Proceedings of Symposia in Applied Mathematics
Volume **00**, 0000

# Algorithms for quadratic orders

INGRID BIEHL AND JOHANNES BUCHMANN

Abstract. We describe deterministic algorithms for solving the following algorithmic problems in quadratic orders: Computing fundamental unit and regulator, principal ideal testing, solving prime norm equations, computing the structure of the class group, computing the order of an ideal class and determining discrete logarithms in the class group. We also prove upper bounds for the time and space complexity of the algorithms.

## 1. Introduction

A *quadratic discriminant* is an integer $\Delta \in \mathbb{Z}$ which is not a square in $\mathbb{Z}$ such that $\Delta \equiv 0, 1 \mod 4$. By $\mathcal{D}$ we denote the set of all quadratic discriminants.

Let $\Delta \in \mathcal{D}$. By $\sqrt{\Delta}$ we denote the square root of $\Delta$ in $\mathbb{C}$ which for $\Delta > 0$ is positive and for $\Delta < 0$ has positive imaginary part. Then

$$\mathcal{O}_\Delta = \mathbb{Z} + \frac{\Delta + \sqrt{\Delta}}{2}\mathbb{Z}$$

is the *quadratic order* of discriminant $\Delta$.

Important computational problems for $\mathcal{O}_\Delta$ are the following:

- For $\Delta > 0$ find the fundamental unit and the regulator $\mathrm{R}_\Delta$ of $\mathcal{O}_\Delta$.
- Decide whether an ideal of $\mathcal{O}_\Delta$ is principal. If it is, find a generator.
- Decide whether a prime number $p$ is a norm in $\mathcal{O}_\Delta$. If it is, find a number in $\mathcal{O}_\Delta$ of norm $p$.
- Find the ideal class group of $\mathcal{O}_\Delta$.
- Solve the discrete logarithm problem in the class group of $\mathcal{O}_\Delta$.

In this paper we describe deterministic algorithms for solving these problems and we prove upper bounds for their time and space bit complexity. To give a

1991 *Mathematics Subject Classification.* 11Y16.
This paper is in final form and no version of it will be submitted for publication elsewhere.

simpler form for these upper bounds we write $T = O^*(f)$ for functions $f, T : \mathbb{Z}_{>0} \to \mathbb{Z}_{>0}$ if $T \le f \cdot p$ with some polynomial function $p : \mathbb{Z}_{>0} \to \mathbb{Z}_{>0}$. Also, for $\Delta < 0$ we let $\mathrm{R}_\Delta = 1$. We prove the following upper bounds both for time and space complexity:

| Problem | complexity |
|---|---|
| Fundamental unit | $O^*(\mathrm{R}_\Delta^{1/2})$ |
| Principal ideal testing | $O^*(\mathrm{R}_\Delta^{1/2})$ |
| Norm $p$ problem | $O^*(p^{1/2} + \mathrm{R}_\Delta^{1/2})$ |
| Class group | $O^*(\|\Delta\|^{1/2})$ |
| Discrete logarithm | $O^*(\|\Delta\|^{1/4})$ |

Algorithms for solving all the computational problems mentioned above were already developed by Lagrange, Gauß and others in the 18th century. While the running time of these classical methods is roughly the square of the running time of the algorithms described in this paper, they only require space $O^*(1)$ (see [**8**]). So they are more useful for hand computations.

With the invention of computers it became possible to handle large sets of data. It was Shanks [**16**], [**17**] who discovered that the running times of all the classical methods could be reduced to their square root at the expense of using a lot more storage. Some of the ideas of Shanks are explained and analyzed in papers of Lenstra [**11**] and Schoof [**15**]. There, however, the following problem was not solved. If for $\Delta > 0$ one represents the fundamental unit, the generator of an ideal, or a number of norm $p$ in the standard way as $(x + y\sqrt{\Delta})/2$ with $x, y \in \mathbb{Z}$, then the number of bits required to write down this representation may be of the order of magnitude $\mathrm{R}_\Delta$. It is therefore not possible to prove the asserted complexity results using that representation. One can only prove upper running time bounds for computing an approximation to the regulator and for deciding principality or the existence of a number of norm $p$. The complexity results can only be shown with a new *binary representation* of numbers in the field of fractions $\mathrm{K}_\Delta$ of $\mathcal{O}_\Delta$. This representation was introduced in [**2**]. The binary length of the binary representation for the fundamental unit, for example, is $O^*(1)$. Most of the operations that one wants to perform with numbers in $\mathrm{K}_\Delta$ can be carried out in polynomial time with numbers given in binary representation. One can, for example, verify in polynomial time that a unit given in binary representation is a unit, i.e., generates the principal ideal $\mathcal{O}_\Delta$. Similar ideas were used in [**4**] and [**7**]. While Shanks relied on using probabilistic algorithms for finding a generating system for the class group, we restrict ourselves in this paper to deterministic methods.

Generalizations of our complexity results to arbitrary number fields can be found in [**1**]. Stronger results can be obtained if one is willing to assume the truth of the generalized Riemann hypothesis or if one applies randomized algorithms. There are various results in this direction. A systematic presentation will be given in a forthcoming paper.

The paper is organized as follows. After a few preliminary remarks we present algorithms for finite abelian groups which are given in such a way that multiplication, inversion and equality testing is possible. In particular, we describe algorithms for finding the structure of the group and for computing discrete logarithms. All these algorithms are based on ideas of Shanks. These algorithms are then applied to the class group. We define that group, and we describe how elements are represented using reduced ideals. We show how multiplication and inversion can be performed in polynomial time. We also show how equality testing is possible in polynomial time if $\Delta < 0$. We then discuss in great detail the problem of deciding whether an ideal class belongs to a given set of ideal classes for $\Delta > 0$. Again, we use an idea of Shanks to develop an algorithm for solving that problem. At the same time we are able to present algorithms for determining the fundamental unit, for principal ideal testing and for solving the norm $p$ problem. At the end of the paper we present an algorithm of H.W. Lenstra [**13**] for computing the class group.

## 2. Preliminaries

**2.1. Representation of rational numbers.** With a few exceptions, which will be explicitly mentioned, we use for *integers* the binary representation. The *binary length* of an integer $z \neq 0$ is $\operatorname{size}(z) = \lfloor \log |z| \rfloor + 2$, where log denotes the logarithm to base 2. The extra bit encodes the sign of $z$. We also set $\operatorname{size}(0) = 2$.

Each mathematical object which we consider in this paper is represented by a finite sequence of integers. The binary length of such a representation is the sum of the binary lengths of the sequence elements. A rational number $q$, for example, is represented as $q = u/v$ with $u, v \in \mathbb{Z}$, $v > 0$.

**2.2. Approximations to complex numbers.** Let $c$ be a complex number, and let $q$ be an integer. An *approximation of precision* $q$ to $c$ is a number $\hat{c} \in 2^{-(q+1)}\mathbb{Z}[i]$ such that $|c - \hat{c}| < 2^{-q}$.

**2.3. Ordered sets.** An *ordered set* is a pair $G = (S(G), <_G)$, where $S(G)$ is a set and $<_G$ is a total ordering of $S(G)$. We also consider $\emptyset$ as an ordered set. Instead of explicitly defining $S(G)$ and $<_G$, we often just write the series of elements of $S(G)$ in increasing order with respect to $<_G$. So, for example, $(g, h, k)$ is the ordered set $(S(G), <_G)$ with $S(G) = \{g, h, k\}$ and $g <_G h$, $h <_G k$ and $g <_G k$.

Let $G$ be an ordered set. We write $\sharp G$ for cardinality of $S(G)$, and if $h \in S(G)$, we write $h \in G$. A *subset* of $G$ is an ordered set $G'$ where $S(G') \subset S(G)$ and $<_{G'}$ is $<_G$ restricted to $S(G')$. If $G'$ is another ordered set which is disjoint from $G$, i.e., $S(G)$ and $S(G')$ are disjoint, then $G \circ G' = (S(G) \cup S(G'), <_{G \circ G'})$, where on $G$ and $G'$ the ordering $<_{G \circ G'}$ agrees with $<_G$ and $<_{G'}$, respectively, and where $g <_{G \circ G'} g'$ for every $g \in G$, $g' \in G'$.

The set of all maps from $S(G)$ to $\mathbb{Z}$ is denoted by $\mathbb{Z}^G$. For $v, w \in \mathbb{Z}^G$ we write $v < w$ if $v(g) < w(g)$ for all $g \in G$. Moreover, we use the *lexicographic ordering* on $\mathbb{Z}^G$: $v$ is less than $w$ with respect to the lexicographic ordering if $v(g) < w(g)$ for the smallest element $g \in G$ on which $v$ and $w$ disagree.

A map $f \in \mathbb{Z}^G$ can be interpreted as a vector $(f(g))_{g \in G}$ and vice versa. By 1 we denote the vector in $\mathbb{Z}^G$ whose first component is 1 and whose other components are 0. Let $G'$ be a subset of $G$. Then each $y \in \mathbb{Z}^{G'}$ can be *extended* to a vector in $\mathbb{Z}^G$ by setting $y(g) = 0$ for $g \in G$, $g \notin G'$. For a set $T$ the system of all ordered subsets of $T$ is denoted by $\mathcal{G}(T)$.

## 3. Finite abelian groups

In order to describe the computation of the ideal class group, we present an algorithm for finding the structure of a finite abelian group $H$. We also solve the problem of computing orders of elements and determining discrete logarithms.

**3.1. Group operations.** We assume that $H$ is a multiplicatively written finite abelian group. The elements of $H$ are represented by finite sequences of integers. That representation is not necessarily unique. But we assume that we can carry out the following *group operations*: multiplication, inversion and deciding equality. If we can decide equality of two elements, then we can also decide whether an element belongs to a subset $T$ of $H$. Such an operation is called a *containment test*.

**3.2. Relation lattice.** Let $G$ be an ordered subset of $H$. For $e \in \mathbb{Z}^G$ we set

$$G^e = \prod_{g \in G} g^{e(g)}.$$

The *subgroup of $H$ generated by the elements of $G$* is

$$\langle G \rangle = \{G^e : e \in \mathbb{Z}^G\}.$$

If $\langle G \rangle = H$, then $G$ is called a *generating system* for $H$.

The map

$$\mathbb{Z}^G \to \langle G \rangle, \quad (e(g))_{g \in G} \mapsto \prod_{g \in G} g^{e(g)}$$

is a surjective homomorphism. Its kernel $L(G)$ is a $\sharp G$-dimensional sublattice of $\mathbb{Z}^G$ and

$$\langle G \rangle \cong \mathbb{Z}^G / L(G).$$

The lattice $L(G)$ is called the *relation lattice* for $G$. Its elements are called *relations.* Lattices are represented by bases.

**3.3. Structure.** Let $s_1, \ldots, s_k$ be the *elementary divisors* of $H$, i.e., the $s_i$ are positive integers for $1 \leq i \leq k$, $s_i$ divides $s_{i+1}$ for $1 \leq i \leq k-1$ and

$$H \cong \bigoplus_{i=1}^{k} \mathbb{Z}/s_i\mathbb{Z}. \tag{1}$$

If we know a generating system $G$ and the relation lattice $L(G)$, then the elementary divisors of $H$, the isomorphism (1) and its inverse can be determined in polynomial time using the algorithm of [**6**]. Hence, finding $(G, L(G))$ means determining the structure of $G$.

**3.4. The bounded relation problem.** The *bounded relation problem* is the following. Given an ordered set $G$ in $H$ and an upper bound $u \in \mathbb{Z}^G$, $u > 1$, decide whether there is a relation $y$ on $G$ with $1 \leq y < u$. If there is one, find the minimal one with respect to the lexicographic ordering.

The problems of computing the structure of $H$, determining the order of an element in $H$ and finding discrete logarithms in $H$ can be reduced to solving the bounded relation problem.

The solution of the bounded relation problem is based on the following statement.

**Proposition 3.1.** *Let $G$ be an ordered subset of $H$, $u \in \mathbb{Z}^G$, $u > 1$. Let $y$ be the minimal such relation on $G$ with $y \geq 1$. If $y < u$, then $y$ can be written as $y = q\lfloor\sqrt{u}\rfloor + r$ with $q, r \in \mathbb{N}^G$, $0 \leq q^2 < u$ and $1 \leq r^2 < u$.*

This statement leads immediately to the following algorithm RELATION which is based on [**16**] and in which we use NEXT which, given $G$ and $x \in \mathbb{Z}^G$, computes the successor of $x$ according to the lexicographical ordering.

For RELATION we get the next theorem.

**Theorem 3.2.** *RELATION is correct and requires $O(\sqrt{k})$ operations and tests for containment in a subset of $H$ with $O(\sqrt{k})$ elements, where $k = \prod_{g \in G} u(g)$. RELATION also requires space for $O(\sqrt{k})$ elements of $H$.*

**Algorithm 3.3.** (RELATION)

**Input:** $H,\ G \in \mathcal{G}(H),\ u \in \mathbb{Z}^G,\ u > 1$.
**Output:** *If there is* $y \in L(G)$ *with* $1 \le y < u$, *then the minimal such* $y$, *else* $y = 0$.

| | | | |
|---|---|---|---|
| $S := \emptyset;\ r := 1;\ y := 0$ | | | *(1)* |
| *REPEAT* | | | *(2)* |
| | $h := G^r$ | | *(3)* |
| | $S := S \cup \{(h, r)\}$ | | *(4)* |
| | $r :=$ *NEXT*$(G, r)$ | | *(5)* |
| *UNTIL* $r^2 \ge u$ *or* $h = 1$ | | | *(6)* |
| $q := 0$ | | | *(7)* |
| *REPEAT* | | | *(8)* |
| | $h := G^{-q\lfloor\sqrt{u}\rfloor}$ | | *(9)* |
| | *IF* $(h, r) \in S$ *for some* $r$ | | *(10)* |
| | *THEN* | $y := q\lfloor\sqrt{u}\rfloor + r$ | *(11)* |
| | *ELSE* | $q :=$ *NEXT*$(G, q)$ | *(12)* |
| *UNTIL* $y \ne 0$ *or* $q^2 \ge u$ | | | *(13)* |

**3.5. Computing relation lattices.** The algorithm BASIS which finds the relation lattice $L(G)$ for an ordered set $G$ in $H$ and which is described in this section is based on the following statement.

**Proposition 3.4.** *Let* $G$ *be an ordered subset of* $H$. *For* $h \in G$ *let* $e(h) = (e(g, h))_{g \in G}$ *be the extension of the minimal relation on* $\{g \in G : g \ge h\}$ *to* $G$. *Then* $e = (e(h))_{h \in G}$ *is a basis of* $L(G)$ *with* $e(g, h) \ge 0$ *and*

$$e(g,h)\begin{cases} = 0 & \text{if } g <_G h,\\ > 0 & \text{if } g = h,\\ < e(g,g) & \text{if } h <_G g,\end{cases}$$

*for all* $g, h \in G$. *Also, the basis* $e$ *is uniquely determined by these conditions.*

We say that the basis from Proposition 3.4 is in *Hermite normal form* (HNF).

**Theorem 3.5.** *BASIS is correct and requires* $O(\sharp G(\log k)\sqrt{k})$ *operations and tests for containment in each of* $O(\sharp G \log k)$ *subsets of* $H$ *with* $O(\sqrt{k})$ *elements, where* $k = \sharp\langle G\rangle$. *BASIS also requires space for* $O(\sqrt{k})$ *elements in* $G$.

*Proof.* The number of iterations of the FOR loop is $\sharp G$. Let $e \in \mathbb{Z}^{G\times G}$ be the HNF basis of $L(G)$. Also, let $k = \sharp\langle G\rangle$. Suppose that the algorithm is executing an iteration of the FOR loop. Then $u(g) = e(g,g)$ for every $g$ with $g >_G h$. We also always have $u(h) \leq 2e(h,h)$. Therefore, the number of iterations in the REPEAT loop is $O(\log e(h,h))$. Since $k = \prod_{g \in G} e(g,g)$, it follows from Theorem 3.2 that the iteration of the FOR loop requires $O((\log k)\sqrt{k})$ operations and tests for containment in each of $\log k$ subsets of $H$ with $O(\sqrt{k})$ elements. The space required is $O(\sqrt{k})$. □

**Algorithm 3.6.** (BASIS)

**Input:** *$H$, $G \in \mathcal{G}(H)$.*
**Output:** *The HNF basis $e$ of $L(G)$.*

| | |
|---|---|
| $G' := \emptyset$ | *(1)* |
| *FOR all $h \in G$ in descending order* | *(2)* |
|     $G' := \{h\} \circ G'$; $u(h) := 2$ | *(3)* |
|     *REPEAT* | *(4)* |
|         $e(h) := \mathit{RELATION}(H, G', u)$ | *(5)* |
|         $u(h) := 2 * u(h)$ | *(6)* |
|     *UNTIL* $e(h) \neq 0$ | *(7)* |
|     *Extend $e(h)$ to $G$;* $u(h) := e(h,h)$ | *(8)* |

**3.6. Discrete logarithms.** For $g, h \in H$ with $g \in \langle h \rangle$, the *discrete logarithm* of $g$ to base $h$ is the minimal positive integer $y$ such that $g = h^y$. It is denoted by $\log_h g$.

**Algorithm 3.7.** (DISCRETE LOG)

**Input:** *$H$, $g, h \in H$.*
**Output:** *$y = \log_h g$, if it exists, and $y = 0$ otherwise.*

| | |
|---|---|
| $u(g) := u(h) := 2$ | *(1)* |
| *REPEAT* | *(2)* |
|     $y := \mathit{RELATION}(H, (g,h), u)$ | *(3)* |
|     $o := \mathit{RELATION}(H, (h), u(h))$ | *(4)* |
|     $u(h) := 2u(h)$ | *(5)* |
| *UNTIL* $y + o \neq 0$ | *(6)* |

The algorithm for computing discrete logarithms is based on the following observation.

**Proposition 3.8.** *Let $g, h \in H$. If $g \neq h^y$ for all $y$ with $1 \leq y \leq$ order $h$, the discrete logarithm of $g$ to base $h$ does not exist.*

**Theorem 3.9.** *DISCRETE LOG is correct. The algorithm requires $O((\log k)\sqrt{k})$ operations and tests for containment in each of $O(\log k)$ subsets of $H$ with $O(\sqrt{k})$ elements as well as space for $O(\sqrt{k})$ elements of $H$. If the discrete logarithm of $g$ to base $h$ exists, then $k = \log_h g$. Otherwise, $k =$ order $h$.*

## 4. Quadratic numbers

Let $\Delta$ be a quadratic discriminant. The *field of fractions* of $\mathcal{O}_\Delta$ is $\mathrm{K}_\Delta = \mathbb{Q}(\sqrt{\Delta})$. The *standard representation* of a number $\alpha \in \mathrm{K}_\Delta$ is $\alpha = (x+y\sqrt{\Delta})/(2z)$ with $x, y, z \in \mathbb{Z}$, $z > 0$ and $\gcd(x, y, z) = 1$. The integer $z$ is called the *denominator* of $\alpha$ with respect to $\Delta$ we write $z = d(\alpha)$. The *conjugate* of $\alpha$ is $\overline{\alpha} = (x - y\sqrt{\Delta})/(2z)$. The *height* of $\alpha$ is $\mathrm{H}(\alpha) = \max\{|\alpha|, |\overline{\alpha}|\}$. The *norm* of $\alpha$ is $\mathrm{N}(\alpha) = \alpha\overline{\alpha}$. If $S$ is a set of numbers in $\mathrm{K}_\Delta$ then we write $\overline{S} = \{\overline{\alpha} : \alpha \in S\}$.

## 5. Ideals

We wish to use the algorithms from the previous section in class groups. In order to explain what they are, and how operations in those groups are performed, we first talk about ideals.

Let $\Delta$ be a quadratic discriminant. By an *ideal* of $\mathcal{O}_\Delta$ we always mean an *invertible ideal* of $\mathcal{O}_\Delta$.

**5.1. Representation.** A subset A of $\mathcal{O}_\Delta$ is an ideal of $\mathcal{O}_\Delta$ if and only if

$$\mathrm{A} = q\left(\mathbb{Z} + \frac{b+\sqrt{\Delta}}{2a}\mathbb{Z}\right), \tag{2}$$

where $a, b \in \mathbb{Z}$, $q \in \mathbb{Q}$, $a, q > 0$, $c = (b^2 - \Delta)/(4a) \in \mathbb{Z}$ and $\gcd(a, b, c) = 1$. In this representation, the numbers $a, q$ are uniquely determined and the integer $b$ is unique modulo $2a$. We write $a = a(\mathrm{A})$ and $q = q(\mathrm{A})$. A representation (2) is called *standard representation* if $|b| \leq \max\{a, \sqrt{|\Delta|}\}$. Ideals are always given in some standard representation. The *norm* of A is $q^2 a$. The ideal A is called *normal* if $q = 1$. The *denominator* $d(\mathrm{A})$ of A is the smallest positive integer $d$ such that $d\mathrm{A} \subset \mathcal{O}_\Delta$. If A is normal, then $d(\mathrm{A}) = a(\mathrm{A})$. The set $\mathrm{I}_\Delta$ of invertible ideals of $\mathcal{O}_\Delta$ is an abelian group.

**5.2. Inverse.** The *inverse* of an ideal A of $\mathcal{O}_\Delta$ is $\mathrm{A}^{-1} = (1/\mathrm{N}(\mathrm{A}))\overline{\mathrm{A}}$.

**5.3. Product.** The *product* of two ideals A, B can be determined as follows. Let $b(\mathrm{A})$, $b(\mathrm{B})$ be the numbers $b$ appearing in some standard representation of A and B (see § 5.1). Compute $m = \gcd(a(\mathrm{A}), a(\mathrm{B}), (b(\mathrm{A})+b(\mathrm{B}))/2)$. Let $j, k, \ell \in \mathbb{Z}$

with $ja(\mathrm{A}) + ka(\mathrm{B}) + \ell(b(\mathrm{A}) + b(\mathrm{B}))/2 = m$. Then $a(\mathrm{AB}) = a(\mathrm{A})a(\mathrm{B})/(m^2)$, $b(\mathrm{AB}) \equiv (ja(\mathrm{A})b(\mathrm{B}) + ka(\mathrm{B})b(\mathrm{A}) + \ell(b(\mathrm{A})b(\mathrm{B}) + \Delta)/2)/m \mod a(\mathrm{AB})$ and $q(\mathrm{AB}) = q(\mathrm{A})q(\mathrm{B})m$.

**5.4. Principal ideals.** By $\mathrm{K}_\Delta^*$ we denote the set of nonzero elements of $\mathrm{K}_\Delta$. A *principal ideal* of $\mathcal{O}_\Delta$ is an ideal of the form $\alpha\mathcal{O}_\Delta$ with $\alpha \in \mathrm{K}_\Delta^*$. The number $\alpha$ is called a *generator* of $\mathcal{O}_\Delta$. Let $\alpha \in \mathrm{K}_\Delta^*$, $\alpha = (x + y\sqrt{\Delta})/(2z)$, be its standard representation. The *principal ideal* $\alpha\mathcal{O}_\Delta$ can be found as follows. Compute $m = \gcd(y, (x + y\Delta)/2)$ and let $k, \ell \in \mathbb{Z}$ such that $ky + \ell(x + y\Delta)/2 = m$. Then $a(\alpha\mathcal{O}_\Delta) = a = |x^2 - \Delta y^2|/(4m^2)$, $b(\alpha\mathcal{O}_\Delta) \equiv b = (kx + \ell(x + y)\Delta/2)/m \mod 2a$ and $q(\alpha\mathcal{O}_\Delta) = ma/z$.

**5.5. Prime ideals.** We denote by $\mathbb{P}_\Delta$ the set of all prime numbers $p$ such that there is an invertible prime ideal of norm $p$ in $\mathcal{O}_\Delta$. For a prime number $p \in \mathbb{P}_\Delta$ there is a prime ideal

$$\mathrm{A}_p = \mathbb{Z} + \frac{b_p + \sqrt{\Delta}}{2p}\mathbb{Z},$$

where $b_p$ is the uniquely determined integer with $0 < b_p < p$, and $b_p^2 \equiv \Delta \mod 4p$. The only possible other prime ideal of norm $p$ is $\overline{\mathrm{A}_p}$.

On input of $\Delta$ and a prime number $p$, the procedure PRIME IDEAL returns $\mathrm{A}_p$ if $p \in \mathbb{P}_\Delta$ and 0 otherwise. In PRIME IDEAL the square root of $\Delta$ mod $p$ must be found. The fastest deterministic algorithm for factoring polynomials $\mod p$ requires time and space $O^*(\sqrt{p})$ (see [**18**]). Therefore, also PRIME IDEAL requires time and space $O^*(\sqrt{p})$.

## 6. Ideal class group

In this section we introduce the ideal class group and we explain how to perform the basic operations in these groups.

**6.1. Definition.** The set $\mathrm{P}_\Delta$ of all principal ideals of $\mathcal{O}_\Delta$ is a subgroup of the group $\mathrm{I}_\Delta$ of all ideals of $\mathcal{O}_\Delta$. The elements of the *ideal class group* $\mathrm{Cl}_\Delta = \mathrm{I}_\Delta/\mathrm{P}_\Delta$ are called *ideal classes* of $\mathcal{O}_\Delta$. The number $\mathrm{h}_\Delta = \sharp\mathrm{Cl}_\Delta$ is called *class number* of $\Delta$. Ideals $\mathrm{A}, \mathrm{B} \in \mathcal{O}_\Delta$ are called *equivalent* if they belong to the same ideal class. The class of an ideal A is denoted by [A].

**6.2. Representation.** For $a, b \in \mathbb{Z}$, $a > 0$, we denote by $\tau(a, b)$ the uniquely determined number $b'$ such that $b' \equiv b \mod 2a$ and

$$\begin{array}{ll} -2a + \sqrt{\Delta} < b' < \sqrt{\Delta} & \text{for } \Delta > 0 \text{ and } a < \sqrt{\Delta}, \\ -a < b' \le a & \text{for } \Delta < 0 \text{ or } \Delta > 0 \text{ and } a > \sqrt{\Delta}. \end{array}$$

For an ideal A of $\mathcal{O}_\Delta$ we define

$$b(\mathrm{A}) = \tau(b, a), \quad c(\mathrm{A}) = |b(\mathrm{A})^2 - \Delta|/(4a(\mathrm{A}))$$

and

$$\gamma(\mathrm{A}) = \frac{b(\mathrm{A}) + \sqrt{\Delta}}{2a(\mathrm{A})}.$$

An ideal A of $\mathcal{O}_\Delta$ is called *reduced* if it is normal and

$$\begin{array}{ll} b(\mathrm{A}) \geq \max\{0, 2a(\mathrm{A}) - \sqrt{\Delta}\} & \text{for } \Delta > 0, \\ |b(\mathrm{A})| \leq a(\mathrm{A}) \leq c(\mathrm{A}) & \text{for } \Delta < 0. \end{array}$$

Each ideal class contains a reduced ideal. Hence, we can in fact represent each ideal class by a reduced ideal. For $\Delta > 0$, however, this representation is far from being unique.

Here are a few properties of reduced ideals. If A is reduced, then $a(\mathrm{A}), |b(\mathrm{A})| < \sqrt{|\Delta|}$, and if $\Delta > 0$, then $\gamma(\mathrm{A}) > 1$ and $-1 < \overline{\gamma(\mathrm{A})} < 0$. Also, a normal ideal A is reduced if $a(\mathrm{A}) < \sqrt{|\Delta|}/2$.

**6.3. Multiplication, inversion.** If we want to determine the inverse of an ideal class or the product of two ideal classes, then we compute the product or the inverse of their reduced representatives. The resulting ideals are, in general, not reduced. Therefore, in this section we describe a polynomial-time algorithm REDUCE which on input of an ideal A computes a reduced ideal in the ideal class of A. Thus, computing inverses and products of ideal classes can be effected in polynomial time.

Let A be a normal ideal. Define

$$\rho(\mathrm{A}) = (1/\gamma(\mathrm{A}))\mathrm{A} = \mathbb{Z} + \frac{1}{\gamma(\mathrm{A})}\mathbb{Z}.$$

Then $\rho(\mathrm{A})$ is normal,

$$a(\rho(\mathrm{A})) = c(\mathrm{A}), \quad b(\rho(\mathrm{A})) = \tau(-b(\mathrm{A}), c(\mathrm{A})).$$

Here is the reduction algorithm which works in polynomial time.

**Algorithm 6.1.** (REDUCE)

**Input:** $\Delta \in \mathcal{D}$, $A \in I_\Delta$.
**Output:** *A reduced ideal* $B$ *in the class of* $A$, $\alpha \in A$ *with* $A = \alpha B$ *and* $H(\alpha) \le q(A)$.

| | |
|---|---|
| $\alpha := 1$; $B := A$ | *(1)* |
| *REPEAT* | *(2)* |
|     $\alpha := \alpha\gamma(B)$ | *(3)* |
|     $B := \rho(B)$ | *(4)* |
| *UNTIL* $B$ *is reduced* | *(5)* |

The output $\alpha$ of the algorithm REDUCE is called the reducing number of A.

**6.4. Containment problem.** Let A and B be ideals of $\mathcal{O}_\Delta$ in the same ideal class. Then there is $\alpha \in \mathrm{K}_\Delta^*$ such that $\mathrm{A} = \alpha\mathrm{B}$. The number $\alpha$ is called a *generator of A relative to B.* For $\Delta > 0$ we assume without loss of generality that $\alpha > q(\mathrm{A})$. So in this case, the *minimal generator* is well defined. For $\Delta < 0$ each generator of A relative to B is called minimal.

The *containment problem* for $\Delta$ is the following. Given a quadratic discriminant $\Delta$, a set $T$ of reduced ideals in $\mathcal{O}_\Delta$ and a reduced ideal A of $\mathcal{O}_\Delta$, i.e., decide whether A is equivalent to a reduced ideal B in $T$. If so, find the minimal generator of A relative to B.

**6.5. Solving the containment problem for $\Delta < 0$.** If $\Delta < 0$, then there are at most two reduced ideals in each ideal class. More precisely, let A be a reduced ideal of $\mathcal{O}_\Delta$; then the only other possible reduced ideal of the same class is $\rho(\mathrm{A})$.

Hence, for $\Delta < 0$ the containment problem can be solved as follows. Let $T$ be a set of reduced ideals and A be a reduced ideal of $\mathcal{O}_\Delta$. We order the elements of $T$ such that binary search is possible. Then we determine the reduced ideals contained in [A], and using binary search, we check whether one of these ideals belongs to $T$. If so, the generator of A relative to that ideal is either 1 or $\gamma(\mathrm{A})$. Thus, we have proved the following statement.

**Proposition 6.2.** *For* $\Delta \in \mathcal{D}$, $\Delta < 0$ *the containment problem can be solved using time and space* $O^*(\sharp T)$.

**6.6. The equivalence problem for $\Delta > 0$.** If $\Delta > 0$ then there can be more than $\sqrt{\Delta}$ reduced ideals in one ideal class. Hence in this case, the containment problem is more difficult to solve. We first consider a special case.

The *bounded equivalence problem* is the following. Given a positive quadratic discriminant $\Delta$, reduced ideals $\mathrm{A}, \mathrm{B}$ of $\mathcal{O}_\Delta$ and a positive integer $u$ , i.e., decide whether A has a generator $\alpha$ relative to B with $\alpha < u$. If so, find the minimal $\alpha$.

This problem is very similar to the discrete logarithm problem in an abelian group. It is, however, not clear whether the decision problem belongs to the complexity class NP. In the sequel, we will prove that this is the case. We will also present an algorithm to solve the equivalence problem. This algorithm is very similar to the discrete logarithm procedure from above.

**6.7. Minima.** Let $\Delta > 0$. To show that the equivalence decision problem belongs to NP, we study the representation of relative generators. Let A be an ideal of $\mathcal{O}_\Delta$. A generator of A relative to a reduced ideal of $\mathcal{O}_\Delta$ is called a *minimum* of A. The logarithm of a minimum of A can be considered as some sort of discrete logarithm. These logarithms form a discrete subset of the real line. Each minimum has a right and a left neighbor which is easily computable. This will now be explained in more detail.

The ideal A is reduced if and only if 1 is a minimum of A. Let $\alpha$ be a minimum of A; then

$$|\mathrm{N}(\alpha)| < \mathrm{N}(\mathrm{A})\sqrt{\Delta}.$$

The *right neighbor* of a minimum $\alpha$ in A is the uniquely determined minimum $\beta$ of A with minimal $\beta > \alpha$. It can be computed via

$$\mathrm{R\ MIN}(\Delta, \mathrm{A}, \alpha) = \alpha\gamma((1/\alpha)\mathrm{A}).$$

In particular, if A is reduced, then $\gamma(\mathrm{A})$ is the right neighbor of 1 in A. The *left neighbor* of a minimum $\alpha$ in A is the uniquely determined minimum $\beta$ of A with minimal $|\overline{\beta}| > |\overline{\alpha}|$. It can be computed via

$$\mathrm{L\ MIN}(\Delta, \mathrm{A}, \alpha) = \alpha\overline{\gamma(\overline{(1/\alpha)\mathrm{A}})}.$$

Let $\alpha, \beta$ be minima of A. Then $\beta$ is the right neighbor of $\alpha$ if and only if $\alpha$ is the left neighbor of $\beta$. Also $|\overline{\alpha}|, |\overline{\beta}|$ are minima of $\overline{\mathrm{A}}$ and $|\overline{\beta}|$ is the left neighbor of $\overline{\alpha}$ if $\beta$ is the right neighbor of $\alpha$.

Fix some minimum $\alpha_0$ of A. Let $(\alpha_i)_{i\in\mathbb{Z}}$ be the sequence of minima in A with the property that $\alpha_i$ is the right neighbor of $\alpha_{i-1}$ for $i \in \mathbb{Z}$. Then we have

$$\log\alpha_{i+2} - \log\alpha_i \geq 1, \quad i \in \mathbb{Z}, \tag{3}$$

and

$$0 < \log\alpha_{i+1} - \log\alpha_i < (\log\Delta)/2, \quad i \in \mathbb{Z}. \tag{4}$$

The previous statements show that minima have a fairly regular distribution on the real line.

The sequence $(\alpha_i)_{i\in\mathbb{Z}}$ is periodic in the following sense. There is $k = k(\mathrm{A}) \in \mathbb{Z}_{>0}$ such that $\alpha_i/\alpha_j$ is a unit in $\mathcal{O}_\Delta$ if and only if $i \equiv j \mod k$ for any $i, j \in \mathbb{Z}$. The number $\varepsilon = \alpha_k/\alpha_0$ is the fundamental unit of $\mathcal{O}_\Delta$. For every $i, l \in \mathbb{Z}$ we have $\alpha_{lk+i}/\alpha_i = \varepsilon^l$. The number $k(\mathrm{A})$ is an invariant of the ideal class of A. It is called the *period length* of [A]. If $k(\mathrm{A}) = 1$, then

$$\log\alpha_{i+1} - \log\alpha_i > 1/2, \quad i \in \mathbb{Z}. \tag{5}$$

**6.8. Reduced ideals for $\Delta > 0$.** Let A be an invertible ideal of $\mathcal{O}_\Delta$. The reduced ideals in [A] are exactly the ideals $(1/\alpha)\mathrm{A}$ where $\alpha$ is a minimum in A.

If A reduced, then the *right neighbor* of A is

$$\mathrm{R\ IDEAL}(\Delta, \mathrm{A}) = \rho(\mathrm{A})$$

and the *left neighbor* of A is

$$\mathrm{L\ IDEAL}(\Delta, \mathrm{A}) = \overline{\rho(\overline{\mathrm{A}})}.$$

If B and C are reduced ideals in the class of A, $\mathrm{B} = (1/\beta)\mathrm{A}$, $\mathrm{C} = (1/\gamma)\mathrm{A}$ with minima $\beta, \gamma$ of A, then C is the right neighbor of B if and only if $\gamma$ is the right neighbor of $\beta$. The graph of reduced ideals in the class of A is a cycle, the *cycle of reduced ideals* in the class of A. The number of elements of that cycle is $k(\mathrm{A})$.

**6.9. Binary representation of minima.** Let A be a reduced ideal of $\mathcal{O}_\Delta$ and $\alpha$ a minimum of A. A *binary representation* of $\alpha$ in A is of the form

$$\alpha = \prod_{i=1}^{n} \beta_i^{2^{n-i}},$$

where $n \in \mathbb{Z}_{\geq 1}$, $\beta_i \in \mathrm{K}_\Delta^*$, $1 \leq i \leq n$ satisfy

- $n \leq \log\log\alpha + 2$,
- $\mathrm{H}(\beta_i) < 16\Delta^{5/4}$ for $1 \leq i \leq n$,
- $\alpha_j = \prod_{i=1}^{j} \beta_i^{2^{j-i}}$ is a minimum in A for $1 \leq j \leq n$.

We write

$$\alpha = (\beta_1, \ldots, \beta_n).$$

It follows that the binary length of the standard representation of each $\beta_i$ is $O^*(1)$, and the binary length of a binary representation of $\alpha$ is $O^*(\log\log \mathrm{H}(\alpha))$. We will show below that all minima have binary representations. Minima in reduced ideals will be always given in binary representation. Note that

$$\log\alpha = \sum(\log\beta_i)2^{n-i}.$$

This looks like a real binary representation for the "discrete logarithm" $\log\alpha$ except that the $\log\beta_i$ are not bits but "small" numbers.

It is easy to see that there is a polynomial-time algorithm IDEAL that on input of $\Delta$, A and a minimum $\alpha$ of A in binary representation determines the ideal $(1/\alpha)$A. It is also easy to see that there is a polynomial-time algorithm APPROX that on input of $\Delta$, A, a minimum $\alpha$ of A in binary representation and a positive integer $q$ in unary representation determines an approximation of precision $q$ to $\log \alpha$.

**6.10. Finding minima of a given size.** Let A be a reduced ideal of $\mathcal{O}_\Delta$ and $s \in \mathbb{Q}_{>0}$. A minimum $\alpha$ in A is called *close to $s$* if

$$|s - \log \beta| > |s - \log \alpha| - 3/4 \tag{6}$$

for every minimum $\beta$ in A. From (3) and (4) one can deduce

$$|s - \log \alpha| < (\log \Delta + 3)/4, \tag{7}$$

and

$$\begin{aligned} &\max\{2^{s-3/4}\Delta^{-1/4}, 2^{-s-3/4}\mathrm{N(A)}\Delta^{-1/4}\} \\ &\leq \mathrm{H}(\alpha) \\ &\leq \max\{2^{s+3/4}\Delta^{1/4}, 2^{-s+3/4}\mathrm{N(A)}\Delta^{3/4}\}. \end{aligned} \tag{8}$$

We now present the algorithm CLOSE which on input of $\Delta$, A and $s$ finds in polynomial time a minimum $\alpha$ of A which is close to $s$. The main tool that is used in CLOSE is the procedure DOUBLE, which on input of $t \in \mathbb{Q}_{>0}$ and a minimum $(\alpha_1, \ldots, \alpha_{k-1})$ of A which is close to $t$ determines in polynomial time a minimum $(\alpha_1, \ldots, \alpha_{k-1}, \alpha_k)$ of A which is close to $2t$. Using DOUBLE as a subroutine, CLOSE works as follows.

**Algorithm 6.3.** (CLOSE)

**Input:** $\Delta \in \mathcal{D}$, $\Delta > 0$, *a reduced ideal $A$ of $\mathcal{O}_\Delta$*, $s \in \mathbb{Q}_{\geq 0}$.
**Output:** *A minimum $\alpha = (\alpha_1, \ldots, \alpha_n)$ of $A$ close to $s$, where $n = \lceil \log s \rceil + 1$ and $H(\alpha_i) \leq 4\Delta^{5/4}$ for $1 \leq i \leq n$ if $s > 0$, otherwise 1.*

| | | | |
|---|---|---|---|
| *IF* $s = 0$ | | | *(1)* |
| *THEN* | $\alpha = 1$ | | *(2)* |
| *ELSE* | $n := \lceil \log s \rceil + 1$ | | *(3)* |
| | $\alpha := 1$ | | *(4)* |
| | $t := s/2^n$ | | *(5)* |
| | *FOR* $i = 1, 2, \ldots, n$ | | *(6)* |
| | | $\alpha := DOUBLE(A, \alpha, t)$ | *(7)* |
| | | $t := 2t;$ | *(8)* |

**Proposition 6.4.** *CLOSE is correct and has running time $O^*(1)$.*

*Proof.* The assertion follows from the fact that DOUBLE has polynomial running time and that we have initially $t < 1/2$. □

Next we present the procedure DOUBLE. DOUBLE uses the procedure NEAREST, which on input of an ideal B and a rational number $u$ finds in polynomial time a minimum $\beta$ in B *very close to* $u$, i.e., for every minimum $\gamma$ of B we have

$$|u - \log\gamma| > |u - \log\beta| - 1/4.$$

**Algorithm 6.5.** (DOUBLE)

**Input**: $\Delta \in \mathcal{D}$, $\Delta > 0$, *a reduced ideal* $A$ *of* $\mathcal{O}_\Delta$, $t \in \mathbb{Q}_{>0}$, *a minimum* $(\alpha_1, \ldots, \alpha_{k-1})$ *of* $A$.

**Output**: *A minimum* $(\alpha_1, \ldots, \alpha_{k-1}, \alpha_k)$ *of* $A$ *close to* $2t$ *with* $H(\alpha_k) \leq 4\Delta^{5/4}$.

| | |
|---|---|
| $B := (1/\alpha^2)A$ | *(1)* |
| $u := 2(t - \mathit{APPROX}(\Delta, \alpha, 3))$ | *(2)* |
| $\alpha_k := \mathit{NEAREST}(\Delta, B, u)$ | *(3)* |

**Proposition 6.6.** *DOUBLE is correct and has running time $O^*(1)$.*

*Proof.* NEAREST determines a minimum $\alpha_k$ in B. Hence, $\alpha' = (\alpha_1, \ldots, \alpha_{k-1}, \alpha_k) = \alpha_k\alpha^2$ is a minimum in A. Let $\beta$ be a minimum in A. Then $\gamma = \beta/\alpha^2$ is a minimum in B. It follows that

$$\begin{aligned}|2t - \log\beta| &\geq |u - \log\gamma| - |2 \cdot \mathrm{APPROX}(\Delta, \alpha, 3) - 2\log\alpha| \\ &\geq |u - \log\gamma| - 1/4 \geq |u - \log\alpha_k| - 1/2 \\ &\geq |2t - \log\alpha'| - 3/4.\end{aligned}$$

Thus, $\alpha'$ is close to $2t$.

Next we estimate the parameter $u$ that is input for NEAREST. Since initially the minimum $\alpha$ is close to $t$, it follows that

$$|u| = 2|t - \mathrm{APPROX}(\Delta, \alpha, 3)| < (\log\Delta)/2 + 7/4. \tag{9}$$

It is easy to verify that $(1/\alpha^2)$A can be determined in polynomial time. Since B is the square of a reduced ideal, it follows that the running time of NEAREST is polynomial in $\log\Delta$. Also, it follows from (9) and (8) that $\mathrm{H}(\alpha_k) \leq 8\Delta^{5/4}$. □

Finally, we present NEAREST.

**Algorithm 6.7.** (NEAREST)

**Input:** $\Delta \in \mathcal{D}$, $\Delta > 0$, *an ideal* $B$ *of* $\mathcal{O}_\Delta$, $u \in \mathbb{Q}_{>0}$.
**Output:** *A minimum* $\beta \in B$ *very close to* $u$.

```
(C, γ) := REDUCE(Δ, B); c := APPROX(Δ, γ, 2)        (1)
IF  c ≤ u                                            (2)
THEN  REPEAT                                         (3)
          β := γ                                     (4)
          γ := R MIN(R MIN(Δ, B, γ))                 (5)
          c := APPROX(Δ, γ, 2)                       (6)
      UNTIL  c > u                                   (7)
ELSE  REPEAT                                         (8)
          γ := L MIN(L MIN(Δ, B, γ))                 (9)
          c := APPROX(Δ, γ, 2)                       (10)
          β := γ                                     (11)
      UNTIL  c < u                                   (12)
β0 := L MIN(Δ, B, β)                                 (13)
β1 := β                                              (14)
β2 := R MIN(Δ, B, β)                                 (15)
β3 := R MIN(Δ, B, β2)                                (16)
β4 := R MIN(Δ, B, β3)                                (17)
For β choose that element from {β0, β1, β2, β3, β4} such that   (18)
|u − APPROX(Δ, β, 2)| is minimal
```

**Proposition 6.8.** *NEAREST is correct and has running time* $O^*(1)$.

*Proof.* We show that NEAREST works correctly by proving that the *optimal minimum* of B, i.e., the minimum $\beta'$ with $|u - \log\gamma| \geq |u - \log\beta'|$ for every minimum $\gamma$ of B, belongs to the set $\{\beta_i : 0 \leq i \leq 4\}$ from which the output is chosen.

NEAREST starts by computing the reducing number $\gamma$ for B. Let $\gamma_0 = \gamma$, and let $(\gamma_i)_{i\in\mathbb{Z}}$ be the sequence of minima as in § 6.7. In the sequence $S = (\dots, \gamma_{-4}, \gamma_{-2}, \gamma_0, \gamma_2, \gamma_4, \dots)$ NEAREST determines the last minimum $\beta = \gamma_k$ such that APPROX$(\Delta, \beta, 2) \leq u$. That minimum is found in the two REPEAT loops since by (3), APPROX$(\Delta, \gamma_{i+2}, 2) -$ APPROX$(\Delta, \gamma_i, 2) \geq 1/2$.

If $\log\gamma_k > u$, then for $i \geq 1$ the minimum $\gamma_{k+i}$ cannot be optimal. Also, since

$\log\gamma_k < u + 1/4$, it follows from (3) that for $i \geq 2$ the minimum $\log\gamma_{k-i}$ cannot be optimal. If $\log\gamma_k < u$, then for $i \geq 1$ the minimum $\gamma_{k-i}$ cannot be optimal. Also, since $\log\gamma_{k+2} > u - 1/4$, it follows from (3) that for $i \geq 4$ the minimum $\log\gamma_{k+i}$ cannot be optimal.

Next, we must show that the running time of NEAREST is polynomial. In NEAREST we have initially $\mathrm{N}(\mathrm{B}) \leq \mathrm{H}(\beta) \leq q(\mathrm{B})$. It follows from this inequality and from (4) for all minima $\gamma$ computed in the algorithm that

$$\min\{\log(\mathrm{N}(\mathrm{B})), u - \log\Delta - 1/4\} \leq \log\gamma \leq \max\{0, u + \log\Delta + 1/4\}$$

for every minimum $\gamma$ computed in the algorithm. This shows that the sizes of the input to the procedures used in NEAREST are $O^*(1)$. □

**6.11. Computing binary representations for minima.** The procedure CLOSE is used in BINARY, which determines a binary representation for any minimum $\alpha$ of a reduced ideal A of $\mathcal{O}_\Delta$.

**Algorithm 6.9.** (BINARY)

**Input:** *$\Delta \in \mathcal{D}$, $\Delta > 0$, a reduced ideal $A$ of $\mathcal{O}_\Delta$, for some minimum $\alpha$ of $A$ the ideal $B = (1/\alpha)A$ and an approximation $a$ of $\log\alpha$ of precision 3.*

**Output:** *A binary representation of $\alpha$ in $A$.*

| | |
|---|---|
| $\alpha_1 :=$ *CLOSE*$(\Delta, A, a)$ | *(1)* |
| $\alpha_0 :=$ *L MIN*$(\Delta, B, \alpha_1)$; $\alpha_2 :=$ *R MIN*$(\Delta, B, \alpha_1)$ | *(2)* |
| *For $\alpha$ choose the element in $\{\alpha_0, \alpha_1, \alpha_2\}$ such that* <br> $\lvert$*APPROX*$(\Delta, \alpha, 3) - a\rvert \leq 1/2$ *and* $(1/\alpha)B = A$ | *(3)* |

BINARY is based on the following statement.

**Proposition 6.10.** *Let $\alpha$ be a minimum in the reduced ideal $A$. Let $a$ be an approximation to $\log\alpha$ of precision 3. Let $\alpha_1$ be a minimum of $A$ which is close to $a$, $\alpha_0$ the left neighbor of $\alpha_1$, $\alpha_2$ the right neighbor of $\alpha_1$, $M = \{\alpha_0, \alpha_1, \alpha_2\}$. Then $\alpha \in M$, and if $\alpha' \in M$, $a'$ an approximation of precision 3 to $\log\alpha'$, then $\alpha' = \alpha$ if and only if $|a' - a| < 1/4$ and $(1/\alpha')B = (1/\alpha)B$.*

*Proof.* Since $a$ is an approximation of precision 3 to $\log\alpha$, we have $|a - \log\alpha| < 1/8$. Since $\alpha_1$ is close to $a$, it follows from (6) that $|a - \log\alpha_1| < 3/4 + 1/8$, and thus $|\log\alpha - \log\alpha_1| < 1$. By (3) we have $|\log\beta - \log\alpha_1| > 1$ for every minimum $\beta$ in the sequence of right neighbors of $a_2$ and also for every minimum $\beta$ in the sequence of left neighbors of $\alpha_0$. Hence, $\alpha$ belongs to the set $M$.

Let $\alpha' \in M$ and $a'$ an approximation of precision 3 to $\log\alpha'$. If $\alpha = \alpha'$, then we obviously have $|a' - a| < 1/4$ and $(1/\alpha')\mathrm{A} = (1/\alpha)\mathrm{A}$. On the other hand, assume that $\alpha \neq \alpha'$. If $k(\mathrm{A})$ exceeds 1, then the ideals $(1/\alpha_0)\mathrm{A} \neq (1/\alpha_1)\mathrm{A}$ and $(1/\alpha_2)\mathrm{A} \neq (1/\alpha_1)\mathrm{A}$. Also, $\log\alpha_2 - \log\alpha_0 > 1$, so $(1/\alpha')\mathrm{A} \neq (1/\alpha)\mathrm{A}$ or $|a - a'| > 1/4$. If $k(\mathrm{A}) = 1$, then by (5), $|\log\alpha' - \log\alpha| > 1/2$ and so $|a' - a| > 1/4$. $\square$

As an immediate consequence of Proposition 6.4 we obtain the following running time estimate.

**Proposition 6.11.** *BINARY is correct and has running time $O^*(1)$.*

**6.12. Solving the bounded equivalence problem.** Let $\Delta > 0$. The algorithm BOUNDED EQUIVALENCE is based on the following statement.

**Proposition 6.12.** *Let $\Delta \in \mathcal{D}$, $\Delta > 0$. Let $A, B$ be reduced ideals, $u \in \mathbb{Z}_{>0}$, $q = \max\{0, \lceil(\log\alpha - ((\log\Delta)/4+1))/\sqrt{u} - 1\rceil\}$. Let $\beta$ be a minimum in $A$ which is close to some $s$ with $q\sqrt{u} \le s < q\sqrt{u} + 1/4$. For $q = 0$ let $\beta = 1$. If there is a generator $\alpha$ of $A$ relative to $B$ with $0 < \log\alpha < u$, then for $\gamma = \alpha/\beta$ the inverse $1/\gamma$ is a minimum in $B$ and $0 < \log\gamma < \sqrt{u} + (\log\Delta)/2 + 2$. Also, $q < \sqrt{u}$.*

*Proof.* Let $\alpha$ be a generator of A relative to B with $\log\alpha < u$. Let $q$ be as above. Suppose that $q = 0$. Then $\gamma = \alpha$ and $\log\gamma = \log\alpha \le \sqrt{u} + (\log\Delta)/4 + 1$.

Now assume that $q > 0$. Then

$$(q-1)\sqrt{u} < \log\alpha - ((\log\Delta)/4 + 1) - \sqrt{u} \le q\sqrt{u}$$

and therefore

$$(\log\Delta)/4 + 1 < \log\alpha - q\sqrt{u} \le \sqrt{u} + (\log\Delta)/4 + 1. \tag{10}$$

Since $\log\alpha < u$, this implies that $q < \sqrt{u}$. By (7) we have

$$|\log\beta - q\sqrt{u}| < (\log\Delta)/4 + 1. \tag{11}$$

Since $\gamma = \alpha/\beta$, the inverse $1/\gamma$ is a minimum in B, and it follows from (10) and (11) that $0 < \log\gamma < \sqrt{u} + (\log\Delta)/2 + 2$. $\square$

BOUNDED EQUIVALENCE first determines all the numbers $\gamma$ such that $1/\gamma$ is a minimum in B and $\log\gamma$ is below $\sqrt{u}$ plus some error, which is of technical relevance. Next, the algorithm constructs minima in A close to $q\sqrt{u}$ for $q = 1, 2, \ldots$ until $\beta\gamma$ is a generator of A relative to B. All the minima are stored in binary representation. To multiply two minima, one can determine the sum of approximations of precision 4 to the log of the factors and then use BINARY to construct the product. In the algorithm, we use the procedure APP, which determines approximations to values of elementary functions.

**Algorithm 6.13.** (BOUNDED EQUIVALENCE)

**Input:** *$\Delta \in \mathcal{D}$, $\Delta > 0$, reduced ideals $A, B$ of $\mathcal{O}_\Delta$, $u \in \mathbb{N}$.*

**Output:** *The minimal generator $\alpha$ of $A$ relative to $B$, if it exists, and* $\log \alpha < u$.

| | | |
|---|---|---|
| $\gamma := 1;\ S := \emptyset;\ \alpha := 0;\ C := B;\ d = APP((\log \Delta)/2, 2)$ | | |
| *REPEAT* | | |
| | $\gamma := \gamma(L\ MIN(\Delta, C, 1))$ | |
| | $c := APPROX(\Delta, \gamma, 2)$ | |
| | $C := L\ IDEAL(\Delta, C)$ | |
| | $S := S \cup \{(C, \gamma)\}$ | |
| *UNTIL* $(c - d - 5/2)^2 \geq u$ *or* $C = A$ | | |
| *Sort* $S$ *according to first components* | | |
| $q := 0$ | | |
| *REPEAT* | | |
| | $s = APP(q\sqrt{u} + 1/8, 3)$ | |
| | $\beta := CLOSE(\Delta, A, s)$ | |
| | $C := (1/\beta)A$ | |
| | *IF* $(C, \gamma) \in S$ *for some* $\gamma$ | |
| | *THEN* | $\alpha := \beta\gamma$ |
| | *ELSE* | $q := q + 1;$ |
| *UNTIL* $q^2 \geq u$ *or* $\alpha \neq 0$ | | |

**Proposition 6.14.** *BOUNDED EQUIVALENCE is correct and requires time and space* $O^*(\sqrt{u})$.

*Proof.* Clearly, the algorithm terminates, and if the output is $\alpha \neq 0$, then $\alpha$ is a generator of A relative to B.

Suppose that there is a generator $\alpha$ of A relative to B and that $\log \alpha < u$. Let $q$ be as in Proposition 6.12. Then $q < \sqrt{u}$. Let $\beta$ be the minimum found in the second REPEAT loop, $\gamma = \alpha/\beta$. Then by Proposition 6.12 we have $0 < \log \gamma < \sqrt{u} + (\log \Delta)/2 + 2$ and therefore, $(c - d - 5/2)^2 < u$ for any approximation of precision 2 to $\log \gamma$. This implies that $S$ contains the pair $(\gamma \mathrm{B}, \gamma)$ and thus, the algorithm finds a generator of A relative to B.

Next we show that this generator is the minimal one. Let $\alpha' = \beta\gamma'$ be the generator found in the algorithm. Let $c'$ be the approximation of $\log \gamma'$ computed in the first REPEAT loop. Assume that $\alpha \neq \alpha'$. Since $(1/\alpha)\mathrm{A} = (1/\alpha')\mathrm{A} = \mathrm{B}$, it follows from the results of § 6.7 that $\log \alpha' - \log \alpha > 1/2$ and thus, $\log \gamma' - \log \gamma > 1/2$. This implies that $c < c'$ for every approximation $c$ of precision 2 to $\log \gamma$.

Hence, $S$ must contain the pairs $(\gamma\mathrm{B}, \gamma)$ and $(\gamma'\mathrm{B}, \gamma')$, and this is impossible since the first components of all pairs in $S$ are pairwise distinct.

Finally, we prove the running time estimate. The number of elements in $S$ is $O^*(\sqrt{u})$, so the time and space spent in that loop is $O^*(1)$. Since for the IF decision we can use binary search, the time and space required for the second REPEAT loop is also $O^*(\sqrt{u})$. $\square$

**6.13. Principal ideal testing.** Let B be a principal ideal of $\mathcal{O}_\Delta$. Let $\beta$ be the reducing number for B and let $\alpha$ be the minimal generator of $\mathrm{A} = (1/\beta)\mathrm{B}$ relative to $\mathcal{O}_\Delta$. Then $\alpha\beta$ is a generator of B and it is called the *minimal generator* of B. It is represented as the pair $(\alpha, \beta)$. That representation is of binary length $O^*(1)$.

**Algorithm 6.15.** (PRINCIPAL)

**Input:** $\Delta \in \mathcal{D}$, $B \in I_\Delta$.
**Output:** *If $B$ is principal, then the minimal generator $(\alpha, \beta)$ of $B$, else $(0, \beta)$ with some $\beta$.*

| | | | |
|---|---|---|---|
| $(A, \beta) := REDUCE(B);\ \alpha := 0$ | | | (1) |
| *IF* $\Delta < 0$ | | | (2) |
| *THEN* | *IF* $A = \mathcal{O}_\Delta$ | | (3) |
| | *THEN* | $\alpha := 1$ | (4) |
| | *ELSE* | $\alpha := 0$ | (5) |
| *ELSE* | $u := 1;\ \varepsilon := 0$ | | (6) |
| | *REPEAT* | | (7) |
| | | $\varepsilon :=$ *BOUNDED EQUIVALENCE*$(\Delta, A, A, u)$ | (8) |
| | | $\alpha :=$ *BOUNDED EQUIVALENCE*$(\Delta, A, B, u)$ | (9) |
| | | $u := 2u$ | (10) |
| | *UNTIL* $\alpha \neq 0$ *or* $\varepsilon \neq 0$ | | (11) |
| | *IF* $\varepsilon \neq 0$ *and* $\alpha = 0$ | | (12) |
| | *THEN* | $R =$ *APPROX*$(\Delta, \log\varepsilon, 4)$ | (13) |
| | | $\alpha :=$ *BOUNDED EQUIVALENCE*$(\Delta, A, B, R)$ | (14) |

**Proposition 6.16.** *For $\Delta < 0$, PRINCIPAL has polynomial running time.*

*Let $\Delta > 0$. If $B$ is a principal ideal, then PRINCIPAL finds its minimal generator $(\alpha, \beta)$ using time and space $O^*((\log|\alpha|)^{1/2})$. Otherwise, PRINCIPAL decides in time and space $O^*(R_\Delta^{1/2})$ that $B$ is not principal.*

*Proof.* If $u$ is such that $\mathrm{R}_\Delta < u$ and A is principal, then $\log\alpha < u$ for the minimal generator $\alpha$ of A. This proves the correctness. The assertions concerning time and space follow from § 6.5 and Proposition 6.14. □

**6.14. Computing the fundamental unit.** Let $\Delta > 0$; then the fundamental unit $\varepsilon$ of $\mathcal{O}_\Delta$ can be computed via $(\varepsilon, 1) :=$ PRINCIPAL$(\Delta, \mathcal{O}_\Delta)$.

**Proposition 6.17.** *For $\Delta > 0$ the fundamental unit of $\mathcal{O}_\Delta$ can be determined in time and space $O^*(R_\Delta^{1/2})$.*

**6.15. Solving norm equations.** Another application of PRINCIPAL is the solution of norm equations. We consider only the case of prime norms. By means of combinatorial techniques, the method can be extended to arbitrary norms.

**Algorithm 6.18.** (NORM EQUATION)

**Input**: $\Delta \in \mathcal{D}$, *and a prime number* $p$.
**Output**: *A number $\alpha$ in $\mathcal{O}_\Delta$ of norm $p$, if it exists, and $\alpha = 0$ otherwise.*

| | | |
|---|---|---|
| *IF* $\left(\frac{\Delta}{p}\right) = 1$ | | *(1)* |
| *THEN* | $\alpha :=$ *PRINCIPAL*$(\Delta,$ *PRIME IDEAL*$(\Delta, p))$ | *(2)* |
| *ELSE* | $\alpha := 0$ | *(3)* |

**Proposition 6.19.** *NORM EQUATION is correct. If there is a number of norm $p$ in $\mathcal{O}_\Delta$, then NORM EQUATION requires time and space $O^*(p^{1/2} + (\log|\alpha|)^{1/2})$, where $\alpha$ is the minimal such number with $|\alpha| > 1$. Otherwise, the algorithm requires time and space $O^*(p^{1/2} + R_\Delta^{1/2})$.*

**6.16. Solving the containment problem.** Let $T$ be a set of reduced ideals of $\mathcal{O}_\Delta$, and let A be a reduced ideal of $\mathcal{O}_\Delta$. We wish to find out whether A is equivalent to some $\mathrm{B} \in T$. If so, we wish to find the minimal generator of A relative to B.

For $\Delta < 0$ that problem was already solved in § 6.5. So assume that $\Delta > 0$. Since we might want to solve the containment problem for a fixed set $T$, but for many different reduced ideals A, we proceed in two steps. In a precomputation we determine the regulator of $\mathcal{O}_\Delta$ and we compute an *expansion* of $T$. As it was necessary in BOUNDED EQUIVALENCE to determine the set $S$ of pairs $(\mathrm{C}, \gamma)$, it is now necessary to determine a similar set for each element of $T$. The union of all these sets is the expansion $S$ of $T$.

**Algorithm 6.20.** (BABY STEPS)

**Input**: $\Delta \in \mathcal{D}$, *a set $T$ of reduced ideals.*
**Output**: *The expansion $S$ of $T$, $u = \lceil\sqrt{R_\Delta}\rceil$.*

```
S := {(C, 0) : C ∈ T}                                        (1)
IF  Δ > 0                                                    (2)
THEN  ε := PRINCIPAL(Δ, O_Δ); u := ⌈(log ε)^{1/2}⌉           (3)
      FOR  all C ∈ T                                         (4)
          γ := 1                                             (5)
          REPEAT                                             (6)
              γ := γ(L MIN(Δ, C, 1))                         (7)
              C := L IDEAL(Δ, C)                             (8)
              c := APPROX(Δ, γ, 4)                           (9)
              S := S ∪ {(C, c)}                              (10)
          UNTIL  c ≥ u + (log Δ)/2 + 1 or C = A              (11)
```

**Algorithm 6.21.** (CONTAINMENT)

**Input:** *$\Delta \in \mathcal{D}$, The expansion $S$ of set $T$ of reduced ideals of $\mathcal{O}_\Delta$, a reduced ideal $A$ of $\mathcal{O}_\Delta$.*

**Output:** *If $A$ is equivalent to some $B \in T$, then $(B, \alpha)$, where $\alpha$ is the minimal generator of $A$ relative to $B$, else $(A, 0)$.*

```
B := A; α := 0                                               (1)
IF  Δ < 0                                                    (2)
THEN  IF  (A, 0) ∈ S                                         (3)
      THEN  α := 1                                           (4)
      ELSE  IF  (ρ(A), 0) ∈ S                                (5)
            THEN  B := ρ(A); α := γ(A)                       (6)
ELSE  q := 0                                                 (7)
      REPEAT                                                 (8)
          β := CLOSE(Δ, A, qu)                               (9)
          C := (1/β)A                                        (10)
          IF  (C, c) ∈ S for some c                          (11)
          THEN  b := APPROX(Δ, β, 4)                         (12)
                α := BINARY(Δ, A, B, b + c)                  (13)
          ELSE  q := q + 1;                                  (14)
      UNTIL  q = u or α ≠ 0                                  (15)
```

**Proposition 6.22.** *BABY STEPS is correct. For $\Delta < 0$ it requires time and space $O^*(\sharp T)$. For $\Delta > 0$ it requires time and space $O^*(\sharp T(R_\Delta)^{1/2})$.*

*Proof.* Each individual operation in BABY STEPS can be carried out in polynomial time. For $\Delta < 0$ the number of operations is $O^*(\sharp T)$. For $\Delta > 0$ it follows from (3) that the number of operations is $O^*(\sharp T(\mathrm{R}_\Delta)^{1/2})$. □

**Proposition 6.23.** *CONTAINMENT is correct. For $\Delta < 0$ it requires polynomial time. For $\Delta > 0$ it requires time and space $O^*((R_\Delta)^{1/2})$.*

## 7. Orders and discrete logarithms in the class group

**Proposition 7.1.** *Any discrete logarithm problem in $Cl_\Delta$ can be solved in time and space $O^*(|\Delta|^{1/4})$.*

*Proof.* Let $\Delta < 0$. From [**14**], pp. 400-401, there follows $\mathrm{h}_\Delta = O^*(|\Delta|^{1/2})$.

Multiplication, inversion in the class group can be performed in polynomial time. By Proposition 6.2, containment tests in the class group for a set with $O^*((\mathrm{h}_\Delta)^{1/2})$ elements requires a precomputation which uses time and space $O^*(|\Delta|^{1/4})$. Each individual containment test can then be effected in polynomial time. Hence, the result follows from § 3.6.

Let $\Delta > 0$. We have $\mathrm{h}_\Delta \mathrm{R}_\Delta = O^*(\Delta^{1/2})$. Again, multiplication and inversion in the class group can be carried out in polynomial time. By Proposition 6.2, containment tests in the class group for a set with $O^*((\mathrm{h}_\Delta)^{1/2})$ elements requires a precomputation which uses time and space $O^*((\mathrm{h}_\Delta \mathrm{R}_\Delta)^{1/2}) = O^*(|\Delta|^{1/4})$. Each individual containment test can then be effected using time and space $O^*(\mathrm{R}_\Delta^{1/2})$. Hence, the result follows from § 3.6. □

## 8. Computing the class group

We now sketch an algorithm CLASS GROUP, which finds the class group in time $O^*(|\Delta|^{1/2})$. CLASS GROUP is based on an idea of H.W. Lenstra, Jr. [**13**]. The idea is the following. By a theorem of Minkowski, each ideal class of $\mathcal{O}_\Delta$ contains an ideal of norm $\leq c\sqrt{|\Delta|}$, where $c = 1/2$ for $\Delta > 0$ and $c = 2/\pi$ for $\Delta < 0$. Therefore, the ideals of prime norm $p \leq c\sqrt{|\Delta|}$ generate the class group. The best known upper bound for the time for determining that generating system is $O^*(|\Delta|^{3/4})$. Hence, we do not determine that system but we only write down the list $P$ of all prime numbers $p \leq c\sqrt{|\Delta|}$ such that there is an ideal of $\mathcal{O}_\Delta$ of norm $p$. We delete its first element $p$ and determine the ideal $\mathrm{A}_p$ of norm $p$. Then we compute all the elements in the subgroup $H$ generated by $[\mathrm{A}_p]$, and all the ideals of prime norm in $P$ which, when input to REDUCE, yield a reduced ideal in a class of $H$. These prime ideals cannot contribute anything new to the class group. These norms are deleted from $P$. Then we pick the first prime number $q$ in $P$ that has survived. We compute the elements in $H = \langle [\mathrm{A}_p], [\mathrm{A}_q] \rangle$ and again we eliminate in $P$ all norms of ideals which, when reduced, belong to a class of $H$. Iterating that process, we find the whole class group. Each time we determine a new prime ideal, the number of elements of $H$ grows by a factor $\geq 2$. Hence the total number of prime ideals that we must compute is only $O^*(1)$.

CLASS GROUP uses PRIME NORMS. On input of a reduced ideal A, the PRIME NORMS yields the set of all prime numbers $p$ such that $\mathrm{A} = \mathrm{REDUCE}(\mathrm{A}_p)$. It is based on the following statement.

**Proposition 8.1.** *Let $p$ be a prime number with $p < c\sqrt{|D|}$. If $\Delta > 0$, then $A_p$ is reduced. If $\Delta < 0$, then $A_p$ is reduced, or $A = \rho(A_p)$ is reduced and $b(A_p) = \sqrt{4a(A)p + \Delta}$.*

CLASS GROUP is a modification of BASIS. The details are left to the reader.

**Proposition 8.2.** *A generating system $G$ for $Cl_\Delta$ with $O^*(1)$ elements and its relation lattice $L(G)$ can be determined in time and space $O^*(\sqrt{|\Delta|})$.*

## References


1. J. Buchmann, *Zur Komplexität der Berechnung von Einheiten und Klassenzahlen algebraischer Zahlkörper*, Habilitationsschrift, Düsseldorf, 1987.
2. J. Buchmann, C. Thiel, H. C. Williams, *Short representation of quadratic integers*, to appear in Proceedings of CANT 1992.
3. J. Buchmann, S. Paulus, *Algorithms for finite abelian groups*, in preparation.
4. J. Buchmann, H. C. Williams, *On the existence of a short proof for the value of the class number and regulator of a real quadratic field*, Number Theory and Applications, in: A. Molin, NATO ASI Series, Kluwer Academic Press, 1989, pp. 327-345.
5. G. W. Fung, H. C. Williams, *Compact representation of the fundamental unit in a complex cubic field,* unpublished manuscript, 1991.
6. J. L. Hafner, K. S. McCurley, *Asymptotically fast triangularization of matrices over rings*, SIAM J. Comput. **20** (1991), pp. 1068-1083.
7. J. C. Lagarias, *Succinct certificates for the solvability of binary quadratic polynomials*, Proc. 20th IEEE Conference on Foundations of Computer Science, 1979, pp. 47-54.
8. J. C. Lagarias, *Worst-case complexity bounds in the theory of integral quadratic forms*, J. Algorithms **1** (1980), pp. 142-186.
9. J. C. Lagarias, *Succinct certificates for the solvability of binary quadratic Diophantine equations*, in preparation.
10. A.K. Lenstra, H.W. Lenstra, Jr., *Algorithms in number theory*, in J. van Leeuwen, *Handbook of theoretical computer science*, Elsevier Science Publishers B.V., 1990, pp. 673–715.
11. H. W. Lenstra, Jr., *On the computation of regulators and class numbers of quadratic fields*, Lond. Math. Soc. Lect. Note Ser. **56** (1982), pp. 123-150.
12. H. W. Lenstra, Jr., *Algorithms in algebraic number theory*, Bull. Amer. Math. Soc. (N.S) **26** (1992), pp. 211-244.
13. H. W. Lenstra, Jr., Private communication 1992.
14. W. Narkiewiecz, *Elementary and analytic theory of algebraic numbers*, Polish Scientific Publishers, Warszawa, 1974.
15. R.J. Schoof, *Quadratic fields and factorization*, in: H.W. Lenstra. Jr., R. Tijdeman, Computational methods in number theory, 1982, vol. 2, pp. 235–286.
16. D. Shanks, *Class Number, A Theory of Factorization and Genera*, Proc. Sympos. Pure Math. **20**, American Mathematical Society, 1970, pp. 415–440.

17. D. Shanks, *The infrastructure of a real quadratic field and its applications*, Proc. 1972 Number Theory Conference, Boulder, 1972, pp. 217-224.
18. V. Shoup, *On the deterministic complexity of factoring polynomials over finite fields*, Inform. Process. Lett., 1990, no. 33, pp. 261–267.
19. C. Thiel, *Short proofs using compact representations of algebraic integers*, to appear, 1993.

Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, Germany

*E-mail address*: ingi@cs.uni-sb.de

Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, Germany

*E-mail address*: buchmann@cs.uni-sb.de