A Unified Approach to HGCD Algorithms for polynomials and integers

Klaus Thull and Chee K. Yap*

Freie Universität Berlin
Fachbereich Mathematik
Arnimallee 2-6
D-1000 Berlin 33
West Germany

March, 1990

Abstract

We present a unified framework for the asymptotically fast Half-GCD (HGCD) algorithms, based on properties of the norm. Two other benefits of our approach are (a) a simplified correctness proof of the polynomial HGCD algorithm and (b) the first explicit integer HGCD algorithm. The integer HGCD algorithm turns out to be rather intricate.

Keywords: Integer GCD, Euclidean algorithm, Polynomial GCD, Half GCD algorithm, efficient algorithm.

*On leave from the Courant Institute, New York University, 251 Mercer Street, New York, NY 10012. This author is supported by a grant from the German Science Foundation (DFG), the ESPRIT II Basic Research Action of the E.C. under contract No.3075 (Project ALCOM), and also NSF Grant #DCR-84-01898 and #CCR-87-03458.

1 Introduction

The first subquadratic algorithm for integer GCD came from Knuth (1970) [6] who described an $O(n \log^5 n \log\log n)$ algorithm. The ideas go back to Lehmer [8] ([7, p.328]). Schönhage (1971) [11] improved it into an $O(n \log^2 n \log\log n)$ algorithm. Moenck [9] adapted Schönhage's algorithm into an $O(n \log^2 n)$ algorithm for polynomial GCD. But it's correctness was restricted to input polynomials that form "normal remainder sequences". The widely cited book of Aho, Hopcroft and Ullman [1] gives a false proof of correctness for Moenck's algorithm. The Moenck-Aho-Hopcroft-Ullman algorithm is erroneous for non-trivial reasons, and is still not completely understood – we discuss this in an appendix. Brent-Gustavson-Yun [2] published a little-known[1] corrected algorithm, but without proof of correctness. Since correctness is not obvious, and the approach of [1] seems to be less than transparent (considering how its incorrectness had eluded detection), it seems desirable to have for the record an incontrovertible algorithm. We provide a such correctness proof here, which we believe is new and simple.

One benefit of our unified approach is that it almost gives two algorithms for the price of one: we will present a parallel development for an integer HGCD algorithm. In fact, there has not been a previous Half-GCD algorithm, and the one here has various subtleties. Finally, our framework can perhaps be extended to other Euclidean domains (recall that GCD's are more generally defined in unique factorization domains) for which there are currently no subquadratic GCD algorithms (e.g., Gaussian integers). Currently, the most general GCD algorithms are due to Kaltofen and Rolletschek [5]; their GCD algorithms work in quadratic integer domains which need be Euclidean.

One caveat is that although the algorithms of Schönhage and Moenck, *et al* are worst-case asymptotically the fastest for their respective domains, it seems that modular methods are favored in practice. Also, the use of probabilistic algorithms (see Schönhage [12], Kaltofen [4]) may be more practical. There is also an entirely different development of GCD algorithms over the ring $D[x]$ where D is an integral domain [3]. Finally, Strassen [13] has shown that the method of Schönhage is asymptotically optimal with a bound of $\Theta(n \log n)$, using a different model of computation (counting only non-scalar multiplications).

Notations. In this paper, we are interested in two domains, the ring of integers and the polynomial ring $K[x]$ over some field K. We let D denote either of these domains. We use the infix binary operators **div** and **mod** to denote the quotient and remainder functions (see below for precise definition) on D. As expected, the following equation holds: $a = (a \textbf{ div } b) \cdot b + (a \textbf{ mod } b)$, where $a, b \in D$, $b \neq 0$. The *(Euclidean) remainder sequence* for a, b is the sequence

[1] In fact, we had independently developed our algorithm in ignorance of their result. An earlier paper of Yun [14] also noted an error in the algorithm of [1] but there he seems to be referring to what we called the "easily-fixed" error in the appendix.

$(a_0, a_1, \ldots, a_k)$ of non-zero elements where $a_{i+1} = a_{i-1} \bmod a_i$ $(i = 1, \ldots, k-1)$ and $0 = a_{k-1} \bmod a_k$.

Half-GCD. Let us briefly recall that algorithm of Moenck is based on the 'Half-GCD' function, HGCD(a, b). By definition, this function on input polynomials $a, b \in K[x]$ $(\deg(a) > \deg(b) \geq 0)$, outputs the 2 by 2 matrix R such that if $\begin{pmatrix} c \\ d \end{pmatrix} = R \begin{pmatrix} a \\ b \end{pmatrix}$ then

$$\deg(c) \geq \lceil \deg(a)/2 \rceil > \deg(d).$$

Once we have a Half-GCD algorithm, it is a simple matter to implement an algorithm for the GCD running in the same asymptotic time complexity (see [1]). An Integer HGCD function can also be defined (see below) with similar properties. So in the rest of this paper, we focus on the Half-GCD problem.

2 Some properties of the Norm

We review some familiar properties of polynomial and integer that form the basis for the HGCD algorithms. Let D be either the ring of integers or the polynomial ring $K[x]$ over a field K.

Define the *norm* $\|a\|$ of $a \in D$ to be $\log_2 |a|$ in case a is an integer, and $\deg(a)$ in case a is a polynomial. Hence

$$\|a\| \in \{-\infty\} \cup \mathbf{R}^*$$

($\mathbf{R}^*$ is the set of non-negative real numbers) with $\|a\| = -\infty$ iff $a = 0$. This norm is an (additive) valuation in the usual sense that the following two additional properties hold:

$$\|ab\| = \|a\| + \|b\|$$

and

$$\|a + b\| \leq 1 + \max\{\|a\|, \|b\|\}.$$

However, polynomials satisfy the stronger *non-Archimedean property*:

$$\|a + b\| \leq \max\{\|a\|, \|b\|\}.$$

It is the non-Archimedean property that makes polynomials relatively easier than integers. This property implies that $\|a + b\| = \max\{\|a\|, \|b\|\}$ if $\|a\| \neq \|b\|$.

The *division property* for D relative to the norm $\|\cdot\|$ holds: for any $a, b \in D$, $b \neq 0$, there exists $q, r \in D$ such that

$$a = qb + r, \quad \|r\| < \|b\|.$$

We call r a *remainder* of the pair (a, b). In the polynomial case, the remainder is unique. For integers, a zero remainder is unique. Otherwise, (a, b) has two

remainders, one positive and one negative. In this paper, we define the function a **mod** b to always pick the non-negative remainder. In the polynomial case, we have

$$\|a \textbf{ mod } x^m\| \leq \min\{\|a\|, m-1\} \tag{1}$$

$$\|a \textbf{ div } x^m\| = \begin{cases} \|a\| - m & \text{if } \|a\| \geq m \\ -\infty & \text{else.} \end{cases} \tag{2}$$

We let $\gcd(a,b)$ denote the greatest common divisor of $a, b \in D$. Since the greatest common divisor is defined up to associates, we specify $\gcd(a,b)$ to be non-negative for integers and monic for polynomials.

In this paper, 'matrices' and 'vectors' are understood to be 2 by 2 matrices and column 2-vectors, respectively. A matrix of the form $Q = \begin{pmatrix} q & 1 \\ 1 & 0 \end{pmatrix}$, where $\|q\| > 0$ in the case of polynomials, and $q > 0$ in the case of integers, is said to be *elementary*, and q is called the *partial quotient* of Q. We also write $Q = \langle q \rangle$ in this case. A *regular matrix* $Q = Q_1 \cdots Q_k$ is a product of some $k \geq 0$ elementary matrices. If $Q_i = \langle q_i \rangle$ for all i then Q is also denoted $\langle q_1, \ldots, q_k \rangle$. When $k = 0$, Q is the identity matrix, denoted by E or $\langle\rangle$. Regular matrices satisfy the following *ordering property*: If $Q = \begin{pmatrix} p & q \\ r & s \end{pmatrix}$ is regular, then

$$Q \neq E \Rightarrow \quad \|p\| \geq \max\{\|q\|, \|r\|\} \geq \min\{\|q\|, \|r\|\} \geq \|s\|, \quad \|p\| > \|s\|. \tag{3}$$

For vectors U, V and matrix Q, we write

$$U \xrightarrow{Q} V$$

(or simply, $U \longrightarrow V$) and say V is a *reduction* of U (*via* Q) if $U = QV$ where Q is a regular matrix. If, in addition, $U = \begin{pmatrix} a \\ b \end{pmatrix}, V = \begin{pmatrix} a' \\ b' \end{pmatrix}$ such that $\|a\| > \|b\|$ and $\|a'\| > \|b'\|$, then we say this is an *Euclidean reduction*. Clearly if $U \xrightarrow{Q} V$ and $V \xrightarrow{Q'} W$ then $U \xrightarrow{QQ'} W$. Observe that Euclid's algorithm for a pair $a, b \in D$ ($\|a\| > \|b\|$) can be regarded as a sequence of Euclidean reductions.

Note that regular matrices are unimodular (determinant ± 1), and hence they are trivial to invert. We say U, V are *equivalent* if $U = QV$ for some unimodular matrix Q. If $U = \begin{pmatrix} a \\ b \end{pmatrix}$ then we write $\gcd(U)$ for the GCD of a and b. It is easy to see that U and V are equivalent if and only if $\gcd(U) = \gcd(V)$.

We use the following fact about (Euclidean) remainder sequences:

Fact 1 *Given a, b, a', b' such that $\|a\| > \|b\| \geq 0$. The following are equivalent.*
(i) *a', b' are consecutive elements in a remainder sequence of a, b.*

(ii) *There is a regular matrix Q such that*

$$\begin{pmatrix} a \\ b \end{pmatrix} \xrightarrow{Q} \begin{pmatrix} a' \\ b' \end{pmatrix} \tag{4}$$

and either $\|a'\| > \|b'\| \geq 0$ (polynomial case) or or $a' > b' > 0$ (integer case).

Proof: If (i) holds then we can (by Euclid's algorithm) find some regular matrix Q satisfying (ii). Conversely assume (ii). We show (i) by induction on the number of elementary matrices in the product Q. The result is immediate if $Q = E$. If Q is elementary, then (i) follows from the division property for D. Otherwise let $Q = Q''Q'$ where Q' is elementary and Q'' is regular. Then for some a'', b'',

$$\begin{pmatrix} a \\ b \end{pmatrix} \xrightarrow{Q''} \begin{pmatrix} a'' \\ b'' \end{pmatrix} \xrightarrow{Q'} \begin{pmatrix} a' \\ b' \end{pmatrix}.$$

But $a'' = a'q' + b', b'' = a'$ where q' is the partial quotient of Q'. We verify that this means $\|a''\| > \|b''\|$. By induction, a'', b'' are consecutive elements in a remainder sequence of a, b. Then (i) follows. Q.E.D.

Note that we require $b' \neq 0$ in (ii). If $b' \neq 0$ then $a' = \gcd(a, b)$.

3 The Polynomial Case

We describe a fast polynomial HGCD algorithm and prove its correctness. The correctness criteria for the HGCD algorithm is that on input $a, b \in K[x]$ with $\deg(a) > \deg(b) \geq 0$, the output is a regular matrix Q such that

$$\begin{pmatrix} a \\ b \end{pmatrix} \xrightarrow{Q} \begin{pmatrix} c' \\ d' \end{pmatrix}, \qquad \deg(c') \geq \left\lceil \frac{\deg(a)}{2} \right\rceil > \deg(d'). \tag{5}$$

ALGORITHM POLYNOMIAL HGCD(a, b):

[1] $m \leftarrow \left\lceil \frac{\deg(a)}{2} \right\rceil$; {This is the magic threshold}
if $\deg(b) < m$ then return(E);

[2] $a_0 \leftarrow a$ **div** x^m; $b_0 \leftarrow b$ **div** x^m;
{now $\deg(a_0) = m'$ where $m + m' = \deg(a)$}
$R \leftarrow$ **HGCD**(a_0, b_0);
{$\left\lceil \frac{m'}{2} \right\rceil$ is the magic threshold for this recursive call}
$\begin{pmatrix} a' \\ b' \end{pmatrix} \leftarrow R^{-1} \begin{pmatrix} a \\ b \end{pmatrix}$;

[3] if $\deg(b') < m$ then return(R);

[4] $q \leftarrow a'$ **div** b'; $\begin{pmatrix} c \\ d \end{pmatrix} \leftarrow \begin{pmatrix} b' \\ a' \textbf{ mod } b' \end{pmatrix}$;

[5] $l \leftarrow \deg(c)$; $k \leftarrow 2m - l$; {now $l - m < \left\lceil \frac{m'}{2} \right\rceil$}

[6] $c_0 \leftarrow c$ **div** x^k; $d_0 \leftarrow d$ **div** x^k; {now $\deg(c_0) = 2(l - m)$}
$S \leftarrow$ **HGCD**(c_0, d_0); {$l - m$ is magic threshold
for this recursive call. If $\begin{pmatrix} c \\ d \end{pmatrix} = S \begin{pmatrix} c' \\ d' \end{pmatrix}$ then
$\deg(c') \geq m$ and $\deg(d') < m$.}

[7] $Q \leftarrow R \cdot \begin{pmatrix} q & 1 \\ 1 & 0 \end{pmatrix} \cdot S$; return$(Q)$;

The following correctness criteria is central to our analysis. First we set up the notations:

Let $a, b \in K[x]$, $\|a\| > \|b\| \geq 0$ and $m \geq 1$ be given. As in the HGCD algorithm above, let $a_0 = a$ **div** x^m and $b_0 = b$ **div** x^m. This determines a_1, b_1 via the equation

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a_0 x^m + a_1 \\ b_0 x^m + b_1 \end{pmatrix} = \begin{pmatrix} a_0 & a_1 \\ b_0 & b_1 \end{pmatrix} \begin{pmatrix} x^m \\ 1 \end{pmatrix}. \tag{6}$$

Now let Q be any given regular matrix. This determines a'_0, b'_0, a'_1, b'_1 via

$$\begin{pmatrix} a'_0 & a'_1 \\ b'_0 & b'_1 \end{pmatrix} = Q^{-1} \begin{pmatrix} a_0 & a_1 \\ b_0 & b_1 \end{pmatrix}. \tag{7}$$

Finally, define a', b' via

$$\begin{pmatrix} a' \\ b' \end{pmatrix} = \begin{pmatrix} a'_0 & a'_1 \\ b'_0 & b'_1 \end{pmatrix} \begin{pmatrix} x^m \\ 1 \end{pmatrix}. \tag{8}$$

Note that

$$\begin{pmatrix} a_0 \\ b_0 \end{pmatrix} \xrightarrow{Q} \begin{pmatrix} a'_0 \\ b'_0 \end{pmatrix}, \quad \begin{pmatrix} a \\ b \end{pmatrix} \xrightarrow{Q} \begin{pmatrix} a' \\ b' \end{pmatrix}.$$

Lemma 1 (Correctness Criteria) *Let a, b, m, Q be given as above, and define the remaining notations $a_i, b_i, a'_i, b'_i, a', b'$ $(i = 0, 1)$ as indicated. If*

$$\begin{aligned} \|a'_0\| &> \|b'_0\|, & (9)\\ \|a_0\| &\leq 2\|a'_0\| & (10) \end{aligned}$$

then

$$\begin{aligned} \|a'\| &= m + \|a'_0\|, & (11)\\ \|b'\| &\leq m + \max\{\|b'_0\|, \|a_0\| - \|a'_0\| - 1\}. & (12) \end{aligned}$$

In particular, $\|a'\| > \|b'\|$.

Proof: Let $Q = \begin{pmatrix} p & q \\ r & s \end{pmatrix}$. First observe that equation (9) and $a_0 = a'_0 p + b'_0 q$ implies $\|a_0\| = \|a'_0\| + \|p\|$. Hence equation (10) is equivalent to

$$\|p\| \leq \|a'_0\|. \tag{13}$$

Since $Q^{-1} = \pm \begin{pmatrix} s & -q \\ -r & p \end{pmatrix}$ and $a'_1 = \pm(a_1 s - b_1 q)$,

$$\|a'_1\| \leq \max\{\|a_1 s\|, \|b_1 q\|\} < m + \|p\| \leq m + \|a'_0\|$$

Since $a' = a'_0 x^m + a'_1$, we have proven equation (11).

From $b'_1 = \pm(-a_1 r + b_1 p)$ we get $\|b'_1\| \leq m - 1 + \|p\| = m - 1 + \|a_0\| - \|a'_0\|$. Since $b' = b'_0 x^m + b'_1$, equation (12) follows. Q.E.D.

Remarks The requirement (9) is understandable from Fact 1. But we call the requirement (10) the (lower) "threshold" for $\|a'_0\|$. This threshold is the reason for the lower bound on $\deg(c')$ in the HGCD output specification (5). The algorithm of Moenck-Aho-Hopcroft-Ullman fails because their algorithm does not respect this threshhold bound.

We are ready to prove the correctness of the HGCD algorithm.

Lemma 2 (HGCD Correctness) *Algorithm HGCD is correct: with input polynomials a, b where $\|a\| > \|b\| \geq 0$, it returns a regular matrix Q satisfying (5).*

Proof: The algorithm returns a matrix in steps [1], [3] or [7]. It is clear that in each case the returned matrix is regular. So it remains to check (5). In step [1], the result is clearly correct.

Consider the matrix R returned in step [3]: the notations m, a_0, b_0, a', b' in the algorithm conforms to those in lemma 1, after substituting R for Q. By induction hypothesis, the matrix R returned by the first recursive call (step [2]) satisfies

$$\|a'_0\| \geq \lceil \|a_0\|/2 \rceil > \|b'_0\|$$

where $\begin{pmatrix} a_0 \\ b_0 \end{pmatrix} \xrightarrow{R} \begin{pmatrix} a'_0 \\ b'_0 \end{pmatrix}$. Then lemma 1 implies $\|a'\| = m + \|a'_0\| \geq m + \lceil \|a\|/2 \rceil$. Since $m > \|b'\|$ is the condition for exit at step [3], it follows that (5) is satisfied on exit at step [3].

Finally consider the matrix Q returned in step [7]. With the notations of steps [6] and [7], it is clear that $\begin{pmatrix} a \\ b \end{pmatrix} \xrightarrow{Q} \begin{pmatrix} c' \\ d' \end{pmatrix}$. So it suffices to show $\|c'\| \geq m > \|d'\|$. Since we did not exit in step [3], we have $m \leq \|b'\|$. In step [4] we form the quotient q and remainder d of a' divided by b'. Also we renamed b' to c. Hence $m \leq l$ where $l = \|c\|$. In the second recursive call to HGCD, HGCD(c_0, d_0) returns S. By induction,

$$\|c'_0\| \geq \lceil \|c_0\|/2 \rceil > \|d'_0\| \tag{14}$$

where

$$\begin{pmatrix} c_0 \\ d_0 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} c'_0 \\ d'_0 \end{pmatrix}.$$

But $\|c_0\| = l - k = 2(l - m)$ so (14) becomes

$$\|c'_0\| \geq l - m > \|d'_0\|.$$

Now let $\begin{pmatrix} c \\ d \end{pmatrix} \xrightarrow{S} \begin{pmatrix} c' \\ d' \end{pmatrix}$. Another application of lemma 1 (substituting k for m, S for Q, c for a, d for b, etc) shows that

$$\|c'\| = k + \|c'_0\| \geq k + l - m = m$$

and

$$\begin{aligned} \|d'\| &\leq k + \max\{\|d'_0\|, \|c_0\| - \|c'_0\| - 1\} \\ &\leq k + \max\{l - m - 1, l - m - 1\} \\ &= k + l - m - 1 = m - 1. \end{aligned}$$

This shows $\|c'\| \geq m > \|d'\|$ and hence (5). Q.E.D.

The proof shows that we could have used $k = 2m - l - 1$ as well. The complexity of the HGCD algorithm here is $O(n \log^2 n)$ following the usual analysis [1].

4 The Integer Case

Let now D be the set of integers. We develop an integer version of the HGCD algorithm. Two simple but crucial tricks are used. The first is to recover the non-Archimedean property thus: for $a, b, c \in D$,

$$a = b + c, \quad bc \leq 0 \qquad \Longrightarrow \qquad \|a\| \leq \max\{\|b\|, \|c\|\}.$$

In the HGCD algorithm for polynomials a, b, we recursively call HGCD on a **div** x^m, b **div** x^m for a suitable m. The integer analogue would be to call HGCD on a **div** $2^m, b$ **div** 2^m. Instead, the second trick will call HGCD on

$$a_0 \ := \ 1 + (a \ \mathbf{div} \ 2^m), \quad b_0 \ := \ b \ \mathbf{div} \ 2^m. \tag{15}$$

Setup. The following notations will be fixed for this section, especially for the next two lemmas.

Assume that we are given $a > b > 0$ and $m \geq 1$ where $a \geq 2^m$. This determines the values a_0, a_1, b_0, b_1 via

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a_0 & -a_1 \\ b_0 & b_1 \end{pmatrix} \begin{pmatrix} 2^m \\ 1 \end{pmatrix}, \quad 0 < a_1 \leq 2^m, \quad 0 \leq b_1 < 2^m. \tag{16}$$

It is worth noting that both tricks are incorporated in (16). Defining a_0, b_0 as in (15) is the same as choosing $a_1 \ := \ 2^m - (a \ \mathbf{mod} \ 2^m)$ and $b_1 \ := \ b \ \mathbf{mod} \ 2^m$. This choice ensures $a_0 > b_0$, as required when recursively calling the algorithm on a_0, b_0.

We are also given a regular matrix Q. This determines the values $a'_0, b'_0, a'_1, b'_1, a', b'$ via

$$\begin{pmatrix} a'_0 & a'_1 \\ b'_0 & b'_1 \end{pmatrix} = Q^{-1} \begin{pmatrix} a_0 & -a_1 \\ b_0 & b_1 \end{pmatrix} \tag{17}$$

and

$$\begin{pmatrix} a' \\ b' \end{pmatrix} = \begin{pmatrix} a'_0 & a'_1 \\ b'_0 & b'_1 \end{pmatrix} \begin{pmatrix} 2^m \\ 1 \end{pmatrix}. \tag{18}$$

Hence we have

$$\begin{pmatrix} a_0 \\ b_0 \end{pmatrix} \xrightarrow{Q} \begin{pmatrix} a'_0 \\ b'_0 \end{pmatrix}, \quad \begin{pmatrix} a \\ b \end{pmatrix} \xrightarrow{Q} \begin{pmatrix} a' \\ b' \end{pmatrix}.$$

Finally, we assume two key inequalities:

$$a'_0 > b'_0 \geq 0 \tag{19}$$

$$2\|a'_0\| - 1 > \|a_0\| \tag{20}$$

Now write Q as

$$Q = \begin{pmatrix} p & q \\ r & s \end{pmatrix}, \qquad Q^{-1} = D \begin{pmatrix} s & -q \\ -r & p \end{pmatrix} \tag{21}$$

where $D = \det Q = \pm 1$. From (17) we obtain

$$a'_1 = -D(sa_1 + qb_1), \quad b'_1 = D(ra_1 + pb_1). \tag{22}$$

The proof below uses (22) to predict the signs of a'_1, b'_1, assuming the sign of D. This is possible thanks to the second trick.

The following is the integer analogue of lemma 1:

Lemma 3 (Partial Correctness Criteria)
Let a, b, m, Q be specified as above.

(–) *Suppose* $\det Q = -1$.
(–a) $\|a'\| = m + \|a'_0\| + \epsilon_1$, $(0 \leq \epsilon_1 < 1)$.
(–b) $\|b'\| \leq m + \max\{\|b'_0\|, \|a_0\| - \|a'_0\| + 1\}$.
Moreover, $\|a'\| > \|b'\|$.

(+) *Suppose* $\det Q = +1$.
(+a) $\|a'\| \leq m + \|a'_0\|$.
(+b) $\|b'\| \leq 1 + m + \max\{\|b'_0\|, \|a_0\| - \|a'_0\| + 1\}$.
Furthermore $b' \geq 0$.

In both cases (–) *and* (+), $a' > 0$.

Proof: Since $a_0 = pa'_0 + qb'_0$, Q has the ordering property (3), and (19) holds, we get

$$\|a_0\| = \|p\| + \|a'_0\| + \epsilon_2 \quad (0 \leq \epsilon_2 < 1).$$

Hence (20) is equivalent to

$$\|p\| + \epsilon_2 < \|a'_0\| - 1.$$

We now prove cases (–) and (+) in parallel.

Part (a). From (22),

$$\begin{aligned} \|a'_1\| &\leq \max\{\|sa_1\|, \|qb_1\|\} + 1 \\ &\leq \|p\| + m + 1 \qquad \text{(by (3))} \\ &< \|a'_0\| + m. \end{aligned}$$

Hence $\|a'_0 2^m\| > \|a'_1\|$ and so $a' = a'_0 2^m + a'_1 > 0$. This is as stated in the lemma. If $D = -1$ then $\|a'\| = m + \|a'_0\| + \epsilon_1$ for some $0 \leq \epsilon_1 < 1$. This proves subcase (–a). If $D = +1$ then $a'_1 \leq 0$ (by (22)) and subcase (+a) follows (but now we have no lower bound on $\|a'\|$).

Part (b). Again from (22),

$$\begin{aligned} \|b'_1\| &\leq \max\{\|ra_1\|, \|pb_1\|\} + 1 \\ &\leq \|p\| + m + 1 \\ &\leq \|a_0\| - \|a'_0\| + m + 1. \end{aligned}$$

In case $D = +1$, $b'_1 \geq 0$ and hence $b' = b'_0 2^m + b'_1 \geq 0$, as desired. Also subcase (+b) easily follows. In case $D = -1$, $b'_0 b'_1 \leq 0$ and we have the non-Archimedean inequality:

$$\|b'\| \leq \max\{\|b'_0 2^m\|, \|b'_1\|\}.$$

This proves subcase (–b).

Finally, we must show that $D = -1$ implies $\|a'\| > \|b'\|$: this follows immediately from (19), (20) and subcases (–a) and (–b). Q.E.D.

To motivate the next lemma, we now state:

Correctness Criteria: *on input $a > b \geq 0$, the HGCD algorithm outputs a regular matrix Q such that*

$$a \leq 7 \Rightarrow \quad Q = E \tag{23}$$

$$a \geq 8 \Rightarrow \|a'\| \geq 1 + \left\lceil \frac{\|a\|}{2} \right\rceil > \|b'\|, \quad a' > b' \geq 0 \tag{24}$$

where $\begin{pmatrix} a \\ b \end{pmatrix} \xrightarrow{Q} \begin{pmatrix} a' \\ b' \end{pmatrix}$.

Note that 8 is the smallest value of n_0 such that for all $a \geq n_0$, if $a > b \geq 0$ then there exists a Q satisfying (24).

Let us see how the integer analogue of the polynomial HGCD might proceed. As in the polynomial case, our first recursive call is $\mathrm{HGCD}(a_0, b_0)$, where m is now chosen to be $1 + \lceil \|a\|/2 \rceil$ and a_0, b_0 are given by (15). If the call returns Q then define a', b' as in (17) and (18). The correctness criteria (24) can be broken down into four conditions:

$$a' > 0 \tag{25}$$

$$\|a'\| \geq m \tag{26}$$

$$m > \|b'\| \tag{27}$$

$$b' \geq 0 \tag{28}$$

Assuming the parameters are properly set up, it is not hard to see that the partial correctness lemma could be applied to ensure conditions (25) and (27). Unfortunately, *depending on the sign of* $\det Q$, the lemma can only ensure one or the other of conditions (26) and (28). If $\det Q = -1$ then we do have a lower bound on $\|a'\|$, so condition (26) could be fulfilled, but we have no information on the sign of b'. If $\det Q = +1$ we fulfill condition (28) but have no lower bound on $\|a'\|$.

Hence we need a analysis deeper than in the above "partial" correctness criteria. It turns out that we only have to modify Q slightly to obtain some regular matrix Q^* such that

$$\begin{pmatrix} a \\ b \end{pmatrix} \xrightarrow{Q^*} \begin{pmatrix} a^* \\ b^* \end{pmatrix}$$

and a^*, b^* satisfy the correctness criteria, $\|a^*\| \geq m > \|b^*\|$, $a^* > b^* \geq 0$. The "fixup lemma" below shows how to do obtain this.

First we make some easily verified observations (cf. basic properties of regular continued fractions, e.g., in [10]). It is convenient to let

$$\langle q_1, \ldots, q_k \rangle,$$

where q_i's are positive integers, denote the regular matrix whose partial quotient sequence is $(q_1, \ldots, q_k)$.

- Let $Q = \langle q_1, q_2, \ldots, q_k \rangle$ be the matrix such that

$$\begin{pmatrix} a \\ b \end{pmatrix} \xrightarrow{Q} \begin{pmatrix} a' \\ b' \end{pmatrix}.$$

  Call $T = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}$ the "toggling" matrix. This is so-called because T is idempotent ($T^2 = E$) and QT is equal to $\langle q_1, \ldots, q_{k-1}, q_k - 1, 1 \rangle$ in case $q_k > 1$, and $QT = \langle q_1, \ldots, q_{k-2}, q_{k-1} + 1 \rangle$ in case $q_k = 1$ and $k > 1$. (If $q_k = 1$ and $k = 1$, QT is not a regular matrix.) In either case, we have

$$\begin{pmatrix} a \\ b \end{pmatrix} \xrightarrow{QT} \begin{pmatrix} a' + b' \\ -b' \end{pmatrix}.$$

- Below we will need to recover the last partial quotient x from a regular matrix $Q = \begin{pmatrix} p & q \\ r & s \end{pmatrix}$. Note that $Q = E$ if and only if $q = 0$, but in this case x is undefined. Hence assume $Q \neq E$. Then Q is elementary if and only if $s = 0$, and in this case $x = p$. So we next assume that Q is not elementary. Write

$$Q' = \begin{pmatrix} p' & q' \\ r' & s' \end{pmatrix}, \qquad Q = Q' \cdot \begin{pmatrix} x & 1 \\ 1 & 0 \end{pmatrix}$$

  where $Q' \neq E$ and $p = xp' + q', q = p'$. There are two cases. **Case of $q = 1$:** Clearly $p' = 1$. Since $p' \geq q' \geq 1$ (Q' is not elementary), we must have $q' = 1$. Hence x equals $p - 1$. **Case of $q > 1$:** Then $p' > q'$ (there are two possibilities to check, depending on whether Q' is elementary or not). This implies $x = p$ **div** q. In summary, the last partial quotient of Q is given by

$$x = \begin{cases} \text{undefined} & \text{if } q = 0 \\ p & \text{if } s = 0 \\ p - 1 & \text{if } q = 1 \\ p \textbf{ div } q & \text{otherwise} \end{cases}$$

Lemma 4 (Fixing Up)
Retaining the notations from the previous lemma and assuming (16)-(21), let t be any number (the "fixup threshold") such that

$$\|a'_0\| \geq t > \max\{\|b'_0\|, \|a_0\| - \|a'_0\| + 1\}. \tag{29}$$

Moreover, if we write Q as $\langle q_1, \ldots, q_k \rangle$ and Q^* is as specified below, then

$$\|a^*\| \geq m + t > \|b^*\| \tag{30}$$

and

$$b^* \geq 0, \tag{31}$$

where $\begin{pmatrix} a \\ b \end{pmatrix} \xrightarrow{Q^*} \begin{pmatrix} a^* \\ b^* \end{pmatrix}$. Here Q^* is the regular matrix specified as follows:

(–) *Suppose* $\det Q = -1$.
(–A) *If* $b' \geq 0$ *then* $Q^* := Q$.
(–B) *Else if* $\|a' + b'\| \geq m + t$ *then* $Q^* := QT$.
(–C) *Else if the last partial quotient* $q_k \geq 2$ *then* $Q^* := \langle q_1, \ldots, q_{k-1}, q_k - 1\rangle$.
(–D) *Else* $Q^* := \langle q_1, \ldots, q_{k-3}, q_{k-2}\rangle$.

(+) *Suppose* $\det Q = +1$.
(+A) *If* $\|a'\| \geq m + t$ *then there is a unique regular matrix* $\tilde{Q}$ *that is a product of at most two partial quotients such that*

$$\begin{pmatrix} a' \\ b' \end{pmatrix} \xrightarrow{\tilde{Q}} \begin{pmatrix} a^* \\ b^* \end{pmatrix}$$

and a^*, b^* *satisfy (30) and (31). Let* $Q^* := Q\tilde{Q}$.
(+B) *Else let* $Q^* := \langle q_1, \ldots, q_{k-1}\rangle$.

Proof: Observe from (19) and (20) that t exists. It is useful to note the following:

(*) Suppose that $\|a'_0\| \geq 1 + \lceil \|a_0\|/2 \rceil > \|b'_0\|$ (cf. (24)) and $t = 1 + \lceil \|a_0\|/2 \rceil$. Then both (20) and (29) are satisfied.

In the following proof, we often appeal to (*) to enforce the conditions (20) and (29) required for the application of the partial correctness lemma.

First assume $D = -1$.

Subcase (–A) In this subcase, (31) is automatic, and (30) follows from case (–) of the partial correctness lemma.

Subcase (–B) In this subcase, we are simply "toggling" and, as noted earlier, $\begin{pmatrix} a^* \\ b^* \end{pmatrix} = \begin{pmatrix} a' + b' \\ -b' \end{pmatrix}$. Recall that toggling is invalid in case $k = 1$ and $q_1 = 1$. But if this were the case then

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a' \\ b' \end{pmatrix}.$$

This implies $a' + b' = a > b = a'$ and so $b' > 0$, contradicting the fact that we had already excluded subcase (–A). Again, (31) is immediate, and (30) follows from case (–) of the partial correctness lemma (and $\|a^*\| = \|a' + b'\| \geq m + t$ by assumption).

Subcase (–C) In this subcase, Q^* can be written as $Q \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$, and so $\begin{pmatrix} a^* \\ b^* \end{pmatrix} = \begin{pmatrix} a' \\ a' + b' \end{pmatrix}$. We see that (31) holds by virtue of $a' + b' > 0$ (since $\|a'\| > \|b'\|, a' > 0$). Also (30) holds because the partial correctness lemma implies $\|a'\| \geq m + t$ and, since subcase (–B) fails, $\|a' + b'\| < m + t$.

Subcase (–D) Now $q_k = 1$ and Q^* omits the last two partial quotients $(q_{k-1}, q_k) = (x, 1)$ where we write x for q_{k-1}. We ought to show $k \geq 2$, but this is the same argument as in subcase (–B). Hence $Q^* = Q \begin{pmatrix} 1 & -x \\ -1 & x+1 \end{pmatrix}$ and $\begin{pmatrix} a^* \\ b^* \end{pmatrix} = \begin{pmatrix} a'(x+1) + b'x \\ a' + b' \end{pmatrix}$. It is not hard to check that the remainder sequence of a, b "corresponding" to Q has the form

$$(a, b, \ldots, a^*, b^*, a', b').$$

Then (31) holds because $a' + b' > 0$. To see (30), it is clear that $m + t > \|b^*\|$ and it remains to show $\|a^*\| \geq m + t$. Suppose $\begin{pmatrix} a_0 \\ b_0 \end{pmatrix} \xrightarrow{Q^*} \begin{pmatrix} a_0^* \\ b_0^* \end{pmatrix}$. Then $a_0^* = (x+1)a_0' + xb_0'$ and $b_0^* = a_0' + b_0'$, where a_0', b_0' are defined as in (17). Since $\det Q^* = -1$, $a_0^* > b_0^* \geq 0$ and $2\|a_0^*\| - 1 > \|a_0\|$, we see that case (–) of the partial correctness lemma is applicable with Q^* instead of Q, a_0^* instead of a_0', etc. This shows

$$\|a^*\| \geq m + \|a_0^*\| > m + \|a_0'\| \geq m + t.$$

Now we consider the case $D = +1$.

Subcase (+A) By assumption, $\|a'\| \geq m + t$. Also, we check that the partial correctness lemma is applicable to show $\|b'\| < 1 + m + t$. By a simple and well-known fact, at most two steps of the usual Euclidean algorithm suffices to reduce an integer remainder by half. Hence $\tilde{Q}$ is a product of at most two elementary matrices, as claimed.

Subcase (+B) So $\|a'\| < m + t$. Note that $a^* = b' + q_k a'$ and $b^* = a' > 0$. Note that $\det Q^* = -1$, and the conditions of the partial correctness lemma are satisfied using Q^* instead of Q, etc. Hence, from subcase (–a) of that lemma, we get $\|a^*\| \geq m + t$. Hence (30) and (31) holds. Q.E.D.

It is best to understand the subcases of the Fixing Up lemma as follows:

(–A) Nothing to fix

(–B) Toggle

(–C) Correct slight overshoot in last partial quotient

(–D) Backup two steps

(+A) Forward at most two steps

(+B) Backup one step

It is interesting to note that in tests on randomly generated numbers of about 45 digits, subcases (–A) and (+A) both occur in comparable frequencies and in the overwhelming number of subcases; subcases (–B) and (+B) both also occur in comparable frequences, but considerably less frequently than the (A) subcases. Subcase (–C) occurred once but subcase (–D) never arose.

This lemma (and its proof) provides us with a specific procedure to convert the tentative output matrix Q of the HGCD algorithm into a valid one. To be specific, let

$$\mathrm{FIXUP}_1(Q, a, b, m, t)$$

denote the subroutine that returns Q^*, using the notations of the Fixing Up lemma. Of course, we assume that the input parameters satisfy the conditions (16)–(20) and (29) for the Fixing Up lemma.

In applications of the Fixing Up lemma, we rely on observation (*) that $\|a_0'\| \geq 1 + \lceil \|a_0\|/2 \rceil > \|b_0'\|$ (cf. (24)) ensures (29). But as noted following (24), this generally means $a_0 \geq 8$. Hence we must fix-up Q in a different way when $a_0 \leq 7$. We use the following alternative method. If $a_0 \leq 7$, then $b \ \mathbf{div}\ 2^m = b_0 \leq a_0 - 1 \leq 6$, and hence $b < 7 \cdot 2^m$. Again exploiting the well-known fact that starting from $\begin{pmatrix} a \\ b \end{pmatrix}$, at most 6 steps of the usual Euclidean algorithm will give us a pair $\begin{pmatrix} a'' \\ b'' \end{pmatrix}$,

$$\begin{pmatrix} a \\ b \end{pmatrix} \xrightarrow{Q^*} \begin{pmatrix} a'' \\ b'' \end{pmatrix}$$

where $\|a''\| \geq m > \|b''\|$. We encode this in a simple (the obvious) subroutine

$$\mathrm{FIXUP}_0(a, b, m)$$

which returns the regular matrix Q^*. Assuming we invoke $\mathrm{FIXUP}_0(a, b, m)$ only when $a_0 = 1 + (a \ \mathbf{mod}\ 2^m) \leq 7$, Q^* is a product of at most six elementary matrices.

5 The Algorithm

Now we describe the integer HGCD algorithm, prove its correctness and analyze its complexity.

Input: integers a, b with $a > b \geq 0$.

Output: a regular matrix Q satisfying the correctness criteria (23) or (24).

function *HGCD*(a, b);

[1] $m \leftarrow 1 + \left\lceil \frac{\|a\|}{2} \right\rceil$; { this is the magical threshold }

if $\|b\| < m$ **then** *return*(E);

[2] $a_0 \leftarrow 1 + (a \ \mathbf{div}\ 2^m)$; $b_0 \leftarrow b \ \mathbf{div}\ 2^m$;

$t \leftarrow 1 + \lceil \frac{\|a_0\|}{2} \rceil$;

if $a_0 \leq 7$ **then** *return*$(\mathrm{FIXUP}_0(a, b, m))$;

else $R \leftarrow \mathrm{FIXUP}_1(HGCD(a_0, b_0), a, b, m, t)$;

$\begin{pmatrix} a' \\ b' \end{pmatrix} \leftarrow R^{-1} \begin{pmatrix} a \\ b \end{pmatrix}$;

[3] **if** $\|b'\| < m$ **then** *return*(R);

[4] $q \leftarrow a'$ **div** b'; $\begin{pmatrix} c \\ d \end{pmatrix} \leftarrow \begin{pmatrix} b' \\ a' \textbf{ mod } b' \end{pmatrix}$;

[5] $l \leftarrow \lceil \|c\| \rceil$; $k \leftarrow 2m - l - 1$; { We claim $k \geq 0$ }

[6] $c_0 \leftarrow 1 + (c \textbf{ div } 2^k)$; $d_0 \leftarrow d \textbf{ div } 2^k$;

$t' \leftarrow 1 + \left\lceil \frac{\|c_0\|}{2} \right\rceil$;

if c **div** $2^m \leq 6$ **then** *return*$(R * \begin{pmatrix} q & 1 \\ 1 & 0 \end{pmatrix} * \mathrm{FIXUP}_0(c, d, m))$;

else $S \leftarrow \mathrm{FIXUP}_1(HGCD(c_0, d_0), c, d, k, t')$; { We claim $c_0 \geq 8$ here }

[7] $\begin{pmatrix} c' \\ d' \end{pmatrix} \leftarrow S^{-1} \begin{pmatrix} c \\ d \end{pmatrix}$;

{ We claim d' **div** $2^m \leq 1$ and $\|c'\| \geq m + 1 > \|d'\|$ }

$T \leftarrow \mathrm{FIXUP}_0(c', d', m)$;

$Q \leftarrow R * \begin{pmatrix} q & 1 \\ 1 & 0 \end{pmatrix} * S * T$; *return*$(Q)$;

Correctness: HGCD returns in five places in the algorithm. We check that the matrix returned at each of these places is correct.

a) In case the algorithm returns the identity matrix E in step [1], we note that (23) or (24) holds. (As a matter of fact, if $a \leq 8$ then we would surely return at step [1].)

b) If $a_0 \leq 7$ then we would return the resulting matrix from $\mathrm{FIXUP}_0(a, b, m)$. By definition of FIXUP_0, this matrix is correct, i.e., (24) holds.

c) If $a_0 \geq 8$, the first recursive call to HGCD returns some regular matrix R' which is fixed up as R by FIXUP_1. If $\begin{pmatrix} a_0 \\ b_0 \end{pmatrix} \xrightarrow{R'} \begin{pmatrix} a'_0 \\ b'_0 \end{pmatrix}$ then, by induction hypothesis, $\|a'_0\| \geq t > \|b'_0\|, a'_0 > b'_0 \geq 0$. Using the observation (*), we see that equations (20) and (29) hold. Hence we may apply the Fixing Up lemma, with R' instead of Q: this means the matrix R returned by FIXUP_1 reduces $\begin{pmatrix} a \\ b \end{pmatrix}$ to $\begin{pmatrix} a' \\ b' \end{pmatrix}$ such that $\|a'\| \geq m + t > \|b'\|$. If the test $\|b'\| < m$ succeeds in step [3], then the matrix R returned by HGCD is clearly correct, i.e., (24) holds.

d) Now suppose we reach step [6]. First we show that $k \geq 0$, i.e., $2m - 1 \geq l$.

$$\begin{aligned} \|c\| = \|b'\| &< m + t = m + 1 + \left\lceil \frac{\|1 + (a \textbf{ div } 2^m)\|}{2} \right\rceil \\ &\leq m + 1 + \left\lceil \frac{1 + \lfloor \|a \textbf{ div } 2^m\| \rfloor}{2} \right\rceil \quad (\text{since } \|1 + x\| \leq 1 + \lfloor \|x\| \rfloor) \end{aligned}$$

$$
\begin{aligned}
&\leq m+1+\left\lceil \frac{1+\lfloor \|a/2^m\| \rfloor}{2} \right\rceil \quad (\text{since } \lfloor \|x \textbf{ div } y\| \rfloor = \lfloor \|x/y\| \rfloor) \\
&< m+1+\left\lceil \frac{1+\lfloor \|a\| \rfloor - m}{2} \right\rceil \\
&\leq (1+\lceil \|a\|/2 \rceil)+\lceil (m+1)/2 \rceil \\
&= m+\lceil (m+1)/2 \rceil \leq 2m-1,
\end{aligned}
$$

since $m \geq 3$ (using the fact $a \geq 8$). Hence $l = \lceil \|c\| \rceil \leq 2m-1$, as desired. If HGCD returns the matrix returned by FIXUP_0 in this step, the result is clearly correct.

e) Otherwise, the matrix Q formed in step [7] is returned. Suppose Q reduces $\begin{pmatrix} a \\ b \end{pmatrix}$ to $\begin{pmatrix} a' \\ b' \end{pmatrix}$. So we want to show

$$
\|a'\| \geq m > \|b'\|, \quad a' > b' \geq 0. \tag{32}
$$

First we show the claim $c_0 \geq 8$, stated at the end of step [6]. This would imply that the recursive call to $\text{HGCD}(c_0, d_0)$ returns some matrix Q' that satisfies the analogue of (24). We know that $c \textbf{ div } 2^m \geq 7$ (from the test in step [6]). Hence $c \geq 7 \cdot 2^m$ and $l = \lceil \|c\| \rceil \geq 3+m$ or $l-m \geq 3$.

$$
\begin{aligned}
\|c_0 \textbf{ div } 2^{t'}\| &> \|c_0/2^{t'}\| - 1 = \|c_0\| - t' - 1 \\
&= -2 + \lfloor \|c_0\|/2 \rfloor \quad (\text{by definition of } t') \\
&\geq -2 + \lfloor \|c \textbf{ div } 2^k\|/2 \rfloor \\
&\geq -2 + \left\lfloor \frac{\lceil \|c/2^k\| \rceil - 1}{2} \right\rfloor \quad (\text{since } \|x \textbf{ div } y\| \geq \lceil \|x/y\| \rceil - 1) \\
&= -2 + (l-m) \geq 1 \quad (\ l-k = 2l-2m+1)
\end{aligned}
$$

Hence $\|c_0 \textbf{ div } 2^{t'}\| > 1$ and $c_0 \textbf{ div } 2^{t'} \geq 3$. Then $c \textbf{ div } 2^m \geq 7$ implies $c \textbf{ div } 2^k \geq 7$ $(k \leq m)$ and $t' \geq 3$. Thus $c_0 \geq (c_0 \textbf{ div } 2^{t'}) \cdot 2^{t'} \geq 24$, which is considerably more than we need.

The conditions of the Fixing Up lemma hold for the call to FIXUP_1 in step [6]; again we use observation (*), with c's and d's instead of a's and b's, k instead of m, t' instead of t, Q' instead of Q. This means the matrix S returned by FIXUP_1 reduces $\begin{pmatrix} c \\ d \end{pmatrix}$ to $\begin{pmatrix} c' \\ d' \end{pmatrix}$ such that $\|c'\| \geq k + t' > \|d'\|$. We claim

$$
k + t' = m + 1.
$$

We see that $l - k = 2l - 2m + 1$ and

$$
t' \quad = \quad 1 + \lceil \|c_0\|/2 \rceil
$$

$$
\begin{aligned}
&= 1 + \left\lceil \frac{\lceil \|c_0\| \rceil}{2} \right\rceil \\
&= 1 + \left\lceil \frac{\lceil \|\epsilon + (c/2^k)\| \rceil}{2} \right\rceil \quad (0 < \epsilon \leq 1) \\
&= 1 + \left\lceil \frac{l - k + \delta}{2} \right\rceil \quad (\delta = 0 \text{ or } 1) \\
&= 1 + \left\lceil \frac{2l - 2m + 1 + \delta}{2} \right\rceil \\
&= 2 + l - m.
\end{aligned}
$$

Thus $k + t' = m + 1$, as desired. Hence $d' < 2^{m+1}$, d' **div** $2^m < 2$. We have thus justified the comments in step [7]. The matrix T returned by FIXUP_0 will clearly reduce $\begin{pmatrix} c' \\ d' \end{pmatrix}$ to our desired $\begin{pmatrix} a' \\ b' \end{pmatrix}$ in (32). This concludes our correctness proof. Q.E.D.

Complexity Analysis: The algorithm has to perform comparisons of the kind

$$\|a\| : m,$$

and compute ceiling functions in the special forms

$$\lceil \|a\| \rceil, \qquad \lceil \|a\|/2 \rceil,$$

where a, m are positive. (In the algorithm a may be zero, but for simplicity here, we treat those as special cases.) Since $\|a\|$ is generally not rational, we do not want to explicitly compute it. Instead we can reduce the above operations to checking if a is a power of two, and to computing the function $\langle a \rangle$, which is defined to be the number of bits in the binary representation of a positive integer a. So $\langle a \rangle = 1 + \lfloor \log_2 a \rfloor$ and clearly this function is easily computed in linear time. Then we have

$$\|a\| \geq m \Leftrightarrow \langle a \rangle - 1 \geq m$$

and

$$\|a\| > m \Leftrightarrow \begin{cases} \langle a \rangle - 1 > m & \text{if } a \text{ is a power of 2} \\ \langle a \rangle > m & \text{else.} \end{cases}$$

and finally,

$$\left\lceil \frac{\|a\|}{2} \right\rceil = \begin{cases} \left\lceil \frac{\langle a \rangle - 1}{2} \right\rceil & \text{if } a \text{ is a power of 2} \\ \left\lceil \frac{\langle a \rangle}{2} \right\rceil & \text{else} \end{cases}$$

The global structure of the complexity analysis is standard [1]: let $M(n)$ denote the time complexity of multiplying two n-bit integers, $H(n)$ the time

complexity of our HGCD algorithm where both inputs have at most n bits. It is known that $M(n) = O(n \log n \log\log n)$. It is not hard to see that FIXUP_0 and FIXUP_1 takes time $O(M(n))$. Hence

$$H(n) = O(M(n)) + 2H(n/2 + O(1)).$$

Using the fact that $n = O(M(n))$ and $M(n+O(1)) = M(n)+O(n)$, we conclude that $H(n) = O(M(n)\log n) = O(n \log^2 n \log\log n)$.

The HGCD algorithm as presented is not locally optimized, since we prefer whenever possible to make the correctness proof more transparent. Both the integer and polynomial HGCD algorithms have been implemented and tested using *Mathematica*.

6 Summary

We have developed an integer analogue of the polynomial HGCD algorithm. Its proof of correctness is modeled after corresponding properties used to show the correctness of the polynomial HGCD. Although the general outline of the integer case is not hard to understand because of its analogy to the polynomial case, it is still rather delicate and detailed. Further simplification of our integer HGCD algorithm should be possible.

Acknowledgement: Chee Yap would like to thank Lars Ericson for helping to implement and test the polynomial and integer HGCD algorithms in the computer algebra system *Mathematica*.

APPENDIX

We construct an example to show where the polynomial HGCD algorithm of [1] (henceforth referred to as the 'Old HGCD') breaks down. Roughly speaking, the output criterion for HGCD algorithms has two components: (i) the output matrix gives rise to a pair of consecutive members of the remainder sequence, and (ii) the degrees of these two members must "straddle" half the original maximum degree. We will construct input polynomials that violates (ii) but not (i). This means that the complexity analysis of Old HGCD is invalid. We have not yet found any example that violates (i). Indeed, we do not even know if Old HGCD algorithm terminates on all input polynomials. We will say an output matrix of a HGCD algorithm is *pseudo-correct* if it satisfies (i).

It is worth remarking that finding a pair of bad input polynomials to the Old HGCD algorithm requires some thought. This is because it is known that Old HGCD *is correct* for inputs with so-called *regular* remainder sequences, that is, where the degrees of consecutive members of the remainder sequence differ by one (except for the initial pair). This was in the original proof of Moenck

[9]. Furthermore, randomly chosen inputs are highly likely to be regular (since irregular remainder sequences depend on vanishing of certain subdeterminants).

The Old HGCD algorithm from [1] is as follows. Input are polynomials a, b with $\deg(a) > \deg(b) \geq 0$. Output is supposed to be an inverse regular matrix T with

$$\begin{pmatrix} g \\ h \end{pmatrix} = T \begin{pmatrix} a \\ b \end{pmatrix}$$

such that

$$\deg(g) > \left\lfloor \frac{\deg(a)}{2} \right\rfloor \geq \deg(h).$$

(Matrix T is roughly the inverse of our R.)

function *OldHGCD*(a, b);

[1] $m \leftarrow \left\lfloor \frac{\deg(a)}{2} \right\rfloor$;
if $\deg(b) \leq m$ **then return**(E);

[2] $a_1 \leftarrow a$ **div** x^m; $b_1 \leftarrow b$ **div** x^m; $R \leftarrow$ *OldHGCD*(a_1, b_1); $\begin{pmatrix} d \\ e \end{pmatrix} \leftarrow R \begin{pmatrix} a \\ b \end{pmatrix}$;

[3] { at this point, nothing prevents e to be the 0-polynomial }

[4] $f \leftarrow d$ **mod** e; $q \leftarrow d$ **div** e;

[5] $k \leftarrow \lfloor \frac{m}{2} \rfloor$;

[6] $e_1 \leftarrow e$ **div** x^k; $f_1 \leftarrow f$ **div** x^k; $S \leftarrow$ *OldHGCD*(e_1, f_1);

[7] **return**$(S * \begin{pmatrix} 0 & 1 \\ 1 & -q \end{pmatrix} * R)$;

Note that the matrix R in our algorithm is the inverse of theirs, but this is an unessential point.

First of all, we note that at the (null) step [3], the polynomial e may be zero, and the next step would give us a divide-by-zero error. But this is easily fixed: we should check here if $\deg(e) < m$, and if so, immediately return the matrix R. The non-trivial error is that in step [5], the choice of k as m **div** 2 is in general incorrect. Accordingly we cook up a pair of polynomials to illustrate this. Consider the pair of polynomials

$$f_0 = x^{20} + x^{17} + x^{15} + x^{12} + x^9$$

and

$$g_0 = x^{16} + x^{13} + x^{11}.$$

Their remainder sequence and corresponding partial quotients are

$$x^{12} + x^9, \quad x^{11}, \quad x^9 \tag{33}$$

$$x^4, \quad x^4, \quad x, \quad x^2. \tag{34}$$

The two remainders conforming to Old HGCD's output specifications are the two of degree 11 and 9 (i.e., $11 > \lfloor 20/2 \rfloor \geq 9$) for which the transformation

matrix is

$$\begin{pmatrix} -x^4 & x^8+1 \\ x^5+1 & -x^9-x^4-x \end{pmatrix}. \tag{35}$$

(For comparability, this is the inverse of the matrix returned by our HGCD.) Old HGCD, however, returns the matrix

$$\begin{pmatrix} x^5+1 & -x^9-x^4-x \\ -x^7-x^4-x^2 & x^{11}+x^8+x^6+x^3+1 \end{pmatrix} \tag{36}$$

which yields the two remainders of degree 9 and $-\infty$.

A detailed trace of Old HGCD's operations discloses why this is so. The first subcall of Old HGCD returns (correctly) the remainders of degrees 16 and 12, and a subsequent mid step division yields the next remainder of degree 11. Now k is (incorrectly) calculated as 5 and so the arguments to the second subcall of Old HGCD are the truncated remainders of degrees 6 and 5, whence (again correctly) the remainders of degrees 4 and $-\infty$ are computed which are post-processed to be the remainders of degree 9 and $-\infty$.

Nevertheless, the matrix is pseudo-correct. We extend this example to show even stranger phenomena. Let

$$f = x^{40} + x^{37} + x^{35} + x^{32} + x^{29}$$

and

$$g = x^{36} + x^{33} + x^{31} \underline{+ x^{19}}.$$

So the original pair f_0, g_0 is equal to f **div** x^{20}, g **div** x^{20}. Here and below, we have underlined the "added terms" which comes from f **mod** x^{20} and g **mod** x^{20}. The remainder sequence of f, g begins with

$$x^{32} + x^{29}\underline{-x^{23}}, \quad x^{31}\underline{+x^{27}+x^{19}}, \quad x^{29}\underline{-x^{28}-x^{23}-x^{20}},$$

$$\underline{x^{28}+x^{27}+x^{25}+x^{24}+x^{23}+x^{22}+x^{21}+x^{20}+x^{19}}$$

(cf. (33)) and corresponding partial quotients are

$$x^4, \quad x^4, \quad x, \quad x^2\underline{+x+1}$$

(cf. (34)).

The correct matrix (35) applied to the vector $\begin{pmatrix} f \\ g \end{pmatrix}$ yields the consecutive remainders

$$x^{31}\underline{+x^{27}+x^{19}}, \quad x^{29}\underline{-x^{28}-x^{23}-x^{20}}, \tag{37}$$

while the incorrect matrix (36) computes the pair

$$x^{29}\underline{-x^{28}-x^{23}-x^{20}}, \quad \underline{x^{30}+x^{27}+x^{25}+x^{22}+x^{19}}. \tag{38}$$

Note that the degrees in (38) are in inverted order: this behaviour is not unexpected from our analysis! When compared to the degrees in (37), it is clear that (38) cannot belong to the same strict remainder sequence as (37). This error occurs within the computation of Old HGCD(f, g), and the surprise is that Old HGCD nevertheless returns a matrix that is pseudo-correct: the matrix returned by Old HGCD computes the pair

$$4(-x^{24} + x^{20} + x^{19}), \quad (2x^{23} + x^{21} - x^{19})/4$$

while the correct HGCD computes

$$32x^{20} + 12x^{19}, \quad 23x^{19}/256.$$

Both are still consecutive members of the remainder sequence! Another surprise is that the degrees of the remainders returned by the Old HGCD turned out to be more than the correct degrees (initially they were less). The interested reader may further compare the behaviour of the HGCD's on other variations of f, g (say, $f + x^{19}, g - x^{19}$, and $x^{40}f + x^{39}, x^{40}g$).

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley, 1974.

[2] R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun. Fast solution of Toeplitz systems of equations and computation of Pade approximants. *J. Algorithms*, 1:259–295, 1980.

[3] B. Buchberger, G. E. Collins, and R. Loos (eds.). *Computer Algebra*. Springer-Verlag, 1983.

[4] E. Kaltofen. Greatest common divisors of polynomials given by straight-line programs. *Journal of the ACM*, 35:231–264, 1988.

[5] E. Kaltofen and H. Rolletschek. Computing greatest common divisors and factorizations in quadratic number fields. *Math. Comp.*, 52:697–720, 1989.

[6] D. E. Knuth. The analysis of algorithms. In *Actes du Congrés International des Mathématiciens*, pages 269–274, Nice, France, 1970. Gauthier-Villars.

[7] D. E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms*, volume 2. Addison-Wesley, 2nd edition edition, 1981.

[8] D.H. Lehmer. Titlexxx. *American Mathematical Monthly*, 45:227–233, 1937.

[9] R. Moenck. Fast computations of GCD's. *Proc. 5th ACM Symposium on Theory of Computation*, pages 142–171, 1973.

[10] Oskar Perron. *Die Lehre von den Kettenbrüchen.* Teubner, Leipzig, 2nd edition edition, 1929.

[11] Arnold Schönhage. Schnelle Berechnung von Kettenbruchentwicklungen. *Acta Informatica*, 1:139–144, 1971.

[12] Arnold Schönhage. Probabilistic computation of integer polynomial gcd's. *J. Algorithms*, 9:365–371, 1988.

[13] Volker Strassen. The computational complexity of continued fractions. *SIAM Journal Computing*, 12:1–27, 1983.

[14] David Y. Yun. Uniform bounds for a class of algebraic mappings. *SIAM Journal Computing*, 12:348–356, 1983.