CS70: 00 - 09

Satish Rao
LaTeX by Jisung
University of California, Berkeley

Spring 2025

Contents

Chapter 1

00 - Review of Sets and Mathematical Notation

1.1 Review of Sets and Mathematical Notation

A **set** is a well-defined collection of objects. These objects are called *elements* or *members* of the set, and they can be anything, including numbers, letters, people, cities, or even other sets. Sets are usually denoted by capital letters and can be described by listing their elements inside curly braces.

Example.

> Let A be the set of the first five prime numbers:
>
> $$A = \{2, 3, 5, 7, 11\}$$
>
> If x is an element of A, we write $x \in A$. Otherwise, we write $x \notin A$.

Two sets A and B are **equal**, written $A = B$, if they contain the same elements. The order and repetition of elements do not matter:

$$\{red, white, blue\} = \{blue, white, red\} = \{red, white, white, blue\}$$

A more general way to define a set is using set-builder notation:

$$Q = \left\{ \frac{a}{b} \mid a, b \in \mathbb{Z}, b \neq 0 \right\}$$

This describes the set of all rational numbers.

1.1.1 Cardinality

The **cardinality** of a set A, denoted $|A|$, is the number of elements in A.

Example.

> If $A = \{1, 2, 3, 4\}$, then $|A| = 4$.

A set may have infinitely many elements, such as $\mathbb{N}$, the set of natural numbers.

1.1.2 Subsets and Proper Subsets

A set A is a **subset** of B, written $A \subseteq B$, if every element of A is also in B. A **proper subset**, denoted $A \subset B$, means that A is strictly contained in B.

Example.

> Let $B = \{1, 2, 3, 4, 5\}$. Then $\{1, 2, 3\}$ is a proper subset of B, but $\{1, 2, 3, 4, 5\}$ is only a subset, not a proper subset.

Proposition 1.1.1

- The empty set $\emptyset$ is a subset of every set.
- Every set A is a subset of itself: $A \subseteq A$.

1.1.3 Intersections and Unions

The **intersection** of two sets A and B, written $A \cap B$, consists of elements that are in both A and B. If $A \cap B = \emptyset$, the sets are **disjoint**.

The **union** of A and B, written $A \cup B$, consists of elements that are in either A or B (or both).

Example.

> Let A be the set of positive even numbers and B the set of positive odd numbers. Then:
>
> $$A \cap B = \emptyset, \quad A \cup B = \mathbb{Z}^+$$

Proposition 1.1.2

- $A \cup B = B \cup A$
- $A \cap B = B \cap A$
- $A \cup \emptyset = A$
- $A \cap \emptyset = \emptyset$

1.1.4 Complements

If $A \subseteq B$, the **relative complement** of A in B, written $B \setminus A$, is the set of elements in B but not in A.

Example.

> If $B = \{1, 2, 3\}$ and $A = \{3, 4, 5\}$, then $B \setminus A = \{1, 2\}$.

Proposition 1.1.3

- $A \setminus A = \emptyset$
- $A \setminus \emptyset = A$
- $\emptyset \setminus A = \emptyset$

1.1.5 Significant Sets

Some common sets in mathematics include:

- $\mathbb{N}$: natural numbers $\{0, 1, 2, 3, \dots\}$
- $\mathbb{Z}$: integers $\{\dots, -2, -1, 0, 1, 2, \dots\}$
- $\mathbb{Q}$: rational numbers
- $\mathbb{R}$: real numbers
- $\mathbb{C}$: complex numbers

1.2 Mathematical Notation

1.2.1 Sums and Products

Summations and products provide compact notation for large operations:

$$\sum_{i=1}^{n} i = 1 + 2 + \cdots + n$$

$$\prod_{i=1}^{n} i = 1 \cdot 2 \cdot \cdots \cdot n$$

1.2.2 Universal and Existential Quantifiers

A **universal quantifier** ($\forall$) expresses that a statement holds for all elements of a set:

$$\forall n \in \mathbb{N}, n^2 + n + 41 \text{ is prime}$$

This is false, as counterexamples exist.

An **existential quantifier** ($\exists$) states that there exists at least one element satisfying a condition:

$$\exists x \in \mathbb{Z}, x < 2 \text{ and } x^2 = 4$$

This is true for $x = -2$.

Remark.

Universal and existential quantifiers can be combined to form more complex statements, such as:

- $\forall x \in \mathbb{Z}, \exists y \in \mathbb{Z}, y > x$ (true: given an integer, we can always find a larger one)
- $\exists y \in \mathbb{Z}, \forall x \in \mathbb{Z}, y > x$ (false: there is no largest integer)

Chapter 2

01 - Propositional Logic

2.1 Propositional Logic

Propositional logic is the foundation of mathematical reasoning. It deals with propositions, which are statements that are either true or false. Understanding propositional logic is crucial for constructing and analyzing mathematical arguments.

2.1.1 Propositions

Definition 2.1.1: Proposition

A proposition is a declarative statement that is either true or false. Examples include:

- $\sqrt{3}$ is irrational.
- $1 + 1 = 5$.
- Julius Caesar had 2 eggs for breakfast on his 10th birthday.

Statements that are not propositions include:

- $2 + 2$ (not a complete statement).
- $x^2 + 3x = 5$ (depends on the value of x).
- Arnold Schwarzenegger often eats broccoli (fuzzy term ”often”).

2.1.2 Logical Connectives

Propositions can be combined using logical connectives to form more complex statements.

Definition 2.1.2: Conjunction

The conjunction of two propositions P and Q, denoted $P \wedge Q$, is true only when both P and Q are true.

Definition 2.1.3: Disjunction

The disjunction of two propositions P and Q, denoted $P \vee Q$, is true when at least one of P or Q is true.

Definition 2.1.4: Negation

The negation of a proposition P, denoted $\neg P$, is true when P is false.

2.1.3 Truth Tables

Truth tables are used to describe the possible values of propositional forms based on the truth values of their variables.

Example.

> The truth table for conjunction:

P	Q	$P \wedge Q$
T	T	T
T	F	F
F	T	F
F	F	F

2.1.4 Implication

Definition 2.1.5: Implication

An implication $P \implies Q$ ("P implies Q") is false only when P is true and Q is false. It is logically equivalent to $\neg P \vee Q$.

Remark.

> Implications can be vacuously true if the hypothesis P is false. For example, "If pigs can fly, then horses can read" is vacuously true.

2.1.5 Logical Equivalence

Definition 2.1.6: Logical Equivalence

Two propositional forms are logically equivalent if they have the same truth table. For example, $P \implies Q$ is logically equivalent to $\neg P \vee Q$.

2.2 Quantifiers

Quantifiers allow us to make statements about collections of propositions. The two main quantifiers are the universal quantifier ($\forall$) and the existential quantifier ($\exists$).

2.2.1 Universal Quantifier

Definition 2.2.1: Universal Quantifier

The universal quantifier $\forall$ (”for all”) asserts that a proposition is true for every element in a specified universe. For example, $(\forall n \in \mathbb{N})(n^2 + n + 41 \text{ is prime})$.

2.2.2 Existential Quantifier

Definition 2.2.2: Existential Quantifier

The existential quantifier $\exists$ (”there exists”) asserts that there is at least one element in the universe for which the proposition is true. For example, $(\exists x \in \mathbb{Z})(x \text{ is even and odd})$.

2.2.3 Negation of Quantifiers

Definition 2.2.3: Negation of Quantifiers

The negation of a universally quantified statement is an existentially quantified statement, and vice versa:

$$\neg(\forall x P(x)) \equiv \exists x \neg P(x)$$

$$\neg(\exists x P(x)) \equiv \forall x \neg P(x)$$

2.3 De Morgan’s Laws

De Morgan’s Laws describe how to negate conjunctions and disjunctions.

Theorem 2.3.1: De Morgan’s Laws

For any propositions P and Q:

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$$

$$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$$

Proof. The proof follows from constructing the truth tables for both sides of the equivalences and observing that they match. $\square$

2.4 Tricky Quantifier Examples

Quantifiers can be combined in complex ways to express more nuanced statements.

Example.

> To express ”there are at least three distinct integers x that satisfy $P(x)$”:
>
> $$\exists x \exists y \exists z (x \neq y \land y \neq z \land z \neq x \land P(x) \land P(y) \land P(z))$$

Example.

> To express ”there are at most three distinct integers x that satisfy $P(x)$”:
>
> $$\exists x \exists y \exists z \forall d (P(d) \implies d = x \lor d = y \lor d = z)$$

Example.

> To express ”there are exactly three distinct integers x that satisfy $P(x)$”:
>
> $$\exists x \exists y \exists z (x \neq y \land y \neq z \land z \neq x \land P(x) \land P(y) \land P(z)) \land \forall d (P(d) \implies d = x \lor d = y \lor d = z)$$

Chapter 3

02 - Proofs

3.1 Proofs

Proofs are the cornerstone of mathematical reasoning, providing absolute certainty about the truth of a statement. Unlike in science, where evidence is gathered through experiments, mathematical proofs rely on logical deductions from axioms and previously established statements.

3.1.1 What is a Proof?

Definition 3.1.1: Proof

A proof is a finite sequence of logical deductions that establishes the truth of a statement. It starts from axioms (statements accepted without proof) and applies the rules of logic to derive the desired conclusion.

3.1.2 Types of Statements in Computer Science

In computer science, we often need to prove statements about programs, such as:

- Does program P halt on every input?
- Does program P correctly compute the function $f(x)$ for every x?

These statements often involve infinitely many cases, making testing insufficient. A rigorous proof is necessary to guarantee their truth.

3.2 Notation and Basic Facts

Before diving into proof techniques, we establish some notation and basic mathematical facts.

3.2.1 Notation

- $\mathbb{Z}$: The set of integers, $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$.

- $\mathbb{N}$: The set of natural numbers, $\mathbb{N} = \{0, 1, 2, \ldots\}$.
- $a \mid b$: a divides b, meaning there exists an integer q such that $b = aq$.
- $p \geq 2$ is prime if it is divisible only by 1 and itself.

3.2.2 Basic Facts

- The sum or product of two integers is an integer.
- The set of natural numbers is closed under addition and multiplication.

3.3 Direct Proof

Definition 3.3.1: Direct Proof

A direct proof of $P \implies Q$ starts by assuming P and then derives Q through a chain of logical implications.

Theorem 3.3.2: Theorem 2.1

For any $a, b, c \in \mathbb{Z}$, if $a \mid b$ and $a \mid c$, then $a \mid (b + c)$.

Proof. Assume $a \mid b$ and $a \mid c$. Then, there exist integers q_1 and q_2 such that $b = q_1 a$ and $c = q_2 a$. Thus, $b + c = (q_1 + q_2)a$. Since $\mathbb{Z}$ is closed under addition, $q_1 + q_2 \in \mathbb{Z}$, so $a \mid (b + c)$. □

Theorem 3.3.3: Theorem 2.2

Let $0 < n < 1000$ be an integer. If the sum of the digits of n is divisible by 9, then n is divisible by 9.

Proof. Let $n = 100a + 10b + c$. Assume $a + b + c = 9k$ for some integer k. Then, $n = 9k + 99a + 9b = 9(k + 11a + b)$, so n is divisible by 9. □

Theorem 3.3.4: Theorem 2.3

Let $0 < n < 1000$ be an integer. If n is divisible by 9, then the sum of the digits of n is divisible by 9.

Proof. Assume n is divisible by 9. Then, $n = 9l$ for some integer l. Writing $n = 100a + 10b + c$, we have $99a + 9b + (a + b + c) = 9l$, so $a + b + c = 9(l - 11a - b)$, which is divisible by 9. □

3.4 Proof by Contraposition

Definition 3.4.1: Proof by Contraposition

To prove $P \implies Q$, we instead prove $\neg Q \implies \neg P$, which is logically equivalent.

Theorem 3.4.2: Theorem 2.4

Let n be a positive integer and let d divide n. If n is odd, then d is odd.

Proof. We proceed by contraposition. Assume d is even. Then, $d = 2k$ for some integer k. Since $d \mid n$, $n = dl = 2(kl)$, so n is even. $\square$

Theorem 3.4.3: Theorem 2.5 (Pigeonhole Principle)

Let n and k be positive integers. If $n > k$, then placing n objects into k boxes guarantees that at least one box contains multiple objects.

Proof. We proceed by contraposition. If each box contains at most one object, then the number of objects n is at most the number of boxes k, i.e., $n \leq k$. $\square$

3.5 Proof by Contradiction

Definition 3.5.1: Proof by Contradiction

To prove P, we assume $\neg P$ and derive a contradiction, showing that $\neg P$ must be false, and thus P is true.

Theorem 3.5.2: Theorem 2.6

There are infinitely many prime numbers.

Proof. Assume there are only finitely many primes $p_1, p_2, \ldots, p_k$. Let $q = p_1 p_2 \cdots p_k + 1$. By Lemma 2.1, q has a prime divisor p, which must be one of the p_i. But then $p \mid (q - p_1 p_2 \cdots p_k) = 1$, which is a contradiction. $\square$

Theorem 3.5.3: Theorem 2.7

$\sqrt{2}$ is irrational.

Proof. Assume $\sqrt{2}$ is rational, so $\sqrt{2} = \frac{a}{b}$ for integers a and b with no common factors. Then, $a^2 = 2b^2$, so a is even. Let $a = 2c$. Then, $b^2 = 2c^2$, so b is even. This contradicts the assumption that a and b have no common factors. $\square$

3.6 Proof by Cases

Definition 3.6.1: Proof by Cases

When a statement can be divided into multiple cases, we prove the statement for each case separately.

Theorem 3.6.2: Theorem 2.8

There exist irrational numbers x and y such that x^y is rational.

Proof. Let $x = \sqrt{2}$ and $y = \sqrt{2}$. Consider two cases:

- If $\sqrt{2}^{\sqrt{2}}$ is rational, we are done.
- If $\sqrt{2}^{\sqrt{2}}$ is irrational, let $x = \sqrt{2}^{\sqrt{2}}$ and $y = \sqrt{2}$. Then, $x^y = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = 2$, which is rational.

□

3.7 Common Errors in Proofs

3.7.1 Assuming What You Need to Prove

Claim: Claim 2.1

$-2 = 2$.

Proof for Claim.

The proof assumes $-2 = 2$ and derives a true statement, but this does not prove the claim.

3.7.2 Dividing by Zero

Claim: Claim 2.2

$1 = 2$.

Proof for Claim.

The proof divides by $x - y = 0$, which is invalid.

3.7.3 Mixing Negative Numbers and Inequalities

Claim: Claim 2.3

$4 \leq 1$.

Proof for Claim.

The proof incorrectly squares both sides of an inequality involving negative numbers.

3.8 Style and Substance in Proofs

When writing proofs, it is important to:

- Justify each step clearly.
- Break down complex proofs into lemmas.
- Make lemmas as general as possible for reuse.

Chapter 4

03 - Induction

4.1 Mathematical Induction

Mathematical induction is a powerful proof technique used to establish that a statement holds for all natural numbers. It allows us to reason about infinitely many cases using finite means.

4.1.1 Introduction to Induction

Definition 4.1.1: Mathematical Induction

Mathematical induction is a method of proving statements of the form $\forall n \in \mathbb{N}, P(n)$, where $P(n)$ is a proposition about the natural number n. The proof consists of two steps:

- **Base Case**: Prove that $P(0)$ is true.
- **Inductive Step**: Assume $P(k)$ is true for some arbitrary $k \geq 0$ (the induction hypothesis), and then prove that $P(k+1)$ is true.

4.1.2 Example: Sum of the First n Natural Numbers

Theorem 4.1.2: Theorem 3.1

For all $n \in \mathbb{N}$, $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$.

Proof. We proceed by induction on n.

Base Case ($n = 0$): $\sum_{i=0}^{0} i = 0 = \frac{0(0+1)}{2}$, so the base case holds.

Induction Hypothesis: Assume that $\sum_{i=0}^{k} i = \frac{k(k+1)}{2}$ for some arbitrary $k \geq 0$.

Inductive Step: We show that $\sum_{i=0}^{k+1} i = \frac{(k+1)(k+2)}{2}$:

$$\sum_{i=0}^{k+1} i = \left(\sum_{i=0}^{k} i\right) + (k+1) = \frac{k(k+1)}{2} + (k+1) = \frac{(k+1)(k+2)}{2}.$$

By the principle of mathematical induction, the theorem holds. ☐

4.1.3 Another Example: Divisibility by 3

Theorem 4.1.3: Theorem 3.2

For all $n \in \mathbb{N}$, $n^3 - n$ is divisible by 3.

Proof. We proceed by induction on n.

Base Case ($n = 0$): $0^3 - 0 = 0$, which is divisible by 3.

Induction Hypothesis: Assume that $k^3 - k$ is divisible by 3 for some arbitrary $k \geq 0$.

Inductive Step: We show that $(k+1)^3 - (k+1)$ is divisible by 3:

$$(k+1)^3 - (k+1) = k^3 + 3k^2 + 3k + 1 - (k+1) = (k^3 - k) + 3(k^2 + k).$$

By the induction hypothesis, $k^3 - k$ is divisible by 3, and $3(k^2 + k)$ is clearly divisible by 3. Thus, $(k+1)^3 - (k+1)$ is divisible by 3. ☐

4.2 Strengthening the Induction Hypothesis

Sometimes, the induction hypothesis needs to be strengthened to make the proof work. This is particularly useful when the original hypothesis is too weak to prove the inductive step.

4.2.1 Example: Sum of the First n Odd Numbers

Theorem 4.2.1: Theorem 3.4

For all $n \geq 1$, the sum of the first n odd numbers is n^2.

Proof. We proceed by induction on n.

Base Case ($n = 1$): The first odd number is 1, which equals 1^2.

Induction Hypothesis: Assume that the sum of the first k odd numbers is k^2.

Inductive Step: The $(k+1)$st odd number is $2k + 1$. Thus, the sum of the first $k+1$ odd numbers is:

$$k^2 + (2k+1) = (k+1)^2.$$

By the principle of induction, the theorem holds. ☐

4.3 Strong Induction

Strong induction is a variant of mathematical induction where the induction hypothesis assumes that the statement holds for all values up to k, rather than just for k.

Definition 4.3.1: Strong Induction

To prove $\forall n \in \mathbb{N}, P(n)$ using strong induction:

- **Base Case**: Prove that $P(0)$ is true.
- **Induction Hypothesis**: Assume that $P(m)$ is true for all $0 \leq m \leq k$.
- **Inductive Step**: Prove that $P(k+1)$ is true.

4.3.1 Example: Postage Stamp Problem

Theorem 4.3.2: Theorem 3.6

For every natural number $n \geq 12$, $n = 4x + 5y$ for some $x, y \in \mathbb{N}$.

Proof. We proceed by strong induction on n.

Base Cases:

- $n = 12$: $12 = 4 \cdot 3 + 5 \cdot 0$.
- $n = 13$: $13 = 4 \cdot 2 + 5 \cdot 1$.
- $n = 14$: $14 = 4 \cdot 1 + 5 \cdot 2$.
- $n = 15$: $15 = 4 \cdot 0 + 5 \cdot 3$.

Induction Hypothesis: Assume that for all $12 \leq m \leq k$, $m = 4x + 5y$ for some $x, y \in \mathbb{N}$.

Inductive Step: For $n = k + 1 \geq 16$, note that $(k+1) - 4 \geq 12$. By the induction hypothesis, $(k+1) - 4 = 4x' + 5y'$ for some $x', y' \in \mathbb{N}$. Thus, $k + 1 = 4(x' + 1) + 5y'$. $\square$

4.4 Recursion and Induction

There is a deep connection between recursion and induction. Recursive definitions often lend themselves naturally to inductive proofs.

4.4.1 Example: Fibonacci Numbers

The Fibonacci sequence is defined recursively as follows:

$$F(0) = 0, \quad F(1) = 1, \quad F(n) = F(n-1) + F(n-2) \text{ for } n \geq 2.$$

Theorem 4.4.1: Theorem 3.7

Every natural number $n > 1$ can be written as a product of one or more primes.

Proof. We proceed by strong induction on n.

Base Case ($n = 2$): 2 is a prime number, so it is trivially a product of primes.

Induction Hypothesis: Assume that for all $2 \leq m \leq k$, m can be written as a product of primes.

Inductive Step: For $n = k + 1$, if $k + 1$ is prime, we are done. Otherwise, $k + 1 = xy$ for some $1 < x, y < k + 1$. By the induction hypothesis, x and y can be written as products of primes, so $k + 1$ can also be written as a product of primes. □

4.5 False Proofs

It is easy to make mistakes in proofs, especially when using induction. A classic example is the ”proof” that all horses are the same color.

Theorem 4.5.1: Theorem 3.8

All horses are the same color.

Proof. We proceed by induction on the number of horses, n.

Base Case $(n = 1)$: A single horse is trivially the same color as itself.

Induction Hypothesis: Assume that any set of k horses are the same color.

Inductive Step: Consider a set of $k + 1$ horses. By the induction hypothesis, the first k horses are the same color, and the last k horses are also the same color. Since the middle $k - 1$ horses are in both sets, all $k + 1$ horses must be the same color. □

Remark.

> The error in this proof lies in the inductive step when $k = 1$. The argument that the middle horses are the same color fails because there are no middle horses when $k = 1$.

Chapter 5

04 - Stable Matching

5.1 The Stable Matching Problem

The Stable Matching Problem is a fundamental problem in algorithms, where the goal is to match two sets of entities (e.g., jobs and candidates) based on their preferences in such a way that no pair has an incentive to deviate from the matching.

5.1.1 Introduction

Definition 5.1.1: Stable Matching Problem

Given two sets of entities (e.g., jobs and candidates), each with a preference list over the other set, the goal is to find a matching such that there are no ”rogue couples” — pairs that prefer each other over their current matches.

5.1.2 Example

Consider the following example with three jobs and three candidates:

Jobs	Candidates		
Approximation Inc.	Anita	Bridget	Christine
Basis Co.	Bridget	Anita	Christine
Control Corp.	Anita	Bridget	Christine

Candidates	Jobs		
Anita	Basis Co.	Approximation Inc.	Control Corp.
Bridget	Approximation Inc.	Basis Co.	Control Corp.
Christine	Approximation Inc.	Basis Co.	Control Corp.

The goal is to find a stable matching between jobs and candidates.

5.2 The Propose-and-Reject Algorithm

The Propose-and-Reject Algorithm (also known as the Gale-Shapley Algorithm) is a simple and efficient algorithm for solving the Stable Matching Problem.

5.2.1 Algorithm Description

The algorithm proceeds in days, with the following steps:

- **Every Morning**: Each job proposes to the most preferred candidate on its list who has not yet rejected it.
- **Every Afternoon**: Each candidate collects all offers and accepts the best one, rejecting the rest.
- **Every Evening**: Each rejected job removes the candidate from its list.

The algorithm terminates when no more rejections occur, and each candidate has a job offer.

5.2.2 Example Execution

For the example above, the algorithm proceeds as follows:

Days	Candidates	Offers
1	Anita, Bridget, Christine	Approximation Inc., Control Corp., Basis Co.
2	Anita, Bridget, Christine	Approximation Inc., Basis Co., Control Corp.
3	Anita, Bridget, Christine	Approximation Inc., Basis Co., Control Corp.

The final matching is:

$$\{(\text{Approximation Inc., Anita}), (\text{Basis Co., Bridget}), (\text{Control Corp., Christine})\}.$$

5.3 Properties of the Propose-and-Reject Algorithm

The algorithm has two key properties: it always terminates, and it produces a stable matching.

5.3.1 Termination

Lemma 5.3.1: Lemma 4.1

The Propose-and-Reject Algorithm always halts.

Proof. Each job can propose to at most n candidates, and there are n jobs. Thus, the algorithm terminates in at most n^2 steps. $\square$

5.3.2 Stability

Theorem 5.3.2: Theorem 4.1

The matching produced by the Propose-and-Reject Algorithm is always stable.

Proof. Suppose for contradiction that there is a rogue couple (J, C) in the final matching. Since J proposed to C before its current match, C must have rejected J in favor of a better offer. By the Improvement Lemma, C prefers her current job to J, so (J, C) cannot be a rogue couple. $\square$

5.4 Optimality

The Propose-and-Reject Algorithm not only produces a stable matching but also ensures that the matching is optimal for one side of the matching.

5.4.1 Job Optimality

Definition 5.4.1: Optimal Candidate for a Job

The optimal candidate for a job J is the highest-ranked candidate on J's preference list that J can be paired with in any stable matching.

Theorem 5.4.2: Theorem 4.2

The matching produced by the Propose-and-Reject Algorithm is job optimal.

Proof. Suppose for contradiction that some job J is rejected by its optimal candidate C^*. Then, there exists a stable matching where J is paired with C^*, but this leads to a contradiction since C^* would have rejected J in favor of a better offer. □

5.4.2 Candidate Pessimality

Theorem 5.4.3: Theorem 4.3

If a matching is job optimal, then it is candidate pessimal.

Proof. Suppose for contradiction that a candidate C is not paired with her pessimal job in the job optimal matching. Then, there exists a stable matching where C is paired with a worse job, leading to a contradiction. □

Chapter 6

05 - Graphs

6.1 Graph Theory: An Introduction

Graph theory is a fundamental area of computer science that deals with the study of graphs, which are mathematical structures used to model pairwise relations between objects. Graphs are used to represent networks, such as social networks, computer networks, and transportation systems.

6.1.1 Formal Definitions

Definition 6.1.1: Graph

A graph G is defined by a set of vertices V and a set of edges E. In an undirected graph, edges are unordered pairs of vertices, while in a directed graph, edges are ordered pairs.

6.1.2 Example: Königsberg Bridges

The Seven Bridges of Königsberg problem is a classic example in graph theory. The problem asks whether it is possible to walk through the city crossing each bridge exactly once and return to the starting point. Euler proved that this is impossible by modeling the city as a graph and showing that such a walk (now called an Eulerian tour) cannot exist.

6.2 The Propose-and-Reject Algorithm

The Propose-and-Reject Algorithm, also known as the Gale-Shapley Algorithm, is used to solve the Stable Matching Problem. The algorithm proceeds in rounds where jobs propose to candidates, and candidates reject or accept proposals based on their preferences.

6.2.1 Algorithm Description

- **Every Morning**: Each job proposes to the most preferred candidate on its list who has not yet rejected it.

- **Every Afternoon**: Each candidate collects all offers and accepts the best one, rejecting the rest.
- **Every Evening**: Each rejected job removes the candidate from its list.

The algorithm terminates when no more rejections occur, and each candidate has a job offer.

6.2.2 Properties of the Algorithm

Lemma 6.2.1: Lemma 4.1

The Propose-and-Reject Algorithm always halts.

Proof. Each job can propose to at most n candidates, and there are n jobs. Thus, the algorithm terminates in at most n^2 steps. $\square$

Theorem 6.2.2: Theorem 4.1

The matching produced by the Propose-and-Reject Algorithm is always stable.

Proof. Suppose for contradiction that there is a rogue couple (J, C) in the final matching. Since J proposed to C before its current match, C must have rejected J in favor of a better offer. By the Improvement Lemma, C prefers her current job to J, so (J, C) cannot be a rogue couple. $\square$

6.3 Eulerian Tours

An Eulerian tour is a walk in a graph that uses each edge exactly once and returns to the starting point. Euler's Theorem provides a necessary and sufficient condition for a graph to have an Eulerian tour.

Theorem 6.3.1: Theorem 5.1

An undirected graph $G = (V, E)$ has an Eulerian tour if and only if G is connected and every vertex has even degree.

Proof. **Only if**: If G has an Eulerian tour, then every vertex must have even degree because the tour enters and exits each vertex the same number of times.

If: We provide a recursive algorithm to construct an Eulerian tour. The algorithm uses the fact that every vertex has even degree to ensure that a tour can be found. $\square$

6.4 Planar Graphs

A graph is planar if it can be drawn on a plane without any edges crossing. Planar graphs have many interesting properties, including Euler's formula, which relates the number of vertices, edges, and faces of a planar graph.

Theorem 6.4.1: Theorem 5.2

For every connected planar graph, $\nu + f = e + 2$, where ν is the number of vertices, e is the number of edges, and f is the number of faces.

Proof. The proof is by induction on the number of edges. The base case is a tree, which has $f = 1$ and $e = \nu - 1$. For graphs with cycles, removing an edge reduces both e and f by one, preserving the formula. □

6.5 Graph Coloring

Graph coloring is the assignment of colors to the vertices of a graph such that no two adjacent vertices share the same color. The Four Color Theorem states that any planar graph can be colored with at most four colors.

Theorem 6.5.1: Theorem 5.4

Every planar graph can be colored with five colors.

Proof. The proof is by induction on the number of vertices. The key idea is that every planar graph has a vertex of degree at most five, which can be colored using one of the five colors. □

6.6 Important Classes of Graphs

6.6.1 Complete Graphs

A complete graph K_n is a graph where every pair of distinct vertices is connected by a unique edge. Complete graphs are maximally connected but require a large number of edges.

6.6.2 Trees

A tree is a connected graph with no cycles. Trees have many equivalent definitions, including:

- A connected graph with $n - 1$ edges.
- A graph where the removal of any edge disconnects the graph.

Theorem 6.6.1: Theorem 5.5

The statements "G is connected and contains no cycles" and "G is connected and has $n - 1$ edges" are equivalent.

Proof. The proof is by induction on the number of vertices. The base case is a single vertex with no edges. For the inductive step, removing a vertex and applying the induction hypothesis shows that the graph must have $n - 1$ edges. □

6.6.3 Hypercubes

A hypercube is a graph where each vertex represents an n-bit string, and edges connect vertices that differ in exactly one bit. Hypercubes are highly connected and have many applications in parallel computing.

Theorem 6.6.2: Theorem 5.6

Let $S \subseteq V$ be such that $|S| \leq |V - S|$. Then, the number of edges connecting S to $V - S$ is at least $|S|$.

Proof. The proof is by induction on the dimension of the hypercube. The base case is a 1-dimensional hypercube with two vertices and one edge. For the inductive step, the hypercube is divided into two subcubes, and the induction hypothesis is applied to each subcube. □

Chapter 7

06 - Modular Arithmetic

7.1 Modular Arithmetic

Modular arithmetic is a fundamental concept in number theory and cryptography. It involves performing arithmetic operations within a fixed range of numbers, typically from 0 to $m-1$, where m is a positive integer known as the modulus. This system is analogous to the way a clock wraps around after 12 hours.

7.1.1 Basic Definitions and Examples

Definition 7.1.1: Modular Arithmetic

Given an integer $m > 0$, we say that two integers x and y are **congruent modulo** m if they differ by a multiple of m. This is denoted as:

$$x \equiv y \pmod{m} \iff m \text{ divides } (x-y).$$

The set of all integers congruent to a given integer x modulo m is called the **residue class** of x modulo m.

Example.

Calculating Time

When calculating time, we use modular arithmetic with $m = 12$. For example, 13 hours after 1 pm is 2 am, not 14. This is because $13 \equiv 1 \pmod{12}$, so we wrap around after 12 hours.

7.1.2 Arithmetic Operations in Modular Arithmetic

In modular arithmetic, addition, subtraction, and multiplication can be performed by reducing the result modulo m. This simplifies calculations by keeping intermediate results within the range $\{0, 1, \ldots, m-1\}$.

Claim: Modular Addition and Multiplication

For any integers x, y, and $m > 0$,

$$(x + y) \pmod{m} \equiv (x \pmod{m} + y \pmod{m}) \pmod{m},$$

and

$$(x \cdot y) \pmod{m} \equiv (x \pmod{m} \cdot y \pmod{m}) \pmod{m}.$$

Proof for Claim.

This follows directly from the definition of congruence modulo m. By reducing intermediate results modulo m, we ensure that the final result remains within the desired range.

7.1.3 Set Representation of Modular Arithmetic

Definition 7.1.2: Residue Classes

For a given modulus m, the set of integers can be partitioned into m disjoint sets called **residue classes**. Each residue class contains all integers that are congruent to a specific integer modulo m. These classes are represented by the unique elements in the range $\{0, 1, \dots, m - 1\}$.

Remark.

The residue classes modulo m form a complete set of representatives for the integers under congruence modulo m. This means that every integer belongs to exactly one residue class modulo m.

7.2 Exponentiation in Modular Arithmetic

Exponentiation is a common operation in cryptographic algorithms. Efficient computation of $x^y \pmod{m}$ is crucial for performance, especially when dealing with large exponents.

7.2.1 Repeated Squaring Algorithm

Theorem 7.2.1: Repeated Squaring

The repeated squaring algorithm computes $x^y \pmod{m}$ efficiently by recursively squaring the base and reducing modulo m at each step. The algorithm runs in $O(\log y)$ time, making it suitable for large exponents.

Proof. The algorithm works by expressing y in binary and using the fact that $x^{2a} = (x^a)^2$ and $x^{2a+1} = x \cdot (x^a)^2$. This reduces the number of multiplications required from $O(y)$ to $O(\log y)$. □

7.3 Bijections and Inverses

A bijection is a function that is both injective (one-to-one) and surjective (onto). In modular arithmetic, bijections are closely related to the existence of inverses.

Definition 7.3.1: Bijection

A function $f : A \to B$ is a **bijection** if every element in B has a unique pre-image in A. This means that f is both injective and surjective.

Lemma 7.3.2: Bijection and Inverses

For a finite set A, a function $f : A \to A$ is a bijection if and only if it has an inverse function $g : A \to A$ such that $g(f(x)) = x$ for all $x \in A$.

7.4 Multiplicative Inverses in Modular Arithmetic

In modular arithmetic, the multiplicative inverse of an integer x modulo m is an integer y such that $x \cdot y \equiv 1 \pmod{m}$. The existence of such an inverse depends on the greatest common divisor (gcd) of x and m.

Theorem 7.4.1: Existence of Multiplicative Inverse

Let m and x be positive integers. The multiplicative inverse of x modulo m exists if and only if $\gcd(x, m) = 1$. Moreover, when the inverse exists, it is unique modulo m.

Proof. If $\gcd(x, m) = 1$, then by the Extended Euclidean Algorithm, there exist integers a and b such that $ax + bm = 1$. Reducing this equation modulo m gives $ax \equiv 1 \pmod{m}$, so a is the multiplicative inverse of x modulo m. Conversely, if an inverse exists, then $\gcd(x, m) = 1$. $\square$

7.5 Euclid's Algorithm

Euclid's Algorithm is an efficient method for computing the greatest common divisor (gcd) of two integers. It is based on the principle that $\gcd(x, y) = \gcd(y, x \pmod{y})$.

Theorem 7.5.1: Euclid's Algorithm

For any integers x and y, $\gcd(x, y) = \gcd(y, x \pmod{y})$. This allows us to compute the gcd efficiently by repeatedly applying this reduction.

Proof. The correctness of Euclid's Algorithm follows from the fact that any common divisor of x and y is also a common divisor of y and $x \pmod{y}$. The algorithm terminates when one of the numbers becomes zero, at which point the other number is the gcd. $\square$

7.5.1 Extended Euclid's Algorithm

The Extended Euclidean Algorithm not only computes the gcd of two integers but also finds integers a and b such that $ax + by = \gcd(x, y)$.

Theorem 7.5.2: Extended Euclidean Algorithm

For any integers x and y, the Extended Euclidean Algorithm computes integers a and b such that $ax + by = \gcd(x, y)$.

Proof. The algorithm works by keeping track of the coefficients a and b as it recursively applies Euclid's Algorithm. The base case is when $y = 0$, in which case $a = 1$ and $b = 0$. The recursive step updates the coefficients based on the quotient and remainder of the division $x \div y$. □

7.6 Fundamental Theorem of Arithmetic

The Fundamental Theorem of Arithmetic states that every integer greater than 1 can be uniquely factored into a product of prime numbers, up to the order of the factors.

Theorem 7.6.1: Fundamental Theorem of Arithmetic

Every integer $n > 1$ can be expressed uniquely as a product of prime numbers:

$$n = p_1 p_2 \cdots p_k,$$

where each p_i is a prime number, and the factorization is unique up to the order of the factors.

Proof. The existence of a prime factorization can be shown by induction. The uniqueness follows from the fact that if a prime p divides a product of primes, it must divide one of them. This is a consequence of the Euclidean Algorithm and the properties of gcd. □

7.7 Chinese Remainder Theorem

The Chinese Remainder Theorem (CRT) provides a way to solve a system of simultaneous congruences with pairwise coprime moduli.

Theorem 7.7.1: Chinese Remainder Theorem

Let $n_1, n_2, \ldots, n_k$ be pairwise coprime integers. Then, for any integers $a_1, a_2, \ldots, a_k$, there exists a unique integer x modulo $N = n_1 n_2 \cdots n_k$ such that:

$$x \equiv a_i \pmod{n_i} \quad \text{for all } i = 1, 2, \ldots, k.$$

Proof. The proof constructs a solution using the inverses of the moduli and shows that the solution is unique modulo N. The uniqueness follows from the fact that the difference between any two solutions must be a multiple of N, which is impossible if the solutions are within the range $\{0, 1, \ldots, N-1\}$. $\square$

Chapter 8

07 - Euclid, FLT, CRT, RSA

8.1 Public Key Cryptography

Public key cryptography is a revolutionary concept in the field of cryptography, allowing secure communication between two parties without the need for a shared secret key. The most famous public-key cryptosystem is RSA, named after its inventors Rivest, Shamir, and Adleman.

8.1.1 Basic Setting

In the traditional cryptographic setting, Alice and Bob wish to communicate securely over an insecure channel, while Eve, an eavesdropper, tries to intercept and decipher their messages. Alice encrypts her message x using an encryption function E, sending $E(x)$ to Bob. Bob then decrypts the message using his decryption function D, recovering x as $D(E(x)) = x$.

Remark.

> In private-key cryptography, Alice and Bob must share a secret key beforehand. Public-key cryptography, however, allows Alice to send Bob a message without any prior shared secret, using a digital lock that only Bob can open.

8.1.2 RSA Cryptosystem

The RSA cryptosystem is based on modular arithmetic and the difficulty of factoring large composite numbers. Each user has a public key (N, e) and a private key d. The public key is known to everyone, while the private key is kept secret.

Definition 8.1.1: RSA Key Generation

To generate an RSA key pair:

1. Choose two large prime numbers p and q.
2. Compute $N = pq$.
3. Choose an integer e such that $\gcd(e, (p-1)(q-1)) = 1$.
4. Compute d as the modular inverse of e modulo $(p-1)(q-1)$, i.e., $d \equiv e^{-1} \pmod{(p-1)(q-1)}$.

The public key is (N, e), and the private key is d.

8.1.3 Encryption and Decryption

Definition 8.1.2: RSA Encryption

To encrypt a message x (an integer modulo N), Alice computes:

$$y \equiv x^e \pmod{N}$$

and sends y to Bob.

Definition 8.1.3: RSA Decryption

To decrypt the message y, Bob computes:

$$x \equiv y^d \pmod{N}$$

and recovers the original message x.

Example.

> RSA Example

Let $p = 5$, $q = 11$, and $N = pq = 55$. Choose $e = 3$, which is coprime to $(p-1)(q-1) = 40$. The private key is $d \equiv 3^{-1} \pmod{40} \equiv 27$. If Alice sends $x = 13$, the encrypted message is $y \equiv 13^3 \pmod{55} \equiv 52$. Bob decrypts it as $52^{27} \pmod{55} \equiv 13$, recovering the original message.

8.2 Correctness of RSA

The correctness of RSA relies on Fermat's Little Theorem and the Chinese Remainder Theorem.

Theorem 8.2.1: Fermat's Little Theorem

For any prime p and integer a such that $\gcd(a,p)=1$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Theorem 8.2.2: Correctness of RSA

For any message $x \in \{0,1,\ldots,N-1\}$,

$$(x^e)^d \equiv x \pmod{N}.$$

Proof. The proof uses Fermat's Little Theorem and the fact that $ed \equiv 1 \pmod{(p-1)(q-1)}$. By Fermat's Little Theorem, $x^{k(p-1)(q-1)} \equiv 1 \pmod{p}$ and $x^{k(p-1)(q-1)} \equiv 1 \pmod{q}$ for some integer k. Thus, $x^{ed} \equiv x \pmod{p}$ and $x^{ed} \equiv x \pmod{q}$. By the Chinese Remainder Theorem, $x^{ed} \equiv x \pmod{N}$. □

8.3 Security of RSA

The security of RSA is based on the difficulty of factoring large composite numbers. Eve, the eavesdropper, cannot easily compute x from $y \equiv x^e \pmod{N}$ without knowing the private key d. The only known way to compute d is to factor N into p and q, which is computationally infeasible for large N.

Remark.

> The security of RSA has not been formally proven, but it is widely believed that breaking RSA is as hard as factoring large numbers.

8.4 Implementation of RSA

Implementing RSA involves two main tasks: generating large prime numbers and performing modular exponentiation.

8.4.1 Generating Large Primes

Theorem 8.4.1: Prime Number Theorem

Let $\pi(n)$ denote the number of primes less than or equal to n. Then,

$$\pi(n) \sim \frac{n}{\ln n}.$$

This means that the density of primes around n is approximately $1/\ln n$.

Remark.

> To generate a large prime, Bob can randomly select numbers of the desired size and test them for primality. The Prime Number Theorem ensures that he will find a prime after testing roughly $\ln n$ numbers.

8.4.2 Modular Exponentiation

Modular exponentiation can be efficiently computed using the repeated squaring algorithm, which runs in $O(\log y)$ time for computing $x^y \pmod{N}$.

Remark.

> The repeated squaring algorithm allows Alice and Bob to perform modular exponentiation efficiently, even for very large exponents.

Chapter 9

08 - Polynomials, Secret Sharing

9.1 Polynomials

Polynomials are a fundamental class of functions with wide applications in various fields such as cryptography, communication, and computational geometry. In this note, we explore the properties of polynomials, particularly in the context of modular arithmetic, and their application in secret-sharing schemes.

9.1.1 Basic Definitions

Definition 9.1.1: Polynomial

A polynomial in a single variable x is an expression of the form:

$$p(x) = a_d x^d + a_{d-1} x^{d-1} + \cdots + a_1 x + a_0,$$

where $a_d, a_{d-1}, \ldots, a_0$ are coefficients, and d is the degree of the polynomial. The polynomial can be evaluated at any value of x by substituting the value into the expression.

Definition 9.1.2: Root of a Polynomial

A value a is called a **root** of the polynomial $p(x)$ if $p(a) = 0$. For example, the polynomial $p(x) = x^2 - 4$ has roots at $x = 2$ and $x = -2$.

9.1.2 Key Properties of Polynomials

Polynomials have two key properties that make them particularly useful:

Theorem 9.1.3: Property 1: Maximum Number of Roots

A non-zero polynomial of degree d has at most d roots.

Theorem 9.1.4: Property 2: Unique Polynomial Interpolation

Given $d+1$ pairs $(x_1, y_1), \dots, (x_{d+1}, y_{d+1})$ with all x_i distinct, there exists a unique polynomial $p(x)$ of degree at most d such that $p(x_i) = y_i$ for all i.

9.2 Polynomial Interpolation

Polynomial interpolation is the process of finding a polynomial that passes through a given set of points. The Lagrange interpolation method is a standard technique for this purpose.

9.2.1 Lagrange Interpolation

Definition 9.2.1: Lagrange Interpolation

Given $d+1$ points $(x_1, y_1), \dots, (x_{d+1}, y_{d+1})$, the Lagrange interpolation polynomial $p(x)$ is given by:

$$p(x) = \sum_{i=1}^{d+1} y_i \Delta_i(x),$$

where $\Delta_i(x)$ is the Lagrange basis polynomial defined as:

$$\Delta_i(x) = \frac{\prod_{j \neq i}(x - x_j)}{\prod_{j \neq i}(x_i - x_j)}.$$

Example.

> Lagrange Interpolation Example

Suppose we want to find a degree-2 polynomial passing through the points $(1, 1)$, $(2, 2)$, and $(3, 4)$. The Lagrange basis polynomials are:

$$\Delta_1(x) = \frac{(x-2)(x-3)}{(1-2)(1-3)}, \quad \Delta_2(x) = \frac{(x-1)(x-3)}{(2-1)(2-3)}, \quad \Delta_3(x) = \frac{(x-1)(x-2)}{(3-1)(3-2)}.$$

The interpolating polynomial is:

$$p(x) = 1 \cdot \Delta_1(x) + 2 \cdot \Delta_2(x) + 4 \cdot \Delta_3(x).$$

9.3 Polynomial Division

Polynomial division is analogous to integer division and is useful for proving properties of polynomials.

Definition 9.3.1: Polynomial Division

Given two polynomials $p(x)$ and $q(x)$, we can write:

$$p(x) = q'(x)q(x) + r(x),$$

where $q'(x)$ is the quotient, and $r(x)$ is the remainder with degree less than the degree of $q(x)$.

Example.

> Polynomial Division Example

Divide $p(x) = x^3 + x^2 - 1$ by $q(x) = x - 1$. The result is:

$$p(x) = (x^2 + 2x + 2)(x - 1) + 1.$$

9.4 Finite Fields

Polynomial properties such as interpolation and the maximum number of roots also hold when working over finite fields, such as integers modulo a prime p.

Definition 9.4.1: Finite Field

A finite field, denoted F_p or $\mathrm{GF}(p)$, is a set of integers $\{0, 1, \ldots, p-1\}$ with addition, subtraction, multiplication, and division (except by zero) defined modulo p.

Remark.

> Working over finite fields is particularly useful in cryptography because it allows for exact computations without the need for floating-point arithmetic.

9.5 Secret Sharing

Secret sharing is a cryptographic technique that allows a secret to be distributed among a group of participants such that only a specified subset of participants can reconstruct the secret.

9.5.1 Shamir's Secret Sharing

Definition 9.5.1: Shamir's Secret Sharing

Given a secret s and a prime $q > s$, a polynomial $P(x)$ of degree $k-1$ is chosen such that $P(0) = s$. Each participant i receives a share $P(i)$. Any k participants can reconstruct $P(x)$ using Lagrange interpolation and recover $s = P(0)$. However, any group of fewer than k participants gains no information about s.

Example.

> Secret Sharing Example

Suppose $s = 1$, $n = 5$, and $k = 3$. Choose $P(x) = 3x^2 + 5x + 1$ over GF(7). The shares are:

$$P(1) = 2, \quad P(2) = 2, \quad P(3) = 1, \quad P(4) = 6, \quad P(5) = 3.$$

Any three participants can reconstruct $P(x)$ and recover $s = 1$, but any two participants cannot.

Chapter 10

09 - Error Correcting Codes

10.1 Error Correcting Codes

Error correcting codes are essential for reliable communication over unreliable channels. These codes introduce redundancy into messages to protect against errors such as packet loss or corruption. In this note, we focus on algebraic codes, particularly Reed-Solomon codes, which are based on polynomial interpolation over finite fields.

10.1.1 Erasure Errors

Erasure errors occur when packets are lost during transmission. To handle up to k erasures, we encode a message of n packets into $n+k$ packets using polynomial interpolation.

Definition 10.1.1: Reed-Solomon Encoding

Given a message $m_1, m_2, \ldots, m_n$, we construct a polynomial $P(x)$ of degree $n-1$ such that $P(i) = m_i$ for $1 \leq i \leq n$. The encoded codeword consists of $n+k$ packets, where $c_j = P(j)$ for $1 \leq j \leq n+k$.

Example.

Erasure Error Example

Suppose $n = 4$ and $k = 2$. The message $m_1 = 3$, $m_2 = 1$, $m_3 = 5$, $m_4 = 0$ is encoded into $c_1 = 3$, $c_2 = 1$, $c_3 = 5$, $c_4 = 0$, $c_5 = 6$, $c_6 = 1$. If packets 2 and 6 are lost, Bob can reconstruct $P(x)$ using the remaining packets and recover the original message.

10.1.2 General Errors

General errors occur when packets are corrupted during transmission. To handle up to k general errors, we encode a message of n packets into $n+2k$ packets.

Definition 10.1.2: Reed-Solomon Encoding for General Errors

Given a message $m_1, m_2, \ldots, m_n$, we construct a polynomial $P(x)$ of degree $n-1$ such that $P(i) = m_i$ for $1 \leq i \leq n$. The encoded codeword consists of $n+2k$ packets, where $c_j = P(j)$ for $1 \leq j \leq n+2k$.

Example.

General Error Example

Suppose $n = 3$ and $k = 1$. The message $m_1 = 3$, $m_2 = 0$, $m_3 = 6$ is encoded into $c_1 = 3$, $c_2 = 0$, $c_3 = 6$, $c_4 = 0$, $c_5 = 3$. If c_1 is corrupted to 2, Bob can use the Berlekamp-Welch algorithm to reconstruct $P(x)$ and recover the original message.

10.2 Polynomial Interpolation

Polynomial interpolation is the process of finding a polynomial that passes through a given set of points. The Lagrange interpolation method is a standard technique for this purpose.

Definition 10.2.1: Lagrange Interpolation

Given $d+1$ points $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$, the Lagrange interpolation polynomial $p(x)$ is given by:

$$p(x) = \sum_{i=1}^{d+1} y_i \Delta_i(x),$$

where $\Delta_i(x)$ is the Lagrange basis polynomial defined as:

$$\Delta_i(x) = \frac{\prod_{j \neq i}(x - x_j)}{\prod_{j \neq i}(x_i - x_j)}.$$

Example.

Lagrange Interpolation Example

Suppose we want to find a degree-2 polynomial passing through the points $(1, 1)$, $(2, 2)$, and $(3, 4)$. The Lagrange basis polynomials are:

$$\Delta_1(x) = \frac{(x-2)(x-3)}{(1-2)(1-3)}, \quad \Delta_2(x) = \frac{(x-1)(x-3)}{(2-1)(2-3)}, \quad \Delta_3(x) = \frac{(x-1)(x-2)}{(3-1)(3-2)}.$$

The interpolating polynomial is:

$$p(x) = 1 \cdot \Delta_1(x) + 2 \cdot \Delta_2(x) + 4 \cdot \Delta_3(x).$$

10.3 Berlekamp-Welch Algorithm

The Berlekamp-Welch algorithm is used to correct general errors in Reed-Solomon codes. It involves solving a system of linear equations to find the error-locator polynomial and the original message polynomial.

Definition 10.3.1: Berlekamp-Welch Algorithm

Given $n + 2k$ received packets $r_1, r_2, \ldots, r_{n+2k}$, the algorithm solves for the error-locator polynomial $E(x)$ and the polynomial $Q(x) = P(x)E(x)$. The original message polynomial $P(x)$ is then obtained by dividing $Q(x)$ by $E(x)$.

Example.

> Berlekamp-Welch Example

Suppose $n = 3$, $k = 1$, and the received packets are $r_1 = 2$, $r_2 = 0$, $r_3 = 6$, $r_4 = 0$, $r_5 = 3$.

The algorithm solves for $E(x) = x - 1$ and $Q(x) = x^3 + 6$, yielding $P(x) = x^2 + x + 1$.

10.4 Minimum Distance of Reed-Solomon Codes

The minimum distance of a code is the smallest Hamming distance between any two codewords. Reed-Solomon codes have optimal distance properties.

Theorem 10.4.1: Minimum Distance of Reed-Solomon Codes

The Reed-Solomon code that takes n message characters to a codeword of size $n+2k$ has a minimum distance of $2k + 1$.

Proof. The proof involves showing that the minimum distance is both $\leq 2k+1$ and $\geq 2k+1$. The first part is shown by constructing two codewords with distance $2k + 1$, and the second part is shown by contradiction, assuming the distance could be less than $2k + 1$. $\square$