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

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

Contents

Ti	itle P	age		į
C	onte	nts		iii
P	refac	e		vii
Α	ckno	wledg	jmentsx	xii
I	Fo	unda	ntions	1
1			tical and Algorithmic Prerequisites	3
	1.1		ete Mathematics	4
	1.1	1.1.1	Sets, Functions, and Relations	4
		1.1.2	Combinatorics: Permutations, Combinations, and Binomial Coefficients .	4
		1.1.3	Graph Theory Basics	4
		1.1.4	Proof Techniques: Induction, Contradiction, and Contrapositive	4
	1.2		entary Probability Theory	4
	1.2	1.2.1		4
		1.2.1	Sample Spaces, Events, and Probability Measures	4
		1.2.3	Basic Distributions: Uniform, Bernoulli, Geometric, Binomial	4
		1.2.3	Linearity of Expectation	4
		1.2.5	Conditional Probability and Independence	4
		1.2.6	Variance and Standard Deviation	4
		1.2.7	Moment Generating Functions (Brief Introduction)	4
	1.0			
	1.3		ematical Analysis	4
			Limits, Continuity, and Asymptotic Behavior	4
		1.3.2	Sequences and Series	4
		1.3.3	Summations and Closed Forms	4
		1.3.4	Integration and Differentiation (Brief Review)	4
		1.3.5	Taylor Series and Asymptotic Expansions	4
		1.3.6	Stirling's Approximation	4
	1.4		Algebra (Brief Overview)	4
		1.4.1	Vectors, Matrices, and Linear Transformations	4
		1.4.2	Eigenvalues and Eigenvectors	4
		1.4.3	Applications to Markov Chains and Graph Algorithms	4

		1.4.4	Matrix Operations and Complexity	4
	1.5	Numbe	er Theory Essentials	4
		1.5.1	Divisibility and Modular Arithmetic	4
		1.5.2	Prime Numbers and Factorization	4
		1.5.3	Greatest Common Divisor and Euclidean Algorithm	4
		1.5.4	Applications to Cryptography and Hashing	4
	1.6	Additio	nal Mathematical Tools	4
		1.6.1	Logarithms and Exponentials	4
		1.6.2	Floor and Ceiling Functions	4
		1.6.3	Asymptotic Notation (Informal Preview)	4
	1.7	Self-As	ssessment Exercises	4
		1.7.1	Discrete Mathematics Problems	4
		1.7.2	Probability Problems	4
		1.7.3	Analysis Problems	4
		1.7.4	Linear Algebra Problems	4
		1.7.5	Number Theory Problems	4
	1.8	Furthe	r Reading and Resources	4
		1.8.1	Recommended Textbooks for Each Topic	4
		1.8.2	Online Resources and Video Lectures	4
		1.8.3	Practice Problem Collections	4
ı	Fo	unds	ations of Algorithmic Analysis	5
•				
2	Intro	oducti	on to Algorithm Analysis	6
	2.1	What Is	s Algorithm Analysis?	6
		2.1.1	Correctness vs. Efficiency	6
		2.1.2	Resource Measures: Time, Space, Energy, I/O	6
		2.1.3	The Need for Mathematical Models	6
	2.2	The RA	AM Model of Computation	6
		2.2.1	Basic Operations and Unit-Cost Assumption	6
		2.2.2	Memory Access Model	6
		2.2.3	Limitations and Extensions of the RAM Model	6
	2.3	Measu	ring Algorithm Performance	6
		2.3.1	Input Size and Problem Instances	6
		2.3.2	Counting Basic Operations	6

		2.3.3	Exact vs. Asymptotic Analysis	6
	2.4	Overvi	ew of Complexity Classes	6
		2.4.1	P, NP, NP-Complete, and NP-Hard (Brief Introduction)	6
		2.4.2	Why We Focus on Polynomial-Time Algorithms	6
3	Asy	mptot	ic Notation	7
	3.1	The Ne	eed for Asymptotic Analysis	8
		3.1.1	Why Exact Counts Are Often Impractical	8
		3.1.2	Growth Rates and Scalability	8
	3.2	Big-O	Notation (O)	8
		3.2.1	Formal Definition	8
		3.2.2	Intuition: Upper Bounds	8
		3.2.3	Common Functions and Their Growth Rates	8
		3.2.4	Examples and Non-Examples	8
		3.2.5	Properties of Big-O	8
	3.3	Big-On	nega Notation (Ω)	8
		3.3.1	Formal Definition	8
		3.3.2	Intuition: Lower Bounds	8
		3.3.3	Examples and Applications	8
		3.3.4	Relationship Between O and Ω	8
	3.4	Big-Th	eta Notation (Θ)	8
		3.4.1	Formal Definition	8
		3.4.2	Intuition: Tight Bounds	8
		3.4.3	When to Use Θ vs. O	8
		3.4.4	Examples of Tight Bounds	8
	3.5	Little-o	and Little-omega Notation (o,ω)	8
		3.5.1	Formal Definitions	8
		3.5.2	Strict Asymptotic Bounds	8
		3.5.3	Applications in Analysis	8
	3.6	Comm	on Misconceptions and Pitfalls	8
		3.6.1	Confusing O with Θ	8
		3.6.2	Ignoring Constants in Practice	8
		3.6.3	Misapplying Asymptotic Notation to Small Inputs	8
	3.7	Compa	aring Functions	8

		3.7.1	L'Hôpital's Rule for Limits	8
		3.7.2	Logarithmic vs. Polynomial vs. Exponential Growth	8
		3.7.3	Hierarchy of Common Complexity Classes	8
	3.8	Exercis	ses	8
4	Rec	urren	ce Relations and Their Solutions	9
	4.1	Introdu	uction to Recurrence Relations	10
		4.1.1	What Are Recurrences?	10
		4.1.2	Why They Arise in Algorithm Analysis	10
		4.1.3	Examples from Divide-and-Conquer Algorithms	10
	4.2	The Su	ubstitution Method	10
		4.2.1	Guessing the Solution	10
		4.2.2	Proving by Induction	10
		4.2.3	Examples: Mergesort, Binary Search	10
		4.2.4	Strengthening the Inductive Hypothesis	10
	4.3	The Re	ecursion-Tree Method	10
		4.3.1	Visualizing the Recurrence	10
		4.3.2	Summing Over Levels	10
		4.3.3	Examples and Illustrations	10
		4.3.4	Limitations and When to Use	10
	4.4	The Ma	aster Theorem	10
		4.4.1	Statement of the Master Theorem (Standard Form)	10
		4.4.2		10
		4.4.3	Proof Sketch (Via Recursion Trees)	10
		4.4.4	Examples: $T(n) = aT(n/b) + f(n) \dots \dots \dots \dots$	10
		4.4.5	Regularity Condition and Edge Cases	10
		4.4.6	Extended Master Theorem (Akra-Bazzi)	10
	4.5	The Ak	kra-Bazzi Method	10
		4.5.1	Motivation: Unequal Subproblem Sizes	10
		4.5.2	Statement and Conditions	10
		4.5.3	Examples and Applications	10
		4.5.4	Proof Overview (Advanced)	10
	4.6	Linear	Recurrences with Constant Coefficients	10
		4.6.1	Homogeneous Linear Recurrences	10

		4.6.2	Characteristic Equations
		4.6.3	Solving Fibonacci-Type Recurrences
		4.6.4	Non-Homogeneous Recurrences and Particular Solutions
	4.7	Genera	ating Functions
		4.7.1	Introduction to Generating Functions
		4.7.2	Solving Recurrences with Generating Functions
		4.7.3	Examples: Catalan Numbers, Stirling Numbers
	4.8	Advand	ced Topics
		4.8.1	Full History Recurrences
		4.8.2	Recurrences with Variable Coefficients
		4.8.3	Probabilistic Recurrences (Preview)
	4.9	Exercis	ses 10
5	Bes	t-Case	e, Worst-Case, and Average-Case Analysis 11
	5.1	Definin	ng Input Classes
		5.1.1	What Constitutes an "Input"?
		5.1.2	Problem Instances and Instance Distributions
	5.2	Best-C	ase Analysis
		5.2.1	Definition and Purpose
		5.2.2	Examples: Insertion Sort, Linear Search
		5.2.3	When Best-Case Matters (and When It Doesn't) 12
	5.3	Worst-	Case Analysis
		5.3.1	Definition and Motivation
		5.3.2	Guarantees and Robustness
		5.3.3	Examples: Quicksort, Searching in Unsorted Arrays
		5.3.4	Lower Bounds and Optimality
	5.4	Averag	pe-Case Analysis
		5.4.1	Definition: Expected Running Time
		5.4.2	Assumptions About Input Distributions
		5.4.3	Probabilistic Models: Uniform, Gaussian, etc
		5.4.4	Examples: Quicksort, Hashing, Skip Lists
	5.5	Probab	pilistic Analysis vs. Randomized Algorithms
		5.5.1	Distinction Between the Two Concepts
		5.5.2	Randomized Quicksort: Expected $O(n \log n)$

		5.5.3	Las Vegas vs. Monte Carlo Algorithms	12
	5.6	Smoot	hed Analysis	12
		5.6.1	Motivation: Beyond Worst-Case Pessimism	12
		5.6.2	Introduction to Smoothed Analysis	12
		5.6.3	Case Study: Simplex Algorithm	12
	5.7	Exercis	ses	12
6	Pro	babilis	stic Analysis of Algorithms	13
	6.1	Found	ations of Probabilistic Analysis	14
		6.1.1	Random Variables in Algorithm Analysis	14
		6.1.2	Indicator Random Variables	14
		6.1.3	Linearity of Expectation	14
	6.2	Expect	ted Running Time	14
		6.2.1	Formal Definition	14
		6.2.2	Computing Expectations via Indicator Variables	14
		6.2.3	Examples: Hiring Problem, Randomized Quicksort	14
	6.3	Probab	pilistic Bounds	14
		6.3.1	Markov's Inequality	14
		6.3.2	Chebyshev's Inequality	14
		6.3.3	Chernoff Bounds	14
		6.3.4	Applications to Load Balancing and Hashing	14
	6.4	Rando	mized Algorithms	14
		6.4.1	Randomized Quicksort (Detailed Analysis)	14
		6.4.2	Randomized Selection (Quickselect)	14
		6.4.3	Hashing and Universal Hash Functions	14
		6.4.4	Bloom Filters and Probabilistic Data Structures	14
	6.5	Analys	sis of Randomized Data Structures	14
		6.5.1	Skip Lists	14
		6.5.2	Treaps	14
		6.5.3	Hash Tables with Chaining and Open Addressing	14
	6.6	High-P	Probability Results	14
		6.6.1	What Does "With High Probability" Mean?	14
		6.6.2	Concentration Inequalities	14
		6.6.3	Union Bound and Probabilistic Method	14

	6.7	Exerci	ses	4
Ш	Α	dvan	ced Analysis Techniques 1	5
7	Am	ortize	d Analysis	6
	7.1	Introdu	uction to Amortized Analysis	7
		7.1.1	Motivation: Why Average Per-Operation Cost?	17
		7.1.2	Amortized vs. Average-Case Analysis	17
		7.1.3	When to Use Amortized Analysis	17
	7.2	Aggre	gate Analysis	7
		7.2.1	Definition and Methodology	17
		7.2.2	Example: Dynamic Array (Vector) Resizing	17
		7.2.3	Example: Binary Counter Increment	17
		7.2.4	Example: Stack with Multipop	17
	7.3	The A	ccounting Method	7
		7.3.1	Conceptual Framework: Credits and Debits	17
		7.3.2	Defining Amortized Costs	17
		7.3.3	Example: Dynamic Array via Accounting	17
		7.3.4	Example: Splay Trees (Introduction)	17
		7.3.5	Ensuring Non-Negative Credit Balance	17
	7.4	The Po	otential Method	7
		7.4.1	Potential Functions: Definition and Intuition	17
		7.4.2	Relating Amortized Cost to Actual Cost	17
		7.4.3	Designing Good Potential Functions	17
		7.4.4	Example: Dynamic Array via Potential Method	17
		7.4.5	Example: Binary Counter via Potential Method	17
		7.4.6	Example: Fibonacci Heaps (Overview)	17
	7.5	Comp	aring the Three Methods	7
		7.5.1		17
		7.5.2	When to Choose Which Method	17
		7.5.3	Equivalence of Methods (Informal Discussion)	17
	7.6	Advan	ced Applications	17
		7.6.1		17
		7.6.2	•	17
		7.6.3	•	17

	7.7	Exercis	ses
8	Spa	ce Co	mplexity Analysis 18
	8.1	Introdu	uction to Space Complexity
		8.1.1	Why Space Matters
		8.1.2	Types of Memory: Stack, Heap, Static
		8.1.3	In-Place vs. Out-of-Place Algorithms
	8.2	Measu	ring Space Usage
		8.2.1	Auxiliary Space vs. Total Space
		8.2.2	Recursive Call Stack Depth
		8.2.3	Implicit vs. Explicit Data Structures
	8.3	Examp	oles of Space Complexity Analysis
		8.3.1	Iterative Algorithms: Loops and Arrays
		8.3.2	Recursive Algorithms: Mergesort, Quicksort
		8.3.3	Dynamic Programming: Memoization vs. Tabulation
		8.3.4	Graph Algorithms: BFS, DFS, Shortest Paths
	8.4	Space	-Time Tradeoffs
		8.4.1	Caching and Memoization
		8.4.2	Lookup Tables and Precomputation
		8.4.3	Compression and Succinct Data Structures
	8.5	Stream	ning and Online Algorithms
		8.5.1	Sublinear Space Algorithms
		8.5.2	Sketching and Sampling Techniques
		8.5.3	Examples: Distinct Elements, Heavy Hitters
	8.6	Space	Complexity Classes
		8.6.1	L, NL, PSPACE (Brief Overview)
		8.6.2	Savitch's Theorem
	8.7	Exercis	ses 19
9	Cac	he-Aw	vare and I/O Complexity
	9.1		uction to the Memory Hierarchy
		9.1.1	Registers, Cache (L1, L2, L3), RAM, Disk
		9.1.2	Latency and Bandwidth Characteristics
		9.1.3	Why Algorithm Design Must Consider Memory
	9.2	The Ex	kternal Memory Model (I/O Model)

		9.2.1	Parameters: N (data size), M (memory size), B (block size)	21
		9.2.2	I/O Complexity: Counting Block Transfers	21
		9.2.3	Comparison with RAM Model	21
	9.3	I/O-Effi	cient Algorithms	21
		9.3.1	Scanning and Sorting	21
		9.3.2	Matrix Operations	21
		9.3.3	Graph Algorithms	21
	9.4	Cache-	Oblivious Algorithms	21
		9.4.1	Motivation: Optimal Without Knowing M and B	21
		9.4.2	Cache-Oblivious Sorting (Funnelsort)	21
		9.4.3	Cache-Oblivious Matrix Multiplication	21
		9.4.4	Cache-Oblivious B-Trees (van Emde Boas Layout)	21
	9.5	Cache-	Aware Analysis	21
		9.5.1	Modeling Cache Behavior	21
		9.5.2	Locality of Reference: Temporal and Spatial	21
		9.5.3	Blocking and Tiling Techniques	21
	9.6	Real-W	orld Considerations	21
		9.6.1	Multi-Level Caches	21
		9.6.2	Cache Replacement Policies (LRU, LFU, etc.)	21
		9.6.3	Prefetching and Speculative Execution	21
		9.6.4	False Sharing and Cache Line Effects	21
	9.7	Case S	Studies	21
		9.7.1	Database Query Processing	21
		9.7.2	External Memory Sorting in Practice	21
		9.7.3	Scientific Computing and Large-Scale Simulations	21
	9.8	Exercis	ses	21
10	Cac	he-Aw	vare Scheduling and Analysis for Multicores	22
	10.1	Introdu	ction to Multicore and Parallel Computing	23
		10.1.1	Shared vs. Distributed Memory	23
		10.1.2	Parallel Models: PRAM, Fork-Join, Work-Stealing	23
		10.1.3	Performance Metrics: Work, Span, Parallelism	23
	10.2	Cache	Coherence and Consistency	23
		10.2.1	MESI and MOESI Protocols	23

	11 ?	11.2.3 Selection	Implications and Optimal Algorithms	26 26
		11.2.2	$\Omega(n \log n)$ Lower Bound for Comparison Sorting	26
		11.2.1	Information-Theoretic Argument	26
	11.2	Ū	Lower Bound	26
	11.0	11.1.2	Height of Decision Trees and Worst-Case Complexity	26
		11.1.1	Modeling Algorithms as Decision Trees	26
	11.1		Modeling Algorithms on Decision Trees	26
1				
•			unds for Comparison-Based Algorithms	
V	L	ower	Bounds and Optimality	24
	10.8	Exercis	es	23
		10.7.3	VLDB 2019: Concurrent Hash Tables and Cache Performance	23
		10.7.2	Cache-Aware Scheduling for Multicores (Embedded Systems)	23
		10.7.1	ECRTS 2007: Cache-Aware Real-Time Scheduling	23
	10.7	Case S	Studies from Research	23
		10.6.3	Cache Miss Analysis in Parallel Programs	23
		10.6.2	Brent's Theorem and Greedy Scheduling	23
		10.6.1	DAG-Based Analysis (Cilk Model)	23
	10.6	Analysi	s Techniques	23
		10.5.3	Affinity Scheduling for Cache Locality	23
		10.5.2	Work-Stealing Algorithms	23
		10.5.1	Static vs. Dynamic Scheduling	23
	10.5	Schedu	uling Strategies	23
		10.4.3	Cache Partitioning and Locking	23
		10.4.2	Predictability vs. Average-Case Performance	23
		10.4.1	WCET Analysis in Cache-Aware Contexts	23
	10.4	Real-Ti	me and Embedded Systems	23
		10.3.3	Load Balancing and Task Granularity	23
		10.3.2	Parallel Matrix Multiplication (Strassen, Coppersmith-Winograd)	23
		10.3.1	Parallel Sorting with Cache Awareness	23
	10.3	Cache-	Aware Parallel Algorithms	23
		10.2.3	Impact on Algorithm Design	23
		10.2.2	False Sharing in Multicore Systems	23

		11.3.1	Finding the Minimum: $\Omega(n)$	26
		11.3.2	Finding Median: Adversary Arguments	26
		11.3.3	Searching in Sorted Arrays: $\Omega(\log n)$	26
	11.4	Advers	ary Arguments	26
		11.4.1	General Framework	26
		11.4.2	Examples: Merging, Element Uniqueness	26
	11.5	Exercis	ses	26
12	Alge	ebraic	and Non-Comparison Lower Bounds	27
	12.1	Algebra	aic Decision Trees	27
		12.1.1	Extending Beyond Comparisons	27
		12.1.2	Element Distinctness Lower Bound	27
	12.2	Comm	unication Complexity	27
		12.2.1	Models and Definitions	27
		12.2.2	Applications to Data Structures	27
	12.3	Cell-Pr	obe Model	27
		12.3.1	Lower Bounds for Data Structures	27
		12.3.2	Dynamic vs. Static Data Structures	27
	12.4	Exercis	ses	27
٧	Sp	ecia	lized Topics and Applications	28
13	Ana	lvsis (of Specific Algorithm Paradigms	29
		•	and-Conquer Algorithms	30
	10.1	13.1.1	General Framework and Recurrence Relations	30
		13.1.2	Examples: Mergesort, Quicksort, Strassen's Algorithm	30
		13.1.3	Optimality and Lower Bounds	30
	13.2	Greed	/ Algorithms	30
		13.2.1	Correctness via Exchange Arguments	30
		13.2.2	Matroid Theory (Brief Introduction)	30
		13.2.3	Examples: Huffman Coding, Kruskal's MST	30
	13.3	Dynam	iic Programming	30
		13.3.1	Optimal Substructure and Overlapping Subproblems	30
		13.3.2	Memoization vs. Tabulation	30
		13.3.3	Time and Space Complexity Analysis	30
		10.0.0	Time and opace complexity finally constructions.	

	13.4	Backtra	acking and Branch-and-Bound	30
		13.4.1	Pruning the Search Space	30
		13.4.2	Worst-Case Exponential, Average-Case Better	30
		13.4.3	Examples: N-Queens, Traveling Salesman	30
	13.5	Exercis	ses	30
14	Onli	ine Alç	gorithms and Competitive Analysis	31
	14.1	Introdu	ction to Online Algorithms	31
		14.1.1	Online vs. Offline Problems	31
		14.1.2	Competitive Ratio	31
	14.2	Examp	oles of Online Problems	31
		14.2.1	Paging and Caching (LRU, FIFO, LFU)	31
		14.2.2	Load Balancing	31
		14.2.3	Online Scheduling	31
	14.3	Compe	etitive Analysis Techniques	31
		14.3.1	Deterministic vs. Randomized Algorithms	31
		14.3.2	Lower Bounds via Adversary Arguments	31
	14.4	Exercis	ses	31
15	Арр	roxim	ation Algorithms	32
	15.1	Introdu	oction to Approximation	32
		15.1.1	NP-Hardness and Intractability	32
		15.1.2	Approximation Ratios	32
	15.2	Examp	les of Approximation Algorithms	32
		15.2.1	Vertex Cover (2-Approximation)	32
		15.2.2	Set Cover (Greedy, $\log n$ -Approximation)	32
		15.2.3	Traveling Salesman Problem (Metric TSP)	32
	15.3	Analys	is Techniques	32
		15.3.1	Bounding Optimal Solutions	32
		15.3.2	Linear Programming Relaxations	32
	15.4	Exercis	ses	32
16	Para	amete	rized Complexity	33
	16.1	Introdu	oction to Parameterized Algorithms	33
		16.1.1	Fixed-Parameter Tractability (FPT)	33
		16.1.2	Kernelization	33

	16.2	Examp	les and Analysis	33
		16.2.1	Vertex Cover Parameterized by Solution Size	33
		16.2.2	Treewidth and Graph Algorithms	33
	16.3	W-Hiera	archy and Hardness	33
		16.3.1	W[1], W[2], and Beyond	33
	16.4	Exercis	es	33
VI	Р	ractio	cal Considerations and Case Studies	34
17	Fror	n Theo	ory to Practice	35
			Constants and Lower-Order Terms	35
		17.1.1	When $O(n \log n)$ Beats $O(n)$ in Practice	35
		17.1.2	Empirical Performance Measurements	35
	17.2	Algorith	nm Engineering	35
		17.2.1	Profiling and Benchmarking	35
		17.2.2	Tuning for Specific Hardware	35
		17.2.3	Libraries and Implementations (STL, Boost, etc.)	35
	17.3	Parallel	and Distributed Algorithm Analysis	35
		17.3.1	Scalability and Speedup	35
		17.3.2	Amdahl's Law and Gustafson's Law	35
	17.4	Energy	Efficiency	35
		17.4.1	Energy as a Resource	35
		17.4.2	Green Computing and Mobile Devices	35
	17.5	Exercis	es	35
18	Cas	e Stud	lies	36
	18.1	Sorting	Algorithms in Practice	36
		18.1.1	Timsort, Introsort, Radix Sort	36
		18.1.2	Comparison of Theoretical vs. Empirical Performance	36
	18.2	Graph A	Algorithms in Large-Scale Systems	36
		18.2.1	Web Graphs and PageRank	36
		18.2.2	Social Network Analysis	36
	18.3	Machin	e Learning and Data Science	36
		18.3.1	Complexity of Training Algorithms	36
		18.3.2	SGD, AdaGrad, Adam: Time and Space Analysis	36

	18.4	Database Systems	36
		18.4.1 Query Optimization	36
		18.4.2 Indexing Structures (B-Trees, LSM-Trees)	36
	18.5	Exercises	36
Α	Mat	hematical Background	37
	A.1	Summation Formulas	37
	A.2	Logarithms and Exponentials	37
	A.3	Recurrence Relations (Quick Reference)	37
	A.4	Probability Distributions	37
	A.5	Matrix Operations	37
В	Pse	udocode Conventions	38
	B.1	Notation and Style	38
	B.2	Common Data Structures	38
С	Solu	utions to Selected Exercises	39
D	Glos	ssary of Terms	40
Ε	Inde	ex of Algorithms	41
F	Ann	otated Bibliography	42
	F.1	Foundational Texts	42
	F.2	Research Papers by Topic	42
	F3	Online Courses and Resources	42

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

HE FOUNDATION of algorithmic analysis rests upon a bedrock of mathematical rigor and computational intuition. Before embarking on the journey of analyzing algorithms asymptotically, understanding recurrences, or diving into amortized analysis, we must first ensure our mathematical toolkit is complete and our understanding of fundamental concepts is solid.

This part serves as both a review and a deep dive into the essential mathematical and algorithmic prerequisites that underpin everything that follows. Whether you are a student encountering these topics for the first time or a practitioner seeking to refresh your knowledge, this foundation will prepare you for the sophisticated analysis techniques explored in subsequent parts.

"In mathematics, you don't understand things. You just get used to them."

— JOHN VON NEUMANN

First Edition • 2025

Mathematical and Algorithmic Prerequisites

1.1 Discrete Mathematics

1.1.1 Sets, Functions, and Relations

Set Theory Basics

Operations on Sets

Functions: Injective, Surjective, Bijective

Relations and Equivalence Classes

Order Relations and Partial Orders

1.1.2 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

1.1.3 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

2.1	What Is Algorithm Analysis?		
2.1.1	Correctness vs. Efficiency		
2.1.2	Resource Measures: Time, Space, Energy, I/O		
2.1.3	The Need for Mathematical Models		
2.2	The RAM Model of Computation		
2.2.1	Basic Operations and Unit-Cost Assumption		
2.2.2	Memory Access Model		
2.2.3	Limitations and Extensions of the RAM Model		
2.3	Measuring Algorithm Performance		
2.3.1	Input Size and Problem Instances		
2.3.2	Counting Basic Operations		
2.3.3	Exact vs. Asymptotic Analysis		
2.4	Overview of Complexity Classes		
2.4.1	P. NP. NP-Complete, and NP-Hard (Brief Introduct		

2.4.2 Why We Focus on Polynomial-Time Algorithms

Asymptotic Notation

3.1 The Need for Asymptotic Analysis	3.1	The Need	for Asym	ptotic Ana	ılysis
--------------------------------------	-----	----------	----------	------------	--------

- 3.1.1 Why Exact Counts Are Often Impractical
- 3.1.2 Growth Rates and Scalability
- 3.2 Big-O Notation (O)
- 3.2.1 Formal Definition
- 3.2.2 Intuition: Upper Bounds
- 3.2.3 Common Functions and Their Growth Rates
- 3.2.4 Examples and Non-Examples
- 3.2.5 Properties of Big-O

Transitivity

Addition and Multiplication Rules

Reflexivity and Asymmetry

3.3 Big-Omega Notation (Ω)

- 3.3.1 Formal Definition
- 3.3.2 Intuition: Lower Bounds
- 3.3.3 Examples and Applications
- 3.3.4 Relationship Between O and Ω

3.4 Big-Theta Notation (Θ)

Recurrence Relations and Their Solutions

4.1	Introduction	to Recurrence	Relations
T. T	IIIIIUuucuuli	to recurrence	inclauons

- 4.1.1 What Are Recurrences?
- 4.1.2 Why They Arise in Algorithm Analysis
- 4.1.3 Examples from Divide-and-Conquer Algorithms

4.2 The Substitution Method

- 4.2.1 Guessing the Solution
- 4.2.2 Proving by Induction
- 4.2.3 Examples: Mergesort, Binary Search
- 4.2.4 Strengthening the Inductive Hypothesis

4.3 The Recursion-Tree Method

- 4.3.1 Visualizing the Recurrence
- 4.3.2 Summing Over Levels
- 4.3.3 Examples and Illustrations
- 4.3.4 Limitations and When to Use

4.4 The Master Theorem

4.4.1 Statement of the Master Theorem (Standard Form)

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

5.1 l	Defini	ng Inpu	ıt Classes
-------	--------	---------	------------

- 5.1.1 What Constitutes an "Input"?
- 5.1.2 Problem Instances and Instance Distributions
- 5.2 Best-Case Analysis
- 5.2.1 Definition and Purpose
- 5.2.2 Examples: Insertion Sort, Linear Search
- 5.2.3 When Best-Case Matters (and When It Doesn't)
- 5.3 Worst-Case Analysis
- 5.3.1 Definition and Motivation
- 5.3.2 Guarantees and Robustness
- 5.3.3 Examples: Quicksort, Searching in Unsorted Arrays
- 5.3.4 Lower Bounds and Optimality
- 5.4 Average-Case Analysis
- 5.4.1 Definition: Expected Running Time
- 5.4.2 Assumptions About Input Distributions
- 5.4.3 Probabilistic Models: Uniform, Gaussian, etc.
- 5.4.4 Examples: Quicksort, Hashing, Skip Lists

Probabilistic Analysis of Algorithms

6.1	Foundations of Probabilistic Analysis
6.1.1	Random Variables in Algorithm Analysis
6.1.2	Indicator Random Variables
6.1.3	Linearity of Expectation
6.2	Expected Running Time
6.2.1	Formal Definition
6.2.2	Computing Expectations via Indicator Variables
6.2.3	Examples: Hiring Problem, Randomized Quicksort
6.3	Probabilistic Bounds
6.3.1	Markov's Inequality
6.3.2	Chebyshev's Inequality
	Chebyshev's inequality
6.3.3	Chernoff Bounds
	Chernoff Bounds Applications to Load Balancing and Hashing
6.3.4 6.4	Chernoff Bounds Applications to Load Balancing and Hashing
6.3.46.46.4.1	Chernoff Bounds Applications to Load Balancing and Hashing Randomized Algorithms

6.4.4 Bloom Filters and Probabilistic Data Structures

Part III Advanced Analysis Techniques

Amortized Analysis

- 7.1.1 Motivation: Why Average Per-Operation Cost?
- 7.1.2 Amortized vs. Average-Case Analysis
- 7.1.3 When to Use Amortized Analysis
- 7.2 Aggregate Analysis
- 7.2.1 Definition and Methodology
- 7.2.2 Example: Dynamic Array (Vector) Resizing
- 7.2.3 Example: Binary Counter Increment
- 7.2.4 Example: Stack with Multipop

7.3 The Accounting Method

- 7.3.1 Conceptual Framework: Credits and Debits
- 7.3.2 Defining Amortized Costs
- 7.3.3 Example: Dynamic Array via Accounting
- 7.3.4 Example: Splay Trees (Introduction)
- 7.3.5 Ensuring Non-Negative Credit Balance

7.4 The Potential Method

Space Complexity Analysis

8.1	Introduction to Space Complexity
8.1.1	Why Space Matters
8.1.2	Types of Memory: Stack, Heap, Static
8.1.3	In-Place vs. Out-of-Place Algorithms
8.2	Measuring Space Usage
8.2.1	Auxiliary Space vs. Total Space
8.2.2	Recursive Call Stack Depth
8.2.3	Implicit vs. Explicit Data Structures
8.3	Examples of Space Complexity Analysis
8.3.1	Iterative Algorithms: Loops and Arrays
8.3.2	Recursive Algorithms: Mergesort, Quicksort
8.3.3	Dynamic Programming: Memoization vs. Tabulation
8.3.4	Graph Algorithms: BFS, DFS, Shortest Paths
8.4	Space-Time Tradeoffs
8.4.1	Caching and Memoization
8.4.2	Lookup Tables and Precomputation
8:14 dit 3 :	202Compression and Succinct Data Structures

Cache-Aware and I/O Complexity

9.	1	Introduction	to	the	Memory	Hierarch	V

- 9.1.1 Registers, Cache (L1, L2, L3), RAM, Disk
- 9.1.2 Latency and Bandwidth Characteristics
- 9.1.3 Why Algorithm Design Must Consider Memory
- 9.2 The External Memory Model (I/O Model)
- 9.2.1 Parameters: N (data size), M (memory size), B (block size)
- 9.2.2 I/O Complexity: Counting Block Transfers
- 9.2.3 Comparison with RAM Model

9.3 I/O-Efficient Algorithms

9.3.1 Scanning and Sorting

External Merge Sort

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

9.3.2 Matrix Operations

Matrix Transposition

Matrix Multiplication

9.3.3 Graph Algorithms

I/O-Efficient BFS and DFS

Minimum Spanning Tree

21 | 42

9.4 Cache-Oblivious Algorithms

Cache-Aware Scheduling and Analysis for Multicores

10.1	Introduction to Multicore and Parallel Computing
10.1.1	Shared vs. Distributed Memory
10.1.2	Parallel Models: PRAM, Fork-Join, Work-Stealing
10.1.3	Performance Metrics: Work, Span, Parallelism
10.2	Cache Coherence and Consistency
10.2.1	MESI and MOESI Protocols
10.2.2	False Sharing in Multicore Systems
10.2.3	Impact on Algorithm Design
10.3	Cache-Aware Parallel Algorithms
10.3.1	Parallel Sorting with Cache Awareness
10.3.2	Parallel Matrix Multiplication (Strassen, Coppersmith-Winograd)
10.3.3	Load Balancing and Task Granularity
10.4	Real-Time and Embedded Systems
10.4.1	WCET Analysis in Cache-Aware Contexts

Cache Partitioning and Locking

Part IV Lower Bounds and Optimality

Lower Bounds for Comparison-Based Algorithms

44	4	\mathbf{r}	•	•			
11.	1	1)	111	110	n	Tre	Δc
\mathbf{L}	• L	$\mathbf{p}_{\mathbf{c}}$			111	116	C 2

- 11.1.1 Modeling Algorithms as Decision Trees
- 11.1.2 Height of Decision Trees and Worst-Case Complexity

11.2 Sorting Lower Bound

- 11.2.1 Information-Theoretic Argument
- 11.2.2 $\Omega(n \log n)$ Lower Bound for Comparison Sorting
- 11.2.3 Implications and Optimal Algorithms

11.3 Selection and Searching Lower Bounds

- **11.3.1** Finding the Minimum: $\Omega(n)$
- 11.3.2 Finding Median: Adversary Arguments
- **11.3.3** Searching in Sorted Arrays: $\Omega(\log n)$

11.4 Adversary Arguments

- 11.4.1 General Framework
- 11.4.2 Examples: Merging, Element Uniqueness

11.5 Exercises

Algebraic and Non-Comparison Lower Bounds

	1119001410 2 00101011 11000
12.1.1	Extending Beyond Comparisons
12.1.2	Element Distinctness Lower Bound
12.2	Communication Complexity
12.2.1	Models and Definitions

12.1 Algebraic Decision Trees

- 12.3 Cell-Probe Model
- 12.3.1 Lower Bounds for Data Structures

12.2.2 Applications to Data Structures

- 12.3.2 Dynamic vs. Static Data Structures
- 12.4 Exercises

Part V

Specialized Topics and Applications

Analysis of Specific Algorithm Paradigms

13.1	Divide-and-Conquer Algorithms
13.1.1	General Framework and Recurrence Relations
13.1.2	Examples: Mergesort, Quicksort, Strassen's Algorithm
13.1.3	Optimality and Lower Bounds
13.2	Greedy Algorithms
13.2.1	Correctness via Exchange Arguments
13.2.2	Matroid Theory (Brief Introduction)
13.2.3	Examples: Huffman Coding, Kruskal's MST
13.3	Dynamic Programming
13.3.1	Optimal Substructure and Overlapping Subproblems
13.3.2	Memoization vs. Tabulation
13.3.3	Time and Space Complexity Analysis
13.3.4	Examples: Knapsack, Edit Distance, Matrix Chain Multiplication
13.4	Backtracking and Branch-and-Bound

Online Algorithms and Competitive Analysis

14.1	Introduction to Online Algorithms
14.1.1	Online vs. Offline Problems
14.1.2	Competitive Ratio
14.2	Examples of Online Problems
14.2.1	Paging and Caching (LRU, FIFO, LFU)
14.2.2	Load Balancing
14.2.3	Online Scheduling
14.3	Competitive Analysis Techniques
14.3.1	Deterministic vs. Randomized Algorithms
14.3.2	Lower Bounds via Adversary Arguments
14.4	Exercises

Approximation Algorithms

15.1	Introduction to Approximation
15.1.1	NP-Hardness and Intractability
15.1.2	Approximation Ratios
15.2	Examples of Approximation Algorithms
15.2.1	Vertex Cover (2-Approximation)
15.2.2	Set Cover (Greedy, $\log n$ -Approximation)
15.2.3	Traveling Salesman Problem (Metric TSP)
15.3	Analysis Techniques
15.3.1	Bounding Optimal Solutions
15.3.2	Linear Programming Relaxations
15.4	Exercises

Parameterized Complexity

16.1	Introduction to Parameterized Algorithms
16.1.1	Fixed-Parameter Tractability (FPT)

- 16.1.2 Kernelization
- 16.2 Examples and Analysis
- 16.2.1 Vertex Cover Parameterized by Solution Size
- 16.2.2 Treewidth and Graph Algorithms
- 16.3 W-Hierarchy and Hardness
- 16.3.1 W[1], W[2], and Beyond
- 16.4 Exercises

Part VI

Practical Considerations and Case Studies

From Theory to Practice

17.1	Hidden Constants and Lower-Order Terms
17.1.1	When $O(n \log n)$ Beats $O(n)$ in Practice
17.1.2	Empirical Performance Measurements
17.2	Algorithm Engineering
17.2.1	Profiling and Benchmarking
17.2.2	Tuning for Specific Hardware
17.2.3	Libraries and Implementations (STL, Boost, etc.)
17.3	Parallel and Distributed Algorithm Analysis
	Parallel and Distributed Algorithm Analysis Scalability and Speedup
17.3.1	
17.3.1 17.3.2	Scalability and Speedup
17.3.1 17.3.2 17.4	Scalability and Speedup Amdahl's Law and Gustafson's Law
17.3.1 17.3.2 17.4 17.4.1	Scalability and Speedup Amdahl's Law and Gustafson's Law Energy Efficiency

Case Studies

18.1	Sorting Algorithms in Practice
18.1.1	Timsort, Introsort, Radix Sort
18.1.2	Comparison of Theoretical vs. Empirical Performance
18.2	Graph Algorithms in Large-Scale Systems
18.2.1	Web Graphs and PageRank
18.2.2	Social Network Analysis
18.3	Machine Learning and Data Science
18.3.1	Complexity of Training Algorithms
18.3.2	SGD, AdaGrad, Adam: Time and Space Analysis
18.4	Database Systems
18.4.1	Query Optimization
18.4.2	Indexing Structures (B-Trees, LSM-Trees)
18.5	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