ART

_ _ . _ . .

ALGORITHM ANALYSIS

FROM FOUNDATIONS TO PRACTICE

 $\Theta(\log n)$

^{Ω(n²)} Mahdi

LIVING FIRST EDITION

Ahlaly

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 18, 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: 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 18, 2025	
<u> Бисстье</u> шы пришеи. Остобет 10, 2025	

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

Contents

Ti	itle P	age		į
C	onte	nts		iii
P	refac	e		viii
Α	ckno	wledg	jmentsxx	xiii
I	Fo	unda	itions	1
1	Wh	y "Pre	ecise Analysis" Matters — From Theory to Engineering	2
	1.1	The III	usion of Intuitive Understanding	3
		1.1.1	Case Study 1: When $O(n^2)$ Beats $O(n \log n)$ in Practice	3
		1.1.2	Case Study 2: The QuickSort Paradox — $O(n^2)$ Worst-Case, Yet Industry	
			Standard	3
		1.1.3	Case Study 3: Hash Tables vs. Binary Search Trees — Theory vs. Reality	3
	1.2	The G	ap Between Theoretical Complexity and Real-World Performance	3
		1.2.1	Hidden Constants in Big-O Notation	3
		1.2.2	Lower-Order Terms That Dominate at Practical Scales	3
		1.2.3	Memory Hierarchy Effects Invisible to RAM Model	3
		1.2.4	Architecture-Specific Considerations	3
	1.3	The R	ole of Constants, Lower-Order Terms, and Hardware	3
		1.3.1	Quantifying Constants: From Theory to Measurement	3
		1.3.2	When Lower-Order Terms Matter	3
		1.3.3	Cache Effects and Memory Bandwidth	3
		1.3.4	Branch Prediction and Pipeline Stalls	3
		1.3.5	SIMD and Vectorization Opportunities	3
	1.4	Bridgi	ng the Theory-Practice Gap	3
		1.4.1	Algorithm Engineering Principles	3
		1.4.2	When to Trust Theory	3
		1.4.3	When to Question Theory	3
		1.4.4	Hybrid Approaches: Combining Analytical and Empirical Methods	3
	1.5	Why V	Ve Still Need Asymptotic Analysis	3
		1.5.1	Scalability Prediction	3
		1.5.2	Algorithm Comparison Across Architectures	3

		1.5.3	Identifying Bottlenecks	3
		1.5.4	Guiding Optimization Efforts	3
	1.6	Exercis	ses	3
2	Mat	hemat	ical and Algorithmic Prerequisites	4
	2.1	Discret	e Mathematics	5
		2.1.1	Sets, Functions, and Relations	5
		2.1.2	Combinatorics: Permutations, Combinations, and Binomial Coefficients .	5
		2.1.3	Graph Theory Basics	5
		2.1.4	Proof Techniques: Induction, Contradiction, and Contrapositive	5
	2.2	Eleme	ntary Probability Theory	5
		2.2.1	Sample Spaces, Events, and Probability Measures	5
		2.2.2	Random Variables and Expectations	5
		2.2.3	Basic Distributions: Uniform, Bernoulli, Geometric, Binomial	5
		2.2.4	Linearity of Expectation	5
		2.2.5	Conditional Probability and Independence	5
		2.2.6	Variance and Standard Deviation	5
		2.2.7	Moment Generating Functions (Brief Introduction)	5
	2.3	Mather	matical Analysis	5
		2.3.1	Limits, Continuity, and Asymptotic Behavior	5
		2.3.2	Sequences and Series	5
		2.3.3	Summations and Closed Forms	5
		2.3.4	Integration and Differentiation (Brief Review)	5
		2.3.5	Taylor Series and Asymptotic Expansions	5
		2.3.6	Stirling's Approximation	5
	2.4	Linear	Algebra (Brief Overview)	5
		2.4.1	Vectors, Matrices, and Linear Transformations	5
		2.4.2	Eigenvalues and Eigenvectors	5
		2.4.3	Applications to Markov Chains and Graph Algorithms	5
		2.4.4	Matrix Operations and Complexity	5
	2.5	Numbe	er Theory Essentials	5
		2.5.1	Divisibility and Modular Arithmetic	5
		2.5.2	Prime Numbers and Factorization	5
		2.5.3	Greatest Common Divisor and Euclidean Algorithm	5
		2.5.4	Applications to Cryptography and Hashing	5

	2.6	Additio	onal Mathematical Tools	5
		2.6.1	Logarithms and Exponentials	5
		2.6.2	Floor and Ceiling Functions	5
		2.6.3	Asymptotic Notation (Informal Preview)	5
	2.7	Self-A	ssessment Exercises	5
		2.7.1	Discrete Mathematics Problems	5
		2.7.2	Probability Problems	5
		2.7.3	Analysis Problems	5
		2.7.4	Linear Algebra Problems	5
		2.7.5	Number Theory Problems	5
	2.8	Furthe	er Reading and Resources	5
		2.8.1	Recommended Textbooks for Each Topic	5
		2.8.2	Online Resources and Video Lectures	5
		2.8.3	Practice Problem Collections	5
Ш	Fo	ounda	ations of Algorithmic Analysis	6
				7
3			ion to Algorithm Analysis	_
	3.1		Is Algorithm Analysis?	7
		3.1.1	Correctness vs. Efficiency	7
		3.1.2	Resource Measures: Time, Space, Energy, I/O	7
		3.1.3	The Need for Mathematical Models	7
	3.2	The R	AM Model of Computation	7
		3.2.1	Basic Operations and Unit-Cost Assumption	7
		3.2.2	Memory Access Model	7
		3.2.3	Limitations and Extensions of the RAM Model	7
	3.3	Measu	uring Algorithm Performance	7
		3.3.1	Input Size and Problem Instances	7
		3.3.2	Counting Basic Operations	7
		3.3.3	Exact vs. Asymptotic Analysis	7
	3.4	Overvi	iew of Complexity Classes	7
		3.4.1	P, NP, NP-Complete, and NP-Hard (Brief Introduction)	7
		3.4.2	Why We Focus on Polynomial-Time Algorithms	7
4	Asy	mptot	tic Notation	8
	4.1	The No	eed for Asymptotic Analysis	9

	4.1.1	Why Exact Counts Are Often Impractical	Ś
	4.1.2	Growth Rates and Scalability	S
4.2	Big-O	Notation (O)	ć
	4.2.1	Formal Definition	S
	4.2.2	Intuition: Upper Bounds	ç
	4.2.3	Common Functions and Their Growth Rates	S
	4.2.4	Examples and Non-Examples	S
	4.2.5	Properties of Big-O	S
4.3	Big-Or	mega Notation (Ω)	ę
	4.3.1	Formal Definition	S
	4.3.2	Intuition: Lower Bounds	S
	4.3.3	Examples and Applications	Ś
	4.3.4	Relationship Between O and Ω	S
4.4	Big-Th	eta Notation (Θ)	ć
	4.4.1	Formal Definition	S
	4.4.2	Intuition: Tight Bounds	Ś
	4.4.3	When to Use Θ vs. O	ç
	4.4.4	Examples of Tight Bounds	S
4.5	Little-o	and Little-omega Notation (o,ω)	ę
	4.5.1	Formal Definitions	S
	4.5.2	Strict Asymptotic Bounds	Ś
	4.5.3	Applications in Analysis	ç
4.6	Comm	on Misconceptions and Pitfalls	ç
	4.6.1	Confusing O with Θ	ç
	4.6.2	Ignoring Constants in Practice	S
	4.6.3	Misapplying Asymptotic Notation to Small Inputs	S
4.7	Compa	aring Functions	ç
	4.7.1	L'Hôpital's Rule for Limits	S
	4.7.2	Logarithmic vs. Polynomial vs. Exponential Growth	Ś
	4.7.3	Hierarchy of Common Complexity Classes	S
4.8	Exercis	ses	ç
Red	urren	ce Relations and Their Solutions	10
5.1		uction to Recurrence Relations	11
٠			

5

	5.1.1	What Are Recurrences?	11
	5.1.2	Why They Arise in Algorithm Analysis	11
	5.1.3	Examples from Divide-and-Conquer Algorithms	11
5.2	The Su	ubstitution Method	11
	5.2.1	Guessing the Solution	11
	5.2.2	Proving by Induction	11
	5.2.3	Examples: Mergesort, Binary Search	11
	5.2.4	Strengthening the Inductive Hypothesis	11
5.3	The Re	ecursion-Tree Method	11
	5.3.1	Visualizing the Recurrence	11
	5.3.2	Summing Over Levels	11
	5.3.3	Examples and Illustrations	11
	5.3.4	Limitations and When to Use	11
5.4	The M	aster Theorem	11
	5.4.1	Statement of the Master Theorem (Standard Form)	11
	5.4.2	Three Cases and Their Intuition	11
	5.4.3	Proof Sketch (Via Recursion Trees)	11
	5.4.4	Examples: $T(n) = aT(n/b) + f(n) \dots \dots \dots \dots \dots$	11
	5.4.5	Regularity Condition and Edge Cases	11
	5.4.6	Extended Master Theorem (Akra-Bazzi)	11
5.5	The Al	kra-Bazzi Method	11
	5.5.1	Motivation: Unequal Subproblem Sizes	11
	5.5.2	Statement and Conditions	11
	5.5.3	Examples and Applications	11
	5.5.4	Proof Overview (Advanced)	11
5.6	Linear	Recurrences with Constant Coefficients	11
	5.6.1	Homogeneous Linear Recurrences	11
	5.6.2	Characteristic Equations	11
	5.6.3	Solving Fibonacci-Type Recurrences	11
	5.6.4	Non-Homogeneous Recurrences and Particular Solutions	11
5.7	Genera	ating Functions	11
	5.7.1	Introduction to Generating Functions	11
	5.7.2	Solving Recurrences with Generating Functions	11
	5.7.3	Examples: Catalan Numbers, Stirling Numbers	11

7	Pro	babilis	stic Analysis of Algorithms1	4
	6.7	Exercis	ses 1	13
		6.6.3	Case Study: Simplex Algorithm	13
		6.6.2	Introduction to Smoothed Analysis	13
		6.6.1	Motivation: Beyond Worst-Case Pessimism	13
	6.6	Smoot	hed Analysis	13
		6.5.3	Las Vegas vs. Monte Carlo Algorithms	13
		6.5.2	Randomized Quicksort: Expected $O(n \log n)$	13
		6.5.1	Distinction Between the Two Concepts	13
	6.5	Probab	oilistic Analysis vs. Randomized Algorithms	13
		6.4.4	Examples: Quicksort, Hashing, Skip Lists	13
		6.4.3	Probabilistic Models: Uniform, Gaussian, etc	13
		6.4.2	Assumptions About Input Distributions	13
		6.4.1	Definition: Expected Running Time	13
	6.4	Averag	ge-Case Analysis	13
		6.3.4	Lower Bounds and Optimality	13
		6.3.3	Examples: Quicksort, Searching in Unsorted Arrays	13
		6.3.2	Guarantees and Robustness	13
		6.3.1	Definition and Motivation	13
	6.3	Worst-	Case Analysis	13
		6.2.3	When Best-Case Matters (and When It Doesn't)	13
		6.2.2	Examples: Insertion Sort, Linear Search	13
		6.2.1	Definition and Purpose	13
	6.2	Best-C	Case Analysis	13
		6.1.2	Problem Instances and Instance Distributions	13
		6.1.1	What Constitutes an "Input"?	13
	6.1	Definin	ng Input Classes	13
6	Bes	t-Case	e, Worst-Case, and Average-Case Analysis 1	2
	5.9	Exercis	ses	11
		5.8.3	,	11
		5.8.2		11
		5.8.1	Full History Recurrences	11
	5.8	Advan	ced Topics	11

	7.1	Founda	ations of Probabilistic Analysis	15
		7.1.1	Random Variables in Algorithm Analysis	15
		7.1.2	Indicator Random Variables	15
		7.1.3	Linearity of Expectation	15
	7.2	Expect	ed Running Time	15
		7.2.1	Formal Definition	15
		7.2.2	Computing Expectations via Indicator Variables	15
		7.2.3	Examples: Hiring Problem, Randomized Quicksort	15
	7.3	Probab	pilistic Bounds	15
		7.3.1	Markov's Inequality	15
		7.3.2	Chebyshev's Inequality	15
		7.3.3	Chernoff Bounds	15
		7.3.4	Applications to Load Balancing and Hashing	15
	7.4	Rando	mized Algorithms	15
		7.4.1	Randomized Quicksort (Detailed Analysis)	15
		7.4.2	Randomized Selection (Quickselect)	15
		7.4.3	Hashing and Universal Hash Functions	15
		7.4.4	Bloom Filters and Probabilistic Data Structures	15
	7.5	Analys	is of Randomized Data Structures	15
		7.5.1	Skip Lists	15
		7.5.2	Treaps	15
		7.5.3	Hash Tables with Chaining and Open Addressing	15
	7.6	High-P	robability Results	15
		7.6.1	What Does "With High Probability" Mean?	15
		7.6.2	Concentration Inequalities	15
		7.6.3	Union Bound and Probabilistic Method	15
	7.7	Exercis	ses	15
Ш	A	dvan	ced Analysis Techniques	16
8	Amo	ortized	d Analysis	17
	8.1		·	18
	0.1	8.1.1	Motivation: Why Average Per-Operation Cost?	18
		8.1.2	Amortized vs. Average-Case Analysis	18
		8.1.3	When to Use Amortized Analysis	18
		0.1.0	Whom to Ose America Analysis	10

	8.2	Aggreg	gate Analysis	18
		8.2.1	Definition and Methodology	18
		8.2.2	Example: Dynamic Array (Vector) Resizing	18
		8.2.3	Example: Binary Counter Increment	18
		8.2.4	Example: Stack with Multipop	18
	8.3	The Ac	counting Method	18
		8.3.1	Conceptual Framework: Credits and Debits	18
		8.3.2	Defining Amortized Costs	18
		8.3.3	Example: Dynamic Array via Accounting	18
		8.3.4	Example: Splay Trees (Introduction)	18
		8.3.5	Ensuring Non-Negative Credit Balance	18
	8.4	The Po	otential Method	18
		8.4.1	Potential Functions: Definition and Intuition	18
		8.4.2	Relating Amortized Cost to Actual Cost	18
		8.4.3	Designing Good Potential Functions	18
		8.4.4	Example: Dynamic Array via Potential Method	18
		8.4.5	Example: Binary Counter via Potential Method	18
		8.4.6	Example: Fibonacci Heaps (Overview)	18
	8.5	Compa	aring the Three Methods	18
		8.5.1	Strengths and Weaknesses	18
		8.5.2	When to Choose Which Method	18
		8.5.3	Equivalence of Methods (Informal Discussion)	18
	8.6	Advand	ced Applications	18
		8.6.1	Splay Trees: Full Analysis	18
		8.6.2	Fibonacci Heaps	18
		8.6.3	Disjoint-Set Union (Union-Find)	18
	8.7	Exercis	ses	18
9	Spa	ce Co	mplexity Analysis	19
	• 9.1			20
		9.1.1	Why Space Matters	20
		9.1.2	Types of Memory: Stack, Heap, Static	20
		9.1.3	In-Place vs. Out-of-Place Algorithms	20
	9.2	Measu	•	20

		9.2.1	Auxiliary Space vs. Total Space	20
		9.2.2	Recursive Call Stack Depth	20
		9.2.3	Implicit vs. Explicit Data Structures	20
	9.3	Examp	les of Space Complexity Analysis	20
		9.3.1	Iterative Algorithms: Loops and Arrays	20
		9.3.2	Recursive Algorithms: Mergesort, Quicksort	20
		9.3.3	Dynamic Programming: Memoization vs. Tabulation	20
		9.3.4	Graph Algorithms: BFS, DFS, Shortest Paths	20
	9.4	Space-	Time Tradeoffs	20
		9.4.1	Caching and Memoization	20
		9.4.2	Lookup Tables and Precomputation	20
		9.4.3	Compression and Succinct Data Structures	20
	9.5	Stream	ing and Online Algorithms	20
		9.5.1	Sublinear Space Algorithms	20
		9.5.2	Sketching and Sampling Techniques	20
		9.5.3	Examples: Distinct Elements, Heavy Hitters	20
	9.6	Space	Complexity Classes	20
		9.6.1	L, NL, PSPACE (Brief Overview)	20
		9.6.2	Savitch's Theorem	20
	9.7	Exercis	es	20
0	Cac	he-Aw	are and I/O Complexity	21
	10.1	Introdu	ction to the Memory Hierarchy	22
		10.1.1	Registers, Cache (L1, L2, L3), RAM, Disk	22
		10.1.2	Latency and Bandwidth Characteristics	22
		10.1.3	Why Algorithm Design Must Consider Memory	22
	10.2	The Ex	ternal Memory Model (I/O Model)	22
		10.2.1	Parameters: N (data size), M (memory size), B (block size)	22
		10.2.2	I/O Complexity: Counting Block Transfers	22
		10.2.3	Comparison with RAM Model	22
	10.3	I/O-Effi	cient Algorithms	22
		10.3.1	Scanning and Sorting	22
		10.3.2	Matrix Operations	22
		10.3.3	Graph Algorithms	22

10.4	Cache	-Oblivious Algorithms	22
	10.4.1	Motivation: Optimal Without Knowing M and B	22
	10.4.2	Cache-Oblivious Sorting (Funnelsort)	22
	10.4.3	Cache-Oblivious Matrix Multiplication	22
	10.4.4	Cache-Oblivious B-Trees (van Emde Boas Layout)	22
10.5	Cache	-Aware Analysis	22
	10.5.1	Modeling Cache Behavior	22
	10.5.2	Locality of Reference: Temporal and Spatial	22
	10.5.3	Blocking and Tiling Techniques	22
10.6	Real-V	Vorld Considerations	22
	10.6.1	Multi-Level Caches	22
	10.6.2	Cache Replacement Policies (LRU, LFU, etc.)	22
	10.6.3	Prefetching and Speculative Execution	22
	10.6.4	False Sharing and Cache Line Effects	22
10.7	Case S	Studies	22
	10.7.1	Database Query Processing	22
	10.7.2	External Memory Sorting in Practice	22
	10.7.3	Scientific Computing and Large-Scale Simulations	22
10.8	Exercis	ses	22
11 Cac	he-Aw	vare Scheduling and Analysis for Multicores	23
11.1	Introdu	uction to Multicore and Parallel Computing	24
	11.1.1	Shared vs. Distributed Memory	24
	11.1.2	Parallel Models: PRAM, Fork-Join, Work-Stealing	24
	11.1.3	Performance Metrics: Work, Span, Parallelism	24
11.2	Cache	Coherence and Consistency	24
	11.2.1	MESI and MOESI Protocols	24
	11.2.2	False Sharing in Multicore Systems	24
	11.2.3	Impact on Algorithm Design	24
11.3	Cache	-Aware Parallel Algorithms	24
	11.3.1	Parallel Sorting with Cache Awareness	24
	11.3.2	Parallel Matrix Multiplication (Strassen, Coppersmith-Winograd)	24
	11.3.3	Load Balancing and Task Granularity	24
11.4	Real-T	ime and Embedded Systems	24

		11.4.1	WCET Analysis in Cache-Aware Contexts	24
		11.4.2	Predictability vs. Average-Case Performance	24
		11.4.3	Cache Partitioning and Locking	24
	11.5	Schedu	uling Strategies	24
		11.5.1	Static vs. Dynamic Scheduling	24
		11.5.2	Work-Stealing Algorithms	24
		11.5.3	Affinity Scheduling for Cache Locality	24
	11.6	Analys	is Techniques	24
		11.6.1	DAG-Based Analysis (Cilk Model)	24
		11.6.2	Brent's Theorem and Greedy Scheduling	24
		11.6.3	Cache Miss Analysis in Parallel Programs	24
	11.7	Case S	Studies from Research	24
		11.7.1	ECRTS 2007: Cache-Aware Real-Time Scheduling	24
		11.7.2	Cache-Aware Scheduling for Multicores (Embedded Systems)	24
		11.7.3	VLDB 2019: Concurrent Hash Tables and Cache Performance	24
	11.8	Exercis	es	24
IV	- 14	ΛWΔr	Rounds and Ontimality	25
IV			'	25
12	Low	er Bo	unds for Comparison-Based Algorithms	2526
12	Low	er Bo	unds for Comparison-Based Algorithms	
12	Low	er Bo	unds for Comparison-Based Algorithms	26
12	Low	ver Bo	unds for Comparison-Based Algorithms	26
12	Low 12.1	Decision 12.1.1 12.1.2	unds for Comparison-Based Algorithms	26 27 27
12	Low 12.1	Decision 12.1.1 12.1.2	unds for Comparison-Based Algorithms on Trees	26 27 27 27
12	Low 12.1	Decision 12.1.1 12.1.2 Sorting	unds for Comparison-Based Algorithms on Trees Modeling Algorithms as Decision Trees Height of Decision Trees and Worst-Case Complexity Lower Bound	26 27 27 27 27
12	Low 12.1	Decision 12.1.1 12.1.2 Sorting 12.2.1	unds for Comparison-Based Algorithms on Trees Modeling Algorithms as Decision Trees Height of Decision Trees and Worst-Case Complexity Lower Bound Information-Theoretic Argument	26 27 27 27 27 27
12	Low 12.1 12.2	Decision 12.1.1 12.1.2 Sorting 12.2.1 12.2.2 12.2.3	unds for Comparison-Based Algorithms on Trees Modeling Algorithms as Decision Trees Height of Decision Trees and Worst-Case Complexity Lower Bound Information-Theoretic Argument $\Omega(n \log n)$ Lower Bound for Comparison Sorting	26 27 27 27 27 27 27 27
12	Low 12.1 12.2	Decision 12.1.1 12.1.2 Sorting 12.2.1 12.2.2 12.2.3	unds for Comparison-Based Algorithms on Trees Modeling Algorithms as Decision Trees Height of Decision Trees and Worst-Case Complexity Lower Bound Information-Theoretic Argument $\Omega(n \log n)$ Lower Bound for Comparison Sorting Implications and Optimal Algorithms	26 27 27 27 27 27 27
12	Low 12.1 12.2	Decision 12.1.1 12.1.2 Sorting 12.2.1 12.2.2 12.2.3 Selection	unds for Comparison-Based Algorithms on Trees Modeling Algorithms as Decision Trees Height of Decision Trees and Worst-Case Complexity Lower Bound Information-Theoretic Argument $\Omega(n \log n)$ Lower Bound for Comparison Sorting Implications and Optimal Algorithms on and Searching Lower Bounds	26 27 27 27 27 27 27 27
12	Low 12.1 12.2	Decision 12.1.1 12.1.2 Sorting 12.2.1 12.2.2 12.2.3 Selection 12.3.1	unds for Comparison-Based Algorithms on Trees Modeling Algorithms as Decision Trees Height of Decision Trees and Worst-Case Complexity Lower Bound Information-Theoretic Argument $\Omega(n\log n)$ Lower Bound for Comparison Sorting Implications and Optimal Algorithms on and Searching Lower Bounds Finding the Minimum: $\Omega(n)$	26 27 27 27 27 27 27 27 27 27
12	Low 12.1 12.2	Decision 12.1.1 12.1.2 Sorting 12.2.1 12.2.2 12.2.3 Selection 12.3.1 12.3.2 12.3.3	unds for Comparison-Based Algorithms on Trees Modeling Algorithms as Decision Trees Height of Decision Trees and Worst-Case Complexity Lower Bound Information-Theoretic Argument $\Omega(n \log n)$ Lower Bound for Comparison Sorting Implications and Optimal Algorithms on and Searching Lower Bounds Finding the Minimum: $\Omega(n)$ Finding Median: Adversary Arguments	26 27 27 27 27 27 27 27 27 27 27
12	Low 12.1 12.2	Decision 12.1.1 12.1.2 Sorting 12.2.1 12.2.2 12.2.3 Selection 12.3.1 12.3.2 12.3.3	unds for Comparison-Based Algorithms on Trees Modeling Algorithms as Decision Trees Height of Decision Trees and Worst-Case Complexity Lower Bound Information-Theoretic Argument $\Omega(n \log n)$ Lower Bound for Comparison Sorting Implications and Optimal Algorithms on and Searching Lower Bounds Finding the Minimum: $\Omega(n)$ Finding Median: Adversary Arguments Searching in Sorted Arrays: $\Omega(\log n)$	26 27 27 27 27 27 27 27 27 27 27
12	Low 12.1 12.2	Decision 12.1.1 12.1.2 Sorting 12.2.1 12.2.2 12.2.3 Selection 12.3.1 12.3.2 12.3.3 Advers 12.4.1	unds for Comparison-Based Algorithms on Trees Modeling Algorithms as Decision Trees Height of Decision Trees and Worst-Case Complexity Lower Bound Information-Theoretic Argument $\Omega(n \log n)$ Lower Bound for Comparison Sorting Implications and Optimal Algorithms on and Searching Lower Bounds Finding the Minimum: $\Omega(n)$ Finding Median: Adversary Arguments Searching in Sorted Arrays: $\Omega(\log n)$ ary Arguments	26 27 27 27 27 27 27 27 27 27 27 27

13	Alge	ebraic	and Non-Comparison Lower Bounds	28
	13.1	Algebra	aic Decision Trees	28
		13.1.1	Extending Beyond Comparisons	28
		13.1.2	Element Distinctness Lower Bound	28
	13.2	Comm	unication Complexity	28
		13.2.1	Models and Definitions	28
		13.2.2	Applications to Data Structures	28
	13.3	Cell-Pr	robe Model	28
		13.3.1	Lower Bounds for Data Structures	28
		13.3.2	Dynamic vs. Static Data Structures	28
	13.4	Exercis	ses	28
V	Sp	ecia	lized Topics and Applications 2	29
14	Ana	lysis (of Specific Algorithm Paradigms	30
	14.1	Divide-	and-Conquer Algorithms	31
		14.1.1	General Framework and Recurrence Relations	31
		14.1.2	Examples: Mergesort, Quicksort, Strassen's Algorithm	31
		14.1.3	Optimality and Lower Bounds	31
	14.2	Greedy	/ Algorithms	31
		14.2.1	Correctness via Exchange Arguments	31
		14.2.2	Matroid Theory (Brief Introduction)	31
		14.2.3	Examples: Huffman Coding, Kruskal's MST	31
	14.3	Dynam	nic Programming	31
		14.3.1	Optimal Substructure and Overlapping Subproblems	31
		14.3.2	Memoization vs. Tabulation	31
		14.3.3	Time and Space Complexity Analysis	31
		14.3.4	Examples: Knapsack, Edit Distance, Matrix Chain Multiplication	31
	14.4	Backtra	acking and Branch-and-Bound	31
		14.4.1	Pruning the Search Space	31
		14.4.2	Worst-Case Exponential, Average-Case Better	31
		14.4.3	Examples: N-Queens, Traveling Salesman	31
	14.5	Exercis	ses	31
15	Onli	ine Al	gorithms and Competitive Analysis	32

	15.1	1 Introduction to Online Algorithms				
		15.1.1	Online vs. Offline Problems	32		
		15.1.2	Competitive Ratio	32		
15.2 Examples of Online Problems			les of Online Problems	32		
		15.2.1	Paging and Caching (LRU, FIFO, LFU)	32		
		15.2.2	Load Balancing	32		
		15.2.3	Online Scheduling	32		
	15.3	Compe	etitive Analysis Techniques	32		
		15.3.1	Deterministic vs. Randomized Algorithms	32		
		15.3.2	Lower Bounds via Adversary Arguments	32		
	15.4	Exercis	ses	32		
16	Арр	roxim	ation Algorithms	33		
	16.1	Introdu	ction to Approximation	33		
		16.1.1	NP-Hardness and Intractability	33		
		16.1.2	Approximation Ratios	33		
	16.2	Examp	les of Approximation Algorithms	33		
		16.2.1	Vertex Cover (2-Approximation)	33		
		16.2.2	Set Cover (Greedy, $\log n$ -Approximation)	33		
		16.2.3	Traveling Salesman Problem (Metric TSP)	33		
	16.3	Analys	is Techniques	33		
		16.3.1	Bounding Optimal Solutions	33		
		16.3.2	Linear Programming Relaxations	33		
	16.4	Exercis	ses	33		
17	Para	ametei	rized Complexity	34		
			ction to Parameterized Algorithms	34		
		17.1.1	Fixed-Parameter Tractability (FPT)	34		
		17.1.2	Kernelization	34		
	17.2	Examp	les and Analysis	34		
		17.2.1	Vertex Cover Parameterized by Solution Size	34		
		17.2.2	Treewidth and Graph Algorithms	34		
	17.3	W-Hier	archy and Hardness	34		
			W[1], W[2], and Beyond	34		
	17.4	Exercis	ses	34		

VI	P	ractio	cal Considerations and Case Studies	35
18	From	n The	ory to Practice	36
	18.1	Hidden	Constants and Lower-Order Terms	36
		18.1.1	When $O(n \log n)$ Beats $O(n)$ in Practice	36
		18.1.2	Empirical Performance Measurements	36
	18.2	Algorith	nm Engineering	36
		18.2.1	Profiling and Benchmarking	36
		18.2.2	Tuning for Specific Hardware	36
		18.2.3	Libraries and Implementations (STL, Boost, etc.)	36
	18.3	Paralle	I and Distributed Algorithm Analysis	36
		18.3.1	Scalability and Speedup	36
		18.3.2	Amdahl's Law and Gustafson's Law	36
	18.4	Energy	Efficiency	36
		18.4.1	Energy as a Resource	36
		18.4.2	Green Computing and Mobile Devices	36
	18.5	Exercis	ses	36
19	Cas	e Stuc	lies	37
	19.1	Sorting	Algorithms in Practice	37
		19.1.1	Timsort, Introsort, Radix Sort	37
		19.1.2	Comparison of Theoretical vs. Empirical Performance	37
	19.2	Graph	Algorithms in Large-Scale Systems	37
		19.2.1	Web Graphs and PageRank	37
		19.2.2	Social Network Analysis	37
	19.3	Machin	e Learning and Data Science	37
		19.3.1	Complexity of Training Algorithms	37
		19.3.2	SGD, AdaGrad, Adam: Time and Space Analysis	37
	19.4	Databa	se Systems	37
		19.4.1	Query Optimization	37
		19.4.2	Indexing Structures (B-Trees, LSM-Trees)	37
	19.5	Exercis	ses	37
Α	Mat	hemat	ical Background	38
	A.1	0	ation Formulas	38

	A.2	Logarithms and Exponentials	38			
	A.3	Recurrence Relations (Quick Reference)	38			
	A.4	Probability Distributions	38			
	A.5	Matrix Operations	38			
В	Pse	udocode Conventions	39			
	B.1	Notation and Style	39			
	B.2	Common Data Structures	39			
С	Sol	utions to Selected Exercises	40			
D	Glo	ssary of Terms	41			
Ε	Inde	ex of Algorithms	42			
F	Annotated Bibliography					
	F.1	Foundational Texts	43			
	F.2	Research Papers by Topic	43			
	F.3	Online Courses and Resources	43			

Preface

 $E^{\scriptscriptstyle ext{VERY RIGOROUS JOURNEY}}$ 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
- 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

Why "Precise Analysis" Matters — From Theory to Engineering

- 1.1 The Illusion of Intuitive Understanding
- **1.1.1** Case Study 1: When $O(n^2)$ Beats $O(n \log n)$ in Practice
- 1.1.2 Case Study 2: The QuickSort Paradox $O(n^2)$ Worst-Case, Yet Industry Standard
- 1.1.3 Case Study 3: Hash Tables vs. Binary Search Trees Theory vs. Reality
- 1.2 The Gap Between Theoretical Complexity and Real-World Performance
- 1.2.1 Hidden Constants in Big-O Notation
- 1.2.2 Lower-Order Terms That Dominate at Practical Scales
- 1.2.3 Memory Hierarchy Effects Invisible to RAM Model
- 1.2.4 Architecture-Specific Considerations
- 1.3 The Role of Constants, Lower-Order Terms, and Hardware
- 1.3.1 Quantifying Constants: From Theory to Measurement
- 1.3.2 When Lower-Order Terms Matter

3|43

- 1.3.3 Cache Effects and Memory Bandwidth
- 1.3.4 Branch Prediction and Pipeline Stalls

Mathematical and Algorithmic Prerequisites

2.1 Discrete Mathematics

2.1.1 Sets, Functions, and Relations

Set Theory Basics

Operations on Sets

Functions: Injective, Surjective, Bijective

Relations and Equivalence Classes

Order Relations and Partial Orders

2.1.2 Combinatorics: Permutations, Combinations, and Binomial Coefficients

The Fundamental Counting Principles

Permutations with and without Repetition

Combinations and the Binomial Theorem

Pigeonhole Principle and Applications

Inclusion-Exclusion Principle

Combinatorial Identities Useful in Analysis

2.1.3 Graph Theory Basics

Graph Definitions and Representations

Paths, Cycles, and Connectivity

Trees and Their Properties

rst Edition • 2025

Directed Graphs and DAGs

Graph Algorithms Preview (BFS, DFS)

Part II

Foundations of Algorithmic Analysis

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				

Asymptotic Notation

4.1 The Need for Asymptotic Analy	/SİS
-----------------------------------	------

- 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

$\overset{\scriptscriptstyle{\mathsf{First}}}{\mathsf{Edition}}\overset{\scriptscriptstyle{\mathsf{2025}}}{\mathsf{Big}}$ -Theta Notation (Θ)

11 | 43

Chapter 5

Recurrence Relations and Their Solutions

5 1	Introdu	iction	to Re	currence	$\mathbf{R}_{\mathbf{A}}$	lations
.D. I		16.116911	10 12 6	CHIFFENCE	, I/61	IALIOUS

- 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

5.4.2

5.4.1 Statement of the Master Theorem (Standard Form)

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

6.1 Defini	ng Input	Classes
------------	----------	---------

- 6.1.1 What Constitutes an "Input"?
- 6.1.2 Problem Instances and Instance Distributions
- 6.2 Best-Case Analysis
- **6.2.1** Definition and Purpose
- 6.2.2 Examples: Insertion Sort, Linear Search
- 6.2.3 When Best-Case Matters (and When It Doesn't)
- 6.3 Worst-Case Analysis
- 6.3.1 Definition and Motivation
- 6.3.2 Guarantees and Robustness
- 6.3.3 Examples: Quicksort, Searching in Unsorted Arrays
- 6.3.4 Lower Bounds and Optimality
- 6.4 Average-Case Analysis
- 6.4.1 Definition: Expected Running Time
- 6.4.2 Assumptions About Input Distributions
- 6.4.3 Probabilistic Models: Uniform, Gaussian, etc.
- 6.4.4 Examples: Quicksort, Hashing, Skip Lists

Probabilistic Analysis of Algorithms

7.1	Foundations	of Probabilistic A	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 · 202 Hashing and Universal Hash Functions

Part III Advanced Analysis Techniques

Amortized Analysis

8.1	Introduction to Amortized Analysis
8.1.1	Motivation: Why Average Per-Operation Cost?
8.1.2	Amortized vs. Average-Case Analysis
8.1.3	When to Use Amortized Analysis
8.2	Aggregate Analysis
8.2.1	Definition and Methodology
8.2.2	Example: Dynamic Array (Vector) Resizing
8.2.3	Example: Binary Counter Increment
8.2.4	Example: Stack with Multipop
8.3	The Accounting Method
8.3.1	Conceptual Framework: Credits and Debits
8.3.2	Defining Amortized Costs
8.3.3	Example: Dynamic Array via Accounting
8.3.4	Example: Splay Trees (Introduction)
8.3.5	Ensuring Non-Negative Credit Balance
84	The Potential Method

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 9.3.1	
9.3.1	
9.3.1 9.3.2	Iterative Algorithms: Loops and Arrays
9.3.1 9.3.2 9.3.3	Iterative Algorithms: Loops and Arrays Recursive Algorithms: Mergesort, Quicksort
9.3.1 9.3.2 9.3.3 9.3.4	Iterative Algorithms: Loops and Arrays Recursive Algorithms: Mergesort, Quicksort Dynamic Programming: Memoization vs. Tabulation
9.3.1 9.3.2 9.3.3 9.3.4 9.4	Iterative Algorithms: Loops and Arrays Recursive Algorithms: Mergesort, Quicksort Dynamic Programming: Memoization vs. Tabulation Graph Algorithms: BFS, DFS, Shortest Paths

9:4:3 · 202 Compression and Succinct Data Structures

22|43

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
Externa	l Merge Sort
I/O Con	nplexity: $O((N/B)\log_{M/B}(N/B))$
10.3.2	Matrix Operations
Matrix 7	Transposition
Matrix 1	Multiplication
10.3.3	Graph Algorithms
I/O-Effi	cient BFS and DFS

10.4 Cache-Oblivious Algorithms

Minimum Spanning Tree

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
- 11.4.2.2. Predictability vs. Average-Case Performance

Part IV Lower Bounds and Optimality

Lower Bounds for Comparison-Based Algorithms

12.1	Decision Trees
12.1.1	Modeling Algorithms as Decision Trees
12.1.2	Height of Decision Trees and Worst-Case Complexity
12.2	Sorting Lower Bound
12.2.1	Information-Theoretic Argument
12.2.2	$\Omega(n \log n)$ Lower Bound for Comparison Sorting
12.2.3	Implications and Optimal Algorithms
12.3	Selection and Searching Lower Bounds
12.3.1	Finding the Minimum: $\Omega(n)$
12.3.2	Finding Median: Adversary Arguments
12.3.3	Searching in Sorted Arrays: $\Omega(\log n)$
12.4	Adversary Arguments

Examples: Merging, Element Uniqueness

12.5 Exercises

12.4.1 General Framework

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

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 Multipli-

cation

14.4

Backtracking and Branch-and-Bound

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

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

Parameterized Complexity

17.1	Introduction	to Param	eterized A	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

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
	Parallel and Distributed Algorithm Analysis Scalability and Speedup
18.3.1	
18.3.1 18.3.2	Scalability and Speedup
18.3.1 18.3.2 18.4	Scalability and Speedup Amdahl's Law and Gustafson's Law
18.3.1 18.3.2 18.4 18.4.1	Scalability and Speedup Amdahl's Law and Gustafson's Law Energy Efficiency

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