Le Huu Phuoc January 18, 2018

CLASS GROUP METHOD FOR INTEGER FACTORIZATION

1 Preliminary

This chapter tends to recall some elementary background of algebraic number theory: quadratic forms, number fields, and class groups of an order. The main aim is to construct a group structure over the set of quadratic forms and the set of ideals and to point out a group isomorphism between them. [1]

1.1 Quadratic forms

Definition 1. A quadratic form is a function $q : \mathbb{Z}^2 \to \mathbb{Z}$ such that there exists a triplet of integers (a, b, c) verifying:

$$q(x, y) = ax^2 + bxy + cy^2,$$

for every $(x, y) \in \mathbb{Z}$. A form is called primitive if $\gcd(a, b, c) = 1$.

The quantity $D = b^2 - 4ac$ is called the discriminant of (a, b, c). We remark easily that if D is the discriminant of some quadratic form, then D is congruent to 0 or 1 modulus 4. An integer D is called a fundamental discriminant if every quadratic forms of discriminant D is primitive.

Proposition 1. An integer D is a fundamental discriminant if and only if either $D \equiv 1 \pmod 4$ and D is square-free or $D \equiv 0 \pmod 4$, $D/4 \equiv 2, 3 \pmod 4$ and $D/4$ is square-free.

Reciprocally, if D is congruent to 0 or 1 modulus 4, D is the discriminant of $x^2 - \frac{D}{4}y^2$ or $x^2 + xy + \frac{1-D}{4}y^2$, respectively. These types of quadratic forms are called principal forms of discriminant D. They will play the role of unit in the group structure that is going to be constructed later.

We now introduce the notion of equivalence between two forms. Consider the group action of $\{P \in M_2(\mathbb{Z}) \| \det(P) = \pm 1\}$[2] acting on the set of quadratic forms as follows.

Given $P = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in M_2(\mathbb{Z})$, the function $(x, y) \mapsto q(ax + by, cx + dy)$ is obviously a quadratic form. The image of action induced by P on q is hence defined by $P \cdot q = q(ax+by, cx+dy)$. Naturally, we obtain a quotient of the set of quadratic forms by this group action.

Two quadratic forms q and q' are said equivalent if and only if there exists a matrix $P \in A$ such that $q(P \cdot (x, y)) = q'(x, y)$, denote by $q \sim q'$. Moreover, by applying the same argument

[1] Most of contents in this part is extracted from the course "Algebraic number theory" given by Gaetan Chenevier, the book can be found at `http://gaetan.chenevier.perso.math.cnrs.fr/MAT552/TAN_poly_2017.pdf`

[2] This subset of $M_2(\mathbb{Z})$ is indeed a group, since $P^{-1} \in M_2(\mathbb{Z})$ too.

for $P \in SL_2(\mathbb{Z}) = \{M \in M_2(\mathbb{Z}) \| \det(M) = 1\}$, we can deduce another equivalence, called proper equivalence, and say that q and q' are properly equivalent.

With simple calculations with $P \in \left\{ \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} \right\}$, we verify some following elementary equivalences:

$$(a, b, c) \sim (a, -b, c)$$

$$(a, b, c) \overset{+}{\sim} (c - b, a) \overset{+}{\sim} (a, b + 2a, c + b + a) \overset{+}{\sim} (a, b - 2a, c - b + a)$$

These equivalences contain the idea of an algorithmic proof for Lagrange's reduction lemma below, which allows us to shift the general study of quadratic forms of fixed discriminant D to only some relatively small forms. Gauss strengthened the result by a theorem on the uniqueness of this reduction for negative discriminant forms.

Definition 2. A form (a, b, c) of negative discriminant is called reduced if $-a < b \leq a \leq c$ and in the case $a = c$, $b \geq 0$.

Lemma 1 (Lagrange's reduction)**.** Every form of non-square discriminant D is properly equivalent to a form (a, b, c) with $-|a| \leq b \leq |a| \leq |c|$. Such form satisfies $1 \leq |a| \leq \sqrt{\frac{|D|}{3}}$. As a consequence, there are only a finite number of equivalent classes of quadratic forms.

Theorem 1 (Gauss's reduction)**.** Every form of negative discriminant is properly equivalent to a unique reduced form.

We denote the set of properly equivalent classes of (primitive, respectively) quadratic forms of discriminant D by $Cl(D)$ ($P(D)$, respectively) and $h(D) = |P(D)|$. Besides the central study of quadratic forms, which is the classification of all equivalent classes for a given discriminant D, there are also other sub-problems, for instance, the study of numbers represented by a quadratic form, but this subject is not relevant to our goal, so it should be omitted.

To terminate our discussion on quadratic forms, we clarify the difference between equivalence and proper equivalence. We define the opposite form of $q = (a, b, c)$ by the form $q^{opp} = (a, -b, c)$, which is the image under action of $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$. Obviously, $(q^{opp})^{opp} = q$ and $q \overset{+}{\sim} q'$ induces $q^{opp} \overset{+}{\sim} (q')^{opp}$. It is not difficult to prove that the application $[q] \mapsto [q^{opp}]$ is an involution whose orbits are exactly the forms' equivalent classes. This application leads us to observe a special type of forms:

Definition 3. Ambiguous form is the form properly equivalent to its opposite form.

Theorem 2 (Gauss)**.** Suppose that $D < 0$. A reduced form (a, b, c) of discriminant D is ambiguous if and only if $b = 0$, $b = a$ or $c = a$.

1.2 Imaginary quadratic integers

Let α the complex number defined as $\alpha = \sqrt{D/4}$ if $D \equiv 0 \pmod 4$ or $\alpha = \frac{1+\sqrt{D}}{2}$ if $D \equiv 1 \pmod 4$ for a given negative integer D. We consider also the lattice $A_D \subset \mathbb{C}$ of base $(1, \alpha)$:

$$A_D = \mathbb{Z} + \mathbb{Z}\alpha$$

This lattice is moreover a sub-ring of $\mathbb{C}$, and, in fact, is equal to $\mathbb{Z}[\alpha]$. It can be endowed with the norm $N(z) = z\bar{z}$ of $\mathbb{C}$. More explicitly,

$$N(x + y\alpha) = \begin{cases} x^2 - \frac{D}{4}y^2 & \text{if } d \equiv 0 \pmod 4 \\ x^2 + xy + \frac{1-D}{4} & \text{if } d \equiv 1 \pmod 4 \end{cases}$$

Proposition 2. Let A_D be a ring of imaginary quadratic integers, we have these following properties:

1. Every element of A_D can be decomposed as product of irreducible elements (this decomposition is not required to be unique, as in general, A_D is not an UFD).
2. A_D is a Noetherian ring.
3. Every non-zero ideal I of A_D is of finite index, noted $N(I)$ (norm of I).

We can easily remark that the norm defined over A_D coincides with the principal form of discriminant D. A generalization of this observation proposed by Dedekind is as follows.

A $\mathbb{R}$-basis of $\mathbb{C}$ is called direct if its determinant in the base $(1, \alpha)$ is positive. Let I an ideal of A_D. We have that I is a lattice of $\mathbb{C}$, hence, it has a $\mathbb{Z}$-basis of two elements, say (u, v), which is also a $\mathbb{R}$-basis of $\mathbb{C}$. We consider then the application $q_{u,v} : \mathbb{Z}^2 \to \mathbb{Q}$ defined as $q_{u,v}(x, y) = \frac{1}{N(I)} N(xu + yv)$.

Proposition 3. 1. $q_{u,v}$ is an entirely positive quadratic form of discriminant D.

2. The equivalent class of $q_{u,v}$ does not depend on the choice of direct basis (u, v). [3]

The second statement permits us to denote by q_I the application $q_{u,v}$ for some direct basis (u, v) of ideal I.

1.3 Number fields and class groups

Definition 4. Number field is a sub-field of $\mathbb{C}$ which is also a finite-dimensional $\mathbb{Q}$-vector space. The number $d = \dim_{\mathbb{Q}} K$ is denoted by $[K : \mathbb{Q}]$ and called the degree of the number field K. Every number field is a finite extension of $\mathbb{Q}$ and is contained in $\bar{\mathbb{Q}}$.

We denote as usual $\bar{\mathbb{Q}}$ and $\bar{\mathbb{Z}}$ the algrebraic closure of $\mathbb{Q}$ and the ring of algebraic integers. As number fields can be considered as an extension of $\mathbb{Q}$, we generalize the notation of $\bar{\mathbb{Z}}$ for any number field K.

Definition 5. Let K be a number field. The ring of integers of K is $\mathcal{O}_K = K \cap \bar{\mathbb{Z}}$.

The following lemma points out that O_K is pretty large in K.

Lemma 2. For every element of K, there exists an integer m such that $mx \in \mathcal{O}_K$. Particularly, $\mathcal{O}_K$ generates K as $\mathbb{Q}$-vector space.

Proposition 4. Any $\mathcal{O}_K$ is an integrally closed ring.

Theorem 3 (Dedekind)**.** The additive group $\mathcal{O}_K$ possesses a $\mathbb{Z}$-basis of n elements.

Definition 6. An order R in K is a sub-ring of $\mathcal{O}_K$ which contains a $\mathbb{Q}$-basis of K of $n = \deg(K)$ elements.

[3] We will see later that it depends only on the class of I in $Cl(A_D)$.

Let R be an integral domain of quotient field K, a subset S of K is called a fractional ideal if there exist an ideal I in R and an element $x \in K$ satisfying $S = xI$. We say that two ideals I and J are equivalent if there exists an $x \in K^{\times}$ such that $xI = J$. This is, in fact, an equivalence relation whose set of equivalent classes is denoted by $Cl(R)$.

Proposition 5. An ideal I R if and only if I is a principal ideal. In particuliar, $Cl(R)$ has exactly one element if and only if R is PID.

The product of two ideals I and J is defined by the ideal IJ generated by elements of the form xy where $x \in I$ and $y \in J$.

Proposition 6. The multiplication of ideals induces a law of compostition over $Cl(R)$. This law is associative and commutative. The class of principal ideals plays the role of unit element.

Which is missing in the structure of $Cl(R)$ to be a group is the inversibility of its elements. To compensate this fact, we study a subset of special elements, these are invertible. This subset is called Picard group of R, noted $Pic(R)$.

Definition 7. Let I be a fractional ideal of R. We will say that I is invertible if there exists a fractional ideal J of R such that $R = IJ$. Such an ideal J will be called an inverse of I.

Theorem 4. Every equivalent class of $Cl(\mathcal{O}_K)$ is invertible. In other words, $Cl(\mathcal{O}_K) = Pic(\mathcal{O}_K)$.

Theorem 5. If R is an order of some number field K, $Cl(R)$ is finite.

The cornerstone of proof, which is based on the fact that each class associated to an integer and that the set of these integers is bounded, yields methods to compute the cardinal of $Cl(\mathcal{O}_K)$ and also to decompose the ideals of $\mathcal{O}_K$ (since we know that $\mathcal{O}_K$ is a Dedekind ring).

To study $\mathcal{O}_K$ and its orders using geometric approach, a view of $\mathcal{O}_K$ in the space $\mathbb{R}^{r_1} \times \mathbb{C}^{r_2}$ is introduced by using the embedding morphisms $\sum(K, \mathbb{R})$ and $\sum(K, \mathbb{C})$ on each co-ordinate. By this way, $\mathcal{O}_K$ and its orders behave as lattices, those corresponding to orders of larger discriminant are coarser (i.e, have bigger covolume).

In general, the classification of these finite extensions of $\mathbb{Q}$ is believed to be difficult. However, for some simple cases, we understand very well their structure, for example, we will discuss about some extensions of degree 2, which are called imaginary quadratic fields.

Proposition 7. If $d \in \mathbb{Z} \setminus \{0, 1\}$ square-free and $K = \mathbb{Q}(\sqrt{d})$. Then

$$\mathcal{O}_K = \begin{cases} \mathbb{Z} + \mathbb{Z}\sqrt{d} = A_{4d} & \text{if } d \equiv 2, 3 \pmod 4 \\ \mathbb{Z} + \mathbb{Z}\frac{1+\sqrt{d}}{2} = A_d & \text{if } d \equiv 1 \pmod 4 \end{cases}$$

The famous result below gives the relation between $Cl(A_D)$ (seen as an order of some number field) and $Cl(D)$.

Theorem 6 (Dedekind)**.** Let I be an ideal of A_D, $[I]$ be its equivalent class and q_I be the quadratic form associated to I. We have that the application:

$$\begin{aligned} \phi : \ Cl(A_D) &\to Cl(D) \\ [I] &\mapsto q_I \end{aligned}$$

is a bijection.

To prove this theorem, we need to construct the inverse application ψ. Let $q(x,y) = ax^2+bxy+cy^2$ a positive form of discriminant D. We have that

$$q(x,y) = a\left((x+\frac{b}{2a}y)^2 - \frac{D}{4a^2}y^2\right) = aN(x+\tau y),$$

where $\tau = \frac{b+\sqrt{D}}{2a}$. Notice that $a\tau = \frac{b+\sqrt{D}}{2} \in \alpha + \mathbb{Z}$. In particular, $a\tau \in A_D$. Hence, $\psi(q) = a\mathbb{Z} + a\tau\mathbb{Z}$. To finish the proof, we have the proposition below:

Proposition 8. 1. $\psi(q)$ is an ideal of A_D of norm a.

2. $q = q_{a,a\tau}$ (i.e, $(a, a\tau)$ forms a direct basis).

3. If $q_1 \overset{+}{\sim} q_2$, then $\psi(q_1) \sim \psi(q_2)$.

This bijection induces immediately a bijection between $Pic(A_D)$ and $P(D)$. Moreover, proved by Gauss and Dedekind, we can establish a group structure over these sets of equivalent classes as follows:

Definition 8. If q_1 and q_2 are two primitive forms of discriminant D, we define Gauss' composition of q_1 and q_2 by a form q_3 of the same discriminant such that $\phi(\psi(q_1)\cdot\psi(q_2)) = q_3$ (the product defined by the group isomorphism $\phi_{|Pic(A_D)}$).

Proposition 9. If q is primitive, $[q^{opp}] = [q]^{-1}$ in $P(D)$.

Theorem 7 (Gauss, Dedekind)**.** There is an unique law of abelian group over $P(D)$ such that the restricted application $\phi_{|Pic(A_D)} : [I] \mapsto q_I$ is a group isomorphism $Pic(A_D) \simeq P(D)$.

Proposition 10. The following statements are equivalent:

1. D is a fundamental discriminant.
2. A_D is the integer ring of $\mathbb{Q}(\sqrt{D})$.
3. Every non-zero ideals of A_D is invertible.

For this moment, we are able to equip $P(D)$ with a group structure over which we use the law of composition above and inverse elements defined by the opposite forms. The explicit computation of Gauss's composition will be discussed in Subsection 2.2.

2 Computation over class groups

The equivalence established throughout the precedent part between class groups and quadratic forms allows us to transport the computation over the ideals to quadratic forms, which is more realizable using computers. This operation introduced by Gauss in 1798 is called composition of quadratic forms. Also, since we will want to work with a class of forms, we will have a reduction procedure, which, given any quadratic form, will give us the unique reduced form in its class.

2.1 Reduction

By converting the method used in the proof of Lagrange's reduction, we get an algorithm for reducing quadratic forms. This algorithm can be seen as a variant of Euclid's algorithm.

Algorithm 1 Reduction of positive definite forms

```
1: procedure (Given a positive definite form $f = (a, b, c)$ of negative discriminant, output the unique
   reduced form equivalent to $f$.)
2:     if $-a < b \leq a$ then
3:         Go to line 10.
4:     Let $b = 2aq + r$ with $0 \leq r < 2a$.
5:     if $r > a$ then
6:         $r \leftarrow r - 2a$
7:         $q \leftarrow q + 1$
8:     $c \leftarrow c - \frac{1}{2}(b + r)q$
9:     $b \leftarrow r$
10:    if $a > c$ then
11:        $b \leftarrow -b$
12:        swap($a$,$c$)
13:        Go to line 4.
14:    else if $a = c$ and $b < 0$ then
15:        $b \leftarrow -b$
       return $(a, b, c)$
```

Proposition 11. The number of Euclidean steps in Algorithm 1 is at most equal to

$$2 + \left\lceil \lg\left(\frac{a}{\sqrt{|D|}}\right)\right\rceil$$

2.2 Group operations

Let (a_1, b_1, c_1) and (a_2, b_2, c_2) be two quadratic forms with the same discriminant D and consider the corresponding ideals:

$$I_k = a_k\mathbb{Z} + \frac{-b_k + \sqrt{D}}{2}\mathbb{Z}$$

given by the map ϕ_{FI}. We have the following lemma:

Lemma 3. Let I_1 and I_2 be two ideals as above, set $s = \dfrac{b_1 + b_2}{2}$, $d = \gcd(a_1, a_2, s)$ and let u, v, w be integers such that $ua_1 + va_2 + ws = d$. Then we have

$$I_1 \cdot I_2 = d\left(A\mathbb{Z} + \frac{-B + \sqrt{D}}{2}\mathbb{Z}\right)$$

where

$$A = d_0\frac{a_1a_2}{d^2},\ B = b_2 + \frac{2a_2}{d}(v(s - b_2) - wc_2)$$

and $d_0 = 1$ if at least one of the forms (a_1, b_1, c_1) or (a_2, b_2, c_2) is primitive and in general $d_0 = \gcd(a_1, a_2, s, c_1, c_2, n)$ where $n = \dfrac{b_1 - b_2}{2}$.

This gives us an explicit formula for the composition of two quadratic forms as follows. Note that the composition of two primitive forms is also primitive.

Theorem 8. Let $f_1 = (a_1, b_1, c_1)$ and $f_2 = (a_2, b_2, c_2)$ be two quadratic forms of the same discriminant D. Set $s = \dfrac{b_1 + b_2}{2}$, $n = \dfrac{b_1 - b_2}{2}$ and let u, v w and d be such that

$$ua_1 + va_2 + ws = d = \gcd(a_1, a_2, s)$$

and let $d_0 = \gcd(d, c_1, c_2, n)$. The composition of the two forms f_1 and f_2 is given by the formula

$$(a_3, b_3, c_3) = \left(d_0 \frac{a_1 a_2}{d^2}, b_2 + \frac{2a_2}{d}(v(s - b_2) - wc_2), \frac{b_3^2 - D}{4a_3} \right)$$

modulo the action of Γ_∞.

Although the raw formulas given in the theorem can be used directly, they can be improved by careful rearrangements. This leads to the following algorithm, due to a work published by Shanks in 1989, which was later modified by Atkin. Remark that squaring of a form is important and simpler, Atkin, therefore, gives two algorithms, one for duplication (NUDPL) and one for composition (NUCOMP).

Algorithm 2 PARTEUCL

```
1: procedure (This algorithm does an extended partial Euclidean algorithm on $a$ and $b$, but uses
   the variables $v$ and $v_2$ instead of $u$ and $v_1$.)
2:     $v \leftarrow 0$, $d \leftarrow a$, $v_2 \leftarrow 1$, $v_3 \leftarrow b$, $z \leftarrow 0$
3:     if $|v_3| > L$ then
4:         Go to line 9.
5:     else
6:         if $z$ is odd then
7:             $v_2 \leftarrow -v_2$ and $v_3 \leftarrow -v_3$.
8:             Terminate the algorithm.
9:     Let $d = qv_3 + t_3$ be the Euclidean division of $d$ by $v_3$ with $0 \leq t_3 < |v_3|$.
10:    $t_2 \leftarrow v - qv_2$, $v \leftarrow v_2$, $d \leftarrow v_3$, $v_2 \leftarrow t_2$, $v_3 \leftarrow t_3$, $z \leftarrow z + 1$
11:    Go to line 3.
```

Algorithm 3 NUDPL

```
1: procedure (Given a primitive positive definite form $f = (a, b, c)$ of discriminant $D$, output the
   square $f^2 = (a_2, b_2, c_2)$ of $f$. We assume that the constant $L = \lfloor |D/4|^{1/4} \rfloor$ has been precomputed.)
2:     Using Euclid's extended algorithm, compute $(u, v, d_1)$ such that $ub + va = a_1 = \gcd(b, a)$.
3:     $A \leftarrow a/d_1$, $B \leftarrow b/d_1$, $C \leftarrow (-cu \pmod{A})$, $C_1 \leftarrow A - C$ and if $C_1 < C$, set $C \leftarrow -C_1$.
4:     PARTEUCL($A$,$C$)
5:     if $z = 0$ then
6:         $g \leftarrow (Bv_3 + c)/d$, $a_2 \leftarrow d^2$, $c_2 \leftarrow v_3^2$, $b_2 \leftarrow b + (d + v_3)^2 - a_2 - c_2$, $c_2 \leftarrow c_2 + gd_1$
7:         Reduce the form $f_2 = (a_2, b_2, c_2)$.
8:         Output the result and terminate the algorithm.
9:     $e \leftarrow (cv + Bd)/A$, $g \leftarrow (ev_2 - B)/v$, $b_2 \leftarrow ev_2 + vg$.
10:    if $d_1 > 1$ then
11:        $b_2 \leftarrow d_1 b_2$, $v \leftarrow d_1 v$, $v_2 \leftarrow d_1 v_2$
12:    $a_2 \leftarrow d^2$, $c_2 \leftarrow v_3^2$, $b_2 \leftarrow b_2 + (d + v_3)^2 - a_2 - c_2$, $a_2 \leftarrow a_2 + ev$, $c_2 \leftarrow c_2 + gv_2$
13:    Reduce the form $f_2 = (a_2, b_2, c_2)$.
14:    Output the result and terminate the algorithm.
```

Algorithm 4 NUCOMP

```
1: procedure (Given two primitive positive definite forms $f_1 = (a_1, b_1, c_1)$ and $f^2 = (a_2, b_2, c_2)$ of
   the same discriminant $D$, output the product $f_3 = (a_3, b_3, c_3)$ of $f_1$ and $f_2$. We assume that the
   constant $L = \lfloor |D/4|^{1/4} \rfloor$ has been precomputed.)
2:     if $a_1 < a_2$ then
3:         swap($f_1$,$f_2$)
4:     $s \leftarrow \frac{1}{2}(b_1 + b_2)$
5:     $n \leftarrow b_2 - s$
6:     Using Euclid's extended algorithm, compute $(u, v, d)$ such that $ua_2 + va_1 = d = \gcd(a_1, a_2)$.
7:     if $d = 1$ then
8:         $A \leftarrow -un$
9:         $d_1 \leftarrow d$
10:        Go to line 21.
11:    if $d|s$ but $d \neq 1$ then
12:        $A \leftarrow -un$
13:        $d_1 \leftarrow d$
14:        $a_1 \leftarrow a_1/d_1$, $a_2 \leftarrow a_2/d_1$, $s \leftarrow s/d_1$
15:        Go to line 21.
16:    Using Euclid's extended algorithm, compute $(u_1, v_1, d_1)$ such that $u_1 s + v_1 d = d_1 = \gcd(s, d)$.
17:    if $d_1 > 1$ then
18:        $a_1 \leftarrow a_1/d_1$, $a_2 \leftarrow a_2/d_1$, $s \leftarrow s/d_1$, $d \leftarrow d/d_1$
19:    Compute $l \leftarrow -u_1(uc_1 + vc_2) \pmod{d}$
20:    $A \leftarrow -u(n/d) + l(a_1/d)$
21:    $A \leftarrow A \pmod{a_1}$
22:    $A_1 \leftarrow a_1 - A$
23:    if $A_1 < A$ then
24:        $A \leftarrow -A_1$
25:        PARTEUCL($a_1$,$A$)
26:    if $z = 0$ then
27:        $Q_1 \leftarrow a_2 v_3$, $Q_2 \leftarrow Q_1 + n$, $f \leftarrow Q_2/d$
28:        $g \leftarrow (v_3 s + c_2)/d$, $a_3 \leftarrow da_2$, $c_3 \leftarrow v_3 f + gd_1$, $b_3 \leftarrow 2Q_1 + b_2$
29:        Reduce the form $f_3 = (a_3, b_3, c_3)$.
30:        Output the result and terminate the algorithm.
31:    $b \leftarrow (a_2 d + nv)/a_1$, $Q_1 \leftarrow bv_3$, $Q_2 \leftarrow Q_1 + n$, $f \leftarrow Q_2/d$
32:    $e \leftarrow (sd + c_2 v)/a_1$, $Q_3 \leftarrow ev_2$, $Q_4 \leftarrow Q_3 - s$, $g \leftarrow Q_4/v$
33:    if $d_1 > 1$ then
34:        $v_2 \leftarrow d_1 v_2$, $v \leftarrow d_1 v$
35:    $a_3 \leftarrow db + ev$, $c_3 \leftarrow v_3 f + gv_2$, $b_3 \leftarrow Q_1 + Q_2 + d_1(Q_3 + Q_4)$
36:    Reduce the form $f_3 = (a_3, b_3, c_3)$.
37:    Output the result and terminate the algorithm.
```

3 Class group method in factorization

3.1 Rabin-Miller's compositeness test

The problem of decomposing a positive integer N into the product of its prime factors begins in fact with the problem of compositeness test. We introduce here an omnipresent test, due to Miller and Rabin. The running time is essentially the same as that the powering algorithm which is used, in principle $O(\ln^3 N)$. This algorithm is the workhorse of compositeness tests, and belongs in almost any number theory program. Note that it will prove the compositeness of essentially all numbers, but it will never prove their primality.

Definition 9. Let N be an odd positive integer, and a be an integer. Write $N-1 = 2^t q$ with q odd. We say that N is a strong pseudo-prime in base a if either $a^q \equiv 1 \pmod{N}$, or if there exists an e such that $0 \le e < t$ and $a^{2^e q} \equiv -1 \pmod{N}$.

If p is an odd prime, it is easy to see that p is a strong pseudo-prime in any base not divisible by p. Conversely, one can prove that if p is not prime, there exist less than $p/4$ bases a such that $1 < a < p$ for which p is a strong pseudo-prime in base a.

Algorithm 5 Rabin-Miller test

```
1:  procedure (Given an odd integer $N \ge 3$, this algorithm determines with high probability if $N$ is
    composite. If it fails, it will output a message saying that $N$ is probably prime.)
2:      $q \leftarrow N-1$
3:      $t \leftarrow 0$
4:      while $q$ is even do
5:          $q \leftarrow q/2$
6:          $t \leftarrow t+1$.
7:      $c \leftarrow 20$                                                    ▷ 20 rounds
8:      Choose a random number $1 < a < N$.
9:      $e \leftarrow 0$
10:     $b \leftarrow a^q \pmod{N}$
11:     if $b = 1$ then
12:         Go to line 18.
13:     while $b \not\equiv \pm 1 \pmod{N}$ and $e \le t-2$ do
14:         $b \leftarrow b^2 \pmod{N}$
15:         $e \leftarrow e+1$
16:     if $b \ne N-1$ then
17:         return $N$ is composite.
18:     $c \leftarrow c-1$
19:     if $c > 0$ then
20:         Go to line 8.
21:     else
22:         return $N$ is probably prime.
```

3.2 Shank's class group method

Shank's class group method is a $O(N^{\frac{1}{4}})$ factoring method. It is a simple by-product of the computation of the class number of an imaginary quadratic field. Indeed, let $D = -N$ if $N \equiv 3 \pmod 4$ and $D = -4N$ otherwise. If h is the class number of $\mathbb{Q}(\sqrt{D})$ and if N is composite, it is known by Gauss that h must be even. Hence, there must be an element of order exactly equal to 2 in the class group. Since this element is equivalent to its opposite form, it turns out to be a non-unit ambiguous form. Assume that we have found such a reduced ambiguous form, there are three following cases that can happen:

1. Either $b = 0$, hence $D = -4ac$ so $N = ac$.
2. Or $a = b$, hence $D = b(b - 4c)$, hence $N = (b/2)(2c - b/2)$ if b is even, $N = b(4c - b)$ if b is odd.
3. Or finally $a = c$, hence $D = (b - 2a)(b + 2a)$, hence N if b is even, $N = (2a - b)(2a + b)$ if b is odd.

We see that each ambiguous form gives a factorization of N (and this is one-to-one correspondence).

Hence, Shanks' factoring method is roughly as follows: after having computed the class number h, we look for an ambiguous form. Such a form will give a factorization of N (which may be trivial). There must exist a form which gives a non-trivial factorization however, and in practice it is obtained very quickly.

There remains the problem of finding ambiguous forms. Assume that $h = 2^t q$ with q odd. Take a form f at random and compute $g = f^q$. Then g is in the 2-Sylow subgroup of the class group, and if g is not the unit form. There exists an exponent m such that $0 \leq m < t$ and such that g^{2^m} is an ambiguous form.

3.3 The Schnorr-Lenstra factoring method

The class group method due to Schnorr-Lenstra [2] was the first sub-exponential method which required a negligible amount of space, say polynomial space. Although this method is quite attractive because of its running time, which is as good as all the other modern factoring algorithms, it was never widely used in practice. Indeed, the elliptic curve method for instance has the same characteristics as the present one as far as speed and storage are concerned, but the group operation on elliptic curves can be done faster than in class groups.

Definition 10. Given a positive bound B, an integer N is called B-smooth (B-powersmooth) if for every prime factor p of N, p ($p^{\mathrm{ord}_p(N)}$, respectively) is less than or equal to B.

The idea of the method is as follows. We have seen in Shank's method that the determination of the 2-Sylow subgroup of the class group of the quadratic field $\mathbb{Q}(\sqrt{-N})$ is equivalent to knowing all the factorizations of N. We consider the class numbers $h(-kN)$, which is unknown, for several values of k. Then, if $h(-kN)$ is smooth, we will be able to apply the $p-1$ method, replacing the group $\mathbb{F}_p^\times$ by the class group $\mathbb{Q}(\sqrt{-kN})$. Briefly, start with a random primitive form f, we compute a large power of f, say $\prod_{i=2} p_i^{\alpha_i}$. This will fail to return a non-trivial ambiguous form if $\mathrm{ord}(f)$ is odd, in this case, we can try another form. The difference is that, we will split N by using ambiguous forms instead of computing the gcd.

Since we will use $p - 1$ method, we need to specify the bounds B_1 and B_2 to find a $h(-KN)$ that is smooth for a value of K which is not too large. To choose these values appropriately, we need a fundamental theorem about smooth numbers. It is as follows:

Theorem 9 (Canfield-Erdos-Pomerance)**.** Let

$$\psi(x,y) = |\{n \le x, n \text{ is } y\text{-smooth}\}|.$$

Then if we set $u = \ln x / \ln y$, we have

$$\psi(x,y) = xu^{-u(1+o(1))}$$

uniformly for $x \to \infty$ if $(\ln x)^\epsilon < u < (\ln x)^{1-\epsilon}$ for a fixed $\epsilon \in (0,1)$.

In particular, if we set

$$L(x) = e^{\sqrt{\ln x \ln \ln x}},$$

then

$$\psi(x, L(x)^a) = xL(x)^{-1/2a+o(1)}.$$

Notice that there is little quantitative difference between B-smoothness and B-powersmoothness. Hence, it is not unreasonable to apply this theorem to estimate the behavior of powersmoothness of class numbers. In addition, the class number $h(-N)$ is $O(N^{1/2})$.

Hence, if we take $x = \sqrt{N}$ and $B = L(x)^a$, we expect that the probability that a given class number of size around x is B-powersmooth should be at least $L(x)^{-1/2a}$, hence, the expected number of values of K which we have to try before hitting a B-powersmooth number should be approximately $L(x)^{1/2a}$. Hence, ignoring step 2 of the $p-1$ algorithm, the expected running time with this choice of B is $O\left(L(x)^{a+1/2a}\right)$, and this is minimal for $a = 1/\sqrt{2}$. Since $L(x)^{1/\sqrt{2}} \approx L(N)^{1/2}$, we see that the optimal choice of B is approximately $L(N)^{1/2}$, and the expected running time is $L(N)$.

Algorithm 6 Schnorr-Lenstra's algorithm - Stage 1

```
 1: procedure (Let $N$ be a composite number. This algorithm will attempt to split $N$. We assume
    that we have precomputed a table $p[1]$, $p[2]$, ..., $p[k]$ of all primes up to $L(N)^{1/2}$.)
 2:     $B \leftarrow \lfloor L(N)^{1/2} \rfloor$
 3:     $K \leftarrow 1$
 4:     $e \leftarrow \lfloor \lg B \rfloor$.
 5:     $D \leftarrow -KN$ if $KN \equiv 3 \pmod 4$, $D \leftarrow -4KN$ otherwise.
 6:     Let $f_p$ be a random prime form of discriminant $D$. Set $x \leftarrow f_p$, $i \leftarrow 1$.
 7:     while $i < k$ do
 8:         $i \leftarrow i+1$, $q \leftarrow p[i]$, $q_1 \leftarrow q$, $l \leftarrow \lfloor B/q \rfloor$
 9:         while $q_1 \le l$ do
10:             $q_1 \leftarrow q_1 q$
11:         $x \leftarrow x^{q_1}$.
12:     $e_1 \leftarrow 0$
13:     while $x$ is not an ambiguous form and $e_1 < e$ do
14:         $x \leftarrow x^2$ and $e_1 \leftarrow e_1 + 1$
15:     if $x$ is not an ambiguous form then
16:         $K \leftarrow K+1$ and go to line 5.                          ▷ Try the next group
17:     else                                                  ▷ Here $x$ is an ambiguous form
18:         Find the factorization of $KN$ corresponding to $x$
19:         If this does not split $N$, go to line 6.                     ▷ Try another form
20:         Output a non-trivial factor of $N$ and terminate the algorithm.
```

Note that if in line 19, we obtain an ambiguous form which does not succeed in splitting N, this very probably still means that the K used is such that $h(-KN)$ is B-powersmooth. Therefore we must keep this value of K and try another random form in the group, but we should not change the group anymore. Note also that the first prime tried in line 8 is $p[2] = 3$, and not $p[1] = 2$.

As in $p-1$ method, a supplementary step is often added to ameliorate the efficiency in practical applications. By the end of Stage 1 for the form x of $Pic(-KN)$, if an ambiguous form is not found (maybe because $h(-KN)$ is not B_1-powersmooth), we hope that the class number could be still B_1-powersmooth except for a prime factor, which is between B_1 and B_2, which is specified before each execution. So, at that point, the algorithm outputs a form $f = x^{\prod_{i=1}^{k} p_i^{\lfloor \log_{p_i} B_1 \rfloor}}$. We continue to compute consecutively f^{q_j}, where, q_j's for $1 \leq j \leq l$ are all the primes between B_1 and B_2. If one of these forms is ambiguous, say j_0, we return to the form $g = x^{\prod_{i=2}^{k} p_i^{\lfloor \log_{p_i} B_1 \rfloor}}$, which should be stored during the execution of Stage 1, and compute $g^{2^i q_{j_0}}$ for $0 \leq i \leq \lfloor \log_2 B_1 \rfloor$. This procedure must lead to an ambiguous form, thereby move to the same termination step in Stage 1.

Remark that the q_j's are large integers and it costs non-trivial time to compute such larger powers. To compute $q[i]$-power efficiently, we[4] applied the following detail of implementation. First, we precompute and store an array of the difference of prime numbers between B_1 and B_2. Afterwards, we compute f^{p_k}, and f^{d_j} and, by multiplying $f^{p_k} \prod_{j=1}^{t} f^{d_j}$, obtain the form f^{q_t}. This replacement of the q_j-th power by much smaller d_j gains significantly in performance, notice that a single multiplication's cost is negligible in this case. Also, storing d_j requires less space than storing the whole number q_j.

4 Implementation

4.1 Overview

The implementation comprises of

- four `.c` files: `test.c`, `factor.c`, `form.c` `arithmetic.c`
- a header `project.h`
- and `makefile`

To compile, run `make`, an executable file `project` is automatically generated. To execute the code, run `make run`. To clean the file `project`, run `make clean`.

We use GMP 6.1.2[5] to manipulate large integers. Description of the functions implemented can be found below, notice that `L` always signifies $\mathtt{L} = \left\lfloor \sqrt[4]{\frac{|D|}{4}} \right\rfloor$.

- `void square_root_m(mpz_t d, mpz_t a, mpz_t p, mpz_t e, mpz_t q)`
  Compute a number d such that $\mathtt{d}^2 \equiv \mathtt{a} \pmod{)} p$, where $\mathtt{p} = 2^{\mathtt{e}}\mathtt{q}$, `q` odd.
- `void reduction(mpz_t a, mpz_t b, mpz_t c)`
  Reduce the form $(\mathtt{a}, \mathtt{b}, \mathtt{c})$ using Lagrange's reduction
- `void rand_form(mpz_t p, mpz_t D, mpz_t b, mpz_t c)`
  Return a prime form $(\mathtt{p}, \mathtt{b}, \mathtt{c})$ of discriminant `D`.

[4]Note that in Schnorr and Lenstra's paper, they introduced the use of Pollard-Brent Recursion.
[5]Documentation can be found at `https://gmplib.org/gmp-man-6.1.2.pdf`.

- `int is_ambiguous(mpz_t a,mpz_t b,mpz_t c)`
  Return 1 if the form $(\mathtt{a}, \mathtt{b}, \mathtt{c})$ is ambiguous, 0 otherwise.
- `void PARTEUCL(mpz_t a, mpz_t b, mpz_t v, mpz_t d, mpz_t v2, mpz_t v3, mpz_t z,mpz_t L)`
  Sub-algorithm *PARTEUCL*
- `void NUDPL(mpz_t res0,mpz_t res1,mpz_t res2,mpz_t a,mpz_t b,mpz_t c,mpz_t L)`
  Compute the form $(\mathtt{res0}, \mathtt{res1}, \mathtt{res2}) = (\mathtt{a}, \mathtt{b}, \mathtt{c})^2$.
- `void NUCOMP(mpz_t res0, mpz_t res1, mpz_t res2, mpz_t a1, mpz_t b1, mpz_t c1, mpz_t a2, mpz_t b2, mpz_t c2,mpz_t L)`
  Compute the form $(\mathtt{res0}, \mathtt{res1}, \mathtt{res2}) = (\mathtt{a1}, \mathtt{b1}, \mathtt{c1}) \cdot (\mathtt{a2}, \mathtt{b2}, \mathtt{c2})$.
- `int is_composite(mpz_t N)`
  Test whether `N` is composite using Miller-Rabin test.
- `void precompute(unsigned long long *T, unsigned long long limit1, unsigned long long *D, unsigned long long limit2)`
  Compute an array `T` of primes less than `limit1` and an array `D` of differences of consecutive primes between `limit1` and `limit2`.
- `void small_factors(mpz_t N, unsigned long long *primes)`
  Eliminate all the prime factors of `N` contained in array `primes`.
- `void form_pow(mpz_t res0, mpz_t res1, mpz_t res2, mpz_t a, mpz_t b, mpz_t c, mpz_t e, unsigned int p0, mpz_t p1, mpz_t L)`
  Compute $(\mathtt{res0}, \mathtt{res0}, \mathtt{res0}) = (\mathtt{a}, \mathtt{b}, \mathtt{c})^e$, where $(1, \mathtt{p0}, \mathtt{p1})$ is principal form of the same discriminant.
- `void form_pow_ui(mpz_t res0, mpz_t res1, mpz_t res2, mpz_t a, mpz_t b, mpz_t c,unsigned long long e, unsigned int p0, mpz_t p1, mpz_t L)`
  The same as `form_pow` but `e` is changed to type `unsigned integer`.
- `void factor(mpz_t N, mpz_t B, int e, unsigned long long *T, unsigned long long *T2)`
  Find a factor of `N` using Schnorr-Lenstra's method with precomputed arrays `T` and `T2`, the first bound `B` and $\mathtt{e} = \lfloor \lg \mathtt{B} \rfloor$.
- `void factor_N(mpz_t N, mpz_t B, int e, unsigned long long *primes, unsigned long long *differences)`
  Execute consecutively `int is_composite(mpz_t N)`, `small_factors` and `factor` to find a factor of `N`.

`PARTEUCL`, `NUDPL`, `NUCOMP` and Stage 1 of Schnorr-Lenstra's algorithm are implemented based on the details found in [1]. For the Stage 2, we follow the discussion at the end of subsection 3.3.

4.2 Testing

We choose $B_1 = 2 \cdot 10^5$, $B_2 = 10^6$ for testing. The test comprises of several tasks to split integers of the length $\approx 10^{356}$. Each of these integers is product of two primes between 10^{17}, which are generated using the command `random_prime(`10^{18}`,False,`10^{17}`)` in SageMath.

[6]This implementation can also factorize numbers of size $\approx 10^{40}$.

The test results:

N	K	Time (s)
47793138262670072284845714186648797	22	114
14676371866934859265634303380157069	1	1
57918280050916548002438727093800071	118	619
20272385402149233460989212981693441	1	19
12302017685950170306861393059534539	3	10
26293046435745083233231795895401999	10	45
42994765589808311593686560897367809	11	52
36006106371898705160949906015278549	19	111
16513300357759709230030559174164837	37	182
6939540551278162863255415530664619	7	33
75695099028024725429165658647490713	29	152
49888211251091813941793247369415783	14	65
36523697424255874565871901293127621	7	34
22239676399044557858771787547328233	1	2
58177392723630644275059805522729661	6	33
21586970088700686648421943458738001	30	149
5042603501081084143574841900026477	6	28
11101249196361947076898570862695613	3	12
15871575249193621292531853382219049	3	11
11366085872341607242270114215742189	19	92

We remark that it takes around 2.5 seconds per round for each Stage to be executed. The time taken to find a split of N now depends on the value of K, in general, $t \approx 5K$ (s). For $B_1 = 4 \cdot 10^5$ and $B_2 = 2 \cdot 10^6$, each round of Stage 1 or Stage 2 takes 7 seconds. For the larger size of N, these figures can vary slightly.

By increasing the bound, it is more likely that we can find an appropriate K sooner by accepting to lose performance at each round. According to the discussion in [1], the bound B_1 is optimally chosen as $L(N)^{1/2}$. However, we still keep $B_1 = 2 \cdot 10^5$ to make sure it pass all the tests with a reasonably small K.

References

[1] Henri Cohen. *A Course in Computational Algebraic Number Theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1993.

[2] C.P. Schnorr and H.W. Lenstra. A Monte Carlo factoring algorithm with linear storage. *Mathematics of computation*, 43(167):289–311, 7 1984.