, there are at most $2\deg(g(x))$ solutions to $g(x) = \pm 1$ and at most $2\deg(h(x))$ solutions to $h(x) = \pm 1$. $\square$

Example 2.12.6. *This is easy to implement on in a computer algebra system. For example, if we want to test $p(x) = x^3 + 3x + 9$ (which fails the Eisenstein criterion) then out of the integers $k = 1, 2, ..., 16$, there are 7 k's such that $p(k)$ is a prime ($k = 1, 2, 5, 7, 10, 11, 16$). By the previous test, $p(x)$ is irredecible over the integers.*

In 1857 Bouniakowsky conjectured that if $p(x)$ is an irreducible polynomial in $\mathbb{Z}[x]$ such that no number greater than 1 divides all the values of $p(i)$ for every integer i, then $p(i)$ is prime for infinitely many integers i.

2.12.3 Factoring $x^n - 1$ over $\mathbb{Q}$

In other words, we shall express $x^n - 1$ as a product of irreducible polynomials having coefficients in $\mathbb{Q}$.

This is harder - the constraint "having coefficients in $\mathbb{Q}$" is more restrictive than "having coefficients in $\mathbb{C}$". First, let us consider some examples:

$n = 1$ $x - 1$ is already irreducible.

$n = 2$ $x^2 - 1 = (x - 1)(x + 1)$ is the factorization into irreducibles over $\mathbb{Q}$.

$n = 3$ $x^3 - 1 = (x - 1)(x^2 + x + 1)$ is the factorization into irreducibles over $\mathbb{Q}$.

Exercise 2.12.7. *Show $x^2 + x + 1$ is irreducible over $\mathbb{Q}$.*

$n = 4$ $x^4 - 1 = (x^2 - 1)(x^2 + 1) = (x - 1)(x + 1)(x^2 + 1)$ is the factorization into irreducibles over $\mathbb{Q}$.

Exercise 2.12.8. *Show $x^2 + 1$ is irreducible over $\mathbb{Q}$.*

In general, we have a "formula" for the irreducible factors of $x^n - 1$. To state this we need a definition.

Definition 2.12.9. *Let $n = p_1^{e_1}...p_k^{e_k}$ be the prime factorization of an integer $n > 1$. Define the* **Möbius function** *$\mu : \mathbb{N} \to \{0, -1, 1\}$ by: $\mu(1) = 1$ and, if $n > 1$ then*

$$\mu(n) = \begin{cases} (-1)^k, & \text{if } e_1 = ... = e_k = 1, \\ 0, & \text{if some } e_i > 1. \end{cases}$$

For each $m > 1$, define the m^{th} **cyclotomic polynomial** *by*

$$C_m(x) = \prod_{d|m} (x^d - 1)^{\mu(m/d)}.$$

Since cyclotomic polynomials are useful for the construction and theory of finite fields, they are also important in algebraic coding theory.

Theorem 2.12.10. *For each $m > 1$, $C_m(x)$ is irreducible over $\mathbb{Q}$.*

For the proof, which goes beyond the scope of this book, see Lang [La], Ch VIII, §3, or Theorem 6.5.5 of Herstein [Her].

Theorem 2.12.11. *We have*

$$x^n - 1 = \prod_{m|n} C_m(x),$$

where $C_m(x)$ is as above.

This follows from the Möbius inversion formula (see Lidl, Pilz [LP], Ch. 3, §13).

Example 2.12.12. *According to this theorem, $x^4 - 1 = C_1(x)C_2(x)C_4(x)$, where $C_1(x) = 1$, $C_2(x) = \frac{x^2-1}{x-1}$, and $C_4(x) = (x-1)^0(x^2-1)^{-1}(x^4-1)^1 = x^2 + 1$. This is the same result as the calculation "by hand" above.*

Exercise 2.12.13. *Show $C_5(x) = x^4 + x^3 + x^2 + x + 1$. More generally, show that for any prime p, $C_p(x) = x^{p-1} + ... + 1$.*

Exercise 2.12.14. *Completely factor $x^8 - 1$ into irreducibles over $\mathbb{Q}$.*

Exercise 2.12.15. *Completely factor $x^{10} - 1$ into irreducibles over $\mathbb{Q}$.*

Exercise 2.12.16. *Completely factor $x^{12} - 1$ into irreducibles over $\mathbb{Q}$.*

Exercise 2.12.17. *Compute $C_i(x)$, $1 \leq i \leq 9$.*

2.13 Polynomials and rings using GAP

2.13.1 Finite fields

GAP knows about all the finite fields.

To create the finite field $\mathbb{F}_p$ in GAP, type `F:=GF(p);` . For example, `F:=GF(5); Elements(F);` returns `[ 0*Z(5), Z(5)^0, Z(5), Z(5)^2, Z(5)^3 ]` . Type `Int(Z(5));` to determine the value of $Z(5)$ (as an integer mod 5). In general, $Z(p)$ is the smallest primitive root mod p, where p is a prime. Addition is carried out using $+$ and multiplication using $*$.

The finite fields in GAP are of the form $\mathbb{F}_{p^k} = GF(p^k)$, where p is a prime power and $k \geq 1$ is an integer. Let's start with something simple: enter $\mathbb{F}_2$ and $\mathbb{F}_3$ and print out all their elements.

```
gap> GF2:=GF(2);
GF(2)
gap> gf2:=Elements(GF2);
[ 0*Z(2), Z(2)^0 ]
gap> GF3:=GF(3);
GF(3)
gap> gf3:=Elements(GF3);
[ 0*Z(3), Z(3)^0, Z(3) ]
```

What are $Z(2)$, $Z(3)$? In general, $\mathbb{F}_{p^k}^\times$ is a cyclic group. GAP uses this fact and lets $Z(p^k)$ denote a generator of this cyclic group. If $k = 1$ then giving a generator of $\mathbb{F}_p^\times = (\mathbb{Z}/p\mathbb{Z})^\times$ is equivalent to giving a primitive root mod p. In fact, $Z(p)$ is the smallest primitive root mod p, as is obtained from the `PrimitiveRootMod` function. Here's how to verify this for $p = 3, 5$:

```
gap> PrimitiveRootMod(3);
2
gap> 2*Z(3)^0=Z(3);
true
gap> PrimitiveRootMod(5)*Z(5)^0=Z(5);
true
```

Onto fields $\mathbb{F}_{p^k}$ with $k > 1$.

```
gap> GF4:=GF(4);
GF(2^2)
gap> gf4:=Elements(GF4);
[ 0*Z(2), Z(2)^0, Z(2^2), Z(2^2)^2 ]
gap> GF7:=GF(7);
GF(7)
gap> GF8:=GF(8);
GF(2^3)
gap> gf8:=Elements(GF8);
[ 0*Z(2), Z(2)^0, Z(2^3), Z(2^3)^2, Z(2^3)^3, Z(2^3)^4, Z(2^3)^5, Z(2^3)^6 ]
gap> GF9:=GF(9);
GF(3^2)
gap> gf9:=Elements(GF9);
[ 0*Z(3), Z(3)^0, Z(3), Z(3^2), Z(3^2)^2, Z(3^2)^3, Z(3^2)^5, Z(3^2)^6,
  Z(3^2)^7 ]
```

Note that $GF(8)$ contains $GF(2)$ but not $GF(4)$. It is a general fact that $GF(p^m)$ contains $GF(p^k)$ as a subfield if and only if $k|m$.

What are `Z(2^2)`, `Z(2^3)`, `Z(3^2)`, ...? They are less easy to explicitly explain. $Z(p^k)$ is a root of a certain irreducible polynomial mod p called a Conway polynomial. We can check this in the case $p^k = 8$ in GAP by plugging this supposed root into the polynomial and see if we get 0 or not:

```
gap> R:=PolynomialRing(GF2,["x"]);
<algebra-with-one over GF(2), with 1 generators>
gap> p:=ConwayPolynomial(2,3);
Z(2)^0+x+x^3
gap> Value(p,Z(8));
0*Z(2)
```

This tells us that the generator of $GF(8)$ is a root of the polynomial x^3+x+1 mod 2.

Next, let's try adding, subtracting, and multiplying field elements in GAP.

```
gap> gf9[4]; gf9[5]; gf9[4]+gf9[5];
Z(3^2)
Z(3^2)^2
Z(3^2)^3
```

```
gap> gf9[3]; gf9[6]; gf9[3]+gf9[6];
Z(3)
Z(3^2)^3
Z(3^2)^5
gap> gf9[3]; gf9[6]; gf9[3]*gf9[6];
Z(3)
Z(3^2)^3
Z(3^2)^7
gap> gf9[4]; gf9[6]; gf9[4]*gf9[6];
Z(3^2)
Z(3^2)^3
Z(3)
gap> gf8[4]; gf8[6]; gf8[4]*gf8[6];
Z(2^3)^2
Z(2^3)^4
Z(2^3)^6
gap> gf8[4]; gf8[6]; gf8[4]+gf8[6];
Z(2^3)^2
Z(2^3)^4
Z(2^3)
```

2.13.2 Some vector spaces

The GAP code

```
vecs:=[[1,0,0],[1,1,1],[0,1,1]];
V:=VectorSpace(Rationals,vecs);
GeneratorsOfVectorSpace(V);
B:=Basis(V);
dim:=Length(B);
```

constructs the vector space over $\mathbb{Q}$ spanned by the vectors $(1,0,0), (1,1,1), (0,1,1)$, finds a basis, and computes its dimension.

Exercise 2.13.1. *Construct the vector space over $\mathbb{Q}$ spanned by the vectors $(1,0,0), (1,1,1), (0,-1,1)$, find a basis, and compute its dimension. Do the same, but with $\mathbb{Q}$ replaced by $GF(3)$.*

Exercise 2.13.2. *List all the elements of $GF(7)$. Create a multiplication table for $GF(7)$.*

The command `GF` returns a field for any prime power p^k. GAP assigns a fixed primitive element to each field $GF(p^k)$: `Z(p^k)`. This is a root of a Conway polynomial, which is the primitive polynomial over $GF(p)$ defining $GF(p^k)$.

Exercise 2.13.3. *List all the elements of* $GF(25)$.

Exercise 2.13.4. *Using GAP, create a multiplication table for* $GF(4)$.

The command `AlgebraicExtension` constructs an extension field.

```
gap> x:=Indeterminate(Rationals,"x");;
gap> p:=x^4+3*x^2+1;;
gap> e:=AlgebraicExtension(Rationals,p);
<field in characteristic 0>
gap> a:=RootOfDefiningPolynomial(e);
(a)
gap> a^5;
(-1*a-3*a^3)
```

Exercise 2.13.5. *Using GAP, let*

```
gap> x:=Indeterminate(GF(5),"x");;
gap> p:=x^4+x^2+2;;
gap> IsIrreducible(p);
true
gap> F:=AlgebraicExtension(GF(5),p);
<field of size 625>
gap> a:=RootOfDefiningPolynomial(F);
(a)
```

Compute $a, a^2, a^3, a^4, a^5, a^6$.

2.13.3 Rings

GAP knows a lot of rings. The ring of integers, $\mathbb{Z}$, and the ring of Gaussian integers, $\mathbb{Z} + i\mathbb{Z}$ are predefined (`Integers` and `GaussianIntegers`, resp.), and many others can be easily constructed.

Example 2.13.6. *For example, if you type* `R:= GaussianIntegers;`, `Sqrt(-1) in R;`, *and* `1 in R;` *then GAP will return true to the last two commands.*

A ring in GAP need not be commutative

2.13.4 Polynomials

A polynomial in GAP is defined over a certain type of ring (or field) that GAP understands. For example, GAP knows the ring of integers, $\mathbb{Z}$. A variable (such as x) cannot be typed into GAP without first being defined.

Example 2.13.7. *To define a variable for polynomials over the integers, call it x, type* `x:= Indeterminate( Integers, "x" );;`. *Now a polynomial such as $x^2 - 1$ can be defined:* `f:=x^2-1;`. *To multiply polynomials, use a $*$: type* `g:=(x^2-1)*(x-1);` *and GAP will return the (expanded) product. To evaluate a polynomial at a ring element, use the* `Value` *command: for example,* `Value(f,2);` *returns* 3.

Example 2.13.8. *To define two variables for polynomials over the field $GF(4)$, say x and y, type* `x:= Indeterminate( GF(4), "x" );;`, `y:= Indeterminate( GF(4), "y" );;`. *Now a polynomial such as $x^2 - y^2$ can be defined:* `f:=x^2-y^2;`. *To multiply polynomials, use a $*$: type* `g:=(x^2-y^2)*(y-1);` *and GAP will return the (expanded) product. To evaluate a polynomial at a ring element, use the* `Value` *command: for example,* `Value(f,[x,y],[2,-1]);` *returns* `1=Z(2)^0`.

Type `x:= Indeterminate( Rationals, "x" );;` and `GcdRepresentation( f, g );` for the extended Euclidean algorithm applied to the polynomials f, g in x. For example,

```
GcdRepresentation( x^2+1, x^3+1 );
```

returns

```
[ 1/2-1/2*x-1/2*x^2, 1/2+1/2*x ].
```

To double check this: note

```
(1/2-1/2*x-1/2*x^2)*(x^2+1)+(1/2+1/2*x )*(x^3+1);
```

returns 1, and

```
Gcd(x^2+1,x^3+1);
```

returns 1.

Exercise 2.13.9. *Find $gcd(x^2 + 1, x^3 + 1)$; over $GF(2)$.*

Type `R:= PolynomialRing( Rationals, 1 );`,
`x:= IndeterminatesOfPolynomialRing( R )[1];` and `Factors( R, f );`
to obtain the factorization of f in R. For example, `Factors( R, x^2-1 );`
gives the factorization

```
[ -1+x_1, 1+x_1 ].
```

`R:=PolynomialRing(Integers,[''x'']);` and the command `C:=CompanionMat(p);`
gives, for a monic polynomial p of degree n over a ring R, the companion matrix C for p as an element of $M_n(R)$.

Exercise 2.13.10. *Verify the multiplication table in Example 2.7.5.*

Exercise 2.13.11. *Find all irreducible polynomials of degree* 2 *in* $\mathbb{F}_3[x]$. *Let* $p_1(x), ..., p_k(x)$ *be all the irreducibles you found. Compute* $x^i \bmod p_j(x)$, *for all* $i > 0$ *and for each* j. *(Hint: Use* `PowerMod`.*)*
Find a primitive element of $\mathbb{F}_9$.

Exercise 2.13.12. *Find all irreducible polynomials of degree* 4 *in* $\mathbb{F}_2[x]$. *Let* $p_1(x), ..., p_k(x)$ *be all the irreducibles you found. Compute* $x^i \bmod p_j(x)$, *for all* $i > 0$ *and for each* j. *(Hint: Use* `PowerMod`.*)*
Find a primitive element of $\mathbb{F}_{16}$.

2.13.5 Gröbner bases

`PolynomialReduction` can be used to determine remainders. The first entry is the remainder, the second the coefficients with respect to the list of basis elements. (Note that the division algorithm works slightly different than the one in the book, thus if G is not a Gröbner basis you might get different remainders.)

```
gap> PolynomialReduction(x^5*y,G,MonomialGrlexOrder);
[ x, [ 1+y+y^2+y^3+x^4+x^3*y+x^2*y^2+x*y^3+y^4, 1+y+y^2+y^3+y^4, 0 ] ]
```

Exercise 2.13.13. *Let* F *be a field and let* $R = F[x, y]$. *Let* $<$ *denote the lexicographical order with* $x > y$. *Let* $f(x, y) = x^2y + xy^2 + y^2$, $f_1(x, y) = xy - 1$, *and* $f_2(x, y) = y^2 - 1$. *Find the normal form of* f *mod* $F = \{f_1, f_2\}$. *(Use* `PolynomialReduction`.*)*

To sort the monomials using the lexicographical ordering in GAP, one can use the following commands (available in GAP):

```
gap> x:=X(Rationals,"x");
x
gap> y:=X(Rationals,"y");
y
gap> r:=DefaultRing(x,y);
PolynomialRing(..., [ x, y ])
gap> elms:=[x^5*y+y,y^2*x^2+2*x];
[ y+x^5*y, 2*x+x^2*y^2 ]
gap> GroebnerBasis(elms,MonomialLexOrdering());
[ y+x^5*y, 2*x+x^2*y^2, -y^2+2*x^4, -2*y-x*y^3, -4*x^3-y^4, 8*x^2-y^6,
  -16*x-y^8, -32*y+y^11 ]
gap> ReducedGroebnerBasis(elms,MonomialLexOrdering());
[ -32*y+y^11, -16*x-y^8 ]
gap> order:=MonomialLexOrdering();
MonomialLexOrdering()
gap> List(Cartesian([0..3],[0..3]),i->x^i[1]*y^i[2]);
[ 1, y, y^2, y^3, x, x*y, x*y^2, x*y^3, x^2, x^2*y, x^2*y^2, x^2*y^3, x^3,
  x^3*y, x^3*y^2, x^3*y^3 ]
gap> l:=last;
[ 1, y, y^2, y^3, x, x*y, x*y^2, x*y^3, x^2, x^2*y, x^2*y^2, x^2*y^3, x^3,
  x^3*y, x^3*y^2, x^3*y^3 ]
gap> Sort(l,MonomialComparisonFunction(order));
gap> l;
[ 1, y, y^2, y^3, x, x*y, x*y^2, x*y^3, x^2, x^2*y, x^2*y^2, x^2*y^3, x^3,
  x^3*y, x^3*y^2, x^3*y^3 ]
```

`LeadingTerm` gives the leading term of a polynomial.

```
gap> LeadingTerm(x*y^2+2*x^2*y,MonomialGrlexOrder);
2*x^2*y
```

Exercise 2.13.14. *Rewrite the following polynomial, ordering its terms according to the lex, grlex and grevlex ordering and give $LM(f)$, $LT(f)$ and multideg(f) in each case.*

$$f(x, y, z) = 2x^2y^8 - 3x^5yz^4 + xyz^3 - xy^4.$$

Exercise 2.13.15. *Let $B = (x^2y - z, xy - 1)$ and $f = x^3 - x^2y - x^2z + x$.*

(a) Compute the remainder of dividing f by B for both the lex and the grlex ordering.

(b) Repeat part (a) with the order of the pair B reversed.

Exercise 2.13.16. *Compute the S-polynomial for $x^4y - z^2$ and $3xz^2 - y$ with respect to the lex ordering with $x > y > z$ and for $z > y > x$.*

To compute Gröbner bases using GAP, one has to define indeterminates and polynomials. Indeterminates are displayed either by their internal numbers or you can prescribe names. (Note however that the names hide the internal numbers and these numbers are basis for the monomial orderings. The best is to define a set of variables at the start and then not to redefine them afterwards.

`GBasis` computes a Gröbner basis. (You can set the info level as done here to get some information about the calculations.)

```
gap> SetInfoLevel(InfoGrobner,1);
gap> B:=[x*y-y^2,y^2-x];
[ x*y-y^2, -x+y^2 ]
gap> G:=GBasis(B,MonomialGrlexOrder);
[ x*y-y^2, -x+y^2, x-x^2 ]
```

If you want to change the variable ordering, the best way is to permute the variables in the polynomials instead (in the inverse order!). For example to change to an order $y > x$ we could do:

```
gap> B2:=List(B,i->OnIndeterminates(i,(1,2)));
[ -x^2+x*y, -y+x^2 ]
gap> G:=GBasis(B2,MonomialGrlexOrder);
[ -x^2+x*y, -y+x^2, y-x*y, -y+y^2 ]
```

Exercise 2.13.17. *Let $I = \langle x^5 + y^4 + z^3 - 1, x^3 + y^2 + z^2 - 1\rangle$. Compute a Gröbner basis for I with respect to the* lex, grlex, *and* grevlex *orderings. Compare.*
Repeat the calculations for $I = \langle x^5 + y^4 + z^3 - 1, x^3 + y^3 + z^2 - 1\rangle$ (only one exponent changed!)

2.14 Polynomials and rings using MAGMA

2.14.1 Finite fields

MAGMA knows about all the finite fields.

To create the finite field $\mathbb{F}_p$ in MAGMA, type `F:=GF(p);` . For example, `F:=GF(5); Set(F);` returns `{ 0, 1, 2, 3, 4 }` . Addition is carried out using $+$ and multiplication using $*$.

Exercise 2.14.1. *List all the elements of $GF(7)$. Create a multiplication table for $GF(7)$.*

The command `GF` returns a field for any prime power p^k. MAGMA assigns a fixed primitive element to each field $GF(p^k)$. This is a root of a Conway polynomial, which is the primitive polynomial over $GF(p)$ defining $GF(p^k)$. If you don't want to use a root of the Conway polynomial as a generator, one can also specify some other polynomial.

To define the field of 3 elements, type

```
F3 := FiniteField(3);
```

We can define the field of 3^4 elements in several different ways. We can use the Conway polynomial:

```
> F<z> := FiniteField(3^4);
> F;
Finite field of size 3^4
> DefiningPolynomial(F);
$.1^4 + 2*$.1^3 + 2
```

We can define it as an extension of the field of 3 elements, using the Conway polynomial:

```
> F<z> := ext< F3 | 4 >;
> F;
Finite field of size 3^4
```

We can supply our own polynomial, say $x^4 + x^3 + 2$:

```
> P<x> := PolynomialRing(F3);
> p:=x^4+x^3+2;
```

```
> IsIrreducible(p);
true
> F<z> := ext< F3 | p >;
> F;
Finite field of size 3^4
```

We can define it as an extension of the field of 3^2 elements:

```
> F9<w> := ext< F3 | 2 >;
> F<z> := ext< F9 | 2 >;
> F;
Finite field of size 3^4
```

Each element $a \in F$ is a polynomial of degree at most 3 in the primitive element z:

$$a = a_0 + a_1 z + a_2 w^2 + a_3 w^3.$$

The coefficients a_i are obtained from the `Eltseq` command:

```
> w:=PrimitiveElement(F);
>
> a:=Random(F);
> a;
z^9
> Eltseq(a);
[ 0, 2, 1, 1 ]
> 2*w+w^2+w^3; //check
z^9
>
> a:=Random(F);
> a;
z^37
> Eltseq(a);
[ 1, 2, 0, 0 ]
> 1+2*w; //check
z^37
```

Exercise 2.14.2. *List all the elements of $GF(25)$, as poewers of a primitive element and as polynomials of degree 1 or less in the primitive element.*

Exercise 2.14.3. *Using MAGMA, create a multiplication table for $GF(4)$.*

2.14.2 Rings

MAGMA knows a lot of rings. The ring of integers, $\mathbb{Z}$, and more generally the ring of integers of any number field are predefined (`Integers()` and `Integers(NumberField(p(x)))`, resp., where $p(x)$ is a polynomial), and many others can be easily constructed.

Example 2.14.4. *For example, if you type* `R:= Integers( QuadraticField(-1));`, `i:=Basis(R)[2]; i^2; i in R;`, *and* `1 in R;` *then MAGMA will return true to the last two commands.*

2.14.3 Polynomials

Type `R<x>:= PolynomialRing( Rationals(), 1 );` to define the variable x. It is necessary to define x before you can type a polynomial in x into MAGMA. Once you gave a polynomial, say `f:=x^2-1;`, you can use MAGMA to do several things: evaluate f at a particular value of x, differentiate or integrate f, find the coefficients of f, factor f, and so on. Type `Factorization( f );` to obtain the factorization of f in R. For example, `Factorization( x^2-1 );` gives the factorization

`[<x - 1, 1>,<x + 1, 1>].`

2.14.4 RSA for polynomials in MAGMA

Pick two irreducible polynomials p, q over a finite field F and let $n = pq$. Form the quotient ring $R := F[x]/nF[x]$.

```
> F:=GF(3);
> P<x> := PolynomialRing(F);
> p:=ConwayPolynomial(3,5);
> q:=ConwayPolynomial(3,4);
> n:=p*q;
> n;
x^9 + 2*x^8 + x^5 + 2*x^4 + 2*x^3 + x + 2
> I:=ideal<P|n>;
> R:=quo<P|I>;
> IsFinite(R);
true 19683
```

Next, we need to find out how many elements are in the group $(F[x]/nF[x])^\times$. Denote this number by N. Note that this is not the Euler phi function value of the integer $|F[x]/nF[x]|$.

```
> f:=x^12+x+1;
> f:=R!f;
> IsUnit(f);
true
> L:=[f:f in R];
> U:=[f:f in R|IsUnit(f)];
> #U;
19360
> #R;
19683
> EulerPhi(#R);
13122
> Factorization(#U);
[ <2, 5>, <5, 1>, <11, 2> ]
>
```

Now, as with RSA, pick a secret encryption exponent, e. Let d denote its inverse mod N. This is the secret decryption exponent. Pick a message for Alice to send to Bob, say $x^4 + x + 1 \in R = F[x]/nF[x]$. (The message must have degree less than that of n.) MAGMA uses a capital X to distinguish elements of $F[x]$ and those of its quotient R.

Alice sends $c = m^e$ to Bob. Bob decrypts Alice's cipher-text by using that fact that

$$c^d = (m^e)^d = m^{ed} = m^{1+kN} = m(m^N)^k = m.$$

```
> e:=7;
> R<X>:=quo<P|I>;
>
> m:=X^4+X+1;
> m in R;
true
> c:=m^e;
> c;
X^8 + 2*X^6 + 2*X^5 + 2*X^4 + 2*X
```

```
> Z_N:=ResidueClassRing(#U);
>
> E:=Z_N!e;
>  d:=E^-1;
> d;
11063
> D:=Integers()!d;
> D;
11063
> c^D;
X^4 + X + 1
> m;
X^4 + X + 1
```

Exercise 2.14.5. *(RSA for polynomials) Using* $m = x^5 + x^3 + 2$ *and the same* e*, find* d *and* c *as above and check that* $c^d = m$*.*

Exercise 2.14.6. *Using* `R<x>:=PolynomialRing(Integers(),1);`, `p:=x^8-1`, *and* `Factorization(p);`, *find the factorization over the integers of* p*.*

Exercise 2.14.7. *Using* `R<x>:=PolynomialRing(GF(2),1);`, `p:=x^8-1`, *and* `Factorization(p);`, *find the factorization over the integers of* p*.*

To evaluate $f(x) = x^2 - 1$ at $x = 12$ in MAGMA, use the `Evaluate(f,x0);` command. For example, Using `R<x>:=PolynomialRing(Integers(),1);`, `p:=x^8-1`, and `Evaluate(f,3);` evaluates $3^8 - 1$ in MAGMA.

In MAGMA, type `R<x>:=PolynomialRing(Integers());`, `a(x)^n mod m(x);` to compute $a(x)^n \bmod m(x)$. For example, `R<x>:=PolynomialRing(Integers());`

```
x^17 mod (x^2+1);
```

returns x.

Exercise 2.14.8. *Verify Example 2.5.4.*

In MAGMA, `R<x>:=PolynomialRing(Integers(),1);` and the command `C:=CompanionMatrix(p);` gives, for a monic polynomial p of degree n over a ring R, the companion matrix C for p as an element of $M_n(R)$.

Exercise 2.14.9. *Verify the multiplication table in Example 2.7.5.*

In MAGMA, first type `R<x,y,z>:=PolynomialRing(Integers(),3);`. The leading monomial can be computed using the `LeadingTerm` command. For example,

```
p:=320*x*y^2+9*x*y^4-96*x*y^4*z^2;
```

and `LeadingTerm(p);` returns $-96xy^4z^2$.

```
> L:=[x^i*y^j : i in [0..4],j in [0..4]];
> R<x,y>:=PolynomialRing(GF(5),2);
> L:=[x^i*y^j : i in [0..4],j in [0..4]];
> f:=function(a,b)
function> if a lt b then return -1; end if;
function> if a eq b then return 0; end if;
function> if a gt b then return 1; end if;
function> end function;
> L;
[
    1,
    x,
    x^2,
    x^3,
    x^4,
    y,
    x*y,
    x^2*y,
    x^3*y,
    x^4*y,
    y^2,
    x*y^2,
    x^2*y^2,
    x^3*y^2,
    x^4*y^2,
    y^3,
    x*y^3,
    x^2*y^3,
    x^3*y^3,
    x^4*y^3,
    y^4,
```

```
    x*y^4,
    x^2*y^4,
    x^3*y^4,
    x^4*y^4
]
> Sort(L,f);
[
    1,
    y,
    y^2,
    y^3,
    y^4,
    x,
    x*y,
    x*y^2,
    x*y^3,
    x*y^4,
    x^2,
    x^2*y,
    x^2*y^2,
    x^2*y^3,
    x^2*y^4,
    x^3,
    x^3*y,
    x^3*y^2,
    x^3*y^3,
    x^3*y^4,
    x^4,
    x^4*y,
    x^4*y^2,
    x^4*y^3,
    x^4*y^4
]
(2, 6)(3, 11)(4, 16)(5, 21)(8, 12)(9, 17)(10, 22)(14, 18)(15, 23)(20, 24)
>
```

Exercise 2.14.10. *Rewrite the following polynomial, ordering its terms according to the lex, grlex and grevlex ordering and give $LM(f)$, $LT(f)$ and*

multideg(f) *in each case.*

$$f(x,y,z) = 2x^2y^8 - 3x^5yz^4 + xyz^3 - xy^4.$$

Exercise 2.14.11. *Find the leading terms of some polynomials of your own choosing.*

Exercise 2.14.12. *Let* $B = (x^2y - z, xy - 1)$ *and* $f = x^3 - x^2y - x^2z + x$.
a) Compute the remainder of dividing f *by* B *for both the lex and the grlex ordering.*
b) Repeat part a) with the order of the pair B *reversed.*

Exercise 2.14.13. *Compute the S-polynomial for* $x^4y - z^2$ *and* $3xz^2 - y$ *with respect to the lex ordering with* $x > y > z$ *and for* $z > y > x$.

Exercise 2.14.14. *Let* F *be a field and let* $R = F[x,y]$. *Let* $<$ *denote the lexicographical order with* $x > y$. *Let* $f(x,y) = x^2y + xy^2 + y^2$, $f_1(x,y) = xy - 1$, *and* $f_2(x,y) = y^2 - 1$. *Find the normal form of* f *mod* $F = \{f_1, f_2\}$.
(Use `NormalForm`.*)*

Exercise 2.14.15. *Find equations of the curve parameterized by* $(t^2, t, 1+t)$ *using Grobner bases.*

`R<x>:=PolynomialRing(Integers(),1);` and the command `C:=CompanionMatrix(p);` gives, for a monic polynomial p of degree n over a ring R, the companion matrix C for p as an element of $M_n(R)$.

Exercise 2.14.16. *Verify the multiplication table in Example 2.7.5.*

Exercise 2.14.17. *Let* $I = (x^5 + y^4 + z^3 - 1, x^3 + y^2 + z^2 - 1)$. *Compute a Gröbner basis for* I *with respect to the* lex, grlex, *and* grevlex *orderings. Compare.*
Repeat the calculations for $I = (x^5 + y^4 + z^3 - 1, x^3 + y^3 + z^2 - 1)$ *(only one exponent changed!)*

Chapter 3

Error-correcting codes

In this chapter, we apply some of what we have learned in the last chapter about finite fields to constructing error-correcting codes. This is not only of interest in itself, but serves to create a number of naturally arising examples and gives us practice in doing explicit computations with some of the relatively abstract ideas we've seen so far.

3.1 Background on vector spaces

First, some notation and background.

If S and T are any two sets, let $S \times T$ denote the **Cartesian product** of S and T defined by

$$S \times T = \{(s,t) \mid s \in S,\ t \in T\}.$$

In other words, $S \times T$ is simply the set of ordered pair of elements in S and T. For example, $\mathbb{R} \times \mathbb{R}$ (which is also written as $\mathbb{R}^2$) is the Cartesian plane, where the first coordinate usually corresponds to the "x-axis" and the second coordinate to the "y-axis". More generally, if S_1, S_2, ..., S_n are n sets, then $S_1 \times S_2 \times ... \times S_n$ is defined by

$$S_1 \times S_2 \times ... \times S_n = \{(s_1, s_2, ..., s_n) \mid s_i \in S_i,\ i = 1, 2, ..., n\}.$$

If all these sets S_i are equal to each other, say $S = S_1 = ... = S_n$, then we write this as S^n. In other words, S^n is simply the set of ordered n-tuples of elements in S. For example, $\mathbb{R}^3$ is Cartesian 3-space, where the first

coordinate usually corresponds to the "x-axis", the second coordinate to the "y-axis", and the third coordinate the "z-axis".

Let F be any field and let $V = F^n$ denote the set of n-tuples of elements of F. Define **vector addition** $+$ and **scalar multiplication** $\cdot$ as follows: for $\mathbf{v} = (v_1, ..., v_n), \mathbf{w} = (w_1, w_2, ..., w_n) \in V$ and $c \in F$, define vector addition and scalar multiplication "component-wise"

$$\mathbf{v} + \mathbf{w} = (v_1 + w_1, v_2 + w_2, ..., v_n + w_n),$$

$$c \cdot \mathbf{v} = (cv_1, cv_2, ..., cv_n).$$

It is easy to check that with these two operations, V satisfies the following axioms for a vector space.

Definition 3.1.1. *Let F be a field and V be a set with operations $+ : V \times V \to V$, written $+ : (\mathbf{u}, \mathbf{v}) \longmapsto \mathbf{u} + \mathbf{v}$, and $\cdot : F \times V \to V$, $+ : (a, \mathbf{v}) \longmapsto a \cdot \mathbf{v}$. V is called a* **vector space over** F *(or an* F**-vector space**$)$ *if the following properties hold. For all $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$ and $a, b \in F$,*

- $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$ *(commutativity)*
- $(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w})$ *(associativity)*
- *the vector $\mathbf{0} = (0, 0, ..., 0) \in V$ satisfies $\mathbf{u} + \mathbf{0} = \mathbf{u}$ (the zero vector $\mathbf{0}$ is the additive identity),*
- *for each $\mathbf{v} \in V$ the element $(-1)\mathbf{v} = -\mathbf{v} \in V$ satisfies $\mathbf{v} + (-\mathbf{v}) = \mathbf{0}$ (each element $\mathbf{v}$ has an additive inverse $-\mathbf{v}$)*
- $(a + b)\mathbf{v} = a\mathbf{v} + b\mathbf{v}$ *and* $a(\mathbf{v} + \mathbf{w}) = a\mathbf{v} + a\mathbf{w}$ *(distributive laws)*
- $(ab)\mathbf{v} = a(b\mathbf{v})$
- $1 \cdot \mathbf{v} = \mathbf{v}$.

In particular, every vector space must contain at least one element (the zero vector). The elements of V are called **vectors** and the elements of F are called **scalars**. A **linear combination** of vectors $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k$ of V is an expression of the form

$$c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + ... + c_k\mathbf{v}_k,$$

for some scalar coefficients $c_i \in F$ $(i = 1, 2, ..., k)$. Since V is closed under addition and scalar multiplication, the linear combination of any set of vectors in V is also in V.

Any subset $W \subset V$ of a vector space which is closed under vector addition (restricting $+$ from V to W) and scalar multiplication is called a **subspace** of V. In other words, a subset of V is a subspace if and only if it is closed under taking arbitrary linear combinations.

Example 3.1.2. *If $V = F^3$ (which reduces to the usual Euclidean 3-space when $F = \mathbb{R}$) then $W = \{(x, y, 0) \mid x, y \in F\}$ is a vector subspace but $W = \{(x, y, 1) \mid x, y \in F\}$ is not.*

Suppose V is a vector space over a field F. If there are non-zero vectors $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k$ such that every $\mathbf{v} \in V$ can be written as a linear combination of these $\mathbf{v}_i$'s, i.e.,

$$\mathbf{v} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + ... + c_k\mathbf{v}_k,$$

for some scalar coefficients $c_i \in F$ $(i = 1, 2, ..., k)$ then we say that V is **spanned** by the elements $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k$. We also say that elements $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k$ form a **spanning set** for V.

Example 3.1.3. *If $V = F^3$ is then V is spanned by $\mathbf{e}_1 = (1, 0, 0), \mathbf{e}_2 = (0, 1, 0), \mathbf{e}_3 = (0, 0, 1)$.*

In fact, if $\mathbf{f}_1 = (1, 1, 0), \mathbf{f}_2 = (1, 0, 0), \mathbf{f}_3 = (0, 0, 1), \mathbf{f}_4 = (0, 1, 0)$ then $\mathbf{f}_1, ..., \mathbf{f}_4$ forms a spanning set for V as well (since it contains the set $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$).

Example 3.1.4. *If $V = F[x]$ is the set of all polynomials with coefficients in F then V is spanned by the powers of x: $\mathbf{e}_1 = 1$, $\mathbf{e}_2 = x$, $\mathbf{e}_3 = x^2$, ... If $\mathbf{v} = c_n x^n + c_{n-1}x^{n-1} + ... + c_1 x + c_0 \in V$ then*

$$\mathbf{v} = c_0\mathbf{v}_1 + c_1\mathbf{v}_2 + ... + c_n\mathbf{v}_{n+1},$$

for coefficients $c_i \in F$ $(i = 0, 2, ..., n)$. This set is a vector space under ordinary addition and scalar multiplication of polynomials.

Definition 3.1.5. *If a vector space V has a finite spanning set then we say that V is* **finite dimensional**. *The* **dimension** *of V over F is the smallest number of vectors in a spanning set of V. The dimension of V is written $dim(V) = k$, or $dim_F(V) = k$ when we want to specify the ground field. If V is of dimension k then there are k vectors which span V and any such set of k spanning vectors of V is called a* **basis** *of V.*

For example,
(a) $V = F[x]$ is an infinite dimensional vector space,
(b) $dim(F^n) = n$.

Lemma 3.1.6. *Let V be a k-dimensional vector space over F and let $\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_k$ be a basis of V. For each $\mathbf{v} \in V$ there is a unique set of coefficients $c_1, ..., c_k$ in F such that*

$$\mathbf{v} = c_1\mathbf{f}_1 + c_2\mathbf{f}_2 + ... + c_k\mathbf{f}_k.$$

proof: Since $\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_k$ is a spanning set, for each $\mathbf{v} \in V$ there is a set of coefficients $c_1, ..., c_k$ such that the above equality holds. To see that this is unique, suppose

$$\mathbf{v} = c_1\mathbf{f}_1 + c_2\mathbf{f}_2 + ... + c_k\mathbf{f}_k = c_1'\mathbf{f}_1 + c_2'\mathbf{f}_2 + ... + c_k'\mathbf{f}_k.$$

We must show $c_1 = c_1'$, ..., $c_k = c_k'$. Suppose not, to get a contradiction. Suppose $c_i \neq c_i'$, for some i. For typographical simplicity we will assume $i = 1$. Subtracting, we get $\mathbf{0} = (c_1 - c_1')\mathbf{f}_1 + (c_2 - c_2')\mathbf{f}_2 + ... + (c_k - c_k')\mathbf{f}_k$. By our assumption,

$$\mathbf{f}_1 = \frac{c_2 - c_2'}{c_1' - c_1}\mathbf{f}_2 + ... + \frac{c_k - c_k'}{c_1' - c_1}\mathbf{f}_k.$$

Let $\mathbf{w} \in V$ be arbitrary. Substitutinig this expression into a basis expansion

$$\mathbf{w} = b_1\mathbf{f}_1 + b_2\mathbf{f}_2 + ... + b_k\mathbf{f}_k, \qquad \mathbf{w} \in V,$$

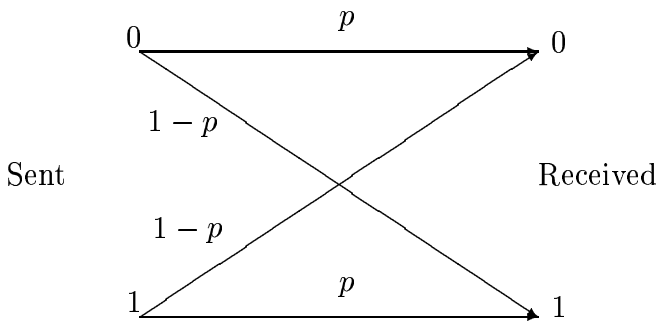
yields a shorter expansion of the form

$$\mathbf{w} = b_2'\mathbf{f}_2 + ... + b_k'\mathbf{f}_k.$$

Since $\mathbf{w}$ was arbitrary, this implies V is $k-1$ dimensional, a contradiction. □

This lemma is important. It says that we may identify any vector in a finite-dimensional vector space with its list of coefficients with respect to a fixed basis. If V is a k-dimensional vector space over F and let $B = \{\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_k\}$ is a basis of V then, in the notation of the lemma, we write $[\mathbf{v}]_B = (c_1, ..., c_k)$. We call the c_i's the **coordinates of v with respect to** B.

Lemma 3.1.7. *The vectors $\mathbf{e}_1 = (1, 0, 0, ..., 0)$, $\mathbf{e}_2 = (0, 1, 0, ..., 0)$,..., $\mathbf{e}_n = (0, 0, ..., 0, 1)$, form a basis for F^n.*



The basis in the above lemma is called the **standard basis**.

proof: If $\mathbf{v} = (v_1, ..., v_n) \in F^n$ then $\mathbf{v} = v_1\mathbf{e}_1 + ...v_n\mathbf{e}_n$, so the vectors $\mathbf{e}_1, ..., \mathbf{e}_n$ span F^n. If any one of these vectors were omitted, say $\mathbf{e}_i$, then it would be impossible to span all of V ($\mathbf{e}_i$ itself would not be in the span of the others, for example). Thus no proper subset of $\{\mathbf{e}_1, ..., \mathbf{e}_n\}$ spans F^n, so they form a basis. □

3.2 Background on information theory

This section assumes that the reader has some familiarity with some basic probability.

3.2.1 Binary symmetric channel

Consider a source sending messages through a noisy channel; for example, a CD player reading from a scratched music CD, or a wireless cellphone capturing a weak signal from a relay tower which is too far away.

For simplicity, assume that the message being sent is a sequence of 0's and 1's. Assume that, due to noise, when a 0 is sent, the probability that a 0 is (correctly) received is p and the probability that a 1 is (incorrectly) received is $1 - p$. Assume also that the noise of the channel is not dependent on the symbol sent: when a 1 is sent, the probability that a 1 is (correctly) received is p and the probability that a 0 is (incorrectly) received is $1 - p$. The following diagram summarizes this.

This channel is called the **binary symmetric channel**.

3.2.2 Uncertainty

We want to formalize the notion of uncertainty. Consider two experiments. In the first, you flip a fair coin. In the second, you roll a fair dice. It is reasonable to say that the outcome of the second experiment is more uncertain than the outcome of the first, simply because there are more possibilities to choose from.

Consider the uncertainty (whatever that is) of a random variable X which takes on only the distinct values x_1, ..., x_n with non-zero probabilities p_1, ..., p_n, resp., where $p_1 + ... + p_n = 1$. The uncertainty of X should not depend on the values of X but only on the probabilites p_i, $1 \leq i \leq n$. Likely events should be less uncertain than rare events. With this motivation, we present Claude Shannon's definition.

Definition 3.2.1. *The* **uncertainty** *or* **entropy** *of the above random variable X is defined to be*

$$H(X) = H(p_1, ..., p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i,$$

where $\log_2$ *is the logarithm to the base* 2.

Example 3.2.2. *Suppose that X is the signal received by a (noisy) binary symmetric channel. Then*

$$H(X) = H(p, 1-p) = -p \log_2 p - (1-p) \log_2 (1-p).$$

The maximum uncertainty is when $p = 1/2$, in which case $H(X) = 1$.

It is intuitively obvious that when the channel creates a lot of errors then there is a limitation to the information which can be sent. The next definition makes this idea more precise.

Definition 3.2.3. *The* **capacity** *of the channel is $cap(X) = cap(p) = 1 - H(X)$.*

In the case of the binary symmetric channel, the capacity is $cap(X) = 1 + p \log_2 p + (1-p) \log_2 (1-p)$. The minimum capacity is when $p = 1/2$.

To justify the formula which defines the capacity, we need Shannon's fundamental theorem of information theory (also called the noisy channel theorem). It will be stated in section §3.3.2.

Exercise 3.2.4. *Suppose that we send a sequence of length n consisting of 0's and 1's.*

(a) What is the probability of exactly one error? (Ans: $np(1-p)^{n-1}$. Why?)

(b) What is the probability of exactly two errors? (Ans: $\frac{1}{2}n(n-1)p^2(1-p)^{n-2}$. Why?)

(c) What is the probability of exactly k errors?

3.3 Motivation and notation for codes

Before the formal definition is given, let us go straight to an example.

Example 3.3.1. *"Hi Bob".*

"What?"

"Hi Bob" (louder).

"What?"

"Hi Bob" (even louder).

"Oh, hi."

This illustrates a "repetition code". More precisely, the **q-ary repetition code** *of length n is the set of all n-tuples of the form $(x, x, ..., x)$, for $x \in \mathbb{F}_q$. (We leave it as an exercise to verify that this is a vector space over $\mathbb{F}_q$.) We think of x as representing information you want to send. It could be the "greyness" of a pixel in a picture or a letter (represented in ASCII code) in a word, for example. Since the channel might contain noise, we send $(x, x, ..., x)$ instead, with the understanding that the receiver should perform a "majority vote" to decode the vector. (For example, if $(0, 1, 0, ..., 0)$ was received then 0 "wins the vote").*

This wasn't a very efficient example. Let's try again.

Example 3.3.2. *In this example, we will design a code which will send a number in $\{0, 1, 2, ..., 7\}$ to another person.*

First, write the number in binary, 0 as 000, 1 as 001, ..., 7 as 111. If $c_i \in \mathbb{F}_2$ for $1 \leq i \leq 3$, then define

$$c_4 = c_1 + c_2, \qquad c_5 = c_1 + c_2 + c_3, \qquad c_6 = c_3.$$

For example, 4 or 100 is encoded as $(1, 0, 0, 1, 1, 0)$. Let $C \subset \mathbb{F}_2^6$ denote the set of all vectors $c = (c_1, ..., c_6)$ where $c_i \in \mathbb{F}_2$, for $1 \leq i \leq 3$, and c_4, c_5, c_6

are defined as above. In other words, define

$$C = \{v \in \mathbb{F}_2^6 \mid Hv = \mathbf{0}\},$$

where

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

This is a 3 dimensional vector space over $\mathbb{F}_2$. Why is this not a "good" code? Suppose for instance Alice wants to send the number 2 to Bob. In binary, 2 is 010, so she encodes $(0,1,0)$ as $c = (0,1,0,1,1,0)$. Suppose Bob receives $v = (0,0,0,1,1,0)$. Bob knows that an error occurred in transmission since $Hv \neq \mathbf{0}$, so $v \notin C$. Suppose only one error occurred. (This is more likely than having two or more errors, and it is easier to detect exactly one error that two or more.) Where did the error occur? Bob doesn't know where since v is just as close to $(1,0,0,1,1,0)$ as it is to $(0,1,0,1,1,0)$ in the sense that they differ by the same number of bits. He can't tell which is correct.

The problem with this example boils down to the fact that the first and second column of H are the same. We shall explain this type of deficency more later.

3.3.1 Basic definitions

Enough examples - now for the definition.

Definition 3.3.3. *Let F be a finite field. A subset C of $V = F^n$ is called a* **code of length** *n. A subspace of V is called a* **linear code of length** *n. If $F = \mathbb{F}_2$ then C is called a* **binary** *code. If $F = \mathbb{F}_3$ then C is called a* **ternary** *code. If F has q elements then C is called a* **q-ary** *code. The elements of a code C are called* **code words** *or* **code vectors**. *Sometimes elements of F^n which do not necessarily belong to C are called "received vectors".*

The **information rate** *of C is*

$$R = \frac{\log_q |C|}{n},$$

where $|C|$ denotes the number of elements of C.

3.3.2 The Hamming metric

We've seen so far come simple examples of codes. What is needed is some notion of how to compare codewords. Geormetrically, two codewords are "far" from each other if there are "a lot" of coordinates where they differ. This notion is made more precide in the following definition.

Definition 3.3.4. *If* $\mathbf{v} = (v_1, v_2, ..., v_n)$, $\mathbf{w} = (w_1, w_2, ..., w_n)$ *are vectors in* $V = F^n$ *then we define*

$$d(\mathbf{v}, \mathbf{w}) = |\{i \mid 1 \leq i \leq n,\ v_i \neq w_i\}|$$

to be the **Hamming distance** *between* $\mathbf{v}$ *and* $\mathbf{w}$. *The function* $d : V \times V \to \mathbb{N}$ *is called the* **Hamming metric**. *The* **weight** *of a vector (in the Hamming metric) is* $d(\mathbf{v}, \mathbf{0})$.

Note that

$$d(\mathbf{v}, \mathbf{w}) = |\{i \mid 1 \leq i \leq n,\ v_i - w_i \neq 0\}| = d(\mathbf{v} - \mathbf{w}, \mathbf{0}) \tag{3.1}$$

for any vectors $\mathbf{v}, \mathbf{w} \in F^n$ (or, more generally, any vectors in a linear code). Using this, it is easy to show that d satisfies the properties of a metric:

- $d(\mathbf{v}, \mathbf{w}) \geq 0$ for all $\mathbf{v}, \mathbf{w} \in F^n$ and $d(\mathbf{v}, \mathbf{w}) = 0$ if and only if $\mathbf{v} = \mathbf{w}$.
- $d(\mathbf{v}, \mathbf{w}) = d(\mathbf{w}, \mathbf{v})$, for all $\mathbf{v}, \mathbf{w} \in F^n$.
- $d(\mathbf{u}, \mathbf{w}) \leq d(\mathbf{u}, \mathbf{v}) + d(\mathbf{v}, \mathbf{w})$, for all $\mathbf{u}, \mathbf{v}, \mathbf{w} \in F^n$.

Let $v \in F^n$ and let

$$B(v, r, F^n) = \{w \in Fn \mid d(v, w) \leq r\}.$$

This is called the **ball of radius** r **about** v. Since F^n is finite, this ball has only a finite number of elements. It is not hard to count them using a little bit of basic combinitorics. Since this count shall be needed later, we record it in the following result.

Lemma 3.3.5. *If* $0 \leq r \leq n$ *and* $q = |F|$ *then*

$$|B(v, r, F^n)| = \sum_{i=0}^{r} \binom{n}{i} (q-1)^i.$$

proof: Let

$$B_i(v, r, F^n) = \{w \in Fn \mid d(v, w) = i\}.$$

This is called the **shell of radius** i **about** v. It is consists of all vectors with exactly i coordinates different from v. There are $\begin{pmatrix} n \\ i \end{pmatrix}$ ways to choose i out of n coordinates. There are $(q-1)^i$ ways to choose these i coordinates to be different from those in v. Thus,

$$|B(v, r, F^n)| = \sum_{i=0}^{r} |B_i(v, r, F^n)| = \sum_{i=0}^{r} \begin{pmatrix} n \\ i \end{pmatrix} (q-1)^i.$$

□

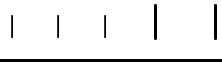
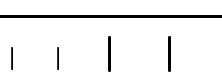
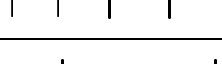
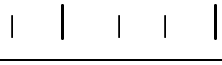
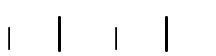
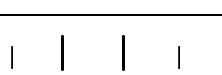
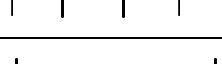
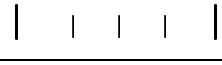
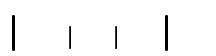
Example 3.3.6. *If* $F = \mathbb{F}_{11}$ *and* $V = F^{10}$ *then*

$$C = \{(x_1, x_2, ..., x_{10}) \mid x_i \in F,\ x_1 + 2x_2 + 3x_3 + ... + 9x_9 + 10x_{10} \equiv 0 \pmod{11}\}$$

is called the **ISBN code**. *This is an* 11*-ary linear code of length* 10. *This is the same code used in book numbering except that the number* 10 *is denoted by* X *on the inside cover of a book.*

For example, $(1, 0, 0, 0, 0, 0, 0, 0, 0, 1)$ *and* $(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$ *are code words. Their Hamming distance is* 8.

Example 3.3.7. *The U. S. Post Office puts a bar code on each letter to help with its delivery. What are these funny symbols? Translated into digits, they are given in the following table.*

number	*bar code*
1	
2	
3	
4	
5	
6	
7	
8	
9	
0	

Each "word" in the postal bar-code has 12 digits, each digit being represented by short bars (we regard as a 0*) and longer bars (which are regarded as a* 1*), as above. The 12 digits are interpreted as follows: The first 5 digits are your zip code, the next 4 digits are the extended zip code, the next 2 digits are the delivery point digits, and the last digit is a check digit (all the digits must add to* 0 *mod* 10*).*

For example, suppose that after translating the bars into digits, you found that the postal code on an envelope was ?62693155913*, where* ? *indicates a digit which was smudged so you couldn't read it. Since the sum must be* 0 *mod* 10*, we must have* $? = 0$.

Definition 3.3.8. *The* **weight distribution vector** *of a code* $C \subset F^n$ *is the vector*

$$W(C) = (W_0, W_1, ..., W_n)$$

where W_i *denote the number of codewords in* C *of weight* 1*. Note that for an linear code* C*,* $W_0 = 1$*, since any vector space must contain the zero vector.*

Example 3.3.9. *In Example 3.3.2, the code C has weight distribution vector* $W(C) = (1, 0, 1, 3, 2, 1)$.

3.3.3 Properties of the minimum distance

Let $C \subset F^n$ be a code. The number

$$d = \min_{\mathbf{v} \neq \mathbf{w}} d(\mathbf{v}, \mathbf{w}),$$

where the minimum runs over all distinct code words in C, is called the **minimum distance** of C. This (as we will see) measures the ability for the code to correct errors which might be introduced in transmission.

Theorem 3.3.10. *If C is a linear code and if d denotes the minimum distance of C then*

$$d = \min_{\mathbf{v} \neq \mathbf{0}} d(\mathbf{v}, \mathbf{0}),$$

where the minimum runs over all non-zero code words in C.

In other words, the minimum distance of a linear code is the length of its shortest non-zero vector.

proof: Clearly, $d \leq \min_{\mathbf{v} \neq \mathbf{0}} d(\mathbf{v}, \mathbf{0})$. Conversely, suppose $\mathbf{v}, \mathbf{w}$ are code words such that $d = d(\mathbf{v}, \mathbf{w})$. By (3.1), $d = d(\mathbf{v} - \mathbf{w}, \mathbf{0})$, so $\min_{\mathbf{v} \neq \mathbf{0}} d(\mathbf{v}, \mathbf{0}) \leq d$. $\square$

The above theorem helps us find the minimum distance of the ISBN code C in Example 3.3.6. This is a subspace of $\mathbb{F}_{11}^{10}$. There is no codeword of weight 1. This is due to the fact that $(x_1, x_2, ..., x_{10}) \in \mathbb{F}_{11}^{10}$ is a codeword if and only if $x_1 + 2x_2 + 3x_3 + ... + 9x_9 + 10x_{10} \equiv 0 \pmod{11}$. But if only one coordinate is non-zero then 11 divides ix_i, for some $1 \leq i \leq 10$. This forces $11 | x_i$ but in $\mathbb{F}_{11}$ that says $x_i = 0$. However, there is a codeword of weight 2, for example, $(1, 0, ..., 0, 1)$ ($x_1 = x_{10} = 1$, all other $x_i = 0$). By the above theorem, the minimum distance d of C is therefore $d = 2$.

Definition 3.3.11. *We say that C is e-***error correcting** *if the following property always holds: given any $\mathbf{w} \in F^n$ whose Hamming distance from some $\mathbf{c} \in C$ is $\leq e$ then any $\mathbf{c}' \in C$, $\mathbf{c} \neq \mathbf{c}'$, must satisfy $d(\mathbf{w}, \mathbf{c}') > e$ (in other words, a codeword at most distance e from $\mathbf{w}$ is unique if $\mathbf{w}$ has at most e errors).*

In the next section we shall show that if d denotes the minimum distance of C then the smallest e for which G is e-error correcting is the integer part of $\frac{d-1}{2}$. For now, we explore some of this definitions more basic consequences.

This definition means that if you have sent off a code word $\mathbf{c} \in C$ and *know* the receiver received a vector $\mathbf{v} \in F^n$ which has e errors or less (i.e., $d(\mathbf{c}, \mathbf{v}) \leq e$) then the code word closest to $\mathbf{v}$ (in the sense of the Hamming distance) is $\mathbf{c}$. In this case, to decode the received vector $\mathbf{v}$, all the receiver has to do to determine c is search C (which is a finite set) and find a code word $\mathbf{c}'$ which is as close as possible to $\mathbf{v}$. This strategy is called the **nearest neighbor algorithm**. If C is e-error correcting and $\mathbf{v}$ has no more than e errors then $\mathbf{c}' = \mathbf{c}$. If C is "too large" (where the definition of "large" is a function of the speed of your computer and your level of patience) then this algorithm is too slow for practical purposes.

The following theorem of Shannon is fundamental to the theory of error correcting codes.

Theorem 3.3.12. *(fundamental theorem of information theory) Consider a binary symmetric channel with $p > 1/2$. Let $\epsilon > 0$ and $\delta > 0$ be given. For all sufficiently large n, there is a code $C \subset \mathbb{F}_2^n$ with information rate R satisfying $cap(p) - \epsilon < R < cap(p)$, such that the nearest neighbor algorithm decoding has average probability of incorrect decoding less that δ.*

Shannon's theorem guarantees us that "very good" codes (in the sense that they are close to being best possible) exist. They may not be linear and even if they are, the theorem does not suggest that they are practical (i.e., "fast" encoding and decoding algorithms exist). However, we shall very briefly discuss in a later chapter "low density parity check codes", which can be both "very good" and practical. The proof of Shanon's theorem is not easy and goes beyond the scope of this book. For a proof and further discussion, see Ash [Ash], §3.5.

Figure 3.1 graphs the capacity for a binary code.

3.4 $[n, k, d]$-codes and error correction

How does the minimum distance related to the number of errors which can be corrected? This section addresses this and similar questions.

Definition 3.4.1. *Let F be a finite field. A code $C \subset F^n$ is called an* (n, M, d)**-code** *if it is length n, cardinality $|C| = M$, and minimum distance*

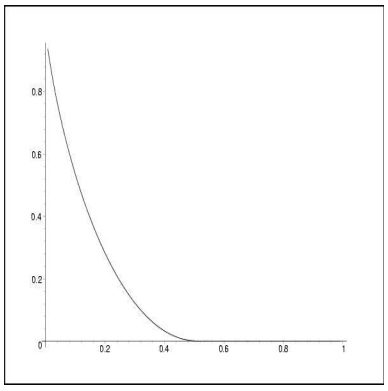


Figure 3.1: The capacity function for binary codes.

d. A linear code $C \subset F^n$ *is called an* $[n, k, d]$**-code** *if it is length* n*, dimension* k*, and minimum distance* d*. If* d *is unknown then we call a linear code* $C \subset F^n$ *of dimension* k *a* $[n, k]$**-code**.

For example, the ISBN code is a $[10, 9, 2]$-code over the field $\mathbb{F}_{11}$.

Proposition 3.4.2. *Let* $C \subset F^n$ *be a code which is defined by*

$$C = \{v \in F^n \mid Hv = 0\},$$

where H *is some* $r \times n$ *matrix with entries in* F*. The* C *is* 1*-error correcting if and only if all the columns of* H *are distinct.*

The matrix H in the above proposition is called a **parity check matrix** of C. We have yet to show that a given linear code C has a parity check matrix (see Proposition 3.6.3).

proof: (sketch) Generalizing suitably example 3.3.2 shows that if all the columns of H are not distinct then C is not 1-error correcting.

Conversely, suppose that v contains exactly one error and that all the columns of H are distinct. Then $v = c + e_i$, for some c and some i, where $c \in C$ and where

$$e_1 = (1, 0, ..., 0), \quad e_2 = (0, 1, 0, ..., 0), ..., e_n = (0, 0, ..., 0, 1).$$

To correct v, we need to determine i. Note $Hv = Hc + He_i = 0 + He_i = h_i$, where h_i denotes the i^{th} column of H. Since all the columns are H are distinct, h_i uniquely determines i. $\square$

Proposition 3.4.3. *Let $C \subset F^n$ be a code which is of minimum distance d. Then C is $[(d-1)/2]$-error-correcting.*

proof: Suppose that a code word $\mathbf{c} \in C$ is sent and a vector $\mathbf{v} \in F^n$ is received with e errors, where $e \leq [(d-1)/2]$. We must show that the receiver, who does not know $\mathbf{c}$ (though he does know C), can recover $\mathbf{c}$ from $\mathbf{v}$.

We claim that the code word closest to $\mathbf{v}$ is $\mathbf{c}$. Suppose not, say $\mathbf{c}' \in C$, $\mathbf{c} \neq \mathbf{c}'$, is closer to $\mathbf{v}$ than $\mathbf{c}$. Then

$$d(\mathbf{v}, \mathbf{c}') \leq d(\mathbf{v}, \mathbf{c}) \leq e,$$

so, by the triangle inequality, $d \leq d(\mathbf{c}, \mathbf{c}') \leq d(\mathbf{c}, \mathbf{v}) + d(\mathbf{v}, \mathbf{c}') \leq e + e = 2e \leq d - 1$. This is a contradiction, which proves the claim is true.

To recover $\mathbf{c}$ from $\mathbf{v}$, the receiver can apply the nearest neighbor algorithm. □

3.4.1 The generator matrix

Since a code is a finite dimensional vector space over a finite field, it only has finitely many elements. To represent a code in a computer one may of course store all the elements. But there must be a simpler way of representing a code than by listing all of its elements, right? Yes, there is and this motivates the following definition.

Definition 3.4.4. *If $\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_k$ is a basis for an $[n, k]$-code C, where*

$$\begin{aligned} \mathbf{f}_1 &= (f_{11}, f_{12}, ..., f_{1n}), \\ \mathbf{f}_2 &= (f_{21}, f_{22}, ..., f_{2n}), \\ &\vdots \\ \mathbf{f}_k &= (f_{k1}, f_{k2}, ..., f_{kn}), \end{aligned}$$

then the matrix

$$G = \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_k \end{pmatrix} = \begin{pmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \vdots & & & \\ f_{k1} & f_{k2} & \cdots & f_{kn} \end{pmatrix}$$

is called a **generator matrix** *for* C.

A code is often represented by simply giving a generator matrix. To compute a generator matrix of a given code C of length n, first determine a basis for the code as a vector space over $\mathbb{F}_q$, then put these basis vectors into a $k \times n$ matrix, where $k = \dim_{\mathbb{F}_q}(C)$.

Example 3.4.5. *For the ISBN code,*

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

is a generator matrix.

Definition 3.4.6. *If G is any matrix with entries in a field F then replacing any row of G by*

(a) its sum with another row, or

(b) its scalar multiple with any non-zero element of F,

is called an **elementary row operation**.

If G is any matrix with entries in a field F then

(a) swapping any two columns, or

(b) replacing any column of G by its scalar multiple with any non-zero element of F,

is called a **simple column operation**.

Lemma 3.4.7. *If G is a generator matrix for a linear code C then so is G', where G' is any matrix obtained from G by an elementary row operation.*

proof: The rows of G' still form a basis for the vector space C over F. □

Definition 3.4.8. *If G is a generator matrix for a linear code C and C' is the linear code with generator matrix G', where G' is any matrix obtained from G by elementary row operations or simple column operations, then we say that the codes C and C' are* **equivalent**.

Example 3.4.9. *We know a generator matrix G for the ISBN code by Example 3.4.5. By the above lemma, another generator matrix is given by*

$$G' = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

where we replaced the second row of G by its sum with the first row.

However, if we swap the first two columns of G, to get

$$G'' = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

then G'' generates a code C'' which is different from C. However, the codes C and C'' are equivalent.

Example 3.4.10. *Let $\mathcal{P} = \{P_1, ..., P_n\} \subset \mathbb{F}_q$ be a subset of n distinct points. Let*

$$L(a) = \{f \in \mathbb{F}_q[x] \mid \deg(f) \leq a\}$$

denote the vector space of all polynomials of degree less than or equal to a. The dimension of $L(a)$ is $a+1$ since the polynomials 1, x, ..., x^a form a basis. Assume $n > a$. Define the **evaluation map** *by*

$$E_{\mathcal{P}} : L(a) \to \mathbb{F}_q^n$$

$$f \longmapsto (f(P_1), ..., f(P_n)).$$

This is 1-1 if $n > a$ since no polynomial of degree a can have more than a zeros. Its image is denoted C. This is a linear $[n, a+1, n-a]$-code over $\mathbb{F}_q$.

These are called **generalized** *(or* **shortened** *or* **extended***)* **Reed-Solomon codes**. *(As we shall see later, in the context of cyclic codes, Reed-Solomon codes have $n = q - 1$.) Note the parameters satisfy the Singleton bound, so these are MDS codes.*

The generator matrix for C is of the form $G = (P_i^j)_{0 \leq j \leq a, 1 \leq i \leq n}$.

The **encoding matrix** of a linear code $C \subset F^n$ of dinension k is an $n \times k$ matrix $E : F^k \to F^n$ whose image is C. To compute the encoding matrix from the generator matrix is easy.

Proposition 3.4.11. *$E = G^t$.*

The proof of this is left as an exercise.

Example 3.4.12. *Consider the binary code with generator matrix*

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}.$$

Then

$$E = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

and the vector $v = (1, 1, 1)$ gets encoded as $Ev = (1, 1, 1, 0, 0, 0)$.

Many general properties about a linear code may be reduced to a corresponding property of its generator matrix. Can one determine the minimum distance of a code from its generator matrix? The following result answers this.

Theorem 3.4.13. *Let C be a linear $[n, k, d]$-code with parity check matrix H. Then any subset of $d-1$ columns of H are linearly independent but there are d columns of H which are linearly dependent.*

proof: By definition of d, there is a code word in C having exactly d non-zero entries. Thus by definition of H, there are d columns of H which are linearly dependent. If there were $d-1$ columns of H which are linearly independent then there would be (by definition of H) a code word in C having exactly $d-1$ non-zero entries. This would contradict the definition of d. $\square$

3.4.2 The binary Hamming $[7, 4, 3]$ code

This section provides a good detailed example of the above ideas. It is also historically one of the first error-correcting codes developed. Hamming deveoped the idea in the late 1940's. Hamming, who worked for many years as a mathematician for the telephone company, thought that computers should be able to correct bit errors. We was right and discovered a family of 1-error correcting codes, now called "Hamming codes". We shall focus only on one of them in this section and define the more general codes later.

Let $F = \mathbb{F}_2$ and let C be the set of all vectors in the third column below (for simplicity, we write 0000000 instead of $(0, 0, 0, 0, 0, 0, 0)$, for example).

decimal	binary	hamming (7,4)
0	0000	0000000
1	0001	0001110
2	0010	0010101
3	0011	0011011
4	0100	0100011
5	0101	0101101
6	0110	0110110
7	0111	0111000
8	1000	1000111
9	1001	1001001
10	1010	1010010
11	1011	1011100
12	1100	1100100
13	1101	1101010
14	1110	1110001
15	1111	1111111

This is a linear code of length 7, dimension 4, and minimum distance 3.

It is called the **Hamming** $[7,4,3]$**-code**. In fact, there is a mapping from F^4 to C given by $\phi(x_1,x_2,x_3,x_4)=\mathbf{y}$, where

$$\mathbf{y}=\begin{pmatrix} y_1\\ y_2\\ y_3\\ y_4\\ y_5\\ y_6\\ y_7 \end{pmatrix}=\begin{pmatrix} 1&0&0&0\\ 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1\\ 1&0&1&1\\ 1&1&0&1\\ 1&1&1&0 \end{pmatrix}\begin{pmatrix} x_1\\ x_2\\ x_3\\ x_4 \end{pmatrix} \pmod 2.$$

A basis for C is given by the vectors $\phi(1,0,0,0)=(1,0,0,0,1,1,1)$, $\phi(0,1,0,0)=(0,1,0,0,0,1,1)$, $\phi(0,0,1,0)=(0,0,1,0,1,0,1)$, $\phi(0,0,0,1)=(0,0,0,1,1,1,0)$. Therefore, the matrix

$$G=\begin{pmatrix} 1&0&0&0&1&1&1\\ 0&1&0&0&0&1&1\\ 0&0&1&0&1&0&1\\ 0&0&0&1&1&1&0 \end{pmatrix}$$

is a generator matrix. The matrix

$$E=\begin{pmatrix} 1&0&0&0\\ 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1\\ 1&0&1&1\\ 1&1&0&1\\ 1&1&1&0 \end{pmatrix}$$

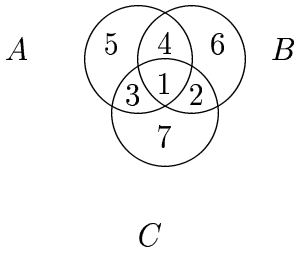
is an encoding matrix.

An algorithm for decoding the Hamming $[7,4,3]$ **code**

Denote the received word by $\mathbf{w}=(w_1,w_2,w_3,w_4,w_5,w_6,w_7)$.

1. Put w_i in region i of the Venn diagram above, $i=1,2,...,7$.

2. Do parity checks on each of the circles A, B, and C.



parity failure region(s)	error position
none	none
A, B, and C	1
B and C	2
A and C	3
A and B	4
A	5
B	6
C	7

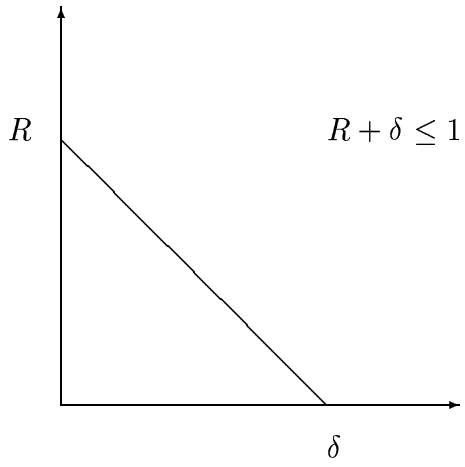
Exercise 3.4.14. *Decode* $(0,1,0,0,0,0,1)$ *in the binary Hamming code using this algorithm.*

3.5 Bounds on the parameters of a code

In this section, we prove the Singleton bound and the Gilbert-Varshamov bound. For a fixed length n, the Singleton bound is an upper bound on the parameters k, d. The Gilbert-Varshamov bound is a lower bound - telling us that there must exist a code with "good parameters".

Theorem 3.5.1. *(The Singleton bound) Every linear* $[n,k,d]$*-code* C *over* $\mathbb{F}_q$ *satisfies*

$$k + d \leq n + 1.$$



A code C whose parameters satisfy $k + d = n + 1$ is called **maximum distance separable** or **MDS**. Such codes, when they exist, are in some sense best possible.

proof: Fix a basis of $\mathbb{F}_q^n$ and write all the codewords in this basis. Delete the first $d-1$ coordinates in each code word. Call this new code C'. Since C has minimum distance d, these codewords of C' are still distinct. There are therefore q^k of them. But there cannot be more that $q^{n-d+1} = |\mathbb{F}_q^{n-d+1}|$ of them. This gives the inequality. $\square$

Before going on to an example, let us discuss a simple consequence of this. Let C_i be a sequence of linear codes with parameters $[n_i, k_i, d_i]$ such that $n_i \to \infty$ as i goes to infinity, and such that both the limits

$$R = \lim_{i\to\infty} \frac{k_i}{n_i}, \qquad \delta = \lim_{i\to\infty} \frac{d_i}{n_i},$$

exist. The Singleton bound implies

$$\frac{k_i}{n_i} + \frac{d_i}{n_i} \leq 1 + \frac{1}{n_i}.$$

Letting i tend to infinity, we obtain

$$R + \delta \leq 1.$$

This bound implies that any sequence of codes must have, in the limit, information rate and relative minimum distance in the triangle pictured below.

Theorem 3.5.2. *(Gilbert-Varshamov bound) There exists a code $C \subset \mathbb{F}_q^n$ such that*

$$\frac{q^n}{\sum_{i=0}^{d-1} \left(\begin{array}{c} n \\ i \end{array} \right) (q-1)^i} \leq |C|.$$

proof: Suppose that C has minimum distance d and block length n. Suppose moreover that C is as large as possible with these properties. In other words, we cannot increase the size of C by adding another vector into C and still keeping the minimum distance d. This implies that each $v \in F^n$ has distance $\leq d-1$ from some codeword in C. This implies

$$F^n \subset \cup_{c \in C} B(c, d-1, F^n).$$

Thus

$$q^n = |F^n| = |\cup_{c \in C} B(c, d-1, F^n)| \leq |C| \cdot |B(c_0, d-1, F^n)| \leq |C| \sum_{i=0}^{d-1} \left(\begin{array}{c} n \\ i \end{array} \right) (q-1)^i.$$

□

Taking δ fixed, and various values of $r = [\delta n]$, as n goes to infinity, gives Figure 3.2.

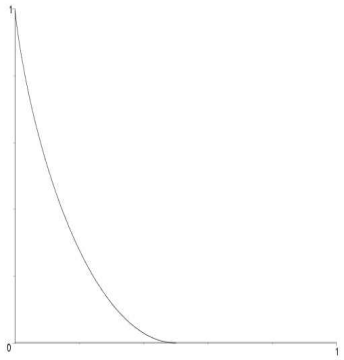


Figure 3.2: The Gilbert-Varshamov bound for binary codes.

Theorem 3.5.3. *(Hamming or sphere-packing bound) For any (not necessarily linear) code $C \subset \mathbb{F}_q^n$ having M elements, we have*

$$M \sum_{i=0}^{e} \left(\begin{array}{c} n \\ i \end{array} \right) (q-1)^i \leq q^n,$$

where $e = [(d-1)/2]$.

proof: For each codeword of C, construct a ball of radius e about it. These are non-intersecting, by definition of d and Proposition 3.4.3. By Lemma 3.3.5, each such ball has

$$M \sum_{i=0}^{e} \left(\begin{array}{c} n \\ i \end{array} \right) (q-1)^i$$

elements. The result follows from the fact that $C \subset \mathbb{F}_q^n$ and $|\mathbb{F}_q^n| = q^n$. $\square$

3.5.1 Question: What is "the best" code?

What is the "best" code of a given length? This natural, but very hard, question motivates the following definition.

Definition 3.5.4. *Let F be a finite field with q elements. Let $A_q(n,d)$ denote the largest M such that there exists a (n,M,d) code in F^n.*

Determining $A_q(n,d)$ is one of the main problems in the theory of error-correcting codes[1]. At the time of this writing, $A_2(n,d)$ is known for $n < 30$, d arbitrary. The previous example implies that $A_2(7,3) \geq 16$. (It turns out that $A_2(7,3) = 16$.)

Exercise 3.5.5. *Show that the following result holds. Let C be any (possibly non-linear) code $C \subset \mathbb{F}_q^n$. Then*

$$\log_q |C| + d \leq n + 1.$$

(Hint: Modify the proof of Theorem 3.5.1.)

[1] In general, much more is known about $A_q(n,d)$ for "very small" and "very large" values of d than for "arbitrary values" of d. See [MS] for further information.

Exercise 3.5.6. *Show that the minimum distance of the code in Example 3.4.10 is $n - a$.*

Exercise 3.5.7. *Let $q = 5$, $a = 2$ and $\mathcal{P} = 1, 2, 3$ in the code in Example 3.4.10. Compute the code words of C and the generator matrix.*

Exercise 3.5.8. *Find the parameters n, k, d of the ISBN code in Example 3.3.6.*

Exercise 3.5.9. *Show that as $n \to \infty$, the Gilbert-Varshamov bound shows that there is a code of parameters (n, k, d) over $\mathbb{F}_q$ for which the point $(d/n, k/n)$ lies above the Gilbert-Varshamov curve*

$$y = 1 - x\log_q(q-1) - x\log_q(x) - (1-x)\log_q(1-x), \qquad 0 \leq x \leq \frac{q-1}{q}.$$

3.6 The dual code

The space of all vectors orthogonal to a code C is another code. Since the ground field is finite, it is possible for all the codewords in a code to be orthogonal to themselves!

Definition 3.6.1. *If C is a $[n, k]$-code then the* **dual code** *$C^\perp$ is a $[n, n-k]$-code defined by*

$$C^\perp = \{\mathbf{v} \in F^n \mid \mathbf{v} \cdot \mathbf{c} = 0, \ \forall \mathbf{c} \in C\},$$

where

$$\mathbf{v} \cdot \mathbf{w} = v_1 w_1 + v_2 w_2 + ... + v_n w_n \in F,$$

for all $\mathbf{v} = (v_1, ..., v_n)$ and $\mathbf{w} = (w_1, ..., w_n)$.

Example 3.6.2. *As a very simple example, let C be the binary repetition code of length 2:*

$$C := \{(0,0), (1,1)\}.$$

This is self-dual: $C = C^\perp$.

Finally, we can show that a parity check matrix exists.

Proposition 3.6.3. *Let C be a linear code. A parity check matrix of C exists.*

proof: Any generator matrix for $C^\perp$ is a parity check matrix of C. □

Exercise 3.6.4. *Show $(C^\perp)^\perp = C$.*

3.7 Computing the check matrix and the encoding matrix

The following property of H is perhaps the key reason for introducing the dual code.

Lemma 3.7.1. *For all $\mathbf{v} \in F^n$, we have $\mathbf{v} \in C$ if and only if $\mathbf{v} \cdot H^t = \mathbf{0}$.*

The proof is left as an exercise.

In particular, a vector $\mathbf{v} \in F^n$ is a code word if and only if it satisfies the conditions $\mathbf{v} \cdot \mathbf{h_i} = 0$ $(1 \leq i \leq n-k)$, where $\mathbf{h_i} \in F^n$ is the $i-th$ row of H. This is a very convenient condition for checking if an error has been made in transmission.

Sometimes a generator matrix G can, after elementary row operations, be put in the form

$$G = (I_k|A),$$

where I_k is the $k \times k$ identity matrix and A is an $k \times (n-k)$ matrix. If this can be done, then we say that the generator matrix can be put in **standard form**. In this case, the parity check matrix can be computed.

Proposition 3.7.2. *If $G = (I_k \mid A)$ is the generator matrix for C then $H = (-A^t \mid I_{n-k})$ is a parity check matrix.*

The proof of this will be left as an exercise.

One other interesting fact about codes in standard form is that the information bits of the codewords are easy to distinguish. If we denote the row vectors of $G = (I_k \mid A)$ by $g_1, g_2, ..., g_k$ then these form a basis for C. The information bits of a codeword are the first k coordinates. Moreover, to to encode a message $m = (m_1, ..., m_k)$ into a codeword, simply compute the n-tuple $G^t m$ (or mG, depending on if you're writing m as a row vector or a column vector).

Example 3.7.3. *If C is the code over $\mathbb{F}_2$ with generator matrix*

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

The codewords are all of the form $(c_1, c_2, c_3, c_4, c_1 + c_3 + c_4, c_1 + c_2 + c_3, c_2 + c_3 + c_4)$, *where* $c_i \in \mathbb{F}_2$. *In other words, all codewords are of the form* $G^t m$, *where* $m = (c_1, c_2, c_3, c_4) \in \mathbb{F}_2^4$. *A check matrix for* C *is given by*

$$H = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Since $(C^\perp)^\perp = C$, we have, as a consequence of this, the following result.

Corollary 3.7.4. *If* $H = (I_k \mid B)$ *is the parity check matrix for* C *then* $G = (-B^t \mid I_{n-k})$ *is a generator matrix.*

Example 3.7.5. *Consider the subcode of the ISBN code with generator matrix*

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

We can add the first, second and third rows to the last two rows to put this in the upper-triangular form

$$G' = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

This is not in standard form. In fact, the code C *does not have a generator matrix in standard form. However, let* C' *be the code obtained from* C *by swapping the* 6^{th} *and* 8^{th} *coordinates. These codes* C *and* C' *are equivalent. We leave it as an exercise to show that* C' *has a generator matrix in standard form.*

Example 3.7.6. *One way to define the* **Reed-Muller code** *is as follows. Let $\{s_i\}$ be a sequence defined by an equation such as (1.3) in the above section on linear recussion §1.7.6, where $a_i \in \mathbb{F}_q$ and $s_i \in \mathbb{F}_q$. Assume that the characteristic polynomial of A in §1.7.6 is irreducible and primitive. Let N be the period. The set*

$$RM(k,1) = \{(s_1, ..., s_N) \text{ satisfying (1.3)} \mid (s_1, ..., s_k) \in \mathbb{F}_q^k \}$$

is a vector space, called the **first order Reed-Muller code**.

These were used by Marinar 9 on its 1972 flight to Mars.

Exercise 3.7.7. *Let C' be as in Example 3.7.5. Show that C' has a generator matrix in standard form.*

Exercise 3.7.8. *Prove Lemma 3.7.1. (Hint: The proof uses from the definition of H.)*

Exercise 3.7.9. *Prove Proposition 3.7.2 (Hint: The proof uses from the definition of H.)*

Exercise 3.7.10. *Show that the code C generated by*

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

satisfies $C = C^\perp$. Show that $n = 8$, $k = 4$, and $d = 4$.

3.8 Syndrome decoding

Let C be a linear code of length over $\mathbb{F}_q$ having check matrix H. Let $V = \mathbb{F}_q^n$.

Definition 3.8.1. *Let $\mathbf{v} \in V$. The set*

$$\mathbf{v} + C = \{\mathbf{v} + \mathbf{c} \mid \mathbf{c} \in C\},$$

is called the **coset of v**. *The set of all cosets is denoted V/C. The vector $S(\mathbf{v}) = H\mathbf{v}$ is called the* **syndrome of v**. *The set of all cosets is denoted $Im(H)$ or $H(V)$.*

Proposition 3.8.2. *The cosets are in 1-1 correspondence with the cosets: there is a bijection*

$$\rho : V/C \to H(V),$$

given by $\rho(\mathbf{v} + C) = H(\mathbf{v})$.

proof: First, we show that the map is well-defined (i.e., independent of the choice of coset representative. For all $\mathbf{u}, \mathbf{v} \in V$, we have $\mathbf{u} + C = \mathbf{v} + C$ if and only if $\mathbf{u} - \mathbf{v} \in C$ if and only if $H(\mathbf{u} - \mathbf{v}) = 0$ if and only if $H\mathbf{u} = H\mathbf{v}$. Thus ρ is well-defined. ρ is 1-1 by the same reasoning. ρ is onto by definition. □

Let $\mathbf{v} \in V$. An element in $\mathbf{v}+C$ of lowest weight is called a **coset leader**, and intuitively represents the "mostly likely error vector".

Algorithm:

- Precomputation. Compute all syndromes $\mathbf{s} = H\mathbf{w}$, for $\mathbf{w} \in V$ and the corresponding coset leaders $L(\mathbf{s})$. (The table of values of the function f is called the **look-up table**.)
- If $\mathbf{v}$ is the received vector, compute $\mathbf{s} = H\mathbf{v}$.
- Let $\mathbf{e} = L(\mathbf{s})$ be the corresponding coset leader obtained from the look-up table.
- Decode $\mathbf{v}$ as $\mathbf{c} = \mathbf{v} - \mathbf{e}$.

This algorithm is fast, assuming that assessing the look-up table takes no time, but may require a lot of time and memory initially to build the look-up table in the first place.

Example 3.8.3. *If C is the code over $\mathbb{F}_2$ with check matrix given by*

$$H = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Here is the look-up table:

syndrome	*coset leader*
$(0,0,0)$	$(0,0,0,0,0,0,0)$
$(1,0,0)$	$(0,0,0,0,1,0,0)$
$(0,1,0)$	$(0,0,0,0,0,1,0)$
$(0,0,1)$	$(0,0,0,0,0,0,1)$
$(1,1,0)$	$(1,0,0,0,0,0,0)$
$(1,0,1)$	$(0,0,0,1,0,0,0)$
$(0,1,1)$	$(0,1,0,0,0,0,0)$
$(1,1,1)$	$(0,0,1,0,0,0,0)$

Exercise 3.8.4. *Let C be the code in Example 3.8.3. Decode $\mathbf{v} = (1,1,1,1,1,0,0)$ using the syndrome decoding algorithm. (Ans: $\mathbf{v} = (1,0,1,1,1,0,0)$.)*

Exercise 3.8.5. *Count the number of vectors in a ball of radius 1 in $\mathbb{F}_2^7$.*

Exercise 3.8.6. *Show that the number of vectors in a ball of radius r in F^n does not depend on the radius. In other words, if $v, v' \in F^n$ then show $|B(v,r,F^n)| = |B(v',r,F^n)|$.*

Exercise 3.8.7. *In Example 3.3.2, the code C has minimum distance 2.*

Exercise 3.8.8. *In Example 3.3.6, the code C has minimum distance 2.*

Exercise 3.8.9. *Pick a book and check that its ISBN satisfies the condition $x_1 + 2x_2 + 3x_3 + ... + 9x_9 + 10x_{10} \equiv 0 \pmod{11}$, in the notation of Example 3.3.6.*

Exercise 3.8.10. *For the code in Example 3.3.2, use the nearest neighbor algorithm to decode $(1,0,0,1,0,0)$ and $(1,0,1,0,1,1)$.*

Exercise 3.8.11. *Consider the code with generator matrix*

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

Find the codeword obtained by encoding $(1,0,1,1)$.

Exercise 3.8.12. *For G for the code in Example 3.3.2.*

Exercise 3.8.13. *Prove this Proposition 3.4.11.*

Exercise 3.8.14. *The vector 1010011 was receieved. Assuming only one error was made, use the nearest neighbor algorithm to decode it.*

Exercise 3.8.15. *Show that the Hamming $[7,4,3]$-code has minimum distance 3.*

3.9 Hamming codes

In this section, we present a commonly used linear error-correcting code which corrects exactly one error. In addition, it is the "shortest" code (linear or not) to do so. It was discovered by Hamming in the 1940's, in the days when an computer error would crash the computer and force the programmer to retype his punch cards. Out of frustration, he tried to design a system whereby the computer could automatically correct certain errors.

3.9.1 Binary hamming codes

The easiest way to define the Hamming code, which is a vector space over $\mathbb{F}_2$, is by defining a generator matrix. Incidently, calling it "the" Hamming code is a slight abuse of terminology. First, for each integer $r \geq 2$ there is a different Hamming code. Second, we often implicitly identify two equivalent codes, so two equivalent forms of a Hamming code are regarded as the same.

Definition 3.9.1. *Let $r > 1$. The* **Hamming $[n, k, 3]$-code** *C is the linear code with*

$$n = 2^r - 1, \qquad k = 2^r - r - 1,$$

and parity check matrix H defined to be the matrix whose columns are all the (distinct) non-zero vectors in $\mathbb{F}_2^r$. (Later we will show that it has minimum distance $d = 3$.)

Example 3.9.2. *$r = 2$: The Hamming $(3, 1)$-code has parity check matrix*

$$H = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

By Corollary 3.7.4, the matrix $G = (1, 1, 1)$ is a generating matrix.

$r = 3$: The Hamming $(7, 4)$-code has parity check matrix

$$H = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

By Corollary 3.7.4, the matrix

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

is a generating matrix. Though this may not look like the code presented in an earlier section, they are equivalent.

Lemma 3.9.3. *Every binary Hamming code C has minimum distance* 3.

proof: Indeed, if C has a code wode of weight 1 then the parity check matrix H of C would have to have a column which consists of the zero vector, contradicting the definition of H. Likewise, if C has a code wode of weight 2 then the parity check matrix H of C would have to have two identical columns, contradicting the definition of H. Thus $d \geq 3$.

Since

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \text{ and } \begin{pmatrix} 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

form three columns of the parity check matrix H of C - say the 1^{st}, 2^{nd}, and 3^{rd} columns - the vector $(1, 1, 1, 0, ..., 0)$ must be a code word. Thus $d \leq 3$. #

Remark 3.9.4. *Dual to the binary Hamming code is the "simplex code". The line segments connecting the codewords in a simplex code form a regular simplex.*

3.9.2 q-ary Hamming codes

Let p be a prime and let $\mathbb{F}_p^r - \{0\}$ denote the set of all non-zero r-tuples, i.e., the non-zero vectors in the r-dimensional vector space $\mathbb{F}_p^r$ over $\mathbb{F}_p$. We may represent each vector by an r-tuple of integers belonging to $\{0, 1, ..., p-1\}$. If $\mathbf{a} = (a_1, ..., a_r) \in \mathbb{F}_p^r - \{0\}$ and $0 \leq a_i < p - 1$ the define

$$n(\mathbf{a}) = a_1 + a_2 p + ... + a_r p^{r-1}.$$

This is a map $x : \mathbb{F}_p^r - \{0\} \to \{0, 1, ..., p^r - 1\}$. Let $\mathbf{a} = (a_1, ..., a_r)$ and $\mathbf{b} = (b_1, ..., b_r)$ be two such vectors. Define $a <_x b$ if $x(\mathbf{a}) < x(\mathbf{b})$. Now take each vector $\mathbf{a} = (a_1, ..., a_r) \in \mathbb{F}_p^r - \{0\}$ and divide every entry by it's first non-zero entry. Let S be the set of them. There are $n = (p^r - 1)/(p - 1)$ elements in S. Write the elements of the set S in increasing order, using the ordering $<_x$ above. Let H be the $r \times N$ matrix whose i^{th} column is the i^{th}

vector in S (written as a column vector). We know that the first column of H is

$$\begin{matrix} 1 \\ 0 \\ \vdots \\ 0 \end{matrix}$$

and the last column is

$$\begin{matrix} 1 \\ p-1 \\ \vdots \\ p-1 \end{matrix}$$

Example 3.9.5. *For example, if $p = 3$ and $r = 2$ then*

$$H = \begin{matrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 2 & 1 \end{matrix}$$

is a parity check matrix and

$$H = \begin{matrix} 1 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{matrix}$$

is a generator matrix for the $(4, 2, 3)$ Hamming code over $\mathbb{F}_3$.

Exercise 3.9.6. *Write down H if $p = 3$ and $r = 3$.*

The code C whose parity check matrix is this matrix H constructed above is the **p-ary Hamming code** of length $n = (p^r - 1)/(p - 1)$ and dimension $k = n - r$. More generally, we have the following definition.

Definition 3.9.7. *Let $r > 1$ and let q be a prime power. The* **Hamming $[n, k, 3]$-code** *C over $\mathbb{F}_q$ is the linear code with*

$$n = (q^r - 1)/(q - 1), \qquad k = n - r,$$

and parity check matrix H defined to be the matrix whose columns are all the (distinct) non-zero vectors in $\mathbb{F}_q^r$, normalized to have first non-zero coordinate equal to 1.

Like its binary cousin, it has minimum distance 3. It is left as an exercise to prove the analog of Lemma 3.9.3 in this case.

Remark 3.9.8. *It is also possible to define the RM code of order* 1 *in Example 3.7.6 as the dual code of the Hamming code extended by one parity check bit: Let* C *be the Hamming code over* $\mathbb{F}_q$ *of length* $n = (q^r - 1)/(q-1)$. *Extend this code to a length* $n+1$ *code,* C', *where the extra "bit" is defined by a parity check:* $c_1 + ... + c_n + c_{n+1} = 0$, *where* $(c_1, ..., c_n) \in C$. *It is known that* $(C')^\perp$ *is equivalent to the first order Reed-Muller code (this is a special case of Theorem 11 in chapter 13 of [MS]).*

3.9.3 Decoding p-ary Hamming codes

Here's the decoding algorithm: Let y be the received vector. Let

$$\mathbf{e}_1 = (1, 0, ..., 0), \quad \mathbf{e}_2 = (0, 1, 0, ..., 0), ..., \mathbf{e}_n = (0, ..., 0, 1),$$

be the standard basis vectors in $\mathbb{F}_p^n$. Assume that y has ≤ 1 error.

(1) Compute $y \cdot H^t$. This is an n-tuple, so it must be of the form $c \cdot \mathbf{s}$, for some $\mathbf{s} \in S$ and some $c \in \mathbb{F}_p^\times$.

(2) If $\mathbf{s}$ is the i^{th} element of S then the decoded vector is $\mathbf{y} - c \cdot \mathbf{e}_i$.

Exercise 3.9.9. *Decode* $(0, 1, 0, 0, 0, 0, 1)$ *in the binary Hamming code of §3.4.2 using this algorithm.*

Exercise 3.9.10. *Decode* $(0, 0, 1, 1)$ *in the* 3*-ary* $(4, 2, 3)$*-Hamming code of Example 3.9.5 using this algorithm.*

3.10 Cyclic codes

Let F be a finite field with q elements.

Here's a constructive definition of a **cyclic code** of length n.

- Pick a monic polynomial $g(x) \in F[x]$ dividing $x^n - 1$. This is called the **generating polynomial** of the code.
- For each polynomial $p(x) \in F[x]$, compute

$$p(x)g(x) \pmod{x^n - 1}.$$

  Denote the answer by $c_0 + c_1 x + ... + c_{n-1}x^{n-1}$.

- $\mathbf{c} = (c_0, c_1, ..., c_{n-1})$ is a code word in C. Every codeword in C arises in this way (from some $p(x)$).

Since polynomial multiplication can be performed quickly on a computer, this construction illustrated one reason why cyclic codes have such fast encoding algorithms. The "polynomial notation" for the code is to call $c_0 + c_1 x + ... + c_{n-1}x^{n-1}$ the code word (instead of $(c_0, c_1, ..., c_{n-1})$).

Example 3.10.1. *Let $g(x) = x^3 + 1 \in \mathbb{F}_2[x]$. This divides $x^{15} - 1$. Let us compute the four codewords corresponding to $p(x) = 1, x, x^2, x^{14}$. The first 3 are easy:*

$$1 \cdot g(x) = x^3 + 1 \leftrightarrow \mathbf{c} = (1,0,0,1,0,0,0,0,0,0,0,0,0,0,0),$$

$$x \cdot g(x) = x^4 + x \leftrightarrow \mathbf{c} = (0,1,0,0,1,0,0,0,0,0,0,0,0,0,0),$$

$$x^2 \cdot g(x) = x^5 + x^2 \leftrightarrow \mathbf{c} = (0,0,1,0,0,1,0,0,0,0,0,0,0,0,0),$$

$$x^{14} \cdot g(x) = x^{17} + x^{14} \pmod{x^{15}-1} = x^{14} + x^2 \leftrightarrow \mathbf{c} = (0,0,1,0,0,1,0,0,0,0,0,0,0,0,1).$$

Another way to characterize a cyclic code is as follows: A q-ary **cyclic code** is a linear code $C \subset F^n$ with the following property: if $\mathbf{c} = (c_0, c_1, ..., c_{n-1}) \in C$ then $(c_{n-1}, c_0, ..., c_{n-2}) \in C$, for all $\mathbf{c} \in C$. Though the above constructive definition might be easier to understand, this one is much simpler.

If $\mathbf{c} = (c_0, c_1, ..., c_{n-1}) \in C$ is a non-zero code word in C such that the degree of

$$g(x) = c_0 + c_1 x + ... + c_{n-1}x^{n-1} \in F[x]$$

is minimum and such that $g(x)$ is monic then $g(x)$ is called a **generating polynomial** of C.

Remark 3.10.2. *We shall not worry too much about showing that these definitions describe the same code. However, here are some hints for the reader who wants to prove it for him/herself. The fact that this polynomial $g(x)$ satisfies the property described in the above constructive definition requires Lemma 3.10.4 below. You must also show that the space V in the Lemma is a principal ideal generated by $g(x)$.*

Example 3.10.3. *A simple example is the code*

$$C = \{(0,0,0,0), (1,0,1,0), (0,1,0,1), (1,1,1,1)\} \subset \mathbb{F}_2^4.$$

The polynomial $g(x) = 1 + x^2$ is a generating polynomial.

It is easy to generate a cyclic code. Pick a vector $\mathbf{w}_1 = (v_1, v_2, ..., v_r, 0, ...0) \in F^n$. Form $n - r$ cyclically shifted vectors

$$\mathbf{w}_2 = (0, v_1, v_2, ..., v_r, 0, ...0), \ ..., \mathbf{w}_{n-r+1} = (0, ..., 0, v_1, v_2, ..., v_r).$$

These form the basis of a cyclic code $C = span(\mathbf{w}_1, ..., \mathbf{w}_{n-r+1})$.

Let us associate to each code word $\mathbf{c} = (c_0, c_1, ..., c_{n-1}) \in C$ the polynomial

$$p_{\mathbf{c}}(x) = c_0 + c_1 x + ... + c_{n-1}x^{n-1} \in F[x].$$

This correspondence is "linear", i.e., the sum of the code words is associated to the sum of the polynomials, $p_{\mathbf{c}+\mathbf{c}'}(x) = p_{\mathbf{c}}(x) + p_{\mathbf{c}'}(x)$ for all $\mathbf{c}, \mathbf{c}' \in C$, and the scalar multiple of a code word is associated to the scalar muliple of the polynomial, $p_{a\mathbf{c}}(x) = ap_{\mathbf{c}}(x)$ for all $\mathbf{c} \in C$ and $a \in F$. Note that

$$xp_{\mathbf{c}}(x) \equiv c_0x+c_1x^2+...+c_{n-1}x^n \equiv c_{n-1}+c_0x+c_1x^2+...+c_{n-2}x^{n-1} \pmod{x^n-1},$$

since for polynomials modulo $x^n - 1$ we may identify x^n with 1.

In other words, we have the following result.

Lemma 3.10.4. *A cyclic code $C \subset F^n$ may be identified with (via $\mathbf{c} \longmapsto p_{\mathbf{c}}(x)$) a F-vector space of polynomials $V \subset F[x]/(x^n - 1)$. In other words, the elements of C are in 1–1 correspondence with the elements of V and this correspondence preserves the vector space operations of addition and scalar multiplication.*

In fact, the image V of C is an ideal in the ring $F[x]/(x^n-1)$. Conversely, any such ideal is the image of some cyclic code under the map $c \longmapsto p_c(x)$.

Many codes we've run across already are equivalent to a cyclic code, including the Hamming codes and their dual codes.

Example 3.10.5. *The binary code C of length 7 with generator polynomial $x^3 + x + 1$ has generator matrix*

$$G = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Its elements, sorted according to their weight, are

$$C = \{(0,0,0,0,0,0,0), (0,0,0,1,0,1,1), (0,0,1,0,1,1,0),$$
$$(0,1,0,1,1,0,0), (0,1,1,0,0,0,1), (1,0,1,1,0,0,0),$$
$$(1,0,0,0,1,0,1), (1,1,0,0,0,1,0), (0,0,1,1,1,0,1),$$
$$(0,1,0,0,1,1,1), (0,1,1,1,0,1,0), (1,0,1,0,0,1,1),$$
$$(1,0,0,1,1,1,0), (1,1,1,0,1,0,0), (1,1,0,1,0,0,1),$$
$$(1,1,1,1,1,1,1)\}$$

This code is equivalent to the Hamming $(7,4,3)$*-code.*

Let C be a cyclic code of length n with generator $g(x) \in \mathbb{F}_q[x]$. The polynomial

$$h(x) = (x^n - 1)/g(x),$$

is called the **check polynomial** of C. This terminology is motivated by the following property.

Proposition 3.10.6. *Let* C *be as above. Let* $\mathbf{v} = (v_0, v_1, ..., v_{n-1}) \in \mathbb{F}_q^n$. $\mathbf{v} \in C$ *if and only if*

$$(v_0 + v_1 x + ... + v_{n-1}x^{n-1})h(x) \equiv 0 \pmod{x^n - 1}.$$

proof: By the constructive definition of a cyclic code above, we know $\mathbf{v} \in C$ if and only if $v_0 + v_1 x + ... + v_{n-1}x^{n-1} = p(x)g(x)$, for some polynomial $p(x) \in \mathbb{F}_q[x]$. In this case,

$$(v_0 + v_1 x + ... + v_{n-1}x^{n-1})h(x) = p(x)g(x)h(x) = p(x)(x^n - 1) \equiv 0 \pmod{x^n - 1}.$$

Conversely, if $(v_0 + v_1 x + ... + v_{n-1}x^{n-1})h(x) \equiv 0 \pmod{x^n - 1}$ then there is a polynomial $p(x)$ for which $(v_0 + v_1 x + ... + v_{n-1}x^{n-1})h(x) = p(x)(x^n - 1)$. By definition of $h(x)$, this is equivalent to saying $v_0 + v_1 x + ... + v_{n-1}x^{n-1} = p(x)g(x)$. $\square$

Example 3.10.7. *Let* $g(x) = (x-2)(x-4) = x^2 + 4x + 3 \in \mathbb{F}_5[x]$. *This polynomial divides* $x^4 - 1$ *in* $\mathbb{F}_5[x]$. *Let* C *denote the cyclic code of length* 4 *generated by* $g(x)$. *Its check polynomial is*

$$h(x) = (x^4 - 1)/g(x) = x^2 + x + 3.$$

The codeword $c = (1, 4, 0, 2)$ *(note* $1 + 4x + 2x^3 = (2 + 2x)g(x)$*, so* c *is a codeword by the constructive definition of a cyclic code) satisfies* $(1 + 4x + 2x^3)h(x) \equiv 0$ *since*

$$(1 + 4x + 2x^3)(x^2 + x + 3) \equiv 10x^3 + 5x^2 + 15x + 5 \pmod{x^4 - 1},$$

which is 0 *in* $\mathbb{F}_5[x]$.

Definition 3.10.8. *Let* n *be a positive integer relatively prime to* q *and let* α *be a primitive* n*-th root of unity. Each generator polynomial* g *of a cyclic code* C *of length* n *has a factorization of the form*

$$g(x) = (x - \alpha^{k_1})...(x - \alpha^{k_r}),$$

where $\{k_1, ..., k_r\} \subset \{0, ..., n-1\}$*. The numbers* α^{k_i}*,* $1 \leq i \leq r$*, are called the* **zeros** *of the code* C*. They do not depend on the choice of* g*.*

Definition 3.10.9. *If* $k_1 = b$*,* $k_2 = b + 1$*, ...,* $k_{b+\delta} = b + \delta - 2$*, then* C *is called a* **Reed-Solomon code of designed distance** δ*. Here* $b \geq 1$ *is an integer (often* $b = 1$*).*

3.10.1 Quadratic residue codes

Let $m > 2$ be a prime number. The integers $1 \leq a \leq m - 1$ for which $a \equiv x^2 \pmod{m}$ for some $x \in \mathbb{Z}$, are called **quadratic residues mod** m . The remaining elements of $\{1, 2, ..., m - 1\}$ are called **quadratic non-residues mod** m.

Let n be a positive integer relatively prime to q and let α be a primitive n-th root of unity. Let p and n be distinct primes and assume that p is a quadratic residue mod n. The **quadratic residue code** of length n over $\mathbb{F}_p$ is the cyclic code whose generator polynomial has zeros

$$\{\alpha^k \mid k \text{ is a square mod } n\}.$$

Example 3.10.10. *The* **binary Golay code** GC_{23} *is the quadratic residue code of length 23 over* $\mathbb{F}_2$*. It is a* $[23, 12, 7]$*-code. A generating matrix for*

GC_{23} is

$$\begin{pmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1\\
0&1&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&1&1&1&0&0&0&1\\
0&0&1&0&0&0&0&0&0&0&0&0&0&1&1&0&1&1&1&0&0&0&1&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&1&1&1&0&0&0&1&0&1\\
0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&1&0&1&1\\
0&0&0&0&0&1&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1&1&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&0&0&0&1&0&1&1&0&1\\
0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&1&0&1&1&0&1&1\\
0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&1&0&1&1&0&1&1&1\\
0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&1&0&1&1&0&1&1&1&0\\
0&0&0&0&0&0&0&0&0&0&1&0&1&1&0&1&1&0&1&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&1&0&1&1&1&0&0&0&1
\end{pmatrix}.$$

The **binary Golay code** *GC_{24} is the code of length 24 over $\mathbb{F}_2$ obtained by appending onto GC_{23} a zero-sum check digit. It is a $[24, 12, 8]$-code. Moreover, it is self-dual in the sense that $GC_{24}^{\perp} = GC_{24}$. A generating matrix for GC_{24} is*

$$\begin{pmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1\\
0&1&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&1&1&1&0&0&0&1&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&1&1&0&1&1&1&0&0&0&1&0&1\\
0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&1&1&1&0&0&0&1&0&1&1\\
0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&1&0&1&1&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&1&1&1&0&0&0&1&0&1&1&0&1\\
0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&0&0&0&1&0&1&1&0&1&1\\
0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&1&0&1&1&0&1&1&1\\
0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&1&0&1&1&0&1&1&1&0\\
0&0&0&0&0&0&0&0&0&1&0&0&1&0&1&0&1&1&0&1&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&1&1&0&1&1&0&1&1&1&0&0&0&1
\end{pmatrix}.$$

Example 3.10.11. *The* **ternary Golay code** *GC_{11} is the quadratic residue code of length 11 over $\mathbb{F}_3$. It is a $[11, 6, 5]$-code. A generating matrix for GC_{11} is*

$$\begin{pmatrix}
1&0&0&0&0&0&1&1&1&1&1\\
0&1&0&0&0&0&0&1&2&2&1\\
0&0&1&0&0&0&1&0&1&2&2\\
0&0&0&1&0&0&2&1&0&1&2\\
0&0&0&0&1&0&2&2&1&0&1\\
0&0&0&0&0&1&1&2&2&1&0
\end{pmatrix}.$$

The **ternary Golay code** GC_{12} *is the code of length 12 over* $\mathbb{F}_3$ *obtained by appending onto* GC_{11} *a zero-sum check digit. It is a* $[12, 6, 6]$*-code. Moreover, it is self-dual in the sense that* $GC_{12}^{\perp} = GC_{12}$*. A generating matrix for* GC_{12} *is*

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 2 & 2 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 2 & 2 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 2 & 1 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 1 & 0 \end{pmatrix}.$$

Exercise 3.10.12. *Find all quadratic residues mod* m*, where*

(a) $m = 11$*,*

(b) $m = 13$*,*

(c) $m = 23$*.*

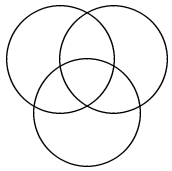
Exercise 3.10.13. *Show that if* a, b *are quadratic residues mod* m *then so is* ab*.*

Exercise 3.10.14. *Verify the generator matrix in Example 3.10.5 above is correct.*

Exercise 3.10.15. *Show that the minimum distance of* C *in Example 3.10.7 is* 3*.*

Exercise 3.10.16. *(a) Show that the code in Example 3.10.5 is equivalent to the Hamming* $[7, 4, 3]$*-code in §3.4.2.*

(b) Find the analog of the decoding diagram in Figure 3.4.2. Fill in the following blanks:



3.11 Application: The hats problem

Students from a coding theory class meet in a classroom. To be specific, suppose that there are 7 of them. Each are told to close their eyes and a white or black hat is placed on their head. The color of each hat is determined by a coin toss, with the outcome of one coin toss having no effect on the others. Each can see the other players' hats but not his own. No communication of any sort is allowed, except for an initial strategy session before the game begins. Once they have had a chance to look at the other hats, the players must simultaneously guess the color of their own hats or pass. The group shares a hypothetical $ 1 million prize if at least one player guesses correctly and no players guess incorrectly. The general problem is to find a strategy for the group that maximizes its chances of winning the prize.

Here is a solution that connects to coding theory: A strategy induced by a binary code C of length 7 is as follows. Player i looks at the hat colors of the other players, and obtains a vector $(x_1, .., x_{i-1}, *, x_{i+1}, ..., x_n)$, wherein x_j is the hat color of player j, and $*$ is the unknown hat color of player i. If this vector has distance at most 1 from a codeword, i.e., if $*$ can be chosen such that the resulting word is a codeword, then player i guesses his hat color to be the opposite of $*$; otherwise, the player passes.

What is the success probability of this strategy? All players guess wrong, if the vector of hat colors is in fact a codeword. All players pass if the vector of hat colors has distance larger than 1 from a codeword (and the team loses). All guessing players guess correct only if the vector has distance exactly one from a codeword. Therefore, the probability of success of this

strategy is equal to the fraction of words of distance 1 from the code. The $[7, 4, 3]$ Hamming code maximize this fraction. It turns out that if we replace 7 by n students and let n go to infinity, there is an analogous strategy using punctured Hamming codes for which the success probability goes to 1.

3.12 Application: Searching with lies

In Stanislaw Ulam's 1976 autobiography [U] (and, in a completely different form, in Elwyn Berlekamp's 1964 PhD thesis [B]), one finds the following problem concerning two players playing a generalization of the childhood game of "20 questions".

PROBLEM Player 1 secretly chooses a number from 1 to M (M is large but fixed). Player 2 asks a series of "yes/no questions" in an attempt to determine that number. Player 1 *may* lie at most e times ($e \geq 0$ is fixed). What is the minimum number of "yes/no questions" Player 2 must ask to (always) be able to correctly determine the number Player 1 chose?

Definition 3.12.1. *If answers between successive questions ("feedback") is allowed, call this minimum number of "yes/no questions" player 2 must ask, $f(M, e)$. If feedback is not allowed, call this minimum number $g(M, e)$.*

Clearly,

$$f(M, e) \leq g(M, e).$$

We shall only discuss the non-feedback case and shall not mention $f(M, e)$ further. In practice, we not only want to know $g(M, e)$ but an efficient algorithm for determining the "yes/no questions" Player 2 should ask.

In the feedback allowed case, Berlekamp was concerned with a discrete analog of a problem which arises when trying to design a analog-to-digital converter for sound recording which compensates for possible feedback.

Lemma 3.12.2. *Fix any e and M. $g(M, e)$ is the smallest n such that $A_2(n, 2e + 1) \geq M$.*

We shall not prove this here.

Record the n answers to the "yes/no questions" as an n-tuple of 0's (for "no") and 1's (for "yes"). The binary code C is the set of all codewords corresponding to *correct* answers. It is known that if a binary code $C \subset \mathbb{F}_2^n$ is e-error correcting then $|C| \leq A_2(n, 2e + 1)$. We want the smallest possible n satisfying this condition.

The above-mentioned fact allows us to reduce the problem of determining the solution to the seaching with lies without feedback to the (very hard) problem below.

PROBLEM Determine $A_2(n, d)$ for all n, d.

As we already mentioned, this is perhaps the central problem in coding theory and is unknown for most $n > 30$. In addition, of course one would also like constructive fact encoding and decoding procedures for the code $C \subset \mathbb{F}_2^n$ such that $|C| = A_2(n, 2e + 1)$.

3.12.1 The case of one lie

The theory of Hamming codes allows us to solve the searching with lies (without feedback) problem when only one lie is told. In the notation of Definition 3.12.1 above, we have the following solution to searching with one lie.

Lemma 3.12.3. *If* $M = 2^{2^r - r - 1}$, $r = 2, 3, ...,$ *then* $g(M, 1) = 2^r - 1$.

Algorithmic "proof": A number from 1 to M has been choosen. One lie is allowed.

1. Write the number choosen in binary. Encode the number using a binary Hamming code C with parameters

$$\begin{gathered} \text{length}=2^r - 1 \\ \text{dimension} = 2^r - r - 1 \\ \text{minimum distance}=3. \end{gathered}$$

   (There is a "good" algorithm for this.)

   Recall that each Hamming (n, k)-code is 1-error correcting.

2. Let $n = 2^r - 1$. Ask the n Questions: "Is the i^{th} digit of the encoded number a 1?", $i = 1, 2, ..., n$.

3. Let $w_i = 1$ if the i^{th} answer is "yes", let $w_i = 0$ otherwise, and write $\mathbf{w} = (w_1, w_2, ..., w_n)$.

4. Decode $\mathbf{w}$ to a Hamming codeword $\mathbf{c}$. (There is a "good" algorithm for this.)

5. Translate **c** into decimal.

This solves the searching with lies problem in the case where M is the above power of 2. The general problem has a similar solution, though the reasoning is a little more complicated. We refer to [Hi], [N][2], and [Mont], for more details.

3.13 Some unsolved problems in coding theory

To begin, a "code" in this section is any subset of $\mathbb{F}^n$, where $\mathbb{F}$ is a finite field. More precisely, a **code** is a map $e : \mathbb{F}^m \to \mathbb{F}^n$, where $\mathbb{F}^m$ represents the original message space and $\mathbb{F}^n$ the space of transmitted messages.

Question: Given $M > 1$ and $n > 1$, what is the largest d for which there is a code C of size M and minimum distance d?

At the moment, this is only known for $d = 1$ or if $d > 1$ and M is relatively small (depending on d).

To make the problem easier, let use restrict to the subclass of linear codes. In the case of linear codes, the question can be worded more precisely.

Question: Given $k > 1$ and $n > 1$, what is the largest d for which there is a linear code C of length n, dimension k, and minimum distance d?

Again, this isn't known, except in special cases.

Back to general codes.

Question: Given $d > 1$ and $n > 1$, what is the smallest $M > 1$ for which there is a code C of size M and minimum distance d?

This related problem is not known either (except for some very special cases, as above).

Again, to make the problem easier, let use restrict to the subclass of linear codes. In the case of linear codes, the question can be worded more precisely.

Question: Given $d > 1$ and $n > 1$, what is the smallest $k > 1$ for which there is a code C of length n, dimension k, and minimum distance d?

Again, in general this isn't known.

In another attempt to make these problems easier, we could ask for something less accurate but still useful. For example, in the last question, instead of fixing n we could allow $n \to \infty$ and simply ask for a sequence of $k = k_n$'s

[2]Some errors in this paper are corrected in [Mont].

which asymptotically approach the optimally best dimension. This isn't known either!

3.14 Coding theory exercises using GAP

GAP 4.2 and above has the GUAVA package which allows one to do some coding theory explorations.

3.14.1 Background on finite fields

The finite fields in GAP are of the form $\mathbb{F}_{p^k} = GF(p^k)$, where p is a prime power and $k \geq 1$ is an integer. Let's start with something simple: enter $\mathbb{F}_2$ and $\mathbb{F}_3$ and print out all their elements.

```
gap> GF2:=GF(2);
GF(2)
gap> gf2:=Elements(GF2);
[ 0*Z(2), Z(2)^0 ]
gap> GF3:=GF(3);
GF(3)
gap> gf3:=Elements(GF3);
[ 0*Z(3), Z(3)^0, Z(3) ]
```

What are $Z(2)$, $Z(3)$? In general, $\mathbb{F}_{p^k}^\times$ is a cyclic group. GAP uses this fact and lets $Z(p^k)$ denote a generator of this cyclic group. If $k = 1$ then giving a generator of $\mathbb{F}_p^\times = (\mathbb{Z}/p\mathbb{Z})^\times$ is equivalent to giving a primitive root mod p. In fact, $Z(p)$ is the smallest primitive root mod p, as is obtained from the `PrimitiveRootMod` function. Here's how to verify this for $p = 3, 5$:

```
gap> PrimitiveRootMod(3);
2
gap> 2*Z(3)^0=Z(3);
true
gap> PrimitiveRootMod(5)*Z(5)^0=Z(5);
true
```

Onto fields $\mathbb{F}_{p^k}$ with $k > 1$.

```
gap> GF4:=GF(4);
GF(2^2)
gap> gf4:=Elements(GF4);
[ 0*Z(2), Z(2)^0, Z(2^2), Z(2^2)^2 ]
gap> GF7:=GF(7);
GF(7)
gap> GF8:=GF(8);
GF(2^3)
gap> gf8:=Elements(GF8);
[ 0*Z(2), Z(2)^0, Z(2^3), Z(2^3)^2, Z(2^3)^3, Z(2^3)^4, Z(2^3)^5, Z(2^3)^6 ]
gap> GF9:=GF(9);
GF(3^2)
gap> gf9:=Elements(GF9);
[ 0*Z(3), Z(3)^0, Z(3), Z(3^2), Z(3^2)^2, Z(3^2)^3, Z(3^2)^5, Z(3^2)^6,
  Z(3^2)^7 ]
```

Note that $GF(8)$ contains $GF(2)$ but not $GF(4)$. It is a general fact that $GF(p^m)$ contains $GF(p^k)$ as a subfield if and only if $k|m$.

What are `Z(2^2)`, `Z(2^3)`, `Z(3^2)`, ...? They are less easy to explicitly explain. $Z(p^k)$ is a root of a certain irreducible polynomial mod p called a Conway polynomial. We can check this in the case $p^k = 8$ in GAP by plugging this supposed root into the polynomial and see if we get 0 or not:

```
gap> R:=PolynomialRing(GF2,["x"]);
<algebra-with-one over GF(2), with 1 generators>
gap> p:=ConwayPolynomial(2,3);
Z(2)^0+x+x^3
gap> Value(p,Z(8));
0*Z(2)
```

This tells us that the generator of $GF(8)$ is a root of the polynomial x^3+x+1 mod 2.

Next, let's try adding, subtracting, and multiplying field elements in GAP.

```
gap> gf9[4]; gf9[5]; gf9[4]+gf9[5];
Z(3^2)
Z(3^2)^2
Z(3^2)^3
```

```
gap> gf9[3]; gf9[6]; gf9[3]+gf9[6];
Z(3)
Z(3^2)^3
Z(3^2)^5
gap> gf9[3]; gf9[6]; gf9[3]*gf9[6];
Z(3)
Z(3^2)^3
Z(3^2)^7
gap> gf9[4]; gf9[6]; gf9[4]*gf9[6];
Z(3^2)
Z(3^2)^3
Z(3)
gap> gf8[4]; gf8[6]; gf8[4]*gf8[6];
Z(2^3)^2
Z(2^3)^4
Z(2^3)^6
gap> gf8[4]; gf8[6]; gf8[4]+gf8[6];
Z(2^3)^2
Z(2^3)^4
Z(2^3)
```

3.14.2 Some vector spaces

The GAP code

```
vecs:=[[1,0,0],[1,1,1],[0,1,1]];
V:=VectorSpace(Rationals,vecs);
GeneratorsOfVectorSpace(V);
B:=Basis(V);
dim:=Length(B);
```

constructs the vector space over $\mathbb{Q}$ spanned by the vectors $(1,0,0), (1,1,1), (0,1,1)$, finds a basis, and computes its dimension.

Exercise 3.14.1. *Construct the vector space over $\mathbb{Q}$ spanned by the vectors $(1,0,0), (1,1,1), (0,-1,1)$, find a basis, and compute its dimension. Do the same, but with $\mathbb{Q}$ replaced by $GF(3)$.*

3.14.3 Some simple codes

Now we start to investigate codes in GAP. This requires the package GUAVA. To load GUAVA, type `RequirePackage(‘‘guava’’);`. GAP will then display the GUAVA banner:

```
   ____                              |
  /            \           /   --+--  Version 1.5
 /      |    | |\\        //|    |
|    _  |    | | \\      // |     Jasper Cramwinckel
|     \ |    | |--\\    //--|     Erik Roijackers
 \     ||    | |   \\  //   |     Reinald Baart
  \___/  \___/ |    \\//    |     Eric Minkes
                                   Lea Ruscio
true
```

How do you enter a code it using only the list of codes words? Easy, here's an example:

```
gap> C:=ElementsCode(["0000","1111"],"repetition code",GF(2));
a (4,2,1..4)2 repetition code over GF(2)
```

This notation `(4,2,1..4)2` is GUAVA shorthand for: the length is 4, the size is 2, the minimum distance in the Hamming metric is between 1 and 4, and the covering radius in the Hamming metric is 2.

Here's how to check all this is correct:

```
gap> Elements(C);
[ [ 0 0 0 0 ], [ 1 1 1 1 ] ]
gap> MinimumDistance(C);
4
gap> Dimension(C);
1
```

GUAVA didn't know the minimum distance before, but once you type the `MinimumDistance(C);` command it modifies the GAP record for C.

```
gap> C;
a cyclic [4,1,4]2 repetition code over GF(2)
```

Let us not enter a code using a generator matrix G:

```
gap> G := Z(2)*[ [1,0,0,1,1,0], [0,1,0,1,1,0], [0,0,1,0,1,1] ];
[ [ Z(2)^0, 0*Z(2), 0*Z(2), Z(2)^0, Z(2)^0, 0*Z(2) ],
  [ 0*Z(2), Z(2)^0, 0*Z(2), Z(2)^0, Z(2)^0, 0*Z(2) ],
  [ 0*Z(2), 0*Z(2), Z(2)^0, 0*Z(2), Z(2)^0, Z(2)^0 ] ]
gap> C:=GeneratorMatCode(G,GF(2));
a linear [6,3,1..3]2..3 code defined by generator matrix over GF(2)
gap> MinimumDistance(C);
2
gap> IsLinearCode(C);
true
gap> Elements(C);
[ [ 0 0 0 0 0 0 ], [ 0 0 1 0 1 1 ], [ 0 1 0 1 1 0 ], [ 0 1 1 1 0 1 ],
  [ 1 0 0 1 1 0 ], [ 1 0 1 1 0 1 ], [ 1 1 0 0 0 0 ], [ 1 1 1 0 1 1 ] ]
gap> Dimension(C);
3
```

This is not 1 error correcting (try correcting `[ 1 1 0 1 1 0 ]`).

Here's another example,

```
gap> G := Z(2)*[ [1,0,0,1,1,0], [0,1,0,1,0,1], [0,0,1,0,1,1] ];
[ [ Z(2)^0, 0*Z(2), 0*Z(2), Z(2)^0, Z(2)^0, 0*Z(2) ],
  [ 0*Z(2), Z(2)^0, 0*Z(2), Z(2)^0, 0*Z(2), Z(2)^0 ],
  [ 0*Z(2), 0*Z(2), Z(2)^0, 0*Z(2), Z(2)^0, Z(2)^0 ] ]
gap> C:=GeneratorMatCode(G,GF(2));
a linear [6,3,1..3]2 code defined by generator matrix over GF(2)
gap> MinimumDistance(C);
3
gap> Dimension(C);
3
gap> Elements(C);
[ [ 0 0 0 0 0 0 ], [ 0 0 1 0 1 1 ], [ 0 1 0 1 0 1 ], [ 0 1 1 1 1 0 ],
  [ 1 0 0 1 1 0 ], [ 1 0 1 1 0 1 ], [ 1 1 0 0 1 1 ], [ 1 1 1 0 0 0 ] ]
```

This is 1 error correcting.

3.14.4 Hamming codes

The $[7, 4, 3]$-Hamming code over $GF(2)$ having generator matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

is obtained by typing

```
C:=HammingCode(3,GF(2));
G:=GeneratorMat(C);
Display(G);
```

Encoding a message w using G, is simply the map $w \longmapsto wG$. Type

```
Elements(C);
Size(Elements(C));
```

From this, you see all the codewords of C and how many there are.

To get the parity check matrix, type

`H:=CheckMat(C); Display(H);`

Note all the columns of H are distinct and non-zero. To see if a vector in $\mathbb{F}^7$ is a codeword, simply compute Hv and check if it is zero or not. Here's a GAP example:

```
v := Codeword([0,0,1,1,0,1,0]);
v in C;
```

(Here

`v = [ 1 0 1 1 0 1 0 ]+[1,0,0,0,0,0,0].`

which is the code word obtained by encoding $(1, 0, 1, 0)$ plus the error vector $(1, 0, 0, 0, 0, 0, 0)$.
Since this last vector is non-zero, v is not a codeword. If it was a vector received in transmission (with at least one error) then to decode it, hence to find the most likely codeword sent, type

```
Decode(C,v);
```

Exercise 3.14.2. *(a) For the parity check matrix H of the binary Hamming code of length $2^3-1=7$, verify $Hc=0$ for three or four codewords c. Decode* $(1,1,0,0,0,0,0)$.

(b) Find a parity check matrix of the 3-ary Hamming code of length $(3^3-1)/(3-1)=13$. *Verify $Hc=0$ for three or four codewords c. Decode* $(1,2,1,2,1,2,1,2,1,2,1,2,1)$.

To get the dimension of the code, type `Dimension(C);` To get its minimum distance, type `MinimumDistance(C);`

Exercise 3.14.3. *Find the dimension and minimum distance of*
(a) the binary Hamming code of length 15,
(b) the 3-ary Hamming code of length 13.

3.14.5 Reed-Muller codes

Reed-Muller shall be abbreviated RM.

The $[8,4,4]$ RM code over $GF(2)$ having generator matrix

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

is obtained by typing

```
C:=ReedMullerCode(1,3);
G:=GeneratorMat(C);
```

(Use `Display(G);` to see `G` if necessary.) Encoding a message w using G, is simply the map $w \longmapsto wG$. Type

```
Elements(C);
Size(Elements(C));
```

From this, you see all the codewords of C and how many there are.

To get the parity check matrix, type

`H:=CheckMat(C);`

To see if a vector in $\mathbb{F}^8$ is a codeword, simply compute Hv and check if it is zero or not. Here's a GAP example:

```
v:=Codeword([1,0,0,0,0,0,0,0]);
v in C;
```

Since this last vector is non-zero, v is not a codeword. If it was a vector received in transmission (with at least one error) then to decode it, hence to find the most likely codeword sent, type

```
Decode(C,v);
```

Exercise 3.14.4. *(a) For the parity check matrix H of the RM code of length* 8*, verify $Hc = 0$ for three or four codewords c. Decode* $(1, 1, 0, 0, 0, 0, 0, 0)$.

(b) Find a parity check matrix of the RM code of length 16*. Verify $Hc = 0$ for three or four codewords c. Decode* $(1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0)$.

To get the dimension of the code, type `Dimension(C);` To get its minimum distance, type `MinimumDistance(C);`

Exercise 3.14.5. *Find the dimension and minimum distance of the RM code of length* 16.

3.14.6 Cyclic codes

A **cyclic code** is associated to an irreducible polynomial over $\mathbb{F}$ which divides $x^n - 1$.

For example, to get the $[7, 4, 3]$ cyclic code over $GF(2)$ having generator matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

```
R:=PolynomialRing(GF(2),["x"]);;
x:=Indeterminate(GF(2),"x");;
g:=x^3+x+1;
Factors(x^7-1);
C:=GeneratorPolCode(g,7,GF(2));
G:=GeneratorMat(C);
```

To get the parity check matrix, type `H:=CheckMat(C);` Try `IsCyclicCode(C);`

To get the dimension of the code, type `Dimension(C);` To get its minimum distance, type `MinimumDistance(C);`

Exercise 3.14.6. *Check if x^2+1 is a generating polynomial of a cyclic code of length 4 over $GF(2)$. If so, find a basis for this code, as a vector space over $GF(2)$, and construct a parity check matrix.*

3.15 Coding theory exercises using MAGMA, I

Let $\mathbb{F}$ be a finite field. Consider a short exact sequence of vector spaces

$$0 \to \mathbb{F}^k \xrightarrow{\gamma} \mathbb{F}^n \xrightarrow{\theta} \mathbb{F}^{n-k} \to 0.$$

A **linear code** is the image of γ. Since the sequence is exact, a vector $v \in \mathbb{F}^n$ is a codeword if and only if $\theta(v) = 0$. If $\mathbb{F}^i$ is given the usual standard vector space basis then the matrix of γ is the generating matrix and the matrix of θ is the parity check matrix.

3.15.1 Some vector spaces

The MAGMA code

```
V := VectorSpace(Rationals(),3);
W:=sub<V|[1,0,0],[1,1,1],[0,1,1]>;
W;
Basis(W);
Dimension(W);
```

constructs the vector space over $\mathbb{Q}$ spanned by the vectors $(1,0,0), (1,1,1), (0,1,1)$, finds a basis, and computes its dimension.

Exercise 3.15.1. *Construct the vector space over $\mathbb{Q}$ spanned by the vectors $(1,0,0), (1,1,1), (0,-1,1)$, find a basis, and compute its dimension. Do the same, but with $\mathbb{Q}$ replaced by $GF(3)$.*

3.15.2 Hamming codes

The $[7,4,3]$-Hamming code over $GF(2)$ having generator matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

is obtained by typing

```
C:=HammingCode(GF(2),3);
C;
G:=GeneratorMatrix(C);
G;
```

Encoding a message w using G, is simply the map $w \longmapsto wG$. Type

```
W:=InformationSpace(C);
Cwords:=<w*G : w in W>;
Cwords;
#Cwords;
```

From this, you see all the codewords of C and how many there are.

To get the parity check matrix, type `H:=ParityCheckMatrix(C); H;` To see if a vector in $\mathbb{F}^7$ is a codeword, simply compute Hv and check if it is zero or not. Here's a MAGMA example [3] :

```
V:=AmbientSpace(C);
v:=V![1,0,0,0,0,0,0];
v*Transpose(H);
```

Since this last vector is non-zero, v is not a codeword. If it was a vector received in transmission (with at least one error) then to decode it, hence to find the most likely codeword sent, type

```
Decode(C,v);
```

Exercise 3.15.2. *(a) For the parity check matrix H of the binary Hamming code of length $2^3-1=7$, verify $Hc=0$ for three or four codewords c. Decode* $(1,1,0,0,0,0,0)$.

(b) Find a parity check matrix of the 3-ary Hamming code of length $(3^3-1)/(3-1)=13$. *Verify $Hc=0$ for three or four codewords c. Decode* $(1,2,1,2,1,2,1,2,1,2,1,2,1)$.

To get the dimension of the code, type `Dimension(C);` To get its minimum distance, type `MinimumDistance(C);`

Exercise 3.15.3. *Find the dimension and minimum distance of*

(a) the binary Hamming code of length 15,

(b) the 3-ary Hamming code of length 13.

[3] A minor point: here Hv must be typed in as $v*H^t$ since MAGMA (in the Australian tradition?) has matrices acting on the right.

3.15.3 Reed-Muller codes

Reed-Muller shall be abbreviated RM.

The $[8, 4, 4]$ RM code over $GF(2)$ having generator matrix

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

is obtained by typing

```
C:=ReedMullerCode(1,3);
C:
G:=GeneratorMatrix(C);
G;
```

Encoding a message w using G, is simply the map $w \longmapsto wG$. Type

```
W:=InformationSpace(C);
W;
Cwords:=<w*G : w in W>;
Cwords;
#Cwords;
```

From this, you see all the codewords of C and how many there are.

To get the parity check matrix, type `H:=ParityCheckMatrix(C);` To see if a vector in $\mathbb{F}^8$ is a codeword, simply compute Hv and check if it is zero or not. Here's a MAGMA example:

```
V:=AmbientSpace(C);
v:=V![1,0,0,0,0,0,0,0];
v*Transpose(H);
```

Since this last vector is non-zero, v is not a codeword. If it was a vector received in transmission (with at least one error) then to decode it, hence to find the most likely codeword sent, type

```
Decode(C,v);
```

Exercise 3.15.4. *(a) For the parity check matrix H of the RM code of length* 8*, verify $Hc = 0$ for three or four codewords c. Decode* $(1, 1, 0, 0, 0, 0, 0, 0)$.

(b) Find a parity check matrix of the RM code of length 16*. Verify $Hc = 0$ for three or four codewords c. Decode* $(1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0)$.

To get the dimension of the code, type `Dimension(C);` To get its minimum distance, type `MinimumDistance(C);`

Exercise 3.15.5. *Find the dimension and minimum distance of the RM code of length* 16.

3.15.4 Cyclic codes

A **cyclic code** is associated to an irreducible polynomial over $\mathbb{F}$ which divides $x^n - 1$.

For example, to get the $[7, 4, 3]$ cyclic code over $GF(2)$ having generator matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

```
R<x>:=PolynomialRing(GF(2));
g:=x^3+x+1;
Factorization(x^7-1);
C:=CyclicCode(7,g);
C;
G:=GeneratorMatrix(C);
G;
```

To get the parity check matrix, type `H:=ParityCheckMatrix(C);` Try `IsCyclic(C);`

To get the dimension of the code, type `Dimension(C);` To get its minimum distance, type `MinimumDistance(C);`

Exercise 3.15.6. *Check if $x^2 + 1$ is a generating polynomial of a cyclic code of length* 4 *over $GF(2)$. If so, find a basis for this code, as a vector space over $GF(2)$, and construct a parity check matrix.*

Chapter 4

Permutations

Though the main purpose of this section is to provide background necessary for the next chapter on groups, permutation have been studied much longer than groups have. One of their original motivations, in the 1700's by French mathematicians such as Lagrange and Galois, were as a tool to understand how the roots of a polynomials depended on the coefficients.

4.1 Functions

The type of function we will run across here most frequently is a "permutation", defined precisely later, which is roughly speaking a rule which mixes up and swaps around the elements of a finite set.

Let S and T be sets.

Definition 4.1.1. *: A* **function** *f from S to T is a rule which associates to each element $s \in S$ exactly one element $t \in T$. We will use the following notation and terminology for this:*

$$\begin{aligned} f : S \to T \quad & (\text{"f is a function from S to T"}), \\ f : s \longmapsto t \quad & (\text{"f sends s in S to t in T"}), \\ t = f(s) \quad & (\text{"t is the image of s under f"}). \end{aligned}$$

A function is also called a map, mapping, or transformation. We call S the **domain** of f, T the **codomain** (or **range**) of f, and the set

$$f(S) = \{f(s) \in T \mid s \in S\}$$

the **image** of f.

A wonderful tool for learning about permutations is the Rubik's cube. Erno Rubik was born during World War II in Budapest, Hungary. Far from being a mathematician, Rubik was a Professor at the Academy of Applied Arts and Design, teaching interior design. In the mid 1970's, he patented a cube-shaped mechanical puzzle that has since captured the imagination of millions of people worldwide, the Rubik's Cube. Those of you who have a Rubik's cube or similar puzzle have met permutations before. The moves are permutations!

Example 4.1.2.

- *One typical example for us will be when S is the set of all 54 facets of the Rubik's cube and f is a rule for associating to each facet some other facet determined by a Rubik's cube move. This example will be discussed in more detail later (see §4.8.2 below).*

- *If $n > 0$ is any natural number, recall from §1.4.5 that we let $\phi(n)$ denote the number of natural numbers less than or equal to n which have no prime factor in common with n. The function $\phi : \mathbb{N} \to \mathbb{N}$ is Euler's phi function. We have $\phi(1) = 1$, $\phi(2) = 1$, $\phi(3) = 2$, ...*

- *If n is any natural number, let $\pi(n)$ denote the number of prime numbers less than or equal to n. The function $\pi : \mathbb{N} \to \mathbb{N}$ is simply called the π* **function**. *We have $\pi(1) = 0$, $\pi(2) = 1$, $\pi(3) = 2$, ... Though the rough asymptotic behaviour of $\pi(n)$ is known, there are still many unsolved problems regarding $\pi(n)$. For example, the following problem is still unsolved.*

  Question: *Is there $n > 1$ such that $\pi(n^2 + 2n + 1) = \pi(n^2)$ (i.e., ii there always a prime between any two consecutive squares)?*

- *Let $f : \mathbb{Z} \to \mathbb{Z}/m\mathbb{Z}$ denote the map which sends an integer n to its residue $\overline{n} = n \pmod{m}$. This "mod m function" maps an infinite set to a finite set.*

- *Let $a_1, a_2, ...$ denote any sequence of complex numbers. We may regard this sequence as a function $f : \mathbb{N} \to \mathbb{C}$ by $f(n) = a_n$, $n > 0$.*

The **Cartesian product** of two sets S, T is the set of pairs of elements taken from these sets:

$$S \times T = \{(s, t) \mid s \in S, t \in T\}.$$

An element of $S \times T$ is simply a list of two things, the first one from S and the second one from T. This construction of a new set from two given sets is named for the French philosopher René Descartes (March 1596-February 1650) whose work, **La géométrie**, includes his application of algebra to geometry.

Example 4.1.3. *If $\mathbb{R}$ denotes the set of all real numbers then the Cartesian product $\mathbb{R} \times \mathbb{R}$ is simply the set of all pairs or real numbers. In other words, this is the Cartesian plane we are all familiar with from high school.*

More generally, we may repeat this process and take the Cartesian product of the set of real numbers with itself n times (where $n > 1$ is any integer) to get $\mathbb{R} \times ... \times \mathbb{R}$ (n times). This n-fold Cartesian product is denoted more conveniently by $\mathbb{R}^n$. An element of the set $\mathbb{R}^n$ is simply a list of n real numbers. Such a list will be called a **vector** *or an n-***vector** *to be specific.*

Sometimes it is convenient to "picture" a function as the set of its values inside the Cartesian product $S \times T$. The **graph** of a function $f : S \to T$ is the subset

$$\{(s, f(s)) \mid s \in S\} \subset S \times T.$$

Not every subset of $S \times T$ is the graph of some function from S to T.

Definition 4.1.4. *Let $f : S \to T$ and $g : T \to U$ be two functions. We can compose them to get another function, the* **composition**, *denoted $fg : S \to U$:*

$$\begin{array}{ccccc} x & \longmapsto & f(x) & \longmapsto & g(f(x)) \\ S & \to & T & \to & U \end{array}$$

If this definition seems "backwards" to you, note we are not writing $f \circ g(x)$ but $f(g(x))$.

Definition 4.1.5. *If the image of the function $f : S \to T$ is all of T, i.e., if $f(S) = T$, then we call f* **surjective** *(or "onto", or say "f is a surjection"). Equivalently, a function f from S to T is surjective if every $t \in T$ is the image of some $s \in S$ under f.*

For example, the map $f : \mathbb{R} \to \mathbb{R}$ defined by $f(x) = 2x$, for any real number x, is surjective. Another example, let S be the set of all 54 facets of the Rubik's cube. Let $f : S \to S$ be the map which sends a facet to the facet which is diametrically opposite (for instance, the upward center facet would be mapped to the downward center facet). This function is also surjective.

Definition 4.1.6. *: A function $f : S \to T$ is called* **injective** *(or "one-to-one" or say "f is an injection") if each element t belonging to the image $f(S)$ is the image of exactly one s in S.*

In other words, f is an injection if the condition $f(s_1) = f(s_2)$ (for some $s_1, s_2 \in S$) always forces $s_1 = s_2$.

Question: Suppose that $|S| > |T|$. Is there an injective function $f : S \to T$? Explain.

Definition 4.1.7. *A function $f : S \to T$ is called a* **bijection** *if it is both injective and surjective.*

Equivalently, a bijection from S to T is a function $f : S \to T$ for which each $t \in T$ is the image of exactly one $s \in S$.

Definition 4.1.8. *A set S is called* **countable** *if there exists a bijection $f : S \to \mathbb{Z}$ to the set of integers $\mathbb{Z}$.*

Example 4.1.9. *(a) The set of all odd integers $S = \{..., -3, -1, 1, 3, ...\}$ is countable since the map $f : S \to \mathbb{Z}$ defined by $f(x) = (x+1)/2$ is a bijection.*

(b) Recall the set of all rational numbers is denoted by $\mathbb{Q}$. The set S of all rational numbers within the unit interval $0 < r < 1$ is countable since you can define $f : S \to \mathbb{Z}$ as follows: $f(1) = 1/2$, $f(2) = 1/3$, $f(3) = 2/3$, $f(4) = 1/4$, $f(5) = 3/4$, $f(6) = 1/5$, $f(7) = 2/5$, $f(8) = 3/5$, and so on. (There are $\phi(n)$ terms of the form m/n, where m is relatively prime to n and $\phi(n)$ denotes the Euler ϕ-function.).

Example 4.1.10. *Let C be the cube in 3-space having vertices at the points $O = (0,0,0), P_1 = (1,0,0), P_2 = (0,1,0), P_3 = (0,0,1), P_4 = (1,1,0), P_5 = (1,0,1), P_6 = (0,1,1), P_7 = (1,1,1)$. We shall also (to use a notation which will be used more later) denote these by dbl, dfl, dbr, ubl, dfr, ufl, ubr, ufr, resp. Let $C_0 = \{O, P_1, ..., P_7\}$ be the set of the 8 vertices of C, let $C_1 = \{uf, ur, ub, ul, fr, br, bl, fl, df, dr, db, dl\}$ be the set of the 12 edges of C, and let $C_2 = \{F, B, U, D, L, R\}$ be the set of the 6 faces of C. Let r be the rotation of a point (x, y, z) by 180 degrees about the line passing through the points $(1/2, 1/2, 0)$, $(1/2, 1/2, 1)$. Note that $r : \mathbb{R}^3 \to \mathbb{R}^3$ is a function which sends the cube C onto itself.*

This function r induces three functions

$$f_0 : C_0 \to C_0, \qquad f_1 : C_1 \to C_1, \qquad f_2 : C_2 \to C_2.$$

where f_i is the function which sends $x \in C_i$ to its image $r(x)$ under r (which is again in C_i), for $i = 0, 1, 2$. Each f_i is a bijection. For example, $f_1(ub) = uf$ and $f_2(B) = B$.

Suppose $f : S \to T$ is a bijection. Define f^{-1} to be the rule which associates to $t \in T$ the element $s \in S$,

$$f^{-1}(t) = s \qquad \text{if and only if } f(s) = t.$$

This rule defines a function $f^{-1} : T \to S$. This function satisfies $f(f^{-1}(t)) = t$, for all $t \in T$, and $f^{-1}(f(s))) = s$, for all $s \in S$. In other words, the composition $f \circ f^{-1} : T \to T$ sends each element of T to itself, and $f^{-1} \circ f : S \to S$ sends each element of S to itself.

Definition 4.1.11. *The function f^{-1} constructed above is called the* **inverse function** *of f.*

Exercise 4.1.12. *Label the vertices and the facets as in Example 4.1.10 and describe f_1 and f_2 explicitly by filling out the tables*

v	$f_0(v)$	e	$f_1(e)$	f	$f_2(f)$

Exercise 4.1.13. *Suppose that $|S| < |T|$. Is there a surjective function $f : S \to T$? Explain.*

Exercise 4.1.14. *Suppose that $|S| = |T|$. Show that a function $f : S \to T$ is surjective if and only if it is injective.*

Exercise 4.1.15. *If $f : W \to X$, $g : V \to W$, $h : U \to V$ are functions, is $(fg)h = f(gh)$? In other words, if $f(w) = x$, $g(v) = w$, $h(u) = v$, is the value of $(fg)h$ at u equal to the value of $f(gh)$ at u?*

4.2 Permutations

Let $\mathbb{Z}_n = \{1, 2, ..., n\}$ be the set of integers from 1 to a fixed positive integer n. A **permutation** of $\mathbb{Z}_n$ is a bijection from T to itself. For example, on the 3×3 Rubik's cube there are $9 \cdot 6 = 54$ facets. If you label them $1, 2, ..., 54$ (in any way you like) then any move of the Rubik's cube corresponds to a permutation of $\mathbb{Z}_{54}$. In this section we present some basic notation and properties of permutations.

Notation: We may denote a permutation $f : \mathbb{Z}_n \to \mathbb{Z}_n$ by a $2 \times n$ array:

$$f \leftrightarrow \begin{pmatrix} 1 & 2 & ... & n \\ f_1 & f_2 & ... & f_n \end{pmatrix},$$

where $f_1, f_2, ..., f_n$ is simply a linear rearrangement of the integers $1, 2, ..., n$. This notation means that f sends 1 to f_1, f sends 2 to f_2, ..., f sends n to f_n. In other words, $f_1 = f(1)$, $f_2 = f(2)$, ..., $f_n = f(n)$. In this way, we see that there is a 1-1 correspondence between the linear rearrangements of the integers $1, 2, ..., n$ and permutations. (In fact, many books define a permutation to be a linear rearrangement and then prove that it gives rise to a function as above.)

Example 4.2.1. *(a) The* **identity** *permutation, denoted by I, is the permutation which doesn't do anything:*

$$I = \begin{pmatrix} 1 & 2 & ... & n \\ 1 & 2 & ... & n \end{pmatrix}$$

(b) The permutation $f : \mathbb{Z}_3 \to \mathbb{Z}_3$ defined by

$$f = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$$

simply swaps 1 *and* 3 *and leaves* 2 *alone.*

(a) The n**-cycle** *is a permutation which cyclically permutes the values:*

$$\begin{pmatrix} 1 & 2 & ... & n \\ 2 & 3 & ... & 1 \end{pmatrix}$$

Imagine an analog clock with the numbers 1, 2, ..., n *arranged around the dial. An n-cycle simply moves each number forward (clockwise) by one unit. The permutation*

$$\begin{pmatrix} 1 & 2 & ... & n \\ n & 1 & ... & n-1 \end{pmatrix}$$

is also called an n-**cycle**.

Definition 4.2.2. *Let* $f : \mathbb{Z}_n \to \mathbb{Z}_n$ *be a permutation and let*

$$e_f(i) = \#\{j > i \mid f(i) > f(j)\}, \qquad 1 \leq i \leq n-1.$$

Let

$$swap(f) = e_f(1) + ... + e_f(n-1).$$

We call this the **swapping number** *(or* **length***) of the permutation* f *since it counts the number of times* f *swaps the inequality in* $i < j$ *to* $f(i) > f(j)$. *If we plot a bar-graph of the function* f *then* $swap(f)$ *counts the number of times the bar at* i *is higher than the bar at* j. *We call* f **even** *if* $swap(f)$ *is even and we call* f **odd** *otherwise.*

The number

$$sign(f) = (-1)^{swap(f)}$$

is called the **sign** *(or* **signum** *function) of the permutation* f.

The reader may verify that the sign function satisfies the following property.

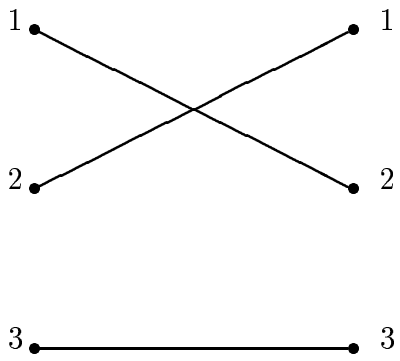
Lemma 4.2.3. *Let* $f : \mathbb{Z}_n \to \mathbb{Z}_n$ *be a permutation. Then*

$$sign(f) = \prod_{i<j\leq n} \frac{f(i) - f(j)}{i - j}.$$

Example 4.2.4. *We define the permutation* $f : \mathbb{Z}_3 \to \mathbb{Z}_3$ *for which* $f(1) = 2, f(2) = 1, f(3) = 3$ *by a* 2×3 *array*

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$$

This swaps 1 *and* 2 *and leaves* 3 *alone. There are 3 inequalities for* $\mathbb{Z}_3$*:* $1 < 2$, $2 < 3$, *and* $1 < 3$. *Applying* f *to these, we get:* $f(1) = 2 > f(2) = 1$, $f(2) = 1 < f(3) = 3$, *and* $f(1) = 2 < f(3) = 3$. *Only the first inequality is changed, so the swapping number is* $swap(f) = 1$. *Another way to picture* f *is by a "crossing diagram", where* f *sends the left-hand column to the right-hand column:*



The number of crosses in this diagram is the swapping number of f, from which we can see that f is an odd permutation.

Definition 4.2.5. *Let $f : \mathbb{Z}_n \to \mathbb{Z}_n$ and $g : \mathbb{Z}_n \to \mathbb{Z}_n$ be two permutations. We can compose them to get another permutation, the* **composition**, *denoted $f \circ g : \mathbb{Z}_n \to \mathbb{Z}_n$ (or sometimes simply fg):*

$$\begin{array}{ccccc} k & \longmapsto & f(k) & \longmapsto & g(f(k)) \\ \mathbb{Z}_n & \to & \mathbb{Z}_n & \to & \mathbb{Z}_n \end{array}$$

Notation We shall follow standard convention and write our compositions of permutations **left-to-right**. (This is of course in contrast to the *right-to-left* composition of functions you may have seen in calculus.) When a possible ambiguity may arise, we call this type of composition "composition as permutations" and call "right-to-left composition" the "composition as functions".

When $f = g$ then we write ff as f^2. In general, we write the n-fold composition $f...f$ (n times) as f^n. Every permutation f has the property that there is some integer $N > 0$, which depends on f, such that $f^N = I$. (In other words, if you repeatedly apply a permutation enough times you will eventually obtain the identity permutation.)

Lemma 4.2.6. *Let f and g be permutations $\mathbb{Z}_n \to \mathbb{Z}_n$. $sign(fg) = sign(f)sign(g)$.*

Definition 4.2.7. *The smallest integer $N > 0$ such that $f^N = I$ is called the* **order** *of f.*

Example 4.2.8. *Let*

$$f = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix} \qquad g = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$$

be permutations of $\mathbb{Z}_n$. *We have*

$$fg = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}, \qquad f^2 = I, \qquad g^3 = I.$$

Exercise 4.2.9. *Compute (a)* fg *and (b) the order of* f *and the order of* g, *where*

$$(a) \qquad f = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix} \qquad g = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$$

$$(b) \qquad f = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix} \qquad g = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$$

Exercise 4.2.10. *Compute (a)* fg *and (b) the order of* f *and the order of* g, *where*

$$(a) \qquad f = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 4 & 1 \end{pmatrix} \qquad g = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 2 & 1 \end{pmatrix}$$

$$(b) \qquad f = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 \end{pmatrix} \qquad g = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 3 & 2 \end{pmatrix}$$

Exercise 4.2.11. *For* f, g *as in Lemma 4.2.6, we have*

$$\prod_{i<j\leq n} \frac{fg(i) - fg(j)}{i-j} = \prod_{i<j\leq n} \frac{g(i) - g(j)}{i-j} \prod_{i<j\leq n} \frac{fg(i) - fg(j)}{g(i) - g(j)}.$$

Why does this imply the lemma?

Exercise 4.2.12. *Express* $f : \mathbb{Z}_3 \to \mathbb{Z}_3$ *given by* $f(1) = 3, f(2) = 1, f(3) = 2$, *as (a) a* 2×3 *array, (b) a crossing diagram. Find its swapping number and sign.*

Exercise 4.2.13. *Express* $f : \mathbb{Z}_4 \to \mathbb{Z}_4$ *given by* $f(1) = 3, f(2) = 4, f(3) = 2, f(4) = 1$, *as (a) a* 2×4 *array, (b) a crossing diagram. Find its swapping number and sign.*

4.3 Inverses

Try to visualize in your mind the graph of a function $f : \mathbb{Z}_n \to \mathbb{Z}_n$. We can think of this as either a point of points $(i, f(i))$, $i = 1, 2, ..., n$, or as a bar graph. In any case, we can look at this graph of f and determine

(a) if f is injective,

(b) if f is surjective,

(c) the inverse f^{-1}, if it exists.

How? Well, from the graph of f we can determine the image $f(\mathbb{Z}_n)$ and this determines if f is surjective or not. The inverse exists only if f is injective (hence, in our case, surjective). Its graph is determined by reflecting the graph of f about the diagonal, $x = y$.

Lemma 4.3.1. *The following statements are equivalent:*

(1) $f : \mathbb{Z}_n \to \mathbb{Z}_n$ *is injective,*

(2) $f : \mathbb{Z}_n \to \mathbb{Z}_n$ *is surjective,*

(3) $|f(\mathbb{Z}_n)| = |\mathbb{Z}_n|$.

proof: The equivalence of the first two statements is left to the interested reader (hints: first show (1) imples (2) using reductio ad absurdum, then show (2) imples (1), again using reductio ad absurdum). (2) is equivalent to (3), by definition of surjectivity. □

Example 4.3.2. *The inverse of*

$$\begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$$

is

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$$

There are two more commonly used ways of expressing a permutation. The first is the "matrix notation":

Definition 4.3.3. *To a permutation* $f : \mathbb{Z}_n \to \mathbb{Z}_n$*, given by*

$$f = \begin{pmatrix} 1 & 2 & ... & n \\ f(1) & f(2) & ... & f(n) \end{pmatrix}$$

we associate to it the matrix $P(f)$ *of* $0's$ *and* $1's$ *defined as follows: the* ij*-th entry of* $P(f)$ *is* 1 *if* $j = f(i)$ *and is 0 otherwise.*

Definition 4.3.4. *A square matrix which has exactly one 1 per row and per column (as $P(f)$ does) is called a* **permutation matrix** .

Lemma 4.3.5. *There are $n!$ distinct $n \times n$ permutation matrices and there are $n!$ distinct permutations of the set $\{1, 2, ..., n\}$.*

Example 4.3.6. *The matrix of the permutation f given by*

$$f = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$$

is

$$P(f) = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Note that matrix multiplication gives

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix}$$

from which we can recover the 2×3 array.

(b) If

$$h = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 1 & 4 \end{pmatrix},$$

then

$$P(h) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Theorem 4.3.7. *If $f : \mathbb{Z}_n \to \mathbb{Z}_n$ is a permutation then*

$$(a) \qquad P(f) \begin{pmatrix} 1 \\ 2 \\ \vdots \\ n \end{pmatrix} = \begin{pmatrix} f(1) \\ f(2) \\ \vdots \\ f(n) \end{pmatrix}$$

Furthermore, the inverse of the matrix of the permutation is the matrix of the inverse of the permutation:

(b) $P(f)^{-1} = P(f^{-1})$,

and the matrix of the product is the product of the matrices:

(c) $P(fg) = P(f)P(g)$.

proof: If $\vec{v}$ is the column vector with entries $v_1, v_2, ..., v_n$ (the v_i are arbitrary real numbers) then $P(f)\vec{v}$ is the column vector whose i^{th} coordinate is equal to v_j if f sends i to j, by definition of $P(f)$. Since, in this case, $j = f(i)$ (here we write $f(i)$ to denote the image of i under the permutation f, even though i really gets plugged into f on the left), this implies that $P(f)\vec{v}$ is the column vector with entries $v_{f(1)}, v_{f(2)}, ..., v_{f(n)}$. This proves (a).

Note (b) is a consequence of (c) so we need only prove (c). We compute $P(fg)\vec{v}$ and $P(f)P(g)\vec{v}$. By the same reasoning as in (a), we find that the i^{th} coordinate of $P(fg)\vec{v}$ is $v_{(fg)(i)}$. Similarly, the i^{th} coordinate of $P(g)\vec{v}$ is $v'_i = v_{g(i)}$. Therefore, the i^{th} coordinate of $P(f)(P(g)\vec{v})$ is $v'_{f(i)} = v_{g(f(i))} = v_{(fg)(i)}$. This implies $P(fg)\vec{v} = P(f)P(g)\vec{v}$. Since the v_i were arbitrary real numbers, this implies the theorem. □

The matrix can be determined from the graph of the function $f : \mathbb{Z}_n \to \mathbb{Z}_n$ as follows: with x and y integers between 1 and n inclusive, let $P(f)_{ij} = 1$ if and only if (i, j) is a plotted point in the graph of f, and let $P(f)_{ij} = 0$, otherwise. The resulting $n \times n$ array is the matrix $P(f)$.

Exercise 4.3.8. *Let*

$$f = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}, \qquad g = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}, \qquad h = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix},$$

(a) Show $f = f^{-1}$, $g = g^{-1}$, $h = fg$.

(b) Show,

$$P(g) = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \qquad P(h) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

(c) Show, using a direct matrix calculation, that $P(f)P(g) = P(fg) = P(h)$ and $P(h^{-1}) = P(g^{-1}f^{-1}) = P(g^{-1})P(f^{-1}) = P(g)P(f)$.

Exercise 4.3.9. *Prove Lemma 4.3.5 using the multiplication counting principle from the section on counting in the previous chapter.*

Exercise 4.3.10. *Graph and determine the inverses of the following permutations:*

$$(a) \qquad f = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$$

$$(b) \qquad f = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}$$

$$(c) \qquad f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 5 & 3 & 4 \end{pmatrix}$$

4.4 Cycle notation

Rubik's cubers will often, without knowing it perhaps, use the following lemma to solve their cube.

Lemma 4.4.1. *Let $r \in S_n$ denote any permutation and let i, j denote distinct integers belonging to $\{1, 2, ..., n\}$. Let s denote a permutation sending i to j:*

$$s(i) = j.$$

Then $s^r = r^{-1}sr$ is a permutation sending $r(i)$ to $r(j)$:

$$s^r(r(i)) = r(j).$$

More specifically (and this is the case which this lemma is most often applied in this book): let $i_1, i_2, ..., i_k$ denote distinct integers belonging to $\{1, 2, ..., n\}$. Let s denote the permutation sending i_j to i_{j+1}:

$$s(i_j) = i_{j+1}, \qquad 1 \leq j < k, \qquad s(i_k) = i_1, \qquad s(m) = m, \ \ \forall m \notin \{i_1, ..., i_k\}.$$

Then $s^r = r^{-1}sr$ is the permutation sending $(i_j)r$ to $(i_{j+1})r$:

$$s^r(r(i_j)) = r(i_{j+1}), \qquad 1 \leq j < k, \qquad s^r(r(i_k)) = r(i_1),$$

$$s^r(m) = m, \ \ \forall m \notin \{r(i_1), ..., r(i_k)\}.$$

Example 4.4.2. *Let us label the 12 edges of the Rubik's cube using the* **Singmaster notation***:*

- *uf denotes the "up, front edge",*
- *ul denotes the "up, left edge",*
- *ur denotes the "up, right edge",*
- *ub denotes the "up, back edge",*
- *df denotes the "down, front edge",*

- *dl denotes the "down, left edge",*
- *dr denotes the "down, right edge",*
- *db denotes the "down, back edge",*
- *fl denotes the "front, left edge",*
- *br denotes the "front, right edge",*
- *fl denotes the "back, left edge",*
- *br denotes the "back, right edge".*

Suppose you have a Rubik's cube move s which is a 3-cycle on 3 particular edges, say

$$uf \longmapsto ul \longmapsto ur \longmapsto uf.$$

Suppose you have another move r which sends these edges somewhere else, say $r = F^2$ *so that* $r : uf \longmapsto df$ *and* $r : fl \longmapsto fr$*, but leaves the other edges alone. Then* $r^{-1}sr$ *is the permutation*

$$df \longmapsto ul \longmapsto ur \longmapsto df.$$

Try it!

The most common notation for a permutation is the "cycle notation". This notation is more compact that the array notation we've been using and from this point on we will switch over to the cycle notation. If $a_1, a_2, ..., a_r \in \mathbb{Z}_n$ then the symbol

$$(a_1\ a_2\ ...\ a_r) \qquad \text{(some r less than or equal to n)}$$

denotes the permutation f of $\mathbb{Z}_n$ which sends a_1 to a_2, sends a_2 to a_3, ..., sends a_{r-1} to a_r, sends a_r to a_1, and leaves all the other numbers in $\mathbb{Z}_n$ alone. In other words,

$$f(a_1) = a_2,\ \ f(a_2) = a_3,\ \ ..., f(a_r) = a_1,$$

and $f(i) = i$, if i is not equal to one of the $a_1, ..., a_r$. Such a permutation is called **cyclic**. The number r is called the **length** of the cycle.

We call two such cycles $(a_1\ a_2\ ...\ a_r)$ and $(b_1\ b_2\ ...\ b_t)$ **disjoint** if the sets $\{a_1, a_2, ..., a_r\}$ and $\{b_1, b_2, ..., b_t\}$ are disjoint.

Lemma 4.4.3. *If f and g are disjoint cyclic permutations of $\mathbb{Z}_n$ then $fg = gf$.*

proof: This is true because the permutations f and g of $\mathbb{Z}_n$ affect disjoint collections of integers, so the permutations may be performed in either order. □

Lemma 4.4.4. *The cyclic permutation $(a_1,\ a_2,\ ..., a_r)$ has order r.*

proof: Note $f(a_1) = a_2$, $f^2(a_1) = a_3$, ..., $f^{r-1}(a_1) = a_r$, $f^r(a_1) = a_1$, by definition of f. Likewise, for any $i = 1, ..., r$, we have $f^r(a_i) = a_i$. □

Definition 4.4.5.

A **transposition** *is a cycle (i, j) of length 2 which interchanges i and j.*

Theorem 4.4.6. *Every permutation $f : \{1, 2, ..., n\} \to \{1, 2, ..., n\}$ is the product of disjoint cyclic permutations. More precisely, if f is a permutation of $\{1, 2, ..., n\}$ (with $n > 1$) then there are non-empty disjoint subsets of distinct integers*

$$\begin{aligned} S_1 &= \{a_{11}, ..., a_{1,r_1}\} \subset \{1, 2, ..., n\}, \\ S_2 &= \{a_{21}, ..., a_{2,r_2}\} \subset \{1, 2, ..., n\}, \\ &..., \\ S_k &= \{a_{k1}, ..., a_{k,r_k}\}, \end{aligned}$$

such that

$$\{1, 2, ..., n\} = S_1 \cup ... \cup S_k, \qquad n = r_1 + r_2 + ... + r_k,$$

and

$$f = (a_{11}, ..., a_{1,r_1})...(a_{k1}, ..., a_{k,r_k}).$$

This product is called a **cycle decomposition** of f . If we rearrange the cardinalities r_i of these sets S_i in decreasing order, say we write this as

$$r'_1 \geq r'_2 \geq ... \geq r'_k,$$

then the k-tuple $(r'_1, ..., r'_k)$ is called the **cycle structure** of f and f is called a $(r'_1, ..., r'_k)$**-cycle**. For example, $(1, 2)(3, 4, 5)$ is a $(3, 2)$-cycle.

proof: The proof is constructive.

Let $f : \mathbb{Z}_n \to \mathbb{Z}_n$ be a permutation. List all the *distinct* elements in $\mathbb{Z}_n$ you get by repeatedly applying f to 1. Call them (for lack of a better notation!), $a_{10}, a_{11}, a_{12}, ..., a_{1,r_1}$, where,

$$O_1 = \{a_{10} = 1, a_{11} = f(1), a_{12} = f^2(1), ..., a_{1,r_1} = f^{r_1}(1)\}.$$

(Incidently, this is called the **orbit of** 1 **under** f. We shall discuss orbits more later in this book.) Now list the elements in the orbit of 2:

$$O_2 = \{a_{20} = 2, a_{21} = f(2), a_{22} = f^2(2), ..., a_{2,r_2} = f^{r_2}(2)\},$$

and so on until we get to the "orbit of n":

$$O_n = \{a_{n0} = n, a_{n1} = f(n), a_{n2} = f^2(n), ..., a_{n,r_n} = f^{r_n}(n)\}.$$

An example: let $f : \mathbb{Z}_3 \to \mathbb{Z}_3$ be the 3-cycle permutation, $f(1) = 2, f(2) = 3, f(3) = 1$. In this case, $O_1 = \{1, 2, 3\}$, $O_2 = \{2, f(2) = 3, f(f(2)) = f(3) = 1\}$, and $O_3 = \{3, f(3) = 1, f(f((3)) = f(1) = 2\}$, so all three orbits are equal to each other in this example.

In general, if you pick any two of these n orbits O_1, ..., O_n, they will either be the same or disjoint. Denote all the *distinct* orbits by O'_1, ..., O'_k. (The O'_1, ..., O'_k are a subsequence of the O_1, ..., O_n. It doesn't matter what order you write the O'_i's in or in what order you write the elements in each individual orbit.) Suppose that

$$\begin{aligned} O'_1 &= \{b_{11}, ..., b_{1,s_1}\} \quad \text{so } |O'_1| = s_1, \\ O'_2 &= \{b_{21}, ..., b_{2,s_2}\} \quad \text{so } |O'_2| = s_2, \\ &\vdots \\ O'_k &= \{b_{k1}, ..., b_{k,s_k}\} \quad \text{so } |O'_k| = s_k. \end{aligned}$$

(The a_{ij}'s have been relabeled as b_{ij}'s to try to simplify the notation.) In this case,

$$\mathbb{Z}_n = \cup_{i=1}^{k} O'_i = O'_1 \cup ... \cup O'_k,$$

and $s_1 + s_2 + ... + s_k = n$. The restriction of f to O'_1, denoted $f_{O'_1} : O'_1 \to O'_1$, is equal to the s_1-cycle $(b_{11}, b_{12}, ..., b_{1,s_1})$. In general, the restriction of f to O'_j, denoted $f_{O'_j} : O'_1 \to O'_j$, is equal to the s_j-cycle $(b_{j1}, b_{j2}, ..., b_{j,s_j})$. Since the O'_j's partition $\mathbb{Z}_n$, the definition of f and our construction implies that

$$f = (b_{11}, b_{12}, ..., b_{1,s_1})...(b_{k1}, b_{k2}, ..., b_{k,s_k}).$$

□

Example 4.4.7.

- *The cycle notation for*

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$$

is $(1,2)(3)$ *or simply* $(1,2)$. *In general, if any of the orbits* O_j *in the above construction is a singleton, it is often omitted from the notation, with the implicit understanding that* f *doesn't permute the omitted numbers.*

- *The cycle notation for*

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$$

is $(1,\ 2,\ 3)$.

- *The cycle notation for*

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix}$$

is $(1,\ 3)(2,\ 4) = (2,\ 4)(1,\ 3)$.

- *The cycle notation for*

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 1 & 5 & 2 \end{pmatrix}$$

is $(1,\ 3)(2,\ 4,\ 5) = (4,\ 5,\ 2)(1,\ 3)$.

- *The disjoint cycle decomposition of* $(2,3,7)(3,7,10)$ *is* $(2,3)(7,10)$.

Lemma 4.4.8. *A cyclic permutation is even if and only if the length of its cycle is odd. A general permutation* $f : \mathbb{Z}_n \to \mathbb{Z}_n$ *is odd if and only if the number of cycles of even length in its cycle decomposition is odd.*

This follows from the definition of even/odd (see Definition 4.2.2) and the fact that $sign(p_1 p_2 ... p_k) = sign(p_1)sign(p_2)...sign(p_n)$, for permutations p_i (because of Lemma 4.2.6). The proof is left as an exercise.

Lemma 4.4.9. *The order of a permutation is the least common multiple (lcm) of the lengths* $r_1, r_2, ..., r_k$ *of the disjoint cycles in its cycle decomposition.*

Example 4.4.10. *The order of* $(1,\ 3)(2,\ 4)$ *is* 2. *It is even. The order of* $(1,\ 3)(2,\ 4,\ 5)$ *is* 6. *It is odd.*

Exercise 4.4.11. *Let* $s = (1,2,3)$ *and* $r = (1,3)(2,4)$ *be permutations of* $\{1,2,3,4\}$. *Compute* r^1sr *using Lemma 4.4.1.*

Exercise 4.4.12. *Let* $s = (1,5)(2,4,3)$ *and* $r = (1,2,3,4,5)$ *be permutations of* $\{1,2,3,4,5\}$. *Compute* r^1sr *using Lemma 4.4.1.*

Exercise 4.4.13. *(a) Show that the inverse of* (a_1,a_2,a_3,a_4) *is* $(a_1,a_4,a_3,a_2) = (a_4,a_3,a_2,a_1)$.

(b) Find the inverse of $(a_1,a_2,a_3,...,a_n)$.

Exercise 4.4.14. *Consider the equation given in* $2 \times n$ *array form as*

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 2 & 3 & 4 \end{pmatrix} \cdot x \cdot \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 2 & 5 & 4 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 1 & 2 & 5 & 3 \end{pmatrix}.$$

Solve for x *in two ways:*

(i) by directly considering what $2 \times n$ *array would 'fit' for* x *(so that the composition on the left hand side would indeed by the right side)*

(ii) solving for x *algebraically (hint: if* $axb^{-1} = c$, *then* $x = a^{-1}cb$*).*

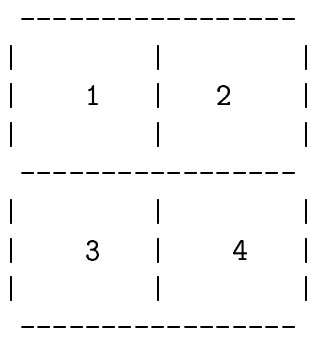
It is recommended that you rewrite the equation first in disjoint cycle notation, then solve.

Exercise 4.4.15. *(a) Show that* $(a_1,a_2,a_3,a_4) = (a_1,a_2)(a_1,a_3)(a_1,a_4)$.

(b) Express $(a_1,a_2,...a_n)$ *as a product of transpositions.*

Exercise 4.4.16. *When first opened, a new package containing a standard deck of 52 cards is typically found to be in a naturally ordered state (and the customer can quickly verify that all cards are present). How does (b) of Exercise 4.4.15 show that any deck of cards, no matter how thoroughly shuffled, can be brought back to its original state simply by the following single operation: repeatedly swapping various cards with the current top card?*

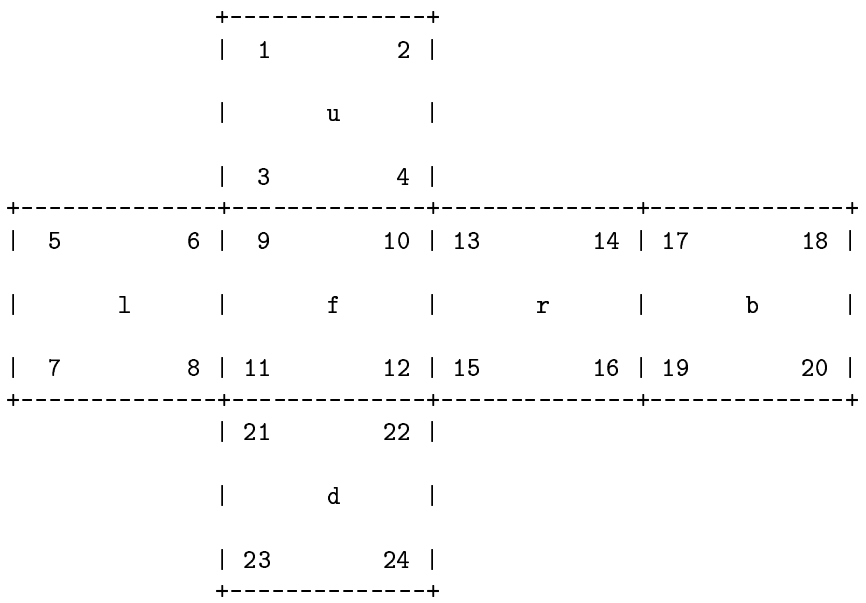
Exercise 4.4.17. *Divide a square into 4 subsquares ("facets") and label them* $1,2,3,4$. *For example,*



Let r denote counterclockwise rotation by 90 degrees. Then, as a permutation on the facets, $r = (1,\ 3,\ 4,\ 2)$. Let f_x denote reflection about the horizonal line dividing the square in two, let f_y denote reflection about the vertical line dividing the square in two. Use the cycle notation to determine the permutations of the facets

(a) r^2
(b) r^3,
(c) f_x,
(d) f_y,
(e) $f_x r f_x$,
(f) $f_x f_y$,
(c) f_x^{-1},
(d) f_y^{-1}.

Exercise 4.4.18. *Label the 24 facets of the 2×2 Rubik's cube as follows:*



(You may want to xerox this page then cut this cube out and tape it together for this.) Let X denote rotation clockwise by 90 degrees of the face labeled x, where $x \in \{r, l, f, b, u, d\}$ (so, for example, if $x = f$ then $X = F$). Use the cycle notation to determine the permutations of the facets given by

(a) R,
(b) L,
(c) F,
(d) B,
(e) U,
(f) D.

4.5 An algorithm to list all the permutations

In Martin Gardner's Scientific American article [Gar1] an algorithm is mentioned which lists all the permutations of $\{1, 2, ..., n\}$. This algorithm gives the fastest known method of listing all permutations of $\{1, 2, ..., n\}$. Rediscovered many times since, the procedure is due originally to the Polish mathematician Hugo Steinhaus (January 1887-February 1972), a student of David Hilbert at Göttingen, did important work on orthogonal series, probability theory, real functions and their applications.

We shall denote each permutation by the second row in its $2 \times n$ array notation. For example, in the case $n = 2$:

$$\begin{matrix} 1 & 2 \\ 2 & 1 \end{matrix}$$

are the permutations.

To see the case $n = 3$, the idea is to

(a) write down each row $n = 3$ times each as follows:

$$\begin{matrix} 1 & 2 \\ 1 & 2 \\ 1 & 2 \\ 2 & 1 \\ 2 & 1 \\ 2 & 1 \end{matrix}$$

(b) "weave" in a 3 as follows

$$\begin{matrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 3 & 1 & 2 \\ 3 & 2 & 1 \\ 2 & 3 & 1 \\ 2 & 1 & 3 \end{matrix}$$

In case $n = 4$, the idea is to

(a) write down each row $n = 4$ times each as follows:

1	2	3
1	2	3
1	2	3
1	2	3
1	3	2
1	3	2
1	3	2
1	3	2
3	1	2
3	1	2
3	1	2
3	1	2
3	2	1
3	2	1
3	2	1
3	2	1
2	3	1
2	3	1
2	3	1
2	3	1
2	1	3
2	1	3
2	1	3
2	1	3

(b) now "weave" a 4 in:

$$\begin{array}{cccc}
1 & 2 & 3 & 4 \\
1 & 2 & 4 & 3 \\
1 & 4 & 2 & 3 \\
4 & 1 & 2 & 3 \\
4 & 1 & 3 & 2 \\
1 & 4 & 3 & 2 \\
1 & 3 & 4 & 2 \\
1 & 3 & 2 & 4 \\
3 & 1 & 2 & 4 \\
3 & 1 & 4 & 2 \\
3 & 4 & 1 & 2 \\
4 & 3 & 1 & 2 \\
4 & 3 & 2 & 1 \\
3 & 4 & 2 & 1 \\
3 & 2 & 4 & 1 \\
3 & 2 & 1 & 4 \\
2 & 3 & 1 & 4 \\
2 & 3 & 4 & 1 \\
2 & 4 & 3 & 1 \\
4 & 2 & 3 & 1 \\
4 & 2 & 1 & 3 \\
2 & 4 & 1 & 3 \\
2 & 1 & 4 & 3 \\
2 & 1 & 3 & 4
\end{array}$$

In general, we have the following

Theorem 4.5.1. *(Steinhaus) There is a sequence of (not necessarily distinct) 2-cycles, (a_1, b_1),...,(a_N, b_N), where $N = n! - 1$, such that each non-trivial permutation f of $\{1, 2, ..., n\}$ may be expressed in the form*

$$f = \prod_{i=1}^{k} (a_i, b_i),$$

for some k, $1 \leq k \leq N$. Furthermore, these products (for $k = 1, 2, ..., N$) are all distinct.

In other words, each permutation can be written as a product of (not necessarily disjoint) 2-cycles[1].

Example 4.5.2. *Let us consider the result of Steinhaus' algorithm in the case of S_3. Recall the output from of the algorithm:*

$$\begin{array}{ccc} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 3 & 1 & 2 \\ 3 & 2 & 1 \\ 2 & 3 & 1 \\ 2 & 1 & 3 \end{array}$$

In disjoint cycle notation,

$$\begin{array}{ll} [1,2,3] = (1) = g_1, & [1,3,2] = (2,3) = g_2 = (2,3)g_1, \\ [3,1,2] = (1,3,2) = (1,2)g_2 = g_3, & [3,2,1] = (1,3) = (2,3)g_3 = g_4, \\ [2,3,1] = (1,2,3) = (1,3)g_4 = g_5, & [2,1,3] = (1,2) = (2,3)g_6 = g_6. \end{array}$$

Note that every element $g \in S_3$ can be written in the form of a product of the simple transpositions $(1,2)$ and $(2,3)$.

Exercise 4.5.3. *Prove Theorem 4.5.1. (Hint: Use Exercise 4.4.15.)*

4.6 Application: The rotation game

On some cell phones, in particular the Nokia 7160 (tm), there is a game called "Rotation". It is a 3×3 grid, of the form

$$\begin{array}{ccc} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{array}.$$

The letters could be scrambled randomly in an actual game. The allowed moves are rotations of the following form, or combinations thereof: In cycle notation,

[1] There is an analogous result valid only for *even* permutations: each even permutation may be written as a product of (not necessarily disjoint) 3-cycles.

- $r_1 = (a_1, a_2, a_5, a_4)$, which sends the above grid to

$$\begin{array}{ccc} a_4 & a_1 & a_3 \\ a_5 & a_2 & a_6 \\ a_7 & a_8 & a_9 \end{array},$$

- $r_2 = (a_2, a_3, a_6, a_5)$, which sends the above grid to

$$\begin{array}{ccc} a_1 & a_5 & a_2 \\ a_4 & a_6 & a_3 \\ a_7 & a_8 & a_9 \end{array},$$

- $r_3 = (a_4, a_5, a_8, a_9)$, which sends the above grid to

$$\begin{array}{ccc} a_1 & a_2 & a_3 \\ a_7 & a_4 & a_6 \\ a_8 & a_5 & a_9 \end{array},$$

- $r_4 = (a_5, a_6, a_9, a_8)$, which sends the above grid to

$$\begin{array}{ccc} a_1 & a_2 & a_3 \\ a_4 & a_8 & a_5 \\ a_7 & a_9 & a_6 \end{array}.$$

In other words, a legal move in the Rotation game is any permutation of the form $r_{i_1} r_{i_2} ... r_{i_k}$, where $1 \leq i_j \leq 4$ for all $1 \leq j \leq N$.

Question: Can each permutation of $\{1, 2, ..., 9\}$ be expressed in the form $r_{i_1} r_{i_2} ... r_{i_k}$, where $1 \leq i_j \leq 4$ for all $1 \leq j \leq N$?

This question turns out to be to answer using group theory and a CA system such as GAP, MAGMA, or MAPLE. The next chapter provides the necessary background.

4.7 Application: Bell ringing

A good reference for this section is A. White [Wh].

Since the seventeenth century, and possibly before, cathedral bells in England have been rung by permutating the order of a "round" of bells. The art and study of such bell ringing is referred to as campanology.

Fabian Stedman provided significant contributions to bell ringing. Born in 1640 to Reverend Francis Stedman, Fabian Stedman's connections with campanology took root at the early age of 15 when he moved to London to work as an apprentice to a Master Printer. While in London Stedman joined a bell-ringing society known as the "Society of Colledg(sic) Youths," which has since been renamed the Ancient Society of College Youths and is still in existence today. Stedman's major contributions to campanology are reflected in his efforts on **Tintinnlogia** and **Campanalogia**, the first two books published on the subject, in 1668 and 1677, respectively.

Below is a glossary of a few essential terms:

- **Change**: the swapping of one or more disjoint pairs of adjacent bells
- **Plain change**: involves swapping one pair of adjacent bells only
- **Cross change**: involves more than one swapping pair of bells
- **Round**: A ordering of the bells (i.e., a permutation of $(1, 2, 3, ..., n)$)

In the beginning, change ringing concerned itself with a single row of bells whose order could be denoted by $(1, 2, 3, ..., n)$. Considering the case where $n = 6$ the concepts of plain and cross changes can be understood more clearly. If we use only plain changes we can generate permutations of the bells as follows:

$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 1 & 3 & 4 & 5 & 6 \\ 2 & 1 & 4 & 3 & 5 & 6 \\ 2 & 1 & 4 & 3 & 6 & 5, \end{array}$$

It should be fairly obvious on inspection that the first plain change swaps 1 and 2, the second swaps 3 and 4, and the third swaps 5 and 6. Considering the same set of six bells acted upon by a cross change, the same result is achieved in one change, as seen below:

$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 1 & 4 & 3 & 6 & 5. \end{array}$$

More useful and interesting patterns can be generated by combining plain and cross changes. The **plain lead on four bells** is one of the most simplistic

patterns and was devised sometime around 1621 by alternating consecutive cross and plain changes as seen below:

$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \\ 2 & 4 & 1 & 3 \\ 4 & 2 & 3 & 1 \\ 4 & 3 & 2 & 1 \\ 3 & 4 & 1 & 2 \\ 3 & 1 & 4 & 2 \\ 1 & 3 & 2 & 4 \\ 1 & 2 & 3 & 4 \end{array}$$

It is easy to see that the pattern which defines the plain lead on four bells is nothing more than a cross change followed by a plain change on the middle two bells until we reach the round, which is where we started.

Generating the permutations contained in the plain lead on four bells can be easily described using the notation for permutations we have developed. We begin by representing the cross change as $a = (1,2)(3,4)$, which swaps the first two and last two bells, and representing the plain change as $b = (2,3)$, which swaps the middle pair. We begin with the first element, a. To generate the next permutation we multiply this first element by b. To generate the third element we simply multiply this second term, ab, by a to get aba. Continuing on in this manner we multiply alternately by a then b to generate the set $D_4 = \{a, ab, aba, (ab)^2, (ab)^2a, (ab)^3, (ab)^3a, (ab)^4\}$. This is the set of permutations in the plain lead on four bells. (It is denoted D_4 here for reasons which will be explained later. Since $(ab)^4$ yields the original round, we say $(ab)^4 = 1$ and $D_4 = \{1, a, ab, aba, (ab)^2, (ab)^2a, (ab)^3, (ab)^3a\}$.

Now for a more complex example. We turn our attention now to the composition which is commonly referred to as **Plain Bob Minimus**. Plain Bob Minimus begins at the round and ends at the round $(1,2,3,4)$ and contains all possible permutations of these four bells (in a particular order, described below):

$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \\ 2 & 4 & 1 & 3 \\ 4 & 2 & 3 & 1 \\ 4 & 3 & 2 & 1 \end{array} \qquad \begin{array}{cccc} 1 & 3 & 4 & 2 \\ 3 & 1 & 2 & 4 \\ 3 & 2 & 1 & 4 \\ 2 & 3 & 4 & 1 \\ 2 & 4 & 3 & 1 \end{array} \qquad \begin{array}{cccc} 1 & 4 & 2 & 3 \\ 4 & 1 & 3 & 2 \\ 4 & 3 & 1 & 2 \\ 3 & 4 & 2 & 1 \\ 3 & 2 & 4 & 1 \end{array}$$

3 4 1 2	4 2 1 3	2 3 1 4
3 1 4 2	4 1 2 3	2 1 3 4
1 3 2 4	1 4 3 2	1 2 4 3
		1 2 3 4

We can now describe this composition using the permutation notation as we did for the plain lead on four bells. Let $a = (1,2)(3,4)$ and $b = (2,3)$ represent possible changes between rows. If we look at the first column of the Plain Bob Minimus composition, we see that it is nothing more than the plain lead on four bells. To generate the second column, we introduce a new permutation $c = (3,4)$ and we simplify our notation by letting $k = (ab)^3ac$. Multiplying through we generate the second column,

$$\{k, ka, kab, kaba, k(ab)^2, k(ab)^2a, k(ab)^3, k(ab)^3a\}.$$

Multiplying through by k again, we generate the third column,

$$\{k^2, k^2a, k^2ab, k^2aba, k^2(ab)^2, k^2(ab)^2a, k^2(ab)^3, k^2(ab)^3a\}.$$

This generation of Plain Bob Minimus shows that it can be expressed as the disjoint union of "translations" of the plain lead on four bells! We shall explain this fact using group theory in the next chapter.

Now, since we chose $a = (1,2)(3,4)$, $b = (2,3)$, and $c = (3,4)$, where b and c are obviously by definition 2-cycles (or transpositions) and a is the product of two such 2-cycles or transpositions, we have shown a further result, that each permutation of $\mathbb{Z}_4$ can be written as a product of 2-cycles. More generally, we can state the following theorem originally due to H. Steinhaus.

Theorem 4.7.1. *Let f be a member of S_n, i.e., let f be any permutation of degree n. Then f can be written as a product of transpositions.*

To sketch a proof of this theorem (following [G]) and hence prove Theorem 4.5.1 as promised, we need only to recall that: Every permutation of S_n can be written uniquely (up to order) as a product of disjoint cycles (Theorem 4.4.6).

proof: It is a fact that any k-cycle can be written as a product of $k-1$ transpositions,

$$(a_1, a_2, ..., a_k) = (a_1, a_k)(a_1, a_{k-1})...(a_1, a_2).$$

(This can be proven by mathematical induction. The argument is left to the reader.) We see that since any permutation can be written in terms of cycles and any cycle can be written as product of transposition, it follows that every permutation of $\mathbb{Z}_n$ can be written as a product of transpositions. □

4.8 Application: Rubik's cubes

First, some general terminology which applies to any Rubik's cube-like book.

A **one person game** is a sequence of moves following certain rules satisfying

- there are finitely many moves at each stage,
- there is a finite sequence of moves which yields a solution,
- there are no chance or random moves,
- there is complete information about each move,
- each move depends only on the present position, not on the existence or non-existence of a certain previous move (such as chess, where castling is made illegal if the king has been moved previously).

The reader will find a fascinating and much fuller discussion of combinatorial game theory in the book by Berlekamp, Conway, and Guy [BCG] (volumes I and II).

A **permutation puzzle** is a one person game with the following five properties listed below. Before listing the properties, we define the "puzzle position" to be the set of all possible legal moves. The five properties of a permutation puzzle are:

1. for some $n > 1$ depending only on the puzzle's construction, each move of the puzzle corresponds to a unique permutation of the numbers in $T = \{1, 2, ..., n\}$,
2. if the permutation of T in (1) corresponds to more than one puzzle move then the the two positions reached by those two respective moves must be indistinguishable,

3. each move, say M, must be "invertible" in the sense that there must exist another move, say M^{-1}, which restores the puzzle to the position it was at before M was performed,

4. if M_1 is a move corresponding to a permutation f_1 of T and if M_2 is a move corresponding to a permutation f_2 of T then $M_1 * M_2$ (the move M_1 followed by the move M_2) is either

   - not a legal move, or
   - corresponds to the permutation $f_1 * f_2$.

Notation: As in step 4 above, we shall always write successive puzzle moves *left-to-right*.

We shall consider briefly the 2×2 and 3×3 Rubik's cubes.

4.8.1 2×2 Rubik's cube

The "pocket" Rubik's cube has 6 sides, or "faces", each of which has $2 \cdot 2 = 4$ "facets", for a total of 24 facets.

Fix an orientation of the Rubik's cube in space. Therefore, we may label the 6 sides as f, b, l, r, u, d, as in the picture. It has 8 subcubes. Each face of the cube is associated to a "slice" of 4 subcubes which share a facet with the face. The face, along with all of the 4 cubes in the "slice", can be rotated by 90 degrees clockwise. We denote this move by the upper case letter associated to the lower case letter denoting the face. For example, F denotes the move which rotates the front face by 90 degrees to clockwise.

We label the 24 facets of the 2×2 Rubik's cube as in Exercise 4.4.18. The 24 facets will be denoted by xyz where x is the face on which the facet lives and y, z (or z, y - it doesn't matter) indicate the 2 edges of the facet. Written in clockwise order:

```
front face:  fru, frd, fld, flu
 back face:  blu, bld, brd, bru
right face:  rbu, rbd, rfd, rfu
 left face:  lfu, lfd, lbd, lbu
   up face:  urb, urf, ulf, ulb
 down face:  drf, drb, dlb, dlf
```

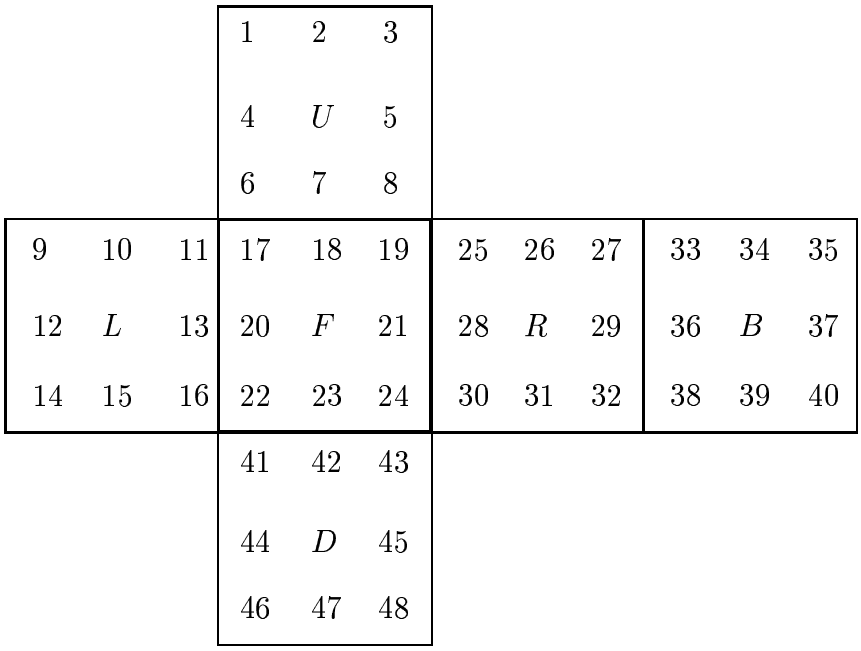
For future reference, we call this system of notation (which we will also use for the 3×3 Rubik's cube) the **Singmaster notation**.

Exercise 4.8.1. *Verify that the properties of a permutation puzzle are satisfied for this puzzle.*

4.8.2 3×3 Rubik's cube

In this section we shall, for the most part, simply introduce enough notation (due to Singmaster [Si]) to allow us to check that the puzzle is in fact a permutation puzzle. We shall also introduce a two-person game which is easier to play and learn than solving the cube.

The Rubik's cube has 6 sides, or "faces", each of which has $3 \cdot 3 = 9$ "facets", for a total of 54 facets. We label these facets $1, 2, ..., 54$ as follows:



The generators, corresponding to the six faces of the cube, may be written

in disjoint cycle notation as:

$$\begin{aligned} F &= (17,19,24,22)(18,21,23,20)(6,25,43,16)(7,28,42,13)(8,30,41,11),\\ B &= (33,35,40,38)(34,37,39,36)(3,9,46,32)(2,12,47,29)(1,14,48,27),\\ L &= (9,11,16,14)(10,13,15,12)(1,17,41,40)(4,20,44,37)(6,22,46,35),\\ R &= (25,27,32,30)(26,29,31,28)(3,38,43,19)(5,36,45,21)(8,33,48,24),\\ U &= (1,3,8,6)(2,5,7,4)(9,33,25,17)(10,34,26,18)(11,35,27,19),\\ D &= (41,43,48,46)(42,45,47,44)(14,22,30,38)(15,23,31,39)(16,24,32,40). \end{aligned} \tag{4.1}$$

The notation for the facets will be similar to the notation used for the 2×2 Rubik's cube. The corner facets will have the same notation and the edge facets will be denoted by xy, where x is the face the facet lives on and y is the face the facet borders to. In clockwise order, starting with the upper right-hand corner of each face:

```
front face:  fru, fr, frd, fd, fld, fl, flu, fu
 back face:  blu, bl, bld, bd, brd, br, bru, bu
right face:  rbu, rb, rbd, rd, rfd, rf, rfu, ru
 left face:  lfu, lf, lfd, ld, lbd, lb, lbu. lu
   up face:  urb, ur, urf, uf, ulf, ul, ulb, ub
 down face:  drf, dr, drb, db, dlb, dl, dlf, df
```

Exercise 4.8.2. *Check that the cycles in (4.1) are correct. (It is helpful to xerox the above diagram, cut it out and tape together a paper cube for this exercise.)*

Exercise 4.8.3. *Verify that the properties of a permutation puzzle are satisfied for this puzzle.*

The following exercises require a Rubik's cube.

Exercise 4.8.4. *Verify that $(R^2U^2)^3$ is the product of 2-cycles $(uf, ub)(fr, br)$ on the edges.*

Exercise 4.8.5. *Let $M = FD^{-1}R$. Verify that $(M^{-1}D^2MU^2)^2$ is the product of 2-cycles $(ufr, ufl)(ubr, ubl)$ on the corners.*

Exercise 4.8.6. *Let $M = R^{-1}D^{-1}R$. Verify that $M^{-1}UMU^{-1}$ is the 3-cycle (bru, dlf, urf) on the corners.*

Exercise 4.8.7. *Verify that $R^2UFB^{-1}R^2F^{-1}BUR^2$ is the 3-cycle (uf, ub, ur) on the edges.*

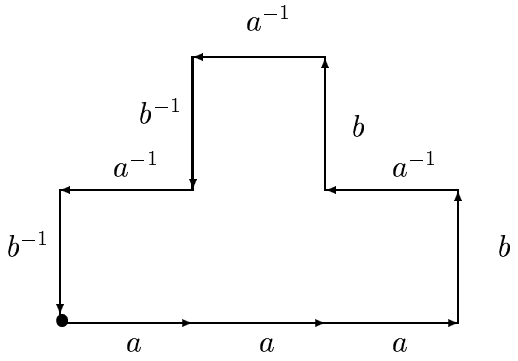
Exercise 4.8.8. *Verify that $(R^{-1}D^2RB^{-1}U^2B)^2$ twists the ufr corner clockwise and the dbl corner counterclockwise.*

4.9 Special project: Tiling with groups

A good reference for this section is Thurston's article [Th].

Let P be any oriented path on a rectangular lattice. Let a, b be any elements of some group G. We associate to P a "word" in a, b as follows. For each move $\rightarrow$ by one unit in P, write a. For each move $\leftarrow$ by one unit in P, write a^{-1}. For each move $\uparrow$ by one unit in P, write b. For each move $\downarrow$ by one unit in P, write b^{-1}.

Example 4.9.1. *Let P be the closed path pictured below, where $\bullet$ denotes the initial (and terminal) point.*



The word associated to P is

$$w(P) = a^3ba^{-1}ba^{-1}b^{-1}a^{-1}b^{-1}.$$

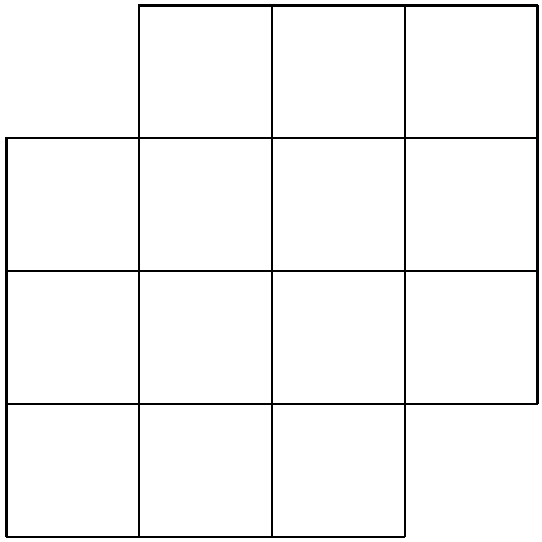
Exercise 4.9.2. *Let $w_P(a, b)$ be the word associated to a path P. Let $-P$ denote the path associated to the path P with every edge having the opposite orientation. Show $w_{-P}(a, b) = w_P(a, b)^{-1}$.*

Exercise 4.9.3. *Conside the tile in Example 4.9.1. Find the words associated to each of its rotations and reversals.*

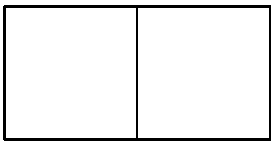
Exercise 4.9.4. *Show that the square 4×4 grid can be tiled using the tile in Example 4.9.1 along with its rotations and reversals.*

Exercise 4.9.5. *Using group theory, show that the square* 10×10 *grid cannot be tiled using the tile in Example 4.9.1 along with its rotations and reversals.*

Exercise 4.9.6. *Show that the* 4×4 *square with missing upper left-hand and lower right-hand corners*



cannot be tiled by the 1×2 *tile*



along with rotations by multiples of 90^o *and reversals.*

4.10 Special project: Latin squares

A **Latin square** of order n consists of n distinct symbols, each occurring n times, arranged in an $n \times n$ square array in such a way that each row and

column is a permutation of the n symbols. For example

$$\begin{array}{cccccccc} 2 & 3 & 4 & 5 & 6 & 7 & 8 & 1 \\ 3 & 2 & 5 & 1 & 4 & 6 & 7 & 8 \\ 4 & 8 & 1 & 2 & 3 & 5 & 6 & 7 \\ 1 & 7 & 2 & 3 & 8 & 4 & 5 & 6 \\ 6 & 1 & 3 & 7 & 2 & 8 & 4 & 5 \\ 5 & 4 & 6 & 8 & 7 & 2 & 1 & 3 \\ 7 & 5 & 8 & 6 & 1 & 3 & 2 & 4 \\ 8 & 6 & 7 & 4 & 5 & 1 & 3 & 2 \end{array}$$

is a Latin square of order 7 on the symbols $\{1, 2, 3, 4, 5, 6, 7\}$. Also,

$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \\ 3 & 4 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{array}$$

is a Latin square of order 5 on the symbols $\{1, 2, 3, 4, 5\}$.

Lemma 4.10.1. *If G is any finite group the the multiplication table of G is a Latin square of order $|G|$.*

We leave the verification of this as an exercise.

A **derangement** of the symbols $\mathbb{Z}_n = \{1, 2, ..., n\}$ is a permutation which, when regarded as a function $f : \mathbb{Z}_n \to \mathbb{Z}_n$, has no fixed points (i.e., $f(x) \neq x$ for all $x \in \mathbb{Z}_n$). For example,

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 2 & 3 & 4 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 5 & 1 & 3 \end{pmatrix}$$

are derangements. In other words, $p \in S_n$ has no fixed points if and only if, when written in $2 \times n$ array notation,

$$p = \begin{pmatrix} 1 & 2 & \dots & n \\ p_1 & p_2 & \dots & p_n \end{pmatrix}, \tag{4.2}$$

the n-cycle $(p_1, p_2, ..., p_n)$ is a derangement.

Exercise 4.10.2. *If L is a Latin square of order n in the symbols $\mathbb{Z}_n$ and if the first row of L is $(1, 2, ..., n)$ then every other row must be a derangement.*

Exercise 4.10.3. *If $L = (L_{ij})$ is a Latin square of order n in the symbols $\mathbb{Z}_n$ and if $P(k) = (p_{ij}(k))$ is the $n \times n$ matrix defined by*

$$p_{ij}(k) = \begin{cases} 1, & L_{ij} = k, \\ 0, & L_{ij} \neq k, \end{cases}$$

then $P(k)$ is a permutation matrix.

Exercise 4.10.4. *If L is a Latin square of order n in the symbols $\mathbb{Z}_n$ and if the diagonal of L consists only of 1's (such as in the second example above) then every other row must be a derangement.*

We call $p, q \in S_n$ **independent** if pq^{-1} has no fixed points in $\{1, 2, ..., n\}$. Let A be an $n \times n$ Latin square with symbols from $\{a_1, ..., a_n\} \subset \mathbb{Z}$. Then there is a unique set S of n independent permutation matrices, $S = \{p_1, ..., p_n\}$, such that

$$A = a_1 p_1 + ... + a_n p_n. \tag{4.3}$$

This correspondence between Latin square and subsets $S \subset S_n$ of n independent permutation matrices is 1-1.

Let $p \in S_n$ be a derangement of order n. Then

$$S = \{1, p, ..., p^{n-1}\}$$

is a set of n independent permutation matrices. Call this the **independent set associated to** p, denoted $S(p)$.

Example 4.10.5. *For instance, in the notation of (4.3), when $n = 4$,*

$$p = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix},$$

$p_1 = 1$, and $p_i = p^i$, $2 \leq i \leq 4$, we get the Latin square

$$\begin{array}{cccc} 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \\ 2 & 3 & 4 & 1 \\ 1 & 2 & 3 & 4 \end{array}$$

Let $p, q \in S_n$ be derangements of order n. To determine when $S(p) = S(q)$, we introduce the following notion.

Definition 4.10.6. *Let $p, q \in S_n$. We say p is* **equivalent** *to q, written $p \sim q$, if $p = q^i$, for some $i \in \mathbb{Z}$.*

Note that if $p \sim q$, p and q have the same order, and $p = q^i$ then i is relatively prime to the order of q.

Here's a "group-theoretical" construction of Latin squares.

(1) Set an element p from a complete set of representatives of the equivalence classes of derangements of order n.

(2) For each $\sigma \in S_{n-1}$, construct the Latin square associated to the independent n-sets
$$\{1, p^{\sigma(1)}, p^{\sigma(2)}, \ldots, p^{\sigma(n-1)}\}.$$

Note each of these Latin squares "contains" the diagonal. Each Latin square can be put into "diagonal form" by first selecting a symbol and then applying a permutation to make that symbol "diagonal".

Exercise 4.10.7. *Construct at least 5 examples of Latin squares using this method.*

Chapter 5

An introduction to groups

Groups are often used to symbolically describe symmetry or invariance of some type. Such situations occur in physics, chemistry, art, or the Rubik's cube [J1], among many other things. When we studied the Rubik's cube in the previous chapter, recall that one of the criteria was that each move was "invertible". Another was that the composition of two moves, one after the other, was itself a move. These are, in fact, two of the conditions for a group. A group is a set G with a **binary operation** (namely a function $* : G \times G \to G$) satisfying certain properties to be given later, one of which is that each element has an inverse element associated to it. Before giving the formal definition of a group in §5.4, we will first provide examples of groups that we have seen in earlier chapters.

Just as for sets, we must decide on how to describe a group. If G is finite then one way is to list all the elements in G and list (or tabulate) all the values of the function $*$. Another method is to describe G in terms of some properties and then define a binary operation $*$ on G. A third method is to give a "presentation" of G. Each of these has its advantages and disadvantages. We shall eventually introduce all three of these approaches.

First, we start with some example.

5.1 Cyclic groups

You've seen cyclic groups before, they were just disguised as "clock arithmetic".

Example 5.1.1. *Let C_{12} be the group whose elements are $\{0, 1, ..., 11\}$ and for which the group operation is simply "addition mod 12", just as one adds time on a clock (except that we call "12 o'clock" "0 o'clock"). Thus $5+8 = 1$, $1 + 11 = 0$, and so on.*

Questions: What is the element (the "inverse of 5") of C_{12} which, when added to 5, gives 0? What is the inverse of 1?

This group is called the (additive) **cyclic group of order 12**.

Definition 5.1.2. *Let $n > 1$ be an integer and let C_n be the group whose elements are $\{0, 1, ..., n-1\}$ (more precisely, $\{\overline{0}, \overline{1}, ..., \overline{n-1}\}$, where $\overline{i}$ is the residue class mod n of i) and for which the group operation is simply "addition modulo n". This group is called the (additive)* **cyclic group of order** n.

We've encountered cyclic groups before in this book - when discussing the discrete log problem in §1.8.5. In the additive case, the discrete log problem boiled down to the following: given $a, b \in C_n$, find $x \in C_n$ (if it exists) such that $ax = b$. As we know, x exists if $gcd(a, n) = 1$ and, in that case, $x = a^{-1}b$. Here $a^{-1} \in C_n$ is the (multiplicative) inverse of a mod n. So, we've already solved the additive case of the discrete log problem in the first chapter of this book.

The multiplicitive case of the discrete log problem is much, much harder. If you can find a "fast" efficient solution to it you'll be famous! For more details, we refer to the book by Crandell and Pomerance [CP].

Exercise 5.1.3. *Solve the following problems for x (or state DNE if x does not exist).*

(a) $5x = 7$ in C_{12},

(b) $5x = 6$ in C_{12},

(c) $4x = 7$ in C_{12} (hint: what happens when you multiply each side by 3?),

(d) $8x = 12$ in C_{12} (see hint above),

(e) $x^5 = 7$ in C_{12},

(f) $5^x = 7$ in C_{12}.

5.2 The dihedral group

Pick an integer $n > 2$ and let R be a regular n-gon centered about the origin in the plane. If $n = 3$ then R is an equilateral triangle, if $n = 4$ then R is

a square, if $n = 5$ then R is a pentagon, and so on. Let G denote the set of all linear transformations of the plane [1] to itself which preserve the figure R. The binary operation $\circ : G \times G \to G$ given by composition of functions gives G the structure of a group. This group is called the **group of symmetries of** R.

Label the vertices of the n-gon as 1, 2, ..., n. The group G permutes these vertices amongst themselves, hence each $g \in G$ may be regarded as a permutation of the set of vertices $V = \{1, 2, ..., n\}$. In this way, we may regard G as a permutation group since it is the subgroup of S_n generated by the elements of G.

The fact that this group has $2n$ elements follows from a simple counting argument: Let $r \in G$ denote the element which rotates R by $2\pi/n$ radians counterclockwise about the center. Let L be a line of symmetry of R which bisects the figure into two halves. Let s denote the element of G which is reflection about L. There are n rotations by a multiple of $2\pi/n$ radians about the center in G: $1, r, r^2, ..., r^{n-1}$. There are n elements of G which are composed of a reflection about L and a rotations by a multiple of $2\pi/n$ radians about the center: $s, s \circ r, s \circ r^2, ..., s \circ r^{n-1}$. These comprise all the elements of G.

One remarkable property of this symmetry group, which we shall use in the example in the next section, is that it is generated by any two distinct reflections in the group:

Lemma 5.2.1. *Pick two distinct lines L, L' of symmetries of R, each of which bisects R in half, and let s, s' (resp.) denote the corresponding reflections, regarded as elements in S_n. Then $G = \langle s, s' \rangle$.*

The interested reader is referred to [NST], [R], or [Ar], chapter 5, §3, for a proof.

The symmetry group of R is known as the **dihedral group of order** $2n$, denoted D_{2n}.

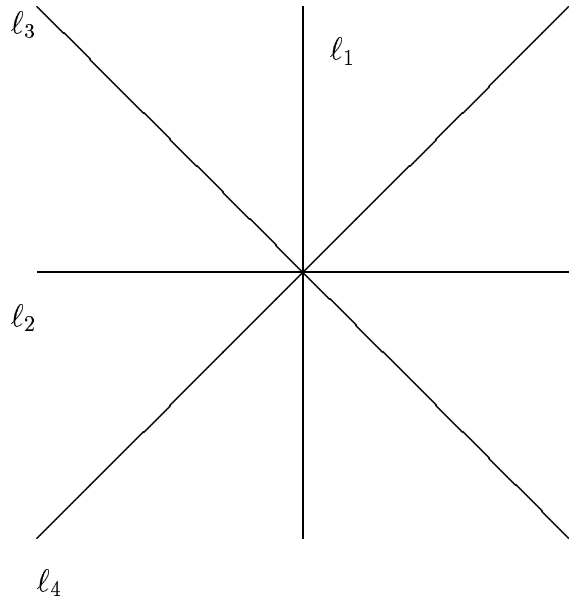
Example 5.2.2. *Let G be the symmetry group of the square: i.e., the group*

[1] If we regard R as a figure in 3-space centered above the origin and let G denote the set of all linear transformations *of* 3-*space* then we obtain a slightly larger group in some cases [NST].

of symmetries of the square generated by the rigid motions

$$\begin{aligned} g_0 &= 90 \text{ degrees clockwise rotation about O,} \\ g_1 &= \text{reflection about } \ell_1, \\ g_2 &= \text{reflection about } \ell_2, \\ g_3 &= \text{reflection about } \ell_3, \\ g_4 &= \text{reflection about } \ell_4, \end{aligned}$$

where ℓ_1, ℓ_2, ℓ_3 denote the lines of symmetry in the picture below:



The elements of G are

$$1, g_0, g_0^2, g_0^3, g_1, g_2, g_3, g_4.$$

Let X be the set of vertices of the square. Then G acts on X.

5.3 The symmetric group

Before defining anything, we shall provide a little motivation for some general notions which will arise later.

Let X be any finite set and let S_X denote the set of all permutations of X onto itself:

$$S_X = \{f : X \to X \mid f \text{ is a bijection}\}.$$

This set has the following properties:

1. if f, g belong to S_X then fg (the composition of these permutations) also belongs to S_X ("closed under compositions"),
2. if f, g, h all belong to S_X then $(fg)h = f(gh)$ ("associativity"),
3. the identity permutation $I : X \to X$ belongs to S_X ("existence of the identity"),
4. if f belongs to S_X then the inverse permutation f^{-1} also belongs to S_X ("existence of the inverse").

The set S_X is called the **symmetric group of** X. We shall usually take for the set X a set of the form $\{1, 2, ..., n\}$, in which case we shall denote the symmetric group by S_n. Note that the elements of S_n are exactly the permutations first introduced in the previous chapter. This group is also called the **symmetric group on** n **letters**.

Example 5.3.1. *Suppose* $X = \{1, 2, 3\}$. *We can describe* S_X *in disjoint cycle notation as*

$$S_X = \{I, s_1 = (1,\ 2), s_2 = (2,\ 3), s_3 = (1,\ 3,\ 2), s_4 = (1,\ 2,\ 3), s_5 = (1,\ 3)\}.$$

We can compute all possible products of two elements of the group and tabulate them:

	I	s_1	s_2	s_3	s_4	s_5
I	I	s_1	s_2	s_3	s_4	s_5
s_1	s_1	I	s_3	s_2	s_5	s_4
s_2	s_2	s_4	I	s_5	s_1	s_3
s_3	s_3	s_5	s_1	s_4	I	s_2
s_4	s_4	s_2	s_5	I	s_3	s_1
s_5	s_5	s_3	s_4	s_1	s_2	I

Exercise 5.3.2. *Verify the four properties of* S_X *mentioned above for Example 5.3.1. (Note that the verification of associativity follows from the associative property of the composition of functions - see Exercise 4.1.15).*

5.4 General definitions

We take the above four properties of the symmetric group as the four defining properties of a group:

Definition 5.4.1. *Let G be a set and suppose that there is a mapping*

$$\begin{aligned} * : G \times G &\longrightarrow G \\ (g_1, g_2) &\longmapsto g_1 * g_2 \end{aligned}$$

(called the group's **operation***) satisfying*

(G1) if g_1, g_2 belong to G then $g_1 * g_2$ belongs to G ("G is closed under $*$"),

(G2) if g_1, g_2, g_3 belong to G then $(g_1 * g_2) * g_3 = g_1 * (g_2 * g_3)$ ("associativity"),

(G3) there is an element $1 \in G$ such that $1 * g = g * 1 = g$ for all $g \in G$ ("existence of an identity"),

(G4) if g belongs to G then there is an element $g^{-1} \in G$, called the **inverse of** *g such that $g * g^{-1} = g^{-1} * g = 1$ ("existence of inverse").*

Then G (along with the operation $*$) is a **group**.

In the above definition, we have not assumed that there was exactly one identity element 1 of G because, in fact, one can show that if there is one then it is unique. (To do this you can use the **cancellation law**: if $a * c = b * c$, where $a, b, c \in G$, then $a = b$.) Likewise, if G is a group and $g \in G$ then the inverse element of g is unique. (Prove it!) There are other properties of a group which can be derived from (G1)-(G4). We shall prove them as needed.

The **multiplication table** [2] of a finite group G is a tabulation of the values of the binary operation $*$. This is also called the Cayley table, after the mathematician Arthur Cayley (1821-1895) who first introduced it in 1854 (along with the definition of an abstract group, as in Definition 5.4.1). Let $G = \{g_1, ..., g_n\}$. The multiplication table of G is:

[2] This terminology is inappropriate when G is given additively; in that case, it is called the **addition table**.

$*$	g_1	g_2	$\cdots$	g_j	$\cdots$	g_n
g_1						
g_2						
$\vdots$						
g_i				$g_i * g_j$		
$\vdots$						
g_n						

Some properties:

Lemma 5.4.2. *(a) Each element $g_k \in G$ occurs exactly once in each row of the table.*

(b) Each element $g_k \in G$ occurs exactly once in each column of the table.

(c) If the $(i,j)^{th}$ entry of the table is equal to the $(j,i)^{th}$ entry then $g_i * g_j = g_j * g_i$.

(d) If the table is symmetric about the diagonal then $g * h = h * g$ for all $g, h \in G$. (In this case, we call G **abelian**.*)*

Exercise 5.4.3. *Compute the multiplication table for C_3.*

Exercise 5.4.4. *Compute the multiplication table for*

$$\{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}\}.$$

Exercise 5.4.5. *Compute the multiplication table for $C_2 \times C_2$ (multiplication is componentwise).*

Exercise 5.4.6. *Compute the addition table for $\mathbb{Z}/4\mathbb{Z}$ $(= C_4)$.*

Exercise 5.4.7. *Consider the mappings from $\mathbb{C} - \{0,1\}$ (all complex numbers except 0 and 1) to itself given by $f_1(x) = x$, $f_2(x) = 1/x$, $f_3(x) = 1 - x$, $f_4(x) = f_3(f_2(x))$, $f_5(x) = f_2(f_3(x))$, $f_6(x) = f_2(f_4(x))$. Let the group operation on $G = \{f_1, f_2, f_3, f_4, f_5, f_6\}$ be composition of functions. Show G is a group. Write the Cayley table for G.*

5.5 The general linear group

Let F be a field. The fist few sections provide background on matrix terminology and how to multiply matrices.

5.5.1 $m \times n$ matrices

An $m \times n$ **matrix** ("over F") is a rectangular array or table of elements of F arranged with m rows and n columns. It is usually written:

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & & & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix}.$$

The $(i,j)^{th}$ **entry** of A is a_{ij}. The i**th row of** A is

$$\begin{bmatrix} a_{i1} & a_{i2} & \cdots & a_{in} \end{bmatrix} \qquad (1 \leq i \leq m)$$

The j**th column of** A is

$$\begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{bmatrix} \qquad (1 \leq j \leq n)$$

A matrix having as many rows as it has columns ($m = n$) is called a **square matrix**. The entries a_{ii} of an $m \times n$ matrix $A = (a_{ij})$ are called the **diagonal entries**, the entries a_{ij} with $i > j$ are called the **lower diagonal entries**, and the entries a_{ij} with $i < j$ are called the **upper diagonal entries**. An $m \times n$ matrix $A = (a_{ij})$ all of whose lower diagonal entries are zero is called an **upper triangular matrix**. This terminology is logical if the matrix is a square matrix but both the matrices below are called upper triangular

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 5 & 6 & 7 \\ 0 & 0 & 8 & 9 \\ 0 & 0 & 0 & 10 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 5 & 6 & 7 \\ 0 & 0 & 8 & 9 \\ 0 & 0 & 0 & 10 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

whether they look triangular of not! A similar definition holds for lower triangular matrices. The square $n \times n$ matrix with 1's on the diagonal and

0's elsewhere,

$$\begin{pmatrix} 1 & & \dots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & & 0 \\ 0 & \vdots & 0 & 1 \end{pmatrix},$$

is called the $n \times n$ **identity matrix** and denoted I or I_n. This is both upper triangular and lower triangular. (In general, any square matrix which is both upper triangular and lower triangular is called a **diagonal matrix**.)

A square $n \times n$ matrix with exactly one 1 in each row and each column, and 0's elsewhere, is called an $n \times n$ **permutation matrix**. The identity I_n is a permutation matrix. We shall discuss these types of matrices in detail in the next chapter.

A square $n \times n$ matrix with exactly one *non-zero entry* in each row and each column, and 0's elsewhere, is called an $n \times n$ **monomial matrix**. We shall discuss these types of matrices later in the book. They have many properties similar to permutation matrices. Monomial matrices occur in the explicit description of the "isometry class of a linear code" (in §5.6 below).

5.5.2 Multiplication and inverses

Fortunately, we shall not be forced to deal in this book too much with computations of matrix multiplications of large matrices. *Roughly speaking*, we shall eventually show how each move of the Rubik's Cube can be expressed in terms of matrices (more precisely, as a pair of matrices - an 8×8 matrix corresponding to the movement of the 8 corners and a 12×12 matrix corresponding to the movement of the 12 edges). Therefore, a little bit of brief background on matrix multiplication is appropriate.

When you multiply an $m \times n$ matrix A by a $n \times p$ matrix B, you get an $m \times p$ matrix AB. The $(i,j)^{th}$ entry of AB is computed as follows:

1. Let $k = 1$ and $c_0 = 0$.

2. If $k = m$, you're done and $a_{ij} = c_m$. Otherwise proceed to the next step.

3. Take the k^{th} entry of the i^{th} row of A and multiply it by the k^{th} entry of the j^{th} row of B. Let $c_k = c_{k-1} + a_{ik}b_{kj}$.

4. Increment k by 1 and go to step 2.

In other words, multiply each element of row i in A with the corresponding entry of column j in B, add them up, and put the result in the (i,j) position of the array for AB. (For those who have had vector calculus, compute the "dot product" of the i^{th} row of A with the j^{th} column of A.)

In particular, if A is an $n \times n$ matrix and if if B is an $n \times 1$ matrix then both B and AB are **column vectors** in F^n and the above multiplication defines a rule which sends column vectors to column vectors. In other words, A defines a map

$$\begin{array}{rcl} A: & F^n \to & F^n, \\ A: & v \longmapsto & Av, \end{array} \tag{5.1}$$

where $v \in F^n$ is written as a column vector.

If A is a square $n \times n$ matrix and if there is a matrix B such that $AB = I_n$ then we call B the **inverse** matrix of A, denoted A^{-1}. If you think of A as a function $A : F^n \to F^n$ then A^{-1} is the inverse function. As a practical matter, if n is 'small' (say, $n \leq 3$) then matrix inverses can be computed by pencil and paper using known techniques (see for example [JN]). For most larger matrices, computers are needed.

5.5.3 Determinants

The determinant of a matrix A, denoted $\det(A)$, is only defined when A is a square matrix, i.e., the number of rows is equal to the number of columns. If A is an $n \times n$ matrix then A may be regarded as a function of F^n, sending vectors to points.

If $F = \mathbb{R}$ then it turns out the matrix A, regarded as a function $A : \mathbb{R}^n \to \mathbb{R}^n$, will send the unit hypercube $[0,1] \times ... \times [0,1]$ (n times) in $\mathbb{R}^n$ to a parallelepiped (the n-dimensional analog of a parallelogram). It is known that the *absolute value* of the determinant of A measures the volume of that parallelpiped. For example, $A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$ sends the vertices of the unit square $[0,1] \times [0,1] = \{(x,y) \mid 0 \leq x \leq 1,\ 0 \leq y \leq 1\}$ to the points $(0,0)$, $(2,1)$, $(1,2)$, $(3,3)$. These points bound a parallelogram having volume $\det(A) = 3$.

If $m = n = 2$ the determinant is easy to define:

$$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc.$$

The easiest *constructive* way to define the determinant of an arbitrary $n \times n$ matrix $A = (a_{ij})$ is to use the **Laplace cofactor expansion** (along the i^{th} row): for any $1 \leq i \leq n$, we have

$$\det(A) = \sum_{j=1}^{n} (-1)^{i+j} \det(A_{ij}), \tag{5.2}$$

where A_{ij} is the $(n-1) \times (n-1)$ 'submatrix of A' obtained by omitting all the entries in the i^{th} row or the j^{th} column. The matrix A_{ij} is called the (i,j)-**minor** of A and $(-1)^{i+j} \det(A_{ij})$ is called the (i,j)-**cofactor**.

The determinant has many remarkable properties. For example,

- You can factor an expression out of any row or column. For example,
$$\det \begin{pmatrix} ra & rb \\ c & d \end{pmatrix} = \det \begin{pmatrix} ra & b \\ rc & d \end{pmatrix} = r \det \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$
- The Lagrange expansion holds for any row or column. For example, the middle column expansion for a 3×3 matrix:
$$\det \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$
$$= -a_{12} \det \begin{pmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{pmatrix} + a_{22} \det \begin{pmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{pmatrix} - a_{23} \det \begin{pmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{pmatrix}.$$
- The determinant, as a function of its rows or columns, is additive. For example,
$$\det(\begin{pmatrix} a+a' & b \\ c+c' & d \end{pmatrix} = \det(\begin{pmatrix} a & b \\ c & d \end{pmatrix} + \det(\begin{pmatrix} a' & b \\ c' & d \end{pmatrix}.$$

For a proof, see any text on linear algebra (e.g., [JN]).

A square matrix A is **singular** if $\det(A) = 0$. Otherwise, A is called **non-singular** or **invertible**.

Example 5.5.1. *Taking $i = 2$,*

$$\det \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$
$$= (-1) \cdot 4 \cdot \det \begin{pmatrix} 2 & 3 \\ 8 & 9 \end{pmatrix} + (+1) \cdot 5 \cdot \det \begin{pmatrix} 1 & 3 \\ 7 & 9 \end{pmatrix} + (-1) \cdot 6 \cdot \det \begin{pmatrix} 1 & 2 \\ 7 & 8 \end{pmatrix}$$
$$= (-4)(18-24) + (5)(9-21) + (-6)(8-14) = 0.$$

This implies A is singular. Indeed, the parallelepiped generated by $(1,2,3)$, $(4,5,6)$, $(7,8,9)$, *must be flat (2-dimensional, hence have* 0 *volume) since* $(4,5,6) = (1,2,3) + (1,1,1)$ *and* $(7,8,9) = (1,2,3) + 2(1,1,1)$.

An important fact about singular matrices, and one that we will use later, is the following.

Lemma 5.5.2. *Suppose A is an $n \times n$ matrix with entries in F. The following are equivalent:*

- $\det(A) \neq 0$,
- *there is no non-zero vector v such that $Av = 0$, where* 0 *is the zero vector in F^n,*
- A^{-1} *exists.*

For a proof, see any text on linear algebra.
We end this section with one last key property of determinants.

Lemma 5.5.3. *If A, B are any two $n \times n$ matrices having entries in F then* $\det(AB) = \det(A)\det(B)$.

More details on all this material can be found in any text book on linear algebra.

5.5.4 The definition of $GL(n)$

Let F denote a field and let $GL(n, F)$ denote the set of all $n \times n$ matrices with entries in F having non-zero determinant. This is called the **general linear group** of degree n over F. Each element $g \in GL(n, F)$ defined a function $g : F^n \to F^n$ by (5.1).

Proposition 5.5.4. *$GL(n, F)$ is a group under ordinary matrix multiplication.*

proof: The identity matrix is the identity element. Associativity is a property of matrix multiplication. (Left to the reader as an exercise.) The set $GL(n, F)$ is closed under multiplication by Lemma 5.5.3. Inverses exist by Lemma 5.5.2. □

Let

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

be a 2×2 matrix with $a, b, c, d \in \mathbb{Z}$. Suppose that the determinant [3] of A is 1, $ad - bc = 1$. Thanks to Theorem 1.4.3, this forces $gcd(a, b) = 1$ (why?). Similarly, we must have $gcd(a, c) = 1$, $gcd(b, d) = 1$, and $gcd(c, d) = 1$. On other words, to each 2×2 integer matrix with determinant 1, is associated several pairs of integers with no common factor.

Conversely, if $a, b \in \mathbb{Z}$ have no common factor then by Theorem 1.4.3 there are $m, n \in \mathbb{Z}$ such that $am + bn = 1$. This means that the matrix

$$A = \begin{pmatrix} a & b \\ -n & m \end{pmatrix}$$

has determinant 1.

Exercise 5.5.5. *Let $SL(n, F)$ be the subset of all $g \in GL(n, F)$ such that $\det(g) = 1$. Show that $SL(n, F)$ is a group.*

Exercise 5.5.6. *Let $SL(2, \mathbb{Z})$ be the subset of all $g \in GL(2, \mathbb{Z})$ such that $\det(g) = 1$. Show that $SL(2, \mathbb{Z})$ is a group.*

Exercise 5.5.7. *Find two matrices of the form*

$$\begin{pmatrix} 5 & 4 \\ * & * \end{pmatrix}$$

having determinant 1.

Exercise 5.5.8. *Find a matrix of the form*

$$\begin{pmatrix} * & 7 \\ * & 9 \end{pmatrix}$$

having determinant 1.

Exercise 5.5.9. *Find a matrix of the form*

$$\begin{pmatrix} * & * \\ 11 & 13 \end{pmatrix}$$

having determinant 1.

[3] Matrices and determinants are defined more formally later in the chapter on groups.

Exercise 5.5.10. *Find a matrix of the form*

$$\begin{pmatrix} 15 & * \\ 11 & * \end{pmatrix}$$

having determinant 1.

Exercise 5.5.11. *Compute*

$$\det \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 0 & 0 & 9 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

using the Lagrange expansion

Exercise 5.5.12. *Let*

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

have determinant 1. *Show that there is a matrix*

$$B = \begin{pmatrix} a' & b' \\ c' & d' \end{pmatrix}$$

with $a', b', c', d' \in \mathbb{Z}$ *and having determinant* 1, *such that* AB *has the form*

$$\begin{pmatrix} x & y \\ 0 & 1 \end{pmatrix}$$

for some $x, y \in \mathbb{Z}$. *(Hint: Take* $a' = d$, $c' = -c$, *and use Theorem 1.4.3 to determine* b', d'.*)*

Exercise 5.5.13. *Imagine a chessboard in front of you. You can place at most* 8 *non-attacking rooks on the chessboard. (Rooks move only horizontally and vertically.) Now imagine you have done this and let* $A = (a_{ij})$ *be the* 8×8 *matrix of* 0*'s and* 1*'s (called a* $(0,1)$**-matrix**) *where* $a_{ij} = 1$ *if there is a rook on the square belonging to the* i^{th} *horizontal down and the* j^{th} *vertical from the left. Call such a matrix a* **rook matrix**. *If there are exactly 8* 1*'s*

in A then we shall call A a **full rook matrix**. *For example,*

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

is a full rook matrix. Show that

(a) a full rook matrix is an 8×8 permutation matrix,

(b) any full rook matrix is invertible,

(c) the product of any two rook matrices is a rook matrix.

5.6 Application: The automorphism group of a code

Let $C \subset \mathbb{F}_q^n$ be a code. It is not necessary at this point for C to be linear. Let

$$Aut(C) = \{A \in GL(n, \mathbb{F}_q) \mid Ac \in C, \text{ for all } c \in C\}.$$

This is called the **automorphism group of** C

Proposition 5.6.1. *$Aut(C)$ is a group under ordinary matrix multiplication.*

proof: The identity matrix is in $Aut(C)$. Associativity is an inherited property from $GL(n, \mathbb{F}_q)$. The set $Aut(C)$ is closed under multiplication since the defining property is preserved. The existence of inverses is inherited from $GL(n, \mathbb{F}_q)$. $\square$

Example 5.6.2. *Let C be the $[7, 4, 3]$ Hamming code over $\mathbb{F}_2$ having generator matrix*

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

The elements of C *are*

$$\begin{array}{c}(0,0,0,0,0,0,0),(1,0,1,0,0,0,1),\\(0,0,0,1,1,0,1),(0,1,1,0,1,0,0),\\(0,0,1,1,0,1,0),(1,1,0,1,0,0,0),\\(0,1,0,0,0,1,1),(1,0,0,0,1,1,0),\\(1,0,0,1,0,1,1),(1,1,1,0,0,1,0),\\(0,1,0,1,1,1,0),(0,1,1,1,0,0,1),\\(1,0,1,1,1,0,0),(0,0,1,0,1,1,1),\\(1,1,0,0,1,0,1),(1,1,1,1,1,1,1)\end{array}$$

The automorphism group is a group of order 168 *generated by*

$$\begin{pmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&1&0&0\\0&0&0&1&0&0&0\\0&0&1&0&0&0&0\\0&0&0&0&0&0&1\\0&0&0&0&0&1&0\end{pmatrix},\quad\begin{pmatrix}0&0&1&0&0&0&0\\0&1&0&0&0&0&0\\1&0&0&0&0&0&0\\0&0&0&0&1&0&0\\0&0&0&1&0&0&0\\0&0&0&0&0&1&0\\0&0&0&0&0&0&1\end{pmatrix},$$

$$\begin{pmatrix}1&0&0&0&0&0&0\\0&0&1&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&0&0&1\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\\0&0&0&1&0&0&0\end{pmatrix},\quad\begin{pmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&0&0&1\\0&0&0&1&0&0&0\\0&0&0&0&0&1&0\\0&0&0&0&1&0&0\\0&0&1&0&0&0&0\end{pmatrix}.$$

These are the permutation matrices associated with the coordinate permutations $(3,5)(6,7)$, $(1,3)(4,5)$, $(2,3)(4,7)$, *and* $(3,7)(5,6)$.

Let $Mon(n,\mathbb{F}_q)$ denote the group of of $n\times n$ monomial matrices with entries in $\mathbb{F}_q^\times$. These are the matrices which have exactly one non-zero element in each row and column.

Let $C, C'\subset\mathbb{F}_q^n$ be two linear codes. We say that $g\in Mon(n,\mathbb{F}_q)$ is an **isometry** between C and C' if

- $g: C\to C'$ is an isomorphism of vector spaces,

- for all $c_1, c_2 \in C$,
$$d(c_1, c_2) = d(g(c_1), g(c_2)).$$

In this case, we say C, C' are isometric (with respect to the Hamming metric).

Exercise 5.6.3. *Show that two codes are isometric if and only if they are equivalent.*

Exercise 5.6.4. *Find the automorphism group of the binary repetition code of length* 3*,*
$$C = \{(0,0,0), (1,1,1)\}.$$

5.7 Abelian groups

What's abelian and purple? An abelian grape!

Definition 5.7.1. *Let g and h be two elements of a group G. We say that g* **commutes** *with h (or that g, h* **commute***) if $g * h = h * g$. We call a group* **commutative** *(or* **abelian***) if every pair of elements g, h belonging to G commute. If G is a group which is not necessarily commutative then we call G* **noncommutative** *(or* **nonabelian***).*

Example 5.7.2. *The integers, with ordinary addition as the group operation, is an abelian group.*

Now the reader should understand the punchline to the joke quoted at the beginning!

Convention: When dealing with groups in general we often drop the $*$ and denote multiplication simply by juxtaposition (that is, sometimes we write gh in place of $g * h$ and g^n in place of $g * g * ... * g$ (n times)). Also, by convention, $g^0 = 1$, the identity element. However, if the group G is abelian then one often replaces $*$ by $+$, $g + g + ... + g$ by ng, and then $+$ is *not* dropped.

Some typical examples of finite abelian groups:

- the additive group $G = \mathbb{Z}/n\mathbb{Z}$ $(= C_n)$,

- the multiplicative group $G = (\mathbb{Z}/n\mathbb{Z})^\times$,

where n is an integer. We have

$$|\mathbb{Z}/n\mathbb{Z}| = n, \quad |(\mathbb{Z}/n\mathbb{Z})^\times| = \phi(n),$$

where $\phi(n)$ is Euler's ϕ-function.

Definition 5.7.3. *If G is a multiplicative group G with only one generator (i.e., there is a $g \in G$ such that $G = \{g^i \mid i = 0, 1, 2, ...\}$) then we say that G is* **cyclic**.

Lemma 5.7.4. *If*

$$G = \langle g \rangle = \{g^i \mid i \in \mathbb{Z}\},$$

is a finite cyclic group and $m > 1$ is the smallest integer such that $g^m = 1$ then $|G| = m$.

proof: Since $g^{m+k} = g^k$ for any $k \in \mathbb{Z}$, we can list all the elements of G as follows:

$$1, g, g^2, ..., g^{m-1}.$$

There are m elements in this list. □

We have seen (multiplicative) cyclic groups before in our discussion of the discrete log problem. The abstract form of the **discrete log problem** is the following: Given a finite cyclic group $G = \langle a \rangle$ with generator a and given $b \in G$, find $x \in \mathbb{Z}$ such that $a^x = b$.

Exercise 5.7.5. *Show that any group having exactly 2 elements is abelian.*

Exercise 5.7.6. *Write down all the elements of $G = (\mathbb{Z}/10\mathbb{Z})^\times$ and compute its multiplication table.*

Exercise 5.7.7. *Write down all the elements of $G = \mathbb{Z}/4\mathbb{Z}$ and compute it's addition table.*

Exercise 5.7.8. *Let $SL(2, \mathbb{Z})$ be the subset of all $g \in GL(2, \mathbb{Z})$ such that $\det(g) = 1$. Show that $SL(2, \mathbb{Z})$ is non-abelian.*

Exercise 5.7.9. *Show that if G is a group such that $g^2 = 1$ for all $g \in G$ then G is abelian.*

5.8 Permutation groups

Now that we know the definition of a group, the question arises: how might they be described? The simplest answer is that we describe a group much as we might describe a set: we could list all its elements and give the multiplication table or we could describe all its elements and their multiplication in terms of some property from which we can verify the four properties of group. Though the first way has the distinct advantage of being explicit, it is this second alternative which is the most common since it is usually more concise.

Our objective is to introduce terminology and techniques which enable us to analyse mathematically permutation puzzles. The type of groups which arise in this context are defined next.

Definition 5.8.1. *Let X be a finite set. Let $g_1, g_2, ..., g_n$ be a finite set of elements of permutations of X (so that they all belong to S_X). Let G be the set of all possible products of the form*

$$g = x_1 * x_2 ... * x_m, \qquad m > 0,$$

where each of the $x_1, ..., x_m$ is taken from the set $\{g_1, ..., g_n\}$. The set G, together with the group operation given by composition of permutations, is called a **permutation group** *with* **generators** *$g_1, ..., g_n$. We sometimes write*

$$G = \langle g_1, ..., g_n \rangle \subset S_X.$$

Example 5.8.2. *Let X be the set of 54 facets of the Rubik's cube and let $R, L, U, D, F, B \in S_X$ denote the basic moves of the Rubik's cube, in the notation introduced in the previous chapter. The permutation group $G = \langle R, L, U, D, F, B \rangle \subset S_X$ is called the* **Rubik's cube group**. *We shall determine the "structure" (i.e., its relationship with "known groups") of this group later in this book.*

.

It is not too hard to justify our terminology:

Lemma 5.8.3. *A permutation group is a group.*

proof: Let G be a permutation group as in the above definition.

We shall only prove that each $g \in G$ has an inverse, leaving the remainder of the properties for the reader to verify. The set $\{g^n \mid n \geq 1\} \subset S_X$ is finite.

There are $n_1 > 0$, $n_2 > n_1$ such that $g^{n_1} = g^{n_2}$. Then $g^{-1} = g^{n_2-n_1-1}$ since $g \cdot g^{n_2-n_1-1} = 1$. $\square$

Remark 5.8.4. *The above definition can be generalized: Replace S_X by any group S which includes all the generators $g_1, ..., g_n$. The resulting set G is called the* **group generated by the elements** $g_1, ..., g_n$.

Algorithm:
Input: The generators $g_1, ..., g_n$ (as permutations in S_X),
Output: The elements of G,
$S = \{g_1, ..., g_n, g_1^{-1}, ..., g_n^{-1}\}$,
$L = S \cup \{1\}$,

```
for g in S do
  for h in L do
   if g*h not in L then L = L union {g*h} endif
  endfor
endfor
```

Note that the size of the list L in the for loop changes after each iteration of the loop. The meaning of this is that the if-then command is to be executed exactly once for each element of the list.

Definition 5.8.5. *If G is a group then the* **order** *of G, denoted $|G|$, is the number of elements of G, if G is a finite set, and $|G| = \infty$, otherwise. If g is an element of the group G then the order of g, denoted $ord(g)$, is the smallest positive integer m such that $g^m = 1$, if it exists. If such an integer m does not exist then we say that g has* **infinite order**.

Example 5.8.6. *For example, there is an even permutation of order 42 in S_{12}, for example $(1,2)(3,4,5)(6,7,8,9,10,11,12)$, and an odd permutation of order 15 in S_8, for example $(1,2,3)(4,5,6,7,8)$.*

We shall be able to make use of the following fact frequently.

Theorem 5.8.7. *(a) (Cauchy) Let p be a prime dividing $|G|$. There is a $g \in G$ of order p.*

(b) (Lagrange) Let n be an integer not dividing $|G|$. There does not exist a $g \in G$ of order n.

Part (a) will be proven below (see Corollary 5.13.4) and part (b) is a corollary of Theorem 5.9.3 below.

As an application of this: we shall see later that the Rubik's cube group G has the property that $|G| = 2^{27}3^{14}5^3 7^2 11$. It follows from this and Lagrange's theorem that there is no move of the Rubik's cube of order 13 but there is one of order 11. Assuming this can you show taht there is no move of order 52?

Exercise 5.8.8. *Find the order of the following elements:*

(a) $\begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix}$ *in* $GL(2, \mathbb{C})$,

(b) $(1,2)(2,3)(3,4)$,

(c) $(1,2,3)(3,4,5)(5,6,7)$,

(d) $(1,2,3)(4,5,6)(7,8)$.

Exercise 5.8.9. *Let* $G = \langle g \rangle$ *and* $|G| = 12$. *Thus,* $G = \{g, g_2, g_3, ..., g_{12} = 1\}$. *Note that* $h = g^9$ *has order* 4 *and* h^k *isn't* 1 *if* $1 < k < 4$.

Exercise 5.8.10. *Let* $G = \langle g \rangle$ *and* $|G| = 15$. *Which elements of* G *have order* 15*? How many elements have order* 5*? Order* 3*?*

Exercise 5.8.11. *Let* $X = \{1, 2, 3\}$. *We use the notation of Example 5.3.1 above.*

(a) Let G *be the permutation group with generator* $s_1, G = \langle s_1 \rangle$. *Verify that there are only two elements in* G.

(b) What is the order of s_5*?*

(c) Let G *be the permutation group with generator* s_3, $G = \langle s_3 \rangle$. *Verify that there are only three elements in* G.

(d) Find the order of s_3.

(e) Show that $S_X = \langle s_1, s_2 \rangle$.

Exercise 5.8.12. *Find the orders of each of the elements in* S_4.

5.9 Subgroups

There are lots of subsets of a group G which still have the structure of a group. These objects are defined more precisely next.

Definition 5.9.1. *Let* G *be a group. A* **subgroup** *of* G *is a subset* H *of* G *such that* H*, together with the operation* $*$ *inherited as a subset of* G*, satisfies the group operations (G1)-(G4) (with* G *replaced by* H *everywhere).*

Notation: If G is a group then we will denote the statement "H is a subgroup of G" by

$$H < G.$$

Example 5.9.2. *For example, if* $G = S_4$, *and*

$$H = \langle (1,3),(1,4) \rangle = \{1,(1,3),(1,4),(1,3,4),(3,4),(1,4,3)\},$$

then $H < G$.

Theorem 5.9.3. *(Lagrange) Let* H *be a subgroup of a finite group* G. *Then* $|H|$ *divides* $|G|$.

proof: For $x, y \in G$, define $x \sim y$ if $xH = yH$, where

$$xH = \{x * h \mid h \in H\}.$$

This is an equivalence relation. (The interested reader can easily verify the reflexive, symmetry, and transitivity properties.) Moreover, the equivalence class of x consists of all elements in G of the form $x * h$, for some $h \in H$, i.e., $[x] = xH$. Let $g_1, ..., g_m \in G$ denote a complete set of representatives for the equivalence classes of G. Because of the cancellation law for groups, $|xH| = |H|$ for each $x \in G$. Furthermore, we know that the equivalence classes partition G, so

$$G = \cup_{i=1}^{m} [g_i] = \cup_{i=1}^{m} g_i H.$$

Comparing cardinalities of both sides, we obtain $|G| = |g_1 H| + ... + |g_m H| = m|H|$. This proves the theorem. □

Definition 5.9.4. *If* H *and* G *are finite groups and* $H < G$ *then the integer* $|G|/|H|$ *is called the* **index** *of* H *in* G, *denoted* $[G : H] = |G|/|H|$.

Example 5.9.5. *A permutation group* G *generated by elements* $g_1, ..., g_n$ *belonging to* S_X *is a subgroup of* S_X, *i.e.,* $G < S_X$.

Example 5.9.6. *Let*

$$A_X = \{g \in S_X \mid g \text{ is even}\}.$$

This is a subgroup of S_n *called the* **alternating subgroup of degree** n .

Definition 5.9.7. *The* **center** *of a group G is the subgroup $Z(G)$ of all elements which commute with every element of G:*

$$Z(G) = \{z \in G \mid z * g = g * z, \text{ for all } g \in G\}.$$

Of course, the identity element always belongs to G. If the identity element is the only element of $Z(G)$ then we say G has **trivial center**. On the other hand, G is commutative if and only if $G = Z(G)$.

Exercise 5.9.8. *For $H = \langle (1,3),(1,4) \rangle$ and $G = S_4$ as above, G is partitioned into four disjoint sets of the form gH. Here are two of them: $H(= (1,3)H = (1,4)H = (1,3,4)H = ...)$ and $(1,2,3)H(= (1,2)H = (1,2,3,4)H = ...)$. Find the other two.*

Exercise 5.9.9. *Let G be a group written multiplicatively and let H be a subset which is closed under multiplication and taking inverses. Show that H is a (sub)group.*

Exercise 5.9.10. *Show, as a corollary to the previous Theorem 5.9.3, that Theorem 5.8.7(b) is true.*

Exercise 5.9.11. *Let G be a group. Show $Z(G)$ is a group.*

Exercise 5.9.12. *Let $G = S_3$. Determine $Z(G)$.*

Exercise 5.9.13. *Let H_1, H_2 and H_3 be subgroups of a group G.*

(a) Must $H_1 \cup H_2$ be a subgroup of G? If so, prove why; if not, give an example.

(b) Must $H_1 \cap H_2$ be a subgroup of G? Again, prove or disprove.

(c) Must $H_1 \cap H_2 \cap H_3$ be a subgroup?

Exercise 5.9.14. *Let $G = S_5$. Determine $Z(G)$.*

Exercise 5.9.15. *The subset xH, $x \in G$, introduced in the proof of Lagrange's Theorem 5.9.3 is called a "left coset" of H in G. (These shall be studied in a later section in more detail.) Show that if xH is a subgroup of G then $x \in H$.*

5.10 Puzzling examples

Not all groups arising from puzzles must come from the Rubik's cube or some related puzzle. The next example illustrated a group arising from chess.

Example 5.10.1. *Consider an infinite chessboard which we imagine being placed on the Cartesian plane. Label one square as* $(0,0)$ *and call it the origin. Label the others* (m,n)*, as one would label the vertices in a lattice in the plane. Place only one chess piece, a king, at* $(0,0)$*. Label the move one square to the right* x*, one square to the left* x^{-1}*, the move one square forward* y*, one square backwards* y^{-1}*, and label the other moves* xy*,* $x^{-1}y$*,* $x^{-1}y^{-1}$*, and* xy^{-1}*, in the obvious way. The set of all possible kings moves may be identifies with the set*

$$King := \{x^m y^n \mid m, n \in \mathbb{Z}\}.$$

Note King is an infinite abelian group under multiplication. This group can be identified with $\mathbb{Z} \times \mathbb{Z}$ *(where the group operation is componentwise addition). The number of ways that the king can reach the square* (m,n) *in* N *moves is the coefficient of* $x^m y^n$ *in the expansion of*

$$(x + x^{-1} + y + y^{-1} + xy + x^{-1}y + x^{-1}y^{-1} + xy^{-1})^N.$$

For example, using this in a computer algebra package (such as MAGMA or GAP or Maple or Mathematica), it is easy to see that there are 19246920 *ways to reach* $(1,1)$ *from the origin in* 10 *moves.*

Further details can be found in the chapter "Wanderungen von Schachfiguren" by K. Fabel in [BFR].

Example 5.10.2. *The collection of all moves of the Rubik's cube may be viewed as a subgroup* G *of* S_{48}*. The center of* G *consists of exactly two elements, the identity and the* **superflip** *move which has the effect of flipping (i.e., rotating by* 180^o*) every edge subcube, leaving all the corners alone and leaving all the subcubes in their original position. One move for the superflip is*

$$\begin{aligned}\text{superflip} &= R*L*F*B*U*D*R*L*F*B*U*F^2*M_R* \\ &\quad *F^2*U^{-1}*M_R^2*B^2*M_R^{-1}*B^2*U*M_R^2*D \\ &= R*L*F*B*U*D*R*L*F*B*U*F^2*R^{-1}* \\ &\quad *L*D^2*F^{-1}*R^2*L^2*D^2*R*L^{-1}* \\ &\quad *F^2*D*R^2*L^2*D \quad (34 \text{ quarterturns}),\end{aligned}$$

where M_R is middle right slice move (rotation by 90 degrees of the middle slice, keeping the right face and left face fixed, as viewed from the right face). Other expressions for this move are Dik T. Winter's move

$$\text{superflip} = F*B*U^2*R*F^2*R^2*B^2*U^{-1}*D*F*U^2*R^{-1}*L^{-1}*U* \\ *B^2*D*R^2*U*B^2*U \quad (28 \text{ quarterturns}),$$

and Mike Reid's move (found with a computer search)

$$\text{superflip} = R^{-1}*U^2*B*L^{-1}*F*U^{-1}*B*D*F*U*D^{-1}*L*D^2*F^{-1}* \\ *R*B^{-1}*D*F^{-1}*U^{-1}*B^{-1}*U*D^{-1} \quad (24 \text{ quarterturns}).$$

Jerry Bryan (in a Feb 19, 1995 posting to the cube-lover's email list, [CL]) showed that no fewer number of quarter turn moves taken from

$$\{R, R^{-1}, L, L^{-1}, U, U^{-1}, D, D^{-1}, B, B^{-1}, F, F^{-1}\},$$

that will also give this move. In the jargon, this move is 'minimal in the quarter-turn metric'. The proof that $Z(G) = \{1, superflip\}$ uses the determination of the group structure of G given later (see also [B]).

By the way, there is a "longer" element of the Rubik's cube group. The superflip composed with the four-spot

$$\text{superflip4spot} = U^2*D^2*L*F^2*U^{-1}*D*R^2*B*U^{-1}*D^{-1}*R* \\ *L*F^2*R*U*D^{-1}*R^{-1}*L*U*F^{-1}*B^{-1}. \quad (26 \text{ quarterturns}).$$

This was also proven by Mike Reid to be minimal.

5.10.1 Example: The Verhoeff check digit scheme

We have seen in §3.3.2 the ISBN check digit scheme. This helps detect an error made in one of the digits. In 1969, J. Verhoeff [V] describes a check digit scheme using the dihedral group.

Let $G = D_{10}$. This group has 10 elements, each element corresponding to the digits $0, 1, ..., 9$. For example, if $D_{10} = \{g[0], g[1], ..., g[9]\}$ are the elements of G in some order (it will be typographically simpler, as you will see, to use $g[i]$ instead of g_i for our notation) then we can associate i to g_i. We let

$$D_{10} = \{1, (2,5)(3,4), (1,2)(3,5), (1,2,3,4,5), (1,3)(4,5), (1,3,5,2,4), \\ (1,4)(2,3), (1,4,2,5,3), (1,5,4,3,2), (1,5)(2,4)\},$$

so for example $g[0] = 1$.

Verhoeff check digit scheme ([Ki], §5.4) Fix an integer $n > 1$ and fix an element σ in the symmetric group of the set $\{0, 1, ..., 9\}$. Let $d_1 d_2 ... d_n$ be a identification number without a check digit. The digit d_{n+1} is appended to $d_1 ... d_n$ provided

$$g[\sigma(d_1)]g[\sigma^2(d_2)]...g[\sigma^{n-1}(d_{n-1})]g[\sigma^n(d_n)]g[d_{n+1}] = 1.$$

(Actually, Verhoeff uses decending powers of σ in his scheme. We use ascending powers since it is easier to remember and because the German Bundesbank used this version for the Deutsche Mark - see [Ki] for more details.)

Example 5.10.3. *Let* $\sigma = (1, 7, 9)(2, 5, 10, 4, 6)$ *and let* $n = 5$.

Consider the ID number 12345*, what is the check digit? We compute* $g[\sigma(1)] = (1, 2)(3, 5)$, $g[\sigma^2(2)] = (1, 2, 3, 4, 5)$, $g[\sigma^3(3)] = (1, 2)(3, 5)$, $g[\sigma^4(4)] = (2, 5)(3, 4)$, $g[\sigma^5(5)] = (1, 3, 5, 2, 4)$. *Thus* d_6 *is determined by*

$$(1, 2)(3, 5)(1, 2, 3, 4, 5)(1, 2)(3, 5)(2, 5)(3, 4)(1, 3, 5, 2, 4)g[d_6] = 1.$$

Since $((1, 2)(3, 5)(1, 2, 3, 4, 5)(1, 2)(3, 5)(2, 5)(3, 4)(1, 3, 5, 2, 4))^{-1} = (1, 4)(2, 4) = g[7]$*, we have* $d_6 = 7$. *The ID number with check digit is* 123457.

Consider the ID number 75872*, what is the check digit? Answer:* $d_6 = 5$. *The ID number with check digit is* 758725. *We leave it as an exercise to check this (see Exercise 5.10.4).*

5.10.2 Example: The two squares group

Let $H = \langle R^2, U^2 \rangle$ denote the group generated by the two square moves, R^2 and U^2 or the Rubik's cube. (The reader with a cube in hand may want to try the **Singmaster magic grip** : the thumb and forefinger of the right hand are placed on the front and back face of the fr, br edge, the thumb and forefinger of the left hand are placed on the front and back face of the uf, ub edge; all moves in this group can be made without taking your fingers off the cube.) This group contains the useful 2-pair edge swap move $(R^2 * U^2)^3$.

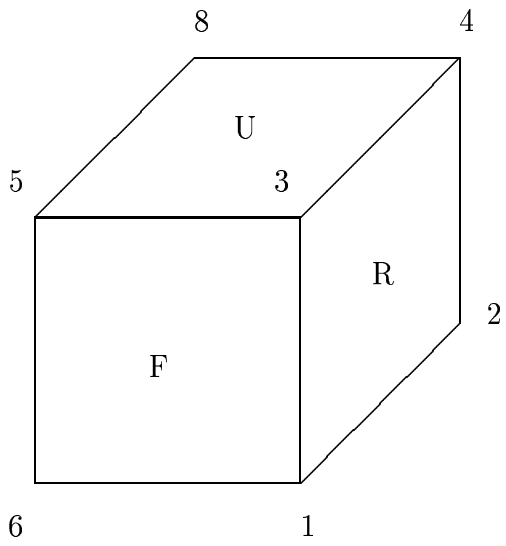
We can find all the elements in this group fairly easily:

$$H = \{1, R^2, R^2 * U^2, R^2 * U^2 * R^2, (R^2 * U^2)^2, (R^2 * U^2)^2 * R^2, (R^2 * U^2)^3,$$
$$(R^2 * U^2)^3 * R^2, (R^2 * U^2)^4, (R^2 * U^2)^4 * R^2, (R^2 * U^2)^5, (R^2 * U^2)^5 * R^2\},$$

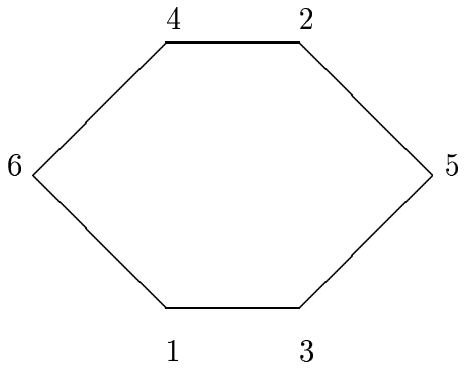
Therefore, $|H| = 12$. Note that $1 = (R^2 * U^2)^6$, $U^2 = (R^2 * U^2)^5 * R^2$, and $U^2 * R^2 = (R^2 * U^2)^5$. (By the way, this listing without repetition of H by

successive multiplication by R^2 then U^2 may be reformulated graphically by saying the "the Cayley graph of H with generators R^2, U^2 has a Hamiltonian circuit".)

To discover more about this group, we label the vertices of the cube as follows:



The move R^2 acts on the set of vertices by the permutation $(1,\ 4)(2,\ 3)$ and the move U^2 acts on the set of vertices by the permutation $(4,\ 5)(3,\ 6)$. We label the vertices of a hexagon as follows:



The permutation $(1,\ 4)(2,\ 3)$ is simply the reflection about the line of symmetry containing both 5 and 6. The permutation $(4,\ 5)(3,\ 6)$ is simply the reflection about the line of symmetry containing both 1 and 2. By a fact stated in section 5.2, these two reflections generate the symmetry group of the hexagon.

Exercise 5.10.4. *Verify the check digit claimed in Example 5.10.3.*

Exercise 5.10.5. *Work out a similar result to that in Example 5.10.1 using a knight in place of a king.*

Exercise 5.10.6. *We refer to Example 5.10.1. In how many ways can a king starting at* $(0,0)$ *reach* $(1,1)$ *in* 5 *moves?*

Exercise 5.10.7. *The set of all possible knight moves also naturally forms a group which is a subgroup of King:*

$$Knight := \{x^m y^n \mid m, n \in \mathbb{Z}, |m| + |n| \equiv 0 \pmod 3\},$$

(where the group operation is again multiplication). In how many ways can knight starting from $(0,0)$ *reach*

(a) $(1,1)$ *in* 5 *moves?*

(b) $(2,1)$ *in* 5 *moves? [Hint:* $(x^2y + x^{-2}y + ... + xy^{-2})^5 = ?$*]*

Exercise 5.10.8. *We refer to Example 5.10.1 and Exercise 5.10.5. How would the group Bishop be defined? Is there a subgroup Pawn? Rook? Queen?*

Exercise 5.10.9. *Consider a square centered at the origin having sides parallel to the coordinate axes, superimposed on the figure in Example 5.2.2. Label the four vertices of this square starting with the upper left-hand vertex and moving around in a clockwise fashion,* $1, 2, 3, 4$. *Compute the symmetry group in Example 5.2.2 as a subgroup of* S_4. *In other words, write down each of the elements of* G *explicitly in disjoint cycle notation.*

5.11 Commutators

Definition 5.11.1. *If* g, h *are two elements of a group* G *then we call the element*

$$[g, h] = g * h * g^{-1} * h^{-1}$$

then **commutator** *of* g, h.

Not that $[g, h] = 1$ if and only if g, h commute. Thus the commutator may be regarded as a rough measurement of the lack of commutativity.

Example 5.11.2. *The commutator of* $(1, 2, 3)$ *with* $(4, 5)$ *in* S_5 *is* 1, *since these elements are disjoint cycles hence commute.*

The commutator of $(1, 2, 3)$ *with* $(3, 4)$ *in* S_5 *is*

$$[(1, 2, 3), (3, 4)] = (1, 2, 3)(3, 4)(1, 3, 2)(3, 4) = (2, 3, 4).$$

Example 5.11.3. *David Singmaster [Si] introduced the following notation for the Rubik's cube. Let* G *be the permutation group generated by the permutations* R, L, U, D, F, B *regarded as permutations in* S_{54} *(i.e., the Rubik's cube group). The* **Y commutator** *is the element*

$$[F, R^{-1}] = F * R^{-1} * F^{-1} * R.$$

The **Z commutator** *is the element*

$$[F, R] = F * R * F^{-1} * R^{-1}.$$

As an exercise for those whose own a Rubik's cube, explain why these moves have their name!

If x, y *are basic moves of the Rubik's cube associated to faces which share an edge then*

(a) $[x, y]^2$ *permutes exactly 3 edges and does not permute any corners,*

(b) $[x, y]^3$ *permutes exactly 2 pairs of corners and does not permute any edges.*

Definition 5.11.4. *Let G be any group. The group G' generated by all the commutators*

$$\{[g,h] \mid g,h \text{ belong to } G\}$$

This is called the **commutator subgroup** *of G.*

This group may be regarded as a rough measurement of the lack of commutativity of the group G.

Remark 5.11.5. *The group generated by the basic moves of the Rubik's cube - R, L, U, D, F, B - has a relatively large commutator subgroup [J1]. In other words, roughly speaking "most" moves of the Rubik's cube can be generated by commutators such as the Y commutator or the Z commutator.*

Exercise 5.11.6. *Let $G = S_3$, the symmetric group on 3 letters. Compute the commutators*

$$[s_1, s_2], \quad [s_2, s_1].$$

Exercise 5.11.7. *Compute the commutator of (a,b) with (b,c), where a, b, c are distinct.*

Exercise 5.11.8. *If you have a Rubik's cube:*

(a) Find the orders of the Y commutator and the Z commutator.

(b) Find the order of $[R,[F,U]]$.

(c) Let R, U be as in the notation for the Rubik's cube moves introduced in the previous chapter. Determine the order of the move $[R,U]$.

5.12 Conjugation

Definition 5.12.1. *: If g, h are two elements of a group G then we call the element*

$$g^h = h^{-1} * g * h$$

the **conjugation** *of g by h.*

Note that $g^h = g$ if and only if g, h commute. Thus the conjugates may be regarded as a rough measurement of the lack of commutativity.

Definition 5.12.2. *: We say two elements g_1, g_2 of G are* **conjugate** *if there is an element $h \in G$ such that $g_2 = g_1^h$.*

For those who have had some linear algebra, it may be helpful to think of the following. If G is a group given as a collection of matrices (i.e., as a subgroup of $GL(n,\mathbb{R})$), then g_1 and g_2 are conjugate if they represent the same linear transformation - but with respect to two different bases. If $g_2 = g_1^h$ then the two bases are: the standard basis and the basis given by the columns of h. (For a proof of this, see almost any text in linear algebra, for example [JN].) We will illustrate the possible advantage of thinking this way in a moment.

It turns out that it is easy to see when two permutations $g_1, g_2 \in S_n$ are conjugate: they are conjugate if and only if the cycles in their respective disjoint cycle decompositions have the same length when arranged from shortest to longest. (The proof of this is left as an exercise.) For example, the elements

$$g_1 = (6,9)(1,3,4)(2,5,7,8), \qquad g_2 = (1,2)(3,4,5)(6,7,8,9)$$

are conjugate. Our comment above helps understand intuitively why g_1 and g_2 are conjugate: g_1 and g_2 are conjugate because the associated permutation matrix $P(g_2)$ "represents the same operation as $P(g_1)$", but with respect to a different basis. **Notation**: The set of equivalence classes of G under the equivalence relation given by conjugation, will be denoted G_*.

In [Si], §5.10D, D. Singmaster asks for the possible orders of the elements of the Rubik's cube group and how many elements of each order there are. This question of Singmaster motivates the following:

Problem: Determine $p_G(t)$ for the Rubik's cube group.

Example 5.12.3. *For S_8, the generating polynomial is*

$$t + 4t^2 + 2t^3 + 4t^4 + t^5 + 5t^6 + t^7 + t^8 + t^{10} + t^{12} + t^{15}$$

and for S_{12} it is

$$\begin{gathered} t + 6t^2 + 4t^3 + 9t^4 + 2t^5 + 16t^6 + t^7 + 4t^8 + 2t^9 + 6t^{10} + \\ t^{11} + 9t^{12} + 2t^{14} + 2t^{15} + t^{18} + 2t^{20} + t^{21} + \\ t^{24} + t^{28} + 3t^{30} + t^{35} + t^{42} + t^{60}. \end{gathered}$$

(Both of these calculations were performed by MAPLE.) For example, it follows that there is an even permutation of order 42 in S_{12} and an odd permutation of order 15 in S_8.

Singmaster [Si] states that the maximal order in the Rubik's cube group is 1260. J. Butler found the following move of this order: $m = RU^2D^{-1}BD^{-1}$ *(see [B], page 51, for another simple move of order* 1260*). So, even though the size of the Rubik's cube group is enormous, about* 4.3×10^{19}*, there are no moves of order more than* 1.3×10^3*.*

Definition 5.12.4. *: Fix an element* g *in a group* G*. The set*

$$Cl(g) = Cl_G(g) = \{h^{-1} * g * h \mid h \in G\}$$

is called the **conjugacy class of** g **in** G*. It is the equivalence class of the element* g *under the relation given by conjugation.*

Note that if $g_1 \in G$ is conjugate to $g_2 \in G$ then $Cl(g_1) = Cl(g_2)$.
The polynomial

$$p_G(t) = \sum_{g \in G_*} t^{ord(g)},$$

is called the **generating polynomial** of the order function on G.

Theorem 5.12.5. *Any finite group may be partitioned into its distinct conjugacy classes,*

$$G = \cup_{g \in G_*} Cl(g).$$

If H is a subgroup of G and if g is a fixed element of G then the set

$$H^g = \{g^{-1} * h * g \mid h \in H\}$$

is a subgroup of G. Such a subgroup of G is called a subgroup **conjugate** to H.

Exercise 5.12.6. *Let* $G = S_3$*, the symmetric group on 3 letters, in the notation of the example above. Compute the conjugations*

$$s_1^{s_2}, \qquad s_2^{s_1}.$$

Exercise 5.12.7. *Let* $s = (1, 2, 5, 3)$ *and* $r = (1, 2, 3)(4, 5)$ *be elements of* S_5*. Compute* r^1sr *using Lemma 4.4.1.*

Exercise 5.12.8. *Find all elements in* S_5 *that are conjugate to the permutation* $g = (1, 2, 3)(4, 5)$*.*

Exercise 5.12.9. *Find all elements in S_4 that are conjugate to the permutation $g = (1, 2, 3, 4)$.*

Exercise 5.12.10. *The elements*

$$g_1 = (6, 9)(1, 3, 4)(2, 5, 7, 8), \qquad g_2 = (1, 2)(3, 4, 5)(6, 7, 8, 9)$$

are conjugate. In other words, there is an $h \in S_9$ such that $g_2 = g_1^h$. Find h.

Exercise 5.12.11. *Let R, U be as in the notation for the Rubik's cube moves introduced in the previous chapter. Determine the order of the move R^U.*

Exercise 5.12.12. *Let S be the set of all subgroups of G. We define a relation R on S by*

$$R = \{(H_1, H_2) \in S \times S \mid H_1 \text{ is conjugate to } H_2\}.$$

Show that R is an equivalence relation.

Exercise 5.12.13. *Let $G = S_n$ and let $H = \langle g \rangle$ be a cyclic subgroup generated by a permutation g of the set $\{1, 2, ..., n\}$. With respect to the equivalence relation in the previous problem, show that a subgroup K of G belongs to the equivalence class $[H]$ of H in G if any only if K is cyclic and is generated by an element k of G conjugate to $g \in G$.*

Exercise 5.12.14. *Show that the notion of conjugate defines an equivalence relation. That is, show that*

*(a) any element $g \in G$ is conjugate to itself (***reflexive***),*

*(b) if g is conjugate to h (g, h belonging to G) then h is conjugate to g (***symmetry***),*

*(c) if g_1 is conjugate to g_2 and g_2 is conjugate to g_3 then g_1 is conjugate to g_3 (***transitivity***).*

Exercise 5.12.15. *Let $x, y \in Cl_G(g)$ show $ord(x) = ord(y) = ord(g)$. (This implies that $ord : G_* \to \mathbb{N}$, $ord(x) = ord(g)$, for any $x \in Cl_G(g)$, is a well-defined function.)*

Hint: Two elements which are conjugate must have the same order since $(h^{-1}gh)^n = (h^{-1}gh)(h^{-1}gh)...(h^{-1}gh) = h^{-1}g^nh$, for $n = 1, 2, ...$ and $g, h \in G$.

Exercise 5.12.16. *Show that two permutations $g_1, g_2 \in S_n$ are conjugate: they are conjugate if and only if the cycles in their respective disjoint cycle decompositions have the same length when arranged from shortest to longest. (Hint: Use Lemma 4.4.1.)*

5.13 Cosets

Let G be a group and H a subgroup of G. For g belonging to G, the subset

$$g * H = gH = \{gh \mid h \in H\},$$

of G is called a **left coset** of H in G and the subset

$$H * g = Hg = \{hg \mid h \in H\},$$

of G is called a **right coset** of H in G.

Example 5.13.1. *If* $H = \langle (1,3),(1,4)\rangle$ *and* $G = S_4$ *as in Example 5.9.2, then* $(1,2,3)H = \{(1,2,3),(1,2,3)(1,3),(1,2,3)(1,4),...\}$ *(* $= (1,2,3)H = \{(1,2,3),(1,2),(1,2,3,4),...\}$*) and* $H(1,2,3) = \{(1,2,3),(1,3)(1,2,3),(1,4)(1,2,3),...\}$ *(*$= \{(1,2,3),(2,3),...\}$*). Does* $(1,2,3)H = H(1,2,3)$*?*

Notation: The set of all left cosets is written G/H and the set of all right cosets of H in G is denoted $H\backslash G$.

These two sets don't in general inherit a group structure from G but they are useful none-the-less. (G/H is a group with the "obvious" multiplication $(g_1 * H) * (g_2 * H) = (g_1 g_2) * H$ if and only if H is a "normal" subgroup of G - we will define "normal" below.)

Let X be a finite set and let G be a finite subgroup of the symmetric group S_X. The elements of G permute X. With this permutation in mind, we say that G **acts** on X. For $x \in X$ and $g \in G$, let $g * x$ denote $g(x) \in X$. This action satisfies the following properties:

- for all $g \in G$, $g : X \to X$ is a bijection,
- for all $x \in X$, $1 * x = x$,
- for all $g, h \in G$, $(gh) * x = g * (h * x)$.

We call the set of images

$$G * x = \{g * x \mid g \in G\}$$

the **orbit** of x under G. For each $x, y \in X$, we define $x \sim y$ if and only if $y = g * x$, for some $g \in G$. We leave it as an exercise to verify that $\sim$ is an equivalence relation on X, in the sense of §1.7.2. The equivalence class of x

is the orbit of x under G: $[x] = G * x = \{g * x \mid g \in G\}$. The verification of this is also left as an exercise.

We call

$$stab_G(x) = \{g \in G \mid g * x = x\} \tag{5.3}$$

the **stabilizer** of x in G. As an example of their usefulness, we have the following relationship between the orbits and the cosets of the stabilizers.

Proposition 5.13.2. *Let G be a finite group acting on a set X. Then*

$$|G * x| = |G/stab_G(x)|,$$

for all x belonging to X.

proof: The map

$$g * stab_G(x) \longmapsto g * x$$

defines a function $f : G/stab_G(x) \to G * x$. The interested reader can easily check that this function is a bijection since it is both an injection and a surjection. □

Corollary 5.13.3. *Let G be a finite group acting on itself by conjugation. Let $S \subset G$ denote a complete set of representatives of the conjugacy classes G_* in G and let $S' = S - Z(G)$, i.e., the subset of S of those elements which are not central. Then*

$$G = \cup_{x \in S} Cl(x) = \cup_{x \in S} G/stab_G(x) = Z(G) \cup \cup_{x \in S'} G/stab_G(x),$$

for all x belonging to X. In particular,

$$|G| = \sum_{x \in S} |G/stab_G(x)| = |Z(G)| + \sum_{x \in S'} |G/stab_G(x)|,$$

proof: In the first displayed equation: The first equation is Theorem 5.12.5. The second equality follows from the above proposition.

By taking cardinalities, the second displayed equation is a consequence of the first. □

Corollary 5.13.4. *Theorem 5.8.7(a) holds.*

proof: The argument is by induction on $|G|$.

The result is trivial if $|G| = 1$ (since then no prime divides $|G|$).

Suppose $|G| > 1$ and let p be a prime dividing $|G|$. By the induction hypothesis, we assume that the result is true for all subgroups H of G with $|H| < |G|$.

Suppose $x \in G$ is not central. Then its centralizer $C_G(x)$ is a proper subgroup of G. If $p||C_G(x)|$ then the result follows from the induction hypothesis. If p does not divide $|C_G(x)|$ (for all non-central x) then since $|G| = |C_G(x)||G/stab_G(x)|$, by Proposition 5.13.2, it follows that $p||G/stab_G(x)|$. By Corollary 5.13.3 above, we must have $p||Z(G)|$. If $Z(G)$ is a proper subgroup of G then we are done by the induction hypothesis.

Thus we may assume $G = Z(G)$ is abelian. If G is cyclic then we leave it to the reader to show that G contains an element of order p. We shall assume that G is not cyclic. Let H be a proper subgroup of G of maximal order (this exists since G is not cyclic) and let $a \in G - H$. Then $\langle a \rangle \cap H = \{1\}$ (else $a \in H$) and $G = H \cdot \langle a \rangle$ (else H would not be maximal). Thus p either divides H or $|\langle a \rangle|$. In either case, the result follows from the induction hypothesis. □

Theorem 5.13.5. *(Lagrange): If G is a finite group and H a subgroup then*

$$|G/H| = |G|/|H|.$$

Corollary 5.13.6. *If H, G are as above then the order of H divides the order of G.*

Now we prove the Theorem.

proof: Let X be the set of left cosets of H in G and let G act on X by left multiplication. Apply the previous lemma with $x = H$. □

Definition 5.13.7. *: Let H be a subgroup of G and let C be a left coset of H in G. We call an element g of G a* **coset representative** *of C if $C = g * H$. A* **complete set of coset representatives** *is a subset of G, $x_1, x_2, ..., x_m$, such that*

$$G/H = \{x_1 * H, ..., x_m * H\},$$

without repetition (i.e., all the $x_i * H$ are disjoint).

Exercise 5.13.8. *Let G be the group of symmetries of the square. Using the notation above, compute $G/\langle g_3 \rangle$ and $G * x_0$.*

Exercise 5.13.9. *For $g_1, g_2 \in G$, define $g_1 \sim g_2$ if and only if g_1 and g_2 belong to the same left coset of H in G.*

(a) Show that $\sim$ is an equivalence relation.

(b) Show that the left cosets of H in G partition G.

Exercise 5.13.10. *If H is finite, show $|H| = |g * H| = |H * g|$.*

Exercise 5.13.11. *Let $G = \mathbb{Z}$ and $H = 3\mathbb{Z}$ (i.e. H is the set of multiples of 3).*

(a) Prove that $H < G$.

(b) What is $|\mathbb{Z}/3\mathbb{Z}|$?

Exercise 5.13.12. *Suppose $|G| < 30$ and G is nonabelian and $g \in G$ is an element of order 11.*

(a) What is $|G|$?

(b) What is G and what is g?

Exercise 5.13.13. *Let G be the symmetry group of a cube. By this, we mean the following: If X is the set of vertices of a cube, e.g. $\{(0,0,0), ..., (1,1,1)\}$, consider the subgroup of the full permutation group of X that arises from physically possible rotations of space. (This is the analog of the symmetry group of the square, D_8, discussed earlier.)*

(a) G can be considered to be a subgroup of S_8. Find G, $|G|$ and $|S_8/G|$.

(b) What is $|stab_G((1,1,1))|$?

Exercise 5.13.14. *If X is a left coset of H in G and x is an element of G, show that $x * X$ is also a left coset of H in G.*

Exercise 5.13.15. *Let $G = S_3$, the symmetric group on 3 letters, and let $H = \langle s_1 \rangle$, in the notation of §5.3 above.*

(a) Compute $|G/H|$ using Lagrange's Theorem.

(b) Explicitly write down all the cosets of H in G.

5.14 Functions between two groups

To compare two groups, we first talk about functions between two groups. A homomorphism between two groups is, roughly speaking, a function between them which preserves the (respective) group operations.

Definition 5.14.1. *Let G_1, G_2 be groups, with $*_1$ denoting the group operation for G_1 and $*_2$ the group operation for G_2. A function $f : G_1 \to G_2$ is a* **homomorphism** *if and only if, for all $a, b \in G_1$, we have*

$$f(a *_1 b) = f(a) *_2 f(b).$$

The subgroup $f(G_1) < G_2$ is called the **image of** f and is sometimes denoted $im(f)$.

Example 5.14.2. *Let G be a group and h a fixed element of G. Define $f : G \to G$ by*

$$f(g) = h^{-1} * g * h, \qquad g \in G.$$

Then the following simple trick

$$f(a * b) = h^{-1} * (a * b) * h = h^{-1} * a * h * h^{-1} * b * h = f(a) * f(b)$$

shows that f is a homomorphism. In this case, $im(f) = G$, i.e., f is surjective.

Example 5.14.3. *The function*

$$sign : S_n \to \{\pm 1\},$$

which assigns to each permutation its sign, is a homomorphism, as was proven in chapter 3. Another reason why this is true is due to the fact that the sign of a permutation g is the determinant of the associated permutation matrix $P(g)$. Since the determinant of the product is the product of the determinants (see Lemma 5.5.3), we have

$$sgn(gh) = \det P(gh) = \det(P(g)P(h)) = \det P(g) \det P(h) = sign(g)sign(h),$$

for all $g, h \in S_n$. From this it follows that sign is a homomorphism.

Lemma 5.14.4. *If $f : G_1 \to G_2$ is a homomorphism then*

(a) $f(e_1) = e_2$, where e_1 denotes the identity element of G_1 and e_2 denotes the identity element of G_2,

(b) $f(x^{-1}) = f(x)^{-1}$, for all x belonging to G_1,

(c) $f(y^{-1} *_1 x *_1 y) = f(y)^{-1} *_2 f(x) *_2 f(y)$, for all x, y belonging to G_1, where $*_1$ denotes the group operation for G_1 and $*_2$ the group operation for G_2.

proof: (a) We have $f(x) = f(x *_1 e_1) = f(x) *_2 f(e_1)$, for any $x \in G_1$. Multiply both sides of this equation on the left by $f(x)^{-1}$.

(b) We have, by part (a), $e_2 = f(e_1) = f(x *_1 x^{-1}) = f(x) *_2 f(x^{-1})$. Multiply both sides of this equation on the left by $f(x)^{-1}$. $\square$

Definition 5.14.5. *Let G_1, G_2 be finite groups. We say that G_1* **embeds** *(or* **injects***) into G_2 if there exists an injective homomorphism $f : G_1 \to G_2$. A homomorphism $f : G_1 \to G_2$ is a* **isomorphism** *if it is a bijection (as a function between sets). In this case, we call G_1 and G_2* **isomorphic** *and write $G_1 \cong G_2$. An isomorphism from a group G to itself is called an* **automorphism**.

The notion of an isomorphism is the notion we will use when we want to same two groups are "essentially the same group", i.e., one is basically a carbon copy of the other with the elements relabeled and the binary operation modified. For example, if you have any two groups of order 2 then they must be isomorphic. (The interested reader can verify that the map which sends the identity of one group to the identity of the other and the only non-identity of one group to the only non-identity of the other must be a group isomorphism.) In other words, there is only one group of order 2, up to isomorphism.

Let $O(n)$ denote the the number of non-isomorphic groups of order n, so O is a function $O : \mathbb{N} \to \mathbb{N}$ (where $\mathbb{N} = \{1, 2, 3, ...\}$). This is a rather curiously behaving function.

Order	Group G	notes
2	C_2	
3	C_3	$G \cong A_3$
4	C_4	
4	$C_2 \times C_2$	Klein 4-group $Aut(G) \cong GL(2, \mathbb{F}_2)$
5	C_5	
6	$C_6 = C_2 \times C_3$	
6	S_3	$Aut(G) = G$ $G \cong GL(2, \mathbb{F}_2)$
7	C_7	
8	C_8	
8	$C_2 \times C_4$	
8	$C_2 \times C_2 \times C_2$	$Aut(G) \cong GL(3, \mathbb{F}_2)$
8	D_4	
8	Q	
9	C_9	
9	$C_3 \times C_3$	$Aut(G) \cong GL(2, \mathbb{F}_3)$
10	$C_{10} = C_2 \times C_5$	
10	D_5	

Exercise 5.14.6. *Pick a group at random from the above table and find a subgroup of the Rubik's cube group which is isomorphic to it. (This is not easy. See the chapter entitles "Words which move" in [J2] for some hints.)*

Exercise 5.14.7. *Prove the following: If $f : G_1 \to G_2$ is a homomorphism of groups then*

$$f(G_1) = \{g \in G_2 \mid g = f(x), \text{ for some } x \in G_1\}$$

is a subgroup of G_2.

Exercise 5.14.8. *Find $O(n)$, $n = 1, 2, ..., 30$. (Hint: Use GAP or MAGMA. This is practically impossible to do "by hand".)*

Exercise 5.14.9. *Let G be the group $G = \{f_1, f_2, f_3, f_4, f_5, f_6\}$ as given in Exercise 5.4.7. Is G isomorphic to S_3? To $C_2 \times C_3$? Explain.*

Exercise 5.14.10. *Let G be any group and let $h \in G$ be any fixed element. Let $\phi : G \to G$ be defined by $\phi(g) = h^{-1}gh$, for $g \in G$. Show that ϕ is an isomorphism.*

Exercise 5.14.11. *Let*

$$G = \{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}\}$$

and $H = \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$. Show that $G \cong H$.

Exercise 5.14.12. *Show the following:*

(a) If $m \geq 1$ then $C_m \cong \mathbb{Z}/m\mathbb{Z}$.

(b) If $m > 1, n > 1$ have no prime divisors in common, i.e., if m, n are relatively prime, then $C_m \times C_n \cong C_{mn}$.

(Hint: (a) Let $a \in C_m$ be a generator. The map $f = f_{m,a} : C_m \to \mathbb{Z}/m\mathbb{Z}$ which satisfies $f(a) = 1$ extends to a unique homomorphism between these groups. Show that this is an isomorphism. (b) First, show $g(x + mn\mathbb{Z}) = (x+m\mathbb{Z}, x+n\mathbb{Z})$, for all $x \in \mathbb{Z}$ defines a homomorphism $\mathbb{Z}/mn\mathbb{Z} \to \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$. Next, using the assumption that m, n are relatively prime, show that this function g is an injection. Conclude, since the domain and range of g both have the same cardinality, that g is also a surjection. Finally, using part (a), show $C_m \times C_n \cong C_{mn}$.)

Exercise 5.14.13. *Show part (c) in Lemma 5.14.4. (Hint: use the definition of homomorphism and part (b)).*

Exercise 5.14.14. *Let*

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Now, let $G = \langle A, B \rangle$ denote the group of all matrices which can be written as any arbitrary product of these two matrices (in any order and with as many

terms as you want). We have

$$G = \{I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}\}$$

(The interested reader may want to try to check this by regarding each such matrix as a permutation matrix.) Define $f : G \to S_3$ *by*

g	$f(g)$
I_3	1
A	s_1
B	s_2
$A * B$	$s_1 * s_2$
$B * A$	$s_2 * s_1$
$A * B * A$	$s_1 * s_2 * s_1$

Show that this is a homomorphism.

5.15 Application: Campanology, revisited

Let us put the material on bell-ringing from the previous chapter in the context of group theory. The fact that this can be done so easily is quite remarkable, considering that group theory wasn't developed for at least 100 years later!

Generating the plain lead on four bells is analogous algebraically to generating the dihedral group of order 8, D_4. If $a = (1,2)(3,4)$, which swaps the first two and last two bells, and if $b = (2,3)$, which swaps the middle pair, then the plain lead on four bells corresponds to

$$D_4 = \{1, a, ab, aba, (ab)^2, (ab)^2a, (ab)^3, (ab)^3a\}.$$

Plain Bob Minimus is equivalent algebraically to generating the symmetric group on 4 elements, S_4. Let a and b be as before. If we look at the first column of the Plain Bob Minimus composition, we see that it is nothing

more than the dihedral group, D_4, which is a subgroup of S_4. To generate the second column of S_4 we introduce $c = (3, 4)$ and let $k = (ab)^3ac$. The second column corresponds to kD_4 and the third column to k^2D_4. The generation of the Plain Bob Minimus shows that S_4 can be expressed as the disjoint union of cosets of the subgroup D_4, that is, the cosets of D_4 in S_4 partition S_4. There is an important generalization of this fact, which states:

Theorem 5.15.1. *For any group G and any subgroup H, the cosets of H in G partition G.*

As White [Wh] concludes in his paper, he is not suggesting "that Fabian Stedman was using group theory explicitly, but rather that group theoretical ideas were implicit in (Stedman's) writings and compositions".

5.16 The Cayley graph of a group

In this section we introduce a graphical interpretation of a permutation group, the Cayley graph. This is then interpreted in the special case of a group arising from a permutation puzzle.

5.16.1 Graphs

To begin, what's a graph? A **graph** is a pair of countable sets (V, E), where

- V is a countable set of singleton elements called **vertices**,
- E is a subset of the set of all unordered pairs $\{\{v_1, v_2\} \mid v_1, v_2 \in V, v_1 \neq v_2\}$. The elements of E are called **edges**.

A graph is drawn by simply connecting points representing vertices together by a line segment if they belong to the same edge.

A **digraph**, or **directed graph**, is a pair of countable sets (V, E), where

- V is a countable set of vertices,
- E is a subset of ordered pairs $\{(v_1, v_2) \mid v_1, v_2 \in V, v_1 \neq v_2\}$ called **edges**.

A digraph is drawn by simply connecting points representing vertices together by an arrow if they belong to the same edge (v_1, v_2), the arrow originating at v_1 and arrowhead pointing to v_2.

If $e = \{v_1, v_2\}$ belongs to E then we say that e is an "edge from v_1 to v_2" (or from v_2 to v_1). If v and w are vertices, a **path** from v to w is a finite sequence of edges beginning at v and ending at w:

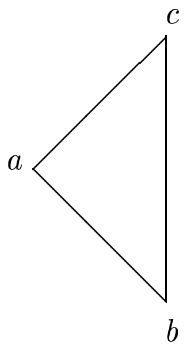
$$e_0 = \{v, v_1\}, e_1 = \{v_1, v_2\}, ..., e_n = \{v_n, w\}.$$

If there is a path from v to w then we say v is **connected** to w. We say that a graph (V, E) is **connected** if each pair of vertices is connected. The number of edges eminating from a vertex v is called the **degree** (or **valence**) of v, denoted degree(v).

Example 5.16.1. *: If*

$$V = \{a, b, c\}, \qquad E = \{\{a, b\}, \{a, c\}, \{b, c\}\},$$

then we may visualize (V, E) *as*



Each vertex has valence 2.

Definition 5.16.2. *: If* v *and* w *are vertices connected to each other in a graph* (V, E) *then we define the* **distance from** v **to** w*, denoted* $d(v, w)$*, by*

$$d(v, w) = \min_{v,w \in V \text{ connected}} \#\{\text{edges in a path from } v \text{ to } w\}$$

By convention, if v and w are not connected then we set $d(v,w) = \infty$. The **diameter** *of a graph is the largest possible distance:*

$$\text{diam}((V,E)) = \max_{v,w \in V} d(v,w).$$

In the above example, the diameter is 1.

5.16.2 Cayley graphs

Let G be a permutation group,

$$G =< g_1, g_2, ..., g_n > < S_X.$$

The **Cayley graph** of G with respect to $X = \{g_1, g_2, ..., g_n\}$ is the graph (V,E) whose vertices V are the elements of G and whose edges are determined by the following condition: if x and y belong to $V = G$ then there is an edge from x to y (or from y to x) if and only if $y = g_i * x$ or $x = g_i * y$, for some $i = 1, 2, ..., n$.

Cayley graphs are named for Arthur Cayley (August 1821-January 1895), who though starting out as a laywer, eventually published over 900 papers and notes covering nearly every aspect of modern mathematics. The most important of his work is in developing the algebra of matrices, work in non-euclidean geometry and n-dimensional geometry.

The **Cayley digraph** of G with respect to $X = \{g_1, g_2, ..., g_n\}$ is the digraph (V,E) whose vertices V are the elements of G and whose edges are determined by the following condition: if x and y belong to $V = G$ then there is an edge from x to y if and only if $y = x * g_i$, for some $i = 1, 2, ..., n$.

Lemma 5.16.3. *Let $\Gamma_G = (V,E)$ denote the Cayley graph associated to the permutation group $G =< g_1, g_2, ..., g_n >$. Let $N = |\{g_1, g_1^{-1}, g_2, g_2^{-1}, ..., g_n, g_n^{-1}\}|$. Then, for all $v \in V$,* degree$(v) = N$.

proof: Assume not. Then there is a $v \in V = G$ with either
(i) degree$(v) < N$, or
(ii) degree$(v) > N$.

First, we note that, for each $h \in \{g_1, g_1^{-1}, g_2, g_2^{-1}, ..., g_n, g_n^{-1}\}$, the set $\{v, h * v\}$ is an edge of Γ_G. This follows from the definition of the Cayley graph.

If $r =$ degree$(v) > N$ then, by definition of the Cayley graph, there are distinct $v_1, ..., v_r \in V$ with $v = h_i * v_i$, for all $1 \leq i \leq r$, where the $h_1, ..., h_r$

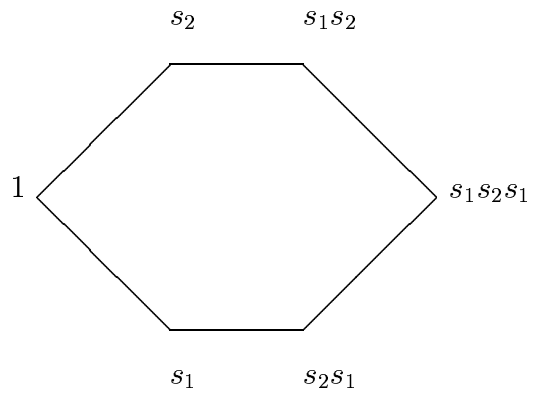
are distinct elements of $\{g_1, g_1^{-1}, g_2, g_2^{-1}, ..., g_n, g_n^{-1}\}$. This contradicts the definition of N.

If $r = \text{degree}(v) < N$ then, by definition of the Cayley graph, there are distinct h_i, h_j in $\{g_1, g_1^{-1}, g_2, g_2^{-1}, ..., g_n, g_n^{-1}\}$ such that $h_i * v = h_j * v$. Since G is a group and $V = G$ (as sets), we may cancel the v's from both sides of the equation $h_i * v = h_j * v$, contradicting the assumption that h_i is distinct from h_j. □

Example 5.16.4. *: Let*

$$G = \langle s_1, s_2 \rangle = S_3,$$

where $s_1 = (1,\ 2)$*, and* $s_2 = (2,\ 3)$*. Then the Cayley graph of* G *with respect to* $X = \{s_1, s_2\}$ *may be visualized as*



Example 5.16.5. *: Let*

$$G = \langle R, L, U, D, F, B \rangle < S_{54}$$

be the group of the 3×3 *Rubik's cube. Each position of the cube corresponds to an element of the group* G *(i.e., the move you had to make to get to that position). In other words, each position of the cube corresponds to a vertex of the Cayley graph. Each vertex of this graph has valence 12.*

Moreover, a solution of the Rubik's cube is simply a path in the graph from the vertex associated to the present position of the cube to the vertex associated to the identity element. The number of moves in the shortest possible solution is simply the distance from the vertex associated to the present

position of the cube to the vertex associated to the identity element. The diameter of the Cayley graph of G is the number of moves in the best possible solution in the worst possible case.

Exercise 5.16.6. *Construct the Cayley graph of C_4, the cyclic group, with respect to the generator $s = (1, 2, 3, 4)$.*

Exercise 5.16.7. *Check that each vertex of the Cayley graph of $G = \langle R, L, U, D, F, B \rangle$ has valence 12.*

Exercise 5.16.8. *Construct the Cayley graph of S_4, the symmetric group on four letters, with respect to the generators $s_1 = (1\ 2)$, $s_2 = (2\ 3)$ and $s_3 = (3\ 4)$.*

Exercise 5.16.9. *Construct the Cayley digraph of S_3 with respect to the generators $f = (1, 3)$, $r = (1, 2, 3)$. (Show, in particular, that f, r do indeed generate S_3.)*

Exercise 5.16.10. *Show that the Cayley graph of a permutation group is connected.*

Exercise 5.16.11. *Construct the Cayley graph of C_4, the cyclic group, with respect to the generator $s = (1, 2, 3, 4)$.*

Exercise 5.16.12. *Construct the Cayley graph of S_4, the symmetric group on four letters, with respect to the generators $s_1 = (1\ 2)$, $s_2 = (2\ 3)$ and $s_3 = (3\ 4)$.*

Exercise 5.16.13. *Construct the Cayley digraph of S_3 with respect to the generators $f = (1, 3)$, $r = (1, 2, 3)$. (Show, in particular, that f, r do indeed generate S_3.)*

5.16.3 Application: God's algorithm

Problem: Let G be the group of a permutation puzzle. Find the diameter of the Cayley graph of G.

Sometimes this is called God's algorithm, though here that term is reserved for a harder version of the problem, stated below. This diameter problem is unsolved for must puzzles (including the 3×3 Rubik's cube) and appears to be very difficult computationally (see [J1] for a discussion of similar problems). One case where it is known includes the 2×2 Rubik's cube puzzle [CFS], which has diameter 14.

Problem: Let G be the group of a permutation puzzle and let v be a vertex in the Cayley graph of G. Find an algorithm for determining a path from v to the vertex v_0 associated to the identity having length equal to the distance from v to v_0.

This problem is much harder. The algorithm, if it exists, is called **God's algorithm**.

Let Γ be a graph. A **Hamiltonian circuit** on Γ is a sequence of edges forming a path in Γ which passes through each vertex exactly once. (If you think of the vertices as cities and the edges as roads then a Hamiltonian circuit is a tour visiting each city exactly once.)

The following unsolved problem was first mentioned in this context (as far as I know) by A. Schwenk.

Problem: Let G be the group of the 3×3 Rubik's cube puzzle. Does the Cayley graph of G have a Hamiltonian circuit? In other words, can we (in principle) "visit" each possible position of the Rubik's cube exactly once, by making one move at a time using only the basic generators R, L, U, D, F, B?

This is a special case of a more general unsolved problem: For an arbitary permutation group with more than two elements, is the Cayley graph Hamiltonian?

An example of one where the Hamiltonian circuit problem is known is the symmetric group.

Example 5.16.14. *Let G be the group S_n with generators given to be the set of all transpositions:*

$$G = S_n, \qquad X = \{(i, j) \mid 1 \leq i < j \leq n\}.$$

(There are many more transpositions than necessary to generate S_n since the subset of transpositions of the form $(i, i+1)$, $1 \leq i \leq n-1$, suffice to generate S_n [R].) The algorithm of Steinhaus (see §4.5) shows that there is a Hamiltonian circuit in the Cayley graph of S_n with respect to X.

The reader interested in more examples is referred to [CG].

Exercise 5.16.15. *Find the Cayley graph of the* **sliced squared group**

$$G = \langle M_R^2, M_F^2, M_D^2 \rangle,$$

where M_R is the middle slice move which turns the middle slice parallel to the right face clockwise 90 degrees (with respect to the right face). Find the diameter of this graph.

5.17 Kernels are normal, some subgroups are not

Let $f : G_1 \to G_2$ be a homomorphism between two groups. Let

$$ker(f) = \{g \in G_1 \mid f(g) = e_2\},$$

where e_2 is the identity element of G_2. This set is called the **kernel** of f.

Lemma 5.17.1. *$ker(f)$ is a subgroup of G_1.*

The interested reader can this for him/herself by using the definition of a homomorphism.

Example 5.17.2. *Let*

$$sgn : S_n \to \{\pm 1\}$$

denote the homomorphism which associates to a permutation either 1, if it is even, or -1, if it is odd. Then $A_n = ker(sgn) \subset S_n$.

The following properties of the kernel are useful:

Lemma 5.17.3. *: Let $f : G_1 \to G_2$ be a homomorphism between two groups.*
(a) f is injective if and only if $ker(f) = \{e_1\}$.
(b) if g belongs to the kernel and x is any element of G_1 then $x^{-1} * g * x$ must also belong to the kernel.

proof: (a) f is injective if and only if $f(g_1) = f(g_2)$ implies $g_1 = g_2$ $(g_1, g_2 \in G_1)$. Note $f(g_1) = f(g_2)$ is true if and only if $f(g_1 * g_2^{-1}) = e_2$. If $ker(f) = \{e_1\}$ then $f(g_1 * g_2^{-1}) = e_2$ implies $g_1 * g_2^{-1} = e_2$, which implies $g_1 = g_2$, which implies f is injective.

Therefore, if $ker(f) = \{e_2\}$ then f is injective. Conversely, if f is injective then $f(x) = f(e_1)(= e_2)$ implies $x = e_1 (x \in G_1)$. This implies $ker(f) = \{e_1\}$.

(b) Multiply both sides of $e_2 = f(g)$ on the left by $f(x)^{-1}$ and on the right by $f(x)$. We get

$$e_2 = f(x)^{-1} * e_2 * f(x) = f(x^{-1}) * f(g) * f(x) = f(x^{-1} * g * x),$$

as desired. □

Definition 5.17.4. *Let H be a subgroup of G. We say that H is a* **normal** *subgroup if, for each $g \in G$, $g^{-1} * H * g = H$ (i.e., for each $g \in G$ and each $h \in H, g^{-1} * h * g$ belongs to H).*

Notation: Sometimes we denote "H is a normal subgroup of G" by

$$H \triangleleft G$$

Example 5.17.5. *(a)* $A_n \triangleleft S_n$ *and* $|A_n| = \frac{1}{2}|S_n|$.

(b) On the other hand, examples of subgroups which are not normal are easy to come by. If $n > 5$ *and* H *is any non-trivial proper subgroup of* A_n *(for example, any non-trivial cyclic subgroup) then* H *is not normal in* A_n *(see Theorem 5.17.7 below).*

We have already shown the following

Lemma 5.17.6. *If* $f : G_1 \to G_2$ *is a homomorphism between two groups then* $ker(f)$ *is a normal subgroup of* G_1.

5.17.1 The alternating group

The following remarkable result about the alternating group will not be needed to understand the structure of the Rubik's cube. However, the theorem below is interesting because of its connection with the fact (due to N. Abel and E. Galois) that you cannot solve the general polynomial of degree 5 or higher using radicals, i.e., that there is no analog of the quadratic formula for polynomials of degree 5 or higher. Explaining this connection would take us too far from our main topic. The interested reader is referred to Artin [Ar], chapter 14.

Theorem 5.17.7. *If* X *has* 5 *elements or greater then* A_X *has no non-trivial proper normal subgroups. In other words, if* $H \triangleleft A_X$ *is a normal subgroup then either* $H = \{1\}$ *or* $H = A_X$.

This will not be proven here. (For an algebraic proof, see for example [R].) For a visual, geometric "proof" in the case when X has 5 elements, see the web site [K]. The discussion there hinges on the fact that the symmetry group of the icosehedron can be identified with A_5, much like the symmetry group of the cube can be identified with S_4. See Exercise 5.13.13.

Unlike the above theorem, the next fact about the alternating group *will* be needed later. It will be used in our determination of the structure of the Rubik's cube group. This fact also arose in connection with our discussion of the "legal positions" of the 15 puzzle in a previous chapter.

Proposition 5.17.8. *Let H be the subgroup of S_n generated by all the 3-cycles in S_n then $H = A_n$.*

proof: Since $sgn : S_n \to \{\pm 1\}$ is a homomorphism, and since any 3-cycle is even, any product of 3-cycles must also be even. Therefore, $H \subset A_n$. If $g \in A_n$ then g must swap an even number of the inequalities $1 < 2 < ... < n-1 < n$, by Definition 4.2.2. Therefore, (since any permuation may be written as a product of 2-cycles, Theorem 4.5.1) g must be composed of permutations of the form $(i,\ j)(k,\ l)$ or $(i,\ j)(j,\ k)$. But $(i,\ j)(k,\ l) = (i,\ j,\ k)(j,\ k,\ l)$ and $(i,\ j)(j,\ k) = (i,\ j,\ k)$. Therefore, $g \in H$. This implies $A_n \subset H$, so $A_n = H$. $\square$

As an application to the Rubik's cube, suppose you have all the corners in the correct position. They might be twisted, that's okay. The point is that they don't need to be swapped or permuted. Suppose that the edges are not in the correct position. It turns out that the permuation which puts the edges in the correct position must be an even permutation. There are simple moves for a 3-cycle on edges: $M_2^2 U^{-1} M_R^{-1} U^2 M_R U^{-1} M_R^2$.

5.18 Quotient groups

One of the most useful facts about normal subgroups is the following result/definition.

Lemma 5.18.1. *If H is a normal subgroup of G then the coset space G/H with the binary operation,*

$$aH * bH = (ab)H, \qquad (aH)^{-1} = a^{-1}H,$$

for all a, b belonging to G, is a group. The identity element of this group is the trivial coset H.

This group G/H is called the **quotient group** of G by H and is sometimes pronounced "G mod H".

Example 5.18.2. *If $f : G_1 \to G_2$ is a homomorphism between two groups then $G_1/ker(f)$ is a group.*

Next we introduce an important idea first emphasized by E. Galois (October 1811-May 1832), the French mathematician mentioned in the introduction. Though Galois attended secondary school, he had trouble entering the

French university system. Sadly, by the time he entered the École Normale Súperieure in November 1829, he also began to get caught up in the activities of the French revolution which toppled Charles X. His political activities resulted in him being expelled from schoolin December 1830. Though apparently he had been a rather rebellious teenager, this only frustrated him further. He died in a fight with another Frenchman.

Galois's work was the first to begin to illuminate the "basic building blocks" of the collection of finite groups (i.e., the analog of the idea that atoms are the "basic building blocks" of molecules). Galois introduced the ideas of solvable groups and normal groups. Roughly speaking, one may say that the basic building blocks of finite groups are those the groups which have no proper non-trivial normal subgroups. Intuitively, this is because a non-trivial quotient group (by a normal subgroup) is closely related to the original group but smaller in size (and hence perhaps subject to analysis by an inductive argument of some type). These basic building blocks are called "simple" groups.

Definition 5.18.3. *A* **simple group** *is a group with no proper normal subgroups other than the trivial subgroup* $\{1\}$.

All finite simple groups have been classified. In fact, the survey book discussing this is free and download able from [GLS].

Example 5.18.4. *If p is a prime then C_p (the cyclic group having p elements) is simple. In fact, if G is any group which is both abelian and simple then there is a prime p such that $G \cong C_p$. If $n > 4$ then A_n is simple (as was stated above in Theorem 5.17.7). These facts are proven in [R].*

Simple groups are not very abundant. In fact, the first non-abelian simple group is of order 60 *(it's A_5).*

Example 5.18.5. *Let $f : \mathbb{Z} \to \mathbb{Z}/m\mathbb{Z}$ denote the map which sends an integer n to its residue $\overline{n} = n \pmod{m}$. This map send every multiple of m to $\overline{0}$. In fact, if $f(n) = \overline{0}$ then $m|n$. Therefore, $ker(f) = m\mathbb{Z}$.*

The following basic result, which generalizes the above example, describes the quotient group $G_1/ker(f)$.

Theorem 5.18.6. *(***First Isomorphism Theorem***) If $f : G_1 \to G_2$ is a homomorphism between two groups then $G_1/ker(f)$ is isomorphic to $f(G_1)$.*

proof: $ker(f)$ is a normal subgroup of G_1, so $G_1/ker(f)$ is a group. We must show that this group is isomorphic to the group $f(G_1)$. Define $\phi : G_1/ker(f) \to f(G_2)$ by $\phi(g \cdot ker(f)) = f(g)$, for $g \in G_1$. We must show

(a) ϕ is well-defined,

(b) ϕ is a homomorphism,

(c) ϕ is a bijection.

If $g \cdot ker(f) = g' \cdot ker(f)$ then $g^{-1}g' \in ker(f)$, since $ker(f)$ is a group. This implies $f(g^{-1}g') \in f(ker(f)) = \{1\}$, so $f(g) = f(g')$. This implies ϕ is well-defined.

Since $ker(f)$ is normal, $(g \cdot ker(f))(g' \cdot ker(f)) = gg'(g'^{-1}ker(f)g')ker(f) = gg' \cdot ker(f)$. Therefore $\phi((g \cdot ker(f))(g' \cdot ker(f))) = \phi(gg' \cdot ker(f)) = f(gg') = f(g)f(g') = \phi(g \cdot ker(f))\phi(g' \cdot ker(f))$, for all $g, g' \in G$. This implies ϕ is a homomorphism.

It is clear that ϕ is surjective. To show that ϕ is a bijection, it suffices to prove ϕ is an injection. Suppose that $\phi(g \cdot ker(f)) = \phi(g' \cdot ker(f))$, for some $g, g' \in G$. Then $f(g) = f(g')$, so $f(g^{-1}g') = 1$. By definition of the kernel, this implies $g^{-1}g' \in ker(f)$, so $g \cdot ker(f) = g' \cdot ker(f)$. This implies ϕ is injective.

□

Example 5.18.7. *If $f : k\mathbb{Z}/lcm(m,k)\mathbb{Z} \to \mathbb{Z}/\frac{m}{gcd(k,m)}\mathbb{Z}$ is the "mod $\frac{m}{gcd(k,m)}$ map" then f is an isomorphism.*

In particular, if $m > 1$ and $k > 1$ are relatively prime integers then $k\mathbb{Z}/mk\mathbb{Z} \cong \mathbb{Z}/m\mathbb{Z}$.

Indeed, by replacing m by $m/gcd(k,m)$ if necessary, we may assume that $m > 1$ and $k > 1$ are relatively prime integers. It suffices to show that the kernel is trivial since both $k\mathbb{Z}/mk\mathbb{Z}$ and $\mathbb{Z}/m\mathbb{Z}$ have the same cardinality. If f sends $\overline{ka}$ to $\overline{0}$ (where $ka \in k\mathbb{Z}$) then $m|ka$, so $mk|ka$ since m, k are relatively prime.

proof: □

Example 5.18.8. *Let $f : \mathbb{Z} \to \mathbb{Z}/k\mathbb{Z}$ denote the map which sends an integer a to its residue $\overline{a} = a \pmod{k}$. Let $g : \mathbb{Z} \to \mathbb{Z}/m\mathbb{Z}$ denote the map which sends an integer a to its residue $\overline{a} = a \pmod{m}$, where $k > 1$, $m > 1$ are integers. Let $H = ker(f)$, $N = ker(g)$, so that $N \cap H = m\mathbb{Z} \cap k\mathbb{Z} = lcm(k,m)\mathbb{Z}$.*

Example 5.18.9. *Let $f_1 : \mathbb{Z} \to \mathbb{Z}/km\mathbb{Z}$ denote the map which sends an integer a to its residue $\overline{a} = a \pmod{km}$. Let $f_2 : \mathbb{Z} \to \mathbb{Z}/m\mathbb{Z}$ denote*

the map which sends an integer a to its residue $\overline{a} = a \pmod m$, where $k > 1$, $m > 1$ are integers. Let $N_1 = ker(f_1)$, $N_2 = ker(f_2)$, so that $N_1 = km\mathbb{Z} \subset N_2 = m\mathbb{Z}$.

Then $N_2/N_1 = m\mathbb{Z}/km\mathbb{Z}$ is a subgroup of $\mathbb{Z}/km\mathbb{Z}$ and the map $f : \mathbb{Z}/km\mathbb{Z} \to \mathbb{Z}/m\mathbb{Z}$ defined by sending $a \pmod{km} \in \mathbb{Z}/km\mathbb{Z}$ to $a \pmod m \in \mathbb{Z}/m\mathbb{Z}$ has kernel $ker(f) = m\mathbb{Z}/km\mathbb{Z}$. By the first isomorphism theorem, we therefore have

$$(\mathbb{Z}/km\mathbb{Z})/(m\mathbb{Z}/km\mathbb{Z}) \cong \mathbb{Z}/m\mathbb{Z}.$$

Exercise 5.18.10. *Let S_n act on the set $\mathbb{Z}_n = \{1, 2, ..., n\}$ by permutations in the usual way. Let $x \in \mathbb{Z}_n$ be arbitary.*

(a) Describe the elements in the stabilizer $stab_{S_n}(x)$ as permutations in $2 \times n$ array form (recall (5.3) was the definition). Is this a normal subgroup of S_n?

(b) Compute the orbit $S_n * x$.

(c) Show that there is a 1-1 correspondence between the elements in the orbit $S_n * x$ and elements in $stab_{S_n}(x)$. (Hint: See the proof of Proposition 5.13.2.)

Exercise 5.18.11. *Let S_n act on the set $\mathbb{Z}_n = \{1, 2, ..., n\}$ by permutations in the usual way. Let $X = \mathbb{Z}_n^k$ denote the set of all k-tuples of $\mathbb{Z}_n$. The group S_n acts on X as well: if $x = (a_1, ..., a_k) \in X$ and $g \in S_n$ then we define $g * x = (g * a_1, ..., g * a_k)$. Show that the stabilizer $stab_{S_n}(x)$ is a normal subgroup of S_n if and only if x has the form $x = (i, i, ..., i) \in X$ for some $i \in \mathbb{Z}_n$. In this case, compute the quotient group $S_n/stab_{S_n}(x)$.*

Exercise 5.18.12. *Find all the normal subgroups N of A_4. For each one, compute A_4/N.*

5.19 Finite groups using GAP

5.19.1 Permutations

In GAP, to obtain the permutation matrix P corresponding to the cyclic permutation of $\mathbb{Z}_n$ which sends $a_1 \longmapsto a_2$, $a_2 \longmapsto a_3$, ..., $a_{n-1} \longmapsto a_n$, and $a_n \longmapsto a_1$, type **P:=PermutationMat((**a_1**,...,**a_n**),n);**. For example, to obtain the permutation matrix P corresponding to the permutation of $\mathbb{Z}_5$ which sends $1 \longmapsto 2$, $2 \longmapsto 3$, $3 \longmapsto 5$, and $5 \longmapsto 1$, type

`P:=PermutationMat((1,2,3,5),5);`. This is printed out as a list of row vectors. To get P to print out as a square matrix, type `PrintArray(P)`. To get the inverse permutation, type `PrintArray(P^(-1));`.

`a^(-1)` is the inverse. For example, if you type

```
a:=(1,2,3,5);
b:=(3,2,1)(4,5)
a*b;
a^(-1);
```

returns the product `(3,4,5)` and the inverse `(1,5,3,2)`.

Exercise 5.19.1. *(a) Compute* $(1,2,3,4,5,6,7,8,9)^3(1,2,3,4,5)^{-5}$.

(b) Make a 6×6 *table of all* 36 *products* ab*, where* a, b *run over all* 6 *permutations of* $\{1,2,3\}$.

5.19.2 Permutation groups

In GAP 4.1, `s8 := SymmetricGroup(8);` returns the GAP structure of the symmetric group on 8 letters. To find the number of elements of this group, type `Size( s8 );`.

To find if it is an abelian group, type `IsAbelian( s8 );`.

Exercise 5.19.2. *Create the symmetric group* S_n*,* $n = 3,4,5$*, and find out how many elements they have.*

In GAP, a permutation group can be entered using the **Group** command. For example, `s7 := Group( (1,2), (1,2,3,4,5,6,7) );` returns the GAP structure of the group generated by $(1,2)$ and $(1,2,3,4,5,6,7)$, which happens to be the symmetric group on 7 letters. To find the number of elements of this group, type `Size( s7 );`.

Exercise 5.19.3. *(a) Create the permutations group generated by* $(1,2)$ *and* $(1,2,3,4,5,6,7)$*, and find out how many elements they have.*

(b) Create the permutations group generated by a, b, c *in the section discussing Plain Bob Minimus and find out how many elements they have.*

Exercise 5.19.4. *Create the subgroup of* S_{10} *generated by* $(1,2,3,4,5,6,7,8)$ *and* $(8,9,10)$*. Compute* $(1,2,3,4,5,6,7,8)^4 * (8,9,10)^3$.

In GAP, to find the order of the move $R * F$ of the Rubik's cube, type

```
U:=( 1, 3, 8, 6)( 2, 5, 7, 4)( 9,33,25,17)(10,34,26,18)(11,35,27,19);
L:= ( 9,11,16,14)(10,13,15,12)( 1,17,41,40)( 4,20,44,37)( 6,22,46,35);
F:= (17,19,24,22)(18,21,23,20)( 6,25,43,16)( 7,28,42,13)( 8,30,41,11);
R:= (25,27,32,30)(26,29,31,28)( 3,38,43,19)( 5,36,45,21)( 8,33,48,24);
B:= (33,35,40,38)(34,37,39,36)( 3, 9,46,32)( 2,12,47,29)( 1,14,48,27);
D:= (41,43,48,46)(42,45,47,44)(14,22,30,38)(15,23,31,39)(16,24,32,40);
G:=Group(R,L,U,D,F,B);
Order(R*F);
```

Exercise 5.19.5. *Find the orders of $R*F$, $R*F*F*B$, $R*F*B*D*U*D$, in the Rubik's cube group*[4].

The Rotation game. Recall the 3×3 grid,

$$\begin{array}{ccc} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{array},$$

from §4.6. The allowed moves are rotations of the following form, or combinations thereof: In cycle notation, $r_1 = (a_1, a_2, a_5, a_4)$, $r_2 = (a_2, a_3, a_6, a_5)$, $r_3 = (a_4, a_5, a_8, a_9)$, or $r_4 = (a_5, a_6, a_9, a_8)$. In other words, a legal move in the Rotation game is any permutation of the form $r_{i_1} r_{i_2} ... r_{i_k}$, where $1 \leq i_j \leq 4$ for all $1 \leq j \leq N$.

Question: Can each permutation of $\{1, 2, ..., 9\}$ be expressed in the form $r_{i_1} r_{i_2} ... r_{i_k}$, where $1 \leq i_j \leq 4$ for all $1 \leq j \leq N$?

Exercise 5.19.6. *Answer this question.*

In GAP,

```
G:=Group((1,2,3), (1,2));
c:=RightCoset(G,(2,3,4));
Elements(c);
```

Exercise 5.19.7. *Let G be the group of symmetries of the square. Compute $G/\langle g_3 \rangle$.*

Exercise 5.19.8. *(a) Find the sign and swapping number of (1,4,7)(2,5).*
(b) Same for the Rubik's cube move B.

[4] Ans: 105, 6, 4, resp.

Exercise 5.19.9. *(a) Multiply (1,2,3,4,5,6,7) times (2,5) times (7,6,5,4,3,2,1).*
*(b) Multiply R*L*U*DD*F*B.*

Exercise 5.19.10. *(a) List all the elements in the group G generated by* $(1,2,3)$ *and* $(1,2)(3,4)$.
(b) List all the elements in the group S_4 generated by $(1,2)$ *and* $(1,2,3,4)$.
(c) How many positions of the Rubik's cube can be obtained by only using the moves R and F? (Hint: Consider the "two faces subgroup" of the Rubik's cube group generated by F and R.)

Let D_4 be the group geberated by $(1,2)(3,4)$, $(1,2,3,4)$.

Exercise 5.19.11. *Find all elements of order 2 in D_4.*

Exercise 5.19.12. *Find the conjugacy class of* $(2,4)$ *in the group D_4.*

Exercise 5.19.13. *(a) Find the conjugacy classes of the group S_3.*
(b) Find a complete set of representatives of each conjugacy class of S_5.

Exercise 5.19.14. *Obtain both left and right coset representatives of D_4 in S_4.*

Exercise 5.19.15. *Answer the question about the Rotation game in §4.6 using GAP.*

5.19.3 GAP project: Why Steinhaus' algorithm works

The argument is by mathematical induction.

1. First, note that Steinhaus' algortithm works for $n = 2, n = 3$.

2. Write down all the elements of S_{n-1} as a list using Steinhaus' algorithm (the induction hypthesis the each differs by a suitable transposition is assumed for this list). We represent each element in the $2 \times (n-1)$ array notation. In our list, we simply record the 2^{nd} row in the list. ie, as an $(n-1)$-tuple. There are $(n-1)!$ elements in this list.

3. For the 1^{st} $(n-1)$-tuple, write an n at the end, for the 2^{nd} $(n-1)$-tuple, write an n at the beginning, for the 3^{rd} $(n-1)$-tuple, write an n at the end, for the 4^{th} $(n-1)$-tuple, write an n at the beginning, and so on. Note we have a sequence of $(n-1)!$ n-tuples.

4. Suppose the i^{th} n-tuple is $(n, a_1, a_2, ..., a_{n-1})$. First, act on this by the transposition (n, a_1), then by the transposition (n, a_2), then by (n, a_3), ..., by (n, a_n). The result of the last one is $(a_1, a_2, ..., a_{n-1}, n)$. Suppose the j^{th} n-tuple is $(b_1, b_2, ..., b_{n-1}, n)$. (Note that by the induction hypothesis, $(a_1, a_2, ..., a_{n-1}, n)$ and $(b_1, b_2, ..., b_{n-1}, n)$ "differ" only by a suitable transposition, in S_{n-1}.) First, act on this by the transposition (b_1, n),then by the transposition (b_2, n), then by (b_3, n), ..., by (b_n, n). The result of the last one is $(n, b_1, b_2, ..., b_{n-1})$.

5. Note that all these (for i, j ranging over even or odd integers from 1 to $n-1$), the resulting $n!$ n-tuples are distinct. This yields a listing of S_n as desired.

Exercise 5.19.16. *Program this algorithm in GAP.*

5.20 Finite groups using MAGMA

5.20.1 Permutations

```
S5 := Sym(5);
a:=S5!(1,2,3,5);
b:=S5!(3,2,1)(4,5);
a*b;
a^(-1);
```

returns the product of the permutations, `(3,4,5)`, and the inverse, `(1,5,3,2)`. Here $S5$ denotes the set of all possible permutations of $\{1, 2, 3, 4, 5\}$.

Exercise 5.20.1. *(a) Compute* $(1, 2, 3, 4, 5, 6, 7, 8, 9)^3(1, 2, 3, 4, 5)^{-5}$.

(b) Make a 6×6 *table of all* 36 *products* ab*, where* a, b *run over all* 6 *permutations of* $\{1, 2, 3\}$.

5.20.2 Permutation groups

In MAGMA `S8 := SymmetricGroup(8);` (or `Sym(8)`) returns the symmetric group on 8 letters. To find the number of elements of this group, type `#S8;`.

To find if it is an abelian group, type `IsAbelian( S8 );`.

Exercise 5.20.2. *Create the symmetric group S_n, $n = 3, 4, 5$, and find out how many elements they have.*

In MAGMA, a permutation group can be entered using the **PermutationGroup** command. For example,
`S7 := PermutationGroup<7 | (1,2), (1,2,3,4,5,6,7)> ;` returns the symmetric group on 7 letters. To find the number of elements of this group, type `#S7;`.

Exercise 5.20.3. *(a) Create the permutations group generated by $(1, 2)$ and $(1, 2, 3, 4, 5, 6, 7)$, and find out how many elements they have.*

(b) Create the permutations group generated by a, b, c in the section discussing Plain Bob Minimus and find out how many elements they have.

Exercise 5.20.4. *Create the subgroup of S_{10} generated by $(1, 2, 3, 4, 5, 6, 7, 8)$ and $(8, 9, 10)$. Compute $(1, 2, 3, 4, 5, 6, 7, 8)^4 * (8, 9, 10)^3$.*

In MAGMA, to find the order of the move $R * F$ of the Rubik's cube, type

```
S48:=Sym(48);
U:=S48!( 1, 3, 8, 6)( 2, 5, 7, 4)( 9,33,25,17)(10,34,26,18)(11,35,27,19);
L:= S48!( 9,11,16,14)(10,13,15,12)( 1,17,41,40)( 4,20,44,37)( 6,22,46,35);
F:=S48!(17,19,24,22)(18,21,23,20)( 6,25,43,16)( 7,28,42,13)( 8,30,41,11);
R:=S48!(25,27,32,30)(26,29,31,28)( 3,38,43,19)( 5,36,45,21)( 8,33,48,24);
B:=S48!(33,35,40,38)(34,37,39,36)( 3, 9,46,32)( 2,12,47,29)( 1,14,48,27);
D:=S48!(41,43,48,46)(42,45,47,44)(14,22,30,38)(15,23,31,39)(16,24,32,40);
G:=PermutationGroup<48!U,L,F,R,B,D>;
Order(R*F);
```

Exercise 5.20.5. *Find the orders of $R*F$, $R*F*F*B$, $R*F*B*D*U*D$, in the Rubik's cube group*[5].

Exercise 5.20.6. *Answer the question about the Rotation game in §4.6 using MAGMA.*

In MAGMA,

[5] Ans: 105, 6, 4, resp.

```
H:=Sym(4);
G:=PermutationGroup<4 | (1,2,3),(1,2)>;
ElementSet(G, G);
h:=H!(2,3,4);
c:=G*h;
```

Exercise 5.20.7. *Let G be the group of symmetries of the square. Compute $G/\langle g_3 \rangle$.*

Sign(g) returns 1 if the permutation g is even, return -1 if g is odd.

Exercise 5.20.8. *(a) Find the sign and swapping number of (1,4,7)(2,5).*
(b) Same for the Rubik's cube move B.

Exercise 5.20.9. *(a) Multiply (1,2,3,4,5,6,7) times (2,5) times (7,6,5,4,3,2,1).*
*(b) Multiply R*L*U*DD*F*B.*

Exercise 5.20.10. *(a) List all the elements in the group G generated by $(1,2,3)$ and $(1,2)(3,4)$.*
(b) List all the elements in the group S_4 generated by $(1,2)$ and $(1,2,3,4)$.
(c) How many positions of the Rubik's cube can be obtained by only using the moves R and F? (Hint: Consider the "two faces subgroup" of the Rubik's cube group generated by F and R.)

Let D_4 be the group geberated by $(1,2)(3,4)$, $(1,2,3,4)$.

Exercise 5.20.11. *Find all elements of order 2 in D_4.*

Exercise 5.20.12. *Find the conjugacy class of $(2,4)$ in the group D_4.*

Exercise 5.20.13. *(a) Find the conjugacy classes of the group S_3.*
(b) Find a complete set of representatives of each conjugacy class of S_5.

Exercise 5.20.14. *Obtain both left and right coset representatives of D_4 in S_4.*

5.20.3 MAGMA project: Why Steinhaus' algorithm works

The argument is by mathematical induction.

1. First, note that Steinhaus' algortithm works for $n = 2, n = 3$.

2. Write down all the elements of S_{n-1} as a list using Steinhaus' algorithm (the induction hypthesis the each differs by a suitable transposition is assumed for this list). We represent each element in the $2 \times (n-1)$ array notation. In our list, we simply record the 2^{nd} row in the list. ie, as an $(n-1)$-tuple. There are $(n-1)!$ elements in this list.

3. For the 1^{st} $(n-1)$-tuple, write an n at the end, for the 2^{nd} $(n-1)$-tuple, write an n at the beginning, for the 3^{rd} $(n-1)$-tuple, write an n at the end, for the 4^{th} $(n-1)$-tuple, write an n at the beginning, and so on. Note we have a sequence of $(n-1)!$ n-tuples.

4. Suppose the i^{th} n-tuple is $(n, a_1, a_2, ..., a_{n-1})$. First, act on this by the transposition (n, a_1), then by the transposition (n, a_2), then by (n, a_3), ..., by (n, a_n). The result of the last one is $(a_1, a_2, ..., a_{n-1}, n)$. Suppose the j^{th} n-tuple is $(b_1, b_2, ..., b_{n-1}, n)$. (Note that by the induction hypothesis, $(a_1, a_2, ..., a_{n-1}, n)$ and $(b_1, b_2, ..., b_{n-1}, n)$ "differ" only by a suitable transposition, in S_{n-1}.) First, act on this by the transposition (b_1, n),then by the transposition (b_2, n), then by (b_3, n), ..., by (b_n, n). The result of the last one is $(n, b_1, b_2, ..., b_{n-1})$.

5. Note that all these (for i, j ranging over even or odd integers from 1 to $n-1$), the resulting $n!$ n-tuples are distinct. This yields a listing of S_n as desired.

Exercise 5.20.15. *Program this algorithm in MAGMA.*

Chapter 6

Special projects: Codes

We've explored several codes already - the Hamming codes, and a fairly broad class of codes called cyclic codes. In this chapter, we will meet some important codes which are in some ways a little more advanced than the ones we've seen already. These new codes will include the alternant codes, lexicodes, extended ternary Golay code, Goppa codes, and low density parity check codes. These sections can be used as a basis for a semester project.

6.1 Alternant codes

Let $\alpha = \{\alpha_1, ..., \alpha_n\}$ be a set of distinct elements of $\mathbb{F}_{q^m}$ and let $h = \{h_1, ..., h_n\}$ be a set of non-zero elements of $\mathbb{F}_{q^m}$. Fix a vector space basis for $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. For $x \in \mathbb{F}_{q^m}$, let $[x]$ denote the column vector representation of x with respect to this basis. Let

$$H' = \begin{pmatrix} [h_1] & [h_2] & \dots & [h_n] \\ [h_1\alpha_1] & [h_2\alpha_2] & \dots & [h_n\alpha_n] \\ \vdots & & & \vdots \\ [h_1\alpha_1^{r-1}] & [h_2\alpha_2^{r-1}] & \dots & [h_n\alpha_n^{r-1}] \end{pmatrix}.$$

Assume $r < n$. Delete any linearly dependent rows. Call the resulting matrix H.

Definition 6.1.1. *An* **alternant code** $\mathcal{A}(\alpha, h)$ *is a code over* $\mathbb{F}_q$ *with parity check matrix of the form* H*, as above.*

Exercise 6.1.2. *Compute* $\mathcal{A}(\alpha, h)$ *over* $\mathbb{F}_9 = \mathbb{F}_3[x]/(x^2+1)$, *where* $\alpha = \{x, x+1, 2x+1\}$ *and* $h = \{1, x, 2x\}$.

Exercise 6.1.3. *Compute all the vectors in an alternant code of your choosing. Verify the following properties.*

- $n - mr \leq \dim\mathcal{A}(\alpha, h) \leq n - r$,
- *the minimum distance of* $\mathcal{A}(\alpha, h)$ *is at least* $r + 1$,
- *the length of* $\mathcal{A}(\alpha, h)$ *is* n.

6.2 Lexicodes

Lexicodes were introducted by Conway and Sloane in [CS]. They form a class of binary codes which are (a) non necessarily linear, (b) in general hard to construct, (c) in general hard to encode and decode. Why study them then? There are several reasons: (a) they often times are linear, (b) they have excellent parameters, (c) one might hope that with further investigation, someone (maybe the person reading this) may in the future discover fast ending and decoding algorithms. One other reason: they have fascinating connections to combinatorial game theory (such as the game of nim, which we have seen already).

6.2.1 The construction

First, suppose v, w are two binary vectors of length n. We say $v < w$ in the **lexicographical ordering** if (a) $v_1 < w_1$, or (b) $v_1 = w_1$ and $v_2 < w_2$, or (c) $v_1 = w_1$ and $v_2 = w_2$ and $v_3 < w_3$, or ... . For example, $(1, 1, 0, 1) < (1, 1, 1, 0)$,

First, we want an algorithm which takes a list L of binary vectors of length n and adds to that list the smallest (in the lexicographical ordering) vector of length n which is not in L and which has distance at least d from an element of L.

Here's the greedy code algorithm to construct a binary lexicode of length n and minimum distance d.

Input: Integers $n > d > 0$. Output: A binary code of length n and minimum distance d.

1. Construct the list L of all binary n-tuples, ordered lexicographically.

2. Let $C = \{(0, ..., 0)\}$.

3. Let c denote the first element (in the lex ordering) of L which is at least distance d from any other element of C, if it exists. If c does not exist then stop and return C.

4. $C = C \cap \{c\}$

5. Go to step 3.

Here's some GAP-like "pseudo-code" in more detail.

```
add_vector:=function(L,n,d)
 #adds smallest binary vector of length n
 #which is at least distance d from any other
 L0:=[]:
 for i from 0 to 2^n-1 do
  LL:=binary representation of i;
  L0:={LL} union L0:
 end for;
 L1:=sort L0 lexicographically;
 for v in L1 do
  L2:=[];
  for y in L1 do
   if y<>v then L2:=[op(L3),y]; fi;
  end for;
  m:=minimum of all Hamming weights of v-w, w in L
  if m >= d then
   RETURN(L union v);
  end if;
  end for;
 RETURN(L);
 end function;
```

Next, we need a function which iterates this `add_vector` procedure over and over.

```
lexicode_binary:=function(M,n,d)
 #M is the number of elements, n is the length,
 #d is the min distance
 L:=[];
 for i from 1 to M do
  L:=add_vector(L,n,d):
 end for;
 RETURN(L);
 end function;
```

For example, `L:=lexicode_binary(16,7,3);` returns

$$\{[0,\,0,\,0,\,0,\,0,\,0,\,0],\ [1,\,1,\,1,\,0,\,0,\,0,\,0],$$
$$[1,\,0,\,0,\,1,\,1,\,0,\,0],\ [0,\,1,\,0,\,1,\,0,\,1,\,0],$$
$$[0,\,0,\,1,\,0,\,1,\,1,\,0],\ [0,\,0,\,1,\,1,\,0,\,0,\,1],$$
$$[0,\,1,\,0,\,0,\,1,\,0,\,1],\ [1,\,0,\,0,\,0,\,0,\,1,\,1],$$
$$[0,\,1,\,1,\,1,\,1,\,0,\,0],\ [1,\,0,\,1,\,1,\,0,\,1,\,0],$$
$$[1,\,1,\,0,\,0,\,1,\,1,\,0],\ [1,\,1,\,0,\,1,\,0,\,0,\,1],$$
$$[1,\,0,\,1,\,0,\,1,\,0,\,1],\ [0,\,1,\,1,\,0,\,0,\,1,\,1],$$
$$[0,\,0,\,0,\,1,\,1,\,1,\,1],\ [1,\,1,\,1,\,1,\,1,\,1,\,1]\}$$

Exercise 6.2.1. *What does* `L:=lexicode_binary(16,8,3);` *return? (Go through every step.)*

Some more examples of lexicodes.

- with n=8, d=3:

$$C = \{[0,0,0,0,0,0,0,0],\ [1,1,1,0,0,0,0,0],$$
$$[1,0,0,1,1,0,0,0],\ [0,1,1,1,1,0,0,0],$$
$$[0,1,0,1,0,1,0,0],\ [1,0,1,1,0,1,0,0],$$
$$[1,1,0,0,1,1,0,0],\ [0,0,1,0,1,1,0,0],$$
$$[1,1,0,1,0,0,1,0],\ [0,0,1,1,0,0,1,0],$$
$$[0,1,0,0,1,0,1,0],\ [1,0,1,0,1,0,1,0],$$
$$[1,0,0,0,0,1,1,0],\ [0,1,1,0,0,1,1,0],$$
$$[0,0,0,1,1,1,1,0],\ [1,1,1,1,1,1,1,0]\}$$

- with n=9, d=3:

$$C = \{[0,0,0,0,0,0,0,0,0], [1,1,1,0,0,0,0,0,0],\\ [1,0,0,1,1,0,0,0,0], [0,1,1,1,1,0,0,0,0],\\ [0,1,0,1,0,1,0,0,0], [1,0,1,1,0,1,0,0,0],\\ [1,1,0,0,1,1,0,0,0], [0,0,1,0,1,1,0,0,0],\\ [1,1,0,1,0,0,1,0,0], [0,0,1,1,0,0,1,0,0],\\ [0,1,0,0,1,0,1,0,0], [1,0,1,0,1,0,1,0,0],\\ [1,0,0,0,0,1,1,0,0], [0,1,1,0,0,1,1,0,0],\\ [0,0,0,1,1,1,1,0,0], [1,1,1,1,1,1,1,0,0]\}$$

- with n=10, d=3:

$$C = \{[0,0,0,0,0,0,0,0,0,0], [1,1,1,0,0,0,0,0,0,0],\\ [1,0,0,1,1,0,0,0,0,0], [0,1,1,1,1,0,0,0,0,0],\\ [0,1,0,1,0,1,0,0,0,0], [1,0,1,1,0,1,0,0,0,0],\\ [1,1,0,0,1,1,0,0,0,0], [0,0,1,0,1,1,0,0,0,0],\\ [1,1,0,1,0,0,1,0,0,0], [0,0,1,1,0,0,1,0,0,0],\\ [0,1,0,0,1,0,1,0,0,0], [1,0,1,0,1,0,1,0,0,0],\\ [1,0,0,0,0,1,1,0,0,0], [0,1,1,0,0,1,1,0,0,0],\\ [0,0,0,1,1,1,1,0,0,0], [1,1,1,1,1,1,1,0,0,0],\\ [1,0,0,0,0,0,0,1,1,0], [0,1,1,0,0,0,0,1,1,0],\\ [0,0,0,1,1,0,0,1,1,0], [1,1,1,1,1,0,0,1,1,0],\\ [1,1,0,1,0,1,0,1,1,0], [0,0,1,1,0,1,0,1,1,0],\\ [0,1,0,0,1,1,0,1,1,0], [1,0,1,0,1,1,0,1,1,0],\\ [0,1,0,1,0,0,1,1,1,0], [1,0,1,1,0,0,1,1,1,0],\\ [1,1,0,0,1,0,1,1,1,0], [0,0,1,0,1,0,1,1,1,0],\\ [0,0,0,0,0,1,1,1,1,0], [1,1,1,0,0,1,1,1,1,0],\\ [1,0,0,1,1,1,1,1,1,0], [0,1,1,1,1,1,1,1,1,0]\}$$

These are all vectors spaces, the first three of dimension 4 and the last of dimension 5.

Exercise 6.2.2. *What does* `L:=lexicode_binary(16,8,3);` *return? (Go through every step.)*

Exercise 6.2.3. *Construct the lexicode of length* $n = 5$ *and* $d = 2$.

Exercise 6.2.4. *Construct the lexicode of length* $n = 6$ *and* $d = 2$.

Exercise 6.2.5. *Construct the lexicode of length $n = 5$ and $d = 3$.*

Exercise 6.2.6. *The first one (a) is equivalent to the Hamming (7,4,3)-code .*

6.3 Tanner graph of a code

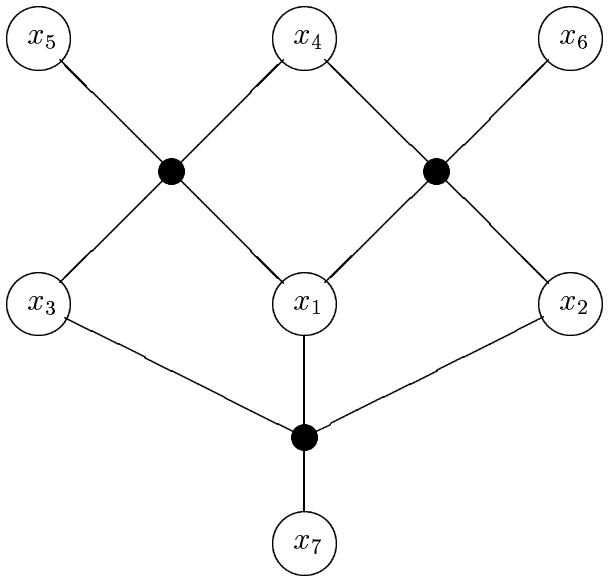
Let H be an $r \times n$ parity check matrix for a binary linear code C.

A **biparite graph** is a graph $\Gamma = (V, E)$, where V denotes the vertices and E the edges, having the property that V is a disjoint union of two subsets $V = V_1 \cup V_2$ and each edge in E connects an element in V_1 to an element in V_2.

Definition 6.3.1. *The* **Tanner graph** *of C is the bipartite graph with $n+r$ vertices, the n* **message vertices** *are labeled by the coordinates of C (x_1, .., x_n, say), and the r* **check vertices** *are unlabeled but are indexed by the rows [1] of H. The i^{th} check vertex is connected by an edge to the x_j message vertex if and only if the $(i,j)^{th}$ entry of H is non-zero (i.e., if and only if x_j occurs in the i^{th} parity check equation defining C).*

Example 6.3.2. *Tanner graph for the binary Hamming $(7,4,3)$-code in §3.4.2.*

[1] The i^{th} row of H corresponds to the i^{th} parity check equation defining C, hence the name "check vertex".



The parity check equations correspond to the solid black vertices, the coordinates of the codes to the labeled vertices, and the edges correspond to the terms occurring in the parity check equation.

Exercise 6.3.3. *Find the Tanner graph of the binary code with parity check matrix*

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

6.4 A brief guide to Goppa codes

Goppa codes, or algebraic geometry codes (also called AG codes), were discovered by V. D. Goppa in the early 1980's [Go]. Other Russian mathematicians contributed to their theory, such as Yu. I. Manin, M. A. Tsfasman, and S. G. Vladut. Certain Goppa codes arising from "modular curves" gave the first infinite family of codes whose parameters beat the Gilbert-Varshamov bound. This was quite an interesting result since before then no infinite familty of codes were known having this property.

In this note we sketch the terrain of AG codes arising from curves over finite fields using MAGMA [MAGMA].

There are two types of Goppa codes: "function codes" and "primary (or residue) codes". One is equivalent to the dual code of the other, so shall

only go into some detail about one of them. Each of them requires a certain amount of data to begin: one must

- choose a finite field $\mathbb{F}$,
- choose a smooth algebraic curve C over $\mathbb{F}$,
- pick rational points $P_1, P_2, ..., P_n$ in $C(\mathbb{F})$,
- choose a (not necessarily rational) divisor D distinct from the P_i's,
- determine a basis for the "Riemann-Roch space" $L(D)$.

Undefined are "smooth algebraic curve", "rational points", "divisors", "Riemann-Roch space". A **divisor** is simply a (formal) finite sum of points of $C(\overline{\mathbb{F}})$, where $\overline{\mathbb{F}}$ is an algebraic closure of $\mathbb{F}$ (see Definition 2.1.7) and $C(\overline{\mathbb{F}})$ is defined below. The other terms are more technical to define and we shall stick to examples and how to do computations with the (using [MAGMA]) rather that formal definitions.

For more precise details, see Pretzel's book [P]. Below, we shall focus on what is needed to work out examples using MAGMA.

6.4.1 Examples of curves

A **curve** is an "absolutely irreducible projective curve defined over a field field". Such a curve is determined by a polynomial $f(x, y) \in \mathbb{F}[x, y]$ of degree d which is irreducible over $\overline{\mathbb{F}}$. The projective curve C is composed of three "affine components"

$$\begin{aligned} C_1 : &\qquad f(x, y) = 0, \\ C_2 : &\qquad g(u, v) = 0, \\ C_3 : &\qquad h(w, z) = 0, \end{aligned}$$

where

$$g(u, v) = v^d f(1/v, u/v), \qquad h(w, z) = w^d f(z/w, 1/w).$$

The **projective version** of C is

$$z^d f(x/z, y/z) = 0.$$

We shall sometimes abuse notation and confuse C and C_1.

The **Klein quartic** is given by $f(x,y) = x^3y + y^3 + x$. The projective version is $x^3y + zy^3 + xz^3 = 0$. (You can compute f, g, and h above by setting $z = 1$, $y = 1$, and $x = 1$, respectively, in the projective version.)

The **Fermat curve** (of degree n) is given by $f(x,y) = x^n + y^n - 1$. The projective version is $x^n + y^n = z^n$.

The **Hermitian curve** is a special case of the Fermat curve. It is defined over $\mathbb{F} = \mathbb{F}_{q^2}$ and is given by $f(x,y) = x^{q+1} + y^{q+1} + 1$.

Exercise 6.4.1. *(a) Compute f, g, and h for the Klein quartic.*
(b) Compute f, g, and h for the Fermat curve.
(c) Compute f, g, and h for the Hermitian curve.

Curves in MAGMA

```
> A<x,y> := AffineSpace(FiniteField(5),2);
> f:=x^3*y+y^3+x;
> C:=Curve(A,f);
> C;
Curve over GF(5) defined by
x^3*y + x + y^3
> D<X,Y,Z> := ProjectiveClosure(C);
>
> D;
Curve over GF(5) defined by
X^3*Y + X*Z^3 + Y^3*Z
```

6.4.2 Examples of points and divisors

The set of all **rational points** of C is the set

$$C(\mathbb{F}) = C_1(\mathbb{F}) \cup C_2(\mathbb{F}) \cup C_3(\mathbb{F}),$$

where

$$C_1(E) = \{(x,y) \in E^2 \mid f(x,y) = 0\},$$
$$C_2(E) = \{(u,v) \in E^2 \mid g(u,v) = 0\},$$
$$C_3(E) = \{(w,z) \in E^2 \mid h(w,z) = 0\},$$

and E is any extension of $\mathbb{F}$. The coordinate map $u = y/x, v = 1/x$ maps C_1 with $x \neq 0$ to C_2, and the coordinate map $w = 1/y, v = x/y$ maps C_1 with

$y \neq 0$ to C_3. These give rise to coordinate maps $\phi : C_1 \to C_2$, $\phi' : C_1 \to C_3$, Generally, there are coordinate maps $\phi_{ij} : C_i \to C_j$, for each $i \neq j$, obtained by composing these maps ϕ and ϕ' and their inverses.

Of course, if a point P of one component is sent to a point P' in a different component by one of these coordinate maps then we regard P and P' to be the same element of $C(\mathbb{F})$.

The concept of "smoothness" is a restriction on the function f: we call C **smooth** (over $\overline{\mathbb{F}}$) if

$$(\frac{\partial f}{\partial x}(x,y), \frac{\partial f}{\partial y}(x,y)) \neq (0,0),$$

for all $(x,y) \in C_1(\overline{\mathbb{F}})$, and

$$(\frac{\partial g}{\partial u}(u,v), \frac{\partial g}{\partial v}(u,v)) \neq (0,0),$$

for all $(u,v) \in C_2(\overline{\mathbb{F}})$,

$$(\frac{\partial h}{\partial w}(w,z), \frac{\partial h}{\partial z}(w,z)) \neq (0,0),$$

for all $(w,z) \in C_3(\overline{\mathbb{F}})$.

Example: The point $p = (3,2)$ is a rational point of the Klein quatric over $\mathbb{F}_5$.

Points using MAGMA

MAGMA calls rational points of a curve C "places of degree 1". MAGMA will find all points in all affine components at the same time. A point in the affine component C_1 is denoted $(a : b : 1)$. A point in the affine component C_2 is denoted $(1 : a : b)$. A point in the affine component C_3 is denoted $(a : 1 : b)$.

```
> Places(C,1);
[
    Place at (0 : 1 : 0),
    Place at (0 : 0 : 1),
    Place at (1 : 0 : 0),
    Place at (4 : 2 : 1),
```

```
    Place at (3 : 2 : 1),
    Place at (3 : 4 : 1)
]
> p:=A![3,2];
> p in C;
true (3, 2)
```

Divisors can be defined as formal sums of points:

```
> P1:=Places(C,1);
> D:=P1[2]+P1[3];
> Div := DivisorGroup(C);
> D0:=Div!D;
> D0;
Divisor 1*Place at (0 : 0 : 1) + 1*Place at (1 : 0 : 0)
> D:=2*P1[2]+P1[3];
> D0:=Div!D;
> D0;
Divisor 2*Place at (0 : 0 : 1) + 1*Place at (1 : 0 : 0)
```

6.4.3 Examples of Riemann-Roch spaces

Before discussing the Riemann-Roch spaces $L(D)$, let us begin by motivating them from the point of view of error-correcting codes.

Suppose that we have a curve C over a finite field $\mathbb{F}$ a finite set of points $\{P_1, ..., P_n\}$, and a finite dimensional space of functions ϕ on C, V. With this set-up, we can define a linear code C by

$$GC = \{(\phi(P_1), ..., \phi(P_n)) \mid \phi \in V\}.$$

This is easy enough. One problem is: how to we find a finite-dimensional vector space of functions V?

As you probably guessed, this is where the Riemann-Roch spaces come into play. The spaces $L(D)$ are finite dimensional vector spaces of rational functions on C. Moreover, if the divisor D is disjoint from the points P_i then ϕ is defined at the points P_i, for all $\phi \in L(D)$. Roughly speaking, if $D = n_1Q_1 + ... + n_rQ_r$ then $L(D)$ is the space of rational functions on C which have do not have poles of order higher than n_i at Q_i, $1 \leq i \leq r$.

Example 6.4.2. The projective line *Let $X = \mathbb{P}^1/F$, where F is algebraically closed.*

Most of the examples below were computed using MAGMA 2.8 [MAGMA].

Let $F = \overline{\mathbb{F}_5}$. Let $P_1 = (1:0)$, $P_2 = (1:1)$, $P_3 = (1:2)$, $P_4 = (1:3)$, $P_5 = (1:4)$, $\infty = (0:1)$.

D	*basis for* $L(D)$
$2\infty - 2P_3$	$(x+3)^2$
$\infty - 2P_3$	$\emptyset$
$2\infty - 2P_1$	x^2
$3\infty - 2P_1$	x^2, x^3
$3\infty - 2P_5$	$x(x+1)^2, (x+1)^2$
$6P_1 - 3P_3 - 2P_5$	$x^{-5}(x+3)^3(x+1)^2$, $x^{-6}(x+3)^3(x+1)^2$
$7P_1 - 3P_3 - 2P_5$	$x^{-5}(x+3)^3(x+1)^2$, $x^{-6}(x+3)^3(x+1)^2$, $x^{-7}(x+3)^3(x+1)^2$
$-2\infty + P_1 - 3P_3 + 2P_5$	$\emptyset$
$-2\infty + 3P_1 - 3P_3 + 3P_5$	$x^{-2}(x+1)^{-3}(x+3)^3$, $(x+1)^{-3}x^{-3}(x+3)^3$
$-2\infty + 3P_2 - 3P_3 + 3P_5$	$x(x+1)^{-3}(x+4)^{-3}(x+3)^3$, $(x+1)^{-3}(x+4)^{-3}(x+3)^3$
$P_1 + 3P_3 - 2P_5$	$x(x+3)^{-3}(x+1)^2$, $(x+3)^{-3}(x+1)^2$, $x^{-1}(x+3)^{-3}(x+1)^2$
$-P_1 + 3P_2 - 2P_3$	$x(x+3)^{-3}(x+1)^2$
$7\infty - 2P_3 - 2P_5$	$x^3(x+3)^2(x+1)^2$, $x^2(x+3)^2(x+1)^2$, $x(x+3)^2(x+1)^2$, $(x+3)^2(x+1)^2$
$3\infty + 3P_2 - 2P_3 - 2P_5$	$p(x) = x^2(x+4)^{-3}(x+3)^2(x+1)^2$, $xp(x)$, $x^2p(x)$

Under certain fairly general conditions, a basis for the Riemann-Roch spaces $L(D)$ can be computed.

Let C be the Klein quartic over $\mathbb{F}_5$ and let $D = 2(4,2) + 2(3,2)$. Then $L(D)$ is 2-dimensional having basis

$$\phi_1 = \tfrac{y}{y+3}, \quad \phi_2 = \tfrac{1}{y+3}.$$

Riemann-Roch spaces in MAGMA

Once the curve C and divisor D are defined, it is an easy matter to get MAGMA to compute $L(D)$:

```
> D:=2*P1[2]+2*P1[3];
> D0:=Div!D;
> D0;
Divisor on Curve over GF(5) defined by
x^3*y + x + y^3
> RiemannRochSpace(D0);
KModule of dimension 2 over GF(5)
Mapping from: KModule of dimension 2 over GF(5) to Algebraic function field
defined over GF(5) by
x^3*y + x + y^3
> B0:=Basis(D0);
> B0;
[ 1, 1/$.2 ]
> D:=2*P1[4]+2*P1[5];
> D0:=Div!D;
> D0;
Divisor on Curve over GF(5) defined by
x^3*y + x + y^3
> RiemannRochSpace(D0);
KModule of dimension 2 over GF(5)
Mapping from: KModule of dimension 2 over GF(5) to Algebraic function field
defined over GF(5) by
x^3*y + x + y^3
> B0:=Basis(D0);
> B0;
[ $.2/($.2 + 3), 1/($.2 + 3) ]
```

Here `$.1` means x and `$.2` denotes y.

6.4.4 Examples of Goppa codes

Suppose that we have a curve C over a finite field $\mathbb{F}$ a finite set of points $\mathcal{P} = \{P_1, ..., P_n\}$, and a divisor D disjoint from the P_i. With this set-up, we define the **Goppa code** $GC = GC(\mathcal{P}, D, C)$ by

$$GC = \{(\phi(P_1), ..., \phi(P_n)) \mid \phi \in L(D)\}.$$

This is the **function code** version. The **primary code** (or **residue code**) is defined by

$$GC' = \{c \in \mathbb{F}^n \mid c \cdot (\phi(P_1), ..., \phi(P_n)) = 0,\ \forall \phi \in L(D)\}.$$

For example, let C be the Klein quartic over $\mathbb{F}_5$, let $\mathcal{P} = (0:1:0) + (0:0:1) + (3:4:1)$, and let $D = 2*(4:2:1) + 3*(3:2:1)$. Then $L(D)$ is 3-dimensional, with basis

$$\phi_1(x,y) = \frac{xy+4}{y^2+y+4}, \quad \phi_2(x,y) = \frac{y+2}{y+3}, \quad \phi_3(x,y) = \frac{1}{y+3}.$$

The Goppa code is a $(3,2,2)$ linear code over $\mathbb{F}_5$ having generator matrix:

$$G = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 4 \end{pmatrix}.$$

Examples of Goppa codes in MAGMA

```
> A<x,y> := AffineSpace(FiniteField(5),2);
> f:=x^3*y+y^3+x;
> C:=Curve(A,f);
> P1:=Places(C,1);
> Div := DivisorGroup(C);
> P1;
[
    Place at (0 : 1 : 0),
    Place at (0 : 0 : 1),
    Place at (1 : 0 : 0),
    Place at (4 : 2 : 1),
    Place at (3 : 2 : 1),
    Place at (3 : 4 : 1)
]
> D:=P1[2]+P1[3];
> P2:=[P1[1],P1[2],P1[6]];
> D:=2*P1[4]+3*P1[5];
> D0:=Div!D;
> D0;
Divisor on Curve over GF(5) defined by
x^3*y + x + y^3
> RiemannRochSpace(D0);
KModule of dimension 3 over GF(5)
Mapping from: KModule of dimension 3 over GF(5) to Algebraic function field
```

```
defined over GF(5) by
x^3*y + x + y^3
> Basis(D0);
[ ($.1*$.2 + 4)/($.2^2 + $.2 + 4), ($.2 + 2)/($.2 + 3), 1/($.2 + 3) ]
> GC:=AlgebraicGeometricCode(P2,D0);
> GC;
[3, 2, 2] Linear Code over GF(5)
Generator matrix:
[1 0 2]
[0 1 4]
```

6.5 Brief guide to low density parity check codes

In some sense one might say that this section is about the "codes of the future". Low density parity check (LDPC) codes were introduced by R. Gallagher in his 1960 MIT PhD thesis [Ga]. At the time, little attention was paid to them due to their impracticality with the current technolgy. However, Gallagher showed that a large proportion of them have "good parameters", i.e., parameters approaching the Shannon limit [MN], [Mac]. Recently, their practicality has improved dramatically and in some cases linear time encoding and decoding algorithms have been established for them [Sp].

There are two ways to define LDPC codes, one using matrices and another using graphs. We shall discuss both.

6.5.1 Sparse check matrix definition

Let H is an $r \times n$ matrix over $\mathbb{F}_2$ with the following properties:

- each row has exactly ρ 1's,
- each column has exactly γ 1's,
- the number of 1's in common between any two columns, denoted, δ, satisfies $\delta \leq 1$,
- ρ/n and γ/r are "small".

The linear binary code C is defined by

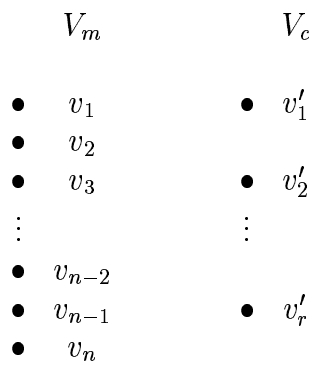
$$C = \{c \in \mathbb{F}_2^n \mid Hc = 0\}.$$

In other words, H is the parity check matrix for C. Since H is sparse, C is called a **low density parity check code** or a **LDPC code**.

More generally, such codes are called **regular** LDPC codes. We shall not discuss "irregular LDPC codes" here, which are defined by somewhat weaker conditions.

6.5.2 Graph-theoretic definition

Let Γ be a bipartite graph whose vertices V partition into two subsets V_m and V_c. The vertices in V_m are called **message nodes**, $V_m = \{v_1, ..., v_n\}$. The vertices in V_c are called **check nodes**, $V_c = \{v'_1, ..., v'_r\}$.



Suppose that each of the message nodes have degree $\rho \geq 1$ and each of the check nodes have degree $\gamma \geq 1$. For $v \in V_m$, let $N(v)$ denote all the neighbors $v' \in V_c$ of v. Define $x = (x_1, ..., x_n) \in \mathbb{F}_2^n$ to be a **codeword** if and only if, for all $v \in V_c$, we have

$$\sum_{v_i \in N(v)} x_i = 0.$$

(Equivalently, if H is the incidence matrix of Γ whose rows are indexed by the check nodes and whose columns are indexed by the message nodes, then x is a codeword if and only if $Hx = 0$.)

Cayley graph construction

Here's an example.

Let G be a finite group of order n generated by $S = \{s_1, ..., s_d\}$. Let Γ denote the Cayley graph of (G, S) and let H be the incidence matrix of Γ. Then the matrix $H = (P_{ij})$ is the parity check matrix of a LDPC code, provided d is "small" with respect to n. See [LR] and the references cited there for more details on such constructions.

Here's a small example using MAGMA. (Note that MAGMA does not have the ability to create a code from its parity check matrix, so we must treat the parity check matrix as the generator matrix of the dual code.)

```
> S3:=PermutationGroup< 3 | (1,2),(1,2,3)>;
> Gamma:=CayleyGraph(S3);
> Gamma;
Digraph
Vertex  Neighbours

1       3 4 ;
2       3 4 ;
3       1 6 ;
4       2 5 ;
5       1 6 ;
6       2 5 ;

> I:=IncidenceMatrix(Gamma);
> I;
[ 1  1  0  0 -1  0  0  0 -1  0  0  0]
[ 0  0  1  1  0  0 -1  0  0  0 -1  0]
[-1  0 -1  0  1  1  0  0  0  0  0  0]
[ 0 -1  0 -1  0  0  1  1  0  0  0  0]
[ 0  0  0  0  0  0  0 -1  1  1  0 -1]
[ 0  0  0  0  0 -1  0  0  0 -1  1  1]
> NumberOfColumns(I);
12
> NumberOfRows(I);
6
> M := KMatrixSpace(FiniteField(2), 6,12);
> G:=M!I;
> C1:=LinearCode(G);
> C:=Dual(C1);
```

```
> Dimension(C);
7
> MinimumDistance(C);
2
> C;
[12, 7, 2] Linear Code over GF(2)
Generator matrix:
[1 0 0 0 0 1 0 0 1 0 0 1]
[0 1 0 0 0 0 0 1 1 0 0 0]
[0 0 1 0 0 1 0 0 0 0 1 0]
[0 0 0 1 0 0 0 1 0 0 1 1]
[0 0 0 0 1 1 0 0 1 0 0 1]
[0 0 0 0 0 0 1 1 0 0 1 1]
[0 0 0 0 0 0 0 0 0 1 0 1]
```

6.5.3 Other constructions

There are several other constructions which gives rise to LDPC codes.

Block matrices of permutation matrices

Let $\{P_{ij}\}$ be a collection of $a \times a$ permutation matrixes, where $1 \leq i \leq k$ and $1 \leq j \leq \ell$. Then the $ak \times a\ell$ matrix $H = (P_{ij})$ is (often) the parity check matrix of a LDPC code. See [FST] and the references cited there for more details on such constructions.

Here's an example using GAP.

```
gap> RequirePackage("guava");

   ----                             |
  /            \           /   --+--  Version 1.5
 /      |    | |\\        //|    |
|    _  |    | | \\      // |     Jasper Cramwinckel
|     \ |    | |--\\    //--|     Erik Roijackers
 \     ||    | |   \\  //   |     Reinald Baart
  \___/  \___/ |    \\//    |     Eric Minkes
                                   Lea Ruscio
true
gap>
gap> random_ldpc_code:=function(n,r,a)
>     ## n = length
>     ## r = number of checks
>     ## a must divide gcd(n,r)
>     ## creates r1xn1 block matrix of
>     ## random axa permutation matrices
>     local C,i,j,H,P,G,p;
> G:=SymmetricGroup(a);
> if GcdInt(n,r)/a <> Int(Gcd(n,r)/a) then
>     Print("Wrong data\n");
>     return [n,r,a];
> fi;
> p:=[];
> for i in [1..Int(r/a)] do
>     p[i]:=[];
>     for j in [1..Int(n/a)] do
>         p[i][j]:=PermutationMat(Random(G),a);
>     od;
> od;
> P:=List([1..Int(r/a)],i->List([1..Int(n/a)],j->[i,j,p[i][j]]));
> H:=BlockMatrix(P[1],Int(r/a),Int(n/a));
> C:=CheckMatCode(H,GF(2));
> return C;
> end;
function( n, r, a ) ... end
gap>
gap> C:=random_ldpc_code(8,4,2);
a linear [8,6,1..2]1..2 code defined by check matrix over GF(2)
gap> MinimumDistance(C);
2
gap> Display(CheckMat(C));
 1 . 1 . 1 . 1 .
 . 1 . 1 . 1 . 1
gap> C:=random_ldpc_code(15,6,3);
a linear [15,12,1..2]1..3 code defined by check matrix over GF(2)
gap> MinimumDistance(C);
2
gap> Display(CheckMat(C));
 1 . . . . 1 . 1 . . 1 . 1 . .
 . 1 . . 1 . 1 . . 1 . . . 1 .
 . . 1 1 . . . . 1 . . 1 . . 1
```

Finite geometries construction

For background on finite geometries, we refer to Assumus, Key [AK].

Let Γ be a finite geometry with n points, $\{P_1, ..., P_n\}$, and r lines, $\{L_1, ..., L_r\}$, such that

- each line has exactly ρ points,
- each point is contained in exactly γ lines,
- any two (distinct) points are contained in exactly one line,
- two points are either parallel (i.e., have no points in common) or have exactly one point in common.

Let $H = (h_{ij})$ be the $r \times n$ matrix, where $h_{ij} = 1$ if the i^{th} line L_i of Γ contains the j^{th} point P_j of Γ, and where $h_{ij} = 0$ otherwise. Then the matrix $H = (h_{ij})$ is the parity check matrix of a LDPC code, provided ρ/n and γ/r are "small". See [FKL] and the references cited there for more details on such constructions.

Exercise 6.5.1. *Using GAP or MAGMA, find the minimum distance of the (binary) LDPC code having*

$$H = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

as its parity check matrix.

Exercise 6.5.2. *Using GAP or MAGMA, find the minimum distance of the (binary) LDPC code using the Cayley graph of* $S_4 = \langle (1,2), (1,2,3,4) \rangle$ *in place of* $S_3 = \langle (1,2), (1,2,3) \rangle$ *as used in the above example.*

6.6 Decoding the ternary Golay code

In this section, we provide an algorithm which uses a combinatorial device called the "MINIMOG" to decode the ternary extended Golay code. Though the algorithm appears to be new and relatively simple, we do not have a proof that it always yields the correct answer. (On the other hand, we know of no counterexample either!)

6.6.1 Introduction

An *m-(sub)set* is a (sub)set with m elements. For integers $k < m < n$, a *Steiner system* $S(k, m, n)$ is an n-set X and a set S of m-subsets having the property that any k-subset of X is contained in exactly one m-set in S. For example, if $X = \{1, 2, ..., 12\}$, a Steiner system $S(5, 6, 12)$ is a set of 6-sets, called *hexads*, with the property that any set of 5 elements of X is contained in ("can be completed to") exactly one hexad.

In [BG], a method using the MOG ("miracle octad generator") is used to decode the binary extended Golay code C_{24}. In this paper, we present and analogous algorithm to decode the ternary extended Golay code C_{12}. The MOG is a device discovered by Curtis (see for example, [CS], chapter 11) which helps one compute octads in the Steiner system $S(5, 8, 24)$. The MINIMOG is a device (discovered by Conway, [CS], chapter 11) which helps one compute hexads in the Steiner system $S(5, 6, 12)$.

Another algorithm was presented in Pless [Pl], based on Conway's "tetracode" construction of C_{12}. (We shall recall the "tetracode" construction below.) The algorithm presented here is different from that in [Pl].

We recall some facts from [CS].

C_{24} is a linear code over $GF(2)$ which is dimension 12 and minimum distance 8.

Lemma 6.6.1. *(Conway) With respect to a fixed basis, coordinates supporting the codewords of weight* 8 *form the* 759 *octads of a Steiner system* $S(5, 8, 24)$.

We recall that C_{12} is a linear code over $GF(3)$ which is dimension 6 and minimum distance 6.

Lemma 6.6.2. *(Conway) With respect to a fixed basis, coordinates supporting the codewords of weight* 6 *form the* 132 *hexads of a Steiner system*

$S(5,8,24)$. *Conversely, for each hexad in this Steiner system, there are exactly two codewords in* C_{12} *supported on that hexad.*

Thus, the two codes are somehow analogous. Based on this, it is natural to ask if there is a decoding algorithm analogous to [BG] for C_{12}. The object of this paper is to answer this question in the affirmative.

6.6.2 Background

First, let us recall some constructions of C_{12} (both due to Conway).

Tetracode construction

Definition 6.6.3. *The* **tetracode** *is the* $GF(3)$ *code* T *with elements*

$$(0,0,0,0),\ (1,0,1,2),\ (1,2,0,1),\ (1,1,2,0),$$

$$(0,1,1,1),\ (2,0,2,1),\ (2,1,0,2),\ (2,2,1,0),\ (0,2,2,2).$$

It is a self-dual $(4,2,3)$ *code.*

Here is Conway's "tetracode" construction of the C_{12}. Represent each 12-tuple $c = (c_1, ..., c_{12}) \in C_{12}$ as a 3×4 array

$$c = \begin{pmatrix} c_1 & c_2 & c_3 & c_4 \\ c_5 & c_6 & c_7 & c_8 \\ c_9 & c_{10} & c_{11} & c_{12} \end{pmatrix}.$$

The **projection** of c is

$$pr(c) = (c_5 - c_9, c_6 - c_{10}, c_7 - c_{11}, c_8 - c_{12}).$$

The **row score** is the sum of the elements in that row. The **column score** is the sum of the elements in that column. Of course, all computations are in $GF(3)$.

Lemma 6.6.4. *(Conway) An array* c *is in* C_{12} *if and only if*

- *the (common) score of all four columns equals the negative of the score of the top row.*
- $pr(c) \in T$.

There are several facts one can derive from this construction.

There are 264 codewords of weight 6, 440 codewords of weight 9, 24 codewords of weight 12, and codewords 729 total.

Pick an arbitrary subset of 9 elements taken from $\{1, 2, 3, ..., 12\}$. It is the support of exactly two codewords of weight 9 in C_{12}. Pick a random subset S of 6 elements taken from $\{1, 2, 3, ..., 12\}$. The probablity that S is the support of some codeword of weight 6 in C_{12} is 1/7.

Lemma 6.6.5. *For each weight* 6 *codeword* c *in* C_{12}*, there is a weight* 12 *codeword* c' *such that* $c + c'$ *has weight* 6.

If we call $c + c'$ *a "complement" of* c *then "the complement" is unique up to sign.*

proof: The support of the codewords of weight 6 form a $S(5, 6, 12)$ Steiner system. Therefore, to any weight 6 codeword c there is a codeword c'' whose support is in the complement of that of c. let $c' = c'' - c$. □

Remark 6.6.6. *Although we shall not need it, it appears that for each weight* 9 *codeword* c *in* C_{12}*, there is a weight* 12 *codeword* c' *such that* $c+c'$ *has weight* 6.

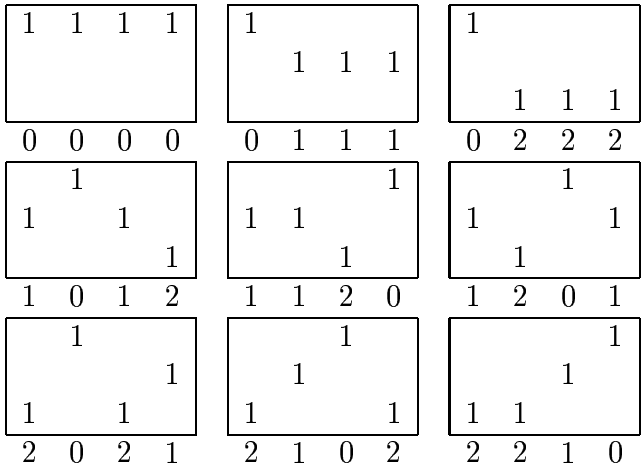
The "col/tet" construction

In fact, this only constructs the codewords of weight 6, but since they generate the code, we can use them to compute a generating matrix for C_{12}.

A *col* (or column) is a placement of three 1's in a column of the array (a blank space represents a 0):

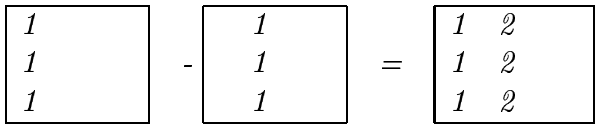
1	1	1	1
1	1	1	1
1	1	1	1

A *tet* (or tetrad) is a placement of 4 1's having entries corresponding (as explained below) to a tetracode.



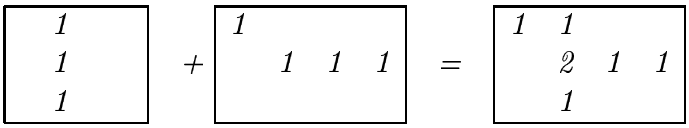
Each line in $\mathbb{F}_3^2$ with finite slope occurs once in the 3×3 part of some tet. Define the **label** of the first (top) row to be 0, of the second row to -1, and of the bottom row to 1. The **odd man out** for a column is the label of the row corresponding to the non-zero digit in that column; if the column has no non-zero digit then the odd man out is a "?". Thus the tetracode words associated in this way to these patterns are the odd men out for the tets.

Example 6.6.7. *Associated to the col-col pattern*



is the tetracode $(0, 0, ?, ?)$.

Associated to the col+tet pattern



is the tetracode $(0, 1, 1, 1)$.

The MINIMOG and $S(5, 6, 12)$

The *MINIMOG in the shuffle numbering* is the 3×4 array

6	3	0	9
5	2	7	10
4	1	8	11

The *signed hexads* are the combinations 6-sets obtained from the MINIMOG from patterns of the form

col-col, col+tet, tet-tet, col+col-tet.

The next result explains Lemma 6.6.2 above in a little more detail.

Lemma 6.6.8. *(Conway, [CS], chapter 11, page 321) If we ignore signs, then from these signed hexads we get the 132 hexads of a Steiner system* $S(5,6,12)$.

6.6.3 $S(5,6,12)$

An *m-(sub)set* is a (sub)set with m elements. For integers $k < m < n$, a *Steiner system* $S(k,m,n)$ is an n-set X and a set S of m-subsets having the property that any k-subset of X is contained in exactly one m-set in S. For example, if $X = \{1,2,...,12\}$, a Steiner system $S(5,6,12)$ is a set of 6-sets, called *hexads*, with the property that any set of 5 elements of X is contained in ("can be completed to") exactly one hexad.

This section focuses on $S(5,6,12)$. If S is a Steiner system of type $(5,6,12)$ in a 12-set X then the symmetric group S_X of X sends S to another Steiner system $\sigma(S)$ of X. It is known that if S and S' are any two Steiner systems of type $(5,6,12)$ in X then there is a $\sigma \in S_X$ such that $S' = \sigma(S)$. In other words, a Steiner system of this type is unique up to relabelings. (This also implies that if one defines M_{12} to be the stabilizer of a fixed Steiner system of type $(5,6,12)$ in $X = \{1,2,...,12\}$ then any two such groups, for different Steiner systems in X, must be conjugate in S_X. In particular, such a definition is well-defined up to isomorphism.)

Curtis' kitten - modulo 11 labeling

J. Conway and R. Curtis [Cu] found a relatively simple and elegant way to construct hexads in a particular Steiner system $S(5,6,12)$ using the arithmetical geometry of the projective line over the finite field with 11 elements. This subsection describes this. First, we shall describe the Steiner system using the "modula 11 labeling" - one which we shall not use but which is perhaps more geometrically motivated. Afterwards, we shall translate it into the "shuffle" labeling we will use.

Let

$$\mathbf{P}^1(\mathbf{F}_{11}) = \{\infty, 0, 1, 2, ..., 9, 10\}$$

denote the projective line over the finite field $\mathbf{F}_{11}$ with 11 elements. Let

$$Q = \{0, 1, 3, 4, 5, 9\}$$

denote the quadratic residues and 0 and let

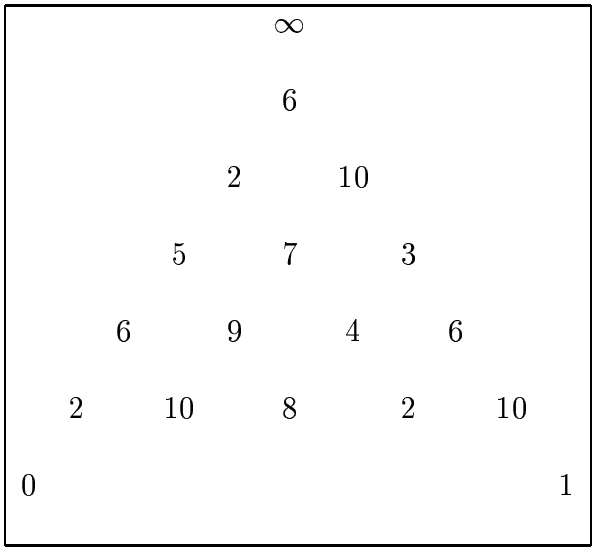
$$L =< \alpha, \beta > \cong PSL(2, \mathbf{F}_{11}),$$

where $\alpha(y) = y + 1$ and $\beta(y) = -1/y$. Let

$$S = \{\lambda(Q) \mid \lambda \in L\}.$$

Lemma 6.6.9. *S is a Steiner system of type $(5, 6, 12)$.*

The elements of S are known as *hexads* (in the "modulo 11 labeling").



Curtis' Kitten.

The "views" from each of the three "points at infinity" is given in the following tables.

6	10	3
2	7	4
5	9	8

picture at ∞

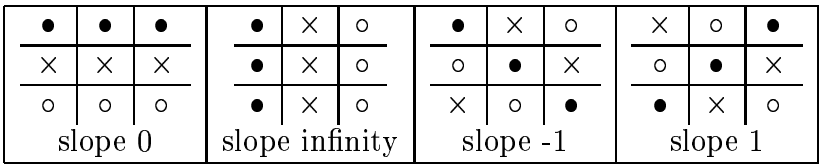
5	7	3
6	9	4
2	10	8

picture at 0

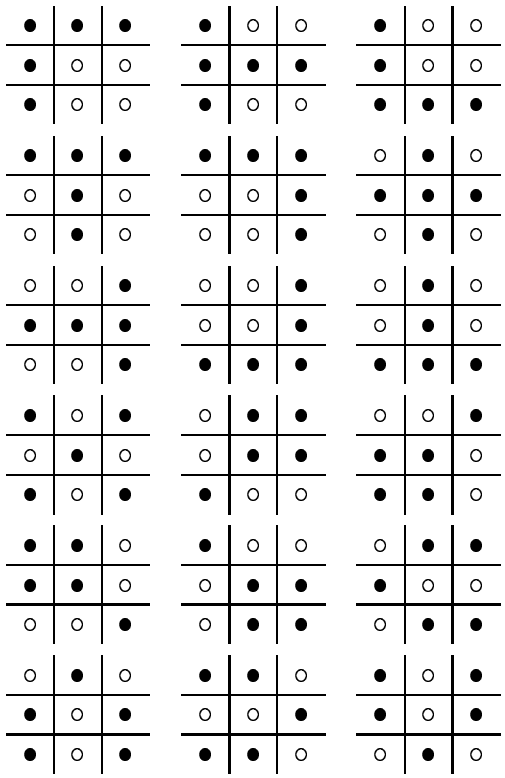
5	7	3
9	4	6
8	2	10

picture at 1

Each of these 3×3 arrays may be regarded as the plane $\mathbf{F}_3^2$. The *lines* of this plane are described by one of the following patterns.



The union of any two perpendicular lines is called a *cross*. There are 18 crosses. The *crosses* of this plane are described by one of the following patterns of filled circles.



The complement of a cross in $\mathbf{F}_3^2$ is called a *square*. Of course there are also 18 squares. The *crosses* of this plane are described by one of the above patterns of hollow circles.

The hexads are

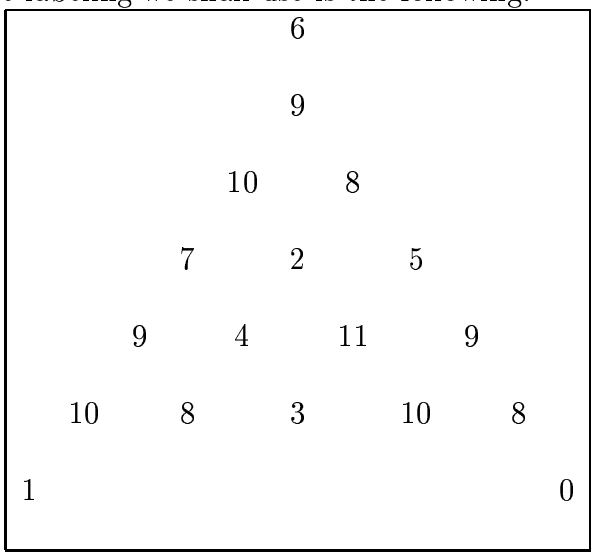
1. $\{0, 1, \infty\} \cup \{\text{any line}\}$,

2. the union of any two (distinct) parallel lines in the same picture,
3. one "point at infinity" union a cross in the corresponding picture,
4. two "points at infinity" union a square in the picture corresponding to the omitted point at infinity.

Lemma 6.6.10. *(Curtis [Cu]) There are 132 such hexads (12 of type 1, 12 of type 2, 54 of type 3, and 54 of type 4). They form a Steiner system of type* $(5, 6, 12)$.

Curtis' kitten - shuffle labeling

The kitten in the labeling we shall use is the following.



The Shuffle Kitten.

The "views" from each of the three "points at infinity" is given in the following tables.

5	11	3	5	11	3	8	10	3
8	2	4	2	4	8	9	11	4
9	10	7	7	9	10	5	2	7
picture at 6			picture at 1			picture at 0		

Example 6.6.11.

- *0,2,4,5,6,11 is a square in the picture at 1.*
- *0,2,3,4,5,7 is a cross in the picture at 0.*

6.6.4 The algorithm

We shall assume in the algorithm below that a look-up table has been constructed which allows one to find the two codewords in C_{12} supported on a given hexad.

Let $r \in \mathbb{F}_3^{12}$ be the received word. We assume that no more than 2 errors have occurred. We shall try to determine the codeword $c \in C_{12}$ which was sent.

Step 1: If the weight of r is less than or equal to 2 then we are done: $c = 0$.

Step 2: If $wt(r) > 6$ then let $c' \in C_{12}$ have the property that the weight of $r + c'$ is as close to 6 as possible and $wt(r + c') \leq 6$. Otherwise, let $c' = 0$. Let $r' = r + c'$. Then $wt(r') \leq 6$.

Step 3: If the weight of r' is less than or equal to 2 then we are done: $c = -c'$.

Step 4: We have $4 \leq wt(r') \leq 8$. Let $S = support(r')$. Find the hexads H in $S(5, 6, 12)$ with "large" intersection with S. Use the lookup table to find the codewords $\pm w_h$, with $h \in H$, having this hexad as support. Let $r'' = r' + c''$ where $c'' \in \{\pm w_h \mid h \in H\}$ is choosen to make the weight of r'' as small as possible.

Step 5: We have $wt(r'') \leq 2$. Let $c = -c' - c''$.

Example 6.6.12. *Let*

$$r = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 2 \end{pmatrix}.$$

We pick $c' = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \end{pmatrix}$, *so* $r' = \begin{pmatrix} 2 & 2 & 1 & 2 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 2 \end{pmatrix}$. *There are two words supported on the same hexad as* r' *is:*

$$\begin{pmatrix} 2 & 2 & 2 & 1 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 2 \end{pmatrix}, \quad \begin{pmatrix} 1 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Let c'' *be the second one and let* $r'' = r' + c'' = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$. *This is*

weight 2, *so* $c = -c' - c'' = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 2 \end{pmatrix}$.

Example 6.6.13. *Let*

$$r = \begin{pmatrix} 2 & 2 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \end{pmatrix}.$$

There is no weight 12 c' *for which* $wt(r + c') < wt(r)$, *so we take* $c' = 0$. *Let* $r' = r$. *There are two words supported on the same hexad as* r' *is:*

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \end{pmatrix}, \quad \begin{pmatrix} 2 & 2 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

Let c'' *be the second one and let* $r'' = r' + c'' = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$. *This is weight* 2, *so* $c = -c' - c'' = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \end{pmatrix}$.

This has been programmed into GAP, see [J2].

Chapter 7

Appendices

7.1 Basics of GAP

We show how to use some simple commands in GAP [Gap]. This note is written for the first-time user of GAP. (Many thanks to Alexander Hulpke for suggestions and some material for this chapter.)

7.1.1 Introduction

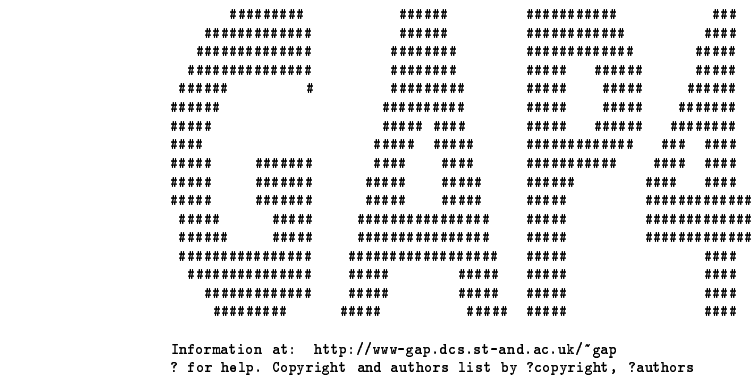
GAP [Gap] is a free software package designed originally for groups but which now does much more. The GAP project was originally headed by Joachim Neubueser at Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany. His students worked on it for many years until his retirement; Martin Schönert especially deserves a lot of credit (this opinion coming from Neubueser himself). After that, GAP moved from Germany to Scotland. Its home is now: The GAP Group, Mathematical Institute, St. Andrews, Scotland (`gap@dcs.st-and.ac.uk`, `http://www.gap-system.org/`), where it is managed by (among others) Steve Linton. Other workers (there are too many to list here) are listed at the URL:

`http://www.gap-system.org/Info/authors.html`

There is also a manual (both in latex and in html), an email list for bugs, `gap-trouble@dcs.st-and.ac.uk` , and one for GAP questions, `gap-forum@dcs.st-and.ac.uk` , (first, go to the URL `http://www.gap-system.org/Info/forum.html`).

It will be assumed that you have GAP up and running. Sometimes a GAP command will be given but the GAP output will not be given (though quite often we give both), since it will be assumed you can see it yourself. Many commands assume that the commands previously discussed have already been entered.

Once you start GAP, you will see a banner like this:



```
   Information at:  http://www-gap.dcs.st-and.ac.uk/~gap
   ? for help. Copyright and authors list by ?copyright, ?authors

  Loading the library. Please be patient, this may take a while.
Gap4, Version: 4.1 fix 7 of 11-Nov-1999, i686-pc-linux-gnu-gcc
Components:  small, small2, small3, id2, id3, trans, prim, tbl,
             tom  installed.
```

From this you know linux is running on a pentium PC and gap, version 4.1, has started.

We recommend you log every session. To do this, type

```
gap> LogTo("/home/wdj/gapfiles/basic1.log");
```

for example. On a windows PC, the command looks instead like

```
gap> LogTo("c:/gap4r1/gapfiles/basic1.log");
```

You don't type " `gap>`", by the way, as that is simply the GAP cursor beginning each line.

7.1.2 Input-output

GAP is entirely a command-line driven interface. There are no mouse buttons or pop-down menus or graphics in GAP.

You will notice that most GAP commands have a lot of capital letters (like MAGMA and MATHEMATICA but unlike MAPLE). If you, like me, tend to forget to capitalize a letter occuring in the middle of a command GAP will be completely unforgiving.

The syntax of GAP is not as "loose" as MAPLE's, in some sense. However, there are some similarities. For example, each command must end in a ; (or ;; if you want to supress the output), as in MAPLE.

Syntax tips:

- GAP is picky about upper case/lower case. `LogTo` is not the same as `logto`.
- All commands end in a semicolon!
- Type `?` followed by a subject name (and no semicolon...) to get the online help.
- If you get syntax errors, use `Ctl p` to get the line back, and correct the errors.
- Incorrect input, interruption of a calculation (by pressing `Ctl c` – this can be useful if you tried out a command that takes too long), or bugs can cause GAP to enter a so-called *break-loop*, indicated by the prompt `brk>`. You can leave this break loop with `quit;` or `Ctl d`.
- If you create larger input, it can be helpful to put it in a text file and to read it from GAP. This can be done with the command `Read("filename");`.
- By terminating a command with a double semicolon `;;` you can avoid GAP displaying the result. (Obviously, this is only useful if assigning it to a variable.)
- everything after a hash mark (`#`) is a comment.

Usually, if you enter a command with poor syntax, GAP will simple issue an error message and then wait for your next command. Sometimes, if you forget to enter a ;, GAP will simply not respond or it will give you an error message after the *next* command. Othertimes, GAP will "break" and give a new type of command line `brk>`. You must type `quit;` to return to the normal mode `gap>`.

command-line history Here's a very handy feature, expecially when correcting typos or reentering commands. GAP, like MAGMA, not only has a command line history but it is has can look up any command you typed by its first few letters. For example, suppose you type

```
gap>a:=2;
gap>b:=3;
gap>c:=4;
gap>d:=5;
```

Now hit the up arrow key at the next `>` prompt. GAP will automatically enter the previous command `>d:=5;`. If you hit the up arrow key twice, MAGMA will automatically enter the command `>c:=4;`. If you type `>a` (do not hit `return`) and then hit the up arrow key, GAP will automatically enter the previous command which started with an `a`, namely `>a:=2;`.

Comments in GAP are, like MAPLE, single line and follow `#`. For example, in

```
>a:=2+3; #this defines a comment
>b:=3*a; # c:=a+b;
>c;
```

GAP will ignore the definition of c, hence the error you will see on your screen, but computed a, b.

To stop a GAP command from executing after it is entered (say it is taking more time that you want to wait for it to finish), type CTL-c.

To quit GAP, type `quit;`.

7.1.3 Polynomials and other functions

Another (major) difference with MAPLE is that GAP does not like undefined variables. For example, in MAPLE, you can type (without declaring x) `>f:=x^2;` then `>diff(f,x);` (resp, `>int(f,x);`) and you will get the derivative (resp., anti-derivavtive) of x^2. However, GAP will complain:

```
gap> f:=x^2;
Variable: 'x' must have a value
```

Instead, you must declare x first, by typing something like

```
gap> R:=PolynomialRing(GF(7),["x"]);
<algebra-with-one over GF(7), with 1 generators>
gap> p:=UnivariatePolynomial(GF(7),[1,2,3,4],1);
1+2*x+3*x^2+4*x^3
```

```
gap> Derivative(p);
0*x^-1+2+6*x+12*x^2
gap> Value(p,1);
10
gap> Value(p,Z(7));
Z(7)^2
gap> q:=UnivariatePolynomial(GF(7),[Z(7),2*Z(7),3*Z(7),4*Z(7)],1);
Z(7)-x+Z(7)^2*x^2+Z(7)^5*x^3
gap> q in R;
true
gap> Derivative(q);
-Z(7)^0+Z(7)^4*x+x^2
```

This shows how to enter a polynomial, differentiate it, and evaluate it at a number. It's odd that GAP (unlike MAGMA) doesn't automatically reduce the coefficients of p mod 7, despite the fact that it was told that its coefficients belong to $GF(7) = \mathbb{Z}/7\mathbb{Z}$.

Exercise 7.1.1. *Using the* `Value` *and* `UnivariatePolynomial` *commands, compute* $x^100 + x + 1$ *mod 31, where* $x = 17$.

7.1.4 Lists

A list of elements of the form $f(i)$, where i ranges over all integers between a and b can be created using the command `L:=[[a..b],i->f(i)];` . Here f is any function whihc can be entered in GAP. For example, f itself can be a list:

```
gap> L:=List( [1..10], i -> [i,i^2+1] );
[ [ 1, 2 ], [ 2, 5 ], [ 3, 10 ], [ 4, 17 ], [ 5, 26 ], [ 6, 37 ], [ 7, 50 ],
  [ 8, 65 ], [ 9, 82 ], [ 10, 101 ] ]
```

If f is Boolean-valued then there is another option for creating a list. The `Filtered` command only lists those elements for which $f(i)$ is true. Here's an example:

```
gap> L:=List( [1..10], i -> IsOddInt(i) );
[ true, false, true, false, true, false, true, false, true, false ]
gap> L:=Filtered( [1..10], i -> IsOddInt(i) );
[ 1, 3, 5, 7, 9 ]
```

Exercise 7.1.2. *Find the sequence of primes from* 2 *to* 100. *(Hint: Use the* `IsPrime` *command.)*

7.1.5 Vectors and matrices

A vector in GAP is simply a list of numbers. One can access elements of a list using square brackets as well. A list of (row) vectors is a matrix. GAP knows matrix arithmetic.

```
gap> vec:=[1,2,3,4];
[ 1, 2, 3, 4 ]
gap> vec[3]+2;
5
gap> mat:=[[1,2,3,4],[5,6,7,8],[9,10,11,12]];
[ [ 1, 2, 3, 4 ], [ 5, 6, 7, 8 ], [ 9, 10, 11, 12 ] ]
gap> mat*vec;
[ 30, 70, 110 ]
gap> mat:=[[1,2,3],[5,6,7],[9,10,12]];
[ [ 1, 2, 3 ], [ 5, 6, 7 ], [ 9, 10, 12 ] ]
gap> mat^5;
[ [ 289876, 342744, 416603 ], [ 766848, 906704, 1102091 ],
  [ 1309817, 1548698, 1882429 ] ]
gap> RankMat(mat);
3
gap> DeterminantMat(mat);
-4
```

The command `Display` can be used to get a nicer output:

```
gap> 3*mat^2-mat;
[ [ 113, 130, 156 ], [ 289, 342, 416 ], [ 492, 584, 711 ] ]
gap> Display(last);
[ [  113,  130,  156 ],
  [  289,  342,  416 ],
  [  492,  584,  711 ] ]
```

We can also calculate in matrices modulo a number:

```
gap> mat:=[[1,2,3],[5,6,7],[9,10,12]]*One(im);
[ [ Z(7)^0, Z(7)^2, Z(7) ], [ Z(7)^5, Z(7)^3, 0*Z(7) ],
```

```
  [ Z(7)^2, Z(7), Z(7)^5 ] ]
gap> Display(mat);
 1 2 3
 5 6 .
 2 3 5
gap> mat^-1+mat;
[ [ Z(7)^4, Z(7)^4, Z(7)^4 ], [ Z(7)^3, Z(7)^0, Z(7)^5 ],
  [ Z(7), Z(7)^0, Z(7)^3 ] ]
```

7.1.6 Using for loops

The list of all squares between 1 and 100 can be created using a for loop. The syntax is similar, though not exactly the same, as MAPLE of MAGMA.

```
gap> K:=[];
[  ]
gap> for i in [1..10] do
> K[i]:=i^2;
> od;
gap> K;
[ 1, 4, 9, 16, 25, 36, 49, 64, 81, 100 ]
```

The combination of a for loop and and if-then statement can produce some odd lists:

```
gap> K:=[];
[  ]
gap> for i in [1..10] do
> if IsOddInt(i) then
> K[i]:=i^2;
> fi;
> od;
gap> K;
[ 1,, 9,, 25,, 49,, 81 ]
gap> K[2];
List Element: <list>[2] must have an assigned value
not in any function
Entering break read-eval-print loop, you can 'quit;' to quit to outer loop,
or you can return after assigning a value to continue
```

```
brk> quit;
gap> K[3];
9
```

This says that the 3-rd element of the list K does not exist and calling it will kick GAP out of its normal mode.

Another command which can be used to construct lists are the `Add` or `Append` command:

```
gap> L:=[1,2,3];Append(L,[11]);L;
[ 1, 2, 3 ]
[ 1, 2, 3, 11 ]
gap> L:=[1,2,3];
[ 1, 2, 3 ]
gap> Add(L,4);
gap> L;
[ 1, 2, 3, 4 ]
```

The `Append` command modifies the list it is applied to. The command `Append(L,[11]);` is similar to the MAGMA command `Append(L ,[11]);`

The inverse of `Append` is `RemoveSet`:

```
gap> L:=[1,2,3];RemoveSet(L,2);L;
[ 1, 2, 3 ]
[ 1, 3 ]
```

The `RemoveSet` command modifies the list it is applied to.

Exercise 7.1.3. *(a) Find the sequence L of integers from -20 to 20.*
(b) Using the `RemoveSet` *command, remove from L the element 0.*
(c) Using the `Append` *command, remove from L the element 100.*

7.1.7 Sets

The `AsSet` command converts a list to a set and back to a list. In effect, it removes all extraneous elements.

```
gap> L:=[3,1];
[ 3, 1 ]
```

```
gap> AsSet(L);
[ 1, 3 ]
gap> L;
[ 3, 1 ]
gap> Append(L,[1]);
gap> L;
[ 3, 1, 1 ]
gap> AsSet(L);
[ 1, 3 ]
```

7.1.8 Adding numbers in a list

For adding (resp., multiplying) all the elements of a list, use the `Sum` (resp., `Product`) command. To add (multiply) all the integers from 1 to 100 is easy:

```
> L:=[1..100];
> Sum(L);
5050
gap> Product(L);
933262154439441526816992388562667004907159682643816214685929638952175999932299\
156089414639761565182862536979208272237582511852109168640000000000000000000000\
00
```

The product is so large that GAP wraps the number to the next line using a \ symbol.

7.1.9 GAP functions and procedures

A function and a procedure are the same thing in GAP. Like MAPLE, you do declare local or global variables.

Here's an example:

```
gap> fibonacci := function(n)
>     if n <= 2 then
>       return 1;
>     else
>       return fibonacci(n-1) + fibonacci(n-2);
>     fi;
>    end;
> fibonacci(10)+fibonacci(12);
199
```

Exercise 7.1.4. *Write a function which computes s_n, where $s_1 = 1, s_2 = 0$, and $s_n = s_{n-1} + s_{n-2}$, for $n > 1$.*

7.1.10 Number theoretic functions in GAP

The GAP command `LogMod( n, r, m )` computes the discrete r-logarithm of the integer n modulo the integer m. It returns a number x such that $r^x \equiv n \pmod{m}$, if such a number exists. Otherwise fail is returned. For example, `LogMod(9,2,11);` returns `6`.

Exercise 7.1.5. *Use GAP to solve or answer the following questions.*

(a) Is 2 *a primitive root mod* 541*? Find the discrete log, if it exists, of* 34 *to the base* 2*, mod* 541*. (Ans: yes,* 100*)*

(b) Find the discrete log of 11 *to base* 5 *mod* 97*, if it exists.*

(c) Is 197 *a prime? Is* 2 *a primitive root mod* 197*? Find the discrete log of* 91 *to base* 2 *mod* 197*, if it exists (ans:* 44*),*

(d) Find how many elements in $\mathbb{Z}/n\mathbb{Z}$ *are invertible for* $n = 4, 8, 16, 128, 7, 35, 49$*. (Hint: Euler's totient function in GAP is* `Phi`*.)*

Use GAP to solve the following Diffie-Hellman problem.

Exercise 7.1.6. *Let* $p = 541$*,* $a = 2$*. Amelia chooses* $x = 17$ *and sends* $a^x = 2^{17} \pmod{541}$ *to Ben. Ben picks* $y = 71$ *and sends* $a^y = 2^{71} \pmod{541}$ *to Amelia. Compute the secret shared key. (Hint: Use the* `PowerMod` *function.)*

Use GAP to solve the following RSA problem.

Exercise 7.1.7. *Let* $p = 3$*,* $q = 7$*, so* $\phi(n) = \phi(21) = 12$*. Let* $e = 11$ *(the public exponent), find* d *(the private exponent). Decrypt the cipher-text* $c = 20$*.*

7.2 Simple exercises in MAGMA

7.2.1 Introduction

MAGMA [MAGMA] is a non-free[1] software package "designed to solve computationally hard problems in algebra, number theory, geometry and combinatorics". The name "magma" is explained by another quote from the MAGMA documentation: "The primary concept in the design of the Magma system is that of a 'magma'. Following Bourbaki, a magma can be defined as

[1] Not free but not for profit either. The cost of the license is put into further development of the package.

a set with a law of composition." Never fear, you don't need to understand all of Bourbaki to be able to use some of the simpler commands in MAGMA!

The MAGMA project has been headed by John Cannon in the Department of Mathematics at the University of Sydney in Australia since the 1970's. It now has many contributors from all over the world (see the section "Acknowledgements" in the documentation). There is also a manual (both in latex and in html), the printed version of which is about 2500 pages. There are also two books provided free with MAGMA [Magma1] (186 pages), [Magma2] (854 pages).

Help is available "on line", like MAPLE, or better (in my opinion) in the form of indexed html pages. In fact, I usually have MAGMA and netscape with the MAGMA help pages running simultaneously. Many people find it useful to get in the habit of also opening separately a file, called `mycommands.mag` let's say, in an editor and either copy-and-paste commands into MAGMA or else entering them directly in the form `magma mycommands.mag`.

It will be assumed that you have MAGMA up and running. It is also a good idea to have your browser (netscape, for example) pointed to MAGMA's documentation in html (which comes with MAGMA but is available online at `http://www.maths.usyd.edu.au:8000/u/magma/htmlhelp/MAGMA.html`).

Once you start MAGMA, you will see a banner like this:

```
Magma V2.7-2      Sun Sep 10 2000 09:16:39 on joynerwd    [Seed = 2097073323]
Type ? for help.  Type <Ctrl>-D to quit.
```

From this you already know the date, that I am on a machine named `joynerwd`, and that I have started MAGMA, version 2.7.2.

I recommend you log every session. To do this, type

```
> SetLogFile("/home/wdj/magma27/basics1.log");
```

for example (we will be doing an example based on basic commands in MAGMA, so the file name will be called `basics1.log`). You don't type "`>`", by the way, as that is simply the MAGMA cursor beginning each line. Also, be sure you save the log to a directory you have read-write access to.

7.2.2 Input-output

MAGMA is entirely a command-line driven interface. There are no mouse buttons or pop-down menus or graphics in MAGMA.

You will notice that most MAGMA commands have a lot of capital letters (like GAP and MATHEMATICA but unlike MAPLE). If you, like me, tend to forget to capitalize a letter occuring in the middle of a command MAGMA will be completely unforgiving. I may have sympathy for you but MAGMA will not!

The syntax of MAGMA is not as "loose" as MAPLE's, in some sense. However, there are some similarities. For example, each command must end in a `;`, as in MAPLE (though MAPLE allows `:` as well).

One difference is that MAGMA will not print an output if you enter a command which simply defines a variable. For example, `>a:=2+3;` will not have any output displayed, unlike MAPLE (which will print out *a:=5*). As you might expect, if you next simply type `>a;`, MAGMA (and MAPLE) will output `5`. Similar to MAPLE, MAGMA has a command called `SetVerbose` which prints out more detailed outputs, but we shall not consider that here.

Usually, if you enter a command with poor syntax, MAGMA will simple issue an error message and then wait for your next command. Sometimes, if you forget to enter a `;`, MAGMA will simply not respond or it will give you an error message after the *next* command. In this respect, it might be a bit nicer than GAP (if you've ever dealt with it) but similar to MAPLE.

command-line history Here's a very handy feature, expecially when correcting typos or re-entering commands. MAGMA not only has a command line history but you can look up any command you typed by its first few letters. For example, suppose you type

```
>a:=2;
>b:=3;
>c:=4;
>d:=5;
```

Now hit the up arrow key at the next `>` prompt. MAGMA will automatically enter the previous command `>d:=5;`. If you hit the up arrow key twice, MAGMA will automatically enter the command `>c:=4;`. If you type `>a` (do not hit `return`) and then hit the up arrow key, MAGMA will automatically enter the previous command which started with an `a`, namely `>a:=2;`.

Comments in MAGMA are either multi-line and have `/*` to begin and `*/` to end, or are single line and follow `//`. For example, in

```
>a:=2+3; //this defines a
>b:=3*a; /* this
>c:=b^2; defines b*/
> c;

>> c;
   ^
User error: Identifier 'c' has not been declared or assigned
> b;
15
```

MAGMA will ignore the definition of c, hence the error, but computed a, b.

To stop a MAGMA command from executing after it is entered (say it is taking more time that you want to wait for it to finish), type CTL-c. This seems to work better than MAPLE's *stop* (mouse button) command. To quit MAGMA, type CTL-d or `quit;`.

7.2.3 Polynomials and other functions

Another (major) difference with MAPLE is that MAGMA does not like undefined variables. It is fair to say that MAGMA can't stand the sight of an undefined variable. For example, in MAPLE, you can type (without declaring x) `>f:=x^2;` then `>diff(f,x);` (resp, `>int(f,x);`) and you will get the derivative (resp., anti-derivavtive) of x^2. However, MAGMA will complain:

```
> f:=x^2;

>> f:=x^2;
      ^
User error: Identifier 'x' has not been declared or assigned
```

Instead, you must declare x first, by typing something like

```
> P<x>:=PolynomialRing(Integers());
> f:=x^2;
```

```
> Derivative(f);
2*x
> Integral(f) ;

>> Integral(f) ;
           ^
Runtime error in 'Integral': Coefficient ring of argument 1 is not a field
```

This says, MAGMA will differentiate f but it will not integrate it since the result, $x^3/3$, will not belong to the ring $P = \mathbb{Z}[x]$ you declared. To integrate, type:

```
> P<x>:=PolynomialRing(Rationals());
>  f:=x^2;
> Evaluate(f,3/2);
9/4
> Integral(f) ;
1/3*x^3
```

You can also do numerical integration of an improper integral:

```
> R := RealField();
> f := map< R -> R | x :-> x^2 >;
> Integral(f, 0, 1);
0.33333333333333333333333333331
> h:= map< R -> R | x :-> Sin(x)/x >;
> h(1/2);
0.95885107720840600054657587 04
> h(0);

???(
    x: 0
)
>> h:= map< R -> R | x :-> Sin(x)/x >;
                                  ^
Runtime error in '/': Division by zero
> Integral(h, 0, 1: Al := "Open");
0.94608307036718301494135330 89
```

Here ``Open'' chooses a certain subroutine to compute $\int_0^1 \frac{\sin(x)}{x} dx$ since the integrand is undefined at $x = 0$.

There are also routines for infinite sums but I have not gotten some of them to work:

```
> g := map< Integers() -> R | x :-> 1/x^2 >;
> InfiniteSum(g,1);

[Interrupted]
> PositiveSum(g,1);
1.64493406684822643647241516
> g2 := map< Integers() -> R | m :-> (-1)^m/(m^2+1) >;
> AlternatingSum(g2,1);
-0.36398547250893341852488170 81
> g3 := map< Integers() -> R | m :-> (-1)^m/Exp(m) >;
> InfiniteSum(g3,1);
-0.26894142136999512074884075817909516105374834825 3
```

The interuption was after about 1 minute (on a 350Mhz pentium). This is probably because the series is slowly converging and MAGMA will compute the result to the default precision. However, MAGMA returned the `PositiveSum(g,1);` and `AlternatingSum(g2,1);` commands instantly.

Exercise 7.2.1. *Differentiate and integrate $g(t) = t^2 + t/2 + a$, where a is an arbitrary but fixed constant* [2].

Exercise 7.2.2. *Compute $\int_0^1 \cos(t)\, dt$ to at least 5 decimal places.*

Exercise 7.2.3. *Compute $\sum_{n=1}^{\infty} 2^{-n}$ to at least 5 decimal places.*

7.2.4 Common data structures

There are a *lot* of data structures in MAGMA! We will only look at the most commonly occurring ones.

[2] Declare both a, T as variables.

7.2.5 Lists, sequences, via for loops

Let's start with lists, sequences, ... in MAGMA. These structures seem to pop up a lot in using programs like MAPLE or GAP, so it may help to see what the syntax is for them first.

The syntax for MAGMA is more complicated than in MAPLE. In MAPLE of GAP, a list is of the form `L:=[1,2,3,4];`. Lists are entered differently in MAGMA.

According to the MAGMA documentation, "A list in Magma is an ordered finite collection of objects. Unlike sequences, lists are not required to consist of objects that have some common parent. " On the other hand, "A sequence in Magma is a linearly ordered collection of objects belonging to some common structure (called the universe of the sequence). " So, lists and sequences are different animals to MAGMA. (Lists in MAGMA are of the form `<...>` and sequences of the form `[...]`. In MAPLE, a sequence is a list `[...]` but without the brackets.) However, as this is supposed to be an elementary and basic introduction, we shall not distinguish them at this point. As long as your elements have the same universe, it doesn't matter which one you use.

A sequence (or list) of consecutive integers is easy. For example, the integers from -3 to 101 is constructed by typing

```
> [-3..101];
```

Here are some other basic lists (more precisely, in MAGMA, these are sequences):

```
> List1:=[2^i:i in [1..3]];
> List1;
[ 2, 4, 8 ]
> List2:=[NextPrime(i):i in [1..10]];
> List2;
[ 2, 3, 5, 5, 7, 7, 11, 11, 11, 11 ]
```

Pretty simple, right?

Using for loops

Another way to define a list is to use a `for` loop:

```
> L:=[];
> for i in [1..10] do
for> L[i]:=i^2;
for> end for;
> L;
[ 1, 4, 9, 16, 25, 36, 49, 64, 81, 100 ]
```

The first line `L:=[];` is to initialize L. If omitted, MAGMA will give an error after you enter the line `for> L[i]:=i^2;`. The append command also works:

```
> L:=[];
> for i in [1..10] do
for> L:=Append(L,i^2);
for> end for;
```

Using the "tilde" notation, a similar command produces the same result:

```
> L:=[];
> for i in [1..10] do
for> Append(~L,i^2);
for> end for;
```

Exercise 7.2.4. *Create the list of all integers of the form* n^2+1, $1 \leq n \leq 20$.

Using if-then statements

What if your list is not created by a formula, like $i \longmapsto i^2$, but instead by a condition, like "all primes p for which 4 divides $p-3$ and $1 \leq p \leq 100$"? One way is to add a conditional statement to the for-loop, for example:

```
> L:=[];
> for i in [1..100] do
for> if IsPrime(i) and i mod 4 eq 3 then
for|if> L:=Append(L,i);
for|if> end if;
for> end for;
> L;
[ 3, 7, 11, 19, 23, 31, 43, 47, 59, 67, 71, 79, 83 ]
```

Exercise 7.2.5. *(a) Find the number of primes which are* < 1000.

To find a subsequence of a sequence S satisfying a property P, use the $[x : x \text{ in } S \mid P(x)]$. For example, to find all the elements in List1 which are less than or equal to 4, type

```
> List1:=[2^i:i in [1..3]];
> List1_2:=[j : j in List1 | j le 4];
> List1_2;
[ 2, 4 ]
```

Exercise 7.2.6. *Create the sequence of all integers from* -20 *to* 20. *Next create the subset those elements which are even.*

7.2.6 Removing elements from a list

First, something funny (funny/strange, not funny/haha):

```
> L:=[];
> L[3]:=10;
> Remove(~L,1);
> L[3];

>> L[3];
    ^
Runtime error in '[]': Sequence element 3 not defined
> L[2];
10
> L[1];

>> L[1];
    ^
Runtime error in '[]': Sequence element 1 not defined
```

MAGMA has deleted `L[1];` (which wasn't even defined), and changed `L[2]` to `L[1]` (which wasn't even defined), changed `L[3]` to `L[2]`!

Exercise 7.2.7. *Using the* `Reverse` *command, find the sequence of integers from* 20 *down to* -20. *Using the* `Exclude` *command, remove from that set the element* 0.

7.2.7 Cartesian products

To create the set

$$\{(a^2, b) \mid 1 \leq a \leq 10,\ -2 \leq b \leq 6\},$$

type

```
> L:=[];
> for i in [1..10] do
for> L:=Append(L,i^2);
for> end for;
> M:=[];
> for i in [-2..6] do
for> M:=Append(M,i);
for> end for;
> LxM:=[<x,y>:x in L,y in M];
```

Now `> LxM;` will return

```
[ <1, -2>, <4, -2>, <9, -2>, <16, -2>, <25, -2>, <36, -2>, <49, -2>, <64, -2>,
<81, -2>, <100, -2>, <1, -1>, <4, -1>, <9, -1>, <16, -1>, <25, -1>, <36, -1>,
<49, -1>, <64, -1>, <81, -1>, <100, -1>, <1, 0>, <4, 0>, <9, 0>, <16, 0>, <25,
0>, <36, 0>, <49, 0>, <64, 0>, <81, 0>, <100, 0>, <1, 1>, <4, 1>, <9, 1>, <16,
1>, <25, 1>, <36, 1>, <49, 1>, <64, 1>, <81, 1>, <100, 1>, <1, 2>, <4, 2>, <9,
2>, <16, 2>, <25, 2>, <36, 2>, <49, 2>, <64, 2>, <81, 2>, <100, 2>, <1, 3>, <4,
3>, <9, 3>, <16, 3>, <25, 3>, <36, 3>, <49, 3>, <64, 3>, <81, 3>, <100, 3>, <1,
4>, <4, 4>, <9, 4>, <16, 4>, <25, 4>, <36, 4>, <49, 4>, <64, 4>, <81, 4>, <100,
4>, <1, 5>, <4, 5>, <9, 5>, <16, 5>, <25, 5>, <36, 5>, <49, 5>, <64, 5>, <81,
5>, <100, 5>, <1, 6>, <4, 6>, <9, 6>, <16, 6>, <25, 6>, <36, 6>, <49, 6>, <64,
6>, <81, 6>, <100, 6> ]
```

Exercise 7.2.8. *Create the list of all pairs of integers of the form* (n, n^2+1), $1 \leq n \leq 20$.

There are also commands `CartesianProduct` and `CartesianPower` in MAGMA. However, they do not return a list or sequence structure, even if you are taking the cartesian product of lists.

7.2.8 Sets

Sets are unordered lists without repeated elements. To construct the set of all squares of integers from 1 to 10, you could type

```
> S:={ 1, 4, 9, 16, 25, 36, 49, 64, 81, 100 };
```

or you could type

```
> S:={};
> for i in [1..10] do
for> S:=Include(S,i^2);
for> end for;
```

as in the case of lists, except we use `Include` rather than `Append`. To find the set of all squares of integers Modulo 3 from 1 to 10, you

```
> S:={};
> for i in [1..10] do
for> S:=Include(S,i^2 mod 3);
for> end for;
> S;
{ 0, 1 }
```

Exercise 7.2.9. *Create the set of all integers of the form* n^3 *mod* 3, $1 \leq n \leq 20$.

7.2.9 Adding numbers in a list

Here's a handy trick for adding (reps., multiplying) all the elements of a list. To add all the squares of the integers from 1 to 100 is easy:

```
> L:=[];
> for i in [1..10] do
for> L:=Append(L,i^2);
for> end for;
> sum:=&+L;
> sum;
385
```

The

```
&+
```

command iterates the + command over all the elements of L. Likewise, to find their product, now type

```
> product:=&*L;
> product;
13168189440000
```

Exercise 7.2.10. *Compute the sum of all integers of the form* n^3, $1 \leq n \leq 20$.

Exercise 7.2.11. *Compute the sum of all numbers of the form* $(-1)^n/n$, $1 \leq n \leq 1000$ *and* n *odd.*

The number of elements in a list L is obtained by typing the command #L;. For example,

```
> L:=[];
> for i in [1..10] do
for> L:=Append(L,i^2);
for> end for;
> #L;
```

returns 10.

7.2.10 Membership

It is also pretty easy to check whether or not something is an element of a list:

```
> 3 in List2;
true
> 11 in List2;
true
```

To find out how many times 11 occurs in List2 is a little trickier. It must first be converted to a "MultiSet" which has the Multiplicity command:

```
> MultList21:=SequenceToMultiset(List2);
> Type(MultList21);
SetMulti
> MultList2:=SequenceToMultiset(List2);
> Type(MultList2);
SetMulti
> Multiplicity(MultList2,11);
4
```

7.2.11 More complicated sets

How do you form the Cartesian product of a list with itself? In some cases, it is better to use "indexed sets". To take the $2nd$ power of $\{2, 4, 8\}$ with itself and then select the $3rd$ element: first create this as a set, then use the `Subsequences( ,2);` command, then convert that to an "indexed set" using the `SetToIndexedSet` command, as follows.

```
> Set1:={2^i:i in [1..3]};
> Set1_2:=Subsequences(Set1,2);
> Set1_2_2:=SetToIndexedSet(Set1_2);
> Set1_2_2[3];
[ 2, 8 ]
```

We could have entered $\{2, 4, 8\}$ as an indexed set by typing `Set1:={@2^i:i in [1..3]@};` but then MAGMA would've objected to the next line since `Subsequences` is not defined for indexed sets.

To add an element to a sequence or set you can use the `Include` command. For example,

```
> newset:=Include(Set1_2_2,[333]);
> newset;
{@
    [ 8, 2 ],
    [ 4, 2 ],
    [ 2, 8 ],
    [ 8, 4 ],
    [ 4, 4 ],
```

```
    [ 8, 8 ],
    [ 2, 2 ],
    [ 4, 8 ],
    [ 2, 4 ],
    [ 333 ]
@}
```

Suppose you want to select a subset of a set having a certain property, how do you do this? Take, for example, the above cartesian product $\{(i,j) \mid i,j \in \{2,4,8\}\}$, and find all the elements which add to 6.

```
> Set1_2_2;
{@
    [ 8, 2 ],
    [ 4, 2 ],
    [ 2, 8 ],
    [ 8, 4 ],
    [ 4, 4 ],
    [ 8, 8 ],
    [ 2, 2 ],
    [ 4, 8 ],
    [ 2, 4 ]
@}
> L:=[ x : x in Set1_2_2 | x[1]+x[2] eq 6];
> L;
[
    [ 4, 2 ],
    [ 2, 4 ]
]
```

7.2.12 for loops

There are basically two "flavors" of for loops, similar to the MAPLE for loops:

```
for i := <expression1> to <expression2> by <expression3> do
  <statements>
end for;
```

(if "`by <expression3>` " is omitted then it automatically increments by +1) and

```
for s in S do
  <statements>
end for;
```

For example,

```
> a:=0;
> for i:=1 to 3 do a:=i+a; end for;
> a;
6
```

and

```
> a:=0;
> for i in [1..3] do a:=i+a; end for;
> a;
6
```

have the same effect.

Exercise 7.2.12. *Construct a for loop over* $i = 1, ..., 20$ *which appends* $[i, \texttt{NextPrime(i)}]$.

Remark 7.2.13. *There is also case statement and a while loop but we shall not discuss these here.*

In some cases, it appears that MAGMA wants you to type the "for do" statement on one line, the "statements" on the next line, and the "end for" on the last line. In other cases, having them all on one line is fine.

7.2.13 if then statements

There are basically two "flavors" of if then statements, similar to the MAPLE:

```
if {\it Boolean expression} then {\it statements} else {\it more statements} end if;
```

and

```
if {\it Boolean expression} then {\it statements} end if;
```

What is a "Boolean expression"? It is usually a statement using the Boolean operators `x and y`, `x or y`, `x xor y`, `not x`, `x in S`, `x eq y` (for $x = y$), `x ne y` (for $x \neq y$), `x ge y` (for $x \geq y$), `x le y` (for $x \leq y$), and a few others.

For example,

```
> if 2^5 eq 32  then
if> print "they're equal";
if> else print "no they're not";
if> end if;
they're equal
```

and

```
> if 2^5 in [1..31]  then k:=0; else k:=1; end if;
> k;
1
```

Exercise 7.2.14. *First, initialize a sequence L to be empty by typing* `L:=[];`. *Next, construct a for loop over a variable x in the set S of integers given by $S = \{-20, ..., 20\}$ which appends x if it is odd.*

Remark 7.2.15. *In some cases, it appears that MAGMA wants you to type the "if then" statent on one line, the "statements" on the next line, and the "end if" on the last line. In other cases, having them all on one line is fine.*

7.2.14 MAGMA functions and procedures

A function is a procedure which returns something. (These things are called procedures in MAPLE.) Unlike MAPLE, you do not declare local or global variables.

Here's an example from the documentation [MAGMA]:

```
> fibonacci := function(n)
function>     if n le 2 then
function|if>        return 1;
function|if>     else
function|if>        return $$(n-1) + $$(n-2);
```

```
function|if>     end if;
function> end function;
> fibonacci(10)+fibonacci(12);
199
```

Here's an example of a silly procedure;

```
> procedure CheckEquality(x, y,~h)
procedure>      if x eq y then
procedure|if>          h := true;
procedure|if>      else
procedure|if>          h := false;
procedure|if>      end if;
procedure>  end procedure;
> CheckEquality(1,2,x);

>> CheckEquality(1,2,x);
                  ^
Runtime error in procedure call: Argument 3 must be a variable reference (use ~)
> CheckEquality(1,2,~x);
> x;
false
```

You must use `~x` in place of `x` in the third argument since it is an undeclared variable.

Exercise 7.2.16. *Write a function which takes an integer input and returns "even" if it is even and "odd" otherwise.*

Exercise 7.2.17.

7.2.15 Printing

There is no "print file" option (as far as I know) which sents the output of the MAGMA session to a printer attached to the computer. To do that, you must log the session using `SetLogFile` and then print out that file using some other program. There is a `PrintFile` command which prints the output to a file.

To have MAGMA print out a string or some variable values to the screen, use the `print` command. For example,

```
> a:=3;
> b:=5;
> print "a is equal to ",a,"\n","b equals  ",b;
a is equal to  3
 b equals   5
```

Note the `\n` is the newline character.

Exercise 7.2.18. *write a for loop over* $i = 1, ..., 10$ *which prints out, on separate lines for each* i*, the pair* (i, i^2)*.*

7.2.16 Basic commands

7.2.17 Calculus

Some of the calculus commands are `Derivative`, `Integral`, `Interpolation`, `SimpsonQuadrature`, `TrapezoidalQuadrature`, `InfiniteSum`, ...

7.2.18 Number theory

Some of the number theory commands are `n mod m`, `v div w`, `IsPrime`, `EulerPhi`, `DivisorSigma`, `GreatestCommonDivisor`, and lots of others. It also has an extensive finite field and number field package.

7.2.19 Combinatorics

Some of the combinatorics commands are `Binomial`, `Multinomial`, `Factorial`, `Partitions`, $\widehat{\text{N}}$umberOfPartitions, `RestrictedPartitions`, `StirlingFirst`, `StirlingSecond`, ...

7.2.20 Linear algebra

Some of the linear algebra commands are `DiagonalMatrix`, `Matrix`, `LowerTriangularMatrix`, `Submatrix`, `SwapRows`, `AddColumn`, `MultiplyRow`, `Nullspace`, `Determinant`, `CharacteristicPolynomial`, `Trace`, `Eigenvalues`, `MinimalPolynomial`, `Eigenspace`, `Rank`, `JordanForm`, `SmithForm`, `ElementaryDivisors`, and lots of others.

7.2.21 Plane curves

Some commands relating to the study of algebraic curves in the plane are `AffineSpace`, `ProjectiveSpace` , `Curve`, `Ideal`, `CoordinateRing` , `Degree` , `Genus` , `SingularPoints`, `IsIrreducible`, `IsNonSingular`, and lots of others.

7.2.22 Solutions to exercises

solution 7.2.19.
```
> P<a,t>:=PolynomialRing(Rationals(),2);
> g:=t^2+t/2+a;
> Derivative(g,t);
2*t + 1/2
> Integral(g,t);
a*t + 1/3*t^3 + 1/4*t^2
```

solution 7.2.20.
```
> R := RealField();
> f := map< R -> R | x :-> Cos(x) >;
> Integral(f,0,1);
0.84147098480789650666525023234
```

solution 7.2.21.
```
> f := map< Integers() -> R | x :-> 1/2^x >;
> InfiniteSum(f,1);
10141204801825835211973625643007/10141204801825835211973625643008
```
for the fractional approximation and

```
> f := map< Integers() -> R | x :-> (1.0)/2^x >;
> InfiniteSum(f,1);
0.99999999999999999999999999999990139238
```

for the decimal approximation.

solution 7.2.22.
```
> S:=[i : i in [-20..20]];
> T:=[j : j in S | j/2 in Integers()];
> T;
[ -20, -18, -16, -14, -12, -10, -8, -6, -4, -2, 0, 2, 4, 6, 8, 10, 12, 14, 16,
18, 20 ]
```

solution 7.2.23.
```
> S:=[i : i in [-20..20]];
> RS:=Reverse(S);
> RS0:=Exclude(RS,0);
> RS0;
[ 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, -1, -2,
-3, -4, -5, -6, -7, -8, -9, -10, -11, -12, -13, -14, -15, -16, -17, -18, -19,
-20 ]
```

solution 7.2.24.
```
> L:=[];
> for x in [1..20] do
for> L:=Append(L,[x,NextPrime(x)]);
for> end for;
> L;
[
```

```
    [ 1, 2 ],
    [ 2, 3 ],
    [ 3, 5 ],
    [ 4, 5 ],
    [ 5, 7 ],
    [ 6, 7 ],
    [ 7, 11 ],
    [ 8, 11 ],
    [ 9, 11 ],
    [ 10, 11 ],
    [ 11, 13 ],
    [ 12, 13 ],
    [ 13, 17 ],
    [ 14, 17 ],
    [ 15, 17 ],
    [ 16, 17 ],
    [ 17, 19 ],
    [ 18, 19 ],
    [ 19, 23 ],
    [ 20, 23 ]
]
```

solution 7.2.25.
```
> L:=[];
> for x in S do
for> if not(x/2 in Integers()) then
for|if> L:=Append(L,x);
for|if> end if;
for> end for;
> L;
[ -19, -17, -15, -13, -11, -9, -7, -5, -3, -1, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19
]
```

solution 7.2.26.
```
> function whattype(n)
function> if n/2 in Integers() then return "even"; end if;
function> if not(n/2 in Integers()) then return "odd"; end if;
function> end function;
> whattype(3);
odd
> whattype(10);
even
```

solution 7.2.27.
```
> for i:= 1 to 10 do
```

```
for> print i,i^2,"\n";
for> end for;
1 1

2 4

3 9

4 16

5 25

6 36

7 49

8 64

9 81

10 100
```

This discussion has only given an overview of some of the most basic features of MAGMA. Think of MAGMA as a very large onion and we have only begun to peel off the first layer!

For more details, see the documentation [MAGMA], or the papers [Magma1], [Magma2].

Bibliography

[AS] W. Adams and D. Shanks, "Strong primality tests that are not sufficient," Math. Comp. 39 (1982), no. 159, 255–300.

[Ar] M. Artin, **Algebra**, Prentice-Hall, 1991

[Art] J. Arthur, "Automorphic representations and number theory", **1980 Seminar on Harmonic Analysis**, (C. Herz, R. Rigelhof, ed.), CMS Conf. Proc. vol 1, 1981.

[Ash] R. Ash, **Information theory**, Dover, 1965

[AK] E. Assmus, J. Key, **Designs and their codes**, Cambridge University Press, 1992

[AM] E. Assmus, Jr. and H. Mattson, "On the auotmorphism groups of Paley-Hadamard matrices", in **Combinatorial mathematics and its applications**, ed. R. Bose, T. Dowling, Univ of North Carolina Press, Chapel Hill, 1969

[B] E. Berlekamp, "Block coding with noiseless feedback," PhD Thesis, Dept EE, MIT, 1964

[BCG] Berlekamp, J. Conway, R. Guy, **Winning ways, II**, Academic Press, 1982

[BC] W. W. R. Ball and H. S. M. Coxeter, **Mathematical recreation and essays**, 13th ed, Dover, 1987

[BFR] E. Bonsdorff, K. Fabel, O. Riihimaa, **Schach und zahl**, Walter Rau Verlag, Düsseldorf, 1966

[BS] E. Bach and J. Shallit, **Algorithmic number theory, vol I**, MIT Press, 1997

[BG] I. Blake and I. Gibson, "Decoding the binary Golay code with miracle octad generators," IEEE Trans. Infor. Theory 24(1978)261-264.

[Br] J. Bruce, "A really trivial proof of the Lucas-Lehmer test," Amer. Math. Monthly, April 1993, pages 370-371.

[Bu] B. Buchburger, "Gröbner bases: an algorithmic method in polynomial ideal theory", in N. Bose (ed.), **Multidimensional systems theory**, D. Reidel Pub. Co., 1985

[C] John H. Conway, **On numbers and games**, Academic Press, 1976

[CS] J. Conway and N. Sloane, **Sphere packings, lattices, and groups**, Springer-Verlag, 1993

[CS2] ——, "Lexicographic codes: error-correcting codes from game theory", IEEE Transactions on Information Theory, 32(1986)337-348

[CSW] —— and A. Wilks, "Gray codes and reflection groups", Graphs and combinatorics 5(1989)315-325

[CLO] D. Cox, J. Little, D. O'Shea, **Ideals, varieties, and algorithms**, Springer-Verlag

[CCNPW] J. H. Conway, R. T. Curtis, S. P. Norton, R. A. Parker, R. A. Wilson, "M12," in **Atlas of Finite Groups**, Clarendon Press, Oxford, 1985.

[CFS] G. Cooperman, L. Finkelstein and N. Sarawagi, "Applications of Cayley graphs", in **Applied algebra...**, Springer-Verlag, Lecture Notes in Computer Science, 508, 1990

[CP] R. Crandall, C. Pomerance **Prime Numbers**, telos Press, 2001.

[CG] S. Curran and J. Gallian, "Hamiltonian cycles and paths in Cayley graphs and diagraphs - survey", Discrete Math. 156(1996)1-18

[CL] ftp archives of the `cube-lovers list` at
`ftp://ftp.ai.mit.edu/pub/cube-lovers/`

[Co] J. Cosgrave, "Number Theory and Cryptography (using Maple)," in **Coding Theory and Cryptology: From Enigma and Geheimschreiber to Quantum Theory** (ed. D. Joyner), New York: Springer Verlag, Lecture Notes in Computational Science and Engineering (1999)

[Cu] R. Curtis, "The Steiner system $S(5,6,12)$, the Mathieu group M_{12}, and the kitten," in **Computational group theory**, ed. M. Atkinson, Academic Press, 1984.

[E] J. Ellis, "The history of non-secret encryption," Cryptologia, Volume XXIII Number 3 (July 1999) , 267-273

[FKL] M. Fossorier, Y. Kou, S. Lin, "Low density parity check codes based on finite geometries: a rediscovery and new results," IEEE Trans. Info. Theory 47(2001)27111-2736

[FS] A. Frey and D. Singmaster, **Handbook of cubik math**, Enslow Pub., 1982

[FST] T. Fuja, D. Srihara, R. M. Tanner, "A class of group-structures LDPC codes," preprint

[Ga] R.G. Gallagher, "Low density parity check codes," IRE Trans. Info. Theory 8(1962)21-28

[Gap] The GAP Group, **GAP – Groups, Algorithms, and Programming, Version 4.2**; 2000, (`http://www.gap-system.org`).

[Gar1] M. Gardner, "Combinatorial card problems" in **Time travel and other mathematical bewilderments**, W. H. Freeman, New York, 1988

[Gol] S. Golomb, **Shift register sequences**, Aegean Park Press, Laguna Hills, Ca, 1967

[Go] V. D. Goppa, **Geometry and codes**, Kluwer, 1988.

[G] A. Granville, "Zaphod Beeblebrox's brain and the fifty-ninth row of Pascal's triangle," Amer. Math. Monthly, April 1992, pp 318-331

[GJ] M. Garey and D. Johnson, **Computers and intractibility**, W. H. Freeman, New York, 1979

[GK1] A. Klapper and M. Goresky, "2-adic shift registers," in *Fast Software Encryption* (Springer Lecture Notes in Computer Science, 1995), New York, NY, pp. 170-178

[GK2] A. Klapper and M. Goresky, "Feedback shift registers, 2-adic span, and combiners with memory," Journal of Cryptology **10** (1997) 111-147.

[GLS] D. Gorenstein, R. Lyons, and R. Solomon, **The Classification of the Finite Simple Groups**, AMS, 1994.
`http://www.ams.org/online_bks/surv40-1/`

[HHLO] Heikki Hamalainen, Iiro Honkala, Simon Litsyn, Patric Ostergard, "Football Pools–A Game for Mathematicians," American Mathematical Monthly, Vol. 102, No. 7. (Aug. - Sep., 1995), pp. 579-588.

[HW] G. Hardy and E. Wright, **An introduction to the theory of numbers**, 5^{th} ed, Oxford Univ Press, 1979

[Her] I. N. Herstein, **Abstract algebra, 3^{rd} ed.**, Prentice Hall, 1996.

[Hi] R. Hill, "Searching with lies,", in **Surveys in Combinatorics**, ed. by P. Rowlinson, London Math Soc, Lecture Notes Series # 218

[H] D. Hofstadler, **Metamathematical themas**, Basics Books, 1985 (Mostly a collection of Scientific American columns he wrote; the articles referred to here were also published in Scientific American, March 1981, July 1982)

[Hum] J. Humphreys, **Reflection groups and coxeter groups**, Cambridge Univ Press, 1990

[J1] D. Joyner, **Adventures in group theory**, Johns Hopkins Unv. Press, 2002.

[J2] ——, *golay12.g package*, at `http://web.usna.navy.mil/~wdj/codes/golay12.g`

[JN] ——, D. Joyner, **Linear algebra and applications**, Brooks-Cole, 1998

[Ki] J. Kirkland, **Identification numbers and check digit schemes**, Classroom resource materials, MAA, 2001

[Kn] D. Knuth, **The art of computer programming: volume 2**, 3rd edition, Addison-Wesley-Longman, 1998

[K] R. Kreminski, at `http://www.tamu-comerce.edu/coas/math/FACULTY/KREMIN/kremin.html`

[LR] J. Lafferty, D. Rockmore, "Spectral techniques for expander codes," preprint

[La] S. Lang, **Algebra,** 2^{nd} **ed.**, Addison-Wesley, 1984.

[LP] R. Lidl, G. Pilz, **Applied abstract algebra,** 2^{nd} **ed.**, Springer, 1998.

[Lu] A. Luers, "The group structure of the pyraminx and the dodecahedral faces of M_{12}", USNA Honors thesis, 1997 (Advisor W. D. Joyner)

`http://web.usna.navy.mil/~wdj/m_12.htm`

[Mac] D. MacKay, "Good error-correcting codes based on very sparse matrices," IEEE Trans. Info. Theory 45 (1999)399-431 (errata: 47(2001)2101)

[MN] ——, R. Neal, "Near Shannon limit performance of low density parity check codes," preprint, 1996 (appeared in Electronic Letters)

[MacOR] John J O'Connor, Edmund F Robertson, **MacTutor History of Mathematics archive**, at `http://www-groups.dcs.st-and.ac.uk/~history/index.html`

[MS] F. MacWilliams and N. Sloane, **The theory of error-correcting codes**, North-Holland, 1977. See also N. Sloane's web site `http://www.research.att.com/~njas/index.html`

[MAGMA] W. Bosma, J. Cannon, C. Playoust, "The MAGMA algebra system, I: The user language," J. Symb. Comp., 24(1997)235-265. (See also the MAGMA homepage at `http://www.maths.usyd.edu.au:8000/u/magma/`)

[Magma1] W. Bosma, J. Cannon, C. Playoust, A. Steel, **Solving problems with MAGMA**, Univ Sydney, 1999.

[Magma2] J. Cannon, C. Playoust, **An introduction to algebraic programming with MAGMA**, Univ Sydney, 1999.

[MKS] W. Magnus, A. Karrus and D. Solitar, **Combinatorial group theory**, 2nd ed, Dover, 1976

[MOV] Alfred J. Menezes, Paul C. Van Oorschot, Scott A. Vanstone **Handbook of applied cryptography**, CRC Press Series on Discrete Mathematics and Its Applications, CRC Press, 1996.

[Mont] J. Montague, "Searching with lies", USNA Honors thesis, 1998 Available on the web at
`http://web.usna.navy.mil/~wdj/montague.htm`

[MZ] G. Marsaglia and A. Zaman, "A new class of random number generators," Annals of Applied Probability **1**(1991)462-490

[NG] G. Nakos and N. Glinos, "Computing Gröbner bases over $\mathbb{Z}$", Mathematica Journal, ...

[N] I. Niven, "Coding theory applied to a problem of Ulam," Math Mag 61(1988)275-281

[NST] P. Neumann, G. Stoy and E. Thompson, **Groups and geometry**, Oxford Univ. Press, 1994

[Pim] A. Pimlott, "How to get rich off Finnish football fans," Harvard Tangents, issue 1.1, 1994

[Pl] V. Pless, "Decoding the Golay codes," IEEE Trans. Infor. Theory 32(1986)561-567.

[P] O. Pretzel, **Codes and algebraic curves**, Oxford Lecture Series, vol 9, Clarendon Press, Oxford, 1998

[RSA] R.L. Rivest, A. Shamir, and L.M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, 21(February 1978), 120-126

[Ro] G. Robin, "Grandes valuers de la fonction somme des diviseurs et hypthése de Riemann," J. Math. Pures Appl. 63(1984)187-213.

[Rom] Steven Roman, **Coding and information theory**, Springer-Verlag, 1992

[R] J. J. Rotman, **An introduction to the theory of groups**, 4th ed, Springer-Verlag, Grad Texts in Math 148, 1995

[Sch] M. R. Schroeder, **Number theory in science and communication,** 3^{rd} **ed.**, Springer, 1997.

[Ser] J.-P. Serre, **Trees**, Springer-Verlag, 1980

[Si] D. Singmaster, **Notes on Rubik's magic cube**, Enslow, 1981

[SING] SINGULAR, A Computer Algebra System for Polynomial Computations. See the SINGULAR homepage at
`http://www.singular.uni-kl.de/`)

[Sp] D. Spielman, "Linear time encodable and decodable error-correcting codes," IEEE Trans. Info. Theory 42(1996)1723-1731

[St] H. Stark, "A complete determination of the complex quadratic fields of class-number one," Michigan Math. J. 14(1967)1–27.

[TW] A. D. Thomas and G. V. Wood, **Group tables**, Shiva Publishing Ltd, Kent, UK, 1980

[Th] W. Thurston, "Conway's tiling groups," Amer. Math. Monthly 97 (1990)757–773

[U] S. Ulam, **Adventures of a mathematician**, Scribner and Sons, New York, 1976

[vdP] A. J. van der Poorten, "Notes on continued fractions and recurrence sequences," in **Number theory and cryptography** (ed. J. H. Loxton), London Math Soc. Lecture Notes, Cambridge Univ Press, 1990

[Va] R. C. Vaughan, **The Hardy-Littlewood method**, Cambridge Univ. Press, 1981

[V] J. Verhoeff, **Error detecting decimal codes**, Math. Centre, Amsterdam, 1969

[Wa] W. P. Wardlaw, "The RSA Public Key Cryptosystem," in **Coding Theory and Cryptology: From Enigma and Geheimschreiber to Quantum Theory** (ed. D. Joyner), New York: Springer Verlag, Lecture Notes in Computational Science and Engineering (1999)

[Wh] White, Arthur, "Fabian Stedman: The First Group Theorist?", American Mathematical Monthly, Nov. 1996, pp771-8.

Index