Intro to Number Theory

Lecture and
LaTeX by Eric Xingyu Yang
Monash University
Rosen [1], [2] are the main references

Semester one 2025

Contents

Chapter 1

Modular Arithmetic and Algebraic Structures

1.1 Division

One of the most fundamental concepts in number theory is the division of integers. Whether a number is divisible by another number? How can we determine the divisibility of two numbers? How can we express one number in terms of another? These questions lead us to the foundation of number theory, which is one of the most abstruse and beautiful branches of mathematics.

Number theory plays a crucial role in many fields:

- **Cryptography**: The security of many encryption algorithms relies on the properties of prime numbers and modular arithmetic.
- **Algebraic Geometry**: Number theory is used to study the properties of algebraic varieties over finite fields. This is crucial in coding theory and cryptography as well. This is also known as the hardest math branch encompassing almost all the math fields. An example is Elliptic Curve Cryptography(ECC).
- **Computer Science**: In computer science, number theory is used in algorithms, data structures, and complexity theory. A typical example is the design of hash functions.
- **Optimization and Operations Research**: Number theory is used in optimization problems, such as integer programming and combinatorial optimization. In some cases, we can use number theory notions to simplify the problem, such as **modular congruence** and **Diophantine equations**.

It might be trivial, since we have been using division since we were kids, but it is necessary to give division a formal definition.

Definition 1.1.1: True Division

Formally, the division of two real numbers $a, b \in \mathbb{R}$ is defined as:

$$a \div b = c$$

where c is the unique number such that:

$$a = b \times c.$$

In other words, $a \div b = c \iff a = b \times c$. The division is defined for all $a \in \mathbb{R}$ and $b \in \mathbb{R} \setminus \{0\}$.

The division is not defined for $b = 0$ because it leads to an undefined result. This is actually a very polemical topic in the history of mathematics, and gave birth to the **division by zero** paradox, contributing to the formalization of infinity, which is a milestone in the development of calculus and analysis.

While this is only an intro, in number theory, we are more interested in the division of integers. Why?

Fact 1.1.2

- **Integers are the building blocks of all numbers**: All numbers can be expressed as a combination of integers.
- **Integers are discrete**: Unlike real numbers, integers are discrete and can be counted. This makes them easier to work with in many cases.
- **Unique factorization**: Every integer can be **uniquely** factored into other numbers, which is a fundamental property of integers.

Definition 1.1.3: Division with Remainder

The division of a number a by a non-zero number b is denoted by $a \div b$ or $\frac{a}{b}$, and it gives the quotient q and possibly a remainder r. The operation can be written as:

$$a = b \times q + r$$

where $0 \leq r < b$ and $q \in \mathbb{Z}$.

Definition 1.1.4: Divisibility

If a and b are integers with $a \neq 0$, we say that a divides b if there is an integer c such that $b = a \times c$ (or equivalently, if b an is an integer). When a divides b we say that a is a factor or divisor of b, and that b is a multiple of a. The notation a | b denotes that a divides b. We write $a \nmid b$ when a does not divide b

Divisibility lays the foundation of number theory, and later we will introduce how to use

it to define modular arithmetic.

Example.

For example, we have $4 \mid 20$, because $20 \div 4$ gives an integer, however, $4 \nmid 21$, as the answer cannot be represented as integer. Besides, we have:

$$a \mid b \iff \exists c(ac = b). \tag{1.1}$$

Problem.

Prove that if a and b are integers and a divides b, then a is odd or b is even.

Proof. We could try proof by contrapositive. Suppose for a is even and b is odd, $a, b \in \mathbb{Z}$, $a \mid b$. Then $b = ka$ for some integer k. Make $a = 2n$, $b = 2n + 1$, $n \in \mathbb{Z}$, then there should be $2n + 1 = 2kn$. However, in this case we have $k = \dfrac{2n+1}{2n}$, meaning that k is not an integer, which contradict the assumption. This completes the proof. □

We may find more properties of divisibility.

Proposition 1.1.5

- **Transitive property**: If $a \mid b$ and $b \mid c$, then $a \mid c$.
  - If $a \mid b$ and $a \mid c$, then $a \mid (b + c)$.
  - If $a \mid b$ then $a \mid bc$ for any integer c.
- **Reflexive property**: For any integer a, $a \mid a$.
- **Symmetric property**: If $a \mid b$, then $b \nmid a$ unless $a = b$.
- **Trivial property**: If $a = 0$, then $0 \mid b$ for all integers b.

Proof. If $a \mid b$ and $b \mid c$, then there exist integers p and q such that $b = ap$ and $c = bq$. Substituting the expression for b into the equation for c gives $c = (ap)q = a(pq)$. Since pq is an integer, $a \mid c$.

Since $a \mid b$ and $a \mid c$, there exist integers m and n such that $b = am$ and $c = an$. Therefore, $b + c = am + an = a(m + n)$, and since $m + n$ is an integer, it follows that $a \mid (b + c)$.

Given $a \mid b$, there exists an integer k such that $b = ak$. For any integer c, $bc = a(kc)$. Since kc is an integer (because the product of two integers is an integer), $a \mid bc$.

□

Problem.

Prove that if a, b, and c are integers, where $a \neq 0$, such that $a \mid b$ and $a \mid c$, then $a \mid mb + nc$ whenever m and n are integers.

Proof. Since $a \mid b$ and $a \mid c$, we have $a \mid b + c$. Hence, we have $b + c = ak_1$ for some integer k_1. Let $b = ak_1 - c$ and $c = ak_1 - b$ in $mb + nc$, we have:

$$\begin{aligned} mb + nc &= m(ak_1 - c) + n(ak_1 - b) \\ &= ak_1(m + n) - mc - nb \\ &= a(m + n)k_1 - mc - nb \end{aligned}$$

Since $b = ak_2$ and $c = ak_3$, we have:

$$\begin{aligned} mb + nc &= a(m + n)k_1 - m(ak_2) - n(ak_3) \\ &= a(m + n)k_1 - ak_2m - ak_3n \\ &= a((m + n)k_1 - k_2m - k_3n), \end{aligned}$$

where $(m + n)k_1 - k_2m - k_3n \in \mathbb{Z}$. Hence, $a \mid mb + nc$. □

1.2 Modular Arithmetic

1.2.1 Div and Mod Operator

Notation.

div and **mod** are two operators that are used to perform division and find the remainder of a division operation, respectively.

- **div**: The div operator is used to perform integer division. It returns the quotient of the division operation, discarding any remainder. For example, $7 \div 3 = 2$ because $3 \times 2 = 6$ and the remainder is discarded.
- **mod**: The mod operator is used to find the remainder of a division operation. It returns the remainder after dividing one number by another. For example, 7 mod $3 = 1$ because $7 = 3 \times 2 + 1$.

  We also use % to denote the mod operator, which is more common in programming languages. But we will use **mod** in mathematical context.

Remark.

Note that, the **mod** operator should be bold in a printed document, since we need to differentiate it with mod , which is used for modular congruence.

You may find that the **div** and **mod** operators are actually binary and can be encapsulated by binary equations.

Definition 1.2.1: Div and Mod

Let $\mathbb{Z}$ denote the set of all integers. We define two functions as follows:

1. The integer division function, denoted $\text{div} : \mathbb{Z} \times (\mathbb{Z} \setminus \{0\}) \to \mathbb{Z}$, is defined by

$$\text{div}(a, b) = \begin{cases} \left\lfloor \frac{a}{b} \right\rfloor, & \text{if } \frac{a}{b} \geq 0, \\ \left\lceil \frac{a}{b} \right\rceil, & \text{if } \frac{a}{b} < 0, \end{cases}$$

   where $\lfloor x \rfloor$ is the floor function returning the greatest integer less than or equal to x, and $\lceil x \rceil$ is the ceiling function returning the smallest integer greater than or equal to x.

   This choice ensures that the quotient $\text{div}(a, b)$ is selected to guarantee a non-negative remainder in the corresponding modulo operation.

2. The modulo function, denoted $\text{mod} : \mathbb{Z} \times (\mathbb{Z} \setminus \{0\}) \to \mathbb{Z}$, is defined by

$$\text{mod}(a, b) = a - b \cdot \text{div}(a, b),$$

   which yields the unique remainder satisfying

$$0 \leq \text{mod}(a, b) < |b|.$$

Rounding direction of $\text{div}(a, b)$ depends on signs of a and b as summarized in the following table:

Case	a	b	$\frac{a}{b}$	$\text{div}(a, b)$ rounding
1	$+$	$+$	$+$	floor
2	$+$	$-$	$-$	ceiling
3	$-$	$+$	$-$	ceiling
4	$-$	$-$	$+$	floor

Here, "floor" means $\left\lfloor \frac{a}{b} \right\rfloor$ and "ceiling" means $\left\lceil \frac{a}{b} \right\rceil$.

These definitions ensure that for any integers a, b with $b \neq 0$, the division algorithm holds:

$$a = b \cdot \text{div}(a, b) + \text{mod}(a, b),$$

with the remainder $\text{mod}(a, b)$ uniquely satisfying

$$0 \leq \text{mod}(a, b) < |b|.$$

Let's try to prove them!

Lemma 1.2.2: Range of Mod Operation

For any integers a and b with $b \neq 0$, the modulo operation satisfies:

$$\mathrm{mod}(a, b) \in [0, |b|),$$

This means that the result of the modulo operation is always a non-negative integer less than the absolute value of b.

Proof for Lemma

According to the Division Algorithm, for any integer a and non-zero integer b, there exist unique integers q and r such that:

$$a = bq + r,$$

and the remainder satisfies

$$0 \leq r < |b|.$$

Here, we let

$$q = \mathrm{div}(a, b), \quad r = \mathrm{mod}(a, b).$$

This means that the value of the modulo operation $\mathrm{mod}(a, b)$ is the remainder r, satisfying the above inequality. By definition, we have

$$\mathrm{mod}(a, b) = a - b \cdot \mathrm{div}(a, b).$$

To ensure $0 \leq \mathrm{mod}(a, b) < |b|$, we need to choose a suitable quotient so that the remainder falls within this interval. In the definition based on the floor function, when $b > 0$, selecting $\mathrm{div}(a, b) = \left\lfloor \frac{a}{b} \right\rfloor$ can meet the requirement. But when $b < 0$, $\left\lfloor \frac{a}{b} \right\rfloor$ may not satisfy this condition. Therefore, to ensure that the remainder is non-negative and less than $|b|$, when $b < 0$, we should define

$$\mathrm{div}(a, b) = \left\lceil \frac{a}{b} \right\rceil,$$

such that

$$0 \leq a - b \cdot \mathrm{div}(a, b) < |b|.$$

In summary, regardless of whether b is positive or negative, there always exists a suitable integer quotient $\mathrm{div}(a, b)$ such that the remainder $\mathrm{mod}(a, b)$ satisfies

$$0 \leq \mathrm{mod}(a, b) < |b|.$$

This completes the proof. ∎

Problem.

Prove that for any integers a and b with $b \neq 0$,

$$a = b \cdot \mathrm{div}(a, b) + \mathrm{mod}(a, b).$$

Proof. By the division algorithm, for any integers a and b with $b \neq 0$, there exist unique integers q and r such that

$$a = bq + r,$$

where

$$0 \leq r < |b|.$$

Here, $\text{div}(a, b)$ is defined as the quotient q, and $\text{mod}(a, b)$ is defined as the remainder r. Hence, by definition,

$$a = b \cdot \text{div}(a, b) + \text{mod}(a, b),$$

with

$$0 \leq \text{mod}(a, b) < |b|.$$

This completes the proof. □

Problem.

Use the div and mod functions from the division algorithm to define the following three Boolean functions:

AND Function: Define the AND operation for Boolean variables $x, y \in \{0, 1\}$, i.e.,

$$\text{and}(x, y) = ?$$

OR Function: Define the OR operation for Boolean variables $x, y \in \{0, 1\}$, i.e.,

$$\text{or}(x, y) = ?$$

XOR Function: Define the XOR operation for Boolean variables $x, y \in \{0, 1\}$, i.e.,

$$\text{xor}(x, y) = ?$$

Please define these functions using expressions containing only div and mod.

Solution

Assuming $x, y \in \{0, 1\}$, consider the sum $x + y$ which can be 0, 1, or 2. Using the division algorithm with divisor 2, we can represent $x + y$ as:

$$x + y = 2 \cdot \text{div}(x + y, 2) + \text{mod}(x + y, 2).$$

1. **AND Function** The AND operation corresponds to whether both inputs are 1, which happens only if the sum is 2:

   $$\text{and}(x, y) = \text{div}(x + y, 2).$$

   This yields 1 only when $x = y = 1$.

Figure 1.1: Truth Table for AND: $\text{and}(x, y) = \text{div}(x + y, 2)$

x	y	$x + y$	$\text{and}(x, y)$
0	0	0	$\text{div}(0, 2) = 0$
0	1	1	$\text{div}(1, 2) = 0$
1	0	1	$\text{div}(1, 2) = 0$
1	1	2	$\text{div}(2, 2) = 1$

2. **OR Function** Using De Morgan's law, the OR can be written in terms of AND as:

$$\text{or}(x, y) = 1 - \text{and}(1 - x, 1 - y).$$

Using the expression for AND above:

$$\text{or}(x, y) = 1 - \text{div}\big(2 - (x + y), 2\big).$$

This expression is 1 if either input is 1, and 0 only when both are 0.

Figure 1.2: Truth Table for OR: $\text{or}(x, y) = 1 - \text{div}(2 - (x + y), 2)$

x	y	$x + y$	$2 - (x + y)$	$\text{div}(2 - (x + y), 2)$	$\text{or}(x, y)$
0	0	0	2	1	$1 - 1 = 0$
0	1	1	1	0	$1 - 0 = 1$
1	0	1	1	0	$1 - 0 = 1$
1	1	2	0	0	$1 - 0 = 1$

3. **XOR Function** The XOR operation corresponds to the parity of $x + y$:

$$\text{xor}(x, y) = \text{mod}(x + y, 2).$$

Intuitively, this gives 1 when exactly one of x, y is 1, and 0 otherwise.

Figure 1.3: Truth Table for XOR: $\text{xor}(x, y) = \text{mod}(x + y, 2)$

x	y	$x + y$	$\text{xor}(x, y)$
0	0	0	$\text{mod}(0, 2) = 0$
0	1	1	$\text{mod}(1, 2) = 1$
1	0	1	$\text{mod}(1, 2) = 1$
1	1	2	$\text{mod}(2, 2) = 0$

1.2.2 Modular Congruence

With basic notion of modular arithmetic, we can define the modular congruence. This is one of the most important relations in mathematics, and it is widely used in number theory, cryptography, and computer science.

Definition of Modular Congruence

Definition 1.2.3: Modular Congruence

Let $a, b \in \mathbb{Z}$ and $m \in \mathbb{N}$ with $m > 0$. We say that a is **congruent** to b **modulo** m if m divides the difference $a - b$; that is, $m \mid (a - b)$, or $a - b = k \cdot m, k \in \mathbb{Z}$. This relationship is denoted by

$$a \equiv b \pmod{m}.$$

Or equivalently, we can say that a and b are congruent modulo m if they have the same remainder when divided by m. In this case, we can also write:

$$a \mod m = b \mod m.$$

We call $a \equiv b \pmod{m}$ a **congruence**, and m is called the **modulus** (plural: moduli). If a and b are not congruent modulo m, we write $a \not\equiv b \pmod{m}$.

Problem.

Prove that $a \equiv b \pmod{m}$, if an only if $a \mod m = b \mod m$.

Proof. We need to prove both directions of the equivalence.

(1) If $a \equiv b \pmod{m}$, then $a \mod m = b \mod m$:

Assume $a \equiv b \pmod{m}$. By definition, this means that m divides the difference $a - b$, i.e., there exists an integer k such that:

$$a - b = k \cdot m.$$

Rearranging gives:

$$a = b + k \cdot m.$$

Taking modulo m on both sides, we have:

$$a \mod m = (b + k \cdot m) \mod m.$$

Since $k \cdot m$ is a multiple of m, it contributes 0 to the modulo operation. Thus:

$$a \mod m = b \mod m.$$

(2) If $a \mod m = b \mod m$, then $a \equiv b \pmod{m}$:

Assume $a \mod m = b \mod m$. This means that both a and b leave the same remainder

when divided by m. Therefore, we can write:

$$a = q_a \cdot m + r,$$

and

$$b = q_b \cdot m + r,$$

where q_a and q_b are the respective quotients when dividing by m, and r is the common remainder. Subtracting these two equations gives:

$$a - b = (q_a - q_b) \cdot m.$$

This shows that m divides the difference $a - b$, which means:

$$a \equiv b \pmod{m}.$$

Hence, we have shown both directions of the equivalence, completing the proof. □

Properties of Modular Congruence

Lemma 1.2.4

Let m be a positive integer. The integers a and b are congruent modulo m if and only if there is an integer k such that $a = b + km$.

Proof. Assume $a \equiv b \pmod{m}$. By definition, this means that m divides the difference $a - b$, i.e., there exists an integer k such that:

$$a - b = k \cdot m.$$

Rearranging gives:

$$a = b + k \cdot m.$$

This shows that if $a \equiv b \pmod{m}$, then there exists an integer k such that $a = b + km$.

Conversely, if there exists an integer k such that $a = b + km$, then we can rearrange this to get:

$$a - b = k \cdot m.$$

This shows that m divides the difference $a - b$, which means:

$$a \equiv b \pmod{m}.$$

Hence, we have shown both directions of the equivalence, completing the proof. □

Theorem 1.2.5: Linearity of Congruences

Let m be a positive integer. If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then the following properties hold:

$$a \pm c \equiv b \pm d \pmod{m},$$

$$a \cdot c \equiv b \cdot d \pmod{m}.$$

Proof. By the previous lemma, we know that if $a \equiv b \pmod{m}$, then there exists an integer k_1 such that $a = b + k_1 m$. Similarly, if $c \equiv d \pmod{m}$, then there exists an integer k_2 such that $c = d + k_2 m$. Now, we can substitute these expressions into the left-hand side of the congruences we want to prove.

$$\begin{aligned} a + c &= (b + k_1 m) + (d + k_2 m) \\ &= (b + d) + (k_1 + k_2)m \\ &\equiv b + d \pmod{m}. \end{aligned}$$

Similarly, for the subtraction:

$$\begin{aligned} a - c &= (b + k_1 m) - (d + k_2 m) \\ &= (b - d) + (k_1 - k_2)m \\ &\equiv b - d \pmod{m}. \end{aligned}$$

For multiplication:

$$\begin{aligned} a \cdot c &= (b + k_1 m)(d + k_2 m) \\ &= bd + (bk_2 + dk_1)m + k_1 k_2 m^2 \\ &\equiv bd \pmod{m}. \end{aligned}$$

Thus, we have shown that if $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then:

$$a \pm c \equiv b \pm d \pmod{m},$$

and

$$a \cdot c \equiv b \cdot d \pmod{m}.$$

Therefore, the theorem is proved. □

Theorem 1.2.6: Modular Arithmetic

Let $a, b, c \in \mathbb{Z}$ and $m \in \mathbb{N}$ with $m > 0$. Then the following properties hold:

1. If $a \equiv b \pmod{m}$, then $a + c \equiv b + c \pmod{m}$.
2. If $a \equiv b \pmod{m}$, then $a - c \equiv b - c \pmod{m}$.
3. If $a \equiv b \pmod{m}$, then $a \cdot c \equiv b \cdot c \pmod{m}$.

Proof. We can prove these using similar strategies as in the previous theorem.

1. Assume $a \equiv b \pmod{m}$. By definition, this means that $m \mid (a-b)$, i.e., there exists an integer k_1 such that:
$$a - b = k_1 m.$$
Adding c to both sides gives:
$$a + c - (b + c) = k_1 m,$$
which implies $m \mid ((a+c)-(b+c))$. Thus, $a + c \equiv b + c \pmod{m}$.
2. The proof for subtraction follows similarly. If $a \equiv b \pmod{m}$, then:
$$a - c - (b - c) = (a - b) + 2c = k_1 m + 2c,$$
which implies $m \mid ((a-c)-(b-c))$. Thus, $a - c \equiv b - c \pmod{m}$.
3. For multiplication, if $a \equiv b \pmod{m}$, then:
$$a \cdot c - b \cdot c = (a - b)c = k_1 mc,$$
which implies $m \mid (a \cdot c - b \cdot c)$. Thus, $a \cdot c \equiv b \cdot c \pmod{m}$.

Therefore, all three properties are proved. □

Remark.

> These two theorems are very useful in modular arithmetic, and on top of that, we can actually take a glimpse of the algebraic significance of the modular arithmetic. If you know basic algebra, you may find that the modular congruence m is actually an equivalence relation, and the set of integers modulo m forms a ring. This explains why we can copy what we do in basic arithmetic on integers to modular arithmetic.

Modular Congruence as Relation

Claim: Modular Congruence is Equivalence

Let $a, b, c \in \mathbb{Z}$ and $m \in \mathbb{N}$ with $m > 0$. The relation $a \equiv b \pmod{m}$ is an equivalence relation on the set of integers. Specifically, it satisfies the following properties:

1. **Reflexivity**: For any integer a, $a \equiv a \pmod{m}$.
2. **Symmetry**: If $a \equiv b \pmod{m}$, then $b \equiv a \pmod{m}$.
3. **Transitivity**: If $a \equiv b \pmod{m}$ and $b \equiv c \pmod{m}$, then $a \equiv c \pmod{m}$.

Proof for Claim.

> To show that $a \equiv b \pmod{m}$ is an equivalence relation, we need to prove the three properties.
>
> 1. **Reflexivity**: For any integer a, we have $a - a = 0$, and since $m \mid 0$, it follows that $a \equiv a \pmod{m}$.

2. **Symmetry**: If $a \equiv b \pmod{m}$, then by definition, there exists an integer k such that $a - b = km$. Rearranging gives $b - a = -km$, which shows that $m \mid (b - a)$. Thus, $b \equiv a \pmod{m}$.

3. **Transitivity**: If $a \equiv b \pmod{m}$ and $b \equiv c \pmod{m}$, then there exist integers k_1 and k_2 such that:

$$a - b = k_1 m,$$

and

$$b - c = k_2 m.$$

Adding these two equations gives:

$$a - c = (a - b) + (b - c) = (k_1 + k_2)m.$$

This shows that $m \mid (a - c)$, which implies $a \equiv c \pmod{m}$.

Thus, we have shown that $a \equiv b \pmod{m}$ is an equivalence relation.

We can define a **congruence class** as the set of all integers that are congruent to a given integer a modulo m. These equivalence classes are very useful in mathematics, and thus mathematicians give them a special name: **residue classes** (or congruence class), as a special case of equivalence classes. To recap on the definition of equivalence classes:

Definition 1.2.7: Equivalence Class

Let R be an equivalence relation on a set S. For any element $a \in S$, the equivalence class of a under the relation R is defined as:

$$[a] = \{x \in S : xRa\}.$$

The set of all equivalence classes forms a partition of the set S.

Substituting the relation in previous definition with modular congruence m, we have:

Definition 1.2.8: Residue/Congruence Class

Let $m \in \mathbb{N}$ with $m > 0$. The residue class of an integer a modulo m is defined as:

$$[a]_m = \{x \in \mathbb{Z} : x \equiv a \pmod{m}\}.$$

The set of all residue classes modulo m forms a partition of the integers $\mathbb{Z}$.

Example.

For example, the residue classes modulo 3 are:

$$[0]_3 = \{\ldots, -6, -3, 0, 3, 6, \ldots\},$$

and

$$[1]_3 = \{\ldots, -5, -2, 1, 4, 7, \ldots\},$$

$$[2]_3 = \{\ldots, -4, -1, 2, 5, 8, \ldots\}.$$

Problem.

Let m be a non-zero integer. Let $\equiv \pmod m$ be the congruence relation defined on $\mathbb{Z}$ by $a \equiv b \pmod m$ if and only if m divides $a - b$. Prove that the equivalence classes of $\mathbb{Z}$ under this relation, namely the congruence classes modulo m, form a partition of $\mathbb{Z}$.

Proof. We want to prove that the set of congruence classes modulo m

$$\mathcal{C} = \{[0]_m, [1]_m, \ldots, [m-1]_m\},$$

forms a partition of the set of integers $\mathbb{Z}$. According to the definition of a partition, we need to verify the following three points:

1. Non-emptiness: For each $r \in \{0, 1, \ldots, m-1\}$, the set $[r]_m$ is non-empty.
2. Pairwise disjointness: If $a, b \in \{0, 1, \ldots, m-1\}$ and $a \neq b$, then $[a]_m \cap [b]_m = \emptyset$.
3. Union covers the set: $\bigcup_{r=0}^{m-1} [r]_m = \mathbb{Z}$.

1. Non-emptiness: For any $r \in \{0, 1, \ldots, m-1\}$, it is clear that $r \in [r]_m$, so $[r]_m \neq \emptyset$.

2. Pairwise disjointness: Assume there exist $a, b \in \{0, 1, \ldots, m-1\}$, with $a \neq b$, such that $[a]_m \cap [b]_m \neq \emptyset$. Then there exists $x \in \mathbb{Z}$ such that

$$x \equiv a \pmod m \quad \text{and} \quad x \equiv b \pmod m.$$

From this, we get

$$m \mid (x - a) \quad \text{and} \quad m \mid (x - b).$$

Therefore, m also divides their difference:

$$m \mid \big((x - a) - (x - b)\big) = b - a.$$

Since $a, b \in \{0, 1, \ldots, m-1\}$ and $a \neq b$, but $m \mid (b - a)$ implies $b \equiv a \pmod m$, which means

$$a = b,$$

a contradiction to the assumption. Thus, $[a]_m \cap [b]_m = \emptyset$ holds for any $a \neq b$.

3. Union covers the set: For any $n \in \mathbb{Z}$, by the division algorithm, there exist unique

integers $q \in \mathbb{Z}$ and $r \in \{0, 1, \ldots, m-1\}$ such that

$$n = qm + r.$$

This is equivalent to

$$n \equiv r \pmod{m},$$

which means

$$n \in [r]_m.$$

Therefore,

$$\mathbb{Z} = \bigcup_{r=0}^{m-1} [r]_m.$$

In summary, the set of congruence classes modulo m forms a partition of the set of integers $\mathbb{Z}$. □

1.2.3 Arithmetic Modulo m

Now we can formally define the arithmetic operations on the residue classes modulo m. The operations are defined as follows:

Definition 1.2.9: Arithmetic Modulo m

Let $m \in \mathbb{N}$ with $m > 0$. The arithmetic operations on the residue classes modulo m are defined as follows:

1. Addition: For any $[a]_m, [b]_m \in \mathcal{C}$, define

$$[a]_m + [b]_m = [a + b]_m.$$

2. Subtraction: For any $[a]_m, [b]_m \in \mathcal{C}$, define

$$[a]_m - [b]_m = [a - b]_m.$$

3. Multiplication: For any $[a]_m, [b]_m \in \mathcal{C}$, define

$$[a]_m \cdot [b]_m = [a \cdot b]_m.$$

Remark.

Note that the operations are well-defined, meaning that the result does not depend on the choice of representatives from the equivalence classes. This is because if $[a]_m = [a']_m$ and $[b]_m = [b']_m$, then:

$$a - a' \equiv 0 \pmod{m} \quad \text{and} \quad b - b' \equiv 0 \pmod{m}.$$

Hence,

$$[a + b]_m = [a' + b']_m,$$

and similarly for subtraction and multiplication.

All the algebraic properties of addition and multiplication hold in the residue classes modulo m. Specifically, the following properties hold:

Let $m \in \mathbb{N}$ with $m > 0$, $\mathcal{C} = \{[0]_m, [1]_m, \ldots, [m-1]_m\}$ be the set of residue classes modulo m. The arithmetic operations on the residue classes modulo m satisfy the following properties:

1. Closure: For any $[a]_m, [b]_m \in \mathcal{C}$, both $[a]_m + [b]_m$ and $[a]_m \cdot [b]_m$ are also in $\mathcal{C}$.

2. Associativity: For any $[a]_m, [b]_m, [c]_m \in \mathcal{C}$,

   $$([a]_m + [b]_m) + [c]_m = [a]_m + ([b]_m + [c]_m),$$

   and

   $$([a]_m \cdot [b]_m) \cdot [c]_m = [a]_m \cdot ([b]_m \cdot [c]_m).$$

3. Commutativity: For any $[a]_m, [b]_m \in \mathcal{C}$,

   $$[a]_m + [b]_m = [b]_m + [a]_m,$$

   and

   $$[a]_m \cdot [b]_m = [b]_m \cdot [a]_m.$$

4. Distributivity: For any $[a]_m, [b]_m, [c]_m \in \mathcal{C}$,

   $$[a]_m([b]_m + [c]_m) = ([a]_m[b]_m + [a]_m[c]_m).$$

You may try to prove these properties as an exercise. The proof is similar to the proof of the properties of addition and multiplication in the integers, but we need to be careful about the equivalence classes.

Definition 1.2.10: Well-defined Operation

Let S be a set whose elements are equivalence classes of some underlying set X with an equivalence relation $\sim$. We say a binary operation $*$ on S is **well-defined** if for any $[x], [y] \in S$ (where $[x]$ and $[y]$ denote the equivalence classes containing x and y respectively), the result $[x] * [y]$ depends *only* on the equivalence classes $[x]$ and $[y]$ themselves, and *not* on the particular choice of representatives used to define or compute it.
More formally, suppose the operation $*$ on S is defined by first choosing representatives $x' \in [x]$ and $y' \in [y]$ and then using an operation $\star$ from the underlying set X (or related to X), such that $[x] * [y] := [x' \star y']$. For this definition to be well-defined, it must be true that if x_1, x_2 are any two representatives of $[x]$ (i.e., $[x_1] = [x_2]$) and y_1, y_2 are any two representatives of $[y]$ (i.e., $[y_1] = [y_2]$), then the result obtained using x_1, y_1 must be the same as the result obtained using x_2, y_2:

$$[x_1 \star y_1] = [x_2 \star y_2]$$

Example.

For example, the addition and multiplication operations on the residue classes modulo m are well-defined. This means that if $[a]_m = [a']_m$ and $[b]_m = [b']_m$, then:

$$[a]_m + [b]_m = [a' + b']_m,$$

and

$$[a]_m \cdot [b]_m = [a' \cdot b']_m.$$

1.3 Algebraic Structures

This section introduces some basic ideas in algebraic structures, which are fundamental concepts in abstract algebra. We will discuss groups, rings, and fields, which are important structures in mathematics, with modular congruence as a motivating example.

1.3.1 Quotient Sets

Definition 1.3.1: Quotient Set

Let S be a set and R be an equivalence relation on S. The quotient set S/R is the set of all equivalence classes of S under the relation R. It is denoted as:

$$S/R = \{[x] : x \in S\}.$$

Example.

Recall in the previous section, we see $\mathcal{C} = \{[0]_m, [1]_m, \ldots, [m-1]_m\}$ forms a set of all equivalence classes of integers modulo m in the set of integers $\mathbb{Z}$. This is a quotient set, and we can denote it as $\mathbb{Z}/m\mathbb{Z}$ or simply $\mathbb{Z}/m$.

Problem.

We say a relation or an operation is **well-defined** if it is **only determined by the objects to which it is applied**, and not by the particular representatives of those objects. Prove that the addition and multiplication operations on the residue classes modulo m on $\mathbb{Z}$ are well-defined. Namely, show that (a' and b' are representatives of $[a]_m$ and $[b]_m$ respectively):

$$[a]_m + [b]_m = [a' + b']_m,$$

$$[a]_m \cdot [b]_m = [a' \cdot b']_m.$$

Note the operations on the left-hand side are defined sequentially, for example,

$$\{1, 2, 3\} + \{4, 5, 6\} = \{5, 7, 9\}.$$

Proof. To show that the addition and multiplication operations on the residue classes modulo m are well-defined, we need to prove that if $[a]_m = [a']_m$ and $[b]_m = [b']_m$, then:

$$[a]_m + [b]_m = [a' + b']_m,$$

and

$$[a]_m \cdot [b]_m = [a' \cdot b']_m.$$

Assume $[a]_m = [a']_m$ and $[b]_m = [b']_m$. By definition of congruence classes, this means:

$$a - a' \equiv 0 \pmod{m} \quad \text{and} \quad b - b' \equiv 0 \pmod{m}.$$

This implies there exist integers k_1, k_2 such that:

$$a - a' = k_1 m,$$

and

$$b - b' = k_2 m.$$

Adding these two equations gives:

$$a + b - (a' + b') = (k_1 + k_2)m.$$

This shows that $m \mid ((a + b) - (a' + b'))$, which means:

$$a + b \equiv a' + b' \pmod{m}.$$

Hence, we have:

$$[a]_m + [b]_m = [a + b]_m.$$

A similar argument can be applied for multiplication. If $[a]_m = [a']_m$ and $[b]_m = [b']_m$, then:

$$a - a' = k_1 m,$$

and

$$b - b' = k_2 m.$$

The product gives:

$$a \cdot b - a' \cdot b' = (a - a')b + a'(b - b').$$

Both terms on the right-hand side are multiples of m, so we can conclude that:

$$a \cdot b \equiv a' \cdot b' \pmod{m}.$$

Thus, we have:

$$[a]_m \cdot [b]_m = [a \cdot b]_m.$$

Therefore, the addition and multiplication operations on the residue classes modulo m are well-defined. □

We see that we can extend similar algebraic and arithmetic properties from one set to another different set that share some common nature. This is the core idea of algebraic structures, where we abstract the properties of a set and define new sets with similar properties. The most basic algebraic structures are groups, rings, and fields.

Definition 1.3.2: Algebraic Structure

An algebraic structure consists of a set S along with one or more operations defined on that set. The operations must satisfy certain properties, which can include closure, associativity, commutativity, and the existence of identity and inverse elements.

Mathematicians study these structures to understand the underlying properties and relationships between different sets and operations. Some of them are widely discovered and studied, these include:

- **Groups**: A set with a single binary operation that satisfies closure, associativity, identity, and invertibility.
- **Rings**: A set with two binary operations (addition and multiplication) that satisfy certain properties, including distributivity.
- **Fields**: A set with two binary operations (addition and multiplication) that satisfy all the properties of a ring, along with the existence of multiplicative inverses for non-zero elements.

1.3.2 Groups

Definition 1.3.3: Group

A group is a set G equipped with a binary operation $*$ that satisfies the following properties:

1. **Closure**: For all $a, b \in G$, $a * b \in G$.
2. **Associativity**: For all $a, b, c \in G$, $(a * b) * c = a * (b * c)$.
3. **Identity Element**: There exists an element $e \in G$ such that for all $a \in G$, $e * a = a * e = a$.
4. **Inverse Element**: For each $a \in G$, there exists an element $b \in G$ such that $a * b = b * a = e$.

The notation $(G, *)$ is often used to denote a group, where G is the set and $*$ is the operation. We say that G is an **abelian group** if the operation $*$ is commutative, i.e., for all $a, b \in G$, $a * b = b * a$.

Example.

- The set of integers $\mathbb{Z}$ under addition is a group. The identity element is 0, and the inverse of any integer a is $-a$.
- The set of non-zero rational numbers $\mathbb{Q}^*$ under multiplication is also a group. The identity element is 1, and the inverse of any non-zero rational number a is $\frac{1}{a}$.
- The addition of integers modulo m forms a group. The identity element is $[0]_m$, and the inverse of any element $[a]_m$ is $[m - a]_m$.

Proposition 1.3.4

Let $\mathbb{Z}_m$ be the set of integers modulo m under addition. Prove that $(\mathbb{Z}_m, +)$ is a group.

Proof for Proposition.

To show that $(\mathbb{Z}_m, +)$ is a group, we need to verify the four properties of a group:

1. **Closure**: For any $[a]_m, [b]_m \in \mathbb{Z}_m$, we have:

$$[a]_m + [b]_m = [a + b]_m.$$

   Since $a + b$ is an integer, $[a + b]_m$ is also in $\mathbb{Z}_m$. Thus, closure holds.

2. **Associativity**: For any $[a]_m, [b]_m, [c]_m \in \mathbb{Z}_m$, we have:

$$([a]_m + [b]_m) + [c]_m = [a + b + c]_m = [a]_m + ([b]_m + [c]_m).$$

Thus, associativity holds.

3. **Identity Element**: The identity element in $(\mathbb{Z}_m, +)$ is $[0]_m$. For any $[a]_m \in \mathbb{Z}_m$:

$$[a]_m + [0]_m = [a+0]_m = [a]_m.$$

Thus, the identity element exists.

4. **Inverse Element**: For any $[a]_m \in \mathbb{Z}_m$, the inverse element is $[-a]_m = [(m-a)]_m$. We have:

$$[a]_m + [-a]_m = [(a-a)]_m = [0]_m.$$

Thus, the inverse element exists.

Since all four properties hold, we conclude that $(\mathbb{Z}_m, +)$ is a group.

Remark.

Or we may just reuse the conclusion that addition on $\mathbb{Z}$ is a group, and the addition on $\mathbb{Z}_m$ is a subset of $\mathbb{Z}$, so it preserves the group structure and properties.
In a nutshell, if a bigger structure have a certain property, then any subset of it will also have the same property. This is a very common and useful idea in mathematics, and we will see more of it in the future.

Problem.

Let $(G, \cdot)$ be a group such that $x^2 = e$ for all $x \in G$, where e is the identity element of G. Prove that G is an Abelian group. That is, prove that $xy = yx$ for all $x, y \in G$.

Proof. To show that G is an abelian group, we need to prove that $xy = yx$ for all $x, y \in G$.
Let $x, y \in G$. We know that $x^2 = e$ and $y^2 = e$. We can compute:

$$xyxy = (xy)(xy) = x(yx)y.$$

Now, using the property $x^2 = e$, we can rewrite this as:

$$e = xyxy = x(yx)y.$$

Multiplying both sides by y^{-1} (which exists since G is a group), we get:

$$y^{-1}e = y^{-1}(xyxy) = y^{-1}(x(yx)y).$$

Since $y^{-1}e = y^{-1}y^2 = e$, we have:

$$e = y^{-1}(x(yx)y).$$

This implies that:

$$x(yx) = e.$$

Since $x^2 = e$, we can also write:

$$x(yx) = xyx.$$

Hence, we have:

$$e = xyx.$$

This means that:

$$yx = xy,$$

proving that G is an abelian group. □

1.3.3 Rings

Definition 1.3.5: Ring

A ring is a set R equipped with two binary operations $+$ and $\cdot$ that satisfy the following properties:

1. **Additive Closure**: For all $a, b \in R$, $a + b \in R$.
2. **Additive Associativity**: For all $a, b, c \in R$, $(a + b) + c = a + (b + c)$.
3. **Additive Identity**: There exists an element $0_R \in R$ such that for all $a \in R$, $a + 0_R = 0_R + a = a$.
4. **Additive Inverse**: For each $a \in R$, there exists an element $-a \in R$ such that $a + (-a) = 0_R$.
5. **Multiplicative Closure**: For all $a, b \in R$, $a \cdot b \in R$.
6. **Multiplicative Associativity**: For all $a, b, c \in R$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
7. **Distributive Property**: For all $a, b, c \in R$: - Left Distributive: $a(b + c) = ab + ac$ - Right Distributive: $(a + b)c = ac + bc$

A ring is called a commutative ring if the multiplication operation is commutative, i.e., for all $a, b \in R, ab = ba$. A ring with unity is called an unital ring if it contains an identity element for multiplication.

Example.

> The set of integers $\mathbb{Z}$ under addition and multiplication is a ring. The additive identity is 0, and the multiplicative identity is 1.

Example.

> The set of polynomials with real coefficients forms a ring under polynomial addition and multiplication. The additive identity is the zero polynomial, and the multiplicative identity is the constant polynomial 1. This is commonly denoted as $\mathbb{R}[x]$. Polynomial rings are important in algebra and number theory, as they provide a way to study polynomial equations and their roots.

Example.

The set of $n \times n$ matrices over $\mathbb{R}$ forms a ring under matrix addition and multiplication. The additive identity is the zero matrix ($\mathbf{0}$), and the multiplicative identity is the identity matrix ($\mathbf{I}$). This is commonly denoted as $M_n(\mathbb{R})$, where n is the size of the matrices and $\mathbb{R}$ is the field over which the matrices are defined.

Matrices is a very important topic in algebra, and it has some very interesting properties and relations on number theory and algebra. If you you are interested, you may see Vandermonde Matrix and Caley-Hamilton Theorem for more details.

Definition 1.3.6: Idempotent Element

An element a in a ring R is called idempotent if $a^2 = a$. In other words, applying the operation to the element with itself yields the same element.

Definition 1.3.7: Integral Domain

An integral domain is a commutative ring R with unity (multiplicative identity) such that:

1. R has no zero divisors, meaning if $ab = 0$ for $a, b \in R$, then either $a = 0$ or $b = 0$.
2. The cancellation property holds: If $a, b, c \in R$ and $ab = ac$, then $b = c$ if $a \neq 0$.

An integral domain is a special type of ring that allows for division (except by zero) and has a well-defined notion of multiplication.

Example.

The set of integers $\mathbb{Z}$ is an integral domain. The product of two non-zero integers is non-zero, and the cancellation property holds.

Example.

The set of Real numbers $\mathbb{R}$ is also an integral domain. The product of two non-zero real numbers is non-zero, and the cancellation property holds.

Example.

The set of polynomials with real coefficients $\mathbb{R}[x]$ is an integral domain. The product of two non-zero polynomials is non-zero, and the cancellation property holds.

Example.

The set of $n \times n$ matrices over $\mathbb{R}$ is not an integral domain because it contains zero divisors. For example, the product of two non-zero matrices can be the zero matrix.

Example.

The set of complex numbers $\mathbb{C}$ is an integral domain. The product of two non-zero complex numbers is non-zero, and the cancellation property holds.

Problem.

The set of real functions $\mathbb{R}^{\mathbb{R}}$ is an integral domain. The product of two non-zero functions is non-zero, and the cancellation property holds. Is this true?

Proof. No, this is not true. The set of real functions $\mathbb{R}^{\mathbb{R}}$ is is indeed a commutative ring, yet it is not an integral domain. The reason is that there exist zero divisors in this set. For example, consider the functions $f(x) = x$ and $g(x) = 1 - x$. Both functions are non-zero for all $x \in \mathbb{R}$, but their product is:

$$f(x) \cdot g(x) = x(1 - x) = 0$$

for $x = 0$ and $x = 1$. This shows that the product of two non-zero functions can be zero, violating the condition for an integral domain. Therefore, $\mathbb{R}^{\mathbb{R}}$ is not an integral domain. □

1.3.4 Polynomial Rings

Definition 1.3.8: Polynomial Ring $R[x]$

Let R be a ring. The polynomial ring $R[x]$ with coefficients in R is defined as the set of all infinite sequences $(a_0, a_1, a_2, \dots)$ satisfying the following conditions:

1. Each element a_i belongs to the ring R ($a_i \in R$ for all $i \geq 0$).
2. Only a finite number of a_i are non-zero. That is, there exists a non-negative integer N (which depends on the sequence) such that for all $i > N$, $a_i = 0$.

We define addition and multiplication operations on this set $R[x]$ as follows:
Let $p = (a_0, a_1, a_2, \dots)$ and $q = (b_0, b_1, b_2, \dots)$ be any two elements in $R[x]$ (i.e., polynomials).

Addition ($+$): The sum $p + q$ is a new sequence $(c_0, c_1, c_2, \dots)$, where the k-th element c_k is defined as the sum of a_k and b_k in the ring R:

$$c_k = a_k +_R b_k$$

Formally: $p + q = (a_0 + b_0, a_1 + b_1, a_2 + b_2, \dots)$, where all additions $a_i + b_i$ are performed in the ring R.

Multiplication ($\cdot$): The product $p \cdot q$ is a new sequence $(d_0, d_1, d_2, \dots)$, where the k-th element d_k is defined by the following sum involving multiplication and addition in the ring R:

$$d_k = \sum_{i=0}^{k} a_i \cdot_R b_{k-i} = a_0 \cdot b_k +_R a_1 \cdot b_{k-1} +_R \cdots +_R a_k \cdot b_0$$

Formally: $p \cdot q = (d_0, d_1, d_2, \dots)$, where $d_k = \sum_{i=0}^{k} a_i b_{k-i}$. This sum is known as the **Cauchy product** or **Convolution**.
With these operations, the set $R[x]$ forms a ring. The zero element is the sequence $(0, 0, 0, \dots)$, and if R has a multiplicative identity 1_R, the multiplicative identity of $R[x]$ is the sequence $(1_R, 0, 0, \dots)$.

Correspondence to Standard Notation: The formal definition using sequences corresponds directly to the standard polynomial notation $\sum_{i=0}^{n} a_i x^i$.

- The sequence $(a, 0, 0, \dots)$ corresponds to the constant polynomial $a \in R$.
- The sequence $(0, 1_R, 0, 0, \dots)$ corresponds to the indeterminate x.
- The sequence $(0, \dots, 0, 1_R$ at position $k, 0, \dots)$ corresponds to x^k.
- The polynomial $a_n x^n + \cdots + a_1 x + a_0$ corresponds to the sequence $(a_0, a_1, \dots, a_n, 0, 0, \dots)$.

The addition and multiplication rules for these sequences precisely mirror the standard rules for polynomial addition (adding like terms) and multiplication (expanding and combining

like terms, which results in the convolution formula for coefficients). Thus, the familiar notation $\sum a_i x^i$ is a convenient representation for these finite sequences under the defined operations.

Remark.

> Polynomial rings can be related to series and generating functions. You may explore these topics further if you are interested.

Proposition 1.3.9

If a ring R is an integral domain, then the polynomial ring $R[x]$ is also an integral domain.

Proof for Proposition.

To prove that $R[x]$ is an integral domain, we need to show that it satisfies the three conditions for an integral domain:

1. $R[x]$ is a commutative ring.
2. $R[x]$ has a unity element $1_{R[x]} \neq 0_{R[x]}$.
3. $R[x]$ has no zero divisors.

1. $R[x]$ is a commutative ring:

The set $R[x]$ consists of polynomials with coefficients in R. Polynomial addition and multiplication are defined as follows: For $p(x) = \sum_{i=0}^{n} a_i x^i$ and $q(x) = \sum_{j=0}^{m} b_j x^j$,

$$(p+q)(x) = \sum_{k=0}^{\max(n,m)} (a_k +_R b_k)x^k \quad (\text{where } a_k = 0 \text{ for } k > n,\ b_k = 0 \text{ for } k > m)$$

$$(p \cdot q)(x) = \sum_{k=0}^{n+m} c_k x^k, \quad \text{where } c_k = \sum_{i=0}^{k} a_i \cdot_R b_{k-i}$$

It is a standard result that $(R[x], +, \cdot)$ forms a ring. We need to verify commutativity of multiplication. Let $p(x) = \sum a_i x^i$ and $q(x) = \sum b_j x^j$. The k-th coefficient of $p(x)q(x)$ is $c_k = \sum_{i=0}^{k} a_i b_{k-i}$. The k-th coefficient of $q(x)p(x)$ is $d_k = \sum_{j=0}^{k} b_j a_{k-j}$. Let $i' = k-j$, so $j = k-i'$. As j ranges from 0 to k, i' ranges from k to 0. Thus, $d_k = \sum_{i'=0}^{k} b_{k-i'} a_{i'}$. Since R is a commutative ring (as it is an integral domain), $a_{i'} b_{k-i'} = b_{k-i'} a_{i'}$. Therefore, $d_k = \sum_{i'=0}^{k} a_{i'} b_{k-i'} = c_k$. Since the coefficients are the same for all k, $p(x)q(x) = q(x)p(x)$. Thus, multiplication in $R[x]$ is commutative.

2. $R[x]$ has a unity element $1_{R[x]} \neq 0_{R[x]}$:

Since R is an integral domain, it has a multiplicative identity $1_R \neq 0_R$. Consider the constant polynomial $1_{R[x]} = 1_R \cdot x^0 = (1_R, 0, 0, \dots)$. Let $p(x) = \sum_{i=0}^{n} a_i x^i = (a_0, a_1, \dots, a_n, 0, \dots)$. The product $p(x) \cdot 1_{R[x]}$ has its k-th coefficient as $\sum_{i=0}^{k} a_i (1_{R[x]})_{k-i}$. The only non-zero coefficient of $1_{R[x]}$ is $(1_{R[x]})_0 = 1_R$. So the sum is non-zero only when $k - i = 0$, i.e., $i = k$. The term is $a_k \cdot (1_{R[x]})_0 = a_k \cdot 1_R = a_k$. Thus, $p(x) \cdot 1_{R[x]} = p(x)$. Similarly, $1_{R[x]} \cdot p(x) = p(x)$. So, $1_{R[x]}$ is the multiplicative identity in $R[x]$. The zero

element in $R[x]$ is the zero polynomial $0_{R[x]} = (0_R, 0_R, \dots)$. Since $1_R \neq 0_R$ in the integral domain R, the first coefficient of $1_{R[x]}$ is different from the first coefficient of $0_{R[x]}$. Therefore, $1_{R[x]} \neq 0_{R[x]}$.

3. $R[x]$ has no zero divisors:

We need to show that if $p(x), q(x) \in R[x]$ are non-zero polynomials, then their product $p(x)q(x)$ is also non-zero. Let $p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0$, where $a_n \neq 0_R$ (so $\deg(p) = n$). Let $q(x) = b_m x^m + b_{m-1}x^{m-1} + \cdots + b_0$, where $b_m \neq 0_R$ (so $\deg(q) = m$). Consider the product $p(x)q(x) = c_{n+m}x^{n+m} + \cdots + c_1 x + c_0$. The coefficient of the highest degree term, x^{n+m}, is given by:

$$c_{n+m} = \sum_{i=0}^{n+m} a_i b_{n+m-i}$$

In this sum, if $i > n$, then $a_i = 0_R$. If $n + m - i > m$ (which means $n > i$), then $b_{n+m-i} = 0_R$. For a term $a_i b_{n+m-i}$ to be potentially non-zero, we must have $i \leq n$ and $n + m - i \leq m$ (which implies $n \leq i$). The only integer i satisfying both $i \leq n$ and $n \leq i$ is $i = n$. Therefore, the sum reduces to a single term:

$$c_{n+m} = a_n b_{n+m-n} = a_n b_m$$

Since R is an integral domain and $a_n \neq 0_R$ and $b_m \neq 0_R$, their product $a_n b_m \neq 0_R$. Thus, the coefficient c_{n+m} of the x^{n+m} term in $p(x)q(x)$ is non-zero. This implies that the product polynomial $p(x)q(x)$ is not the zero polynomial ($0_{R[x]}$). Therefore, $R[x]$ has no zero divisors. **Conclusion:** Since $R[x]$ is a commutative ring with a non-zero unity element and has no zero divisors, $R[x]$ is an integral domain.

1.3.5 Fields

Definition 1.3.10: Field

A **field** is a set F equipped with two binary operations, addition $(+)$ and multiplication $(\cdot)$, satisfying the following axioms:

1. **Additive Structure:**
   (a) (*Closure*) For all $a, b \in F$, $a + b \in F$.
   (b) (*Associativity*) For all $a, b, c \in F$, $(a + b) + c = a + (b + c)$.
   (c) (*Identity*) There exists $0 \in F$ such that $a + 0 = 0 + a = a$ for all $a \in F$.
   (d) (*Inverse*) For each $a \in F$, there exists $-a \in F$ such that $a + (-a) = 0$.
   (e) (*Commutativity*) For all $a, b \in F$, $a + b = b + a$.
2. **Multiplicative Structure:**
   (a) (*Closure*) For all $a, b \in F$, $a \cdot b \in F$.
   (b) (*Associativity*) For all $a, b, c \in F$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
   (c) (*Identity*) There exists $1 \in F$, $1 \neq 0$, such that $a \cdot 1 = 1 \cdot a = a$ for all $a \in F$.
   (d) (*Inverse*) For each $a \in F$, $a \neq 0$, there exists $a^{-1} \in F$ such that $a \cdot a^{-1} = 1$.
   (e) (*Commutativity*) For all $a, b \in F$, $a \cdot b = b \cdot a$.
3. **Distributive Law:** For all $a, b, c \in F$,

$$a \cdot (b + c) = a \cdot b + a \cdot c.$$

In summary, a field is a commutative ring with unity in which every nonzero element has a multiplicative inverse. This means that addition, subtraction, multiplication, and division (except by zero) are all well-defined in F.

A special field is known as a **finite field**, which is a field with a finite number of elements. Finite fields are important in number theory, coding theory, and cryptography. All arithmetic operations are defined on finite fields in undergraduate-level mathematics.

Example.

> The set of real numbers $\mathbb{R}$ is a field. The operations of addition and multiplication satisfy all the field axioms. The additive identity is 0, the multiplicative identity is 1, and every non-zero real number has a multiplicative inverse. Same goes for the set of complex numbers $\mathbb{C}$, set of rational numbers $\mathbb{Q}$, and the set of finite fields $\mathbb{F}_p$ for prime p.

Problem.

> Prove that the set of natural numbers $\mathbb{N}$ is not a field. You may use known results from number theory and algebra.

Proof. By peano axioms, 0 is not a successor of any natural number, and addition on natural numbers is defined recursively as follows:

- $a + 0 = a$
- $a + S(b) = S(a + b)$

Where $S(b)$ is the successor of b.

Now consider the additive inverse. According to the field axioms, for every $a \in \mathbb{N}$, there exists an element $-a \in \mathbb{N}$ such that $a + (-a) = 0$. However, in the set of natural numbers, there is no element $-a$ such that $a + (-a) = 0$ for any $a > 0$. Therefore, we spot a contradiction, and we conclude that $\mathbb{N}$ does not satisfy the field axioms. Thus, $\mathbb{N}$ is not a field.

Remark.

> Actually, the set of natural numbers is Commutative Monoid under addition and multiplication.

□

Problem.

> Let F be a field consisting of exactly three elements $0, 1, x$. Prove that $x + x = 1$ and that $x \cdot x = 1$. Obtain the addition and multiplication tables for F.

Proof. Since F is a field, it has the properties of both a group under addition and a group under multiplication (excluding 0 for the latter).

Proof that $x + x = 1$

1. In a group, every element has an additive inverse. In F, the additive inverse of 0 is 0 itself, since $0 + 0 = 0$.
2. The additive inverse of 1 cannot be 1 itself because $1 + 1 = 1$ would imply $1 = 0$, which is a contradiction. Therefore, 1's additive inverse must be some other element of F, which can only be x. Hence, $1 + x = 0$.
3. The element x must also have an additive inverse in F, which cannot be 0 (as 0's inverse is 0) and cannot be 1 (as 1's inverse is x). The only option left is x itself. Thus, $x + x = 0$.
4. Given $x + x = 0$ and $1 + x = 0$, by the cancellation law, it must be that $x = 1$. However, this contradicts the assumption that x is distinct from 1. Therefore, our assumption that $x + x = 0$ is incorrect.
5. The only remaining possibility is $x + x = 1$.

Proof that $x \cdot x = 1$

1. Similarly, in a multiplicative group (excluding 0), every non-zero element has a multiplicative inverse. For 1, the multiplicative inverse is 1 itself since $1 \cdot 1 = 1$.
2. The element x must have a multiplicative inverse. It cannot be 0 since 0 is not invertible, and it cannot be 1 since 1 is already serving as its own inverse.

3. The only remaining option for the multiplicative inverse of x is x itself. Hence, $x \cdot x = 1$.

Now we construct the addition and multiplication tables for F:

Addition Table:

$+$	0	1	x
0	0	1	x
1	1	x	0
x	x	0	1

Multiplication Table:

$\times$	0	1	x
0	0	0	0
1	0	1	x
x	0	x	1

□

If you know what is a vector space, you may also think of the relation between a field and a vector space. You may check this out. But we will not go into the details of vector spaces in this lecture.

Chapter 2

Primes, GCD, LCM

2.1 Prime Numbers

Definition 2.1.1: Prime Number

An integer p greater than 1 is called prime if the only positive factors of p are 1 and p. A positive integer that is greater than 1 and is not prime is called **composite**.

The reason why prime numbers are so fascinating to mathematicians is that they have so many interesting and unique property, even except for what we have seen in its definition. Below is what we call the fundamental theorem of arithmetic.

Theorem 2.1.2: Fundamental Theorem of Arithmetic

For every integer $n > 1$, there exists a unique factorization into prime numbers, up to the order of the factors. Specifically, n can be expressed as

$$n = p_1^{a_1} \cdot p_2^{a_2} \cdot \ldots \cdot p_k^{a_k}$$

where $p_1 < p_2 < \ldots < p_k$ are prime numbers and $a_1, a_2, \ldots, a_k$ are positive integers. This factorization is unique, apart from the order of the prime factors.

Proof. We prove the theorem in two parts: existence and uniqueness.

Existence: We prove by mathematical induction that every integer greater than 1 can be written as a product of primes.

Base Case: For $n = 2$, the statement holds true since 2 is itself a prime number.

Inductive Step: Assume the statement holds for all integers greater than 1 and less than n. Now consider the integer n.

- If n is prime, then it is trivially a product of primes (itself).
- If n is not prime, it can be written as $n = a \cdot b$ where $1 < a, b < n$. By the inductive hypothesis, both a and b can be factored into a product of primes. Therefore, n can also be expressed as a product of primes by combining the prime factorization of a and b.

This completes the proof of existence.

Uniqueness: Assume, for the sake of contradiction, that there are two distinct prime factorization of n:

$$n = p_1^{a_1} \cdot p_2^{a_2} \cdot \ldots \cdot p_k^{a_k} = q_1^{b_1} \cdot q_2^{b_2} \cdot \ldots \cdot q_m^{b_m}$$

where p_i and q_j are prime numbers, and a_i, b_j are positive integers. To proceed to the rest of the proof, we need to use Euclid's lemma. (Just take it as a known result for now, we will revisit this in Bezout's identity.)

Lemma 2.1.3: Euclid's Lemma

Let p be a prime number. If p divides the product ab, where a and b are integers, then p divides a or p divides b.

Proof for Lemma

Assume p is a prime that divides ab but does not divide a. We need to show that p must divide b.

Since p does not divide a, the greatest common divisor (gcd) of a and p is 1, i.e., $\gcd(a, p) = 1$. According to Bezout's identity, there exist integers x and y such that:

$$ax + py = 1$$

Multiplying both sides of the equation by b, we get:

$$abx + pby = b$$

Since p divides ab (by assumption), p divides abx. Also, p obviously divides pby. Hence, p divides the sum $abx + pby$, which means p divides b.

This completes the proof, showing that if p divides ab and does not divide a, then p must divide b, in accordance with Euclid's lemma. ■

By Euclid's lemma, if a prime divides the product of two numbers, it must divide at least one of those numbers. Thus, p_1 must divide some q_j on the right-hand side. Since q_j is prime, we conclude that $p_1 = q_j$. Applying this argument symmetrically and repeatedly, we find that each set of prime factors must be identical to the other, contradicting the assumption of two distinct factorization.

Therefore, the prime factorization of any integer greater than 1 is unique, up to the order of the factors, which completes the proof of the Fundamental Theorem of Arithmetic. □

Example.

100 can be taken as $100 = 2 \cdot 2 \cdot 5 \cdot 5 = 2^2 \cdot 5^2$. Both 2 and 5 are primes.

2.1.1 Prime Factorization Algorithm

We can use an iterative algorithm to find the prime factorization of a number. The algorithm works by dividing the number by the smallest prime factor repeatedly until the number

becomes 1. The prime factors are collected in a list. The algorithm can be implemented as follows:

Algorithm 1 Prime Factorization

```
function PrimeFactorization(n)
    factors ← an empty list
    for p ← 2 to ∞ do                          ▷ Iterate over all primes
        while n  mod p == 0 do
            Add p to factors
            n ← n/p
        end while
        if p > n/p then
            break
        end if
    end for
    if n > 1 then
        Add n to factors
    end if
    return factors
end function
```

Example.

To find the prime factorization of 8964:

- Start with $n = 8964$.
- Divide by 2: $8964 \div 2 = 4482$ (add 2 to factors).
- Divide by 2: $4482 \div 2 = 2241$ (add 2 to factors).
- Now, 2241 is not divisible by 2, so we try the next prime, which is 3.
- Divide by 3: $2241 \div 3 = 747$ (add 3 to factors).
- Divide by 3: $747 \div 3 = 249$ (add 3 to factors).
- Divide by 3: $249 \div 3 = 83$ (add 3 to factors).
- Now, 83 is a prime number.
- The final prime factorization is: $8964 = 2^2 \cdot 3^3 \cdot 83^1$.

Complexity of naive Prime Factorization

The time complexity of this naive prime factorization algorithm is primarily determined by the outer loop that iterates through potential prime factors p. The inner `while` loop divides the number n by p repeatedly.

Let the initial number be N. The outer loop starts with $p = 2$ and increments p by 1 in each iteration. The loop terminates when $p > n/p$, which is equivalent to $p^2 > n$. Since n is always decreasing (or stays the same) inside the `while` loop, the value of p can reach up to $\approx \sqrt{N}$ in the worst case (specifically, if the largest prime factor is close to $\sqrt{N}$ or N itself).

The total number of iterations of the outer loop is thus roughly proportional to $\sqrt{N}$.

Inside the `while` loop, the number n is divided by p. For a given prime factor p, the `while` loop runs $v_p(N)$ times, where $v_p(N)$ is the exponent of p in the prime factorization of N. The total number of divisions performed across all prime factors is $O(\log N)$, since each division reduces the magnitude of n significantly, and the product of prime factors is N.

However, the dominant cost comes from the outer loop's progression and the modulo operations. For each value of p in the outer loop (up to $\approx \sqrt{N}$), we perform a modulo check $n \mod p == 0$ at least once. The total number of such checks is $O(\sqrt{N})$. Although the `while` loop divisions are efficient in total, the number of attempts to divide by p in the outer loop dominates the runtime.

Therefore, the time complexity of this naive algorithm is $O(\sqrt{N})$.

The space complexity is determined by the size of the list storing the prime factors. The number of prime factors of N (counting multiplicity) is at most $O(\log N)$ (e.g., for $N = 2^k$, there are $k = \log_2 N$ factors of 2). Thus, the space complexity is $O(\log N)$.

Correctness of the Algorithm

To demonstrate the correctness of the algorithm, we need to show that it finds all prime factors of the input number n with their correct multiplicities.

1. **Finding the smallest prime factor:** The algorithm iterates through integers $p = 2, 3, 4, \ldots$ in increasing order. When it finds a number p that divides the current value of n, this p must be the smallest prime factor of the current n. Why? Because if n had a smaller prime factor $q < p$, then q would have been checked earlier in the outer loop, and n would have been divided by q until it was no longer divisible by q. Thus, by the time we check p, any smaller prime factors have already been removed from n.

2. **Handling multiplicity:** The inner `while` loop, while $n \mod p = 0$, repeatedly divides n by the smallest prime factor p as long as it is divisible. This ensures that all occurrences of this prime factor are found and added to the $factors$ list. Each division $n = n/p$ updates n to a smaller number.

3. **The break condition:** The loop terminates when $p > n/p$, which is $p^2 > n$. Consider the value of n when this condition is met. If $n > 1$, suppose n is composite. Then n must have at least one prime factor q. Since n is composite, $n = ab$ for some $1 < a \leq b < n$. The smallest prime factor q of n must satisfy $q \leq a \leq \sqrt{n}$. If the loop reaches p such that $p^2 > n$, it means we have checked all integers (and thus all prime numbers) up to $p - 1$. If the remaining n were still composite, it would have a prime factor $q \leq \sqrt{n}$. Since $p^2 > n$ implies $p > \sqrt{n}$, the factor q must be less than p. But if q was a factor of the current n, we would have divided by q when the outer loop variable was q, contradicting that q is still a factor. Therefore, if $n > 1$ when $p^2 > n$, the remaining value of n must be a prime number itself.

4. **Handling the remaining factor:** The final "If $n > 1$, add n to $factors$" block correctly adds this remaining prime factor (if any) to the list.

Combining these points, the algorithm systematically finds and removes the smallest prime factor until the number is reduced to 1 or a prime number, correctly identifying all prime factors with their respective multiplicities. The process terminates because n

decreases with each division and p increases in the outer loop, leading to the break condition being met.

This algorithm seems trivial, yet it is actually very challenging to implement in practice. Because it takes huge efforts to find the first n primes, and the algorithm is not efficient. Some more efficient algorithms for the purpose are:

- **Pollard's rho algorithm**: This is a probabilistic algorithm for integer factorization that is particularly effective for large numbers.
- **Elliptic Curve Factorization**: This is another advanced method for integer factorization that uses properties of elliptic curves.
- **General Number Field Sieve (GNFS)**: This is the most efficient classical algorithm for factoring large integers.

2.2 Prime Testing and Algorithms

The property of primes brings us to a question: how can we determine whether a number is prime or not? If we want to get many primes, how long will it take? What is the complexity of the algorithm? How can we find the next prime number after a given number?

Definition 2.2.1: Prime Testing

There are several algorithms for testing the primality of a number. Some of the most common ones include:

- **Trial Division**: This is the simplest method, where we check if a number n is divisible by any prime number less than or equal to $\sqrt{n}$. If it is, then n is composite; otherwise, it is prime.
- **Sieve of Eratosthenes**: This algorithm generates all prime numbers up to a given limit n by iteratively marking the multiples of each prime starting from 2.
- **Fermat's Little Theorem**: This theorem provides a probabilistic test for primality. It states that if p is a prime and a is an integer not divisible by p, then $a^{p-1} \equiv 1 \mod p$.

We will introduce the first two algorithms in detail, and the Fermat's Little Theorem will be introduced later for other purposes.

2.2.1 Trial Division

Trial division is a straightforward and naive algorithm for testing the primality of a number. The idea is simple, we can search for primes linearly from 2 to to any range of numbers we want to check. However, to make sure whether a number is prime or not, we can actually limit the search space to the square root of the number.

Proposition 2.2.2

This is because if n is divisible by any number greater than $\sqrt{n}$, it must also be divisible by a number less than $\sqrt{n}$.

Proof. If n is composite, by the definition of a composite integer, we know that it has a factor a with $1 < a < n$. Hence, by the definition of a factor of a positive integer, we have $n = ab$, where b is a positive integer greater than 1. We will show that $a \leq \sqrt{n}$ or $b \leq \sqrt{n}$. If $a > \sqrt{n}$ and $b > \sqrt{n}$, then $ab > \sqrt{n} \cdot \sqrt{n} = n$, which is a contradiction. Consequently, $a \leq \sqrt{n}$ or $b \leq \sqrt{n}$. Because both a and b are divisors of n, we see that n has a positive divisor not exceeding $\sqrt{n}$. This divisor is either prime or, by the fundamental theorem of arithmetic, has a prime divisor less than itself. In either case, n has a prime divisor less than or equal to $\sqrt{n}$. □

The algorithm works as follows:

Algorithm 2 TrialDivision(n)

Require: Integer $n > 1$
Ensure: **True** if n is prime, otherwise **False**

```
for d ← 2 to ⌊√n⌋ do
    if d | n then
        return False          ▷ n is divisible by d, so not prime
    end if
end for
return True                   ▷ n has no divisors other than 1 and itself
```

Example.

> For example, to check if 29 is prime, we only need to check divisibility by $2, 3, 4, 5$ (up to $\lfloor\sqrt{29}\rfloor = 5$). Since 29 is not divisible by any of these numbers, it is prime.

Complexity of Trial Division

The time complexity of the trial division algorithm is $O(\sqrt{n})$ because we only need to check divisibility up to $\sqrt{n}$. This makes it inefficient for large numbers, especially when n is very large.

Space complexity is $O(1)$ since we only need a constant amount of space to store the variables used in the algorithm.

Correctness of Trial Division

The proof is straightforward. You may also try to prove it using induction or loop invariant yourself.

2.2.2 Sieve of Eratosthenes

The Sieve of Eratosthenes is an efficient algorithm for finding all prime numbers up to a specified integer n. The algorithm works by iteratively marking the multiples of each prime number starting from 2. The remaining unmarked numbers are primes.

Algorithm 3 SieveOfEratosthenes(n)

Require: Integer $n > 1$

Ensure: List of prime numbers up to n

```
Create a list of integers from 2 to n
for p ← 2 to √n do
    if p is not marked then
        for k ← p^2 to n with step p do
            Mark k as composite
        end for
    end if
end for
Return the unmarked numbers as primes
```

Example.

To find all prime numbers up to 30:

- Start with a list of numbers from 2 to 30.
- Mark 2 and its multiples: $4, 6, 8, \ldots, 30$.
- Mark 3 and its multiples: $9, 12, 15, \ldots, 30$.
- Continue this process until reaching $\sqrt{30} \approx 5.48$.
- The remaining unmarked numbers are the primes: $2, 3, 5, 7, 11, 13, 17, 19, 23, 29$.

Complexity of Sieve of Eratosthenes

The time complexity of the Sieve of Eratosthenes is $O(n \log(\log(n)))$, which is significantly faster than trial division for large n. The space complexity is $O(n)$ because we need to store the list of integers from 2 to n.

We can show the complexity formally:

Lemma 2.2.3: Sieve of Eratosthenes Complexity

The Sieve of Eratosthenes has a time complexity of $O(n \log(\log(n)))$ and a space complexity of $O(n)$.

Proof for Lemma

To analyze the time complexity, consider the algorithm proceeds by iteratively marking the multiples of each prime p starting from 2 up to $\sqrt{n}$. For each prime p, the number

of multiples marked is approximately $\frac{n}{p} - 1$.

Therefore, the total number of marking operations is bounded by

$$T(n) \leq \sum_{\substack{p \leq \sqrt{n} \\ p \text{ prime}}} \left(\frac{n}{p} - 1 \right) \leq n \sum_{p \leq \sqrt{n}} \frac{1}{p}.$$

It is a classical result in analytic number theory (see Mertens' theorems) that the sum of reciprocals of primes up to x satisfies

$$\sum_{p \leq x} \frac{1}{p} = \log \log x + O(1).$$

By substituting $x = \sqrt{n}$, we get

$$\sum_{p \leq \sqrt{n}} \frac{1}{p} = \log \log \sqrt{n} + O(1) = \log \frac{1}{2} \log n + O(1) = \log \log n + O(1).$$

Hence, the time complexity satisfies

$$T(n) = O\big(n \log \log n\big).$$

Regarding space complexity, the algorithm requires $O(n)$ space to store a boolean array (or similar data structure) representing the primality status of each integer from 2 up to n.

This completes the proof of the time and space complexity of the Sieve of Eratosthenes. ∎

Apart from Sieve of Eratosthenes, Sieve of Ktkin is also a well-known algorithm for finding all prime numbers up to a specified integer n. It is more efficient than the Sieve of Eratosthenes for large values of n.

2.3 Greatest Common Divisor and Least Common Multiple

2.3.1 Greatest Common Divisor

Definition 2.3.1: Greatest Common Divisor

The greatest common divisor (gcd) of two integers a and b, denoted $\gcd(a, b)$, is the largest positive integer that divides both a and b without leaving a remainder.

GCD is a very important concept in number theory. It is used in many algorithms, including the Euclidean algorithm, and solving Diophantine equations, which we will discuss later.

Coprime and Property of GCD

You may have realized that prime numbers can be defined in terms of GCD. As some rational numbers a and b are coprime if and only if $\gcd(a, b) = 1$. This means that the only positive integer that divides both a and b is 1. In other words, a and b have no common prime factors.

We can extend the idea of coprimality to more than two numbers.

Definition 2.3.2: Pairwise Coprime

A set of integers $a_1, a_2, \ldots, a_n$ is said to be pairwise coprime if the gcd of every pair of distinct integers in the set is 1. In other words, for all $i \neq j$, $\gcd(a_i, a_j) = 1$.

Example.

> 9, 16, 23 are pairwise relatively prime, because $\gcd(9, 16) = 0$, $\gcd(9, 23) = 0$, $\gcd(16, 23) = 0$.

If you know basic counting principles, you may realize this is related to the Inclusion-Exclusion Principle.

Determine whether a set of numbers are pairwise coprime or not is not a trivial task. Even though when the numbers are small, we can do this by inspection. But when the numbers are large, we need to use some algorithms to do this. You may tell at a glance that 2, 3, 5, 7 are pairwise coprime, while 4, 6, 8 are not. But how about 10432, 22124, 9342238394?

One common way to check if two numbers are coprime is to iteratively check their prime factorization.

Suppose that the prime factorizations of the positive integers a and b are

$$a = p_1^{a_1} p_2^{a_2} \cdots p_n^{a_n}, b = p_1^{b_1} p_2^{b_2} \cdots p_n^{b_n},$$

where each exponent is a nonnegative integer, and where all primes occurring in the prime factorization of either a or b are included in both factorizations, with zero exponents if necessary. Then $\gcd(a, b)$ is given by

$$\gcd(a, b) = p_1^{\min(a_1, b_1)} p_2^{\min(a_2, b_2)} \cdots p_n^{\min(a_n, b_n)},$$

where $\min(x, y)$ represents the minimum of the two numbers x and y.

Example.

> Find $\gcd(100, 250)$.
>
> $$\gcd(120, 500) = 2^{\min(3,2)} 3^{\min(1,0)} 5^{\min(1,3)} = 2^2 3^0 5^1 = 20.$$

Algorithm 4 GCD Calculation via Prime Factorization

procedure CALCULATEGCD_PRIMEFACTORIZATION(a, b)
  Input: Two positive integers a and b.
  Output: The greatest common divisor $\gcd(a, b)$.
  ▷ Step 1: Obtain Prime Factorizations
  Find the prime factorization of a: $a \leftarrow p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$
  Find the prime factorization of b: $b \leftarrow q_1^{b_1} q_2^{b_2} \cdots q_m^{b_m}$
  ▷ Step 2: Identify Common and Unique Primes
  Let S be the set of all unique prime factors $\{p_1, \ldots, p_k\} \cup \{q_1, \ldots, q_m\}$.
  Let $S = \{r_1, r_2, \ldots, r_n\}$ where $r_1 < r_2 < \cdots < r_n$.
  ▷ Step 3: Standardize Factorizations with all Primes from S
  For each $i \in \{1, \ldots, n\}$:
    Let α_i be the exponent of r_i in the factorization of a (0 if r_i is not in a's factorization).
    Let β_i be the exponent of r_i in the factorization of b (0 if r_i is not in b's factorization).
  ▷ Step 4: Calculate GCD using Minimum Exponents
  Initialize gcd_value $\leftarrow 1$.
  For each $i \in \{1, \ldots, n\}$:
    min_exponent $\leftarrow \min(\alpha_i, \beta_i)$
    gcd_value $\leftarrow$ gcd_value $\times\, r_i^{\text{min\_exponent}}$
  ▷ Step 5: Return Result
  Return gcd_value
end procedure

The time complexity of the algorithm is dependent on the cost of trial division, which is the dominating factor. If we use the naive trial division algorithm, the time complexity is $O(\sqrt{n})$. However, if we use more advanced algorithms for prime factorization, such as Pollard's rho algorithm or elliptic curve factorization, the time complexity can be significantly reduced.

Below is the proof of the correctness of the algorithm.

Proof. We need to show that the number G defined by the formula,

$$G = p_1^{\min(\alpha_1,\beta_1)} p_2^{\min(\alpha_2,\beta_2)} \cdots p_n^{\min(\alpha_n,\beta_n)},$$

satisfies the two defining properties of the greatest common divisor:

1. G is a common divisor of a and b.
2. Any other common divisor of a and b divides G.

By the Fundamental Theorem of Arithmetic, any positive integer greater than 1 can be uniquely represented as a product of prime powers. a, b, and any of their divisors can be written in this form.

Proving that G is a common divisor of a and b:

The prime factorization of G is $G = p_1^{\gamma_1} p_2^{\gamma_2} \cdots p_n^{\gamma_n}$, where $\gamma_i = \min(\alpha_i, \beta_i)$ for each i. To show that G divides a, we must verify that for every prime p_i, its exponent in the

factorization of G is less than or equal to its exponent in the factorization of a. The exponent of p_i in G is $\gamma_i = \min(\alpha_i, \beta_i)$. The exponent of p_i in a is α_i. By the definition of minimum, we have $\min(\alpha_i, \beta_i) \leq \alpha_i$ for all $i = 1, \ldots, n$. Thus, $p_i^{\min(\alpha_i,\beta_i)}$ divides $p_i^{\alpha_i}$ for each i. Since G and a are products of these prime powers, and for each prime the exponent in G is less than or equal to the exponent in a, it follows that G divides a. Similarly, to show that G divides b, we compare the exponents of p_i in G and b. The exponent of p_i in G is $\gamma_i = \min(\alpha_i, \beta_i)$. The exponent of p_i in b is β_i. By the definition of minimum, we have $\min(\alpha_i, \beta_i) \leq \beta_i$ for all $i = 1, \ldots, n$. Thus, $p_i^{\min(\alpha_i,\beta_i)}$ divides $p_i^{\beta_i}$ for each i. Since G and b are products of these prime powers, and for each prime the exponent in G is less than or equal to the exponent in b, it follows that G divides b. Since G divides both a and b, G is a common divisor of a and b.

Proving that any common divisor of a and b divides G:

Let d be any positive common divisor of a and b. By the Fundamental Theorem of Arithmetic, d can also be written as a product of prime powers. Any prime factor of d must also be a prime factor of both a and b. Therefore, the prime factors of d must be a subset of $\{p_1, \ldots, p_n\}$. Let the prime factorization of d be $d = p_1^{\delta_1} p_2^{\delta_2} \cdots p_n^{\delta_n}$, where $\delta_i \geq 0$ for all i. Since d divides a, for every prime p_i, its exponent in the factorization of d must be less than or equal to its exponent in the factorization of a. That is, $\delta_i \leq \alpha_i$ for all $i = 1, \ldots, n$. Since d divides b, for every prime p_i, its exponent in the factorization of d must be less than or equal to its exponent in the factorization of b. That is, $\delta_i \leq \beta_i$ for all $i = 1, \ldots, n$. Since $\delta_i \leq \alpha_i$ and $\delta_i \leq \beta_i$, it must be that δ_i is less than or equal to the minimum of α_i and β_i. That is, $\delta_i \leq \min(\alpha_i, \beta_i) = \gamma_i$ for all $i = 1, \ldots, n$. Now, compare $d = p_1^{\delta_1} \cdots p_n^{\delta_n}$ and $G = p_1^{\gamma_1} \cdots p_n^{\gamma_n}$. We have shown that for every i, $\delta_i \leq \gamma_i$. This means that for every prime p_i, its exponent in d is less than or equal to its exponent in G. Therefore, d divides G.

Conclusion:

We have shown that G is a common divisor of a and b, and that any other common divisor d divides G. These two properties uniquely define the greatest common divisor.

Thus, $\gcd(a, b) = p_1^{\min(\alpha_1,\beta_1)} p_2^{\min(\alpha_2,\beta_2)} \cdots p_n^{\min(\alpha_n,\beta_n)}$. This completes the proof. □

Euclidean Algorithm

In real practice, euclidean algorithm is the most efficient way to compute the GCD of two integers. The Euclidean algorithm is based on the principle that the gcd of two numbers also divides their difference.

Euclidean algorithm is an recursive algorithm for finding the greatest common divisor (gcd) of two integers a and b. The algorithm is based on two key properties:

Proposition 2.3.3

$\gcd(a, b) = \gcd(b, a \mod b)$ for $b \neq 0$, where $a, b \in \mathbb{Z}$.

Proof for Proposition.

$\gcd(a, 0) = |a|$.

To prove this, we can use the definition of gcd. The gcd of two integers a and b is the largest positive integer that divides both a and b. If d divides both a and b, then it also

divides $a-kb$ for any integer k. In particular, if we take $k = \lfloor a/b \rfloor$, then d divides $a-kb = a \mod b$. Conversely, if d divides both b and $a \mod b$, then it must also divide a. Thus, the gcd of a and b is equal to the gcd of b and $a \mod b$.

Proposition 2.3.4

$\gcd(a, b) = \gcd(b, a - kb)$ for any integer k, where $a, b \in \mathbb{Z}$.

Proof for Proposition.

Assume $d = \gcd(a, b)$. Then d divides both a and b, so by definition of divisibility,

$$d \mid a \quad \text{and} \quad d \mid b \implies d \mid (a - kb) \text{ for any integer } k,$$

because $a = k_1 d$, $b = k_2 d$ for some integers k_1, k_2. $a - cb = k_1 d - ck_2 d = (k_1 - ck_2)d$ for some integer c.

Conversely, if d divides both b and $a - kb$, then it must also divide a. Therefore, the gcd of a and b is equal to the gcd of b and $a - kb$.

Definition 2.3.5: Euclidean Algorithm

Given two non-negative integers a and b (where at least one is non-zero), the algorithm proceeds recursively or iteratively:

$$\gcd(a, b) = \begin{cases} |a| & \text{if } b = 0 \\ \gcd(b, a \mod b) & \text{if } b \neq 0 \end{cases}$$

where $a \mod b$ is the remainder of the division of a by b.

The algorithm works as follows:

Algorithm 5 Euclidean Algorithm

Require: Non-negative integers a and b (at least one non-zero).
Ensure: The greatest common divisor of a and b, $\gcd(a, b)$.

```
if b = 0 then
    return a
end if
while b ≠ 0 do
    r ← a mod b        ▷ Calculate the remainder of a divided by b
    a ← b              ▷ Update a to the original b
    b ← r              ▷ Update b to the remainder r
end while
return a
```

Remark.

In python, there is a pretty handy function divmod for handling algorithms related to modulo and division.

Example.

Find $\gcd(45, 12)$ using the Euclidean algorithm.

- Step 1: $45 \mod 12 = 9$ (since $45 = 3 \times 12 + 9$)
- Step 2: $12 \mod 9 = 3$ (since $12 = 1 \times 9 + 3$)
- Step 3: $9 \mod 3 = 0$ (since $9 = 3 \times 3 + 0$)
- Step 4: Since the remainder is now zero, we stop here. The gcd is the last non-zero remainder, which is 3.

Thus, $\gcd(45, 12) = 3$.

Proposition 2.3.6

The time complexity of the Euclidean algorithm is $O(\log(\min(a, b)))$.

Proof for Proposition.

We can prove the complexity by counting the cost of each iteration and the number of iterations. Inside each iteration, we perform only basic arithmetic operations (addition, subtraction, and modulo), which take constant time O(1).

The number of iterations in finding $\gcd(a, b)$ can be expressed by

$$\lceil \log_2(\min(a, b)) \rceil.$$

This is because we always need to put the smaller number in the second position, and in each iteration, the smaller number is reduced by at least half. $b = 1$ if and only if the algorithm reaches the base case, else, we always have $b \geq 2$. Hence, in the worst case, a is continually halved, that will be $a = 2^k$ for some integer k. The number of iterations is at most k, which is $\lceil \log_2(\min(a, b)) \rceil$.

By the definition of big O notation,

$$\exists c > 0 \text{ and } n_0 \in \mathbb{N} \text{ such that } T(n) \leq c \log_2(n) \text{ for all } n \geq n_0.$$

Thus, we can conclude that the time complexity of the Euclidean algorithm is $O(\log(\min(a, b)))$.

Problem.

Let n be a positive integer greater than 1. Consider the set

$$\mathbb{Z}_n^* = \{a \in \{0, 1, \ldots, n-1\} \mid \gcd(a, n) = 1.\}$$

We define a binary operation on $\mathbb{Z}_n^*$ as multiplication modulo n. Prove that $(\mathbb{Z}_n^*, \cdot_n)$ is

a group. (You will need to show closure under multiplication modulo n, associativity, existence of an identity element, and existence of an inverse for each element. The existence of an inverse relies on properties of the greatest common divisor, specifically Bezout's identity).

2.3.2 Least Common Multiple

Definition 2.3.7: Least Common Multiple

The least common multiple of the positive integers a and b is the smallest positive integer that is divisible by both a and b. The least common multiple of a and b is denoted by $\text{lcm}(a, b)$.

Suppose the prime factorizations of a and b are given by

$$a = p_1^{a_1} p_2^{a_2} \dots p_n^{a_n}, \quad b = p_1^{b_1} p_2^{b_2} \dots p_n^{b_n},$$

where a_i and b_i are the exponents of the prime factors of a and b, respectively.

The product ab can be expressed as a prime factorization:

$$ab = p_1^{a_1+b_1} p_2^{a_2+b_2} \dots p_n^{a_n+b_n}.$$

From this, we can deduce that any common multiple of a and b must be a product of their prime factors raised to at least the maximum exponent found in either a or b. Therefore, the $\text{lcm}(a, b)$ can be expressed as

$$\text{lcm}(a, b) = p_1^{\max(a_1,b_1)} p_2^{\max(a_2,b_2)} \dots p_n^{\max(a_n,b_n)}.$$

This ensures that $\text{lcm}(a, b)$ is divisible by both a and b, and it is the smallest such number with this property.

Example.

> Find $\text{lcm}(120, 500)$.
>
> $$\text{lcm}(120, 500) = 2^{\max(2,3)} 3^{\max(1,0)} 5^{\max(1,3)} = 8 \times 3 \times 125 = 3000$$

Have you noticed, that ab is actually the product of $\text{lcm}(a, b)$ and $\gcd(a, b)$? Becuase the order of each term in the prime factorization of ab is a sum of two number, what ever which number is the maximum of the minimum, we always have the order of $a_n + b_n$. For 120 and 500 we have

$$ab = 120 \times 500 = 60000 = \text{lcm}(a, b) \times \gcd(a, b) = 3000 \times 20$$

Theorem 2.3.8: Integer Product as GCD and LCM

Let a and b be positive integers. Then $ab = \gcd(a, b) \cdot \text{lcm}(a, b)$.

Proof. By previous resultsm we know that

$$\gcd(a, b) = p_1^{\min(a_1,b_1)} p_2^{\min(a_2,b_2)} \dots p_n^{\min(a_n,b_n)},$$

$$\text{lcm}(a, b) = p_1^{\max(a_1,b_1)} p_2^{\max(a_2,b_2)} \dots p_n^{\max(a_n,b_n)}.$$

Then we have

$$\gcd(a,b)\cdot\operatorname{lcm}(a,b) = p_1^{\min(a_1,b_1)}p_2^{\min(a_2,b_2)}\dots p_n^{\min(a_n,b_n)}\cdot p_1^{\max(a_1,b_1)}p_2^{\max(a_2,b_2)}\dots p_n^{\max(a_n,b_n)}$$

We can combine the two products:

$$\gcd(a,b)\cdot\operatorname{lcm}(a,b) = p_1^{\min(a_1,b_1)+\max(a_1,b_1)}p_2^{\min(a_2,b_2)+\max(a_2,b_2)}\dots p_n^{\min(a_n,b_n)+\max(a_n,b_n)}$$

We know that $\min(a_i,b_i)+\max(a_i,b_i) = a_i+b_i$ for all i. Thus we have

$$\gcd(a,b)\cdot\operatorname{lcm}(a,b) = p_1^{a_1+b_1}p_2^{a_2+b_2}\dots p_n^{a_n+b_n} = ab.$$

This completes the proof. □

2.3.3 GCD as Linear Combination

Definition 2.3.9: Linear Combination

A linear combination of integers a and b is an expression of the form $ax+by$, where x and y are integers.
More generally for n integers $a_1, a_2, \dots, a_n$, a linear combination is an expression of the form

$$c_1a_1 + c_2a_2 + \dots + c_na_n,$$

where $c_1, c_2, \dots, c_n$ are integers.

The significance of GCD in number theory is accompanied by its representation as a linear combination of integers. This is known as Bézout's identity, or Bézout's lemma. This result is crucial in solving many problems in number theory, including Diophantine equations.

Theorem 2.3.10: Bézout's Identity

For any integers a and b, there exist integers x and y such that

$$\gcd(a,b) = ax+by.$$

Proof. For $a,b\in\mathbb{Z}$, $a\geq b>0$, compute the gcd via:

$$\begin{aligned}
a &= q_1b + r_1, \quad 0\leq r_1 < b\\
b &= q_2r_1 + r_2, \quad 0\leq r_2 < r_1\\
r_1 &= q_3r_2 + r_3, \quad 0\leq r_3 < r_2\\
&\vdots\\
r_{n-2} &= q_{n-1}r_{n-1} + r_n, \quad 0\leq r_n < r_{n-1}\\
r_{n-1} &= q_nr_n + 0
\end{aligned}$$

Thus, $\gcd(a,b) = r_n$.

Define sequences $\{x_k\}, \{y_k\}$ such that:

$$r_k = ax_k + by_k \quad \text{for } k = 0, 1, \ldots, n.$$

Base Cases:

$$\begin{cases} r_0 = a = a \cdot 1 + b \cdot 0 \implies x_0 = 1, y_0 = 0, \\ r_1 = b = a \cdot 0 + b \cdot 1 \implies x_1 = 0, y_1 = 1. \end{cases}$$

Inductive Step: Assume for some $k \geq 1$:

$$r_{k-1} = ax_{k-1} + by_{k-1}, \quad r_k = ax_k + by_k.$$

From the division step $r_{k+1} = r_{k-1} - q_{k+1}r_k$, substitute the expressions for r_{k-1} and r_k:

$$r_{k+1} = (ax_{k-1} + by_{k-1}) - q_{k+1}(ax_k + by_k).$$

Group terms:

$$r_{k+1} = a(x_{k-1} - q_{k+1}x_k) + b(y_{k-1} - q_{k+1}y_k).$$

Define:

$$x_{k+1} = x_{k-1} - q_{k+1}x_k, \quad y_{k+1} = y_{k-1} - q_{k+1}y_k.$$

Thus, r_{k+1} is also a linear combination of a and b. After n steps, the gcd r_n satisfies:

$$r_n = ax_n + by_n,$$

where x_n, y_n are integers derived recursively. Hence, $\gcd(a, b)$ is expressible as $ax + by$ for some $x, y \in \mathbb{Z}$. □

Bézout's identity allows us to implement extended Euclidean algorithm, which not only computes the GCD of two integers but also finds the coefficients x and y such that $ax + by = \gcd(a, b)$.

Definition 2.3.11: Extended Euclidean Algorithm

The extended Euclidean algorithm is an extension of the Euclidean algorithm that computes the GCD of two integers a and b and also finds integers x and y such that

$$\gcd(a, b) = ax + by.$$

Example.

Let $a = 1071$, $b = 462$.

$$1071 = 2 \cdot 462 + 147$$
$$462 = 3 \cdot 147 + 21$$
$$147 = 7 \cdot 21 + 0$$

Back-substitute:

$$21 = 462 - 3 \cdot 147 \quad \text{(from step 2)}$$
$$147 = 1071 - 2 \cdot 462 \quad \text{(from step 1)}$$
$$\implies 21 = 462 - 3 \cdot (1071 - 2 \cdot 462)$$
$$= 7 \cdot 462 - 3 \cdot 1071.$$

Thus, $\gcd(1071, 462) = -3 \cdot 1071 + 7 \cdot 462$.

Chapter 3

Modular Equations and Diophantine Equations

3.1 Solving Linear Congruence

3.1.1 Modular Inverse

3.2 Diophantine Equations

3.2.1 Linear Diophantine Equations

3.3 System of Linear Congruence

3.3.1 Chinese Remainder Theorem

3.4 Fermat's Little Theorem

Chapter 4

Applications of Number Theory

4.1 Cryptography

4.1.1 Basics of Cryptosystems

4.1.2 Diffie-Hellman Key Exchange

RSA Algorithm

4.2 Hash Functions

4.2.1 Modular Hash Functions

4.2.2 Multiplicative Hash Functions

4.2.3 Polynomial Hash Functions

4.2.4 SHA-256

4.3 Representation of Integers

4.3.1 Modular Exponentiation Algorithm

4.3.2 Base Representation and Conversion

Bibliography

[1] K. Rosen, *Discrete Mathematics and Its Applications*, 8th ed. New York (N.Y.): McGraw-Hill Education, Aug. 10, 2018, 2240 pp., ISBN: 978-1-260-09199-1.

[2] K. Rosen, *Elementary Number Theory and Its Application, 6th Edition*, 6th edition. Boston: Pearson, Mar. 30, 2010, 752 pp., ISBN: 978-0-321-50031-1.

[3] E. Lehman, F. T. Leighton, and A. R. Meyer, *Mathematics for Computer Science (Lehman, Leighton, and Meyer)*. Mar. 8, 2017, ISBN: 978-988-8407-06-4. [Online]. Available: `https://eng.libretexts.org/Bookshelves/Computer_Science/Programming_and_Computation_Fundamentals/Mathematics_for_Computer_Science_(Lehman_Leighton_and_Meyer)`.