Analysis and implementation of algorithms in number theory

Preliminary Arizona Winter School, 2025

Juanita Duque-Rosero

Department of Mathematics and Statistics, Boston University

juanita@bu.edu

Draft last updated: November 9, 2025

Introduction

These lecture notes accompany the lectures for the Preliminary Arizona Winter School 2025: Algorithms in number theory. The main references I used are [Coh93, Har21]. Other useful references are [Coh00, Ste, Voi21, vzGG13].

The course consists of an exploration of the algorithms and computational ideas that power modern algebra and number theory. We will start with the basics: analyzing what makes an algorithm *efficient*, and working through classic methods in integer arithmetic and linear algebra. These techniques will come up again and again in the rest of the lectures.

From there, we will study algebraic numbers and number fields. We will see how to represent them and do arithmetic with them. We then move on to working with rings of integers, discriminants, and integral bases. Then, we go back to algorithmic linear algebra to look at the LLL Algorithm and its applications to the study of number fields. Finally, we study ideals, class groups, and units and we pull everything together with examples from imaginary quadratic fields.

Explicit computations

The ideal way to follow these notes is to try examples in your favorite computer algebra system. I personally use Magma, but you are welcome to use whatever you prefer (SageMath, PARI/GP, Oscar, etc.). You can find Magma examples here. Also, here is a list of random tricks that I have compiled and here is a scavenger hunt to get you started.

Prerequisites

This course will assume fluency with algebra at a beginning graduate level and familiarity with the basic objects of algebraic number theory (such as number fields and their rings of integers). Some good references are [Mil, Lan94].

A note from the author

Despite my best efforts, these notes will contain typos. If you spot any, please feel free to email me, I appreciate your help!

An (irrelevant) note from the author

When I think about foundational algorithms that have really made a difference in my own number theory research, *Gröbner bases* are on top of my list. They are very useful for arithmetic geometry, and they have “saved” my work more than once. Unfortunately, I could not find space in these lecture notes to cover them. If you are curious, the book [CLO15] is beautifully written, and has a lot of the relevant theory.

Acknowledgments

I thank John Voight for insightful discussions while I was preparing these notes. Also, thank you to David Harvey for his helpful feedback on an earlier draft. Thanks to Jennifer Balakrishnan, Jerson Caro, Jose Cruz, Fatemeh Jalalvand, Aashraya Jha, and Padmavathi Srinivasan for their valuable input. I am grateful to Hugues Bellemare for following these notes carefully and providing feedback. Finally, thanks to the problem session leaders Thomas Bouchet, Kate Finnerty, Asimina S. Hamakiotes, and Yongyuan Huang for their hard work, careful reading, and feedback.

Contents

Lecture 1: Arithmetic and linear algebra

Even when you are doing advanced computation, you will end up using basic algorithms in arithmetic and linear algebra. In this first lecture, we explore some of those algorithms to compute basic arithmetic, greatest common divisors, and matrices normal forms. They will be useful in the rest of the course. This lecture follows [Har21, §2] and [vzGG13, Chapter 2]. Other useful references are [BZ11] and [Coh93].

1.1 Analysis of algorithms

When analyzing the effectiveness of an algorithm, we can consider many factors, such as the amount of memory used, or the number of operations required for completion. We start by introducing some useful notation.

1.1.1 Big-O notation

Definition 1.1. Let $f(n)$ and $g(n)$ be functions defined on the natural numbers. We say that $f(n)$ is big-O of $g(n)$, and write

$$f(n) = O(g(n)),$$

if there exist constants $C > 0$ and $n_0 \in \mathbb{N}$ such that

$$|f(n)| \leq C\,|g(n)|$$

for all $n \geq n_0$.

You can think of $O(g(n))$ as describing a multiplicative bound on the growth of $f(n)$, for large n. That is, $f(n)$ does not grow faster than a constant multiple of $g(n)$ for sufficiently large inputs.

Lemma 1.2. *Let $f(n)$, $g(n)$, $a(n)$, and $b(n)$ be functions satisfying $f(n) = O(g(n))$ and $a(n) = O(b(n))$. Then*

$$f(n) + a(n) = O\big(\max(|g(n)|,\, |b(n)|)\big)$$

and

$$f(n)a(n) = O\big(g(n)b(n)\big).$$

Exercise 1.3. Prove Lemma 1.2.

In this lecture, we will use the term *input size* for an algorithm, which depends on how the data is represented. Precise analysis sometimes requires care in defining what counts as a single operation (for example, adding two numbers versus multiplying two numbers), but for most algorithms in arithmetic and linear algebra we will focus on basic operations such as integer addition, multiplication, and comparisons.

Lemma 1.4. *Let $f(n)$ be a function defined over $\mathbb{N}$. Then $f(n) = O(\log(n))$ if and only if $f(n) = O(\log_k(n))$ for any $k > 1$. In this case, we write $O(\log(n)) = O(\log_k(n))$.*

Proof. We recall the relation between logarithms: $\log(n) = \log_k(n)/\log_k(10)$. This shows that $f(n) = O(\log(n))$ if and only if there is a constant C such that for all large enough n,

$$|f(n)| \leq C|\log(n)| = C\frac{|\log_k(n)|}{|\log_k(10)|} = \frac{C}{\log_k(10)}|\log_k(n)| = C'|\log_k(n)|.$$

This shows that $f(n) = O(\log(n))$ if and only if $f(n) = O(\log_k(n))$. □

1.1.2 Computational models

When deciding on the complexity of an algorithm, it is of vital importance to decide which computational model we are considering. That is, setting a formal framework for what it means to "compute". We have many choices available, for example, Turing machine, Random Access Machine, or even quantum computer. Each choice makes some analysis cleaner or more awkward. One good reference to learn about this is [Pap94]. In this course, we follow the choice of [Har21] and use the deterministic multitape Turing model. Even though this will not be the focus of this course, we give an intuition of what this model does.

(Deterministic) Turing machines

We can think of a (deterministic) Turing machine as an infinite tape together with a head. The head moves along the tape and can read and modify the contents of each position. For a given Turing machine, you need to pick the alphabet (possible symbols in the tape), a set of states (what is the machine doing at a given time), and a function describing the behavior of the machine given the state and symbol.

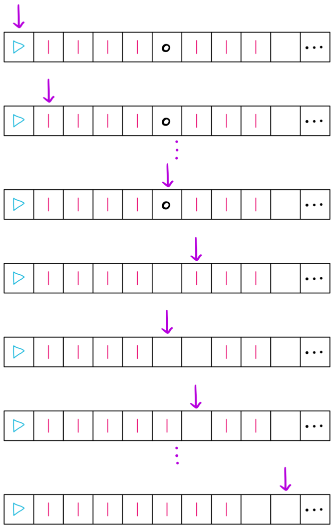

Figure 1.5: A Turing machine to add $4 + 3$.

Example 1.6. Just for fun, we can consider a very simple Turing machine that adds $4 + 3$. We represent the numbers $4 = ||||$ and $3 = |||$. The head always starts at the start symbol $\triangleright$. It moves right until it encounters the symbol $\circ$. Then, it deletes $\circ$, changes it state to "adding", and moves to the right.

While in this state, whenever we encounter a symbol $|$, we delete it, move to the left, add a symbol $|$, and move twice to the right. When the head reads a blank at this state, the new state is "halt" since we are done with the addition. You can see a picture of this in Figure 1.5. You can check that the total number of steps this machine takes to get to "halt" is 15.

Exercise 1.7. Can you describe a Turing machine that adds $4 + 3$ using less steps than in Example 1.6 but using the same alphabet $\{\triangleright, \square, \circ, |\}$?

Exercise 1.8. Can you change the alphabet and describe a Turing machine that adds $4+3$ using less steps than in Example 1.6?

Deterministic multitape Turing machines

Instead of working with one tape in a Turing machine, we can consider the case when we have finitely many tapes and one head that reads one position on each tape. It turns out that multitape Turing machines are as capable as Turing machines, but faster. For instance, one can describe a multitape Turing machine that performs the operation of Example 1.6 in 9 less steps!

Definition 1.9. The complexity of a multitape Turing machine model refers to the number of steps executed by the machine over the course of a computation. Each step that a Turing machine takes is called a bit operation.

Remark 1.10. The complexity of an algorithm depends on what the *alphabet* of the multitape Turing machine is. It it not the same to have a machine that reads any integer, to a machine that only reads $|$ and $\circ$, so we need to be specific about the alphabet.

In practice, we describe the complexity of turing Machines by writing the number of steps in using Big-O notation for functions on the size (number of bits) of the input. We also note that another useful thing to consider is the space complexity, i.e., the amount of memory used by a computation. We will only focus on the time complexity, i.e. the number of steps that it takes to terminate. For certain algorithms the space complexity becomes the main bottleneck in practice, so it is good to remember that this might be a problem.

Definition 1.11. Given an algorithm that takes an input of n-bits and requires at most $O(f(n))$ bit operations to complete, we say the algorithm runs in time $O(f(n))$ or has complexity $O(f(n))$.

Remark 1.12. Because we want to access the number theory and not the computational complexity, through these lectures, we will not be very specific about the particular construction of Turing machines for each algorithm.

1.1.3 An example: addition of integers.

The first question we need to answer is how to represent the objects that we want to input, or what is the alphabet of our Turing machine. In the next section, we discuss representing integers more generally; here, we assume integers are given in binary.

Example 1.13. The integer 431 is represented by the 9-digit binary number 110101111 since

$$431 = 1 \cdot 2^8 + 1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0.$$

Now we are ready to add two integers together. Without loss of generality, we can assume that the length of the expansions is n (we can always pick the maximum length, and then add zeros to the shortest one). Let's look at Algorithm 1.14.

Algorithm 1.14 (Integer addition [HvdH21, Algorithm 2.1]).

The input is two binary expansions $a = (a_{n-1}, \dots, a_1, a_0)$ and $b = (b_{n-1}, \dots, b_1, b_0)$ of length n. This algorithm outputs the binary expansion for the integer c that is the sum $a + b$.

```
1. Set $\gamma_0 := 0$.
2. For $i = 0, \dots, n-1$ do
    (a) Set $c_i := a_i + b_i + \gamma_i$.
    (b) If $c_i \geq 2$, set $c_i := c_i - 2$ and $\gamma_{i+1} := 1$; otherwise, set $\gamma_{i+1} := 0$.
3. Set $c_n := \gamma_n$.
Return $c = (c_n, \dots, c_1, c_0)$.
```

Remark 1.15. The description of the algorithm does not explicitly give a multitape Turing machine (§1.1.2), but you should convince yourself that it gives you all the information you need to rigorously define the machine.

Theorem 1.16. *The complexity of Algorithm 1.14 is $O(n)$.*

Proof. We analyze the complexity step by step. It is useful to recall arithmetic of big-O notation from Lemma 1.2. Step 1 requires one bit operation, so its time is $O(1)$. Step 2 iterates n times. Each iteration performs two bit addition operations, potentially subtracts 2, and sets the carry bit. Each of these operations is a constant-time word operation, so each iteration takes $O(1)$ time. In total, the complexity of Step 2 is $O(n)O(1) = O(n)$. Finally, Step 3 sets one bit, which is time $O(1)$. Altogether, the total complexity of the algorithm is

$$O(1) + O(n) + O(1) = O(n).$$

□

Remark 1.17. The constant n in Theorem 1.16 denotes the length of the inputs, so if a and b are the integers we want to add, then

$$n = \log_2(\max\{a, b\}) = O(\log(\max\{a, b\})),$$

where the last equality follows from Lemma 1.4.

1.2 Integer arithmetic

With basic notation established, we now explore algorithms for integer arithmetic. First, how do we represent integers? Then we move on to addition, multiplication, division, and greatest common divisors.

1.2.1 Representing integers

We will represent numbers in binary, with another bit to represent the sign.

That could be the only line of this subsection. The following is a small detour relates to expressing integers in actual computers.

Modern laptops use a 64-bit processor. That means that each integer can be up to $2^{64} - 1$ in value (unsigned), or from -2^{63} to $2^{63} - 1$ (signed). The CPU can process (add, multiply, etc.) two of these 64-bit numbers in one operation.

We can represent larger integers by an array of 64-bit words as follows

$$a = (-1)^s \sum_{i=0}^{n} a_i 2^{64i}, \tag{1.18}$$

where $s \in \{0, 1\}$, $0 \leq n + 1 < 2^{63}$, and $a_i \in \{0, \ldots, 2^{64} - 1\}$. The numbers a_i are the *digits in base* 2^{64} of a.

Definition 1.19. For an integer a, its standard representation is given by the array

$$(s \cdot 2^{63} + n + 1, a_0, \ldots, a_n),$$

where s and a_i are as in (1.18) and a_n is nonzero if $a \neq 0$. If the standard representation of a has length n we call a an n-bit integer.

1.2.2 Addition

To add integers using their binary representation, we can just use Algorithm 1.14 that has complexity $O(n)$, where n is the number of bits of the integers (in binary). We usually just pick the largest number of bits and set that as n. By Remark 1.17, the complexity of addition is the same as $O(\log(\max(a, b)))$, where a and b are the integers we want to add.

If you went on the detour about the standard representation, you can modify Step 2 (b) of Algorithm 1.14 by

$$\text{If } c_i \geq 2^{64}, \text{ then set } c_i = c_i - 2^{64} \text{ and } \gamma_{i+1} = 1.$$

1.2.3 Multiplication

With addition, we noted that the naive algorithm to add integers (Algorithm 1.14) has complexity $O(n)$, where n is the number of bits. The naive algorithm that we use to multiply integers runs in time $O(n^2)$. This is usually fine for smaller integers, but more efficiency is needed for larger integers.

Exercise 1.20. You can check the time that it takes Magma to run one line by writing `time` at the beginning of the line:

```
> time 2^115032204*3^473444585;
Time: 312.990
```

Can you find two large integers (but maybe not as large as above) for which multiplication takes longer than 0 seconds? How big are your integers? How long does it take to add them?

Exercise 1.21. Write and analyze an algorithm that implements naive multiplication for integers. Your algorithm should use $O(mn)$ word operations, where m and n are the number of bits of the integers you are multiplying.

The theoretical state of the art is the Algorithm presented in [HvdH21]. This algorithm has complexity $O(n \log n)$ and is believed to be optimal but maybe not practical.

In practice, many computer algebra systems use the GMP library, which is a "free library for arbitrary precision arithmetic". GMP implements an algorithm whose theoretical complexity comes very close to $O(n \log n)$ [Har21, Remark 2.12].

1.2.4 Division

Division of large integers is typically accomplished using algorithms based on repeated subtraction, long division, or more advanced methods such as Newton-Raphson iteration for reciprocal approximation. For most practical purposes, the classical long division algorithm suffices.

In general, since $\mathbb{Z}$ is an Euclidean domain, given $a, b \in \mathbb{Z}$, the division algorithm to divide a by b should return the unique integers q and r with $0 \leq r < b$ such that $a = qb + r$. We call r the remainder of dividing a by b, or a modulo b, and denote it as $\mathrm{rem}(a, b)$.

Exercise 1.22. Describe the classical long division algorithm for binary integers. Prove that the algorithm takes time $O((n-m)m)$, where the integers have m and n bits, respectively.

Just like with multiplication, we can improve this bound. A division algorithm that combines fast multiplication with Newton's method gives the same complexity as multiplication: $O(n \log n)$.

1.2.5 Greatest common divisors

The greatest common divisor (gcd) of two integers a and b, denoted $\gcd(a, b)$, is the largest integer that divides both. The standard method to compute gcd's is the Euclidean Algorithm (Algorithm 1.23).

Algorithm 1.23 (Euclidean Algorithm for gcd).

Given integers $a \geq b > 0$, compute $g := \gcd(a, b)$.

```
1. Set $r_0 := a$ and $r_1 := b$.
2. Set $i := 1$ and while $r_i \neq 0$, do the following
    (a) Set $r_{i+1} := \mathrm{rem}(r_{i-1}, r_i)$ and $i := i + 1$.
Return $r_{i-1}$.
```

Exercise 1.24. Explain why Algorithm 1.23 terminates and correctly computes the greatest common divisor.

Theorem 1.25 ([vzGG13, Theorem 3.13])**.** *Algorithm 1.23 for positive n-bit and m-bit integers has complexity $O(mn)$.*

Exercise 1.26. Use Exercises 1.21 and 1.22 to prove Theorem 1.25.

Remark 1.27. As you might know, an essential property of the greatest common divisor is that it corresponds to the smallest positive linear combination of a and b. That is, there exist $x, y \in \mathbb{Z}$ such that $\gcd(a, b) = ax + by$. Running the Euclidean Algorithm for gcd (Algorithm 1.23) is almost enough for computing x and y, we just need to "undo" the operations. An optimized version of this algorithm runs in time $O(n \log^2 n)$, where n is again the number of bits of a and b.

Remark 1.28. In particular, Remark 1.27 implies that we can find the inverse of an n-bit integer in $\mathbb{Z}/M\mathbb{Z}$ for $M \geq 2$ in time $O(n \log^2 n)$.

1.2.6 Large powers

We will look at a basic (but useful) algorithm for computing powers g^k, in the general case when g is an element of any group G and k is an integer.

Algorithm 1.29 (Exponentiation algorithm [Coh93, Algorithm 1.2.1]).

The input is an element g of a multiplicative group G and an integer k. This algorithm computes g^k in G.

```
1. Set $y := 1_G$. If $k = 0$, output $y$ and terminate. If $k < 0$, let $K := -k$ and $z := g^{-1}$.
   Otherwise, set $K := k$ and $z := g$.
2. If $K$ is odd set $y := z \cdot y$.
3. Set $K := \lfloor K/2 \rfloor$. If $K = 0$, output $y$ as the answer and terminate. Otherwise, set
   $z = z \cdot z$ and go to Step 2.
```

Exercise 1.30. Prove that Algorithm 1.29 computes g^k using $O(\log |k|)$ group multiplications. In particular, when g is an integer, prove that the algorithm runs in $O(n \log n)$ time, where n is the number of bits of the input g.

1.2.7 Summary

We end this section with a summary of results on integer arithmetic in Table 1.1.

Operation (of n-bit integers)	**Naive Algorithm**	**Optimized Algorithm**
Addition	$O(n)$	$O(n)$
Multiplication	$O(n^2)$	$O(n \log n)$
Division	$O(n^2)$	$O(n \log n)$
GCD	$O(n^2)$	$O(n \log^2 n)$
Exponentiation (to k-bit integer)	$O(n^2)$	$O(n \log n)$

Table 1.1: Summary of complexity for basic arithmetic algorithms.

1.3 More arithmetic

We will use what we learned about integer arithmetic to study the complexity of modular and polynomial arithmetic.

1.3.1 Modular arithmetic

Let $M \geq 2$. Elements of $\mathbb{Z}/M\mathbb{Z}$ can be represented as integers $x \in \{0, \ldots, M-1\}$. In particular, elements of $\mathbb{Z}/M\mathbb{Z}$ occupy at most $\log(M)$ bits of space. We describe the basic arithmetic operations in this ring and record a summary in Table 1.2.

- To add two elements of $\mathbb{Z}/M\mathbb{Z}$, we can add their representatives and subtract M if the result is $\geq M$. This algorithm has complexity $O(\log(M))$.
- To multiply, we multiply the representatives and then take the reminder modulo M, so the complexity of multiplication is $O(\log M \log\log M)$.
- Division is achieved by running the Euclidean Algorithm as in Remark 1.28 to invert the denominator (complexity $O(\log M(\log\log M)^2)$), and then multiplying by the numerator (complexity $O(\log M \log\log M)$). In total, the complexity of division is $O(\log M(\log\log M)^2)$.
- For exponentiation, one can use Algorithm 1.29 in time $O(\log M(\log\log M)\log k)$.

Remark 1.31. One can estimate the time of performing any arithmetic operation over $\mathbb{Z}/M\mathbb{Z}$ by $O(\log^{1+\epsilon} M)$. This is sometimes enough information for complexity computations.

Operation over $\mathbb{Z}/M\mathbb{Z}$	**Running Time**
Addition	$O(\log M)$
Multiplication	$O(\log M \log\log M)$
Division	$O(\log M \log^2(\log M))$
Exponentiation (x^k)	$O(\log M \log\log M \log k)$

Table 1.2: Summary of running times for modular arithmetic.

1.3.2 Polynomial arithmetic

We can use what we have studied about integer arithmetic to determine the complexity of arithmetic in $\mathbb{Z}[x]$ or $\mathbb{Q}[x]$. In general, polynomials in $\mathbb{Z}[x]$ can be represented as polynomials over finite rings $\mathbb{Z}/M\mathbb{Z}$ for large enough M. Polynomials in $\mathbb{Z}/M\mathbb{Z}[x]$ of degree $< n$ can be represented as a sequence of n coefficients. Also, note that each coefficient can be represented using $O(\log M)$ bits, so to encode a polynomial in $\mathbb{Z}/M\mathbb{Z}[x]$ of degree $< n$, we need space $O(n \log M)$. The running times for algorithms for polynomial multiplication are related to the ones for integer multiplication. Let $\mathsf{M}_{\mathsf{int}}(n)$ denote the cost of multiplying n-bit integers. Then, the cost of polynomial arithmetic is summarized in Table 1.3. The interested reader can look at [Har21, §2].

Operation over $\mathbb{Z}/M\mathbb{Z}[x]$ of deg n	**Running Time**
Addition	$O(n \log M)$
Multiplication	$O(\mathsf{M}_{\mathsf{int}}(n \log(nM)))$
Division	$O(\mathsf{M}_{\mathsf{int}}(n \log(nM)))$

Table 1.3: Summary of running times for polynomial arithmetic.

1.4 Linear algebra

Now that we have looked at basic arithmetic, we move on to linear algebra, an area that also allows us to make (fast) explicit computations. The complexity of arithmetic operations on matrices depends on the complexity of arithmetic over the base field or ring. As we saw in §1.2 and §1.3, this complexity varies and that is why we focus on the number of ring operations (multiplications/divisions) needed for each algorithm. For basic arithmetic of matrices, one can easily find (upper bounds) for the complexity of adding and multiplying matrices with $\mathbb{Z}$ coefficients, so we leave this as an exercise. We will skip the basic arithmetic and move on to more interesting matrix manipulations.

Exercise 1.32. Compute a function $f(n)$ such that $O(f(n))$ represents the number of field multiplications needed to compute the product of two $n \times n$ matrices. Why can you ignore the number of addition operations?

Remark 1.33. The best known bound for the number of operations (over a field) needed to multiply two $n \times n$ matrices is $O(n^{2.3728596})$ [Har21, Remark 2.5.1].

1.4.1 Gaussian elimination

This is perhaps one of the most useful and widely used algorithms in linear algebra. It gives us a way of solving linear systems, compute determinants, and find inverses and pseudo-inverses, etc. Because of the applications, we focus on square matrices.

The number of multiplications/divisions needed in Algorithm 1.34 is $O(n^3/3)$. Now we can look at a couple applications of the ideas from this algorithm.

Algorithm 1.34 (Gaussian Elimination for square matrices [Coh93, Algorithm 2.2.1]). The input is an $n \times n$ matrix M with entries in a field and a vector B of length n. This algorithm returns a vector X such that $MX = B$ if M is invertible and `false` if not.

1. Set $j := 0$.
2. Let $j := j + 1$. If $j > n$, then go to Step 6.
3. If $m_{i,j} = 0$ for all $i \geq j$, then return `false` and terminate. Otherwise, let $i \geq j$ be some index such that $m_{i,j} \neq 0$.
4. If $i > j$, for $l = j, \ldots, n$ exchange $m_{i,l}$ and $m_{j,l}$ and then exchange b_i and b_j.
5. Note that $m_{j,j} \neq 0$. Set $d := m_{j,j}^{-1}$ and for all $k > j$ set $c_k := dm_{k,j}$. For all $k > j$ and $l > j$ set $m_{k,l} := m_{k,l} - c_k m_{j,l}$. Finally, for $k > j$ set $b_k := b_k - c_k b_j$ and go to Step 2.
6. Note that M is now upper-triangular! For $i = n, n-1, \ldots, 1$ set
$$x_i := \left(b_i - \sum_{i<j\leq n} m_{i,j} x_j \right) / m_{i,i},$$
output $X = (x_i)_{1\leq i\leq n}$, and terminate.

Exercise 1.35. Modify Algorithm 1.34 to compute the inverse of a square matrix. Check how many multiplications/divisions does it take to run your algorithm. Can you get the number of operations to be asymptotic to $4n^3/3$?

Exercise 1.36. Modify Algorithm 1.34 to compute the determinant of a square matrix. Check how many multiplications/divisions does it take to run your algorithm. Can you get the number of operations to be asymptotic to $n^3/3$?

Remark 1.37. Algorithm 1.34 hinges on being able to invert $m_{j,j}$ in Step 5. This is an obstacle for computing determinants of matrices with coefficients in integral domains but not fields (which will be essential in the following lectures). The reader can check [Coh93, Algorithm 2.2.6] for an example of an algorithm to solve this. The algorithm takes $O(n^3)$ operations.

1.4.2 Normal forms and picking a basis

Gaussian elimination allows us to represent a matrix by a similar matrix that is simpler. This is definitely not the only normal form for a matrix. In §5.1.1, we will describe and use the Hermite normal form, which works over the integers $\mathbb{Z}$, and allows us to represent ideals in orders.

Other normal forms that we will not focus on here, but which are very useful, include the Smith normal form (which helps with module and abelian group structure computations) and the Jordan normal form.

Finding normal forms is just finding a new basis for your space, in which the linear operator represented by the matrix can be written in a simpler way. The last special basis that we will explore in Lecture 4 is the LLL-reduced basis, for which there is a highly efficient algorithm to compute.

Lecture 2: Algebraic numbers and number fields

Number fields are at the heart of number theory. They are a natural setting where arithmetic and algebraic structures meet. You can find them in the study of Diophantine equations, class field theory, Galois representations, etc. In this lecture, we will focus on studying the basics of number fields and algebraic numbers from a computational perspective. Topics are mostly from [Coh93, Chapters 3 and 4]. For further reading, the reader can consult, for example, [Mil], [Lan94], or [Ste].

2.1 Factoring in $\mathbb{Z}[x]$

As a warm-up, we first go back to the arithmetic of polynomials. Thorough these lectures, we will mainly be working with irreducible polynomials over $\mathbb{Z}$. In this section, we study the difficult problem of factoring polynomials over $\mathbb{Z}[x]$ or certifying that said polynomial is irreducible. Many approaches rely on being able to factor over finite fields and the use of Hensel's lemma. Another way of solving the problem uses the LLL algorithm, which we will study in Lecture 4. In this section, we will investigate a method that uses field operations (which, in practice, gives a faster approach).

Let $f(x)$ be a non-zero polynomial in $\mathbb{Z}[x]$. We may assume that the greatest common divisor of the coefficients of $f(x)$ is 1 by dividing by any common factor; such a polynomial is called primitive. By Gauss's Lemma, deciding if $f(x)$ is irreducible over $\mathbb{Q}[x]$ is equivalent to deciding if it is irreducible over $\mathbb{Z}[x]$, so there is nothing to gain by considering operations over $\mathbb{Q}$.

We sketch the steps to decide if a primitive polynomial $f(x) \in \mathbb{Z}[x]$ is irreducible, or find a nontrivial factorization (for details, see [Coh93, Algorithm 3.5.7]). The method relies on picking a suitable prime number p.

Step 1: reduce to squarefree polynomials

Deciding if $f(x)$ is squarefree reduces to computing $\gcd(f(x), f'(x))$, as shown in Exercise 2.1. If we want to factor the polynomial, then we can then factor $f(x)/\gcd(f(x), f'(x))$, which is squarefree by construction.

Exercise 2.1. Let $f(x) \in \mathbb{Z}[x]$ be a monic polynomial. Show that $f(x)$ is squarefree in $\mathbb{Z}[x]$ if and only if $\gcd(f(x), f'(x)) = 1$.

Step 2: bound the coefficients of the factors

Assume $f(x) \in \mathbb{Z}[x]$ is primitive and squarefree. For integer polynomials $f(x)$ and $g(x)$ with $g(x)|f(x)$, there is an explicit bound for the absolute value of the coefficients of $g(x)$ in terms of the coefficients of $f(x)$ (see [Coh93, Theorem 3.5.1]). This allows us to find $B > 0$

bounding the coefficients of all irreducible factors of $f(x)$ of degree $\leq \deg(f(x))/2$. Let $\ell(f(x))$ be the leading coefficient of $f(x)$. Choose e to be the smallest exponent for which $p^e > 2\ell(\bar{f})B$.

Step 3: find a factorization over a finite field

Let $\mathbb{F}_p$ be the field $\mathbb{Z}/p\mathbb{Z}$. In this step, we find a factorization of $f(x)$ over $\mathbb{F}_p[x]$. There are many algorithms to do this, but we pick one that works well for small p to show the main ideas (see Algorithm 2.3). We will not show that the algorithm is correct, but the interested reader can look at Exercise 2.4. The main idea of this algorithm relies on the following proposition.

Proposition 2.2 ([Coh93, Proposition 3.4.9]). *Let $\bar{f}(x) \in \mathbb{F}_p[x]$ be squarefree and assume that its decomposition into irreducibles is $\bar{f}(x) = \prod_{1\leq i\leq r} f_i(x)$. The polynomials $T(x) \in \mathbb{F}_p[x]$ with $\deg(T(x)) < \deg(\bar{f}(x))$ for which for each i with $q \leq i \leq r$ there exists $s_i \in \mathbb{F}_p$ with $T(x) \equiv s_i \pmod{f_i(x)}$, are exactly the p^r polynomials $T(x)$ such that $\deg(T(x)) < \deg(\bar{f}(x))$ and $T(x)^p \equiv T(x) \pmod{\bar{f}(x)}$.*

Algorithm 2.3 (Berlekamp for small p [Coh93, Algorithm 3.4.10]).

The input is a squarefree polynomial $\bar{f} \in \mathbb{F}_p[x]$ of degree n, this algorithm computes the factorization of $\bar{f}(x)$ into irreducible factors.

1. Compute inductively for $0 \leq k < n$ values $q_{i,k} \in \mathbb{F}_p$ such that
$$x^{pk} \equiv \sum_{0\leq i<n} q_{i,k}x^i \pmod{\bar{f}(x)}.$$
2. Let $Q := [q_{i,k}]_{i,k}$. Find a basis $V_1, \dots, V_r$ of the kernel $Q - I$ such that V_1 is the column vector $(1, 0, \dots, 0)^t$. Set $E := \{\bar{f}(x)\}$, $k := 1$, and $j := 1$.
3. If $k = r$, output E as the set of irreducible factors of $\bar{f}(x)$ and terminate. Otherwise, set $j := j + 1$, and let $T(x) := \sum_{0\leq i<n}(V_j)_i x^i$.
4. For each polynomial $g(x) \in E$ such that $\deg(g(x)) > 1$ do the following. For each $s \in \mathbb{F}_p$ compute $\gcd(g(x), T(x) - s)$. Let F be the set of such gcd's whose degree is greater than or equal to 1. Set $E := (E - g(x)) \cup F$ and $k := k - 1 + |F|$. If in the course of this computation we reach $k = r$, output E and terminate the algorithm. Otherwise, go to Step 3.

Exercise 2.4. Show that Algorithm 2.3 terminates and correctly computes the factorization of $\bar{f}$ into irreducibles. You can follow the following steps.

1. As a warm-up, let $\bar{f}(x) \in \mathbb{F}_p(x)$ be a polynomial of degree n. Show that $\bar{f}(x)$ is irreducible if and only if

(i) $x^{p^n} \equiv x \pmod{\bar{f}(x)}$; and

(ii) for each prime $q|n$, $\gcd(x^{p^{n/q}} - x, \bar{f}(x)) = 1$.

2. Prove Proposition 2.2.

3. Using the notation of Step 2 of the algorithm, show that any polynomial $T(x)$ in the kernel of $Q - I$ holds that $T(x)^p \equiv T(x) \pmod{\bar{f}(x)}$.

4. Explain why the dimension of $\ker(Q - I)$ is exactly r and why the column vector $(1, 0, \dots, 0)^t$ belongs to the kernel.

5. Let $T(x)$ be a polynomial corresponding to a V_j. Explain why the polynomials F from Step 4 of the algorithm correspond to irreducible factors once we have $k = r$.

Step 4: lift the factorization

This step shows a very useful technique when trying to approximate a solution by its residues modulo p. This is done, for example, when working with p-adic integers. The idea is to use Hensel's Lemma, which we recall since it is a fundamental result.

Lemma 2.5 (Hensel's Lemma for integers)**.** *Let $f(x) \in \mathbb{Z}[x]$ and let p be a prime. Suppose there exists $a_0 \in \mathbb{Z}$ such that*

$$f(a_0) \equiv 0 \pmod{p}, \quad \textit{and} \quad f'(a_0) \not\equiv 0 \pmod{p}.$$

Then for every $k \geq 1$, there exists an integer a_k such that

$$f(a_k) \equiv 0 \pmod{p^k} \quad \textit{and} \quad a_k \equiv a_0 \pmod{p}.$$

There is a similar version for polynomials

Lemma 2.6 (Hensel's Lemma for polynomials)**.** *Let p be a prime and $f(x) \in \mathbb{Z}[x]$. Suppose $f(x) \equiv g_0(x)h_0(x) \pmod{p}$, where $g_0(x), h_0(x) \in \mathbb{Z}[x]$ are monic and coprime modulo p. Then, for each $k \geq 1$, there exist monic polynomials $g_k(x), h_k(x) \in \mathbb{Z}[x]$ such that*

$$f(x) \equiv g_k(x)h_k(x) \pmod{p^k}, \quad g_k(x) \equiv g_0(x) \pmod{p}, \quad h_k(x) \equiv h_0(x) \pmod{p},$$

and $g_k(x)$, $h_k(x)$ remain coprime modulo p.

This last Lemma allows us to lift the factorizations from Step 3 to

$$f(x) \equiv \ell(f)\widetilde{T}_1(x) \cdots \widetilde{T}_r(x) \pmod{p^e},$$

where e is as in Step 2 and the polynomials $\widetilde{T}_i(x)$ are monic.

Step 5: combine multiple factors

We now repeat for every $d \in \{1, \dots, r/2\}$. For every combination of factors $\bar{U} := \widetilde{T}_{i_1} \cdots \widetilde{T}_{i_d}$, where we take $i_d := 1$ if $d = 1/2r$, compute the unique polynomial $U \in \mathbb{Z}[x]$ such that all the coefficients of U are in $[-1/2p^e, 1/2p^e)$, and satisfying

$$\begin{aligned} U &\equiv \ell(f)\bar{U} \pmod{p^e}, && \text{if } \deg(\bar{U}) \leq 1/2\deg(f), \\ U &\equiv f/\bar{U} \pmod{p^e}, && \text{if } \deg(\bar{U}) > 1/2\deg(f). \end{aligned}$$

If U divides $\ell(f)f$ in $\mathbb{Z}[x]$, output the factor $F = U/\gcd(u_i)$, set $f(x) = f(x)/F$, and remove the corresponding $\widetilde{T}_i$ from the list of factors modulo p^e. If $d \geq (1/2)r$, terminate the algorithm by outputting $f(x)$.

If we are done looking at all the possible combinations, we have shown that $f(x)$ is irreducible.

Exercise 2.7. Look through the sketch of the steps of the algorithm to factor primitive polynomials $f(x) \in \mathbb{Z}[x]$ and decide what primes are good candidates to run the algorithm on.

Exercise 2.8. You can generate random integer polynomials of degree d with coefficients in $[-b, b]$ in Magma by running the script

```
R<t,y> := PolynomialRing(Integers(),2);
S<x> := PolynomialRing(Integers());
f := Evaluate(Random(d,R,b),[x,1]);
```

Create a polynomial that has 10 random divisors of degree 3. Run the steps of the algorithm with at least two primes and compare results.

2.2 Number fields

We are now ready to embark on our study of algebraic number theory! To establish notation, we first review the basic definitions for number fields.

Definition 2.9. A number field K is a field such that K is a finite-dimensional $\mathbb{Q}$-vector space.

Definition 2.10. Let K be a number field, the degree of K, denoted $[K : \mathbb{Q}]$, is the dimension of K as a $\mathbb{Q}$-vector space.

A really good place to find examples of number fields is the number field database of the LMFDB. They even have pictures(!), like the one in Figure 2.11.

Also, if you want to work with number fields in Magma, you can download the database Anf.tar.gz, “comprising over 2.6 million number fields of degrees between 2 and 9 (inclusive)”.

Now, we are ready to look at some examples.

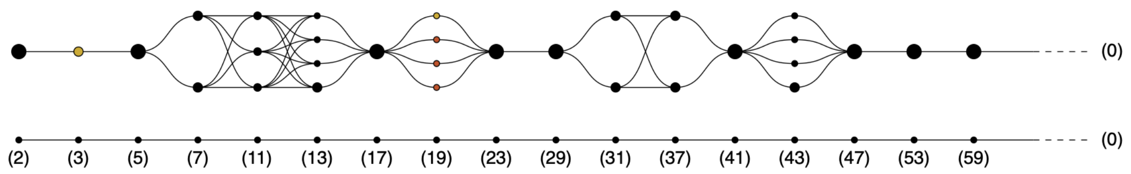

Figure 2.11: Number field 6.0.9747.1.

Example 2.12. The field of rational numbers $\mathbb{Q}$ is a number field (of degree 1).

Example 2.13. Let $d \in \mathbb{Z}$ be a nonzero integer that is not a square. The vector space

$$\mathbb{Q}(\sqrt{d}) := \{a + b\sqrt{d} \, : \, a, b \in \mathbb{Q}\}$$

is a number field. A $\mathbb{Q}$-basis is $\{1, \sqrt{d}\}$, so the number field has degree 2. These number fields are called quadratic fields. To create them in Magma, you can type

```
K<s> := QuadraticField(d);
```

The variable s represents $\sqrt{d}$ (or $-\sqrt{d}$, they are indistinguishable). Indeed, you can check

```
assert s^2 eq d;
```

Exercise 2.14. Prove that $\mathbb{Q}(\sqrt{d})$ is the smallest field containing $\sqrt{d}$.

Example 2.15. Let $n \geq 2$ be an integer and let $\zeta_n := \exp(2\pi i/n)$ be a primitive n-th root of unity. Then

$$\mathbb{Q}(\zeta_n) := \left\{ \sum_{i=0}^{n-1} a_i \zeta_n^i \, : \, a_i \in \mathbb{Q} \right\}$$

is a number field and it is called the n-th cyclotomic field. The degree $[\mathbb{Q}(\zeta_n) : \mathbb{Q}] = \varphi(n)$, where $\varphi(n)$ is Euler's totient function. You can define this number field in Magma as

```
K<z> := CyclotomicField(n);
```

and just as with quadratic fields, z represents ζ_n (or any power ζ_n^k with $\gcd(k, n) = 1$).

Example 2.16. For a more general example, consider an irreducible polynomial $f(x) \in \mathbb{Q}[x]$ of degree d. The quotient $\mathbb{Q}[x]/(f)$ is called an algebraic extension of $\mathbb{Q}$ and is a number field of degree d.

Let $f(x) \in \mathbb{Q}[x]$ be an irreducible polynomial. Let α be a root of $f(x)$. The field $\mathbb{Q}(\alpha)$ denotes the smallest field that contains both $\mathbb{Q}$ and α. Then, we have $\mathbb{Q}[x]/(f(x)) \simeq \mathbb{Q}(\alpha)$ via $x \mapsto \alpha$. In this sense, note that it does not matter which root of $f(x)$ we pick: they all produce isomorphic fields. When we pick a root α, we pick an explicit embedding

$$\mathbb{Q}[x]/(f(x)) \simeq \mathbb{Q}(\alpha) \subseteq \mathbb{C}. \tag{2.17}$$

Exercise 2.18. With the notation above, convince yourself that $\mathbb{Q}[\alpha] = \mathbb{Q}(\alpha)$. Note that it is enough to write the inverse of α as a polynomial (over $\mathbb{Q}$) in α. What is the degree of $\mathbb{Q}(\alpha)$ in terms of the degree of $f(x)$?

Lemma 2.19. *If K and L are number fields with $K \subseteq L$, then $[K:\mathbb{Q}]$ divides $[L:\mathbb{Q}]$.*

Proof. This follows from basic facts about vector spaces. By definition, $[K:\mathbb{Q}]$ is the dimension of K as a $\mathbb{Q}$-vector space, and $[L:\mathbb{Q}]$ is the dimension of L as a $\mathbb{Q}$-vector space. Also, because $K \subseteq L$, then L is a K-vector space of finite dimension n. Choose a $\mathbb{Q}$-basis $k_1, \dots, k_m$ for K, where $m = [K:\mathbb{Q}]$, and a K-basis $\ell_1, \dots, \ell_n$ for L. Then the set

$$\{\ell_i k_j \mid 1 \leq i \leq n,\, 1 \leq j \leq m\}$$

forms a $\mathbb{Q}$-basis for L. Therefore, $[L:\mathbb{Q}] = nm$. □

Examples 2.13 and 2.15 are particular instances of Example 2.16. Indeed,

$$\mathbb{Q}(\sqrt{d}) \simeq \mathbb{Q}[x]/(x^2-d) \quad \text{and} \quad \mathbb{Q}(\zeta_n) \simeq \mathbb{Q}[x]/(\Phi_n(x)),$$

where $\Phi_n(x)$ is the cyclotomic polynomial of degree $\varphi(n)$. It turns out that we have described all number fields just by looking at Example 2.16! The following will be a corollary of Theorem 2.33.

Theorem 2.20. *If K is a number field, then K is an algebraic extension of $\mathbb{Q}$.*

Remark 2.21. Given a number field $K = \mathbb{Q}(\alpha)$ of degree n over $\mathbb{Q}$, the choices of roots of $m_\alpha(x)$ provide n distinct embeddings of K in $\mathbb{C}$, as constructed in (2.17). In fact, we will show that every number field can be written as $\mathbb{Q}(\theta)$, so every number field of degree n comes equipped with n distinct embeddings

$$\sigma_i \colon K \hookrightarrow \mathbb{C}, \qquad\qquad \sigma_i(\alpha) = \alpha_i.$$

In fact, every element α of a number field K is a root of a polynomial in $\mathbb{Q}[x]$, which implies that $\mathbb{Q}(\alpha)$ is a number field too. This invites the following definition.

Definition 2.22. Let $\alpha \in \mathbb{C}$, we say that α is an algebraic number if α is a root of a nonzero polynomial in $\mathbb{Q}[x]$. The minimal polynomial of α is a monic polynomial $m_\alpha(x) \in \mathbb{Q}[x]$ of minimal degree. We call all the roots of $m_\alpha(x)$ the conjugates of α. We denote the set of algebraic numbers as $\bar{\mathbb{Q}}$.

Remark 2.23. If $\alpha \in \mathbb{C}$ is an algebraic number, then we can copy the definition from Example 2.16 to see that $\mathbb{Q}(\alpha) \simeq \mathbb{Q}[x]/(m_\alpha(x))$, where $m_\alpha(x)$ is the minimal polynomial of α.

Example 2.24. Let $\alpha_1, \dots, \alpha_k$ be algebraic numbers. The number field $\mathbb{Q}(\alpha_1, \dots, \alpha_k)$ is the smallest field that contains $\mathbb{Q}$ and $\alpha_1, \dots, \alpha_k$. Equivalently, it is the *compositum* of the number fields $\mathbb{Q}(\alpha_1), \dots, \mathbb{Q}(\alpha_k)$.

Example 2.25. Working with number fields in Magma can be tricky. Let's consider the number fields $K := \mathbb{Q}(\sqrt{6})$ and $L := \mathbb{Q}(\sqrt{2})\mathbb{Q}(\sqrt{3}) = \mathbb{Q}(\sqrt{2}, \sqrt{3})$, the compositum of $\mathbb{Q}(\sqrt{2})$ and $\mathbb{Q}(\sqrt{3})$:

```
K<z> := QuadraticField(6);
L := Compositum(QuadraticField(2), QuadraticField(3));
```

The variable `z` is representing $\sqrt{6}$. Since $\sqrt{6} = \sqrt{2}\sqrt{3}$, we could argue that `z` is an element of L. However, asking something like: `z in L;` gives an error. The problem? You do not know yet an embedding from K to L, so Magma does not know in advance that they are related. You can first ask `IsSubfield(K,L);`, which stores the field embedding $K \subset L$, and allows you to write z as an element of K. Now, `z in L;` returns `true`. Moreover, trying `L!z;` writes `z` in the basis of L.

Lemma 2.26. *Let α be an algebraic number with minimal polynomial $m_\alpha(x)$. Assume that $f(x) \in \mathbb{Q}[x]$ satisfies $f(\alpha) = 0$. Then $m_\alpha(x)$ divides $f(x)$ in $\mathbb{Q}[x]$.*

Exercise 2.27. Prove Lemma 2.26 by using the Euclidean Algorithm.

The set of algebraic numbers, $\bar{\mathbb{Q}}$, is strictly contained in the complex numbers. The numbers in the difference of these sets are called transcendental numbers. You can find an interesting Quanta Magazine article on the history of transcendental numbers here.

Another structural fact about algebraic numbers is that $\bar{\mathbb{Q}}$ forms a field. You might have seen proofs of this, that follow directly from using Theorem 2.20. But what if we care about *representing* $\alpha+\beta$, $\alpha\beta$, and α/β as algebraic numbers for $\alpha, \beta \in \bar{\mathbb{Q}}$? Let's see what we mean.

2.3 Representing algebraic numbers

From an algorithmic perspective, it is useful to be able to represent algebraic numbers efficiently. We will discuss various explicit representations of algebraic numbers and analyze their computational properties, particularly with respect to arithmetic operations such as addition, multiplication, and inversion.

2.3.1 Using minimal polynomials

Let α be an algebraic number with minimal polynomial $m_\alpha(x)$. Remark 2.23 gives an isomorphism $\mathbb{Q}(\alpha) \simeq \mathbb{Q}[x]/(m_\alpha(x))$, so α can be represented as the class of x in the quotient ring $\mathbb{Q}[x]/(m_\alpha(x))$. We now need to figure out how to add, multiply, and divide algebraic numbers using this representation.

Given two algebraic numbers α and β, the minimal polynomials of $\alpha + \beta$, $\alpha\beta$, or α/β can be computed using *resultants*.

Definition 2.28. Let R be an integral domain with fraction field K, and let $\bar{K}$ be the algebraic closure of K. Let $A(x), B(x) \in R[x]$ be polynomials of degree m and n, respectively. Decompose $A(x) = a\prod_{i=1}^{m}(x - \alpha_i)$ and $B(x) = b\prod_{i=1}^{n}(x - \beta_i)$ in $\bar{K}$, so $\alpha_1, \ldots, \alpha_m$ are the roots of A and $\beta_1, \ldots, \beta_n$ are the roots of B. The resultant of $A(x)$ and $B(x)$, denoted

$\operatorname{Res}(A(x), B(x))$, is given by one of the equivalent formulas

$$\begin{aligned}\operatorname{Res}(A(x), B(x)) &= a^n B(\alpha_1) \cdots B(\alpha_m)\\ &= (-1)^{mn} b^m A(\beta_1) \cdots A(\beta_n)\\ &= a^n b^m \prod_{\substack{1 \le i \le m\\ 1 \le j \le n}} (\alpha_i - \beta_j).\end{aligned}$$

Equivalently, the resultant of two polynomials $A(x) = \sum_{i=0}^m a_i x^i$ and $B(x) = \sum_{i=0}^n b_i x^i$ is the determinant of the *Sylvester matrix* associated to $A(x)$ and $B(x)$:

$$\operatorname{Res}(A(x), B(x)) = \det \begin{pmatrix} a_m & a_{m-1} & \cdots & a_0 & 0 & 0 & 0\\ 0 & a_m & a_{m-1} & \cdots & a_0 & 0 & 0\\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots\\ 0 & 0 & 0 & a_m & a_{m-1} & \cdots & a_0\\ b_n & b_{n-1} & \cdots & b_0 & 0 & 0 & 0\\ 0 & b_n & b_{n-1} & \cdots & b_0 & 0 & 0\\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots\\ 0 & 0 & 0 & b_n & b_{n-1} & \cdots & b_0 \end{pmatrix}. \tag{2.29}$$

As we saw in Remark 1.37, computing the determinant of the matrix has cost $O((m+n)^3)$ integer multiplications.

Remark 2.30. If you want to explore a more efficient algorithm for computing resultants, you can check [Coh93, Algorithm 3.3.7].

Now we are ready to compute minimal polynomials! Recall our goal: for two algebraic numbers α and β, we want to find the minimal polynomials of $\alpha + \beta$, $\alpha\beta$, and α/β from the information of the minimal polynomials $m_\alpha(x)$ and $m_\beta(x)$. Let $\alpha_1, \dots, \alpha_m$ and $\beta_1, \dots, \beta_n$ be the roots of $m_\alpha(x)$ and $m_\beta(x)$, respectively.

We add an auxiliary variable y, then compute

$$m_\alpha(x-y) = \prod_{i=1}^m (x - y - \alpha_i) = (-1)^m \prod_{i=1}^m (y - (x - \alpha_i)),$$

so $x - \alpha_i$ are the roots of $m_\alpha(x-y)$, seen as a polynomial in y. Consequently, by definition, the resultant of $m_\alpha(x-y)$ and $m_\beta(y)$ seen as polynomials in y is

$$\operatorname{Res}_y(m_\alpha(x-y), m_\beta(y)) = \prod_{\substack{1 \le i \le m\\ 1 \le j \le n}} ((x - \alpha_i) - \beta_j) = \prod_{\substack{1 \le i \le m\\ 1 \le j \le n}} (x - (\alpha_i + \beta_j)).$$

By construction, this polynomial in x is monic and has $\alpha + \beta$ as a root. It also has coefficients in $\mathbb{Q}$ by (2.29). If the polynomial is irreducible, then it must be the minimal polynomial of $\alpha + \beta$. If not, then the minimal polynomial $m_{\alpha+\beta}(x)$ must divide it by Lemma 2.26. We can then factor it using [Coh93, Algorithm 3.5.7].

Exercise 2.31. Show that $\operatorname{Res}_y(y^m m_\alpha(x/y), m_\beta(y))$ has $\alpha\beta$ as a root, so factoring this polynomial will result on finding the minimal polynomial of $\alpha\beta$. Similarly, show that you can recover the minimal polynomial of α/β from $\operatorname{Res}_y(m_\alpha(xy), m_\beta(y))$.

2.3.2 Primitive element theorem and the standard representation

A much easier situation occurs when we work inside a number field $\mathbb{Q}(\alpha)$, where we can find representations by using information from $m_\alpha(x)$. This is easier to see for quadratic number fields, as shown by the following exercise.

Exercise 2.32. Consider the quadratic number field $K := \mathbb{Q}(\sqrt{-7})$. Note that $\sqrt{-7}+1$ is an element of K. Can you find its minimal polynomial? How is it related to the minimal polynomial of $\sqrt{-7}$? Can you now find an algorithm to compute the minimal polynomial of any element $a + b\sqrt{-7} \in K$? Can you generalize this to any quadratic number field?

In general, we have the following theorem.

Theorem 2.33 (Primitive Element Theorem)**.** *Let K be a number field, then there exists an element $\theta \in K$ such that $K = \mathbb{Q}(\theta)$. We say that θ is a primitive element.*

Sketch of the proof. Let $K = \mathbb{Q}(\alpha_1, \alpha_2, \ldots, \alpha_m)$ be a number field generated over $\mathbb{Q}$ by algebraic numbers α_i. We will show that there is $\theta \in K$ such that $K = \mathbb{Q}(\theta)$. It suffices to show this for $K = \mathbb{Q}(\alpha, \beta)$ (by induction on the number of generators). If α and β are both in $\mathbb{Q}$, then $K = \mathbb{Q}$. Otherwise, consider the elements $\theta = \alpha + c\beta$ for $c \in \mathbb{Q}$. It turns out that $K = \mathbb{Q}(\theta)$ for all but finitely many c (proving this fact requires using automorphisms of K, so we skip it for brevity). Pick one of the infinitely many c so $K = \mathbb{Q}(\theta)$[1]. □

Exercise 2.34. Consider the biquadratic number field $\mathbb{Q}(\sqrt{a}, \sqrt{b})$. Follow the proof of Theorem 2.33 to find a primitive element θ such that $\mathbb{Q}(\sqrt{a}, \sqrt{b}) = \mathbb{Q}(\theta)$. Can you find a way to compute the minimal polynomial of θ? Can you write $\sqrt{a}$ and $\sqrt{b}$ as polynomials in θ? If you want, you can pick specific values for a and b.

Remark 2.35. The proof of Theorem 2.33 gives an explicit algorithm for computing a primitive element for any number field K. Given $K = \mathbb{Q}(\alpha_1, \ldots, \alpha_m)$, you need to try combinations $\theta_{(c)} := \sum_{i=1}^m c_i\alpha_i$ for $c_i \in \mathbb{Q}$ until you get $K = \mathbb{Q}(\theta_{(c)})$. This process will terminate since there are only finitely many vectors (c) for which $\mathbb{Q}(\theta_{(c)}) \subsetneq K$. However, this algorithm depends on being able to compute if two number fields are equal. We will study this problem in Lecture 4.

Going back to our problem of representing algebraic numbers, Theorem 2.33 gives us a way to perform easier arithmetic when the algebraic numbers belong to the same number field (Given $\alpha, \beta \in \bar{\mathbb{Q}}$, we have that $\mathbb{Q}(\alpha, \beta)$ is indeed a number field containing all the relevant quantities we care about).

If K is a number field of degree n, then we can find a primitive element θ with minimal polynomial $m_\theta(x)$ of degree n. Then, all elements $\alpha \in K = \mathbb{Q}(\theta)$ can be represented uniquely as $\sum_{i=0}^{n-1} b_i\theta^i$ for $b_i \in \mathbb{Q}$. Note that this just means that $\{1, \theta, \ldots, \theta^{n-1}\}$ is a $\mathbb{Q}$-basis of K.

[1] In practice, trying $\alpha + \beta$ is always a good choice.

Taking d as the (positive) greatest common divisor of the rational numbers b_i, we arrive to the representation

$$\alpha = \frac{\sum_{i=0}^{n-1} a_i\theta^i}{d}, \quad d > 0, a_i \in \mathbb{Z}, \text{ and } \gcd(a_0, \dots, a_{n-1}, d) = 1,$$

called the standard representation of α with respect to θ.

Remark 2.36. Magma represents elements of number fields using the standard representation with respect to a primitive element which is stored when you create the number field.

Example 2.37. We can consider the number field $K = \mathbb{Q}(\sqrt{2}, \zeta_3)$. we can create this field in Magma by writing it as the compositum of $\mathbb{Q}(\sqrt{2})$ and $\mathbb{Q}(\zeta_3)$. Every time you create a number field in Magma, it comes with a primitive element θ that we can recover

```
N1<s> := QuadraticField(2);
N2<z> := CyclotomicField(3);
K<theta> := Compositum(N1, N2);
```

Then we can check that `z+s eq theta;`, so $s + z$ is chosen as the primitive element of K. We can also check that the elements of K are written in the standard representation with respect to θ. For example, `-(1/50)*s+1+3*z^2+(3/2)*z;` returns

```
1/550*(-148*theta^3 - 222*theta^2 - 159*theta - 730)
```

Let's understand the complexity of working with this standard representation. Adding elements of K reduces to (basically) vector addition in $\mathbb{Q}^{n+1}$, so it takes $O(n)$ integer operations. To study multiplication, let $m_\theta(x) = \sum_{i=0}^{n} t_i x^i \in \mathbb{Q}[x]$ be the minimal polynomial of θ, so $t_n = 1$. We note that we can reduce

$$\theta^n = -t_{n-1}\theta^{n-1} - \cdots - t_0,$$

and the use recursion to reduce any power $k \geq n$. To make multiplication more efficient, we can precompute and store those reductions. Let $k \geq n$ and write

$$\theta^{n+k} = \sum_{i=0}^{n-1} r_{i,k}\theta^i \tag{2.38}$$

so $r_{i,0} = -t_i$ and

$$r_{k+1,i} = \begin{cases} r_{k,i-1} - t_i r_{k,n-1} & \text{if } i \geq 1, \\ -t_0 r_{k,n-1} & \text{if } i = 0. \end{cases}$$

Exercise 2.39. Show that precomputing the constants $r_{i,k}$ as in (2.38) takes $O(kn)$ field operations.

Once we know the coefficients $r_{i,k}$, we can compute the product of any two elements $\beta, \gamma \in K$ using Algorithm 2.40.

Exercise 2.41. Study the complexity of Algorithm 2.40 in terms of the number of integer operations. You might find the estimates of § 1.3.2 useful.

Algorithm 2.40 (Multiplication in standard representation).

The input is two algebraic numbers in $K = \mathbb{Q}(\theta)$, written in the standard representation $\beta = \frac{1}{d_\beta}\sum_{i=0}^{n-1} b_i\theta^i$ and $\gamma = \frac{1}{d_\gamma}\sum_{j=0}^{n-1} c_j\theta^j$, where $b_i, c_j \in \mathbb{Z}$. We also input the precomputed values $r_{i,k}$ up to $k = 2(n-1)$.

```
1. Set $d := d_\beta d_\gamma$.
2. Compute the product polynomial: $h(x) := \big(\sum_{i=0}^{n-1} b_i x^i\big)\big(\sum_{j=0}^{n-1} c_j x^j\big) = \sum_{i=0}^{(n-1)^2} h_i x^i$.
3. Set $a_i := h_i$ for $i \in \{1, \ldots, n-1\}$.
4. For $k$ in $n, \ldots, (n-1)^2$
    (a) Set $a_i := a_i + r_{i,k}$ for $i = 0, \ldots, n-1$.
5. Compute $g := \gcd(a_1, \ldots a_{n-1}, d)$
6. Set $d := d/g$ and the coefficients $a_i := a_i/g$.
```

Return the standard representation of the product: $\frac{1}{d}\sum_{i=0}^{n-1} a_i\theta^i$.

Remark 2.42. As you can see in Algorithm 2.40, multiplying two algebraic numbers in the standard representation is equivalent to computing a product of two polynomials and then reducing that product modulo $m_\theta(x)$. You can learn more about this in [Coh93, Chapter 3]. However, precomputing the coefficients $r_{i,k}$ makes the algorithm more efficient.

For division, we can use our idea of representing elements of K using polynomials in $\mathbb{Z}[x]$ (plus another integer for the denominator). Then, finding the quotient of β by $\gamma \neq 0$ is equivalent to computing the quotient of the corresponding polynomials modulo $m_\theta(x)$. This can be done as follows. Let $B(x)$ and $C(x)$ be the polynomials associated to β and γ, respectively. The polynomial $C(x)$ coprime to $m_\theta(x)$ since $\gamma \neq 0$ and the degree $C(x)$ is at most $n-1$. Then we can explicitly compute $U(x)$ such that

$$U(x)C(x) + V(x)m_\theta(x) = \gcd(C(x), m_\theta(x)) = 1$$

as part of the computation of the gcd (see §1.3.2). Combined with the multiplication algorithm, we can then obtain the standard representation of β/γ.

2.3.3 Other representations

Using minimal polynomials or the standard representation are not the only ways to represent algebraic numbers. We will not discuss more ways because of time. If you are curious, you can look at, for example [Coh93, § 4.2].

Lecture 3: More on number fields

3.1 Working with algebraic numbers

We have already understood the value of knowing what the minimal polynomial of an algebraic number is. There is other useful information that we can use to compute with algebraic numbers: their trace, norm, and characteristic polynomial. Similar to the minimal polynomial, these notions do not distinguish between conjugates – at the end, we are still working over $\bar{\mathbb{Q}}$ and not $\mathbb{C}$. Some useful references are [Ste, Mil, Coh93].

3.1.1 Trace and Norm

Definition 3.1. Given an algebraic number α, the trace (resp. norm) of α, denoted $\mathrm{Tr}(\alpha)$ (resp. $\mathrm{Nm}(\alpha)$), is the sum (resp. product) of all its conjugates.

Remark 3.2. We can easily recover the norm and the trace of an algebraic number from its minimal polynomial. If α is an algebraic number with minimal polynomial $m_\alpha(x) = \sum_{i=0}^{n} a_i x^i$, then

$$\mathrm{Tr}(\alpha) = -a_{n-1} \qquad \text{and} \qquad \mathrm{Nm}(\alpha) = (-1)^n a_0.$$

Recall that we defined the minimal polynomial to be monic. Otherwise, you can just divide by the leading coefficient of the polynomial.

Exercise 3.3. Prove the assertions from the Remark 3.2.

In particular, Remark 3.2 implies that $\mathrm{Tr}(\alpha)$ and $\mathrm{Nm}(\alpha)$ are rational numbers!

Counterexample 3.4. Calling the quantities in Definition 3.1 the trace and the norm of α makes it sound like the function $\mathrm{Tr}\colon \bar{\mathbb{Q}} \to \mathbb{Q}$ should be additive and $\mathrm{Nm}\colon \bar{\mathbb{Q}} \to \mathbb{Q}$ should be multiplicative. This is not the case. Consider, for example, the algebraic numbers $\sqrt{2}$ and ζ_3 as in Example 2.37. Their minimal polynomials are $x^2 - 2$ and $x^2 + x + 1$, respectively. This implies

$$\mathrm{Tr}(\sqrt{2}) = 0, \qquad \mathrm{Tr}(\zeta_3) = -1,$$
$$\mathrm{Nm}(\sqrt{2}) = -2, \qquad \mathrm{Nm}(\zeta_3) = 1.$$

The minimal polynomial of $\sqrt{2} + \zeta_3$ is $x^4 + 2x^3 - x^2 - 2x + 7$ and the minimal polynomial of the product $\sqrt{2}\zeta_3$ is $x^4 + 2x^2 + 4$. In total, we have

$$\mathrm{Tr}(\sqrt{2} + \zeta_3) = -2 \neq \mathrm{Tr}(\sqrt{2}) + \mathrm{Tr}(\zeta_3),$$
$$\mathrm{Nm}(\sqrt{2}\zeta_3) = 4 \neq \mathrm{Nm}(\sqrt{2})\,\mathrm{Nm}(\zeta_3).$$

This issue is resolved by considering the number fields each algebraic number belongs to.

Definition 3.5. Let K be a number field of degree n over $\mathbb{Q}$ and let σ_i denote the n distinct embeddings of K in $\mathbb{C}$ (see Remark 2.21). The trace of α in K, denoted $\mathrm{Tr}_{K/\mathbb{Q}}(\alpha)$, is the sum $\sum_{i=1}^n \sigma_i(\alpha)$. The norm of α in K, denoted $\mathrm{Nm}_{K/\mathbb{Q}}(\alpha)$, is the product $\prod_{i=1}^n \sigma_i(\alpha)$.
Moreover, the characteristic polynomial $C_\alpha(x)$ of $\alpha \in K$ is

$$C_\alpha(x) := \prod_{i=1}^n (x - \sigma_i(\alpha)).$$

Remark 3.6. With the notation of Definition 3.5, for $\alpha \in K$ with characteristic polynomial $C_\alpha(x) = \sum_{i=0}^n c_i x^i$,

$$\mathrm{Tr}_{K/\mathbb{Q}}(\alpha) = -c_{n-1} \qquad \text{and} \qquad \mathrm{Nm}_{K/\mathbb{Q}}(\alpha) = (-1)^n c_0.$$

Remark 3.7. With the notation of Definition 3.5, the fact that the σ_i's are embeddings implies that for all $\alpha, \beta \in K$ we have that

$$\mathrm{Tr}_{K/\mathbb{Q}}(\alpha+\beta) = \mathrm{Tr}_{K/\mathbb{Q}}(\alpha) + \mathrm{Tr}_{K/\mathbb{Q}}(\beta), \text{ and}$$
$$\mathrm{Nm}_{K/\mathbb{Q}}(\alpha\beta) = \mathrm{Nm}_{K/\mathbb{Q}}(\alpha)\,\mathrm{Nm}_{K/\mathbb{Q}}(\beta).$$

Example 3.8. Going back to Counterexample 3.4, let $K := \mathbb{Q}(\sqrt{2}, \zeta_3)$. You can verify that $K = \mathbb{Q}(\sqrt{2} + \zeta_3)$ and that K has degree 4 over $\mathbb{Q}$. The 4 embeddings of K in $\mathbb{C}$ are determined by

$$\sigma_1(\sqrt{2}+\zeta_3) = \sqrt{2}+\zeta_3 \qquad \sigma_2(\sqrt{2}+\zeta_3) = -\sqrt{2}+\zeta_3$$
$$\sigma_3(\sqrt{2}+\zeta_3) = \sqrt{2}+\zeta_3^2 \qquad \sigma_4(\sqrt{2}+\zeta_3) = -\sqrt{2}+\zeta_3^2.$$

Seen as elements of K, we get

$$\begin{aligned} C_{\sqrt{2}}(x) &= (x-\sqrt{2})^2(x+\sqrt{2})^2 = x^4 - 4x^2 + 4, \\ C_{\zeta_3}(x) &= (x-\zeta_3)^2(x-\zeta_3^2)^2 = x^4 + 2x^3 + 3x^2 + 2x + 1, \\ C_{\sqrt{2}+\zeta_3}(x) &= x^4 + 2x^3 - x^2 - 2x + 7, \\ C_{\sqrt{2}\zeta_3}(x) &= x^4 + 2x^2 + 4. \end{aligned}$$

In total, we can now check that the functions are additive and multiplicative! Indeed,

$$\mathrm{Tr}_{K/\mathbb{Q}}(\sqrt{2}) + \mathrm{Tr}_{K/\mathbb{Q}}(\zeta_3) = 0 - 2 = \mathrm{Tr}_{K/\mathbb{Q}}(\sqrt{2}+\zeta_3),$$
$$\mathrm{Nm}_{K/\mathbb{Q}}(\sqrt{2})\,\mathrm{Nm}_{K/\mathbb{Q}}(\zeta_3) = 4 \cdot 1 = \mathrm{Nm}_{K/\mathbb{Q}}(\sqrt{2}\zeta_3).$$

Exercise 3.9. Let K be a number field of degree n over $\mathbb{Q}$ and let $\alpha \in K$ be an algebraic number with minimal polynomial of degree m. Show that the following hold.

1. m divides n.
2. $\mathrm{Tr}_{K/\mathbb{Q}}(\alpha) = \dfrac{n}{m}\,\mathrm{Tr}(\alpha)$.
3. $\mathrm{Nm}_{K/\mathbb{Q}}(\alpha) = (\mathrm{Nm}(\alpha))^{n/m}$.

3.1.2 Discriminant

Proposition 3.10 ([Coh93, Proposition 4.4.1])**.** *Let K be a number field of degree n with embeddings of K into $\mathbb{C}$ given by $\{\sigma_1, \ldots, \sigma_n\}$, and $\{\alpha_1, \ldots, \alpha_n\}$ be a set of n elements of K. Then*

$$\det([\sigma_i(\alpha_j)]_{i,j})^2 = \det([\mathrm{Tr}_{K/\mathbb{Q}}(\alpha_i \alpha_j)]_{i,j}) \tag{3.11}$$

and this quantity is a rational number.

Definition 3.12. With the notation of Proposition 3.10, the discriminant of $\{\alpha_1, \ldots, \alpha_n\}$, denoted $\mathrm{disc}(\alpha_1, \ldots, \alpha_n)$, is the rational number in (3.11).

Exercise 3.13. Let K be the cyclotomic field $\mathbb{Q}(\zeta_5)$ as in Example 2.15. Compute

1. $\mathrm{disc}(1, \zeta_5, \zeta_5^2, 1 + \zeta_5 + \zeta_5^2)$.

2. $\mathrm{disc}(1, \zeta_5, \zeta_5^2, \zeta_5^3)$.

3. $\mathrm{disc}\left(1, \zeta_5, \frac{\zeta_5^2}{5}, \zeta_5^3\right)$.

3.2 Algebraic integers

First, there were the integers $\mathbb{Z}$. Then, we wanted to consider fractions of integers, and we constructed the rationals $\mathbb{Q}$. That is, the fraction field of $\mathbb{Z}$, $\operatorname{Frac}\mathbb{Z}$, is $\mathbb{Q}$. The algebraic integers arise as a generalization of this.

Definition 3.14. For $\alpha \in \bar{\mathbb{Q}}$, we say that α is an algebraic integer or integral if α is a root of a monic polynomial in $\mathbb{Z}[x]$.

Lemma 3.15. *The set of algebraic integers forms a ring.*

Proof. One can use resultants, as explained in §2.3.1, to construct monic polynomials over $\mathbb{Z}$ that have the sum and product of integral elements as roots. □

Example 3.16. The number $\alpha := \frac{1+\sqrt{-7}}{2}$ is integral since it is a root of the polynomial $x^2 - x + 2$. Also, $\beta := 3\sqrt{-7}$ is integral, being a root of $x^2 + 63$. We use Magma to assert that the methods from §2.3.1 are sufficient to show that $\alpha + \beta$ and $\alpha\beta$ are integralⓇ.

```
> K<s> := QuadraticField(-7); alpha := (1+s)/2;  beta := 3*s;
> R<x> := PolynomialRing(Rationals());
> m_alpha := x^2-x+2; m_beta := x^2+63;
> S<y> := PolynomialRing(R);
> sum := R!Resultant(Evaluate(m_alpha,x-y), Evaluate(m_beta,y));
> prod := R!Resultant(S!(y^2*Evaluate(m_alpha,x/y)),Evaluate(m_beta,y));
> assert Evaluate(sum, alpha+beta) eq 0;
> assert Evaluate(prod, alpha*beta) eq 0;
> sum, prod;
x^4 - 2*x^3 + 131*x^2 - 130*x + 3784
x^4 - 189*x^2 + 15876
```

Definition 3.17. Given a number field K, the ring of integers $\mathcal{O}_K$ of K is the set of all elements in K that are roots of monic polynomials in $\mathbb{Z}[x]$. We call the elements of $\mathcal{O}_K$ the algebraic integers of K.

Following our analogy, the ring of integers is indeed a ring by Lemma 3.15. One can check that $\mathrm{Frac}(\mathcal{O}_K) = K$. It also happens to be a $\mathbb{Z}$-order. Let us see what it means.

Definition 3.18. A $\mathbb{Z}$-order R in a number field K is a subring of K which as a $\mathbb{Z}$-module is finitely generated and of maximal rank $n = [K : \mathbb{Q}]$.

Example 3.19. In the number field $K = \mathbb{Q}(\zeta_3)$, we will show that the ring of integers is $\mathbb{Z}[\zeta_3]$. We graph the order $\mathcal{O}_K$ and also a smaller order $S = \mathbb{Z}[2\zeta_3]$ contained in $\mathcal{O}_K$.

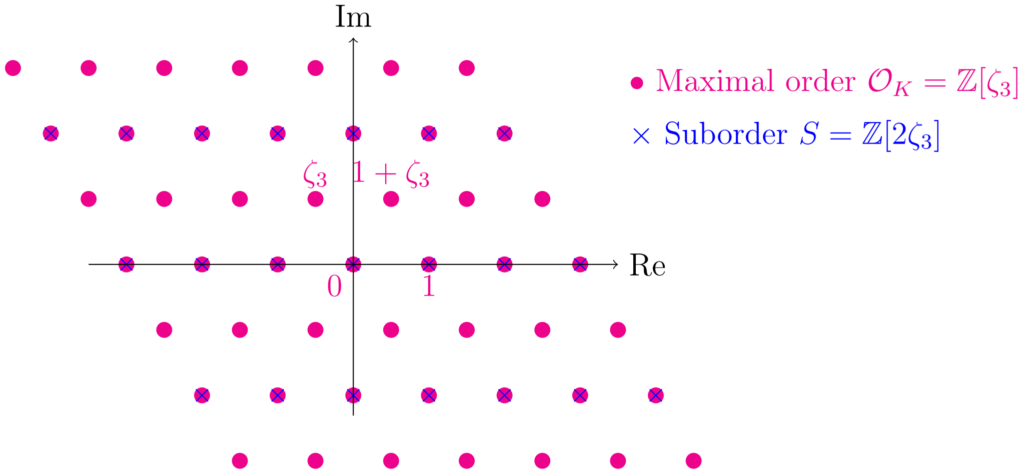
Im
• Maximal order $\mathcal{O}_K = \mathbb{Z}[\zeta_3]$
× Suborder $S = \mathbb{Z}[2\zeta_3]$
ζ_3
$1+\zeta_3$
Re
0
1

Figure 3.20: The order $\mathcal{O}_{\mathbb{Q}(\zeta_3)}$ and a suborder.

The following theorem characterizes the ring of integers of a number field.

Theorem 3.21. *If K is a number field with ring of integers $\mathcal{O}_K$, then $\mathcal{O}_K$ is the unique maximal order in K. That is, $\mathcal{O}_K$ is a subring of K containing 1, integrally closed in K (every element of K that is integral already belongs to $\mathcal{O}_K$), and is maximal among all subrings of K that are finitely generated as $\mathbb{Z}$-modules.*

Counterexample 3.22. Given a number field $K = \mathbb{Q}(\alpha)$ for α an algebraic integer, it is tempting to relate its ring of integers $\mathcal{O}_K$ with $\mathbb{Z}[\alpha]$. Unfortunately, it is only true that $\mathbb{Z}[\alpha] \subseteq \mathcal{O}_K$. For example, the number field $K = \mathbb{Q}\left(\sqrt{5}\right)$ has ring of integers $\mathcal{O}_K = \mathbb{Z}\left[\frac{1+\sqrt{5}}{2}\right]$, with $\mathbb{Z}[\sqrt{5}] \subsetneq \mathcal{O}_K$.

```
K := QuadraticField(5);
OK := RingOfIntegers(K);
assert (2*OK.2-1)^2 eq 5;
```

The ring of integers of a number field is, in particular, a ring. This means that it has ideals. These will be essential to our understanding of arithmetic in number fields and will feature prominently in Lecture 5.

3.2.1 The Structure Theorem

Given a number field K, we saw in Theorem 3.21 that $\mathcal{O}_K$ is a $\mathbb{Z}$-order. To work with such an order, it is really useful to know its dimension as $\mathbb{Z}$-module.

Theorem 3.23 (Structure Theorem)**.** *Let K be a number field of degree n over $\mathbb{Q}$. Then $\mathcal{O}_K$ is a free $\mathbb{Z}$-module of dimension n. In other words, there exist $\omega_1, \dots, \omega_n \in \mathcal{O}_K$ such that every element of $\mathcal{O}_K$ can be written uniquely as $a_1\omega_1 + \cdots + a_n\omega_n$ for $a_i \in \mathbb{Z}$.*

To prove this theorem, we need the following lemma.

Lemma 3.24. *Let K be a number field. For all $\alpha \in K$ there exists a nonzero $d \in \mathbb{Z}$ such that $d\alpha \in \mathcal{O}_K$.*

Exercise 3.25. Show Lemma 3.24 by considering the minimal polynomial of any $\alpha \in K$ and multiplying by the right constant to make it monic.

Sketch of the proof of Theorem 3.23. Let $\alpha_1, \dots, \alpha_n$ be a basis for the number field K over $\mathbb{Q}$. Using Lemma 3.24, we can multiply each basis element by an integer to obtain an algebraic integer, so we can assume without loss of generality that the basis consists of algebraic integers. The free module S generated by the α_i's is contained in $\mathcal{O}_K$. You can then show that $d\mathcal{O}_K \subseteq S$, where $d := \mathrm{disc}(\alpha_1, \dots, \alpha_n)$. The rest follows by properties of $\mathbb{Z}$-modules. □

Definition 3.26. Let K be a number field of degree n over $\mathbb{Q}$. A $\mathbb{Z}$-basis $\{\omega_1, \dots, \omega_n\}$ for $\mathcal{O}_K$ is called an integral basis of K.

Example 3.27. As seen in Counterexample 3.22, $\left\{1, \frac{1+\sqrt{5}}{2}\right\}$ is an integral basis for $\mathbb{Q}(\sqrt{5})$. We can check this in Magma:

```
K<s> := QuadraticField(5);
IntegralBasis(K);
[ 1, 1/2*(s + 1) ]
```

Example 3.28. Let $K = \mathbb{Q}(\zeta_n)$ be a cyclotomic field as in Example 2.15. With n prime, it turns out that the set $\{1, \zeta_n, \zeta_n^2, \dots, \zeta_n^{n-2}\}$ is an integral basis of K. In particular, $\mathcal{O}_K = \mathbb{Z}[\zeta_n]$. You will prove that this is true for n prime in Exercise 3.45.

Lemma 3.29. *If $\{\omega_1, \ldots, \omega_n\}$ and $\{\omega_1', \ldots, \omega_n'\}$ are two integral bases for the ring of integers $\mathcal{O}_K$, then*

$$\operatorname{disc}(\omega_1, \ldots, \omega_n) = \operatorname{disc}(\omega_1', \ldots, \omega_n').$$

Exercise 3.30. Prove Lemma 3.29 by considering the matrix of change of basis.

3.2.2 Representing algebraic integers

Because algebraic integers are, in particular, algebraic numbers, we can represent them by using the methods studied in § 2.3. The extra information that we have is the integral basis of the number field, so it makes sense to write elements of the ring of integers in terms of that integral basis. In this case, algorithms for $\mathbb{Z}$-module arithmetic provide methods to compute inside the order $\mathcal{O}_K$.

Remark 3.31. Magma represents algebraic integers as members of the order $\mathcal{O}_K$, not of the number field K as in Remark 2.36.

The following extract comes from the Magma handbook. Elements of orders are displayed as sequences of integer coefficients, referring to the basis of the order. To convert this $\mathbb{Z}$-basis representation to a polynomial expression in the primitive element of an associated number field, the element should be coerced into the number field (using `!`). To print the element as a linear combination of the basis elements, coerce the element into the field of fractions.

```
> R<x> := PolynomialRing(Integers());
> K<y> := NumberField(x^4-420*x^2+40000);
> O := MaximalOrder(K);
> e := O ! (y^2/40 + y/4);
> f := elt< O | [0, 0, 1, 0]>;
> f eq e;
true
> F<a, b, c, d> := FieldOfFractions(O);
> g := F![0, 0, 1, 0];
> g eq e;
true
> g;
c
```

3.3 The discriminant of a number field

Definition 3.32. Let K be a number field and let $\{\alpha_1, \ldots, \alpha_n\}$ be an integral basis of K. The discriminant of K is

$$\operatorname{disc}(K) := \operatorname{disc}(\alpha_1, \ldots, \alpha_n),$$

where $\operatorname{disc}(\alpha_1, \ldots, \alpha_n)$ is as in Definition 3.12.

Example 3.33. In Exercise 3.13, you computed the discriminant $\operatorname{disc}(1, \zeta_5, \zeta_5^2, \zeta_5^3) = 125$. Because of Example 3.28, that set is an integral basis, so $\operatorname{disc}(\mathbb{Q}(\zeta_5)) = 125$.

The following Proposition will be very useful to compute and work with field discriminants.

Proposition 3.34. *Let $K = \mathbb{Q}(\theta)$, where θ is an algebraic integer with minimal polynomial $m_\theta(x) \in \mathbb{Z}[x]$. Show that the following hold*

1. $\operatorname{disc}(1, \theta, \dots, \theta^{n-1}) = \operatorname{disc}(m_\theta(x))$;
2. *if $f = [\mathcal{O}_K : \mathbb{Z}[\theta]]$, then*
$$\operatorname{disc}(m_\theta(x)) = \operatorname{disc}(K) f^2, \tag{3.35}$$

where $\operatorname{disc}(m_\theta(x))$ denotes the discriminant of the polynomial $m_\theta(x)$: if $\theta_1, \dots, \theta_n$ are the roots of $m_\theta(x)$, then

$$\operatorname{disc}(m_\theta(x)) = \prod_{1 \le i < j \le n} (\theta_i - \theta_j)^2.$$

Exercise 3.36. Prove Proposition 3.34. *Hint:* Let $\omega_1, \dots, \omega_n$ be an integral basis for $\mathcal{O}_K$. Remember that $\operatorname{disc}(K)$ is the determinant of the matrix $\left[\operatorname{Tr}_{K/\mathbb{Q}}(\omega_i \omega_j)\right]_{i,j}$. Express each ω_k as a $\mathbb{Q}$-linear combination of $1, \theta, \dots, \theta^{n-1}$. Compute how the discriminant changes under a change of basis, and use the fact that $[\mathcal{O}_K : \mathbb{Z}[\theta]] = f$. This will relate the discriminants via a square of f.

Warning 3.37. In Magma, the function `Discriminant(K);` returns the discriminant of the polynomial used to define `K`, not the discriminant of `K`. Use

```
Discriminant(RingOfIntegers(K));
```

to get the discriminant of `K`.

We will find an easy way of computing $\operatorname{disc}(m_\theta(x))$, where θ is an algebraic integer. By (3.35) this takes us really close to computing the discriminant of a number field $K = \mathbb{Q}(\theta)$. In particular, if $\operatorname{disc}(m_\theta(x))$ is squarefree, then $\operatorname{disc}(K) = \operatorname{disc}(m_\theta(x))$. Otherwise, we will need to work harder to compute $\operatorname{disc}(K)$ (for example by computing an integral basis of K).

Lemma 3.38. *Let $f(x) \in \mathbb{Z}[x]$ be an irreducible monic polynomial of degree n. Then,*

$$\operatorname{disc}(f(x)) = (-1)^{n(n-1)/2} \operatorname{Res}(f(x), f'(x)),$$

where $f'(x)$ denotes the derivative of $f(x)$.

Proof. Let $\alpha_1, \dots, \alpha_n$ be the roots of $f(x)$. We can compute the derivative and evaluate at a root α_i :

$$f'(x) = \sum_{i=1}^{n} \prod_{\substack{j \in \{1,\dots,n\} \\ j \ne i}} (x - \alpha_j), \qquad f'(\alpha_i) = \prod_{\substack{j \in \{1,\dots,n\} \\ j \ne i}} (x - \alpha_j).$$

By definition of the resultant, we obtain

$$\begin{aligned}\operatorname{Res}(f(x), f'(x)) &= f'(\alpha_1)\cdots f'(\alpha_n)\\ &= \prod_{i\neq j}(\alpha_i-\alpha_j)\\ &= (-1)^{n(n-1)/2}\prod_{i<j}(\alpha_i-\alpha_j)^2.\end{aligned}$$

□

3.3.1 Enumerating number fields by discriminant

One common problem in mathematics and explicit computation is to enumerate all isomorphism classes of a given structure. You can see this everywhere, for example, in the ATLAS of Finite Groups [CCN$^+$85] provides information about many simple groups. Recent efforts finalized the database of all groups of order up to 2000, except those whose order is larger than 500 and divisible by 128 in the LMFDB!

We can consider the same problem with number fields. Can we enumerate all of the possible isomorphism classes up to a given bound? The first thing you might try is to order them by degree, but this will not take us far since there are infinitely many quadratic fields. It turns out, considering the discriminant works.

Theorem 3.39 ([Her57], [Ste, Theorem 8.43])**.** *There are finitely many number fields up to isomorphism with a given discriminant.*

You can find the original article here. We will not show a proof, but the theorem follows from the bound

$$|\operatorname{disc}(K)| > \left(\frac{\pi}{4}\right)^{2s}\left(\frac{n^n}{n!}\right)^2, \tag{3.40}$$

where n is the degree $[K:\mathbb{Q}]$.

Currently, the LMFDB contains a database of 22,171,096 number fields of degree $n \leq 47$, and discriminant $\leq d_n$, where d_n depends on n (as described here).

Much research is currently done to understand the asymptotics of the number of number fields, ordered by absolute discriminant, that satisfy some given restrictions. Malle's conjecture is stated for all extensions of number fields. (In these lecture notes, we have restricted to extensions of $\mathbb{Q}$, but all of the definitions extend nicely. Taking $K=\mathbb{Q}$ should give you familiar notation.)

Conjecture 3.41 (Malle's Conjecture [Mal02])**.** *Let K be a number field and G be a transitive permutation finite group. Define*

$$Z(K,G;x) := \#\{L/K : \operatorname{Gal}(L/K) = G \text{ and } |\operatorname{Nm}_{K/\mathbb{Q}}(\operatorname{disc}(L/K))| \leq x\}. \tag{3.42}$$

Then for all $\epsilon > 0$ there exist constants $c_1(K,G)$, $c_2(K,G,e) \in \mathbb{R}_{>0}$ such that

$$c_1(K,G)x^{a(G)} \leq Z(K,G;x) < c_2(K,G,e)x^{a(G)+\epsilon},$$

for all large enough x, where $a(G)$ is a quantity depending only on the group G.

The conjecture is known in many specific cases such as abelian groups. Settling the full conjecture is an active are of research, see for example all these very recent arXiv papers: 2510.05248, 2505.23690, and 2412.04196.

You can verify the simplest example of Malle's Conjecture with your favorite computer algebra system.

Exercise 3.43. Verify that the number of quadratic fields of absolute discriminant up to x is asymptotic to $\frac{6}{\pi^2}x$. That is, as x grows, your number from (3.42) should approach $\frac{6}{\pi^2}x$. Note that in this case, G is the only group of order 2: $\mathbb{Z}/2\mathbb{Z}$ (up to isomorphism).

3.3.2 Computing integral bases using discriminants

In general determining an integral basis for a number field K is not an easy problem. It turns out that discriminant are essential for algorithms to solve this problem. The following lemma starts giving us an indication of why.

Lemma 3.44 ([Coh93, Proposition 4.4.5])**.** *The algebraic numbers $\alpha_1, \dots, \alpha_n \in K$ form an integral basis for K if and only if* $\operatorname{disc}(\alpha_1, \dots, \alpha_n) = \operatorname{disc}(K)$.

The general approach to find an integral basis is to first compute a primitive element θ such that $K = \mathbb{Q}(\theta)$ (see Remark 2.35). Then, we have that $\mathbb{Z}[\theta] \subseteq \mathcal{O}_K$. We can proceed to check if the orders are equal by computing $\operatorname{disc}(1, \theta, \dots, \theta^n)$ and checking if it has square factors. If not, Proposition 3.34 implies we have found an appropriate integral basis. If not, the way to proceed is trickier. One approach to solving this problem is known as the *Round Two Algorithm*. The main idea of the algorithm is to consider orders modulo p for all primes with p^2 dividing the discriminant of the polynomial defining K. Working now with finite structures, for each prime one can compute a maximal order modulo p and then lift to $\mathcal{O}_K$. The interested reader can find a description of Zassenhaus's Round two algorithm in [Coh93, Algorithm 6.1.8].

Exercise 3.45. Prove that the assertions in Example 3.28 are correct for $n = p$ prime. Can you do the same for composite n?

Lecture 4: LLL and the subfield problem

We now turn our attention to an essential problem when working with number fields. Our approach to solving it will draw from the material that we have already covered, but also from a powerful algorithm in linear algebra: the LLL algorithm. The materials of this lecture come from [Ste, §2.5] and [Coh93, §2.5,§2.7,§4.4].

Definition 4.1. Let K and L be two number fields. The subfield problem consists of determining whether or not K is isomorphic to a subfield of L.

4.1 Lattices

Before we can apply the LLL algorithm to detect subfields, we must first recall some definitions: lattices, positive-definite quadratic forms, and the Gram–Schmidt orthogonalization process.

4.1.1 Definitions

Definition 4.2. Let K be a field of characteristic different from 2, and let V be a K-vector space. We say that a map q from V to K is a quadratic form if the following two conditions are satisfied:

1. For every $\lambda \in K$ and $x \in V$ we have $q(\lambda \cdot x) = \lambda^2 q(x)$.

2. The function $b(x, y) := \frac{1}{2}(q(x + y) - q(x) - q(y))$ is a symmetric bilinear form.

Conversely, one can start with a bilinear form $b(x, y)$ and define the quadratic form $q(x) = b(x, x)$.

Definition 4.3. Let q be a quadratic form on an $\mathbb{R}$-vector space V. We say that q is positive-definite if for all $x \in V$ we have $q(x) > 0$.

Example 4.4. The form $q\colon \mathbb{R}^2 \to \mathbb{R}$ given by $q(x, y) = x^2 + y^2$ is a positive definite quadratic form.

Definition 4.5. A lattice is a free $\mathbb{Z}$-module of finite rank together with a positive definite quadratic form q on $L \otimes \mathbb{R}$.

Remark 4.6. It is common to abuse notation and call a free $\mathbb{Z}$-modulo a lattice, even when there is no associated quadratic form.

Example 4.7. The $\mathbb{Z}$-module $\mathbb{Z}\{(1,0),(0,1)\}$ together with the quadratic form from Example 4.4 is a lattice in $\mathbb{R}^2$.

Example 4.8. Figure 3.20 shows two different lattices in $\mathbb{R}^2$ closely related to the number field $\mathbb{Q}(\zeta_3)$.

Remark 4.9. The quadratic form q associated to a lattice L gives a notion of vector length: for $x \in L$,

$$x \cdot x := q(x) \qquad \text{and} \qquad |x| := \sqrt{x \cdot x}. \tag{4.10}$$

Remark 4.11. If L is a lattice, then $E := L \otimes \mathbb{R}$ is an $\mathbb{R}$-vector space of dimension the rank of L. Also, if $\mathcal{B} = \{b_1, \dots, b_n\}$ is a $\mathbb{Z}$-basis for L, then $\mathcal{B}$ is an $\mathbb{R}$-basis for E. This is why we can identify $L \otimes \mathbb{R}$ with $\mathbb{R}^n$.

Example 4.12. This is a generalization of Example 4.7. Let $n \geq 1$ and let e_i denote the vector with 1 in the i-th coordinate and zeroes everywhere else. Then $L := \mathbb{Z}\{e_1, \dots, e_n\}$ is a lattice of rank n. The bilinear form is given by the usual dot product in $\mathbb{R}^n$: $b(x,y) := x \cdot y$.

The advantage of Remark 4.11 is that allows us to use the methods that we know from $\mathbb{R}$-vector spaces. For example, the dot product and the Gram-Schmidt orthogonalization process, as we will see in §4.1.3.

Let L be a lattice with associated quadratic form $q \colon \mathbb{R}^n \to K$ and $\mathbb{Z}$-basis $\{b_1, \dots, b_n\}$. Then, we can recover q from the matrix

$$Q := [q_{i,j}]_{i,j}, \qquad \text{where } q_{i,j} := b(b_i, b_j), \tag{4.13}$$

with $b(b_i, b_j)$ denoting the bilinear form $b(x,y) := \frac{1}{2}(q(x+y) - q(x) - q(y))$. Indeed, for all $x, y \in \mathbb{R}^n$, we have

$$q(x) = x^{\mathsf{t}} Q x \qquad \text{and} \qquad b(x,y) = y^{\mathsf{t}} Q x.$$

Definition 4.14. Given a quadratic form q, the Gram matrix of q is the matrix Q defined in (4.13).

Exercise 4.15. Let L be a lattice with associated quadratic form q. Show that the determinant of the gram matrix of q is independent of the choice of $\mathbb{Z}$-basis for L.

Definition 4.16. Let L be a lattice with associated quadratic form q. The determinant of a lattice L is $\det(Q)$, where Q is the matrix from (4.13).

4.1.2 Example: the ring of integers of a number field

Just as an example, let us consider number fields. We will explore the main ideas of the construction of a lattice of rank $[K : \mathbb{Q}]$ from a number field K. We will skip the proofs, a good place to read about it is [Mil]. In this section, we fix K a number field with ring of integers $\mathcal{O}_K$, degree n, integral basis $\{\omega_1, \dots, \omega_n\}$, and embeddings to $\mathbb{C}$ given by $\sigma_1, \dots, \sigma_n$.

The structure theorem (Theorem 3.23) or Theorem 3.21 imply that the ring of integers of a number field K is a lattice in K of maximal rank. All that is left is to determine its rank and associated quadratic form.

Let r be the number of embeddings such that $\sigma_i(K) \subseteq \mathbb{R}$ (called real embeddings), and $s := (n-r)/2$.

Exercise 4.17. Show that if $\sigma\colon K \to \mathbb{C}$ is an embedding with image not completely contained in $\mathbb{R}$ (called a complex embedding), then there exists a different embedding, $\bar{\sigma}\colon K \to \mathbb{C}$, given by $\bar{\sigma}(\alpha) = \overline{\sigma(\alpha)}$, the complex conjugate. In particular, conclude that s defined above is an integer and that $n = r + 2s$.

Assume that $\sigma_1, \ldots, \sigma_r$ are the real embeddings of K and that the complex embeddings of K are $\sigma_{r+1}, \bar{\sigma}_{r+1}, \ldots, \sigma_{r+s}, \bar{\sigma}_{r+s}$. These embeddings produce an embedding

$$\begin{array}{llll}\Phi\colon & K & \hookrightarrow & \mathbb{R}^n \simeq \mathbb{R}^s \times \mathbb{R}^{2r} \\ & \alpha & \mapsto & (\sigma_1(\alpha), \ldots, \sigma_r(\alpha), \mathrm{Re}(\sigma_{r+1}(\alpha)), \mathrm{Im}(\sigma_{r+1}(\alpha)), \mathrm{Re}(\sigma_{r+s}(\alpha)), \mathrm{Im}(\sigma_{r+s}(\alpha))),\end{array} \tag{4.18}$$

where $\mathrm{Re}(z)$ and $\mathrm{Im}(z)$ represent the real and imaginary parts of the complex number z, respectively.

Lemma 4.19. *The image of $\mathcal{O}_K$ under the embedding from (4.18) is a lattice of rank n with the usual dot product giving the quadratic form.*

Remark 4.20. Let $\alpha\mathcal{O}_K$. Then

$$\begin{aligned}\mathrm{Tr}_{K/\mathbb{Q}}(\alpha^2) &= \sum_{i=1}^{r} \sigma_i(\alpha)^2 + \sum_{i=r+1}^{r+s} (\sigma_i(\alpha)^2 + \bar{\sigma}_i(\alpha)^2) \\ &= \sum_{i=1}^{r} \sigma_i(\alpha)^2 + 2\sum_{i=r+1}^{r+s} \mathrm{Re}(\sigma_i(\alpha))^2 + \mathrm{Im}(\sigma_i(\alpha))^2)\end{aligned}$$

4.1.3 Gram-Schmidt

The reader might be familiar with the Gram-Schmidt orthogonalization process to compute an orthogonal basis for a finite dimensional vector space equipped with an inner product $(\cdot)$. This basis can be made orthonormal by dividing the vectors by the square root of their norm. Given a basis $\{b_1, \ldots, b_n\}$, we can produce an orthogonal basis $\{b_1^*, \ldots, b_n^*\}$ as

$$b_1^* := b_1$$

and for $i \in \{2, \ldots, n\}$,

$$b_i^* := b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*, \tag{4.21}$$

where

$$\mu_{i,j} := \frac{b_i \cdot b_j^*}{b_j^* \cdot b_j^*}. \tag{4.22}$$

Remark 4.23. The basis can be made orthonormal if you are willing to divide by the norms of the resulting vectors as a last step of your computation.

4.2 The LLL Algorithm

In this section, L will be a lattice in K with a $\mathbb{Z}$-basis $\mathcal{B} := \{b_1, \dots, b_n\}$ and an orthogonal basis $\{b_1^*, \dots, b_n^*\}$ obtained from Gram-Schmidt as in §4.1.3. We are interested in *reducing* the basis $\mathcal{B}$ as much as possible. The following is the idea that Arjen Lenstra, Hendrik Lenstra, and László Lovász had in [LLL82].

Definition 4.24. The basis $\mathcal{B}$ is LLL-reduced if

$$|\mu_{i,j}| \leq \frac{1}{2} \qquad \text{for } 1 \leq j < i \leq n$$

and

$$|b_i^*|^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right) |b_{i-1}^*|^2 \qquad \text{for } 1 < i \leq n,$$

where $\mu_{i,j} := \frac{b_i \cdot b_j^*}{b_j^* \cdot b_j^*}$.

The reason why having a basis be LLL-reduced comes from the following theorem.

Theorem 4.25. *Let $\{b_1, \dots, b_n\}$ be an LLL-reduced basis of a lattice L. Then*

$$d(L) \leq \prod_{i=1}^{n} |b_i| \leq 2^{n(n-1)/4} d(L);$$

$$|b_j| \leq 2^{(i-1)/2} |b_i^*|, \qquad \textit{if } 1 \leq j < i \leq n;$$

$$|b_1| \leq 2^{(n-1)/4} d(L)^{1/n};$$

for every $x \in L$ with $x \neq 0$, we have

$$|b_1| \leq 2^{(n-1)/2} |x|; \tag{4.26}$$

and for linearly independent vectors $x_1, \dots, x_t \in L$, we have

$$|b_j| \leq 2^{(n-1)/2} \max(|x_1|, \dots, |x_t|) \qquad \textit{for } 1 \leq j \leq t.$$

Since we want to focus on the algorithm, we will skip the proof, but encourage the reader to write one on their own.

We have that (4.26) shows in particular that the vector b_1 is not far from being the shortest vector in L. If you are lucky, it might be the case that b_1 is the shortest vector. In general, finding the shortest vector in a lattice is a hard problem; in fact, many encryption protocols use this idea.

Remark 4.27. To see an example of how many branches of algebra are touched by LLL-reduced basis, we can type in Magma:

```
LLL;
```

The result is documentation for LLL intrinsics that take inputs: latices, matrices, ring orders, and ideals.

In §4.3.1 and Section 4.4, we will see applications of LLL-reduced basis to algebraic number theory, but these are not the only ones. Some other examples of applications of LLL are computing integer kernel and images of matrices and finding small vectors in lattices. Hopefully, we have enough justification for wanting to write an algorithm to compute an LLL-reduced basis.

Algorithm 4.28 (LLL Algorithm [Coh93, Algorithm 2.6.3]).

The input is a basis $b_1, \dots, b_n$ of a lattice L with quadratic form q. This algorithm transforms the vectors b_i so that when the algorithm terminates, they form an LLL-reduced basis. In addition, the algorithm gives a change of basis matrix H that writes the new basis in terms of the old one.

1. Set $k := 2$, $k_{\max} := 1$, $b_1^* = b_1$, $B_1 := b_1 \cdot b_1$, and $H := I_n$.
2. If $k \leq k_{\max}$ go to Step 3. Otherwise, set $k_{\max} := k$ and $b_k^* := b_k$. Then, for $j = 1, \dots, k-1$, set $\mu_{k,j} := b_k \cdot b_j^*/B_j$ and $b_k^* := b_k^* - \mu_{k,j} b_j^*$. Finally, set $B_k := b_k^* \cdot B_k^*$.
3. Execute RED$(k, k-1)$ (Algorithm 4.29). If $B_k < (0.75 - \mu_{k,k-1}^2) B_{k-1}$, execute SWAP$(k)$ (Algorithm 4.30). Set $k := \max(2, k-1)$ and go to Step 3. Otherwise, for $l = k-2, k-3, \dots, 1$ execute RED(k, l), then set $k := k+1$.
4. If $k \leq n$, then go to Step 2. Otherwise, output $b_1, \dots, b_n$ and the transformation H, and terminate.

The two subalgorithms are the following.

Algorithm 4.29 (Subalgorithm RED(k, l)).

1. If $|\mu_{k,l}| < 0.5$, terminate.
2. Set $r := \lfloor \mu_{k,l} \rceil = \lfloor 0.5 + \mu_{k,l} \rfloor$, the integer nearest to $\mu_{k,l}$.
3. Set $b_k := b_k - r b_l$, $H_k := H_k - r H_l$, and $\mu_{k,l} := \mu_{k,l} - r$. For all i such that $1 \leq i \leq l-1$, set $\mu_{k,i} := \mu_{k,i} - r\mu_{l,i}$ and terminate.

Remark 4.31. Note that the algorithm does not require an orthogonal basis as an input. Instead, we perform Gram-Schmidt as needed.

Algorithm 4.30 (Subalgorithm SWAP(k)).

1. Exchange the vectors b_k and b_{k-1}, H_k and H_{k-1}, and if $k > 2$, for all j such that $1 \leq j \leq k-2$ exchange $\mu_{k,j}$ with $\mu_{k-1,j}$. Then set (in this order) $\mu := \mu_{k,k-1}$, $B := B_k + \mu^2 B_{k-1}$, $\mu_{k,k-1} := \mu B_{k-1}/B$, $b := b^*_{k-1}$, $b^*_{k-1} := b^*_k + \mu b$, $b^*_k := -\mu_{k,k-1} b^*_k + (B_k/B)b$, $B_k := B_{k-1}B_k/B$, and $B_{k-1} := B$. Finally, for $i = k+2, \ldots, k_{\max}$ set $t := \mu_{i,k}$, $\mu_{i,k} := \mu_{i,k-1} - \mu t$, $\mu_{i,k-1} := t + \mu_{k,k-1}\mu_{i,k}$, and terminate the subalgorithm.

Proof of correctness of Algorithm 4.28. We need to show that the algorithm returns an LLL-reduced basis of L and that, in fact, the algorithm terminates. We can check that at the beginning of Step 4, the conditions in Definition 4.24 hold form $i \leq k-1$, so when we terminate, they hold for all elements in the basis. Moreover, the transformations of the basis all have determinant ±1, so the output is still a basis for the lattice L. □

Exercise 4.32. Show that the running time of the LLL algorithm is at most $O(n^6 \ln^3 B)$ field multiplications/divisions, if $|b_i|^2 \leq B$ for all i.

There are many variants that improve the running time of the LLL Algorithm (Algorithm 4.28). In particular, if the basis $b_1, \ldots, b_n$ is such that the Gram matrix has integer coefficients, there is a variant that returns an LLL-reduced basis with the same property.

4.3 Application: Finding algebraic dependences

Now we are ready to see a useful application of the LLL algorithm. Let $z_1, \ldots, z_n$ be complex numbers and assume that we want to find, if possible, a $\mathbb{Z}$-linear relation between these complex numbers. Otherwise, this will show that $\mathbb{Z}\{z_1, \ldots, z_n\}$ forms a lattice in $\mathbb{C}$. We will assume that none of the vectors $z_1, \ldots, z_n$ are zero.

Assume first that $z_1, \ldots, z_n \in \mathbb{R}$. For a large number N, we can consider the positive definite quadratic form

$$q(x) = x_2^2 + x_3^2 + \ldots + x_n^2 + N(z_1x_1 + z_2x_2 + \cdots + z_nx_n)^2 \tag{4.33}$$

A *short* vector with respect to this quadratic form must hold that $|z_1x_1 + z_2x_2 + \cdots + z_nx_n|$ is small and the coefficients x_i are also not too large. This implies that if the z_i are linearly dependent, picking a suitable constant N and finding a short vector will (hopefully) return a linear relation $z_1x_1 + z_2x_2 + \cdots + z_nx_n = 0$.

If the numbers $z_1, \ldots, z_n$ are complex (and not all real), then we just need to modify (4.33) to be

$$q(x) = x_3^2 + x_3^2 + \ldots + x_n^2 + N|z_1x_1 + z_2x_2 + \cdots + z_nx_n|^2.$$

and the same analysis holds.

But LLL usually helps us find short vectors! This idea gives rise to Algorithm 4.34. The last thing to figure out before we present it is the subtle choice of constant N. If $|z_i|$ are not too far from 1, then we can just pick N between $1/\epsilon$ and $1/\epsilon^2$, where ϵ is very small: if you expect the coefficients x_i to be of the order of a, then set $\epsilon := a^{-1.5n}$.

You can find a detailed example for this algorithm in Appendix A.

Algorithm 4.34 (Linear dependence [Coh93, Algorithm 2.7.4]).

The input is n complex numbers $z_1, \ldots, z_n$ and a large number N. This algorithm finds $\mathbb{Z}$-linear combinations of small modulus between the z_i. Assume that the z_i are nonzero and that if one of the ratios z_i/z_j is not real, the z_i are recorded so the ratio z_2/z_1 is not real.

1. Set $b_i := e_i$, where $e_1, \ldots, e_n$ is the standard basis for $\mathbb{Z}^n$. Set $\mu_{i,j} := 0$ for all $i \neq j$ with $3 \leq j < i \leq n$, $B_1 := |z_1|^2$, $B_2 := \operatorname{Im}(z_1\bar{z}_2)$, $B_k := 1$ for $3 \leq k \leq n$, and $\mu_{i,1} := \operatorname{Re}(z_1\bar{z}_i)/B_1$ for $2 \leq i \leq n$.

   Now if $B_2 \neq 0$ (complex case), set $\mu_{i,2} := \operatorname{Im}(z_1\bar{z}_i)/B_2$ for $3 \leq i \leq n$ and $B_2 := N \cdot B_2^2/B_1$. Otherwise (real case), set $\mu_{i,2} := 0$ for $3 \leq i \leq n$ and $B_2 := 1$.

2. Set $B_1 := NB_1$, $k := 2$, $k_{\max} := n$, and $H := I_n$. Then go to Step 3 of the LLL Algorithm (Algorithm 4.28).

3. Output the coefficients b_i as *small* coefficients of linear combinations of the z_i. The best one is probably b_1.

4.3.1 Using LLL to recognize algebraic numbers

A common way to encounter algebraic numbers is as their approximations as elements of $\mathbb{C}$. For example, consider the polynomial $f(x) = x^3 - 3x^2 + 3x - 3$. I have constructed the polynomial to have roots

$$x = 1 + \sqrt[3]{2}, \qquad x = 1 + \zeta_3\sqrt[3]{2}, \qquad x = 1 + \zeta_3^2\sqrt[3]{2}.$$

In particular, $f(x)$ is the minimal polynomial of any of the roots.

We assume now that all we know is an approximation of one of the roots

$$\omega := 2.25992104989487316476721060728,$$

and we are interested in recovering the polynomial $f(x)$ as the minimal polynomial of the root that ω is approximating. This is achievable using Algorithm 4.34. All we need to do is pick an upper bound d for the degree of the minimal polynomial and run the algorithm with the input $\{\omega, \omega^2, \ldots, \omega^d\}$.

Example 4.35. To prove that this is effective, you can try the following

```
> omega := 1 + Root(ComplexField()!2,3);
> omega;
2.25992104989487316476721060728
> time IntegerRelation([1,omega,omega^2]);
[ 7054563843, 2029924561, -2279514161 ]
Time: 0.000
> time IntegerRelation([1,omega,omega^2,omega^3]);
[ -3, 3, -3, 1 ]
Time: 0.000
```

This suggests that the minimal polynomial for ω is $x^3 - 3x^2 + 3x - 3$. You can check algebraically that this is actually the case.

Exercise 4.36. Try writing your own implementation of LLL and comparing running times for small examples.

4.4 Solving the subfield problem with LLL

Going back to the subfield problem as in Definition 4.1, recall that given two number fields K and L, we want to determine if K is isomorphic to a subfield of L. To set notation for this section, assume that $K = \mathbb{Q}(\alpha)$, $L = \mathbb{Q}(\beta)$, and that the minimal polynomials of α and β are $A(x)$ and $B(x)$, respectively. Then, all we need to determine to solve the subfield problem is if one of the conjugates of α belongs to L.

The first thing that one can check is that for the subfield problem to have a positive answer, it is necessary that the degree of $A(x)$ divides the degree of $B(x)$ (see Lemma 2.19). We assume that this test has passed.

Now, assume that the subfield problem has a positive answer: there is a conjugate of α, α_i such that $\alpha_i \in L$. Then, we should be able to write α_i in the standard basis of L with respect to β (see §2.3.2). That is, there exists a polynomial $P(x) \in \mathbb{Q}[x]$ with $P(\beta) = \alpha_i$. Hence, all we need to do is to try to find linear relations between α_i and $1, \beta, \beta^2, \ldots, \beta^{[L:\mathbb{Q}]-1}$. This is doable using Algorithm 4.34.

If a relation $P(x)$ comes out from running the algorithm, we can prove that the answer is correct by checking that $B(x)$ divides $A(P(x))$.

Remark 4.37. This approach to solving the subfield problem does not need to terminate. Indeed, the coefficients of $P(x)$ might be so large that LLL won't be able to recognize them, or $P(x)$ will not exist at all. There are (more expensive) algorithms to solve the subfield theorem deterministically (see [Coh93, §4.5] for some of them).

Lecture 5: Class and unit groups

In this lecture, we focus on the structure of the ring of integers of a number field. In particular, we will describe and compute ideals, class groups, regulators, and fundamental units. This follows [Coh93, Chapters 2, 5, and 6]. Another reference that the reader might want to look at is [EV25], on computing class groups and unit groups in Magma, but the algorithms and implementation are useful in general.

5.1 The *ring* of integers

We start by presenting the main definitions and results. Some definitions could be done in more generality, but we will focus on rings of integers. We will skip proofs, but direct the curious reader to any algebra textbook. Recall that an order (or $\mathbb{Z}$-order) R in a number field K is a subring of K which as a $\mathbb{Z}$-module is finitely generated and of maximal rank $n = [K : \mathbb{Q}]$ (Definition 3.18). The main example of an order that we have considered is the ring of integers of a number field.

Definition 5.1. An ideal I of $\mathcal{O}_K$ is a sub-$\mathbb{Z}$-module of $\mathcal{O}_K$ such that for every $r \in \mathcal{O}_K$ and $i \in I$ we have $ri \in I$.

Exercise 5.2. Verify that Definition 5.1 is equivalent to the usual definition of ideal of a ring.

Proposition 5.3. *Let I be an ideal of $\mathcal{O}_K$. Then I is a $\mathbb{Z}$-module of maximal rank. In particular, the quotient $\mathcal{O}_K/I$ is finite.*

Remark 5.4. The $\mathbb{Z}$-rank of an ideal in $\mathcal{O}_K$ equals the rank of $\mathcal{O}_K$ (since $\mathcal{O}_K$ is maximal). Also, we can describe bases for ideals (as $\mathbb{Z}$-modules) in terms of the basis for $\mathcal{O}_K$ (so the representation is given by an $n \times n$ matrix).

Example 5.5. Here are some examples of ideals, the diligent reader will show that they are actual examples.

- Let $\alpha \in \mathcal{O}_K$, the set $(\alpha) := \alpha\mathcal{O}_K$, known as the principal ideal generated by α.
- Given a collection of ideals I_j of $\mathcal{O}_K$ for $j \in \mathcal{I}$, their intersection $\bigcap_{j\in\mathcal{I}} I_j$ and their sum:

$$\sum_{j\in\mathcal{I}} I_j := \left\{ \sum_{j\in\mathcal{I}} \alpha_j \ : \ \alpha_j \in I_j \text{ and } \alpha_j = 0 \text{ for all but finitely many } j \in \mathcal{I} \right\}.$$

- For $\{\alpha_j\}_{j\in\mathcal{I}} \subseteq \mathcal{O}_K$, the set
$$(\alpha_j)_{j\in\mathcal{I}} := \sum_{j\in\mathcal{I}} (\alpha_j),$$
known as the ideal generated by $\{\alpha_j\}_{j\in\mathcal{I}}$.

- If I and J are ideals of $\mathcal{O}_K$. The set
$$IJ := \left\{ \sum_{i=1}^{n} x_i y_i \ : \ x_i \in I, y_i \in J \right\}.$$
This can be generalized for any number of ideals taking finite sums.

Definition 5.6. If every ideal of $\mathcal{O}_K$ is principal, then we say that $\mathcal{O}_K$ is a principal ideal domain.

Example 5.7. The Gaussian integers, $\mathbb{Z}[i] = \mathcal{O}_{\mathbb{Q}(i)}$, form a principal ideal domain. We will show this in Example 5.42.

Definition 5.8. Let I be an ideal of $\mathcal{O}_K$, the norm of I, denoted $\mathcal{N}(I)$ is the cardinality of R/I (note that this number is finite because of Proposition 5.3).

This relates to the norm function that we defined in Definition 3.5!

Lemma 5.9. *Let $\alpha \in K$ be nonzero. Then*
$$|\operatorname{Nm}_{K/\mathbb{Q}}(\alpha)| = \mathcal{N}(\alpha\mathcal{O}_K).$$

Some of the most useful ideals to understand are prime ideals.

Definition 5.10. An ideal $I \subseteq \mathcal{O}_K$ is prime if whenever $ab \in I$, we have $a \in I$ or $b \in I$.

Proposition 5.11. *An ideal of $\mathcal{O}_K$ is prime if and only if $\mathcal{O}_K/I$ is a field.*

Remark 5.12. For Proposition 5.11, it is essential that we are only considering prime ideals in $\mathcal{O}_K$ (a Dedekind domain).

Definition 5.13. Let I and J be ideals. We say that I divides J, and write $I|J$, if $J \subset I$.

The slogan that you want to have in mind is: “To divide is to contain”. The following proposition justifies the name *prime* for ideals.

Proposition 5.14. *If $\mathfrak{p}$ is a prime ideal of $\mathcal{O}_K$ and $\mathfrak{p} \supset I_1 \cdots I_k$, where the I_i are ideals of $\mathcal{O}_K$, then there exists some i for which $\mathfrak{p} \supset I_i$.*

In our case of ideals of $\mathcal{O}_K$, we get an even better result.

Theorem 5.15. *Let K be a number field with ring of integers $\mathcal{O}_K$. Then every ideal of $\mathcal{O}_K$ can be written as a (unique) product of prime ideals. Rings with this property are called Dedekind domains.*

Example 5.16. This is the nicest thing (in terms of factorization) that we can ask from rings of integers of number fields in general. For example, consider the number field $\mathbb{Q}(\sqrt{-5})$ with ring of integers $\mathbb{Z}[\sqrt{-5}]$. Then you can factor 9 into "primes" in two different ways (as numbers in $\mathbb{Z}[\sqrt{-5}]$):

$$9 = 3^2 = (2+\sqrt{-5})(2-\sqrt{-5}).$$

Theorem 5.15 states that this is not the case when considering ideals

$$(9) = \left(3, 1+\sqrt{-5}\right)^2 \cdot \left(3, 5+\sqrt{-5}\right)^2.$$

Example 5.17. We can use Example 5.5 to construct different ideals in a number field, and find their factorization into prime ideals.

```
> R<x> := PolynomialRing(Rationals());
> K<theta> := NumberField(x^3 - x^2 - 3*x - 3);
> OK := RingOfIntegers(K);
> I := 3*OK; //The principal ideal generated by 3
> J := ideal<OK | [OK | [2,0,0],[1,1,0]]>; //The ideal generated \
by 2 elements in OK (as coordinate vectors in the integral basis)
> Factorization(I); Factorization(J); Factorization(I*J);
```

5.1.1 Interlude: Hermite normal forms

At the heart of every computation in linear algebra, there is the unique rational canonical form of a matrix. To get them, it is essential to be able to divide any nonzero constant in the matrix. In this section, we present an analogous constructions for integer matrices representing $\mathbb{Z}$-modules.

Definition 5.18. An $m \times n$ matrix $M = (m_{i,j})$ with integer coefficients is in Hermite normal form if there exists $r \leq n$ and a strictly increasing map f from $[r+1, n]$ to $[1, m]$ satisfying:

1. For $r+1 \leq j \leq n$, $m_{f(j),j} \geq 1$; $m_{i,j} = 0$ if $i > f(j)$; and $0 \leq m_{f(k),j} < m_{f(k),k}$ if $k < j$.

2. The first r columns of M are equal zero.

Example 5.19. For $n \geq m$ and $f(k) = k$, a matrix in Hermite normal form has the following shape

$$\begin{pmatrix} 0 & 0 & \cdots & 0 & * & * & \cdots & * \\ 0 & 0 & \cdots & 0 & 0 & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & * \end{pmatrix}$$

Example 5.20. Every matrix in reduced canonical form is in Hermite normal form.

Theorem 5.21. *Let A be an $m \times n$ matrix with coefficients in $\mathbb{Z}$. Then there exists a unique $m \times n$ matrix B in Hermite normal form of the form AU with $U \in \mathrm{GL}_n(\mathbb{Z})$.*

Definition 5.22. With the notation from Theorem 5.21, the matrix W corresponding to the nonzero columns of B is called the Hermite normal form of the matrix A.

The key point of this definition is that, if the columns of A represent a set of generators for a $\mathbb{Z}$-module, then the columns of its Hermite normal form represent the unique basis for the $\mathbb{Z}$-module whose matrix is in Hermite normal form. Such a basis is called the Hermite normal form basis.

Example 5.23. A word of caution: Magma computes row-Hermite normal forms, not column. You can try doing the following, which is progress but not quite what our definition requires.

```
> A := Matrix(2,3,[1,2,3,4,5,6]);
> trB, trU := HermiteForm(Transpose(A));
> Transpose(trB) eq A*Transpose(trU) and IsInvertible(trU);
true
> Transpose(trB);
[1 0 0]
[1 3 0]
```

The columns with zeros appear on the right and the nonzero columns form a lower-triangular matrix and not upper-triangular. From that, it is clear that the form does not match the shape in Example 5.19.

Everything we say about column Hermite normal forms has an equivalent result for row Hermite normal forms, so this will not be an issue in theory, but only in practice. However, you need to be very careful on the distinction if you try to use a computer algebra system for examples.

Now we are ready to present Algorithm 5.24 to compute the Hermite normal form of a matrix with integer coefficients. As a remark, there is a way to generalize this notion to matrices with coefficients in any Dedekind domain (including, of course, rings of integers of number fields).

One useful result is the following proposition.

Proposition 5.25. *Let $M \subseteq R$ be a submodule (of maximal rank) and W be its Hermite normal form. Then the determinant of W is equal to the cardinality of R/M.*

Thus, in the particular case when you take an ideal of $\mathcal{O}_K$, the determinant of the Hermite normal form gives the norm of the ideal.

Corollary 5.26. *For any number field K and integer $N \in \mathbb{Z}_{>0}$, there are finitely many ideals in $\mathcal{O}_K$ of norm N.*

Proof. Since any ideal has maximal rank as a $\mathbb{Z}$-module, its Hermite normal form is an upper-triangular square matrix. Then, the determinant of the matrix is just the product of the diagonal entries (which are all positive). Bounding the norm bounds the diagonal entries, and this in turn bounds the other nonzero entries of the matrix. □

Algorithm 5.24 (Hermite normal form [Coh93, Algorithm 2.4.4]).

The input is an $m \times n$ matrix A with integer coefficients $(a_{i,j})$ and the output is the Hermite normal form W for A.

1. Set $i := m$, $k := n$, $l := 1$ if $m \leq n$ and $l := m - n + 1$ if $m > n$.

2. If all the $a_{i,j}$ with $j < k$ are zero, then if $a_{i,k} < 0$ replace column A_k by $-A_k$ and go to Step 5.

3. Pick among the non-zero $a_{i,j}$ for $j \leq k$ one with the smallest absolute value, say a_{i,j_0}. Then if $j_0 < k$, exchange column A_k with column A_{j_0}. In addition, if $a_{i,k} < 0$ replace column A_k by $-A_k$. Set $b := a_{i,k}$.

4. For $j = 1, \ldots, k-1$ do the following: set $q := \lfloor a_{i,j}/b \rceil$, and $A_j := A_j - qA_k$. Then go to Step 2.

5. Set $b := a_{i,k}$. If $b = 0$, set $k := k + 1$ and go to Step 6. Otherwise, for $j > k$ do the following: set $q := \lfloor a_{i,j}/b \rfloor$, and $A_j := A_j - qA_k$.

6. If $i = l$, them for $j = 1, \ldots, n - k + 1$ set $W_j := A_{j+k-1}$ and terminate the algorithm. Otherwise, set $i := i - 1$, $k := k - 1$, and go to Step 2.

5.1.2 Representing ideals

In this section we fix a number field K of degree n with primitive element θ, minimal polynomial $m_\theta(x)$, and ring of integers $\mathcal{O}_K$. By Remark 5.4, we already know that ideals can be represented as $n \times n$ integer matrices of full rank, where the columns represent coordinate vectors with respect to an integral basis for K. Then, we can take the Hermite normal form basis for the ideal using the methods from §5.1.1. We now collect some results that show the advantages of taking this representation.

Example 5.27. Let's take the number field $K = \mathbb{Q}(\theta)$ and the ideal J from Example 5.17. In this example, θ is integral, so an integral basis for K is $\{1, \theta, \theta^2\}$. The elements that generate the ideal J are 2 and $1 + \theta$. Hence, the elements in the $\mathbb{Z}$-module $J = 2\mathcal{O}_K + (1+\theta)\mathcal{O}_K$ are $\mathbb{Z}$-linear combinations of

$$\{2, 2\theta, 2\theta^2, (1+\theta), (1+\theta)\theta, (1+\theta)\theta^2\}.$$

We can use the integral basis $\{1, \theta, \theta^2\}$ to find M, a matrix representation for the generators of J with respect to this basis. Algorithm 5.24 produces the Hermite normal form W of M:

$$M = \begin{pmatrix} 2 & 0 & 0 & 1 & 0 & 3 \\ 0 & 2 & 0 & 1 & 1 & 3 \\ 0 & 0 & 2 & 0 & 1 & 2 \end{pmatrix} \qquad W = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Hence, a basis for J as a $\mathbb{Z}$-module is $\{2, 1+\theta, 1+\theta^2\}$. You can recover this same result in Magma by typing `Basis(J);`.

One advantage of using this representation is that, by Proposition 5.25, the norm of an ideal equals the determinant of the Hermite normal form of its matrix representation. So in Example 5.27, the norm of J is 2!

Another application is that, since Hermite normal forms are unique, two ideals are equal if and only if their associated Hermite normal form bases are equal.

Finally, to add ideals together, one can just take their associated $n \times n$ Hermite normal forms, W_1 and W_2, and compute Hermite normal form of the $n \times 2n$ matrix obtained by concatenation. In particular, this gives an easy way to check if an ideal I is contained in an ideal J: Compute the sum $I + J$ and check if you get J back.

Remark 5.28. There are other ways of representing ideals of $\mathcal{O}_K$. For example, any ideal of $\mathcal{O}_K$ is generated by two algebraic integers in $\mathcal{O}_K$ ([Coh93, Proposition 4.7.7]). Indeed, this is the representation that Magma uses when you ask for generators of an ideal in $\mathcal{O}_K$.

5.2 The class group

We now focus on the structure of the set of ideals of $\mathcal{O}_K$. Again, much of this story generalizes to other orders, but we focus on the ring of integers. We will give the set of ideals of $\mathcal{O}_K$ a group structure. It is clear that the identity should be $\mathcal{O}_K$ since, by definition, $I\mathcal{O}_K = I$ for any ideal I of $\mathcal{O}_K$. The problem is that we do not have inverses yet.

Definition 5.29. A fractional ideal I in $\mathcal{O}_K$ is a nonzero submodule of K such that there exists a nonzero integer d with dI ideal of $\mathcal{O}_K$. The smallest positive integer d for which this is possible is called the denominator of I.

Example 5.30. Any ideal is a fractional ideal (with denominator 1).

Exercise 5.31. Show that the set of fractional ideals of $\mathcal{O}_K$ forms a group under multiplication.

We can generalize the notion of principal ideal to fractional ideals.

Definition 5.32. An fractional ideal is principal if there is $\alpha \in K$ with $I = \alpha\mathcal{O}_K$. (Note that $\alpha \in K$ does not need to be an algebraic integer.)

Definition 5.33. Let K be a number field with ring of integers $\mathcal{O}_K$. Two fractional ideals I and J are equivalent if there exists $\alpha \in K^\times$ such that $J = \alpha I$. The set of equivalence classes of fractional ideals of K is called the class group of K and is denoted $\mathrm{Cl}(K)$.

The key of the class group of a number field is that it is a finite group.

Theorem 5.34. *For any number field K, the class group $\mathrm{Cl}(K)$ is a finite abelian group, whose cardinality, the* class number*, is denoted $h(K)$.*

The fact that $\mathrm{Cl}(K)$ is an abelian group follows easily from its definition. The content of the theorem is that this abelian group is finite. This will follow from Minkowski's bound (Theorem 5.40) , but before we present this result, we give some generalities and examples.

Remark 5.35. By definition, $h(K) = 1$ if and only if $\mathcal{O}_K$ is a principal ideal domain.

Remark 5.36. This definition can be generalized for non-maximal orders R, but in this case not every ideal is invertible, so one needs to modify the definition of the class group.

Example 5.37. Consider the number field K of degree 4 coming from the irreducible polynomial $x^4 - x^3 + 41x^2 - 26x + 436$ (with LMFDB label 4.0.167625.1).

```
> R<x> := PolynomialRing(Rationals());
> K<a> := NumberField(x^4 - x^3 + 41*x^2 - 26*x + 436);
> Cl, rho :=  ClassGroup(K);
```

In the code, Cl is an abstract finite abelian group, and rho is a map from the abstract group to the set of ideals of $\mathcal{O}_K$. The inverse of rho is also defined.

```
> Cl;
Abelian Group isomorphic to Z/2 + Z/6
Defined on 2 generators
Relations:
    2*Cl.1 = 0
    6*Cl.2 = 0
> rho(Cl.2);
Ideal of OK
Two element generators:
    [2, 0, 0, 0]
    [1, 1, 1, 0]
> I := ideal< OK | [3,4,5,6],[1,1,1,1],[4,3,3,2]>;
> (rho^(-1))(I);
0
```

In particular, we have just checked that the random ideal I is principal.

Exercise 5.38. When looking at imaginary quadratic fields, the following theorem is true.

Theorem 5.39 ([Sta67])**.** *If the imaginary quadratic field $\mathbb{Q}(\sqrt{-p})$ has class number 1 for a prime number p, then $p < 200$.*

1. List all imaginary quadratic fields with class number 1, for p prime and $p < 200$ (you might want to use a computer algebra system).

2. Conclude that your list includes all imaginary quadratic fields with class number.

Theorem 5.40 (Minkowski's bound)**.** *Let K be a number field of degree n over $\mathbb{Q}$ and discriminant* $\operatorname{disc}(K)$ *(as in Definition 3.32). Let $2s$ be the number of complex embeddings of K (just like in §4.1.2). Then every class in* $\operatorname{Cl}(K)$ *contains an ideal I (not fractional) of norm*

$$\mathcal{N}(I) \leq \sqrt{|\operatorname{disc}(K)|}\left(\frac{4}{\pi}\right)^s \frac{n!}{n^n}.$$

This is a standard result about number fields using fundamental domains of lattices. You can find a proof in [Mil, Chapter 4].

Remark 5.41. Theorem 5.34 follows directly from Corollary 5.26 and Theorem 5.40. However, note the number of ideals produced by Minkowski's bound is only an upper bound for the class number since some of the ideals in the list might still be equivalent. Note that we can further reduce the list of candidates by considering, without loss of generality, only prime ideals.

Example 5.42. As promised in Example 5.7, $\mathbb{Z}[i] = \mathcal{O}_{\mathbb{Q}(i)}$ is a principal ideal domain. The degree of $\mathbb{Q}(i)$ is 2, its discriminant is -4, and it has $2 \cdot 1$ complex embeddings. Then, Theorem 5.40 implies that every class in $\operatorname{Cl}(\mathbb{Q}(i))$ contains an ideal of norm at most

$$\sqrt{|-4|}\left(\frac{4}{\pi}\right)^1 \frac{2!}{2^2} \approx 1.27 < 2.$$

Since the norm of an ideal is a positive integer, the only option for a norm is 1. The only ideal of norm 1 is the trivial ideal $\mathcal{O}_K$, so $h(\mathbb{Q}(i)) = 1$ and every ideal of $\mathbb{Z}[i]$ is principal.

Exercise 5.43. For $K = \mathbb{Q}(\sqrt{-5})$, compute Minkowski's bound and list all ideals with norm bounded by it. Which are principal?

Using the Minkowski bound is a very reasonable first step when trying to compute class groups. There are two challenges:

1. Computing a complete list of candidate (prime) ideals of norm bounded by Minkowski's bound.

2. Finding relations in $\operatorname{Cl}(K)$ between the classes represented by the ideals in 1.

A compelling analysis of this method can be found in [EV25, Section 4]. In Section 5.3 we describe an algorithm for computing the class group for imaginary quadratic fields. For higher degree number fields, Cohen gives a general algorithm due to Diaz y Diaz, Oliver, and himself in [Coh93, §6.5].

As a final note, the problem of finding the distribution of class group structures for number fields is a major topic of current research in number theory. One of the key conjectures in this area are given by Cohen and Lenstra. In [CL84], they give heuristic conjectures (supported by data) on the structure of the class group of quadratic number fields. For example, they conjecture that the proportion of imaginary quadratic fields whose class number is divisible by an odd prime p is greater than $1/p$. They also predict that there is a positive proportion of real quadratic fields with class number one, but even showing that there are infinitely many is still an open problem!

5.3 Explicit computations: imaginary quadratic fields

To make everything explicit, we can use the theory of binary quadratic forms. I love this topic and could not stop myself from saying the basics about it. A more detailed explanation can be found in [Coh93, Section 5.2]. Some of the techniques generalize to real quadratic fields. The main advantage of working with quadratic fields is that they are determined by their discriminant and their ideals can be described using quadratic forms (which allows us to use linear algebra-like techniques).

Just to set up, we set some notation for the rest of this section. Let $K = \mathbb{Q}(\theta)$ be a quadratic number field, so that the minimal polynomial of θ is $x^2 + ax + b$. The algebraic number $\theta' = 2\theta + a$ is a root of $x^2 - (a^2 - b)$, so we may assume that any quadratic number field is of the form $\mathbb{Q}(\sqrt{d})$ for an integer d. Moreover, we can factor out any square factors of d so we can assume that $d \neq 1$ is squarefree. Finally, the discriminant of K is d is $d \equiv 1 \pmod 4$, and $4d$ if $d \equiv 2, 3 \pmod 4$.

If K is a quadratic number field of discriminant D, then we can write $K = \mathbb{Q}(\sqrt{D})$ (so, the squarefree generator is D or $D/4$). This notation is nice because we can write an integral basis for K: $\{1, \omega\}$, where $\omega = (D + \sqrt{D})/2$. In this case, we say that D is a fundamental discriminant.

Proposition 5.44. *Any ideal $I \in \mathcal{O}_K$ has a unique Hermite normal form*

$$\begin{pmatrix} a & b \\ 0 & c \end{pmatrix}$$

with respect to the integral basis $\{1, \omega\}$, where $c | a, b$ and $0 \le b < a$. Furthermore, a is the smallest positive integer in I and $\mathcal{N}(I) = ac$.

After some preliminary work, we will be able to use this representation to associate a binary quadratic form to every ideal.

Definition 5.45. An ideal I of $\mathcal{O}_K$ is primitive if $c = 1$. That is, I/n is not an ideal for any integer $n > 1$.

5.3.1 Binary quadratic forms

We have already seen the definition of a quadratic form (see Definition 4.2). For quadratic fields, we will only need binary quadratic forms: quadratic forms in two variables. We also restrict to binary quadratic forms defined over the integers, not a vector space.

Definition 5.46. A binary quadratic form is a function $f(x, y) = ax^2 + bxy + cy^2$, where $a, b, c \in \mathbb{Z}$. We usually denote the function $f(x, y)$ as (a, b, c). The discriminant of (a, b, c) is $D := b^2 - 4ac$. The binary quadratic form is primitive if $\gcd(a, b, c) = 1$. Two binary quadratic forms $f(x, y)$ and $g(x, y)$ are equivalent if there exists a matrix $\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \in \mathrm{SL}_2(\mathbb{Z})$ such that $g(x, y) = f(\alpha x + \beta y, \gamma x + \delta y)$.

Exercise 5.47. Show that two equivalent binary quadratic forms have the same discriminant and they are primitive if and only if the other one is. Can you find two quadratic forms that are not equivalent but have the same discriminant?

The key is that, to every ideal of $\mathcal{O}_K$, we can associate the binary quadratic form (a, b, c), where $a, b, c \in \mathbb{Z}$ are chosen using Proposition 5.44. Moreover, given a fractional ideal I, we can pick $d \in \mathbb{Z}_{>0}$ such that dI is an ideal. Then we associate the quadratic form (da, db, dc) to this fractional ideal.

From the description, it is easy to note that if the quadratic form is primitive, then the associated fractional ideal is an ideal of $\mathcal{O}_K$.

Definition 5.48. A positive definite binary quadratic form (a, b, c) of discriminant D is reduced if $|b| \leq a \leq c$ and if, in addition, when one of the two inequalities is an equality, then $b \geq 0$.

Proposition 5.49. *In every class of positive definite quadratic forms of discriminant $D < 0$, there exists exactly one reduced form.*

Then, counting equivalence classes of binary quadratic forms is equivalent to finding reduced forms of a given discriminant. We skip some details, but the upshot of this construction is the following theorem.

Theorem 5.50 ([Coh93, Theorem 5.2.8])**.** *Let $D < 0$ be congruent to 0 or q modulo 4. Then, $h(D) = h(\mathbb{Q}(\sqrt{D}))$ is equal to the number of primitive reduced binary positive definite quadratic forms of discriminant D.*

5.4 The unit group

One problem when considering principal ideals is that the generator of a principal in $\mathcal{O}_K$ is not unique. For example, $(\alpha) = (-\alpha)$ for any $\alpha \in K \setminus \{0\}$. Two principal ideals (α) and (β) are equal if and only if $\alpha = u\beta$, where u is a unit in $\mathcal{O}_K$. This is just one motivation for describing explicitly the set of units of the ring of integers of a number field.

Definition 5.51. The set of units in $\mathcal{O}_K$ forms a multiplicative group, denoted by $U(K)$.

Similarly to the class group, we want to describe the structure of the unit group. The first thing to note is that, unlike the class group, the unit group is not generally finite, as the following example shows.

Example 5.52. In the number field $K = \mathbb{Q}(\theta)$, where θ has minimal polynomial

$$x^4 + 2x^3 - 3x^2 - 4x + 13,$$

the element $u = (\theta^3 + 4\theta^2 + 10\theta + 6)/15$ is a unit because

$$1 = \frac{\theta^3 + 4\theta^2 + 10\theta + 6}{15} \cdot \frac{-3\theta^3 - 2\theta^2 + 10\theta - 8}{15}.$$

Moreover, under one of the complex embeddings of K, the norm (as a complex number) of u is ≈ 3.7. Because the norm is multiplicative, the multiplicative order of u cannot be finite.

However, not every element in $U(K)$ has infinite order. For example, $(-\theta^2 - \theta + 2)/3$ has multiplicative order 4.

To understand the torsion (finite order part) of the unit group, we take $u \in U(K)$ with finite order. Then there exists $m \geq 1$ such that $u^m = 1$, which implies that u is a root of unity. By definition, any unit of $\mathcal{O}_K$ that is not a root of unity must have infinite order. Let $\mu(K)$ denote the set of roots of unity in $U(K)$.

Remark 5.53. Let K be a number field. If K contains a primitive root on unity, then it must contain all of its conjugates. That implies that K contains a subfield isomorphic to the m-th cyclotomic field $\mathbb{Q}(\zeta_m)$. Hence, the degree $[\mathbb{Q}(\zeta_n) : \mathbb{Q}]$ must divide $n = [K : \mathbb{Q}]$. This shows that the set of roots of unity in a number field is finite and also gives an algorithm on how to compute it: for any possible m (finitely many), solve the subfield problem from Lecture 4 to determine if K has a subfield isomorphic to $\mathbb{Q}(\zeta_m)$. For details, see [Coh93, Algorithm 4.9.10].

Now we are ready to describe the structure of the unit group.

Theorem 5.54 (Dirichlet's Unit Theorem)**.** *Let K be a number field and let r and $2s$ be the number of real and complex embeddings, respectively. Then*

$$U(K) \simeq \mu(K) \times \mathbb{Z}^{r+s-1}.$$

Corollary 5.55. *The unit group of any imaginary quadratic field is finite.*

Proof. If K is a quadratic imaginary field, then K has no real embeddings and $2 \cdot 1$ complex embeddings. Hence, Theorem 5.54 implies that $U(K) \simeq \mu(K) \times \mathbb{Z}^{0+1-1} = \mu(k)$, so the unit group is finite. □

Exercise 5.56. Show that the unit group of a number field K is finite if and only if K is either $\mathbb{Q}$ or an imaginary quadratic field.

The following gives us language for the free part of $U(K)$.

Definition 5.57. With the notation of Theorem 5.54, a set $u_1, \dots, u_{r+s-1}$ of units of K that generates the free part of $U(K)$ is called a system of fundamental units in K.

Dirichlet's Unit Theorem (Theorem 5.54) then implies that, once we pick a system of fundamental units $\{u_i\}$, every unit u of K can be written uniquely as

$$u = \zeta u_1^{k_1} \cdots u_{r+s-1}^{k_{r+s-1}},$$

where ζ is a root of unity and the k_i are integers.

Because of Remark 5.53, the problem of describing the unit group of a number field reduces to computing a set of fundamental units. For real quadratic fields, there is a beautiful construction of the fundamental unit using continued fractions and quadratic forms (see, for example, [Coh93, Algorithm 5.7.1]).

Appendix A: Integer dependences code

This is the Magma code to use LLL explicitly to find integer dependences between real numbers, as explained in Lecture 4.

```
QQ<x> := PolynomialRing(Rationals());
RR<y> := PolynomialRing(RealField(30));
f := x^5 - 2*x^4 - 4*x^3 + 7*x^2 - 1;
vector := [-1,0,7,-4,-2,1]; //This is what we are looking for!

precision := 10^6; //just to coerce everything as integers

r := Roots(Evaluate(f,y))[1][1];
Z := [r^i : i in [0..5]]; //We look for linear relations between these

S<x1,x2,x3,x4,x5,x6> := PolynomialRing(RR,6);

B := IdentityMatrix(IntegerRing(),6); //This is the basis for the lattice.

// Quadratic form

// Pick N large enough
epsilon := 10^(-1.5*6);
N := (1/epsilon)*precision;

// This is the quadratic form
quad := &+[S.i^2: i in [2..6]] + N*( &+[Z[i]*S.i : i in [1..6]] )^2;

// The Gram matrix
function GramMatrixFromQuadratic(f)
    P := Parent(f); R := BaseRing(P); n := Rank(P);
    G := ZeroMatrix(R, n, n);

    for term in Monomials(f) do
        coeff := MonomialCoefficient(f, term);
        exps := Exponents(term);
        supp := [i : i in [1..n] | exps[i] ne 0];

        if #supp eq 1 then G[supp[1],supp[1]] +:= coeff;
        elif #supp eq 2 then
            G[supp[1],support[2]] +:= coeff / 2;
            G[supp[2],support[1]] +:= coeff / 2;
        end if;
    end for;
    return G;
end function;

G := GramMatrixFromQuadratic(quad);
// Now over the integers
GZ := Matrix(Integers(), 6, 6, [ Round(x * precision) : x in Eltseq(G) ]);

L := LatticeWithBasis(B, GZ);
Bv := Basis(LLL(L))[1];

&and[Bv[i] eq vector[i] : i in [1..6]];
print Bv;
```

Bibliography

[BZ11] Richard P. Brent and Paul Zimmermann. *Modern computer arithmetic*, volume 18 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2011. ↑1.

[CCN$^+$85] J. H. Conway, R. T. Curtis, S. P. Norton, R. A. Parker, and R. A. Wilson. $\mathbb{ATLAS}$ *of finite groups*. Oxford University Press, Eynsham, 1985. Maximal subgroups and ordinary characters for simple groups, With computational assistance from J. G. Thackray. ↑30.

[CL84] H. Cohen and H. W. Lenstra, Jr. Heuristics on class groups of number fields. In *Number theory, Noordwijkerhout 1983 (Noordwijkerhout, 1983)*, volume 1068 of *Lecture Notes in Math.*, pages 33–62. Springer, Berlin, 1984. ↑48.

[CLO15] David A. Cox, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms*. Undergraduate Texts in Mathematics. Springer, Cham, fourth edition, 2015. An introduction to computational algebraic geometry and commutative algebra. ↑ii.

[Coh93] Henri Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1993. ↑ii, 1, 7, 10, 12, 13, 19, 22, 23, 25, 31, 32, 36, 38, 39, 40, 44, 45, 47, 48, 49, 50, 51.

[Coh00] Henri Cohen. *Advanced topics in computational number theory*, volume 193 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2000. ↑ii.

[EV25] Andreas-Stephan Elsenhans and John Voight. Computing class groups and unit groups in magma, 2025. Preprint, `arXiv:2510.05501`. ↑40, 47.

[Har21] David Harvey. Counting points on hyperelliptic curves over finite fields, 2021. IAS/Park City Mathematics Series. ↑ii, 1, 2, 6, 9.

[Her57] C. Hermite. Extrait d'une lettre de m. c. hermite à m. borchardt sur le nombre limité d'irrationalités auxquelles se réduisent les racines des équations à coefficients entiers complexes d'un degré et d'un discriminant donn. *Journal für die reine und angewandte Mathematik*, 53:182–192, 1857. ↑30.

[HvdH21] David Harvey and Joris van der Hoeven. Integer multiplication in time $O(n \log n)$. *Ann. of Math. (2)*, 193(2):563–617, 2021. ↑4, 6.

[Lan94] Serge Lang. *Algebraic number theory*, volume 110 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 1994. ↑ii, 12.

[LLL82] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982. ↑35.

[Mal02] Gunter Malle. On the distribution of galois groups. *Journal of Number Theory*, 92(2):315–329, 2002. ↑30.

[Mil] J. S. Milne. Algebraic number theory. `https://www.jmilne.org/math/CourseNotes/ant.html`. ↑ii, 12, 23, 33, 47.

[Pap94] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley Publishing Company, Reading, MA, 1994. ↑2.

[Sta67] H. M. Stark. A complete determination of the complex quadratic fields of class-number one. *Michigan Math. J.*, 14:1–27, 1967. ↑46.

[Ste] William Stein. Algebraic number theory, a computational approach. `https://wstein.org/books/ant/`. ↑ii, 12, 23, 30, 32.

[Voi21] John Voight. *Quaternion algebras*, volume 288 of *Graduate Texts in Mathematics*. Springer, Cham, 2021. ↑ii.

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge University Press, Cambridge, third edition, 2013. ↑ii, 1, 7.