THE

ART

OF

ALGORITHM

ANALYSIS

FROM FOUNDATIONS TO PRACTICE

Mahdi

LIVING FIRST EDITION

Analy

ALGORITHMS • ABSTRACTION • ANALYSIS • ART

"From ancient counting stones to quantum algorithms—every data structure tells the story of human ingenuity."

LIVING FIRST EDITION

Updated October 19, 2025

LICENSE & DISTRIBUTION

The Art of Algorithmic Analysis: Algorithmic Cost Analysis and Asymptotic Reasoning

A Living Architecture of Computing

The Art of Algorithmic Analysis is released under the **Creative Commons Attribution-ShareAlike 4.0 International License** (CC BY-SA 4.0).

FORMAL LICENSE TERMS

You are free to:

- **Share** — copy and redistribute the material in any medium or format for any purpose, even commercially.
- **Adapt** — remix, transform, and build upon the material for any purpose, even commercially.

Under the following terms:

- **Attribution** — You must give appropriate credit to Mahdi, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
- **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

DISTRIBUTION & SOURCE ACCESS

Repository: The complete source code (LaTeX, diagrams, examples) is available at:
https://github.com/m-mdy-m/algorithms-data-structures/tree/main/books/books

Preferred Citation Format:
Mahdi. (2025). *The Art of Algorithmic Analysis*. Retrieved from
https://github.com/m-mdy-m/algorithms-data-structures/tree/main/books/books

Version Control: This is a living document. Check the repository for the most current version and revision history.

WARRANTIES & DISCLAIMERS

No Warranty: This work is provided "AS IS" without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.

Limitation of Liability: In no event shall Mahdi be liable for any direct, indirect, incidental, special, exemplary, or consequential damages arising from the use of this work.

Educational Purpose: This work is intended for educational and research purposes. Practical implementation of algorithms and techniques should be thoroughly tested and validated for production use.

TECHNICAL SPECIFICATIONS

Typeset with: LaTeX using Charter and Palatino font families
Graphics: TikZ and custom illustrations
Standards: Follows academic publishing conventions
Encoding: UTF-8 with full Unicode support
Format: Available in PDF, and LaTeX source formats

License last updated: October 19, 2025

For questions about licensing, contact: bitsgenix@gmail.com

Contents

Preface

EVERY RIGOROUS JOURNEY begins with a question. For this book, that question was deceptively simple: *How do we truly measure the cost of computation?*

Genesis of This Work

During my studies in computer science, I encountered a persistent frustration: algorithmic analysis was presented fragmentedly across courses and textbooks. Asymptotic notation appeared in one context, recurrence relations in another, amortized analysis as an advanced topic buried in data structures courses. Each technique existed in isolation, its connection to broader analytical frameworks obscured.
I wanted something different—a unified treatment that explains not just *how* to analyze algorithms, but *why* these particular analytical methods emerged, *how* they relate to one another, and *when* each provides the deepest insight. Unable to find such a resource, I decided to create it.
This book represents that effort: a comprehensive synthesis of algorithmic analysis techniques, built from extensive research across the literature of computer science, applied mathematics, and complexity theory.

A Living Document

This work is fundamentally different from traditional textbooks in one crucial respect: **it is alive and evolving**.
As I continue researching algorithmic analysis—discovering new connections, understanding techniques more deeply, encountering novel applications—this book grows and improves. Each week brings refinements: clearer explanations, additional examples, connections I hadn't previously recognized, corrections to subtle errors.
The book you're reading today is more complete than the version that existed last month. The version that will exist next month will be more refined than what you see now. This living nature means:

- **Continuous Improvement**: Sections evolve as my understanding deepens through ongoing research

- **Integration of New Research**: Recent developments in algorithmic analysis are incorporated as they emerge
- **Community Feedback**: Reader corrections, suggestions, and insights strengthen the work
- **Transparency About Limitations**: I openly acknowledge where current understanding is incomplete

Traditional textbooks freeze knowledge at publication time. This book remains fluid, growing alongside both my research and the field itself.

Foundation on Giants' Shoulders

While this book represents my synthesis and presentation, the knowledge it contains stands on the foundational work of brilliant computer scientists and mathematicians:

What Makes This Book Distinctive

Several characteristics distinguish this treatment from existing resources:

Analysis-First Perspective Most algorithms texts treat analysis as a supporting tool for understanding algorithms. This book inverts that relationship: analysis techniques are the primary focus, with algorithms serving as examples to illustrate analytical methods.

Comprehensive Coverage From fundamental asymptotic notation to research-level topics like cache-oblivious algorithms and parameterized complexity, the book spans the full spectrum of analytical techniques.

Unified Framework Rather than presenting techniques in isolation, the book constantly highlights connections—how methods relate, when one technique is preferred over another, why certain problems require specific analytical approaches.

Progressive Development Concepts build systematically. Each chapter assumes only material from preceding chapters, allowing readers to develop understanding incrementally without gaps.

Mathematical Rigor with Intuition Every formal development begins with intuitive motivation. Proofs serve understanding, revealing *why* results hold, not merely verifying *that* they hold.

Living Evolution Perhaps most importantly: this book acknowledges its own incompleteness and commits to continuous improvement through ongoing research and community feedback.

Who Should Read This Book

This book serves multiple audiences:

Undergraduate students who have completed introductory algorithms and want deeper analytical understanding. You should be comfortable with basic programming, discrete mathematics, and elementary proofs.

Graduate students needing advanced analysis techniques for research. This book provides the analytical toolkit for reading algorithms research and analyzing novel algorithms.

Practitioners seeking principled frameworks for algorithm selection and performance prediction. The analytical perspective here complements practical engineering experience.

Self-learners with intellectual curiosity about algorithmic efficiency. If you've wondered *why* certain algorithms are considered efficient, this book provides rigorous answers.

Whatever your background, you should bring mathematical maturity—comfort with abstraction, formal definitions, and logical reasoning. Specific prerequisites (discrete mathematics, probability, calculus) are reviewed in Part I, but prior exposure helps.

Structure Overview

The book organizes into six major parts:

1. **Foundations** — Establishing context, prerequisites, and reading strategies
2. **Foundations of Algorithmic Analysis** — Core techniques: asymptotic notation, recurrences, best/worst/average case, probabilistic analysis
3. **Advanced Analysis Techniques** — Amortized analysis, space complexity, memory hierarchy effects, parallel analysis
4. **Lower Bounds and Optimality** — Understanding fundamental limits on algorithmic efficiency

5. **Specialized Topics and Applications** — Analysis techniques for specific algorithm paradigms
6. **Practical Considerations and Case Studies** — Bridging theory to real-world performance

Each part builds on preceding material, developing increasingly sophisticated analytical capabilities.

A Note on Rigor

This book takes mathematical rigor seriously—not as pedantic formalism, but as the discipline enabling precise reasoning about complex systems.

Informal intuition is valuable but insufficient. Rigorous analysis distinguishes what intuition conflates: logarithmic from linear growth, amortized from average cost, theoretical complexity from practical performance. Mathematical precision is not obstacle but tool—enabling reliable reasoning, clear communication, and principled decision-making.

That said, rigor serves understanding. Every formal development begins with intuition. Proofs reveal insights, not just verify results. If formalism feels overwhelming initially, prioritize key ideas on first reading, returning later for proof details.

Final Thoughts

Algorithmic analysis is often presented as necessary prerequisite—technical machinery required before "real" algorithms work begins. This perspective misses something fundamental.

Analysis is not merely evaluation. It is a framework for *thinking* about computation—revealing patterns in costs, exposing hidden problem structure, developing intuition about what solutions might be possible.

Mastering these techniques changes how you approach problems. You develop analytical lenses that transform how you perceive computational challenges. This cognitive shift is the ultimate goal.

The journey ahead is demanding. You will encounter abstract mathematics, work through detailed proofs, solve challenging exercises. But the reward—deep, rigorous understanding of computational cost—justifies the effort.

And remember: this book is alive. As research continues and understanding deepens, the work improves. Your engagement—through questions, corrections, and insights—contributes to that improvement.

Welcome to **The Art of Algorithmic Analysis**. Let's begin.

Mahdi
2025-2026

Acknowledgments

I would like to express my gratitude to everyone who supported me during the creation of this book. Special thanks to the open-source community for their invaluable resources and to all those who reviewed early drafts and provided feedback.

Part I

Foundations

The art of algorithmic analysis demands both mathematical precision and computational intuition. This part provides the rigorous mathematical foundation upon which all subsequent analysis techniques rest.

Unlike typical prerequisite chapters that offer superficial reviews, we present these topics with the depth required for advanced analysis: complete proofs, non-trivial examples, and connections to algorithm design that you won't find elsewhere.

What Makes This Different:

- ***Depth:*** *We prove theorems completely, not just state them*
- ***Motivation:*** *Every topic connects directly to algorithm analysis*
- ***Rigor:*** *We use precise mathematical language throughout*
- ***Practice:*** *Exercises range from warmup to research-level problems*

> *"In mathematics, you don't understand things. You just get used to them."*
>
> — John von Neumann

Chapter 1

Mathematical Foundations I: Discrete Structures

DISCRETE MATHEMATICS is the native language of computer science. Unlike the continuous mathematics of physics and engineering, we work with countable, distinct objects: integers, graphs, logic formulas, finite automata. This chapter develops the discrete structures essential for algorithmic reasoning.

Chapter Organization:

This chapter covers five fundamental areas:

1. **Logic and Proof Theory** — The language of correctness
2. **Set Theory and Relations** — The foundation of abstraction
3. **Combinatorial Analysis** — Counting and enumeration
4. **Graph Theory** — Modeling relationships
5. **Algebraic Structures** — Symmetry and structure

Each section includes:

- Complete proofs of major theorems
- Worked examples from algorithm analysis
- Historical context and development
- Exercises at three levels: Basic, Intermediate, Advanced
- Research-level problems marked with $\star$

Prerequisites: Mathematical maturity at the level of a first-year university mathematics course. Comfort with symbolic manipulation and logical reasoning.

1.1 Logic and Proof Theory

Before we can prove algorithms correct or establish complexity bounds, we must understand the nature of proof itself. This section develops propositional and predicate logic, formal proof systems, and the art of mathematical argumentation.
Why This Matters: *Every correctness proof, every lower bound, every impossibility result rests on logical foundations. Mastery of proof techniques is not optional—it is the essence of computer science.*

1.1.1 Propositional Logic

Syntax: Well-Formed Formulas

Formal Definition.

Parse Trees and Unique Readability.

Inductive Definition of Formulas.

Semantics: Truth Tables and Valuations

Boolean Valuations.

Truth-Functional Connectives: $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$**.**

Tautologies, Contradictions, and Contingencies.

Logical Equivalence and Normal Forms

Substitution and Equivalence Relations.

Conjunctive Normal Form (CNF).

Disjunctive Normal Form (DNF).

Theorem: Every Formula Has Equivalent CNF/DNF.

Proof Systems for Propositional Logic

Natural Deduction.

Sequent Calculus.

Resolution Principle and SAT Solving.

1.1.2 Predicate Logic (First-Order Logic)

Syntax: Terms, Predicates, Quantifiers

Variables, Constants, Function Symbols.

Universal ($\forall$) and Existential ($\exists$) Quantifiers.

Free vs. Bound Variables.

Semantics: Structures and Models

Interpretations and Domains.

Satisfaction and Truth in a Model.

Validity and Logical Consequence.

Prenex Normal Form

Quantifier Scope and Movement Rules.

Skolemization and Herbrand's Theorem.

Decidability and Undecidability

Church-Turing Theorem: First-Order Logic is Undecidable.

Decidable Fragments and Applications to Verification.

1.1.3 Proof Techniques and Strategies

Direct Proof

Structure and Template.

Example: Properties of Even/Odd Integers.

Proof by Contradiction (Reductio ad Absurdum)

Strategy and When to Use.

Example: $\sqrt{2}$ is Irrational.

Example: Infinitude of Primes (Euclid).

Proof by Contrapositive

Logical Equivalence to Direct Proof.

When Contrapositive is Easier.

Mathematical Induction

Weak Induction Principle.

Strong Induction (Complete Induction).

Well-Ordering Principle and Equivalence.

Structural Induction on Recursive Data Types.

Example: Correctness of Merge Sort.

Proof by Construction

Existence Proofs: Constructive vs. Non-Constructive.

Algorithmic Content of Constructive Proofs.

Proof by Counterexample

Disproving Universal Statements.

Minimal Counterexamples.

Pigeonhole Principle

Simple Form: $n + 1$ Objects in n Boxes.

Generalized Form: Distribution Arguments.

Example: Birthday Paradox.

Example: Lower Bounds for Sorting.

1.1.4 Common Proof Errors and How to Avoid Them

Circular Reasoning (Begging the Question)

Proof by Example

Confusion of Necessary and Sufficient Conditions

Quantifier Errors

Vacuous Truth

1.1.5 Advanced Topics in Logic

Temporal Logic and Program Verification

Modal Logic and Epistemic Logic

Gödel's Incompleteness Theorems (Informal)

1.1.6 Exercises

Warmup Problems (20 problems)

Standard Problems (30 problems)

Challenging Problems (25 problems)

Research-Level Problems ($\star$) (10 problems)

1.2 Set Theory, Functions, and Relations

Sets are the foundation of modern mathematics. Functions and relations provide the vocabulary for describing computations and data structures. This section develops these concepts rigorously, with emphasis on applications to algorithm design.

1.2.1 Axiomatic Set Theory (Informal)

Zermelo-Fraenkel Axioms (Intuitive Overview)

Russell's Paradox and the Need for Axioms

The Axiom of Choice and Its Consequences

1.2.2 Basic Set Operations

Set Builder Notation and Comprehension

Union, Intersection, Difference, Symmetric Difference

Power Set and Cartesian Product

De Morgan's Laws and Duality

1.2.3 Cardinality and Counting

Finite, Countable, and Uncountable Sets

Cantor's Diagonal Argument

Schröder-Bernstein Theorem

Continuum Hypothesis

1.2.4 Functions: Mappings Between Sets

Definition and Notation

Domain, Codomain, Range.

Function Composition and Associativity.

Types of Functions

Injective (One-to-One) Functions.

Surjective (Onto) Functions.

Bijective Functions and Inverses.

Theorem: Composition of Bijections is Bijective.

Special Functions

Identity Function.

Constant Functions.

Projection Functions.

Image and Preimage

Properties of Image and Preimage.

Inverse Image and Set Operations.

1.2.5 Relations

Binary Relations

Definition as Subsets of Cartesian Products.

Representation: Matrices, Graphs, Tables.

Properties of Relations

Reflexivity, Irreflexivity.

Symmetry, Antisymmetry, Asymmetry.

Transitivity.

Equivalence Relations

Definition and Examples.

Equivalence Classes and Partitions.

Theorem: Equivalence Relations Induce Partitions.

Quotient Sets.

Partial Orders

Definition: Reflexive, Antisymmetric, Transitive.

Hasse Diagrams.

Maximal and Minimal Elements.

Least Upper Bound (Supremum) and Greatest Lower Bound (Infimum).

Lattices.

Total Orders (Linear Orders)

Definition and Examples.

Well-Orderings.

Application: Topological Sorting.

1.2.6 Exercises

Warmup Problems

Standard Problems

Challenging Problems

Research Problems ($\star$)

1.3 Combinatorial Analysis and Enumeration

Combinatorics is the mathematics of counting. It answers fundamental questions: How many possible inputs exist? How many steps must an algorithm take? What is the size of the search space?
This section goes far beyond basic permutations and combinations, developing generating functions, recurrence relations, and asymptotic enumeration techniques essential for algorithm analysis.

1.3.1 Fundamental Counting Principles

Sum Rule (Addition Principle)

Product Rule (Multiplication Principle)

Division Principle (Quotient Principle)

Bijection Principle

1.3.2 Permutations and Arrangements

Permutations of n Distinct Objects

Formula: $n!$

Stirling's Approximation (Preview).

Permutations with Repetition

Multiset Permutations: $\frac{n!}{n_1!n_2!\cdots n_k!}$

k-Permutations: $P(n,k) = \frac{n!}{(n-k)!}$

Circular Permutations

$(n-1)!$ **Arrangements.**

Derangements

Permutations with No Fixed Points.

Formula: $D_n = n! \sum_{i=0}^{n} \frac{(-1)^i}{i!}$

Asymptotic: $D_n \sim \frac{n!}{e}$

1.3.3 Combinations and Selections

Binomial Coefficients: $\binom{n}{k}$

Definition and Pascal's Triangle.

Symmetry: $\binom{n}{k} = \binom{n}{n-k}$

Vandermonde's Identity.

Binomial Theorem

$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$

Applications to Probability.

Multinomial Coefficients

$\binom{n}{k_1,k_2,\ldots,k_m} = \frac{n!}{k_1!k_2!\cdots k_m!}$

Multinomial Theorem.

Combinations with Repetition

Stars and Bars Method.

Formula: $\binom{n+k-1}{k}$

1.3.4 Inclusion-Exclusion Principle

Two-Set Formula

General Formula for n Sets

Applications

Counting Surjections.

Euler's Totient Function.

Derangement Formula (Alternative Derivation).

1.3.5 Pigeonhole Principle and Ramsey Theory

Simple Pigeonhole Principle

Generalized Pigeonhole Principle

Ramsey's Theorem (Introduction)

$R(3,3) = 6$ **— The Party Problem.**

Bounds on $R(m,n)$.

1.3.6 Generating Functions

> *Generating functions transform combinatorial problems into algebra. They are indispensable for solving recurrences and asymptotic enumeration.*

Ordinary Generating Functions (OGF)

Definition: $A(x) = \sum_{n=0}^{\infty} a_n x^n$

Operations on Generating Functions.

Convolution and Product of GFs.

Exponential Generating Functions (EGF)

Definition: $A(x) = \sum_{n=0}^{\infty} a_n \frac{x^n}{n!}$

When to Use EGF vs. OGF.

Solving Recurrences with Generating Functions

Example: Fibonacci Numbers.

Example: Catalan Numbers.

Coefficient Extraction

Cauchy's Residue Theorem (Overview).

Asymptotic Analysis via Singularities.

1.3.7 Recurrence Relations

Linear Recurrences with Constant Coefficients

Homogeneous Recurrences.

Characteristic Equation Method.

Example: Fibonacci, Tribonacci.

Non-Homogeneous Recurrences

Method of Undetermined Coefficients.

Particular Solutions.

Divide-and-Conquer Recurrences

Form: $T(n) = aT(n/b) + f(n)$

Preview of Master Theorem (Full treatment in Part 2).

1.3.8 Catalan Numbers

Definition and Recurrence

Closed Form: $C_n = \frac{1}{n+1}\binom{2n}{n}$

Applications

Balanced Parentheses.

Binary Search Trees.

Triangulations of Polygons.

1.3.9 Stirling Numbers

Stirling Numbers of the First Kind

Unsigned: Number of Permutations with k Cycles.

Signed: Coefficients in Falling Factorial Expansion.

Stirling Numbers of the Second Kind

Definition: $S(n,k)$ — Partitions of n Elements into k Subsets.

Recurrence: $S(n,k) = k \cdot S(n-1,k) + S(n-1,k-1)$

Bell Numbers

Total Partitions: $B_n = \sum_{k=0}^{n} S(n,k)$

1.3.10 Asymptotic Methods in Combinatorics

Stirling's Approximation (Full Treatment)

Statement: $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$

Proof via Euler-Maclaurin Formula.

More Precise Expansions.

Asymptotic Analysis of Binomial Coefficients

Central Binomial Coefficient: $\binom{2n}{n} \sim \frac{4^n}{\sqrt{\pi n}}$

Saddle-Point Method (Introduction)

1.3.11 Exercises

Warmup Problems (30 problems)

Standard Problems (40 problems)

Challenging Problems (30 problems)

Research Problems ($\star$) (15 problems)

1.4 Graph Theory Foundations

Graphs are the most versatile mathematical structure in computer science. They model networks, dependencies, state spaces, data structures, and more. This section develops graph theory from first principles, emphasizing algorithmic applications.

1.4.1 Basic Definitions and Representations

Graphs: Formal Definition

Undirected Graphs: $G = (V, E)$

Directed Graphs (Digraphs): $G = (V, A)$

Multigraphs and Pseudographs.

Graph Terminology

Vertices (Nodes) and Edges (Arcs).

Degree: In-Degree, Out-Degree.

Handshaking Lemma: $\sum_{v \in V} \deg(v) = 2|E|$

Isolated, Pendant, and Universal Vertices.

Special Types of Graphs

Complete Graphs: K_n

Bipartite Graphs and Complete Bipartite Graphs: $K_{m,n}$

Cycle Graphs: C_n

Wheel Graphs: W_n

Hypercubes: Q_n

Graph Representations

Adjacency Matrix: $O(|V|^2)$ **Space.**

Adjacency List: $O(|V| + |E|)$ **Space.**

Edge List.

Incidence Matrix.

Trade-offs: Space vs. Query Time.

1.4.2 Connectivity and Paths

Walks, Trails, Paths, and Cycles

Definitions and Distinctions.

Simple Paths vs. Paths with Repetition.

Connected Graphs

Connected Components.

Strong and Weak Connectivity in Digraphs.

Distance and Diameter

Shortest Path Length: $d(u, v)$

Diameter: $\max_{u,v} d(u, v)$

Radius and Center.

Eulerian Paths and Circuits

Euler's Theorem: Necessary and Sufficient Conditions.

Fleury's Algorithm.

Applications: DNA Sequencing, Route Planning.

Hamiltonian Paths and Circuits

Definition and Examples.

Dirac's and Ore's Theorems (Sufficient Conditions).

NP-Completeness (Preview).

1.4.3 Trees and Forests

Definition of Trees

Acyclic Connected Graphs.

Equivalent Characterizations of Trees.

Properties of Trees

Theorem: A Tree with n Vertices Has $n-1$ Edges.

Leaves and Internal Nodes.

Rooted Trees

Parent, Child, Sibling, Ancestor, Descendant.

Height and Depth.

Binary Trees, k-ary Trees.

Spanning Trees

Definition and Existence.

Number of Spanning Trees: Cayley's Formula.

Minimum Spanning Tree Problem (Preview).

1.4.4 Directed Acyclic Graphs (DAGs)

Definition and Properties

Topological Sorting

Algorithm and Correctness.

Applications: Task Scheduling, Precedence Constraints.

Transitive Closure

Warshall's Algorithm.

1.4.5 Graph Coloring

Vertex Coloring

Chromatic Number: $\chi(G)$

Bounds: Brooks' Theorem.

Edge Coloring

Chromatic Index: $\chi'(G)$

Vizing's Theorem.

Applications

Register Allocation in Compilers.

Scheduling Problems.

1.4.6 Planar Graphs

Definition and Examples

Euler's Formula: $V - E + F = 2$

Kuratowski's Theorem

Applications: Circuit Design, Geographic Maps.

1.4.7 Advanced Topics

Matchings and Covers

Maximum Matching.

König's Theorem for Bipartite Graphs.

Network Flows (Introduction)

Max-Flow Min-Cut Theorem (Statement).

Expander Graphs

Applications to Derandomization.

1.4.8 Exercises

Warmup Problems (25 problems)

Standard Problems (35 problems)

Challenging Problems (25 problems)

Research Problems ($\star$) (10 problems)

1.5 Algebraic Structures for Algorithm Design

Abstract algebra provides powerful tools for algorithm design, especially in cryptography, coding theory, and symbolic computation. This section introduces groups, rings, and fields with algorithmic applications in mind.

1.5.1 Groups

Definition and Examples

Axioms: Closure, Associativity, Identity, Inverses.

Examples: $(\mathbb{Z}, +)$**,** $(\mathbb{Z}_n, +_n)$**,** $(S_n, \circ)$

Subgroups

Definition and Lagrange's Theorem.

Cyclic Groups

Generators and Order.

Permutation Groups

Symmetric Group S_n**.**

Cayley's Theorem.

Group Homomorphisms

Kernels and Images.

1.5.2 Rings and Fields

Rings

Definition and Examples: $\mathbb{Z}$, $\mathbb{Z}_n$, Polynomial Rings.

Units and Zero Divisors.

Integral Domains.

Fields

Definition and Examples: $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{Z}_p$

Finite Fields (Galois Fields): $GF(p^n)$

Polynomial Arithmetic

Division Algorithm for Polynomials.

Greatest Common Divisor.

1.5.3 Applications to Cryptography

RSA and Group Theory

Diffie-Hellman Key Exchange

Elliptic Curve Cryptography (Overview)

1.5.4 Applications to Coding Theory

Linear Codes

Cyclic Codes and BCH Codes

1.5.5 Exercises

Warmup Problems

Standard Problems

Challenging Problems

Research Problems ($\star$)

Chapter 2

Mathematical Foundations II: Analysis and Probability

ALGORITHM ANALYSIS is fundamentally about asymptotic behavior: what happens as input size grows without bound? This requires tools from mathematical analysis—limits, series, asymptotic expansions. Meanwhile, randomized algorithms demand probability theory. This chapter develops both with the rigor and depth expected of an "Art of" treatment.

> **Chapter Organization:**
>
> This chapter covers four major areas:
>
> 1. **Mathematical Analysis** — Limits, series, asymptotics
> 2. **Probability Theory** — Measure-theoretic foundations to applications
> 3. **Information Theory** — Entropy, coding, lower bounds
> 4. **Number Theory** — Modular arithmetic, primality, factorization
>
> **What Sets This Apart:**
>
> - **Complete proofs** of major theorems (Stirling, Central Limit, Prime Number)
> - **Asymptotic methods** beyond Big-O: Euler-Maclaurin, saddle-point
> - **Rigorous probability** with measure theory foundations
> - **Algorithmic number theory** with complexity analysis
>
> **Prerequisites:** Single-variable calculus (derivatives, integrals, series). Basic probability at the level of a first course.

Part II

Foundations of Algorithmic Analysis

Chapter 3

Introduction to Algorithm Analysis

3.1 What Is Algorithm Analysis?

3.1.1 Correctness vs. Efficiency

3.1.2 Resource Measures: Time, Space, Energy, I/O

3.1.3 The Need for Mathematical Models

3.2 The RAM Model of Computation

3.2.1 Basic Operations and Unit-Cost Assumption

3.2.2 Memory Access Model

3.2.3 Limitations and Extensions of the RAM Model

3.3 Measuring Algorithm Performance

3.3.1 Input Size and Problem Instances

3.3.2 Counting Basic Operations

3.3.3 Exact vs. Asymptotic Analysis

3.4 Overview of Complexity Classes

3.4.1 P, NP, NP-Complete, and NP-Hard (Brief Introduction)

3.4.2 Why We Focus on Polynomial-Time Algorithms

Chapter 4

Asymptotic Notation

4.1 The Need for Asymptotic Analysis

4.1.1 Why Exact Counts Are Often Impractical

4.1.2 Growth Rates and Scalability

4.2 Big-O Notation (O)

4.2.1 Formal Definition

4.2.2 Intuition: Upper Bounds

4.2.3 Common Functions and Their Growth Rates

4.2.4 Examples and Non-Examples

4.2.5 Properties of Big-O

Transitivity

Addition and Multiplication Rules

Reflexivity and Asymmetry

4.3 Big-Omega Notation (Ω)

4.3.1 Formal Definition

4.3.2 Intuition: Lower Bounds

4.3.3 Examples and Applications

4.3.4 Relationship Between O and Ω

4.4 Big-Theta Notation (Θ)

4.4.1 Formal Definition

Chapter 5

Recurrence Relations and Their Solutions

5.1 Introduction to Recurrence Relations

5.1.1 What Are Recurrences?

5.1.2 Why They Arise in Algorithm Analysis

5.1.3 Examples from Divide-and-Conquer Algorithms

5.2 The Substitution Method

5.2.1 Guessing the Solution

5.2.2 Proving by Induction

5.2.3 Examples: Mergesort, Binary Search

5.2.4 Strengthening the Inductive Hypothesis

5.3 The Recursion-Tree Method

5.3.1 Visualizing the Recurrence

5.3.2 Summing Over Levels

5.3.3 Examples and Illustrations

5.3.4 Limitations and When to Use

5.4 The Master Theorem

5.4.1 Statement of the Master Theorem (Standard Form)

5.4.2 Three Cases and Their Intuition

Chapter 6

Best-Case, Worst-Case, and Average-Case Analysis

Chapter 7

Probabilistic Analysis of Algorithms

7.1 Foundations of Probabilistic Analysis

7.1.1 Random Variables in Algorithm Analysis

7.1.2 Indicator Random Variables

7.1.3 Linearity of Expectation

7.2 Expected Running Time

7.2.1 Formal Definition

7.2.2 Computing Expectations via Indicator Variables

7.2.3 Examples: Hiring Problem, Randomized Quicksort

7.3 Probabilistic Bounds

7.3.1 Markov's Inequality

7.3.2 Chebyshev's Inequality

7.3.3 Chernoff Bounds

7.3.4 Applications to Load Balancing and Hashing

7.4 Randomized Algorithms

7.4.1 Randomized Quicksort (Detailed Analysis)

7.4.2 Randomized Selection (Quickselect)

7.4.3 Hashing and Universal Hash Functions

7.4.4 Bloom Filters and Probabilistic Data Structures

Part III

Advanced Analysis Techniques

Chapter 8

Amortized Analysis

Chapter 9

Space Complexity Analysis

9.1 Introduction to Space Complexity

9.1.1 Why Space Matters

9.1.2 Types of Memory: Stack, Heap, Static

9.1.3 In-Place vs. Out-of-Place Algorithms

9.2 Measuring Space Usage

9.2.1 Auxiliary Space vs. Total Space

9.2.2 Recursive Call Stack Depth

9.2.3 Implicit vs. Explicit Data Structures

9.3 Examples of Space Complexity Analysis

9.3.1 Iterative Algorithms: Loops and Arrays

9.3.2 Recursive Algorithms: Mergesort, Quicksort

9.3.3 Dynamic Programming: Memoization vs. Tabulation

9.3.4 Graph Algorithms: BFS, DFS, Shortest Paths

9.4 Space-Time Tradeoffs

9.4.1 Caching and Memoization

9.4.2 Lookup Tables and Precomputation

9.4.3 Compression and Succinct Data Structures

9.5 Streaming and Online Algorithms

Chapter 10

Cache-Aware and I/O Complexity

10.1 Introduction to the Memory Hierarchy

10.1.1 Registers, Cache (L1, L2, L3), RAM, Disk

10.1.2 Latency and Bandwidth Characteristics

10.1.3 Why Algorithm Design Must Consider Memory

10.2 The External Memory Model (I/O Model)

10.2.1 Parameters: N (data size), M (memory size), B (block size)

10.2.2 I/O Complexity: Counting Block Transfers

10.2.3 Comparison with RAM Model

10.3 I/O-Efficient Algorithms

10.3.1 Scanning and Sorting

External Merge Sort

I/O Complexity: $O((N/B)\log_{M/B}(N/B))$

10.3.2 Matrix Operations

Matrix Transposition

Matrix Multiplication

10.3.3 Graph Algorithms

I/O-Efficient BFS and DFS

Minimum Spanning Tree

10.4 Cache-Oblivious Algorithms

Chapter 11

Cache-Aware Scheduling and Analysis for Multicores

11.1 Introduction to Multicore and Parallel Computing

11.1.1 Shared vs. Distributed Memory

11.1.2 Parallel Models: PRAM, Fork-Join, Work-Stealing

11.1.3 Performance Metrics: Work, Span, Parallelism

11.2 Cache Coherence and Consistency

11.2.1 MESI and MOESI Protocols

11.2.2 False Sharing in Multicore Systems

11.2.3 Impact on Algorithm Design

11.3 Cache-Aware Parallel Algorithms

11.3.1 Parallel Sorting with Cache Awareness

11.3.2 Parallel Matrix Multiplication (Strassen, Coppersmith-Winograd)

11.3.3 Load Balancing and Task Granularity

11.4 Real-Time and Embedded Systems

11.4.1 WCET Analysis in Cache-Aware Contexts

11.4.2 Predictability vs. Average-Case Performance

11.4.3 Cache Partitioning and Locking

Part IV

Lower Bounds and Optimality

Chapter 12

Lower Bounds for Comparison-Based Algorithms

Chapter 13

Algebraic and Non-Comparison Lower Bounds

13.1 Algebraic Decision Trees

13.1.1 Extending Beyond Comparisons

13.1.2 Element Distinctness Lower Bound

13.2 Communication Complexity

13.2.1 Models and Definitions

13.2.2 Applications to Data Structures

13.3 Cell-Probe Model

13.3.1 Lower Bounds for Data Structures

13.3.2 Dynamic vs. Static Data Structures

13.4 Exercises

Part V

Specialized Topics and Applications

Chapter 14

Analysis of Specific Algorithm Paradigms

14.1 Divide-and-Conquer Algorithms

14.1.1 General Framework and Recurrence Relations

14.1.2 Examples: Mergesort, Quicksort, Strassen's Algorithm

14.1.3 Optimality and Lower Bounds

14.2 Greedy Algorithms

14.2.1 Correctness via Exchange Arguments

14.2.2 Matroid Theory (Brief Introduction)

14.2.3 Examples: Huffman Coding, Kruskal's MST

14.3 Dynamic Programming

14.3.1 Optimal Substructure and Overlapping Subproblems

14.3.2 Memoization vs. Tabulation

14.3.3 Time and Space Complexity Analysis

14.3.4 Examples: Knapsack, Edit Distance, Matrix Chain Multiplication

14.4 Backtracking and Branch-and-Bound

14.4.1 Pruning the Search Space

14.4.2 Worst-Case Exponential, Average-Case Better

Chapter 15

Online Algorithms and Competitive Analysis

15.1 Introduction to Online Algorithms

15.1.1 Online vs. Offline Problems

15.1.2 Competitive Ratio

15.2 Examples of Online Problems

15.2.1 Paging and Caching (LRU, FIFO, LFU)

15.2.2 Load Balancing

15.2.3 Online Scheduling

15.3 Competitive Analysis Techniques

15.3.1 Deterministic vs. Randomized Algorithms

15.3.2 Lower Bounds via Adversary Arguments

15.4 Exercises

Chapter 16

Approximation Algorithms

16.1 Introduction to Approximation

16.1.1 NP-Hardness and Intractability

16.1.2 Approximation Ratios

16.2 Examples of Approximation Algorithms

16.2.1 Vertex Cover (2-Approximation)

16.2.2 Set Cover (Greedy, $\log n$-Approximation)

16.2.3 Traveling Salesman Problem (Metric TSP)

16.3 Analysis Techniques

16.3.1 Bounding Optimal Solutions

16.3.2 Linear Programming Relaxations

16.4 Exercises

Chapter 17

Parameterized Complexity

17.1 Introduction to Parameterized Algorithms

17.1.1 Fixed-Parameter Tractability (FPT)

17.1.2 Kernelization

17.2 Examples and Analysis

17.2.1 Vertex Cover Parameterized by Solution Size

17.2.2 Treewidth and Graph Algorithms

17.3 W-Hierarchy and Hardness

17.3.1 W[1], W[2], and Beyond

17.4 Exercises

Part VI

Practical Considerations and Case Studies

Chapter 18

From Theory to Practice

18.1 Hidden Constants and Lower-Order Terms

18.1.1 When $O(n \log n)$ Beats $O(n)$ in Practice

18.1.2 Empirical Performance Measurements

18.2 Algorithm Engineering

18.2.1 Profiling and Benchmarking

18.2.2 Tuning for Specific Hardware

18.2.3 Libraries and Implementations (STL, Boost, etc.)

18.3 Parallel and Distributed Algorithm Analysis

18.3.1 Scalability and Speedup

18.3.2 Amdahl's Law and Gustafson's Law

18.4 Energy Efficiency

18.4.1 Energy as a Resource

18.4.2 Green Computing and Mobile Devices

18.5 Exercises

Chapter 19

Case Studies

19.1 Sorting Algorithms in Practice

19.1.1 Timsort, Introsort, Radix Sort

19.1.2 Comparison of Theoretical vs. Empirical Performance

19.2 Graph Algorithms in Large-Scale Systems

19.2.1 Web Graphs and PageRank

19.2.2 Social Network Analysis

19.3 Machine Learning and Data Science

19.3.1 Complexity of Training Algorithms

19.3.2 SGD, AdaGrad, Adam: Time and Space Analysis

19.4 Database Systems

19.4.1 Query Optimization

19.4.2 Indexing Structures (B-Trees, LSM-Trees)

19.5 Exercises

Appendix A

Mathematical Background

A.1 Summation Formulas

A.2 Logarithms and Exponentials

A.3 Recurrence Relations (Quick Reference)

A.4 Probability Distributions

A.5 Matrix Operations

Appendix B

Pseudocode Conventions

B.1 Notation and Style

B.2 Common Data Structures

Appendix C

Solutions to Selected Exercises

Appendix D

Glossary of Terms

Appendix E

Index of Algorithms

Appendix F

Annotated Bibliography

F.1 Foundational Texts

F.2 Research Papers by Topic

F.3 Online Courses and Resources