# Binary quadratic forms

## 1 Introduction

In the last handout, we learned about the class group of an algebraic number field $K$ as the quotient group $J_K/P_K$, where $J_K$ is the group of fractional ideals of the ring of integers, $O_K$ of $K$, and $P_K$ is its subgroup of principal fractional ideals. While this introduced us to class groups and to one way that we can think about them, it wasn't clear how we might go about representing the objects of a class group or performing operations on them. To do that, we'll need to learn about a new thing: binary quadratic forms.

The two concepts, algebraic number fields and binary quadratic forms, are connected. But it's easier to represent and work with class groups in the context of binary quadratic forms, so this handout will explain how to do that.

## 2 Background

**Definition 2.1** (Binary quadratic form)**.** A binary quadratic form is

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

where $a, b, c \in \mathbb{R}$ and $a, b, c$ are not all equal to zero.

We write $f = (a, b, c)$ and call $f$ a "form". These are the objects that we'll be working with.

**Definition 2.2** (Integral form)**.** An integral form is a binary quadratic form where $a, b, c \in \mathbb{Z}$.

**Definition 2.3** (Content of a form)**.** The content of a form $f$ is $\gcd(a, b, c)$. It is denoted by $\text{cont}(f)$.

**Definition 2.4** (Primitive form)**.** A polynomial $f$ is primitive if $\text{cont}(f) = 1$.

## 2.1 Discriminant

**Definition 2.5** (Discriminant)**.** The discriminant of a form $f$ is $\Delta(f) = b^2 - 4ac$.

**Theorem 2.1.** *If $f$ is an integral form, then $\Delta(f) \in \mathbb{Z}$, and*

$$\Delta(f) \equiv 0 \ mod\ 4 \quad or \quad \Delta(f) \equiv 1 \ mod\ 4$$

**Theorem 2.2.** *For any integer $\Delta \in \mathbb{Z}$, such that*

$$\Delta \equiv 0 \ mod\ 4 \quad or \quad \Delta \equiv 1 \ mod\ 4,$$

*$\Delta$ is the discriminant of an integral binary quadratic form.*

**Theorem 2.3.** *If $f = (a, b, c)$ is an integral form, then*

$$b \equiv \Delta(f) \ mod\ 2$$

**Definition 2.6** (Fundamental discriminant)**.** For $\Delta \in \mathbb{Z}$, $\Delta$ is a fundamental discriminant if and only if:

1. $\Delta \equiv 1 \bmod 4$ and $\Delta$ is square-free[1], or
2. $\Delta \equiv 0 \bmod 4$, $\frac{\Delta}{4} \equiv 2, 3 \bmod 4$, and $\frac{\Delta}{4}$ is square-free

**Definition 2.7** (Positive definite binary quadratic form)**.** A binary form is called positive definite if for any pair $(x, y) \in \mathbb{R}^2$ with $(x, y) \neq (0, 0)$, the value $f(x, y)$ is positive, i.e. $f(x, y) > 0$.

**Definition 2.8** (Negative definite binary quadratic form)**.** A binary form is called negative definite if for any pair $(x, y) \in \mathbb{R}^2$ with $(x, y) \neq (0, 0)$, the value $f(x, y)$ is negative, i.e. $f(x, y) < 0$.

**Definition 2.9** (Positive semi-definite binary quadratic form)**.** A binary form is called positive semi-definite if for any pair $(x, y) \in \mathbb{R}^2$ with $(x, y) \neq (0, 0)$, the value $f(x, y)$ is non-negative, i.e. $f(x, y) \geq 0$.

**Definition 2.10** (Negative semi-definite binary quadratic form)**.** A binary form is called negative semi-definite if for any pair $(x, y) \in \mathbb{R}^2$ with $(x, y) \neq (0, 0)$, the value $f(x, y)$ is non-positive, i.e. $f(x, y) \leq 0$.

**Definition 2.11** (Indefinite binary quadratic form)**.** A binary form is called indefinite if for any pair $(x, y) \in \mathbb{R}^2$ with $(x, y) \neq (0, 0)$, the value $f(x, y)$ takes on both positive and negative values.

We mostly just care about positive definite forms.

**Theorem 2.4.** *A form $f$ is positive definite if and only if $\Delta(f) < 0$ and $a > 0$.*

[1] A square-free integer is one which isn't divisible by any perfect squares, i.e. there are no repeated factors in its prime decomposition.

### 2.1.1 A quick detour to build intuition about discriminants and also cause I just wanna

The quantity $b^2 - 4ac$ should look awfully familiar. You probably first saw it as the junk under the square root in the quadratic formula (which lets us solve quadratic equations that are too much of a pain to factor),

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Recall that the quadratic formula allows us find the two roots of a quadratic equation. If we think of the curve represented by that equation, the roots are the places where the curve touches the x-axis.

In the quadratic formula, what happens when $b^2 - 4ac = 0$? The square root reduces to zero, and we're left with $x = \frac{-b}{2a}$. Hence, there is only one (repeated) root. This is the case when the graph of the curve grazes the x-axis at just one point. Relative to binary quadratic forms, these discriminants correspond to the extreme case of the semi-definite binary quadratic forms mentioned above.[2]

When $b^2 - 4ac < 0$, however, the quantity under the square root is negative and the roots are complex. Therefore, graphically the curve has no roots on the real x-axis and therefore never crosses it. This gives a curve which is entirely positive or entirely negative. These cases analogously correspond to the definite binary quadratic forms.

Finally, when $b^2 - 4ac > 0$, two real roots exist, and the curve crosses the x-axis at two points, leading to at least one positive and at least one negative value along the curve. These discriminates are associated with indefinite binary quadratic forms.

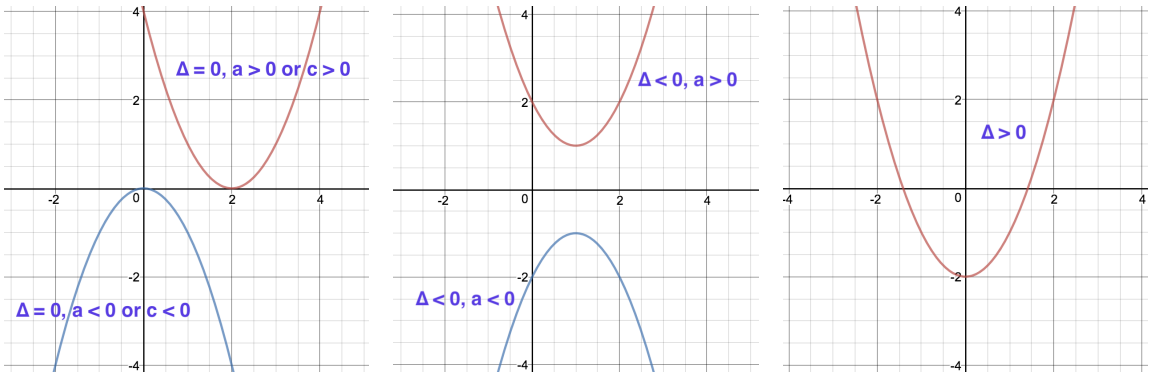


Figure 1: Pictures!

## 2.2 Matrix representations of forms

**Definition 2.12** (Matrix of a form)**.** The matrix of a form $f = (a, b, c)$ is

$$M(f) = \begin{bmatrix} a & b/2 \\ b/2 & c \end{bmatrix}$$

[2]Note that the positive and negative semi-definite quadratic forms also include the positive and negative definite forms, respectively.

with determinant $\det(M(f)) = ac - \frac{b^2}{4}$.

Using its matrix form, and defining $X = \begin{bmatrix} x & y \end{bmatrix}$, $f(x, y)$ can be written as

$$f(x, y) = X\ M(f)X^\top = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a & b/2 \\ b/2 & c \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = ax^2 + bxy + cy^2$$

and we have

$$\Delta(f) = -4\ \det(M(f)).$$

## 2.3 Representation of integers

**Definition 2.13** (Diophantine equation)**.** A Diophantine equation is

$$ax^2 + bxy + cy^2 = n$$

where $n \in \mathbb{Z}$ and $f = (a, b, c)$ is an integral binary quadratic form.

**Definition 2.14** (A representation of $n$ by $f$)**.** A solution $(x, y) \in \mathbb{Z}^2$ to a Diophantine equation for a given $f$ and $n$ is called a representation of $n$ by $f$.

**Definition 2.15** (A proper representation of $n$ by $f$)**.** A solution $(x, y) \in \mathbb{Z}^2$ to a Diophantine equation for a given $f$ and $n$, such that $\gcd(x, y) = 1$, is called a proper representation of $n$ by $f$.

Notice that we can restrict our considerations to primitive forms. This is because, if the content of $f$ is $k \neq 1$, i.e. $gcd(a, b, c) = k$ for some $k \neq 1$ such that, say, $a = kA$, $b = kB$, and $c = kC$, then we could factor out $k$ from the lefthand side of the Diophantine equation, which would mean that $n$, on the righthand side, is also divisible by $k$.

**Theorem 2.5.** *If $\Delta(f) < 0$ and if $n \in \mathbb{R}$, then the Diophantine equation $ax^2 + bxy + cy^2 = n$ has only finitely many solutions, and $x$ and $y$ are bounded by*

$$x^2 \leq \tfrac{4cn}{|\Delta|} \qquad \textit{and} \qquad y^2 \leq \tfrac{4an}{|\Delta|}$$

To find all of the solutions to a Diophantine equation where $\Delta(f) < 0$ and $n \in \mathbb{R}$, one could test for each solution $(x, y)$ that obeys the constraints given above.

**Example 2.1.** *Solve the Diophantine equation*

$$3x^2 + 2xy + 2y^2 = 28.$$

*Notice that*

$$\Delta(f) = b^2 - 4ac = 2^2 - 4(3)(2) = -20 < 0$$

*Therefore*

$$x^2 \leq \tfrac{4(2)(28)}{|-20|} = \tfrac{56}{5} = 11.2 \qquad \textit{and} \qquad y^2 \leq \tfrac{4(3)(28)}{|-20|} = \tfrac{84}{5} = 16.8$$

*And so*

$$|x| \leq 3 \qquad \textit{and} \qquad |y| \leq 4$$

*Testing all pairs of $(x, y)$ which satisfy those constraints, we find that the representations of 28 by $f = (3, 2, 2)$ are $(2, 2)$, $(-2, -2)$, $(-2, 4)$, and $(2, -4)$.* □

# 3 Equivalence

In this section, we'll learn what makes two forms equivalence. Note that if two forms are equivalent then they represent the same integers, although the inverse is not necessarily true. There are two types of equivalence: wide equivalence and proper equivalence.

**Definition 3.1** (Wide equivalence)**.** Two forms $f(x,y) = ax^2 + bxy + cy^2$ and $g(x,y) = Ax^2 + Bxy + Cy^2$ are widely equivalent if there is an invertible change of variables

$$x' = rx + sy, \quad y' = tx + uy \quad \text{with } r,s,t,u \in \mathbb{Z} \text{ and } ru - st = \pm 1$$

such that

$$a(x')^2 + bx'y' + c(y')^2 = Ax^2 + Bxy + Cy^2$$

or in other words, $g(x,y) = f(rx + sy, tx + uy)$.

**Definition 3.2** (Proper equivalence)**.** Two forms $f(x,y) = ax^2 + bxy + cy^2$ and $g(x,y) = Ax^2 + Bxy + Cy^2$ are properly equivalent if the wide equivalence conditions hold and also $ru - st = +1$.

Given the above definitions, a change of variables of a wide equivalence relation can be represented by a matrix[3]

$$T = \begin{bmatrix} r & s \\ t & u \end{bmatrix} \in GL(2,\mathbb{Z})$$

and a change of variables of a proper equivalence relation can be represented by the matrix

$$U = \begin{bmatrix} r & s \\ t & u \end{bmatrix} \in SL(2,\mathbb{Z})$$

In either case, we can equivalently write

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} r & s \\ t & u \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

to describe the change of variables. The invertibility requirement of the change of variables is what imposes the constraint that $ru - st = \pm 1$ since only those $2 \times 2$ matrices whose determinant is equal to $\pm 1$ are invertible.

As far as I can tell, we only care about proper equivalence, so from here on out my discussion is restricted to proper equivalence. What follows doesn't necessarily generalize to wide equivalence, and in particular to improper equivalence (where the matrix is in $GL(2,\mathbb{Z}) \setminus SL(2,\mathbb{Z})$).

Anyway, we can express form equivalency and the change of variables using the matrix representation of forms, i.e. $M(f) = \begin{bmatrix} a & b/2 \\ b/2 & c \end{bmatrix}$.

Suppose that two forms $f(x,y) = ax^2 + bxy + cy^2$ and $g(x,y) = Ax^2 + Bxy + Cy^2$ are properly equivalent. Then we have

[3] See the Appendix for definitions of $GL(2,\mathbb{Z})$ and $SL(2,\mathbb{Z})$.

$$
\begin{aligned}
g(x,y) &= \begin{bmatrix} x & y \end{bmatrix} M(g) \begin{bmatrix} x \\ y \end{bmatrix} \\
&= \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} A & B/2 \\ B/2 & C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\
&= \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} r & t \\ s & u \end{bmatrix} \begin{bmatrix} a & b/2 \\ b/2 & c \end{bmatrix} \begin{bmatrix} r & s \\ t & u \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\
&= \begin{bmatrix} x & y \end{bmatrix} \; U^\top \; M(f) \; U \begin{bmatrix} x \\ y \end{bmatrix}
\end{aligned}
$$

Therefore, $M(g) = U^\top \; M(f) \; U$.

Further, observe that $f(U(x,y)) = f(rx+sy, tx+uy) = a(rx+sy)^2 + b(rx+sy)(tx+uy) + c(tx+uy)^2$. For ease of notation, define $(fU)(x,y) = f(U(x,y))$, and note that $M(fU) = U^\top \; M(f) \; U$.

Then we can state a second definition for proper equivalence as the following.

**Definition 3.3** (Proper equivalence)**.** Two forms $f$ and $g$ are properly equivalent if $g = fU$ for some $U \in SL(2,\mathbb{Z})$. The $SL(2,\mathbb{Z})$-orbit of a form is called the proper equivalence class of that form.[4]

Notice that because $SL(2,\mathbb{Z})$ is a group (and associative under matrix multiplication), equivalence is transitive. To show this, take three forms $f(x,y)$, $g(x,y)$, and $h(x,y)$, and let $g = fU$ and $h = gV$, where $U, V \in SL(2,\mathbb{Z})$. Then

$$M(h) = V^\top \; M(g) \; V = V^\top \; U^\top \; M(f) \; U \; V = W^\top \; M(f) \; W$$

where $W = UV \in SL(2,\mathbb{Z})$ and $W^\top = V^\top U^\top \in SL(2,\mathbb{Z})$. Therefore $h = fW$, where $W \in SL(2,\mathbb{Z})$.

# 4 The class group

Now we can say a little more detail about the connection between ideal class groups (the kind we learned about before) and form class groups (the kind we're learning about now).

Take a negative discriminant $\Delta$ and let $F(\Delta)$ be the set of primitive positive definite binary quadratic forms $ax^2 + bxy + cy^2$ whose discriminant is $\Delta$. Define a proper equivalence relation (we'll call it $\sim$) on $F(\Delta)$ such that $f \sim g$ when $g = fU$ for $U \in SL(2,\mathbb{Z})$. This breaks up $F(\Delta)$ into a set of equivalence classes, $C(\Delta) = F(\Delta)/\sim$.

Now let's consider a quadratic extension field $K = \mathbb{Q}(\sqrt{d})$, where $d$ is square-free. If $d \equiv 1 \bmod 4$, then the discriminant of $K$ is $\Delta = d$. Otherwise, the discriminant of $K$ is $\Delta = 4d$. Let $O_K$ be the ring of integers of $K$, and let $J_K$ be the group of fractional ideals of the ring of integers $O_K$. $P_K$ is the subgroup of $J_K$ consisting of principal fractional ideals. Then we define the ideal class group of $K$ as the quotient group $Cl(K) = J_K/P_K$.[5]

[4] See Appendix for discussion on orbits. This sentence basically means that the proper equivalence class of a form is the set of all the forms you get when you compute $U^\top \; M(f) \; U$ for all of the $U \in SL(2,\mathbb{Z})$.

[5] See the previous handout for details on this stuff.

The connection between the above two types of class groups — and therefore between binary quadratic forms and ideals in quadratic fields — involves bijections between special versions of each type of class group, the details of which depend on whether or not the discriminant in question is positive or negative. For a real quadratic field $K = \mathbb{Q}(\sqrt{d})$ with discriminant $\Delta_K > 0$, there is a bijection between the narrow ideal class group[6] of $K$ and the form class group of primitive integral binary quadratic forms of discriminant $\Delta_K$. And for an imaginary quadratic field $K = \mathbb{Q}(\sqrt{d})$ with discriminant $\Delta_K < 0$, there is a bijection between the ideal class group of $K$ and the form class group of primitive positive-definite integral binary quadratic forms of discriminant $\Delta_K$.

# 5 Reduction

This section will introduce normal forms and reduced forms and give algorithms for normalization and reduction.

## 5.1 Normal forms

**Definition 5.1** (Normal form)**.** A form $f = (a, b, c)$ is called normal if $-a < b \leq a$.

**Definition 5.2** (Normalization operator)**.** We define the normalization operator $\eta(f) = \eta(a, b, c) = (a, b + 2ra, ar^2 + br + c)$, where $r = \left\lfloor \frac{a-b}{2a} \right\rfloor$.

### 5.1.1 Normalization algorithm

The normalization algorithm, given a form $f = (a, b, c)$, such that $\Delta < 0$ and $a > 0$:

1. Compute $r = \left\lfloor \frac{a-b}{2a} \right\rfloor$
2. Compute $\eta(f)$ and update $f = \eta(f)$
3. Return $f$, which is now normalized

Easy peasy.

**Example 5.1.** *Let's normalize the form $f = (11, 49, 55)$. Note that it is not normal because $b > a$.*

*First we compute $r$:*

$$r = \left\lfloor \frac{a-b}{2a} \right\rfloor = \left\lfloor \frac{11-49}{2(11)} \right\rfloor = \lfloor -1.\overline{72} \rfloor = -2$$

*Then we compute $\eta(f)$:*

$\eta(f) = \eta(a, b, c) = (a, b+2ra, ar^2+br+c) = (11, 49+2(-2)(11), 11(-2)^2+49(-2)+55) = (11, 5, 1)$

*And so the normalized version of $f = (11, 49, 55)$ is $f_{norm} = (11, 5, 1)$.* □

[6]The narrow ideal class group is such that instead of using $P_K$ in our class group quotient group, we use $P_K^+$, which is the group of totally positive principal fractional ideals of $K$.

## 5.2 Reduced forms

**Definition 5.3** (Reduced form)**.** A positive definite form $f = (a, b, c)$ is called reduced if it is normal and $a \leq c$, and if $a = c$ then $b \geq 0$.

Reduction of forms is important. For any given $\Delta < 0$, each proper equivalence class of binary quadratic forms of that discriminant contains a unique reduced representative. We can therefore know stuff like the class number of a given discriminant by studying only the reduced forms.

Further, for a reduced form and $\Delta < 0$, we have

$$\begin{aligned} |\Delta| &= 4ac - |b|^2 && (\Delta < 0) \\ &\geq 4a(a) - a^2 && (-a < b \leq a,\ a \leq c) \\ &\geq 3a^2 \end{aligned}$$

and so

$$a \leq \sqrt{\tfrac{|\Delta|}{3}}$$

Therefore, for a given $\Delta < 0$, there are finitely many $a$, and consequently there are finitely many $b$ and $c$.[7] This means that negative discriminants have a finite number of reduced forms and therefore have a finite number of equivalence classes. Cool.

**Theorem 5.1.** *Each primitive positive definite form $f$ is properly equivalent to a unique reduced form.*

**Definition 5.4** (Principal form of discriminant $\Delta$)**.** Let $\Delta$ be a negative integer such that $\Delta \equiv 0, 1 \bmod 4$, i.e. $\Delta$ is a negative discriminant. Let $k = \Delta \bmod 2$. Then

$$f = \left(1, k, \tfrac{k^2 - \Delta}{4}\right)$$

is the unique reduced form $(1, b, c)$ of discriminant $\Delta$. This particular reduced form is called the principal form of discriminant $\Delta$.

**Example 5.2.** *Here are examples of principal forms of several discriminants.*

| | | | |
|---|---|---|---|
| $\Delta = -3$ | $(1,1,1)$ | $\Delta = -4$ | $(1,0,1)$ |
| $\Delta = -7$ | $(1,1,2)$ | $\Delta = -8$ | $(1,0,2)$ |
| $\Delta = -11$ | $(1,1,3)$ | $\Delta = -15$ | $(1,1,4)$ |
| $\Delta = -19$ | $(1,1,5)$ | $\Delta = -20$ | $(1,0,5)$ |

□

**Definition 5.5** (Principal class of discriminant $\Delta$)**.** The equivalence class of the principal form of discriminant $\Delta$ is called the principal class of discriminant $\Delta$.

[7] Because $|b| \leq a$ and $c = \left(\frac{b^2 - \Delta}{4a}\right)$.

**Example 5.3.** *Let's consider the discriminant $\Delta = -23$. This discriminant has a total of three equivalence classes of positive definite binary quadratic forms. Therefore its class number is 3. In particular, the three unique reduced representatives of those equivalence classes are*

$$(1, 1, 6), \quad (2, -1, 3), \quad \textit{and} \quad (2, 1, 3).$$

*The principal form is $(1, 1, 6)$. Every primitive positive definite binary quadratic form with $\Delta = -23$ is properly equivalent to one of the three forms above, and composition in the group boils down to composition between these three forms.* □

**Definition 5.6** (Reduction operator)**.** Given a form $f = (a, b, c)$, the reduction operator is $\rho(f) = \rho(a, b, c) = (c, -b + 2sc, cs^2 - bs + a)$, where $s = \left\lfloor \frac{c+b}{2c} \right\rfloor$. Notice that $\rho(a, b, c)$ is equivalent to the normalization of $(c, -b, a)$.

### 5.2.1 Reduction algorithm

The reduction algorithm, given a form $f = (a, b, c)$ such that $\Delta < 0$ and $a > 0$:

1. Normalize $f$ and update $f = f_{\text{norm}}$
2. If $f$ is reduced, return $f$; if $f$ is not reduced, compute $s$ and $\rho(f)$ and update $f = \rho(f)$
3. Repeat step 2 until a reduced form is produced

**Example 5.4.** *Let's reduce the form $f = (11, 49, 55)$. We first normalize it, which we did in the previous example, finding that $f_{norm} = (11, 5, 1)$. The normalization example from Example 5.1 would be step 1 for the reduction algorithm.*

*But $f = f_{norm}$ is not yet reduced because $a > c$. We implement the reduction operator until we reach a reduced form.*

*Compute $s$:*

$$s = \left\lfloor \frac{c+b}{2c} \right\rfloor = \left\lfloor \frac{1+5}{2(1)} \right\rfloor = \lfloor 3 \rfloor = 3$$

*Compute $\rho(f)$:*

$$\rho(f) = (c, -b + 2sc, cs^2 - bs + a) = (1, -5 + 2(3)(1), 1(3)^2 - 5(3) + 11)) = (1, 1, 5)$$

*This form is reduced and so $(1, 1, 5)$ is the reduced form of $(11, 49, 55)$.* □

Here's an example. In this and in the two later composition algorithms, please excuse my Python.

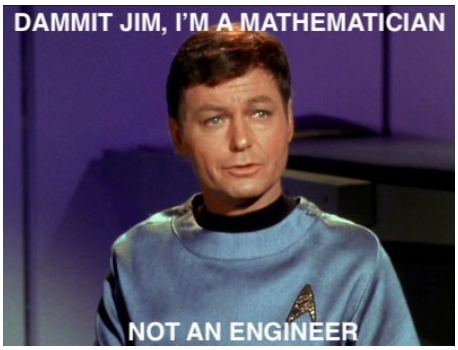


Figure 2: Just kidding (kinda).

---

```
import math

# inputs are the form's coefficients, e.g. 195751 x**2 + 1212121 xy + 1876411 y**2
a = 195751
b = 1212121
c = 1876411

f = (a, b, c)

# find the normalized representation of f
def normalize(a, b, c):
    if b > -a and b <= a:
        return (a, b, c)
    else:
        while b <= -a or b > a:
            aa = a
            bb = b
            cc = c
            r = math.floor( (a - b) / (2 * a) )
            a = aa
            b = bb + 2 * r * aa
            c = aa * r**2 + bb * r + cc
        return (a, b, c)

f_normal = normalize(a, b, c)

# update a, b, c
a = f_normal[0]
b = f_normal[1]
c = f_normal[2]

print("The normalized form of", f, "is", f_normal, ".")
```

```
# find a reduced representation of f that is properly equivalent to f
def reduce(a, b, c):
    if (a < c) or (a == c and b >= 0):
        return (a, b, c)
    else:
        while (a > c) or (a == c and b < 0):
            aa = a
            bb = b
            cc = c
            s = math.floor( (c + b) / (2 * c) )
            a = cc
            b = -bb + 2 * s * cc
            c = cc * s**2 - bb * s + aa
        return (a, b, c)

f_reduced = reduce(a, b, c)

print("The reduced form of", f, "is", f_reduced, ".")
```

---

Running the above yields

```
The normalized form of (195751, 1212121, 1876411) is (195751, 37615, 1807) .
The reduced form of (195751, 1212121, 1876411) is (1, 1, 1) .
```

Notice that the discriminant of all three forms — $f$, $f_{norm}$, and $f_{red}$ — is $-3$.

# 6 Composition

Composition of binary quadratic forms is notoriously technical and complicated, and I'm still fully wrapping my head around it. There's a very nice and reasonable algorithm that I'll present momentarily, but if you want to know more about the details of composition I can try to sludge through it for you. Here's a quote from a text about it that makes me feel better:

> [C]omposition of forms is filled with all kinds of technicalities. I certainly was in good company with my attitudes on composition; Gordon Pall starts his article on Gauss composition with the following words, "At least two recent writers have described Gauss's theory of composition of binary quadratic forms as a tour de force, and not a few mathematicians have told me it was much too complicated." Dan Shanks has made similar experiences: he remarks that, "It was frequently said that composition is 'difficult', sometimes even 'very difficult'," and observes that "many number-theorists seem to have a real fear of composition; we might call it compophobia. They are uncomfortable until they can translate it into ideals, continued fractions, or some other formalism that they feel they understand better."

## 6.1 Composing two forms

An important note is that we only perform compositions between forms which have the same discriminant. This makes sense. We're operating within the class group of a particular discriminant, so although the objects we're operating on are integer 3-tuples, $(a, b, c)$, which represent binary quadratic forms, the more abstract objects that we're operating on are the equivalence classes of the class group, like how when we add $\frac{20}{12}$ and $\frac{5}{35}$, what we're working with are the fraction equivalence classes whose "reduced representatives" are $\frac{5}{3}$ and $\frac{1}{7}$.

## 6.2 Composition algorithm

Shanks, one of the folks in the above quote who remarked on the difficulty of quadratic binary form composition, helpfully developed an algorithm for composition that was further developed by Cohen and then by Jacobson and van der Poorten. This algorithm has the benefit of including partial reduction during the composition. When I tested their algorithm in Python, the resulting form wasn't entirely normalized/reduced, so at the end I pulled in the reduction code that I gave in Section 5.3.

---

```
import math
from reduce import *

# This produces the composition f1 * f2 = f

# f1 = (u1, v1, w1)
u1 = 12
v1 = 11
w1 = 3

# f2 = (u2, v2, w2)
u2 = 93
v2 = 109
w2 = 32

# calculate the discriminant
disc = v1**2 - 4*u1*w1

print("The discriminant of f1 and f2 is", disc)

# precompute L
L = (abs(disc))**(0.25)

# swap f1 and f2 if w1 < w2
if w1 < w2:
    u1, v1, w1, u2, v2, w2 = u2, v2, w2, u1, v1, w1
else:
    pass

s = 0.5 * (v1 + v2)
```

```
m = v2 - s

# extended Euclidean algorithm
def exteucl(a,b):
    x,y = 0,1
    u,v = 1,0
    a_copy = a
    b_copy = b
    while a_copy != 0:
        q = b_copy // a_copy
        r = b_copy % a_copy
        m = x - u*q
        n = y - v*q
        # now update variables
        b_copy = a_copy
        a_copy = r
        x,y = u,v
        u,v = m,n
    gcd = b_copy
    return(gcd,x,y)

e1 = exteucl(u2, u1)
gcdu1u2 = e1[0]
b = e1[1]
c = e1[2]

# check to see if gcd of u1 and u2 divides s and...
# ...do some stuff in either case
def checkdivis(gcdu1u2, s):
    if (s % gcdu1u2) == 0:
        G = gcdu1u2
        Ax = G
        Bx = m * b
        By = u1 / G
        Cy = u2 / G
        Dy = s / G
    else:
        e2 = exteucl(gcdu1u2, s)
        gcdFs = e2[0]
        x = e2[1]
        y = e2[2]
        G = gcdFs
        H = gcdu1u2 / G
        By = u1 / G
        Cy = u2 / G
        Dy = s / G
        l = y * (b * (w1 % H) + c * (w2 % H)) % H
        Bx = b * (m / H) + l * (By / H)
    return G, Ax, Bx, By, Cy, Dy

checkdiv = checkdivis(gcdu1u2, s)
```

```
G = checkdiv[0]
Ax = checkdiv[1]
Bx = checkdiv[2]
By = checkdiv[3]
Cy = checkdiv[4]
Dy = checkdiv[5]

bx = Bx % By

by = By

x = 1
y = 0
z = 0

# extended partial Euclidean algorithm plus some stuff
def partialeucl(bx,by):
    global x
    global y
    global z
    while abs(by) > L and bx != 0:
        q = math.floor(by / bx)
        t = by % bx
        by = bx
        bx = t
        t = y - q * x
        y = x
        x = t
        z += 1
    if z % 2 == 1:
        by = -by
        y = -y
        ax = G * x
        ay = G * y
    elif z % 2 == 0:
        ax = G * x
        ay = G * y
    return ax, ay, bx, by, x, y, z

pebxby = partialeucl(bx,by)

ax = pebxby[0]
ay = pebxby[1]
bx = pebxby[2]
by = pebxby[3]
x = pebxby[4]
y = pebxby[5]
z = pebxby[6]

if z == 0:
```

```
    Q1 = Cy * bx
    cx = (Q1 - m) / By
    dx = (bx * Dy - w2) / By
    u3 = by * Cy
    w3 = bx * cx - G * dx
    v3 = v2 - 2 * Q1
else:
    cx = (Cy * bx - m * x) / By
    Q1 = by * cx
    Q2 = Q1 + m
    dx = (Dy * bx - w2 * x) / By
    Q3 = y * dx
    Q4 = Q3 + Dy
    dy = Q4 / x
    if bx != 0:
        cy = Q2 / bx
    else:
        cy = (cx * dy - w1) / dx
    u3 = by * cy - ay * dy
    w3 = bx * cx - ax * dx
    v3 = G * (Q3 + Q4) - Q1 - Q2

f = (int(u3), int(v3), int(w3))

print("f1 * f2 = f =", f)

fnorm = normalize(int(u3), int(v3), int(w3))

fnorm_a = fnorm[0]
fnorm_b = fnorm[1]
fnorm_c = fnorm[2]

print("Normalizing f gives", fnorm)

fred = reduce(fnorm_a, fnorm_b, fnorm_c)

print("The reduced form of f is", fred)
```

---

Running the above with $(12, 11, 3)$ and $(93, 109, 32)$, both of which have discriminant $-23$ gives

```
The discriminant of f1 and f2 is -23
f1 * f2 = f = (1, -15, 62)
Normalizing f gives (1, 1, 6)
The reduced form of f is (1, 1, 6)
```

Notice[8] that the reduced form of $(12, 11, 3)$ is $(2, -1, 3)$ and the reduced form of $(93, 109, 32)$ is $(2, 1, 3)$. Along with $(1, 1, 6)$, these three forms make up the unique reduced representatives of the equivalence classes of the discriminant $-23$ that we found in Example 5.3.

[8] By "notice" I mean you can run these through the reduction algorithm.

## 6.3 Squaring a form

There's a special algorithm for squaring a binary quadratic form that's a special case of the composition algorithm. This one was also developed by Shanks and then Cohen and then Jacobson and van der Poorten.

### 6.3.1 Squaring algorithm

Here's the squaring algorithm in Python.

---

```
import math
from reduce import *

# This produces the composition f1 * f1 = f

# f1 = (u1, v1, w1)
u = 35
v = 23
w = 99

# calculate the discriminant
disc = v**2 - 4*u*w

print("The discriminant of f1 is", disc)

# precompute L
L = (abs(disc))**(0.25)

# extended Euclidean algorithm
def exteucl(a,b):
    x,y = 0,1
    u,v = 1,0
    a_copy = a
    b_copy = b
    while a_copy != 0:
        q = b_copy // a_copy
        r = b_copy % a_copy
        m = x - u*q
        n = y - v*q
        # now update variables
        b_copy = a_copy
        a_copy = r
        x,y = u,v
        u,v = m,n
    gcd = b_copy
    return(gcd,x,y)

e1 = exteucl(u, v)
```

```
gcduv = e1[0]
x = e1[1]
y = e1[2]

By = u / gcduv
Dy = v / gcduv

bx = (y * w) % By
by = By

x = 1
y = 0
z = 0

# extended partial Euclidean algorithm plus some stuff
def partialeucl(bx,by):
    global x
    global y
    global z
    while abs(by) > L and bx != 0:
        q = math.floor(by / bx)
        t = by % bx
        by = bx
        bx = t
        t = y - q * x
        y = x
        x = t
        z += 1
    if z % 2 == 1:
        by = -by
        y = -y
        ax = gcduv * x
        ay = gcduv * y
    else:
        ax = gcduv * x
        ay = gcduv * y
    return ax, ay, bx, by, x, y, z

pebxby = partialeucl(bx, by)

ax = pebxby[0]
ay = pebxby[1]
bx = pebxby[2]
by = pebxby[3]
x = pebxby[4]
y = pebxby[5]
z = pebxby[6]

if z == 0:
    dx = (bx * Dy - w) / By
    u3 = by**2
```

```
    w3 = bx**2
    v3 = v - (bx + by)**2 + u3 + w3
    w3 = w3 - gcduv * dx
else:
    dx = (bx * Dy - w * x) / By
    Q1 = dx * y
    dy = Q1 + Dy
    v3 = gcduv * (dy + Q1)
    dy = dy / x
    u3 = by**2
    w3 = bx**2
    v3 = v3 - (bx + by)**2 + u3 + w3
    u3 = u3 - ay * dy
    w3 = w3 - ax * dx

f = (int(u3), int(v3), int(w3))

print("f1 * f1 = f =", f)

fnorm = normalize(int(u3), int(v3), int(w3))

fnorm_a = fnorm[0]
fnorm_b = fnorm[1]
fnorm_c = fnorm[2]

print("Normalizing f gives", fnorm)

fred = reduce(fnorm_a, fnorm_b, fnorm_c)

print("The reduced form of f is", fred)
```

---

Running the above with $(12, 11, 3)$ gives

```
The discriminant of f1 is -23
f1 * f1 = f = (12, 13, 4)
Normalizing f gives (12, -11, 3)
The reduced form of f is (2, 1, 3)
```

Running the above with $(2, -1, 3)$ (the reduced form of (12, 11, 3), the previous example) gives the same reduced result:

```
The discriminant of f1 is -23
f1 * f1 = f = (4, -5, 3)
Normalizing f gives (4, 3, 2)
The reduced form of f is (2, 1, 3)
```

It shouldn't be surprising that running the algorithm with a form and with its reduced representation both yield the same reduced result.

# 7 Moral of the story

- The objects we're playing with look like $(a, b, c)$, and they represent the coefficients of binary quadratic forms.
- We can compose these 3-tuples. That's the operation we'll be doing. (I think in particular we're squaring them.)
- Reduction is a blessing and you should love it. Reduce your shit for the sake of efficiency if you deem it efficient. When you compose forms, reduce the result. Reduce, reuse, recycle.
- We need to pick a cool class group. Help me help you figure out which one.

# 8 Appendix

## 8.1 Special matrices

$GL(2,\mathbb{Z})$ is the set of all invertible $2 \times 2$ matrices with integer entries whose determinants are equal to $\pm 1$.

$SL(2,\mathbb{Z})$ is a subgroup of $GL(2,\mathbb{Z})$ defined as the set of all $2 \times 2$ matrices with integer entries whose determinants are equal to 1. The two generating matrices of $SL(2,\mathbb{Z})$ are

$$S = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \quad \text{and} \quad T = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

For our purposes, we mostly just care about $SL(2,\mathbb{Z})$. Note that this set of matrices is a group and so for $U \in SL(2,\mathbb{Z})$ and $V \in SL(2,\mathbb{Z})$, $UV \in SL(2,\mathbb{Z})$.

## 8.2 Orbits

Let $S$ be a non-empty set and let $G$ be a group. A "left action" of $G$ on $S$ is a mapping

$$G \times S \to S, \quad (g,s) \mapsto gs \in S$$

which has the following properties:

- if $1_G s = s$ for all $s \in S$
- for $g, h \in G$ and $s \in S$, $g(hs) = (gh)s$

Given a left action of $G$ on $S$, two elements $s$ and $t$ in $S$ are equivalent if there's an element $g \in G$ such that $t = gs$. This is an equivalence relation on $S$, and the equivalence class $\{gs : g \in G\}$ is called the $G$-orbit of $s$.

We have an analogous definition for a "right action" of a group $G$ on a set $S$. Given a right action of $G$ on $S$ we define the $G$-orbit of $s$ in the same way as with left actions.

Orbits should remind you a bit of the ring ideals from our previous class groups handout.