

# Analysis

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 15, 2025*

© 2025 Mahdi

CREATIVE COMMONS • OPEN SOURCE

# 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

Copyright © 2025 Mahdi

This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.

License URL: <https://creativecommons.org/licenses/by-sa/4.0/>

**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:** L<sup>A</sup>T<sub>E</sub>X 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 15, 2025

For questions about licensing, contact: [bitsgenix@gmail.com](mailto:bitsgenix@gmail.com)

# Contents

<b>Title Page</b> .....	<b>i</b>
<b>Contents</b> .....	<b>iii</b>
<b>Acknowledgments</b> .....	<b>xvii</b>
<b>I Preface</b> .....	<b>1</b>
<b>1 Purpose and Scope of This Book</b> .....	<b>2</b>
1.1 What This Book Covers .....	2
1.2 What This Book Does Not Cover .....	2
1.3 Target Audience: Students, Researchers, and Practitioners .....	2
<b>2 Why “Precise Analysis” Matters — From Theory to Engineering</b> .....	<b>3</b>
2.1 The Gap Between Theoretical Complexity and Real-World Performance .....	3
2.2 Case Studies: When Big-O Isn’t Enough .....	3
2.3 The Role of Constants, Lower-Order Terms, and Hardware .....	3
<b>3 Mathematical and Algorithmic Prerequisites</b> .....	<b>4</b>
3.1 Discrete Mathematics .....	5
3.1.1 Sets, Functions, and Relations .....	5
3.1.2 Combinatorics: Permutations, Combinations, and Binomial Coefficients .....	5
3.1.3 Graph Theory Basics .....	5
3.1.4 Proof Techniques: Induction, Contradiction, and Contrapositive .....	5
3.2 Elementary Probability Theory .....	5
3.2.1 Sample Spaces, Events, and Probability Measures .....	5
3.2.2 Random Variables and Expectations .....	5
3.2.3 Basic Distributions: Uniform, Bernoulli, Geometric, Binomial .....	5
3.2.4 Linearity of Expectation .....	5
3.2.5 Conditional Probability and Independence .....	5
3.2.6 Variance and Standard Deviation .....	5
3.2.7 Moment Generating Functions (Brief Introduction) .....	5
3.3 Mathematical Analysis .....	5
3.3.1 Limits, Continuity, and Asymptotic Behavior .....	5
3.3.2 Sequences and Series .....	5

3.3.3	Summations and Closed Forms . . . . .	5
3.3.4	Integration and Differentiation (Brief Review) . . . . .	5
3.3.5	Taylor Series and Asymptotic Expansions . . . . .	5
3.3.6	Stirling's Approximation . . . . .	5
3.4	Linear Algebra (Brief Overview) . . . . .	5
3.4.1	Vectors, Matrices, and Linear Transformations . . . . .	5
3.4.2	Eigenvalues and Eigenvectors . . . . .	5
3.4.3	Applications to Markov Chains and Graph Algorithms . . . . .	5
3.4.4	Matrix Operations and Complexity . . . . .	5
3.5	Number Theory Essentials . . . . .	5
3.5.1	Divisibility and Modular Arithmetic . . . . .	5
3.5.2	Prime Numbers and Factorization . . . . .	5
3.5.3	Greatest Common Divisor and Euclidean Algorithm . . . . .	5
3.5.4	Applications to Cryptography and Hashing . . . . .	5
<b>4</b>	<b>Structure of the Book: Theorems, Proofs, Examples, and Exercises . . . . .</b>	<b>6</b>
4.1	How to Read This Book . . . . .	6
4.2	Notation and Conventions . . . . .	6
4.3	Types of Exercises: Conceptual, Computational, and Proof-Based . . . . .	6
4.4	Using Examples Effectively . . . . .	6
4.5	The Role of Rigor vs. Intuition . . . . .	6
<b>5</b>	<b>Primary References and Parallel Reading Guide . . . . .</b>	<b>7</b>
5.1	Classic Textbooks (CLRS, Sedgewick, Kleinberg-Tardos) . . . . .	7
5.2	Research Papers and Monographs . . . . .	7
5.3	Online Resources and Lecture Notes . . . . .	7
5.4	Recommended Reading Order and Study Plans . . . . .	7
<b>II</b>	<b>Foundations of Algorithmic Analysis . . . . .</b>	<b>8</b>
<b>6</b>	<b>Introduction to Algorithm Analysis . . . . .</b>	<b>9</b>
6.1	What Is Algorithm Analysis? . . . . .	9
6.1.1	Correctness vs. Efficiency . . . . .	9
6.1.2	Resource Measures: Time, Space, Energy, I/O . . . . .	9
6.1.3	The Need for Mathematical Models . . . . .	9

6.2	The RAM Model of Computation . . . . .	9
6.2.1	Basic Operations and Unit-Cost Assumption . . . . .	9
6.2.2	Memory Access Model . . . . .	9
6.2.3	Limitations and Extensions of the RAM Model . . . . .	9
6.3	Measuring Algorithm Performance . . . . .	9
6.3.1	Input Size and Problem Instances . . . . .	9
6.3.2	Counting Basic Operations . . . . .	9
6.3.3	Exact vs. Asymptotic Analysis . . . . .	9
6.4	Overview of Complexity Classes . . . . .	9
6.4.1	P, NP, NP-Complete, and NP-Hard (Brief Introduction) . . . . .	9
6.4.2	Why We Focus on Polynomial-Time Algorithms . . . . .	9
<b>7</b>	<b>Asymptotic Notation . . . . .</b>	<b>10</b>
7.1	The Need for Asymptotic Analysis . . . . .	11
7.1.1	Why Exact Counts Are Often Impractical . . . . .	11
7.1.2	Growth Rates and Scalability . . . . .	11
7.2	Big-O Notation ( $O$ ) . . . . .	11
7.2.1	Formal Definition . . . . .	11
7.2.2	Intuition: Upper Bounds . . . . .	11
7.2.3	Common Functions and Their Growth Rates . . . . .	11
7.2.4	Examples and Non-Examples . . . . .	11
7.2.5	Properties of Big-O . . . . .	11
7.3	Big-Omega Notation ( $\Omega$ ) . . . . .	11
7.3.1	Formal Definition . . . . .	11
7.3.2	Intuition: Lower Bounds . . . . .	11
7.3.3	Examples and Applications . . . . .	11
7.3.4	Relationship Between $O$ and $\Omega$ . . . . .	11
7.4	Big-Theta Notation ( $\Theta$ ) . . . . .	11
7.4.1	Formal Definition . . . . .	11
7.4.2	Intuition: Tight Bounds . . . . .	11
7.4.3	When to Use $\Theta$ vs. $O$ . . . . .	11
7.4.4	Examples of Tight Bounds . . . . .	11
7.5	Little-o and Little-omega Notation ( $o, \omega$ ) . . . . .	11
7.5.1	Formal Definitions . . . . .	11

7.5.2	Strict Asymptotic Bounds . . . . .	11
7.5.3	Applications in Analysis . . . . .	11
7.6	Common Misconceptions and Pitfalls . . . . .	11
7.6.1	Confusing $O$ with $\Theta$ . . . . .	11
7.6.2	Ignoring Constants in Practice . . . . .	11
7.6.3	Misapplying Asymptotic Notation to Small Inputs . . . . .	11
7.7	Comparing Functions . . . . .	11
7.7.1	L'Hôpital's Rule for Limits . . . . .	11
7.7.2	Logarithmic vs. Polynomial vs. Exponential Growth . . . . .	11
7.7.3	Hierarchy of Common Complexity Classes . . . . .	11
7.8	Exercises . . . . .	11
<b>8</b>	<b>Recurrence Relations and Their Solutions . . . . .</b>	<b>12</b>
8.1	Introduction to Recurrence Relations . . . . .	13
8.1.1	What Are Recurrences? . . . . .	13
8.1.2	Why They Arise in Algorithm Analysis . . . . .	13
8.1.3	Examples from Divide-and-Conquer Algorithms . . . . .	13
8.2	The Substitution Method . . . . .	13
8.2.1	Guessing the Solution . . . . .	13
8.2.2	Proving by Induction . . . . .	13
8.2.3	Examples: Mergesort, Binary Search . . . . .	13
8.2.4	Strengthening the Inductive Hypothesis . . . . .	13
8.3	The Recursion-Tree Method . . . . .	13
8.3.1	Visualizing the Recurrence . . . . .	13
8.3.2	Summing Over Levels . . . . .	13
8.3.3	Examples and Illustrations . . . . .	13
8.3.4	Limitations and When to Use . . . . .	13
8.4	The Master Theorem . . . . .	13
8.4.1	Statement of the Master Theorem (Standard Form) . . . . .	13
8.4.2	Three Cases and Their Intuition . . . . .	13
8.4.3	Proof Sketch (Via Recursion Trees) . . . . .	13
8.4.4	Examples: $T(n) = aT(n/b) + f(n)$ . . . . .	13
8.4.5	Regularity Condition and Edge Cases . . . . .	13
8.4.6	Extended Master Theorem (Akra-Bazzi) . . . . .	13

8.5	The Akra-Bazzi Method . . . . .	13
8.5.1	Motivation: Unequal Subproblem Sizes . . . . .	13
8.5.2	Statement and Conditions . . . . .	13
8.5.3	Examples and Applications . . . . .	13
8.5.4	Proof Overview (Advanced) . . . . .	13
8.6	Linear Recurrences with Constant Coefficients . . . . .	13
8.6.1	Homogeneous Linear Recurrences . . . . .	13
8.6.2	Characteristic Equations . . . . .	13
8.6.3	Solving Fibonacci-Type Recurrences . . . . .	13
8.6.4	Non-Homogeneous Recurrences and Particular Solutions . . . . .	13
8.7	Generating Functions . . . . .	13
8.7.1	Introduction to Generating Functions . . . . .	13
8.7.2	Solving Recurrences with Generating Functions . . . . .	13
8.7.3	Examples: Catalan Numbers, Stirling Numbers . . . . .	13
8.8	Advanced Topics . . . . .	13
8.8.1	Full History Recurrences . . . . .	13
8.8.2	Recurrences with Variable Coefficients . . . . .	13
8.8.3	Probabilistic Recurrences (Preview) . . . . .	13
8.9	Exercises . . . . .	13
<b>9</b>	<b>Best-Case, Worst-Case, and Average-Case Analysis . . . . .</b>	<b>14</b>
9.1	Defining Input Classes . . . . .	15
9.1.1	What Constitutes an “Input”? . . . .	15
9.1.2	Problem Instances and Instance Distributions . . . . .	15
9.2	Best-Case Analysis . . . . .	15
9.2.1	Definition and Purpose . . . . .	15
9.2.2	Examples: Insertion Sort, Linear Search . . . . .	15
9.2.3	When Best-Case Matters (and When It Doesn’t) . . . . .	15
9.3	Worst-Case Analysis . . . . .	15
9.3.1	Definition and Motivation . . . . .	15
9.3.2	Guarantees and Robustness . . . . .	15
9.3.3	Examples: Quicksort, Searching in Unsorted Arrays . . . . .	15
9.3.4	Lower Bounds and Optimality . . . . .	15
9.4	Average-Case Analysis . . . . .	15



9.4.1	Definition: Expected Running Time . . . . .	15
9.4.2	Assumptions About Input Distributions . . . . .	15
9.4.3	Probabilistic Models: Uniform, Gaussian, etc. . . . .	15
9.4.4	Examples: Quicksort, Hashing, Skip Lists . . . . .	15
9.5	Probabilistic Analysis vs. Randomized Algorithms . . . . .	15
9.5.1	Distinction Between the Two Concepts . . . . .	15
9.5.2	Randomized Quicksort: Expected $O(n \log n)$ . . . . .	15
9.5.3	Las Vegas vs. Monte Carlo Algorithms . . . . .	15
9.6	Smoothed Analysis . . . . .	15
9.6.1	Motivation: Beyond Worst-Case Pessimism . . . . .	15
9.6.2	Introduction to Smoothed Analysis . . . . .	15
9.6.3	Case Study: Simplex Algorithm . . . . .	15
9.7	Exercises . . . . .	15
<b>10</b>	<b>Probabilistic Analysis of Algorithms . . . . .</b>	<b>16</b>
10.1	Foundations of Probabilistic Analysis . . . . .	17
10.1.1	Random Variables in Algorithm Analysis . . . . .	17
10.1.2	Indicator Random Variables . . . . .	17
10.1.3	Linearity of Expectation . . . . .	17
10.2	Expected Running Time . . . . .	17
10.2.1	Formal Definition . . . . .	17
10.2.2	Computing Expectations via Indicator Variables . . . . .	17
10.2.3	Examples: Hiring Problem, Randomized Quicksort . . . . .	17
10.3	Probabilistic Bounds . . . . .	17
10.3.1	Markov's Inequality . . . . .	17
10.3.2	Chebyshev's Inequality . . . . .	17
10.3.3	Chernoff Bounds . . . . .	17
10.3.4	Applications to Load Balancing and Hashing . . . . .	17
10.4	Randomized Algorithms . . . . .	17
10.4.1	Randomized Quicksort (Detailed Analysis) . . . . .	17
10.4.2	Randomized Selection (Quickselect) . . . . .	17
10.4.3	Hashing and Universal Hash Functions . . . . .	17
10.4.4	Bloom Filters and Probabilistic Data Structures . . . . .	17
10.5	Analysis of Randomized Data Structures . . . . .	17

10.5.1	Skip Lists . . . . .	17
10.5.2	Treaps . . . . .	17
10.5.3	Hash Tables with Chaining and Open Addressing . . . . .	17
10.6	High-Probability Results . . . . .	17
10.6.1	What Does “With High Probability” Mean? . . . . .	17
10.6.2	Concentration Inequalities . . . . .	17
10.6.3	Union Bound and Probabilistic Method . . . . .	17
10.7	Exercises . . . . .	17
<b>III</b>	<b>Advanced Analysis Techniques . . . . .</b>	<b>18</b>
<b>11</b>	<b>Amortized Analysis . . . . .</b>	<b>19</b>
11.1	Introduction to Amortized Analysis . . . . .	20
11.1.1	Motivation: Why Average Per-Operation Cost? . . . . .	20
11.1.2	Amortized vs. Average-Case Analysis . . . . .	20
11.1.3	When to Use Amortized Analysis . . . . .	20
11.2	Aggregate Analysis . . . . .	20
11.2.1	Definition and Methodology . . . . .	20
11.2.2	Example: Dynamic Array (Vector) Resizing . . . . .	20
11.2.3	Example: Binary Counter Increment . . . . .	20
11.2.4	Example: Stack with Multipop . . . . .	20
11.3	The Accounting Method . . . . .	20
11.3.1	Conceptual Framework: Credits and Debits . . . . .	20
11.3.2	Defining Amortized Costs . . . . .	20
11.3.3	Example: Dynamic Array via Accounting . . . . .	20
11.3.4	Example: Splay Trees (Introduction) . . . . .	20
11.3.5	Ensuring Non-Negative Credit Balance . . . . .	20
11.4	The Potential Method . . . . .	20
11.4.1	Potential Functions: Definition and Intuition . . . . .	20
11.4.2	Relating Amortized Cost to Actual Cost . . . . .	20
11.4.3	Designing Good Potential Functions . . . . .	20
11.4.4	Example: Dynamic Array via Potential Method . . . . .	20
11.4.5	Example: Binary Counter via Potential Method . . . . .	20
11.4.6	Example: Fibonacci Heaps (Overview) . . . . .	20
11.5	Comparing the Three Methods . . . . .	20

11.5.1	Strengths and Weaknesses . . . . .	20
11.5.2	When to Choose Which Method . . . . .	20
11.5.3	Equivalence of Methods (Informal Discussion) . . . . .	20
11.6	Advanced Applications . . . . .	20
11.6.1	Splay Trees: Full Analysis . . . . .	20
11.6.2	Fibonacci Heaps . . . . .	20
11.6.3	Disjoint-Set Union (Union-Find) . . . . .	20
11.7	Exercises . . . . .	20
<b>12</b>	<b>Space Complexity Analysis . . . . .</b>	<b>21</b>
12.1	Introduction to Space Complexity . . . . .	22
12.1.1	Why Space Matters . . . . .	22
12.1.2	Types of Memory: Stack, Heap, Static . . . . .	22
12.1.3	In-Place vs. Out-of-Place Algorithms . . . . .	22
12.2	Measuring Space Usage . . . . .	22
12.2.1	Auxiliary Space vs. Total Space . . . . .	22
12.2.2	Recursive Call Stack Depth . . . . .	22
12.2.3	Implicit vs. Explicit Data Structures . . . . .	22
12.3	Examples of Space Complexity Analysis . . . . .	22
12.3.1	Iterative Algorithms: Loops and Arrays . . . . .	22
12.3.2	Recursive Algorithms: Mergesort, Quicksort . . . . .	22
12.3.3	Dynamic Programming: Memoization vs. Tabulation . . . . .	22
12.3.4	Graph Algorithms: BFS, DFS, Shortest Paths . . . . .	22
12.4	Space-Time Tradeoffs . . . . .	22
12.4.1	Caching and Memoization . . . . .	22
12.4.2	Lookup Tables and Precomputation . . . . .	22
12.4.3	Compression and Succinct Data Structures . . . . .	22
12.5	Streaming and Online Algorithms . . . . .	22
12.5.1	Sublinear Space Algorithms . . . . .	22
12.5.2	Sketching and Sampling Techniques . . . . .	22
12.5.3	Examples: Distinct Elements, Heavy Hitters . . . . .	22
12.6	Space Complexity Classes . . . . .	22
12.6.1	L, NL, PSPACE (Brief Overview) . . . . .	22
12.6.2	Savitch's Theorem . . . . .	22

12.7 Exercises . . . . .	22
<b>13 Cache-Aware and I/O Complexity . . . . .</b>	<b>23</b>
13.1 Introduction to the Memory Hierarchy . . . . .	24
13.1.1 Registers, Cache (L1, L2, L3), RAM, Disk . . . . .	24
13.1.2 Latency and Bandwidth Characteristics . . . . .	24
13.1.3 Why Algorithm Design Must Consider Memory . . . . .	24
13.2 The External Memory Model (I/O Model) . . . . .	24
13.2.1 Parameters: $N$ (data size), $M$ (memory size), $B$ (block size) . . . . .	24
13.2.2 I/O Complexity: Counting Block Transfers . . . . .	24
13.2.3 Comparison with RAM Model . . . . .	24
13.3 I/O-Efficient Algorithms . . . . .	24
13.3.1 Scanning and Sorting . . . . .	24
13.3.2 Matrix Operations . . . . .	24
13.3.3 Graph Algorithms . . . . .	24
13.4 Cache-Oblivious Algorithms . . . . .	24
13.4.1 Motivation: Optimal Without Knowing $M$ and $B$ . . . . .	24
13.4.2 Cache-Oblivious Sorting (Funnelsort) . . . . .	24
13.4.3 Cache-Oblivious Matrix Multiplication . . . . .	24
13.4.4 Cache-Oblivious B-Trees (van Emde Boas Layout) . . . . .	24
13.5 Cache-Aware Analysis . . . . .	24
13.5.1 Modeling Cache Behavior . . . . .	24
13.5.2 Locality of Reference: Temporal and Spatial . . . . .	24
13.5.3 Blocking and Tiling Techniques . . . . .	24
13.6 Real-World Considerations . . . . .	24
13.6.1 Multi-Level Caches . . . . .	24
13.6.2 Cache Replacement Policies (LRU, LFU, etc.) . . . . .	24
13.6.3 Prefetching and Speculative Execution . . . . .	24
13.6.4 False Sharing and Cache Line Effects . . . . .	24
13.7 Case Studies . . . . .	24
13.7.1 Database Query Processing . . . . .	24
13.7.2 External Memory Sorting in Practice . . . . .	24
13.7.3 Scientific Computing and Large-Scale Simulations . . . . .	24
13.8 Exercises . . . . .	24

<b>14 Cache-Aware Scheduling and Analysis for Multicores . . . . .</b>	<b>25</b>
14.1 Introduction to Multicore and Parallel Computing . . . . .	26
14.1.1 Shared vs. Distributed Memory . . . . .	26
14.1.2 Parallel Models: PRAM, Fork-Join, Work-Stealing . . . . .	26
14.1.3 Performance Metrics: Work, Span, Parallelism . . . . .	26
14.2 Cache Coherence and Consistency . . . . .	26
14.2.1 MESI and MOESI Protocols . . . . .	26
14.2.2 False Sharing in Multicore Systems . . . . .	26
14.2.3 Impact on Algorithm Design . . . . .	26
14.3 Cache-Aware Parallel Algorithms . . . . .	26
14.3.1 Parallel Sorting with Cache Awareness . . . . .	26
14.3.2 Parallel Matrix Multiplication (Strassen, Coppersmith-Winograd) . . . . .	26
14.3.3 Load Balancing and Task Granularity . . . . .	26
14.4 Real-Time and Embedded Systems . . . . .	26
14.4.1 WCET Analysis in Cache-Aware Contexts . . . . .	26
14.4.2 Predictability vs. Average-Case Performance . . . . .	26
14.4.3 Cache Partitioning and Locking . . . . .	26
14.5 Scheduling Strategies . . . . .	26
14.5.1 Static vs. Dynamic Scheduling . . . . .	26
14.5.2 Work-Stealing Algorithms . . . . .	26
14.5.3 Affinity Scheduling for Cache Locality . . . . .	26
14.6 Analysis Techniques . . . . .	26
14.6.1 DAG-Based Analysis (Cilk Model) . . . . .	26
14.6.2 Brent's Theorem and Greedy Scheduling . . . . .	26
14.6.3 Cache Miss Analysis in Parallel Programs . . . . .	26
14.7 Case Studies from Research . . . . .	26
14.7.1 ECRTS 2007: Cache-Aware Real-Time Scheduling . . . . .	26
14.7.2 Cache-Aware Scheduling for Multicores (Embedded Systems) . . . . .	26
14.7.3 VLDB 2019: Concurrent Hash Tables and Cache Performance . . . . .	26
14.8 Exercises . . . . .	26
<b>IV Lower Bounds and Optimality . . . . .</b>	<b>27</b>
<b>15 Lower Bounds for Comparison-Based Algorithms . . . . .</b>	<b>28</b>
15.1 Decision Trees . . . . .	29

15.1.1	Modeling Algorithms as Decision Trees . . . . .	29
15.1.2	Height of Decision Trees and Worst-Case Complexity . . . . .	29
15.2	Sorting Lower Bound . . . . .	29
15.2.1	Information-Theoretic Argument . . . . .	29
15.2.2	$\Omega(n \log n)$ Lower Bound for Comparison Sorting . . . . .	29
15.2.3	Implications and Optimal Algorithms . . . . .	29
15.3	Selection and Searching Lower Bounds . . . . .	29
15.3.1	Finding the Minimum: $\Omega(n)$ . . . . .	29
15.3.2	Finding Median: Adversary Arguments . . . . .	29
15.3.3	Searching in Sorted Arrays: $\Omega(\log n)$ . . . . .	29
15.4	Adversary Arguments . . . . .	29
15.4.1	General Framework . . . . .	29
15.4.2	Examples: Merging, Element Uniqueness . . . . .	29
15.5	Exercises . . . . .	29
<b>16</b>	<b>Algebraic and Non-Comparison Lower Bounds . . . . .</b>	<b>30</b>
16.1	Algebraic Decision Trees . . . . .	30
16.1.1	Extending Beyond Comparisons . . . . .	30
16.1.2	Element Distinctness Lower Bound . . . . .	30
16.2	Communication Complexity . . . . .	30
16.2.1	Models and Definitions . . . . .	30
16.2.2	Applications to Data Structures . . . . .	30
16.3	Cell-Probe Model . . . . .	30
16.3.1	Lower Bounds for Data Structures . . . . .	30
16.3.2	Dynamic vs. Static Data Structures . . . . .	30
16.4	Exercises . . . . .	30
<b>V</b>	<b>Specialized Topics and Applications . . . . .</b>	<b>31</b>
<b>17</b>	<b>Analysis of Specific Algorithm Paradigms . . . . .</b>	<b>32</b>
17.1	Divide-and-Conquer Algorithms . . . . .	33
17.1.1	General Framework and Recurrence Relations . . . . .	33
17.1.2	Examples: Mergesort, Quicksort, Strassen's Algorithm . . . . .	33
17.1.3	Optimality and Lower Bounds . . . . .	33
17.2	Greedy Algorithms . . . . .	33

17.2.1	Correctness via Exchange Arguments . . . . .	33
17.2.2	Matroid Theory (Brief Introduction) . . . . .	33
17.2.3	Examples: Huffman Coding, Kruskal's MST . . . . .	33
17.3	Dynamic Programming . . . . .	33
17.3.1	Optimal Substructure and Overlapping Subproblems . . . . .	33
17.3.2	Memoization vs. Tabulation . . . . .	33
17.3.3	Time and Space Complexity Analysis . . . . .	33
17.3.4	Examples: Knapsack, Edit Distance, Matrix Chain Multiplication . . . . .	33
17.4	Backtracking and Branch-and-Bound . . . . .	33
17.4.1	Pruning the Search Space . . . . .	33
17.4.2	Worst-Case Exponential, Average-Case Better . . . . .	33
17.4.3	Examples: N-Queens, Traveling Salesman . . . . .	33
17.5	Exercises . . . . .	33
<b>18</b>	<b>Online Algorithms and Competitive Analysis . . . . .</b>	<b>34</b>
18.1	Introduction to Online Algorithms . . . . .	34
18.1.1	Online vs. Offline Problems . . . . .	34
18.1.2	Competitive Ratio . . . . .	34
18.2	Examples of Online Problems . . . . .	34
18.2.1	Paging and Caching (LRU, FIFO, LFU) . . . . .	34
18.2.2	Load Balancing . . . . .	34
18.2.3	Online Scheduling . . . . .	34
18.3	Competitive Analysis Techniques . . . . .	34
18.3.1	Deterministic vs. Randomized Algorithms . . . . .	34
18.3.2	Lower Bounds via Adversary Arguments . . . . .	34
18.4	Exercises . . . . .	34
<b>19</b>	<b>Approximation Algorithms . . . . .</b>	<b>35</b>
19.1	Introduction to Approximation . . . . .	35
19.1.1	NP-Hardness and Intractability . . . . .	35
19.1.2	Approximation Ratios . . . . .	35
19.2	Examples of Approximation Algorithms . . . . .	35
19.2.1	Vertex Cover (2-Approximation) . . . . .	35
19.2.2	Set Cover (Greedy, $\log n$ -Approximation) . . . . .	35
19.2.3	Traveling Salesman Problem (Metric TSP) . . . . .	35

19.3 Analysis Techniques . . . . .	35
19.3.1 Bounding Optimal Solutions . . . . .	35
19.3.2 Linear Programming Relaxations . . . . .	35
19.4 Exercises . . . . .	35
<b>20 Parameterized Complexity . . . . .</b>	<b>36</b>
20.1 Introduction to Parameterized Algorithms . . . . .	36
20.1.1 Fixed-Parameter Tractability (FPT) . . . . .	36
20.1.2 Kernelization . . . . .	36
20.2 Examples and Analysis . . . . .	36
20.2.1 Vertex Cover Parameterized by Solution Size . . . . .	36
20.2.2 Treewidth and Graph Algorithms . . . . .	36
20.3 W-Hierarchy and Hardness . . . . .	36
20.3.1 $W[1]$ , $W[2]$ , and Beyond . . . . .	36
20.4 Exercises . . . . .	36
<b>VI Practical Considerations and Case Studies . . . . .</b>	<b>37</b>
<b>21 From Theory to Practice . . . . .</b>	<b>38</b>
21.1 Hidden Constants and Lower-Order Terms . . . . .	38
21.1.1 When $O(n \log n)$ Beats $O(n)$ in Practice . . . . .	38
21.1.2 Empirical Performance Measurements . . . . .	38
21.2 Algorithm Engineering . . . . .	38
21.2.1 Profiling and Benchmarking . . . . .	38
21.2.2 Tuning for Specific Hardware . . . . .	38
21.2.3 Libraries and Implementations (STL, Boost, etc.) . . . . .	38
21.3 Parallel and Distributed Algorithm Analysis . . . . .	38
21.3.1 Scalability and Speedup . . . . .	38
21.3.2 Amdahl's Law and Gustafson's Law . . . . .	38
21.4 Energy Efficiency . . . . .	38
21.4.1 Energy as a Resource . . . . .	38
21.4.2 Green Computing and Mobile Devices . . . . .	38
21.5 Exercises . . . . .	38
<b>22 Case Studies . . . . .</b>	<b>39</b>
22.1 Sorting Algorithms in Practice . . . . .	39



22.1.1	Timsort, Introsort, Radix Sort . . . . .	39
22.1.2	Comparison of Theoretical vs. Empirical Performance . . . . .	39
22.2	Graph Algorithms in Large-Scale Systems . . . . .	39
22.2.1	Web Graphs and PageRank . . . . .	39
22.2.2	Social Network Analysis . . . . .	39
22.3	Machine Learning and Data Science . . . . .	39
22.3.1	Complexity of Training Algorithms . . . . .	39
22.3.2	SGD, AdaGrad, Adam: Time and Space Analysis . . . . .	39
22.4	Database Systems . . . . .	39
22.4.1	Query Optimization . . . . .	39
22.4.2	Indexing Structures (B-Trees, LSM-Trees) . . . . .	39
22.5	Exercises . . . . .	39
<b>A</b>	<b>Mathematical Background . . . . .</b>	<b>40</b>
A.1	Summation Formulas . . . . .	40
A.2	Logarithms and Exponentials . . . . .	40
A.3	Recurrence Relations (Quick Reference) . . . . .	40
A.4	Probability Distributions . . . . .	40
A.5	Matrix Operations . . . . .	40
<b>B</b>	<b>Pseudocode Conventions . . . . .</b>	<b>41</b>
B.1	Notation and Style . . . . .	41
B.2	Common Data Structures . . . . .	41
<b>C</b>	<b>Solutions to Selected Exercises . . . . .</b>	<b>42</b>
<b>D</b>	<b>Glossary of Terms . . . . .</b>	<b>43</b>
<b>E</b>	<b>Index of Algorithms . . . . .</b>	<b>44</b>
<b>F</b>	<b>Annotated Bibliography . . . . .</b>	<b>45</b>
F.1	Foundational Texts . . . . .	45
F.2	Research Papers by Topic . . . . .	45
F.3	Online Courses and Resources . . . . .	45

# 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**

## **Preface**

# **Chapter 1**

## **Purpose and Scope of This Book**

### **1.1 What This Book Covers**

### **1.2 What This Book Does Not Cover**

### **1.3 Target Audience: Students, Researchers, and Practitioners**

## **Chapter 2**

# **Why “Precise Analysis” Matters — From Theory to Engineering**

- 2.1 The Gap Between Theoretical Complexity and Real-World Performance**
- 2.2 Case Studies: When Big-O Isn't Enough**
- 2.3 The Role of Constants, Lower-Order Terms, and Hardware**



# Chapter 3

## Mathematical and Algorithmic Prerequisites

### 3.1 Discrete Mathematics

#### 3.1.1 Sets, Functions, and Relations

#### 3.1.2 Combinatorics: Permutations, Combinations, and Binomial Coefficients

#### 3.1.3 Graph Theory Basics

#### 3.1.4 Proof Techniques: Induction, Contradiction, and Contrapositive

### 3.2 Elementary Probability Theory

#### 3.2.1 Sample Spaces, Events, and Probability Measures

#### 3.2.2 Random Variables and Expectations

#### 3.2.3 Basic Distributions: Uniform, Bernoulli, Geometric, Binomial

#### 3.2.4 Linearity of Expectation

#### 3.2.5 Conditional Probability and Independence

#### 3.2.6 Variance and Standard Deviation

#### 3.2.7 Moment Generating Functions (Brief Introduction)

---

### 3.3 Mathematical Analysis

#### 3.3.1 Limits, Continuity, and Asymptotic Behavior

# **Chapter 4**

## **Structure of the Book: Theorems, Proofs, Examples, and Exercises**

**4.1 How to Read This Book**

**4.2 Notation and Conventions**

**4.3 Types of Exercises: Conceptual, Computational, and Proof-Based**

**4.4 Using Examples Effectively**

**4.5 The Role of Rigor vs. Intuition**



# **Chapter 5**

## **Primary References and Parallel Reading Guide**

- 5.1 Classic Textbooks (CLRS, Sedgewick, Kleinberg-Tardos)**
- 5.2 Research Papers and Monographs**
- 5.3 Online Resources and Lecture Notes**
- 5.4 Recommended Reading Order and Study Plans**

## **Part II**

# **Foundations of Algorithmic Analysis**

# Chapter 6

## Introduction to Algorithm Analysis

### 6.1 What Is Algorithm Analysis?

#### 6.1.1 Correctness vs. Efficiency

#### 6.1.2 Resource Measures: Time, Space, Energy, I/O

#### 6.1.3 The Need for Mathematical Models

### 6.2 The RAM Model of Computation

#### 6.2.1 Basic Operations and Unit-Cost Assumption

#### 6.2.2 Memory Access Model

#### 6.2.3 Limitations and Extensions of the RAM Model

### 6.3 Measuring Algorithm Performance

#### 6.3.1 Input Size and Problem Instances

#### 6.3.2 Counting Basic Operations

#### 6.3.3 Exact vs. Asymptotic Analysis

### 6.4 Overview of Complexity Classes

#### 6.4.1 P, NP, NP-Complete, and NP-Hard (Brief Introduction)

#### 6.4.2 Why We Focus on Polynomial-Time Algorithms



# Chapter 7

## Asymptotic Notation

### 7.1 The Need for Asymptotic Analysis

#### 7.1.1 Why Exact Counts Are Often Impractical

#### 7.1.2 Growth Rates and Scalability

### 7.2 Big-O Notation ( $O$ )

#### 7.2.1 Formal Definition

#### 7.2.2 Intuition: Upper Bounds

#### 7.2.3 Common Functions and Their Growth Rates

#### 7.2.4 Examples and Non-Examples

#### 7.2.5 Properties of Big-O

Transitivity

Addition and Multiplication Rules

Reflexivity and Asymmetry

### 7.3 Big-Omega Notation ( $\Omega$ )

#### 7.3.1 Formal Definition

#### 7.3.2 Intuition: Lower Bounds

#### 7.3.3 Examples and Applications

#### 7.3.4 Relationship Between $O$ and $\Omega$

First Edition • 2025

### 7.4 Big-Theta Notation ( $\Theta$ )

#### 7.4.1 Formal Definition



# Chapter 8

## Recurrence Relations and Their Solutions

### 8.1 Introduction to Recurrence Relations

#### 8.1.1 What Are Recurrences?

#### 8.1.2 Why They Arise in Algorithm Analysis

#### 8.1.3 Examples from Divide-and-Conquer Algorithms

### 8.2 The Substitution Method

#### 8.2.1 Guessing the Solution

#### 8.2.2 Proving by Induction

#### 8.2.3 Examples: Mergesort, Binary Search

#### 8.2.4 Strengthening the Inductive Hypothesis

### 8.3 The Recursion-Tree Method

#### 8.3.1 Visualizing the Recurrence

#### 8.3.2 Summing Over Levels

#### 8.3.3 Examples and Illustrations

#### 8.3.4 Limitations and When to Use

### 8.4 The Master Theorem

#### 8.4.1 Statement of the Master Theorem (Standard Form)

#### 8.4.2 Three Cases and Their Intuition





# Chapter 9

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

### 9.1 Defining Input Classes

#### 9.1.1 What Constitutes an “Input”?

#### 9.1.2 Problem Instances and Instance Distributions

### 9.2 Best-Case Analysis

#### 9.2.1 Definition and Purpose

#### 9.2.2 Examples: Insertion Sort, Linear Search

#### 9.2.3 When Best-Case Matters (and When It Doesn’t)

### 9.3 Worst-Case Analysis

#### 9.3.1 Definition and Motivation

#### 9.3.2 Guarantees and Robustness

#### 9.3.3 Examples: Quicksort, Searching in Unsorted Arrays

#### 9.3.4 Lower Bounds and Optimality

### 9.4 Average-Case Analysis

#### 9.4.1 Definition: Expected Running Time

#### 9.4.2 Assumptions About Input Distributions

#### 9.4.3 Probabilistic Models: Uniform, Gaussian, etc.

#### 9.4.4 Examples: Quicksort, Hashing, Skip Lists



# Chapter 10

## Probabilistic Analysis of Algorithms

### 10.1 Foundations of Probabilistic Analysis

#### 10.1.1 Random Variables in Algorithm Analysis

#### 10.1.2 Indicator Random Variables

#### 10.1.3 Linearity of Expectation

### 10.2 Expected Running Time

#### 10.2.1 Formal Definition

#### 10.2.2 Computing Expectations via Indicator Variables

#### 10.2.3 Examples: Hiring Problem, Randomized Quicksort

### 10.3 Probabilistic Bounds

#### 10.3.1 Markov's Inequality

#### 10.3.2 Chebyshev's Inequality

#### 10.3.3 Chernoff Bounds

#### 10.3.4 Applications to Load Balancing and Hashing

### 10.4 Randomized Algorithms

#### 10.4.1 Randomized Quicksort (Detailed Analysis)

#### 10.4.2 Randomized Selection (Quickselect)

#### 10.4.3 Hashing and Universal Hash Functions

#### 10.4.4 Bloom Filters and Probabilistic Data Structures

## **Part III**

# **Advanced Analysis Techniques**



# Chapter 11

## Amortized Analysis

### 11.1 Introduction to Amortized Analysis

#### 11.1.1 Motivation: Why Average Per-Operation Cost?

#### 11.1.2 Amortized vs. Average-Case Analysis

#### 11.1.3 When to Use Amortized Analysis

### 11.2 Aggregate Analysis

#### 11.2.1 Definition and Methodology

#### 11.2.2 Example: Dynamic Array (Vector) Resizing

#### 11.2.3 Example: Binary Counter Increment

#### 11.2.4 Example: Stack with Multipop

### 11.3 The Accounting Method

#### 11.3.1 Conceptual Framework: Credits and Debits

#### 11.3.2 Defining Amortized Costs

#### 11.3.3 Example: Dynamic Array via Accounting

#### 11.3.4 Example: Splay Trees (Introduction)

#### 11.3.5 Ensuring Non-Negative Credit Balance

### 11.4 The Potential Method

#### 11.4.1 Potential Functions: Definition and Intuition

#### 11.4.2 Relating Amortized Cost to Actual Cost



# Chapter 12

## Space Complexity Analysis

### 12.1 Introduction to Space Complexity

#### 12.1.1 Why Space Matters

#### 12.1.2 Types of Memory: Stack, Heap, Static

#### 12.1.3 In-Place vs. Out-of-Place Algorithms

### 12.2 Measuring Space Usage

#### 12.2.1 Auxiliary Space vs. Total Space

#### 12.2.2 Recursive Call Stack Depth

#### 12.2.3 Implicit vs. Explicit Data Structures

### 12.3 Examples of Space Complexity Analysis

#### 12.3.1 Iterative Algorithms: Loops and Arrays

#### 12.3.2 Recursive Algorithms: Mergesort, Quicksort

#### 12.3.3 Dynamic Programming: Memoization vs. Tabulation

#### 12.3.4 Graph Algorithms: BFS, DFS, Shortest Paths

### 12.4 Space-Time Tradeoffs

#### 12.4.1 Caching and Memoization

#### 12.4.2 Lookup Tables and Precomputation

#### 12.4.3 Compression and Succinct Data Structures

### 12.5 Streaming and Online Algorithms





# Chapter 13

## Cache-Aware and I/O Complexity

### 13.1 Introduction to the Memory Hierarchy

#### 13.1.1 Registers, Cache (L1, L2, L3), RAM, Disk

#### 13.1.2 Latency and Bandwidth Characteristics

#### 13.1.3 Why Algorithm Design Must Consider Memory

### 13.2 The External Memory Model (I/O Model)

#### 13.2.1 Parameters: $N$ (data size), $M$ (memory size), $B$ (block size)

#### 13.2.2 I/O Complexity: Counting Block Transfers

#### 13.2.3 Comparison with RAM Model

### 13.3 I/O-Efficient Algorithms

#### 13.3.1 Scanning and Sorting

External Merge Sort

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

#### 13.3.2 Matrix Operations

Matrix Transposition

Matrix Multiplication

#### 13.3.3 Graph Algorithms

I/O-Efficient BFS and DFS

Minimum Spanning Tree

### 13.4 Cache-Oblivious Algorithms



# Chapter 14

## Cache-Aware Scheduling and Analysis for Multicores

### 14.1 Introduction to Multicore and Parallel Computing

#### 14.1.1 Shared vs. Distributed Memory

#### 14.1.2 Parallel Models: PRAM, Fork-Join, Work-Stealing

#### 14.1.3 Performance Metrics: Work, Span, Parallelism

### 14.2 Cache Coherence and Consistency

#### 14.2.1 MESI and MOESI Protocols

#### 14.2.2 False Sharing in Multicore Systems

#### 14.2.3 Impact on Algorithm Design

### 14.3 Cache-Aware Parallel Algorithms

#### 14.3.1 Parallel Sorting with Cache Awareness

#### 14.3.2 Parallel Matrix Multiplication (Strassen, Coppersmith-Winograd)

#### 14.3.3 Load Balancing and Task Granularity

### 14.4 Real-Time and Embedded Systems

#### 14.4.1 WCET Analysis in Cache-Aware Contexts

#### 14.4.2 Predictability vs. Average-Case Performance

#### 14.4.3 Cache Partitioning and Locking

## **Part IV**

# **Lower Bounds and Optimality**



# Chapter 15

## Lower Bounds for Comparison-Based Algorithms

### 15.1 Decision Trees

#### 15.1.1 Modeling Algorithms as Decision Trees

#### 15.1.2 Height of Decision Trees and Worst-Case Complexity

### 15.2 Sorting Lower Bound

#### 15.2.1 Information-Theoretic Argument

#### 15.2.2 $\Omega(n \log n)$ Lower Bound for Comparison Sorting

#### 15.2.3 Implications and Optimal Algorithms

### 15.3 Selection and Searching Lower Bounds

#### 15.3.1 Finding the Minimum: $\Omega(n)$

#### 15.3.2 Finding Median: Adversary Arguments

#### 15.3.3 Searching in Sorted Arrays: $\Omega(\log n)$

### 15.4 Adversary Arguments

#### 15.4.1 General Framework

#### 15.4.2 Examples: Merging, Element Uniqueness

### 15.5 Exercises

# **Chapter 16**

## **Algebraic and Non-Comparison Lower Bounds**

### **16.1 Algebraic Decision Trees**

#### **16.1.1 Extending Beyond Comparisons**

#### **16.1.2 Element Distinctness Lower Bound**

### **16.2 Communication Complexity**

#### **16.2.1 Models and Definitions**

#### **16.2.2 Applications to Data Structures**

### **16.3 Cell-Probe Model**

#### **16.3.1 Lower Bounds for Data Structures**

#### **16.3.2 Dynamic vs. Static Data Structures**

### **16.4 Exercises**



## **Part V**

# **Specialized Topics and Applications**



# Chapter 17

## Analysis of Specific Algorithm Paradigms

### 17.1 Divide-and-Conquer Algorithms

#### 17.1.1 General Framework and Recurrence Relations

#### 17.1.2 Examples: Mergesort, Quicksort, Strassen's Algorithm

#### 17.1.3 Optimality and Lower Bounds

### 17.2 Greedy Algorithms

#### 17.2.1 Correctness via Exchange Arguments

#### 17.2.2 Matroid Theory (Brief Introduction)

#### 17.2.3 Examples: Huffman Coding, Kruskal's MST

### 17.3 Dynamic Programming

#### 17.3.1 Optimal Substructure and Overlapping Subproblems

#### 17.3.2 Memoization vs. Tabulation

#### 17.3.3 Time and Space Complexity Analysis

#### 17.3.4 Examples: Knapsack, Edit Distance, Matrix Chain Multiplication

### 17.4 Backtracking and Branch-and-Bound

#### 17.4.1 Pruning the Search Space

#### 17.4.2 Worst-Case Exponential, Average-Case Better

# **Chapter 18**

## **Online Algorithms and Competitive Analysis**

### **18.1 Introduction to Online Algorithms**

#### **18.1.1 Online vs. Offline Problems**

#### **18.1.2 Competitive Ratio**

### **18.2 Examples of Online Problems**

#### **18.2.1 Paging and Caching (LRU, FIFO, LFU)**

#### **18.2.2 Load Balancing**

#### **18.2.3 Online Scheduling**

### **18.3 Competitive Analysis Techniques**

#### **18.3.1 Deterministic vs. Randomized Algorithms**

#### **18.3.2 Lower Bounds via Adversary Arguments**

### **18.4 Exercises**

# Chapter 19

## Approximation Algorithms

### 19.1 Introduction to Approximation

#### 19.1.1 NP-Hardness and Intractability

#### 19.1.2 Approximation Ratios

### 19.2 Examples of Approximation Algorithms

#### 19.2.1 Vertex Cover (2-Approximation)

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

#### 19.2.3 Traveling Salesman Problem (Metric TSP)

### 19.3 Analysis Techniques

#### 19.3.1 Bounding Optimal Solutions

#### 19.3.2 Linear Programming Relaxations

### 19.4 Exercises

# Chapter 20

## Parameterized Complexity

### 20.1 Introduction to Parameterized Algorithms

#### 20.1.1 Fixed-Parameter Tractability (FPT)

#### 20.1.2 Kernelization

### 20.2 Examples and Analysis

#### 20.2.1 Vertex Cover Parameterized by Solution Size

#### 20.2.2 Treewidth and Graph Algorithms

### 20.3 W-Hierarchy and Hardness

#### 20.3.1 $W[1]$ , $W[2]$ , and Beyond

### 20.4 Exercises

## **Part VI**

# **Practical Considerations and Case Studies**

# Chapter 21

## From Theory to Practice

### 21.1 Hidden Constants and Lower-Order Terms

#### 21.1.1 When $O(n \log n)$ Beats $O(n)$ in Practice

#### 21.1.2 Empirical Performance Measurements

### 21.2 Algorithm Engineering

#### 21.2.1 Profiling and Benchmarking

#### 21.2.2 Tuning for Specific Hardware

#### 21.2.3 Libraries and Implementations (STL, Boost, etc.)

### 21.3 Parallel and Distributed Algorithm Analysis

#### 21.3.1 Scalability and Speedup

#### 21.3.2 Amdahl's Law and Gustafson's Law

### 21.4 Energy Efficiency

#### 21.4.1 Energy as a Resource

#### 21.4.2 Green Computing and Mobile Devices

### 21.5 Exercises



# **Chapter 22**

## **Case Studies**

### **22.1 Sorting Algorithms in Practice**

#### **22.1.1 Timsort, Introsort, Radix Sort**

#### **22.1.2 Comparison of Theoretical vs. Empirical Performance**

### **22.2 Graph Algorithms in Large-Scale Systems**

#### **22.2.1 Web Graphs and PageRank**

#### **22.2.2 Social Network Analysis**

### **22.3 Machine Learning and Data Science**

#### **22.3.1 Complexity of Training Algorithms**

#### **22.3.2 SGD, AdaGrad, Adam: Time and Space Analysis**

### **22.4 Database Systems**

#### **22.4.1 Query Optimization**

#### **22.4.2 Indexing Structures (B-Trees, LSM-Trees)**

### **22.5 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**