Journal of International Mathematical Virtual Institute
ISSN 1840-4359
Vol. 1(2011), 1-16

Former
Bulletin of Society of Mathematicians Banja Luka
ISSN 0354-5792 (p), ISSN 1986-521X (o)
Vol. 18 (2011), 31-46

Variants of an algorithm of J. Stein

Sándor Szabó[1]

Abstract

In 1961 J. Stein proposed an algorithm to compute the greatest common divisor of two integers. In this paper we point out that similar algorithms exist in the ring of integers of various quadratic number fields and also in the non-commutative ring of the Hurwitz quaternions. The implementations of the algorithms are straightforward. However the procedures vary from case to case.

AMS Mathematics Subject Classification (2010): Primary 11A51, Secondary 11Y16

Key words and phrases: gcd in quadratic number fields, Hurwitz quaternion, binary gcd algorithm and its extensions.

1 Introduction

In his book [9] T. W. Körner tells us: "Euclid's algorithm has been known for over 2000 and the reason for its excellence has been well understood for over a 100 years. It has come as a surprise to me to learn that it now has a genuine competitor, invented by J. Stein in 1961." He then goes on describing Stein's algorithm which based on an odd even consideration. For some history and for an analysis of Stein's algorithm see [8]. The reader might find useful background information in [1] and [3]. Both Euclid's and Stein's algorithms imply that the integers have the unique factorization property. It is remarkable that the unique factorization property can be established by an odd even argument.

Only a small minority of the rings of algebraic integers possesses the unique factorization property. Namely those whose ideal class number is one. In some of these rings a Euclidean algorithm is available to compute greatest common divisors. The Euclidean algorithm is used extensively in cryptographic computations. Further the Euclidean algorithm in $Q[i]$ and $Q[\sqrt{-3}]$ is used for

[1]Institute of Mathematics and Informatics, University of Pécs, Ifjúság u. 6, 7624 Pécs, HUNGARY, e-mail: sszabo7@hotmail.com

deciding the biquadratic and cubic character of an integer respectively modulo a prime.

In the binary gcd algorithm (Stein's original algorithm) the divisions can be replaced by shifts, an advantage especially for large numbers. D. E. Knuth [8] reports on numerical experiments comparing the Euclidean and the binary algorithms. A. Weilert [13] is devoted entirely to the complexity analysis of the $(1+i)$-ary algorithm in Gaussian integers. On the other hand papers [2], [6], [7], [12], and [10] deal with the complexity or implementation of the Euclidean algorithm in quadratic fields. Namely, [7] seems to present an analogue of the Lame bound on the number of the division steps while Theorem 4.1 of [6] looks like an $O(t^2)$ bit complexity bound for Euclid's algorithm. The algorithms we present can also be completed using at most $O(t^2)$ bit operations, where t is the total number of input bits.

In this paper we will show that Stein's algorithm works in a more general setting. Let R be a commutative ring with identity element 1 and zero element 0. We will use a function H from R to the non-negative integers. We list four properties related to this H.

(a) $H(ab) = H(a)H(b)$ for each $a, b \in R$,

(b) $H(a) = 1$ if and only if a is a unit in R,

(c) $H(a) = 0$ if and only if $a = 0$,

(d) $H(a) \geq H(b)$ implies $4H(a) \leq H(a+b)$ and equation holds only when $a = b$,

(e) $H(a+b) + H(a-b) = 2[H(a) + H(b)]$ for each $a, b \in R$.

We would like to point out that we are not looking for a function H that satisfies all these properties. It will depend on the particular situation which properties must be satisfied. The point is that no other property outside of this list will be considered.

To each a in R we assign a type. The number of the types must be finite and the type should be computable. The generalized form of Stein's algorithm to compute a greatest common divisor of a and b in R is the following.

Step 1. Set $a_1 = a$, $b_1 = b$, $d_1 = 1$ as initial values.

Step 2. If a_k, b_k, d_k have already been computed, then distinguish 4 cases.

(1) If $a_k = \pm b_k$, then a_k and b_k are associates in R. A greatest common divisor of a_k and b_k is a_k. Set $a_{k+1} = 1$, $b_{k+1} = 1$, $d_{k+1} = d_k a_k$ and the algorithm terminates.

(2) If b_k is a unit in R, then a greatest common divisor of a_k and b_k is 1. Set $a_{k+1} = 1$, $b_{k+1} = 1$, $d_{k+1} = d_k$ and the algorithm terminates.

(3) If $b_k = 0$, then a greatest common divisor of a_k and b_k is a_k. Set $a_{k+1} = 1$, $b_{k+1} = 1$, $d_{k+1} = d_k a_k$ and the algorithm terminates.

(4) If none of the cases (1), (2), (3) holds, then compute the types of a_k and b_k. Let us consider cases depending on the types of a_k and b_k. We list in a table how to compute a_{k+1}, b_{k+1}, d_{k+1} in terms of a_k, b_k, and d_k. The rows are labeled by the possible types of a_k and the columns are labeled by the possible types of b_k. Then go back to the beginning of Step 2.

We would like to add one remark here. In Step 2 in some version of the algorithm it is assumed that $H(a_k) \geq H(b_k)$. If $H(a_k) < H(b_k)$, then we swap a_k and b_k before we continue. However, evaluating the function H can be computationally expensive. When this happens we try to avoid computing $H(a_k)$ and $H(b_k)$.

Let us turn to the analysis of the algorithm. We assume that a_{k+1}, b_{k+1}, d_{k+1} are computed from a_k, b_k, d_k in such a way that

(i) $d \mid a_k d_k$ and $d \mid b_k d_k$ if and only if $d \mid a_{k+1} d_{k+1}$ and $d \mid b_{k+1} d_{k+1}$.

We call the quantity $h_k = H(a_k) + H(b_k)$ the height of the pair (a_k, b_k). In a concrete setting it remains a task to show that the algorithm terminates because the height eventually decreases.

Suppose that the algorithm terminates in n steps. We claim that d_n is a greatest common divisor of a and b. We will show that $d_n \mid a$ and $d_n \mid b$. Then we show that $d \mid a$ and $d \mid b$ imply $d \mid d_n$. In order to verify this claim consider the following list.

$$\begin{array}{ll} a_1 d_1, & b_1 d_1, \\ a_2 d_2, & b_2 d_2, \\ \vdots & \vdots \\ a_n d_n, & b_n d_n. \end{array}$$

Going up on the list step by step we can see that $d_n \mid a_1 d_1$ and $d_n \mid b_1 d_1$. Since $d_1 = a_n = b_n = 1$, $a = a_1$, $b = b_1$ we get that $d_n \mid a$ and $d_n \mid b$. Choose a d that divides both a and b. Going down on the list we get that $d \mid a_n d_n$ and $d \mid b_n d_n$. This means $d \mid d_n$.

In a concrete setting it remains a task to verify that (i) holds.

2 The integers

Let R be the set of the non-negative integers with the usual addition and multiplication. In this case R is not a ring but this will cause no problem. Let the function H be defined by $H(a) = a$ for each $a \in R$. Properties (d), (e) do not hold. But we are not going to use them. We say that the type of an integer a is u if $a \equiv u \pmod 2$ and $0 \leq u \leq 1$. Table 1 summarizes how to compute a_{k+1}, b_{k+1}, d_{k+1}. We marked the cell by letters D and S as "division" or "subtraction" cells. The indices are for easier reference. This is Stein's original algorithm.

In order to see how the algorithm works let us consider an example. Let $a = 28$, $b = 33$. (See Table 2.) A word of warning. By Step 2 of the algorithm we

Table 1: The integers with 2 types

	0	1
0	$D_{1,1}$ $a_{k+1} = \frac{a_k}{2}$ $b_{k+1} = \frac{b_k}{2}$ $d_{k+1} = 2d_k$	$D_{1,2}$ $a_{k+1} = \frac{a_k}{2}$ $b_{k+1} = b_k$ $d_{k+1} = d_k$
1	$D_{2,1}$ $a_{k+1} = a_k$ $b_{k+1} = \frac{b_k}{2}$ $d_{k+1} = d_k$	$S_{2,2}$ $a_{k+1} = \frac{a_k - b_k}{2}$ $b_{k+1} = b_k$ $d_{k+1} = d_k$

Table 2: An example in Z

k	a_k	$H(a_k)$	b_k	$H(b_k)$	d_k	h_k
1	33	33	28	28	1	61
2	33	33	14	14	1	47
3	33	33	7	7	1	40
4	13	13	7	7	1	20
5	7	7	3	3	1	10
6	3	3	2	2	1	5
7	1	1	1	1	1	2

Table 3: An algorithm without subtraction

	0	1
0	$a_{k+1} = \frac{a_k}{2}$ $b_{k+1} = \frac{b_k}{2}$ $d_{k+1} = 2d_k$	$a_{k+1} = \frac{a_k}{2}$ $b_{k+1} = b_k$ $d_{k+1} = d_k$
1	$a_{k+1} = a_k$ $b_{k+1} = \frac{b_k}{2}$ $d_{k+1} = d_k$	$a_{k+1} = \frac{a_k + b_k}{2}$ $b_{k+1} = b_k$ $d_{k+1} = d_k$

Table 4: An algorithm without comparison

	0	1
0	$a_{k+1} = \frac{a_k}{2}$ $b_{k+1} = \frac{b_k}{2}$ $d_{k+1} = 2d_k$	$a_{k+1} = \frac{a_k}{2}$ $b_{k+1} = b_k$ $d_{k+1} = d_k$
1	$a_{k+1} = a_k$ $b_{k+1} = \frac{b_k}{2}$ $d_{k+1} = d_k$	$a_{k+1} = \frac{a_k - b_k}{2}$ $b_{k+1} = \frac{a_k + b_k}{2}$ $d_{k+1} = d_k$

Table 5: The integers with 3 types

	0	1	2
0	$a_{k+1} = \frac{a_k}{3}$ $b_{k+1} = \frac{b_k}{3}$ $d_{k+1} = 3d_k$	$a_{k+1} = \frac{a_k}{3}$ $b_{k+1} = b_k$ $d_{k+1} = d_k$	$a_{k+1} = \frac{a_k}{3}$ $b_{k+1} = b_k$ $d_{k+1} = d_k$
1	$a_{k+1} = a_k$ $b_{k+1} = \frac{b_k}{3}$ $d_{k+1} = d_k$	$a_{k+1} = \frac{a_k - b_k}{3}$ $b_{k+1} = b_k$ $d_{k+1} = d_k$	$a_{k+1} = a_k$ $b_{k+1} = -b_k$ $d_{k+1} = d_k$
2	$a_{k+1} = a_k$ $b_{k+1} = \frac{b_k}{3}$ $d_{k+1} = d_k$	$a_{k+1} = -a_k$ $b_{k+1} = b_k$ $d_{k+1} = d_k$	$a_{k+1} = \frac{a_k - b_k}{3}$ $b_{k+1} = b_k$ $d_{k+1} = d_k$

have to make sure that the inequality $H(a_k) \geq H(b_k)$ holds. So after computing a_k, b_k we compare $H(a_k)$ and $H(b_k)$ and swap a_k and b_k if necessary.

Proposition 1 *The algorithm terminates.*

Proof. If one of $a_k = b_k$, $a_k = 1$, $b_k = 1$ holds, then the algorithm terminates at the next step. We may assume that none of these cases holds and so we are in a cell of Table 1. In cells $D_{1,1}$ and $S_{2,2}$ $h_{k+1} = h_k/2$. In cell $D_{1,2}$ $h_{k+1} = h_k - a_k/2$ and in cell $D_{2,1}$ $h_{k+1} = h_k - b_k/2$. As $a_k > 0$, $b_k > 0$ it follows that $h_{k+1} < h_k$ in each case and the computations cannot go forever. This completes the proof. □

Consider the following divisibility relations.

$$d \mid a_k d_k \quad \text{and} \quad d \mid b_k d_k, \tag{1}$$

$$d \mid a_{k+1} d_{k+1} \quad \text{and} \quad d \mid b_{k+1} d_{k+1}. \tag{2}$$

Proposition 2 *(1) is equivalent to (2).*

Proof. First assume (1) and try to show (2). In the $a_k = b_k$ or $b_k = 0$ case notice that $a_k d_k = a_{k+1} d_{k+1}$ and $b_k d_k = b_{k+1} d_{k+1}$ which plainly means that (1) is equivalent to (2). In the $b_k = 1$ case from $d \mid b_k d_k$ it follows that $d \mid d_k$. Then notice that $a_{k+1} d_{k+1} = d_k$ and $b_{k+1} d_{k+1} = d_k$ and so (2) holds. We may assume that none of $a_k = b_k$, $b_k = 0$, $b_k = 1$ holds and consequently we are in a cell of Table 1.

In cell $D_{1,1}$, $a_k d_k = a_{k+1} d_{k+1}$ and $b_k d_k = b_{k+1} d_{k+1}$ and so nothing left to prove. In cell $D_{1,2}$, $b_k d_k = b_{k+1} d_{k+1}$ holds hence we left with showing that $d \mid a_{k+1} d_{k+1}$. $d \mid a_k d_k = 2a_{k+1} d_{k+1}$ can be written as $dc = 2a_{k+1} d_{k+1}$. If $2 \mid c$, then dividing by 2 we get $d \mid a_{k+1} d_{k+1}$. We show that $2 \mid c$. Suppose that 2 does not divide c and consider

$$dc = 2a_{k+1} d_k. \tag{3}$$

Note that d_k is a power of 2. Indeed, at the beginning $d_1 = 1$. Then d_{k+1} is d_k or $2d_k$ unless one of $a_k = b_k$, $b_k = 0$, $b_k = 1$ holds. But in these cases the algorithm terminates at the next step. Thus at our stage of the computations d_k is a power of 2. From (3) it follows that $2 \mid d$. Simplifying by 2 and repeating the argument we get that $2d_k \mid d$. From $d \mid b_k d_k$ we get that $2d_k \mid b_k d_k$ and so $2 \mid b_k$. This is a contradiction. Therefore $2 \mid c$ as we needed. The case of cell $D_{2,1}$ is similar.

In cell $S_{2,2}$, $b_{k+1} d_{k+1} = b_k d_k$ holds so we need to show only $d \mid a_{k+1} d_{k+1}$. From

$$d \mid a_k b_k = (2a_{k+1} + b_k) d_k = 2a_{k+1} d_k = b_k d_k$$

it follows that $d \mid 2a_{k+1} d_k$ and we can follow the line of arguing we have seen in connection with cell $D_{1,2}$.

Next assume (2) and try to show (1). We consider the same cases as in the first part of the proof. In cells $D_{1,1}$, $D_{1,2}$, $D_{2,1}$ $a_{k+1} d_{k+1} \mid a_k d_k$ and $b_{k+1} d_{k+1} \mid$

$b_k d_k$ and there nothing left to prove. In cell $S_{2,2}$, $b_{k+1}d_{k+1} = b_k d_k$ gives that $d \mid b_k d_k$ and

$$b_k d_k = (a_k - 2a_{k+1})d_k = a_k d_k - 2a_{k+1}d_{k+1}$$

implies $d \mid a_k b_k$. This completes the proof. □

If the algorithm terminates in n steps, then by (i) and by Proposition 2, d_n is the greatest common divisor of a and b among the non-negative integers.

In the remaining part of this section let R be the ring of integers. We describe variants of Stein's algorithm. Let the function H be defined by $H(a) = a^2$, $a \in R$. To an integer a we assign the type u by $a \equiv v \pmod 2$, $0 \leq u \leq 1$ and let a_{k+1}, b_{k+1}, d_{k+1} be computed by the instructions of Table 3. We call the squares in the first row and first column "division" squares and we call the remaining square the "addition" square. If we are in a "division" square, then clearly the height decreases, that is, $h_{k+1} < h_k$. If we are in the "addition" square, then the height decreases again.

$$\begin{aligned} h_{k+1} &= H((a_k + b_k)/2) + H(b_k) \\ &= \frac{1}{4}H(a_k + b_k) + H(b_k) \\ &\leq \left(\frac{1}{4}\right)(4)H(a_k) + H(b_k) \\ &= h_k \end{aligned}$$

Equation holds only when $a_k = b_k$. But in this case the algorithm terminates in the kth step. Interesting to notice that in this variant of the algorithm no subtraction occurs.

Let us consider the algorithm defined by Table 4. In this table there are "division" squares and a "subtraction-addition" square. In a "division" square the height clearly decreases. If we are in a "subtraction-addition" square, then the height decreases again.

$$\begin{aligned} h_{k+1} &= H((a_k - b_k)/2) + H((a_k + b_k)/2) \\ &= \left(\frac{1}{4}\right)[H(a_k - b_k) + H(a_k + b_k)] \\ &= \left(\frac{1}{4}\right)(2)[H(a_k) + H(b_k)] \\ &= \frac{1}{2}h_k \end{aligned}$$

One should notice that in this variant of the algorithm there is no comparison involved.

Since in Table 4 a new type of cell a "subtraction-addition" cell has appeared we check that (i) holds for this type of cells too. Assume (1) and try to prove (2). From $a_k - b_k = 2a_{k+1}$ we get $(a_k - b_k)d_k = 2a_{k+1}d_k$ and then $d \mid 2a_{k+1}d_k$. In the way we have seen in the proof of Proposition 2 it follows that $d \mid a_{k+1}d_{k+1}$. Starting with $a_k + b_k = 2b_{k+1}$ we get $d \mid b_{k+1}d_{k+1}$.

Next assume (2) and try to show (1). Now $b_{k+1} + a_{k+1} = a_k$ and so $b_{k+1}d_{k+1} + a_{k+1}d_{k+1} = a_k d_{k+1} = a_k d_k$ show that $d \mid a_k d_k$. Starting with

Table 6: Polynomials

	0	1
0	$f_{k+1} = \frac{f_k}{x}$ $g_{k+1} = \frac{g_k}{x}$ $d_{k+1} = d_k x$	$f_{k+1} = \frac{f_k}{x}$ $g_{k+1} = g_k$ $d_{k+1} = d_k$
1	$f_{k+1} = f_k$ $g_{k+1} = \frac{g_k}{x}$ $d_{k+1} = d_k$	$f_{k+1} = \frac{f_k + \lambda g_k}{x}$ $g_{k+1} = g_k$ $d_{k+1} = d_k$

$b_{k+1} - a_{k+1} = b_k$ we get $d \mid b_k d_k$. This shows that (i) holds for this type of cell too.

Next assign the type u to the element a of R by $a \equiv u \pmod 3$, $0 \leq u \leq 2$. The remaining details of the algorithm are in Table 5. The squares in the first row and first column will be called "division" squares. The second and third entries of the main diagonal are called "subtraction" squares and the remaining squares are termed as "reflection" squares. The height plainly decreases in a "division" square. If we are in a "subtraction" square, then again the height decreases. If we are in a "reflection" square, then the height does not change and in the next step we cannot end up in a "reflection" square. This means that in two steps the height decreases. One should notice that in this algorithm there is no comparison involved.

Finally, there is an obvious variant where we work modulo 4. We divide when either element is 2 modulo 4, subtract when both are 1 or 3 modulo 4, and reflect when we have odd disagreeing elements.

Proposition 3 *The Stein type algorithms can be performed with $O(t^2)$ bit operations, where t is the total number of bits of the two integers whose gcd is computed.*

Proof. Let t_k be the total number of bits of a_k and b_k. When a_k is even, then we set $a_{k+1} = a_k/2$. In other words we delete the last bit of a_k to get a_{k+1} and so in a "division" cell t_k decreases by 1. In an "addition" cell t_k may remain unchanged. In a "subtraction" cell t_k decreases by 1. From an "addition" or "subtraction" cell we always move to a "division" cell. In short the value of t_k must decrease by at least 1 in every other steps. Therefore the computation terminates in at most $2t$ steps.

Next note that each step can be carried out using at most $4t_k$ bit operations. Indeed, we compare a_k and b_k and swap them if $a_k < b_k$. The comparison can be done with $2t_k$ bit operations and also the swapping can be accomplished with $2t_k$ bit operations.

By checking the last bit we can decide if a number is odd or even. Computing $a_k \pm b_k$ and done with at most $3t_k$ bit operations. Further dividing and multiplying by 2 can be done with at most $2t_k$ bit operation via shifting. □

3 Polynomials over a field

Let R be the ring of polynomials over a field F. Let $f \in R$. We define H by $H(f) = 2^{\deg(f)}$. The degree of the zero polynomial is defined to be $-\infty$ and the degree of a nonzero constant polynomial is 0. One can check that $H(fg) = H(f)H(g)$ holds. Further $H(f) = 1$ if and only if f is a unit of R and $H(f) = 0$ if and only if f is the zero polynomial. Properties (d), (e) do not hold for H but we do not need them. We say that f is of type 0, 1 depending on the constant term of f is zero or nonzero. Stein's algorithm works in R very much the same way as the Euclidean algorithm. The details are enclosed in Table 6. The element λ of F is chosen such that the constant term of $f + \lambda g$ is zero. The algorithm terminates because the height decreases in each step.

4 The Gaussian integers

Elements in the form $a_1 + a_2 i$, where a_1, a_2 are integers, form a subring of the complex numbers. Let $\omega = 1 + i$, let $\alpha = a_1 + a_2 i \in R$. We define $H(\alpha)$ to be $a_1^2 + a_2^2$. Note that $\omega \mid \alpha$ if and only if $a_1 \equiv a_2 \pmod 2$ and the quotient α/ω can be computed easily. Namely, $\alpha/\omega = [(a_2 + a_1)/2] + [(a_2 - a_1)/2]i$. Note further that if $2 \mid a_1$ and $2 \mid a_2$, then $2 \mid \alpha$ and $\alpha/2$ can be computed easily as $\alpha/2 = (a_1/2) + (a_2/2)i$. We say that $\alpha = a_1 + a_2 i$ is of type (u_1, u_2) if

$$a_1 \equiv u_1 \pmod 2, \quad a_2 \equiv u_2 \pmod 2, \quad 0 \le u_1, u_2 \le 1.$$

The details of an algorithm to compute a greatest common divisor of $\alpha = a_1 + a_2 i$ and $\beta = b_1 + b_2 i$ are given in Table 7.

As an example we worked out the details when $\alpha = 5 + 8i$ and $\beta = 2 - 5i$. (See Table 8.)

In Table 7 we distinguish 3 types of entries or squares. "Division" squares in the 1st, 4th rows and columns. "Addition-subtraction" squares in the 2nd, 3rd position in the main diagonal. "Rotation" squares the remaining two squares. If we are in a "division" or in an "addition-subtraction" square, then the height decreases. If we are in a "rotation" square, then the quantity h_k does not change. But in the next step we end up in an "addition-subtraction" square and we have already seen that this leads to a decrease. We can say that in two steps there is a decrease in h_k. This guarantees that the algorithm terminates.

Let $\alpha = a_1 + b_1 i$, $\beta = b_1 + b_2 i$ and let t be the total number of bits of a_1, a_2, b_1, b_2. An argument similar to that we have seen in the proof of Proposition 3 gives that the Stein type algorithm for Gaussian integers can be completed with $O(t^2)$ bit operations. It is important to point out that in the course of

Table 7: $Z[i]$ with 4 types

	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$
$(0,0)$	$\alpha_{k+1}=\frac{\alpha_k}{2}$ $\beta_{k+1}=\frac{\beta_k}{2}$ $\delta_{k+1}=2\delta_k$	$\alpha_{k+1}=\frac{\alpha_k}{2}$ $\beta_{k+1}=\beta_k$ $\delta_{k+1}=\delta_k$	$\alpha_{k+1}=\frac{\alpha_k}{2}$ $\beta_{k+1}=\beta_k$ $\delta_{k+1}=\delta_k$	$\alpha_{k+1}=\frac{\alpha_k}{2}$ $\beta_{k+1}=\frac{\beta_k}{\omega}$ $\delta_{k+1}=\omega\delta_k$
$(0,1)$	$\alpha_{k+1}=\alpha_k$ $\beta_{k+1}=\frac{\beta_k}{2}$ $\delta_{k+1}=\delta_k$	$\alpha_{k+1}=\frac{\alpha_k+\beta_k}{2}$ $\beta_{k+1}=\frac{\alpha_k-\beta_k}{2}$ $\delta_{k+1}=\delta_k$	$\alpha_{k+1}=\alpha_k$ $\beta_{k+1}=\beta_k i$ $\delta_{k+1}=\delta_k$	$\alpha_{k+1}=\alpha_k$ $\beta_{k+1}=\frac{\beta_k}{\omega}$ $\delta_{k+1}=\delta_k$
$(1,0)$	$\alpha_{k+1}=\alpha_k$ $\beta_{k+1}=\frac{\beta_k}{2}$ $\delta_{k+1}=\delta_k$	$\alpha_{k+1}=\alpha_k$ $\beta_{k+1}=\beta_k i$ $\delta_{k+1}=\delta_k$	$\alpha_{k+1}=\frac{\alpha_k+\beta_k}{2}$ $\beta_{k+1}=\frac{\alpha_k-\beta_k}{2}$ $\delta_{k+1}=\delta_k$	$\alpha_{k+1}=\alpha_k$ $\beta_{k+1}=\frac{\beta_k}{\omega}$ $\delta_{k+1}=\delta_k$
$(1,1)$	$\alpha_{k+1}=\frac{\alpha_k}{\omega}$ $\beta_{k+1}=\frac{\beta_k}{2}$ $\delta_{k+1}=\omega\delta_k$	$\alpha_{k+1}=\frac{\alpha_k}{\omega}$ $\beta_{k+1}=\beta_k$ $\delta_{k+1}=\delta_k$	$\alpha_{k+1}=\frac{\alpha_k}{\omega}$ $\beta_{k+1}=\beta_k$ $\delta_{k+1}=\delta_k$	$\alpha_{k+1}=\frac{\alpha_k}{\omega}$ $\beta_{k+1}=\frac{\beta_k}{\omega}$ $\delta_{k+1}=\omega\delta_k$

Table 8: An example in $Z[i]$

k	α_k	$H(\alpha_k)$	β_k	$H(\beta_k)$	δ_k	h_k
1	$5+8i$	89	$2-5i$	29	1	118
2	$5+8i$	89	$5+2i$	29	1	118
3	$5+5i$	50	$3i$	9	1	59
4	5	25	$3i$	9	1	34
5	5	25	-3	9	1	34
6	4	16	1	1	1	17
7	1	1	1	1	1	2

Table 9: $Z[\sqrt{\pm 2}]$ with 2 types

	0	1
0	$\alpha_{k+1} = \frac{\alpha_k}{\omega}$ $\beta_{k+1} = \frac{\beta_k}{\omega}$ $\delta_{k+1} = \delta_k \omega$	$\alpha_{k+1} = \frac{\alpha_k}{\omega}$ $\beta_{k+1} = \beta_k$ $\delta_{k+1} = \delta_k$
1	$\alpha_{k+1} = \alpha_k$ $\beta_{k+1} = \frac{\beta_k}{\omega}$ $\delta_{k+1} = \delta_k$	$\alpha_{k+1} = \frac{\alpha_k + \beta_k}{\omega}$ $\beta_{k+1} = \frac{\alpha_k - \beta_k}{\omega}$ $\delta_{k+1} = \delta_k$

Table 10: An example in $Z[\sqrt{-2}]$

k	α_k	$H(\alpha_k)$	β_k	$H(\beta_k)$	δ_k	h_k
1	$9 + 8\sqrt{-2}$	209	$5 + 6\sqrt{-2}$	97	1	306
2	$7 + 7\sqrt{-2}$	147	$2 + \sqrt{-2}$	6	1	153
3	$7 + 7\sqrt{-2}$	147	$1 - \sqrt{-2}$	3	1	150
4	$3 + 4\sqrt{-2}$	41	$4 + 3\sqrt{-2}$	34	1	75
5	$3 + 4\sqrt{-2}$	41	$3 - 2\sqrt{-2}$	17	1	58
6	$3\sqrt{-2}$	18	$3 + \sqrt{-2}$	11	1	29
7	$3 + \sqrt{-2}$	11	3	9	1	20
8	3	9	$\sqrt{-2}$	2	1	11
9	3	9	1	1	1	10

the computation in this case we do not evaluate the H function. Computing $H(\alpha_k)$, $H(\beta_k)$ using the definition would jeopardize the $O(t^2)$ estimate.

5 The ring $Z[\sqrt{-2}]$

The elements of the form $a + b\sqrt{-2}$, where a, b are integers form a subring R of the complex numbers. Let $\omega = \sqrt{-2}$, $\alpha = a + b\sqrt{-2}$, $\beta = c + d\sqrt{-2}$. The function H is defined by $H(\alpha) = \alpha\overline{\alpha} = a^2 + 2b^2$. Note that $\omega \mid \alpha$ if and only if $2 \mid a$ and the quotient α/ω can be computed by $\alpha/\omega = b - (a/2)\sqrt{-2}$.

We say that $\alpha = a + b\sqrt{-2}$ is of type u if $a \equiv u \pmod 2$, $0 \leq u \leq 1$. The remaining part of the computation is given in Table 9. Clearly the height decreases in a "division" cell. In order to show that the algorithm terminates Note that

$$H(\alpha + \beta) + H(\alpha - \beta) = 2[H(\alpha) + H(\beta)].$$

This gives that

$$H[(\alpha + \beta)/\omega] + H[(\alpha - \beta)/\omega] = H(\alpha) + H(\beta).$$

The computation

$$\begin{array}{rcl} h_{k+1} & = & H[(\alpha_k + \beta_k)/\omega] + H[(\alpha_k - \beta_k)/\omega] \\ & = & H(\alpha_k) + H(\beta_k) \\ & = & h_k \end{array}$$

shows that in the "addition-subtraction" cell the height does not change. However, in the next step we must go to a "division" cell where the height decreases. The algorithm terminates because the quantity $h_k = H(\alpha_k) + H(\beta_k)$ eventually decreases.

As an illustration let us see what happens when $\alpha = 9 + 8\sqrt{-2}$ and $\beta = 5 + 6\sqrt{-2}$. (See Table 10.)

6 The ring $Z[\sqrt{2}]$

The elements of the form $a + b\sqrt{2}$, where a, b are integers form a subring R of the real numbers. Let $\omega = \sqrt{2}$, $\alpha = a + b\sqrt{2}$, $\beta = c + d\sqrt{2}$. The norm of α is $N(\alpha) = a^2 - 2b^2$ and we define the function H by $H(\alpha) = a^2 + 2b^2$. Now H does not have properties (a), (b). However $H(\alpha/\omega) = H(\alpha)/2$ holds. Further $H(\alpha) = 1$ implies $\alpha = -1$ or $\alpha = 1$. These properties are sufficient for our purposes. Note that $\omega \mid \alpha$ if and only if $2 \mid a$ and the quotient α/ω can be computed by $\alpha/\omega = b + (a/2)\sqrt{2}$.

The type of $\alpha = a + b\sqrt{2}$ is u if $a \equiv u \pmod 2$, $0 \leq u \leq 1$. The remaining part of the computation is given in Table 9.

7 The integers in $Q(\sqrt{-3})$

Let $\vartheta = \frac{1}{2}(1+\sqrt{-3})$. The elements of the form $a+b\vartheta$, where a, b are integers form a subring R of the complex numbers. The elements of R can be written in the form $\frac{1}{2}(a+b\sqrt{-3})$, where a, b are integers $a \equiv b \pmod 2$. For $\alpha = \frac{1}{2}(a+b\sqrt{-3})$ we define $H(\alpha) = \frac{1}{4}(a^2+3b^2)$ which is equal to the norm of α.

The modulo 4 type of α is (u, v) if

$$a \equiv u \pmod 4, \quad b \equiv v \pmod 4, \quad 0 \le u, v \le 3.$$

As u, v ranges independently from 0 to 3, the pair (u, v) ranges over 16 values. However, as $u \equiv v \pmod 2$ must hold, there are only 8 choices for (u, v).

Let (u, v) and (r, s) be the types of α_k and β_k respectively. The units in R are ϑ^i, $0 \le i \le 5$. From the equation

$$\alpha\vartheta = \frac{1}{2}(a+b\sqrt{-3})\frac{1}{2}(1+\sqrt{-3}) = \frac{1}{2}\Big[\frac{(a-3b)}{2} + \frac{(a+b)}{2}\sqrt{-3})\Big]$$

it follows that if α_k is of type $(1, 3)$, $(3, 1)$, then $\alpha_k\vartheta$ is of type $(0, 2)$. From the equation

$$\alpha\overline{\vartheta} = \frac{1}{2}(a+b\sqrt{-3})\frac{1}{2}(1-\sqrt{-3}) = \frac{1}{2}\Big[\frac{(a+3b)}{2} + \frac{(b-a)}{2}\sqrt{-3})\Big]$$

it follows that if α_k is of type $(1, 1)$, $(3, 3)$, then $\alpha_k\overline{\vartheta}$ is of type $(2, 0)$. Using suitable rotations if necessary we may assume that the type of α_k and β_k is one of the following $(0, 0)$, $(2, 0)$, $(0, 2)$, $(2, 2)$.

If $u = v$, then 2 (as an element of R) divides α_k and we can set α_{k+1} to be $\alpha_k/2$. In other words we are in a "division" cell. We handle β_k similarly.

In the remaining cases the types of α_k and β_k may be assumed to be $(0, 2)$, $(2, 0)$. In these cases we are in an "addition-subtraction" cell.

The usual consideration on the height shows that the algorithm terminates.

8 The integers of $Q(\sqrt{-7})$

Let $\vartheta = \frac{1}{2}(1+\sqrt{-7})$. The elements of the form $a+b\vartheta$, where a, b are integers form a subring R of the complex numbers. Elements of R can be written in the form $\frac{1}{2}(a+b\sqrt{-7})$, where a, b are integers $a \equiv b \pmod 2$. Let

$$\alpha = \frac{1}{2}(a+b\sqrt{-7}), \quad \beta = \frac{1}{2}(c+d\sqrt{-7}).$$

The function H is defined by $H(\alpha) = \frac{1}{4}(a^2+7b^2)$, that is, $H(\alpha)$ is the norm of α. Clearly $H(\vartheta) = 2$.

Note that $\vartheta \mid \alpha$ if and only if $a \equiv b \pmod 4$. Indeed, from the equation $\alpha = \beta\vartheta$ it follows that $c = \frac{1}{4}(a+7b)$, $d = \frac{1}{4}(b-a)$. Thus $a \equiv b \pmod 4$ imply that c, d are integers and $c \equiv d \pmod 2$. The modulo 4 type of α is (u, v) if

$$a \equiv u \pmod 4, \quad b \equiv v \pmod 4, \quad 0 \le u, v \le 3.$$

Table 11: The quaternions

	$(1,1,1,1)$	$(3,3,3,3)$
$(1,1,1,1)$	$\alpha_{k+1}=\frac{\alpha_k+\beta_k}{2}$ $\beta_{k+1}=\frac{\alpha_k-\beta_k}{2}$ $\delta_{k+1}=\delta_k$	$\alpha_{k+1}=\frac{\alpha_k+\beta_k}{2}$ $\beta_{k+1}=\frac{\alpha_k-\beta_k}{2}$ $\delta_{k+1}=\delta_k$
$(3,3,3,3)$	$\alpha_{k+1}=\frac{\alpha_k+\beta_k}{2}$ $\beta_{k+1}=\frac{\alpha_k-\beta_k}{2}$ $\delta_{k+1}=\delta_k$	$\alpha_{k+1}=\frac{\alpha_k+\beta_k}{2}$ $\beta_{k+1}=\frac{\alpha_k-\beta_k}{2}$ $\delta_{k+1}=\delta_k$

There are 8 possible choices for the type of α.

We compute the modulo 4 types of α_k, β_k. If the type of α_k or β_k is one of $(0,0)$, $(1,1)$, $(2,2)$, $(3,3)$, then we can divide by ϑ, that is, we are in a "division" cell. In the remaining cases we may assume that the type of α_k and β_k is one of the following $(0,2)$, $(2,0)$, $(1,3)$, $(3,1)$. One can check that in each case ϑ divides $\alpha_k \pm \beta_k$ and so we are in an "addition-subtraction" cell. A routine consideration gives that the height decreases in each case.

9 The Hurwitz quaternions

Let $\zeta = \frac{1}{2}(1+i+j+k)$. The elements $a_0 + a_1 i + a_2 j + a_3 \zeta$, where a_0, a_1, a_2, a_3 are integers form a non-commutative subring R of the quaternions, the so-called Hurwitz ring of integral quaternions. It is proved in [5] that the one sided ideals of R are principal. This property of R then was used to prove Lagrange's theorem about four squares of integers. We will show that Stein's algorithm works in R which also establishes the above property of R.

The elements of R can be written in the form $\frac{1}{2}(a_0+a_1 i+a_2 j+a_3 k)$, where $a_0 \equiv a_1 \equiv a_2 \equiv a_3 \pmod 2$. Let $\alpha = \frac{1}{2}(a_0+a_1 i+a_2 j+a_3 k)$. The function H is defined by $H(\alpha) = \frac{1}{4}(a_0^2+a_1^2+a_2^2+a_3^2)$. We say that (u_0,u_1,u_2,u_3) is the type of α if

$$a_0 \equiv u_0, \quad a_1 \equiv u_1, \quad a_2 \equiv u_2, \quad a_3 \equiv u_3 \pmod 4, \quad 0 \le u_0,u_1,u_2,u_3 \le 3.$$

There are 16 types (u_0,u_1,u_2,u_3), where the components take the values 0, 2 independently. We call these types even types. There are 16 types (u_0,u_1,u_2,u_3), where the components take the values 1, 3 independently. We call these types odd types. As usual we describe the algorithm by means of a 32 by 32 table, where the rows are labeled by the possible types of α_k and the columns are labeled by the possible types of β_k.

Let $\omega = 1+i$. The element $\alpha \in R$ is a left multiple of ω if there is a $\beta = \frac{1}{2}(b_0+b_1 i+b_2 j+b_3 k)$ in R such that $\alpha = \beta\omega$. A routine computation

reveals that

$$b_0 = \frac{a_1 + a_0}{2}, \quad b_1 = \frac{a_1 - a_0}{2}, \quad b_2 = \frac{a_2 - a_3}{2}, \quad b_3 = \frac{a_2 + a_3}{2}.$$

It follows that α is a left multiple of ω if and only if the type of α is even. If a row or a column is labeled by an even type, then all of its squares are "division" squares. So we can focus our attention to squares whose rows and columns are labeled with odd types.

Next we use the fact that ζ is a unit in R. Assume that the number of 1's and the number of 3's in the type of α is odd. One can check that in this case the type of $\alpha\zeta$ is even. As a consequence we left with the cases when the number of 1's and the number of 3's is even in the type of α. We will use the fact that i, j, k are units in R. We also will use the following three observations.

If the type of α is $(1,3,3,1)$, $(3,1,1,3)$, then the type of αi is one of $(1,1,1,1)$, $(3,3,3,3)$.

If the type of α is $(1,1,3,3)$, $(3,3,1,1)$, then the type of αj is one of $(1,1,1,1)$, $(3,3,3,3)$.

If the type of α is $(1,3,1,3)$, $(3,1,3,1)$, then the type of αi is one of $(1,1,1,1)$, $(3,3,3,3)$.

Therefore we can focus our attention to the squares that are labeled with the types $(1,1,1,1)$ and $(3,3,3,3)$. The instructions are listed in Table 11. The standard argument shows that the height $h_k = H(\alpha_k) + H(\beta_k)$ eventually decreases and so the computation cannot go forever.

10 Open problems

We close the paper with some topics for future work.

Problem 1 *There are five imaginary quadratic fields admitting Euclidean algorithm. For a reference see [4] page 213. By the present paper four of them admit Stein type algorithms leaving the $Q(\sqrt{-11})$ case undecided. As -11 is congruent to 5 modulo 8, the prime 2 in Z stays prime in $Q(\sqrt{-11})$. Is there a Stein type algorithm similar to the one in $Q(\sqrt{-3})$?*

Problem 2 *There is a list of all quadratic fields that are Euclidean. Do they have binary algorithms?*

Problem 3 *Higher degree Euclidean fields are known. Can binary algorithms be found for them?*

Problem 4 *Is there an analog to the Lehmer method, which works with short approximations rather than the complete number?*

Problem 5 *Is there a simple to state field property that guarantees the existence of a binary algorithm? That is, along with the known lists of Euclidean number fields, could one compile lists of "Steinian" fields?*

Problem 6 *Are there any non-commutative rings worth considering beside the Hurwitz quaternions?*

References

[1] E. Bach and J. Shallit, *Algorithmic Number Theory, Volume I: Efficient Algorithms*, MIT Press 1996.

[2] B. Bougaut, Algorithme explicite pour la recherce du P. G. C. D. dans certains anneaux principaux d'entiers de corps de nombres, *Theoretical Computer Science* **11** (1980), 207–220.

[3] T. H. Cormen, C. L. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd Edition, McGraw-Hill, 2001.

[4] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*, 5th Edition, Oxford, University Press, 1979.

[5] I. N. Herstein, *Topics in Algebra*, 2nd Edition, John Wiley, New York 1975. (pages 371–377)

[6] E. Kartofeln and H. Rolletschek, Computing greatest common divisors and factorizations in quadratic number fields, *Mathematics of Computation* **53** (1989), 697–720.

[7] A. Knopfmacher and J. Knopfmacher, The number of steps in the Euclidean algorithm over complex quadratic fields, *BIT NUMERICAL MATHEMATICS* **31** (1991), Number 2, 286–292.

[8] D. N. Knuth, *The Art of Computer Programming* Volume 2 *Seminumerical algorithms*, 3rd Edition, Addison-Wesley, 1997. (pages 338–341, 348–353)

[9] T. W. Körner, *The Pleasures of Counting*, Cambridge University Press, 1996. (pages 248–249)

[10] M. Nagata, A pairwise algorithm and its applications to $Z[\sqrt{14}]$, in *Proceedings of the Algebraic Geometry Seminar, Singapore, 3-6 November 1987, National University of Singapore* (M. Nagata, T.A. Peng, eds) World Scientific, 1988, 69–74.

[11] I. Niven and H. S. Zuckerman, *An Introduction to the Theory of Numbers*, 4th Edition, John Wiley, New York 1980.

[12] H. Rolletschek, Shortest division chains in imaginary quadratic fields, *Journal of Symbolic Computation* **9** (1990), Issue 3, 321–354.

[13] A. Weilert, $(1+i)$-ary gcd computation in $Z[i]$ as analogue to the binary gcd algorithm, *Journal of Symbolic Computation* **30** (2000), Issue 5, 605–617.

Received by the editors January 21, 2010