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		i
C	onte	nts		iii
P	refac	e		/ii
			mentsxx	
I	Fo	unda	itions	1
1	Mat	hema	tical and Algorithmic Prerequisites	3
	1.1	Discre	te Mathematics: The Language of Algorithms	4
		1.1.1	Sets, Functions, and Relations	4
		1.1.2	Combinatorics: Counting and Arrangements	4
		1.1.3	Graph Theory Essentials	4
		1.1.4	Proof Techniques: The Art of Rigorous Reasoning	5
	1.2	Proba	bility Theory for Algorithm Analysis	5
		1.2.1	Foundations: Sample Spaces and Events	6
		1.2.2	Random Variables and Expectations	6
		1.2.3	Common Distributions in Algorithms	6
		1.2.4	Linearity of Expectation: The Most Powerful Tool	6
		1.2.5	Independence and Conditional Expectation	7
		1.2.6	Variance and Concentration	7
		1.2.7	Moment Generating Functions (Brief Preview)	7
	1.3	Mathe	matical Analysis: Limits and Asymptotics	7
		1.3.1	Limits and Continuity	8
		1.3.2	Sequences and Series	8
		1.3.3	Summations and Closed Forms	8
		1.3.4	Integration and Differentiation	9
		1.3.5	Taylor Series and Asymptotic Expansions	9
		1.3.6	Stirling's Approximation	9
	1.4	Linear	Algebra for Algorithmic Applications	10
		1.4.1	Vectors, Matrices, and Linear Systems	10
		1.4.2	Eigenvalues and Eigenvectors	10
		1 / 2	Markov Chains and Random Walks	10

		1.4.4	Matrix Operations and Complexity	11
	1.5	Numbe	er Theory Essentials	11
		1.5.1	Divisibility and Modular Arithmetic	12
		1.5.2	Prime Numbers and Factorization	12
		1.5.3	Greatest Common Divisor and Euclidean Algorithm	12
		1.5.4	Applications to Cryptography and Hashing	13
	1.6	Additio	onal Mathematical Tools	13
		1.6.1	Logarithms and Exponentials	13
		1.6.2	Floor and Ceiling Functions	13
		1.6.3	Asymptotic Notation: Informal Preview	13
	1.7	Self-As	ssessment and Further Study	14
		1.7.1	Self-Assessment Exercises	14
		1.7.2	Further Reading and Resources	14
		1.7.3	Notation and Conventions Used in This Book	14
П	Fo	ounda	ations of Algorithmic Analysis	15
2			on to Algorithm Analysis	
_				
	2.1		s Algorithm Analysis?	16
		2.1.1	Correctness vs. Efficiency	16
		2.1.2	Resource Measures: Time, Space, Energy, I/O	16 16
	0.0			
	2.2		AM Model of Computation	16
		2.2.1	Basic Operations and Unit-Cost Assumption	16
		2.2.2	Memory Access Model	16
	0.0	2.2.3	Limitations and Extensions of the RAM Model	16
	2.3		ring Algorithm Performance	16
		2.3.1	Input Size and Problem Instances	16
		2.3.2	Counting Basic Operations	16
	0.4	2.3.3	Exact vs. Asymptotic Analysis	16
	2.4		ew of Complexity Classes	16
		2.4.1	P, NP, NP-Complete, and NP-Hard (Brief Introduction)	16
_	_	2.4.2	Why We Focus on Polynomial-Time Algorithms	16
3	Asy	mptot	ic Notation	17
	3.1	The Ne	eed for Asymptotic Analysis	18

	4 1	Introdu	iction to Recurrence Relations	20
4	Rec	urren	ce Relations and Their Solutions	19
	3.8	Exercis	ses	18
		3.7.3	Hierarchy of Common Complexity Classes	18
		3.7.2	Logarithmic vs. Polynomial vs. Exponential Growth	18
		3.7.1	L'Hôpital's Rule for Limits	18
	3.7	Compa	aring Functions	18
		3.6.3	Misapplying Asymptotic Notation to Small Inputs	18
		3.6.2	Ignoring Constants in Practice	18
		3.6.1	Confusing O with Θ	18
	3.6	Comm	on Misconceptions and Pitfalls	18
		3.5.3	Applications in Analysis	18
		3.5.2	Strict Asymptotic Bounds	18
		3.5.1	Formal Definitions	18
	3.5	Little-o	and Little-omega Notation (o,ω)	18
		3.4.4	Examples of Tight Bounds	18
		3.4.3	When to Use Θ vs. O	18
		3.4.2	Intuition: Tight Bounds	18
		3.4.1	Formal Definition	18
	3.4	Big-Th	eta Notation (Θ)	18
		3.3.4	Relationship Between O and Ω	18
		3.3.3	Examples and Applications	18
		3.3.2	Intuition: Lower Bounds	18
	0.0	3.3.1	Formal Definition	18
	3.3		nega Notation (Ω)	18
		3.2.5	Properties of Big-O	18
		3.2.4	Examples and Non-Examples	18
		3.2.3	Common Functions and Their Growth Rates	18
		3.2.1	Intuition: Upper Bounds	18
	3.2	3.2.1	Formal Definition	18
	3.2		Notation (O)	18
		3.1.2	Growth Rates and Scalability	18
		3.1.1	Why Exact Counts Are Often Impractical	18

	4.1.1	What Are Recurrences?	20
	4.1.2	Why They Arise in Algorithm Analysis	20
	4.1.3	Examples from Divide-and-Conquer Algorithms	20
4.2	The Su	ubstitution Method	20
	4.2.1	Guessing the Solution	20
	4.2.2	Proving by Induction	20
	4.2.3	Examples: Mergesort, Binary Search	20
	4.2.4	Strengthening the Inductive Hypothesis	20
4.3	The Re	ecursion-Tree Method	20
	4.3.1	Visualizing the Recurrence	20
	4.3.2	Summing Over Levels	20
	4.3.3	Examples and Illustrations	20
	4.3.4	Limitations and When to Use	20
4.4	The M	aster Theorem	20
	4.4.1	Statement of the Master Theorem (Standard Form)	20
	4.4.2	Three Cases and Their Intuition	20
	4.4.3	Proof Sketch (Via Recursion Trees)	20
	4.4.4	Examples: $T(n) = aT(n/b) + f(n) \dots \dots \dots \dots \dots \dots$	20
	4.4.5	Regularity Condition and Edge Cases	20
	4.4.6	Extended Master Theorem (Akra-Bazzi)	20
4.5	The Al	kra-Bazzi Method	20
	4.5.1	Motivation: Unequal Subproblem Sizes	20
	4.5.2	Statement and Conditions	20
	4.5.3	Examples and Applications	20
	4.5.4	Proof Overview (Advanced)	20
4.6	Linear	Recurrences with Constant Coefficients	20
	4.6.1	Homogeneous Linear Recurrences	20
	4.6.2	Characteristic Equations	20
	4.6.3	Solving Fibonacci-Type Recurrences	20
	4.6.4	Non-Homogeneous Recurrences and Particular Solutions	20
4.7	Genera	ating Functions	20
	4.7.1	Introduction to Generating Functions	20
	4.7.2	Solving Recurrences with Generating Functions	20
	4.7.3	Examples: Catalan Numbers, Stirling Numbers	20

6	Pro	babilis	stic Analysis of Algorithms23
	5.7	Exercis	ses
		5.6.3	Case Study: Simplex Algorithm
		5.6.2	Introduction to Smoothed Analysis
		5.6.1	Motivation: Beyond Worst-Case Pessimism
	5.6	Smoot	hed Analysis
		5.5.3	Las Vegas vs. Monte Carlo Algorithms
		5.5.2	Randomized Quicksort: Expected $O(n \log n)$
		5.5.1	Distinction Between the Two Concepts
	5.5	Probab	pilistic Analysis vs. Randomized Algorithms
		5.4.4	Examples: Quicksort, Hashing, Skip Lists
		5.4.3	Probabilistic Models: Uniform, Gaussian, etc
		5.4.2	Assumptions About Input Distributions
		5.4.1	Definition: Expected Running Time
	5.4	Averag	e-Case Analysis
		5.3.4	Lower Bounds and Optimality
		5.3.3	Examples: Quicksort, Searching in Unsorted Arrays
		5.3.2	Guarantees and Robustness
		5.3.1	Definition and Motivation
	5.3	Worst-	Case Analysis
		5.2.3	When Best-Case Matters (and When It Doesn't)
		5.2.2	Examples: Insertion Sort, Linear Search
		5.2.1	Definition and Purpose
	5.2	Best-C	ase Analysis
		5.1.2	Problem Instances and Instance Distributions
		5.1.1	What Constitutes an "Input"?
	5.1	Definin	ng Input Classes
5	Bes	t-Case	e, Worst-Case, and Average-Case Analysis 21
	4.9	Exercis	ses
		4.8.3	Probabilistic Recurrences (Preview)
		4.8.2	Recurrences with Variable Coefficients
		4.8.1	Full History Recurrences
	4.8	Advand	ced Topics

	6.1	Founda	ations of Probabilistic Analysis	24
		6.1.1	Random Variables in Algorithm Analysis	24
		6.1.2	Indicator Random Variables	24
		6.1.3	Linearity of Expectation	24
	6.2	Expect	ed Running Time	24
		6.2.1	Formal Definition	24
		6.2.2	Computing Expectations via Indicator Variables	24
		6.2.3	Examples: Hiring Problem, Randomized Quicksort	24
	6.3	Probab	pilistic Bounds	24
		6.3.1	Markov's Inequality	24
		6.3.2	Chebyshev's Inequality	24
		6.3.3	Chernoff Bounds	24
		6.3.4	Applications to Load Balancing and Hashing	24
	6.4	Rando	mized Algorithms	24
		6.4.1	Randomized Quicksort (Detailed Analysis)	24
		6.4.2	Randomized Selection (Quickselect)	24
		6.4.3	Hashing and Universal Hash Functions	24
		6.4.4	Bloom Filters and Probabilistic Data Structures	24
	6.5	Analys	is of Randomized Data Structures	24
		6.5.1	Skip Lists	24
		6.5.2	Treaps	24
		6.5.3	Hash Tables with Chaining and Open Addressing	24
	6.6	High-P	robability Results	24
		6.6.1	What Does "With High Probability" Mean?	24
		6.6.2	Concentration Inequalities	24
		6.6.3	Union Bound and Probabilistic Method	24
	6.7	Exercis	Ses	24
	٨	dvon	and Analysis Tashniques	25
III	A	uvan	ced Analysis Techniques	25
7	Am	ortized	d Analysis	26
	7.1	Introdu	oction to Amortized Analysis	27
		7.1.1	Motivation: Why Average Per-Operation Cost?	27
		7.1.2	Amortized vs. Average-Case Analysis	27
		7.1.3	When to Use Amortized Analysis	27

	7.2	Aggre	gate Analysis	27
		7.2.1	Definition and Methodology	27
		7.2.2	Example: Dynamic Array (Vector) Resizing	27
		7.2.3	Example: Binary Counter Increment	27
		7.2.4	Example: Stack with Multipop	27
	7.3	The Ad	counting Method	27
		7.3.1	Conceptual Framework: Credits and Debits	27
		7.3.2	Defining Amortized Costs	27
		7.3.3	Example: Dynamic Array via Accounting	27
		7.3.4	Example: Splay Trees (Introduction)	27
		7.3.5	Ensuring Non-Negative Credit Balance	27
	7.4	The Po	otential Method	27
		7.4.1	Potential Functions: Definition and Intuition	27
		7.4.2	Relating Amortized Cost to Actual Cost	27
		7.4.3	Designing Good Potential Functions	27
		7.4.4	Example: Dynamic Array via Potential Method	27
		7.4.5	Example: Binary Counter via Potential Method	27
		7.4.6	Example: Fibonacci Heaps (Overview)	27
	7.5	Compa	aring the Three Methods	27
		7.5.1	Strengths and Weaknesses	27
		7.5.2	When to Choose Which Method	27
		7.5.3	Equivalence of Methods (Informal Discussion)	27
	7.6	Advan	ced Applications	27
		7.6.1	Splay Trees: Full Analysis	27
		7.6.2	Fibonacci Heaps	27
		7.6.3	Disjoint-Set Union (Union-Find)	27
	7.7	Exercis	ses	27
8	Spa	ice Co	mplexity Analysis	28
	8.1	Introdu	uction to Space Complexity	29
		8.1.1	Why Space Matters	29
		8.1.2	Types of Memory: Stack, Heap, Static	29
		8.1.3	In-Place vs. Out-of-Place Algorithms	29
	8.2	Measu	ring Space Usage	29

		8.2.1	Auxiliary Space vs. Total Space	29
		8.2.2	Recursive Call Stack Depth	29
		8.2.3	Implicit vs. Explicit Data Structures	29
	8.3	Examp	oles of Space Complexity Analysis	29
		8.3.1	Iterative Algorithms: Loops and Arrays	29
		8.3.2	Recursive Algorithms: Mergesort, Quicksort	29
		8.3.3	Dynamic Programming: Memoization vs. Tabulation	29
		8.3.4	Graph Algorithms: BFS, DFS, Shortest Paths	29
	8.4	Space-	-Time Tradeoffs	29
		8.4.1	Caching and Memoization	29
		8.4.2	Lookup Tables and Precomputation	29
		8.4.3	Compression and Succinct Data Structures	29
	8.5	Stream	ning and Online Algorithms	29
		8.5.1	Sublinear Space Algorithms	29
		8.5.2	Sketching and Sampling Techniques	29
		8.5.3	Examples: Distinct Elements, Heavy Hitters	29
	8.6	Space	Complexity Classes	29
		8.6.1	L, NL, PSPACE (Brief Overview)	29
		8.6.2	Savitch's Theorem	29
	8.7	Exercis	ses	29
9	Cac	he-Aw	vare and I/O Complexity	30
	9.1		uction to the Memory Hierarchy	31
	•	9.1.1	Registers, Cache (L1, L2, L3), RAM, Disk	31
		9.1.2	Latency and Bandwidth Characteristics	31
		9.1.3	Why Algorithm Design Must Consider Memory	31
	9.2	The Fx	kternal Memory Model (I/O Model)	31
	0.2	9.2.1	Parameters: N (data size), M (memory size), B (block size)	31
		9.2.2	I/O Complexity: Counting Block Transfers	31
		9.2.3	Comparison with RAM Model	31
	9.3		icient Algorithms	31
	3.5	9.3.1	Scanning and Sorting	31
		9.3.2	Matrix Operations	31
		9.3.3	Graph Algorithms	31
		5.5.5		0 1

9.4	Cache	-Oblivious Algorithms	31
	9.4.1	Motivation: Optimal Without Knowing M and B	31
	9.4.2	Cache-Oblivious Sorting (Funnelsort)	31
	9.4.3	Cache-Oblivious Matrix Multiplication	31
	9.4.4	Cache-Oblivious B-Trees (van Emde Boas Layout)	31
9.5	Cache	-Aware Analysis	31
	9.5.1	Modeling Cache Behavior	31
	9.5.2	Locality of Reference: Temporal and Spatial	31
	9.5.3	Blocking and Tiling Techniques	31
9.6	Real-V	Vorld Considerations	31
	9.6.1	Multi-Level Caches	31
	9.6.2	Cache Replacement Policies (LRU, LFU, etc.)	31
	9.6.3	Prefetching and Speculative Execution	31
	9.6.4	False Sharing and Cache Line Effects	31
9.7	Case S	Studies	31
	9.7.1	Database Query Processing	31
	9.7.2	External Memory Sorting in Practice	31
	9.7.3	Scientific Computing and Large-Scale Simulations	31
9.8	Exerci	ses	31
10 Ca	che-Av	vare Scheduling and Analysis for Multicores	32
10.	1 Introdu	uction to Multicore and Parallel Computing	33
	10.1.1	Shared vs. Distributed Memory	33
	10.1.2	Parallel Models: PRAM, Fork-Join, Work-Stealing	33
	10.1.3	Performance Metrics: Work, Span, Parallelism	33
10.	2 Cache	Coherence and Consistency	33
	10.2.1	MESI and MOESI Protocols	33
	10.2.2	False Sharing in Multicore Systems	33
	10.2.3	Impact on Algorithm Design	33
10.	3 Cache	-Aware Parallel Algorithms	33
	10.3.1	Parallel Sorting with Cache Awareness	33
	10.3.2	Parallel Matrix Multiplication (Strassen, Coppersmith-Winograd)	33
	10.3.3	Load Balancing and Task Granularity	33
40	4 Daal T	ime and Embedded Systems	33

		10.4.1	WCET Analysis in Cache-Aware Contexts	33
		10.4.2	Predictability vs. Average-Case Performance	33
		10.4.3	Cache Partitioning and Locking	33
	10.5	Schedu	ıling Strategies	33
		10.5.1	Static vs. Dynamic Scheduling	33
		10.5.2	Work-Stealing Algorithms	33
		10.5.3	Affinity Scheduling for Cache Locality	33
	10.6	Analysi	s Techniques	33
		10.6.1	DAG-Based Analysis (Cilk Model)	33
		10.6.2	Brent's Theorem and Greedy Scheduling	33
		10.6.3	Cache Miss Analysis in Parallel Programs	33
	10.7	Case S	tudies from Research	33
		10.7.1	ECRTS 2007: Cache-Aware Real-Time Scheduling	33
		10.7.2	Cache-Aware Scheduling for Multicores (Embedded Systems)	33
		10.7.3	VLDB 2019: Concurrent Hash Tables and Cache Performance	33
	10.8	Exercis	es	33
IV	1.	ower	Bounds and Ontimality	34
IV			,	34
_	Low	er Bo	unds for Comparison-Based Algorithms	35
_	Low	er Bo	unds for Comparison-Based Algorithms	35
_	Low	er Bo	unds for Comparison-Based Algorithms In Trees	35
_	Low	ver Bo	unds for Comparison-Based Algorithms	35
_	Low 11.1	Decision 11.1.1.1	unds for Comparison-Based Algorithms In Trees	35 36
_	Low 11.1	Decision 11.1.1.1	unds for Comparison-Based Algorithms on Trees Modeling Algorithms as Decision Trees Height of Decision Trees and Worst-Case Complexity	35 36 36
_	Low 11.1	Decision 11.1.1 11.1.2 Sorting	unds for Comparison-Based Algorithms In Trees Modeling Algorithms as Decision Trees Height of Decision Trees and Worst-Case Complexity Lower Bound	35 36 36 36
_	Low 11.1	Decision 11.1.1 11.1.2 Sorting 11.2.1	unds for Comparison-Based Algorithms In Trees Modeling Algorithms as Decision Trees Height of Decision Trees and Worst-Case Complexity Lower Bound Information-Theoretic Argument	35 36 36 36
_	Low 11.1	Decision 11.1.1 11.1.2 Sorting 11.2.1 11.2.2 11.2.3	unds for Comparison-Based Algorithms In 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	35 36 36 36 36
_	Low 11.1	Decision 11.1.1 11.1.2 Sorting 11.2.1 11.2.2 11.2.3	unds for Comparison-Based Algorithms In 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	35 36 36 36 36 36 36
_	Low 11.1	Decision 11.1.1 11.1.2 Sorting 11.2.1 11.2.2 11.2.3 Selection	unds for Comparison-Based Algorithms In 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	35 36 36 36 36 36 36 36
_	Low 11.1	Decision 11.1.1 11.1.2 Sorting 11.2.1 11.2.2 11.2.3 Selection 11.3.1	unds for Comparison-Based Algorithms In 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: $\Omega(n)$	35 36 36 36 36 36 36 36
_	11.1 11.2	Decision 11.1.1 11.1.2 Sorting 11.2.1 11.2.2 11.2.3 Selection 11.3.1 11.3.2 11.3.3	unds for Comparison-Based Algorithms In 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	35 36 36 36 36 36 36 36 36 36
_	11.1 11.2	Decision 11.1.1 11.1.2 Sorting 11.2.1 11.2.2 11.2.3 Selection 11.3.1 11.3.2 11.3.3	unds for Comparison-Based Algorithms In 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: $\Omega(\log n)$	35 36 36 36 36 36 36 36 36
_	11.1 11.2	Decision 11.1.1 11.1.2 Sorting 11.2.1 11.2.2 11.2.3 Selection 11.3.1 11.3.2 11.3.3 Advers 11.4.1	unds for Comparison-Based Algorithms In 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: $\Omega(\log n)$	35 36 36 36 36 36 36 36 36 36 36 36

12	Alge	ebraic	and Non-Comparison Lower Bounds	. 37
	12.1	Algebra	aic Decision Trees	37
		12.1.1	Extending Beyond Comparisons	37
		12.1.2	Element Distinctness Lower Bound	37
	12.2	Comm	unication Complexity	37
		12.2.1	Models and Definitions	37
		12.2.2	Applications to Data Structures	37
	12.3	Cell-Pr	obe Model	37
		12.3.1	Lower Bounds for Data Structures	37
		12.3.2	Dynamic vs. Static Data Structures	37
	12.4	Exercis	ses	37
٧	Sp	ecia	lized Topics and Applications	38
13	Ana	lysis d	of Specific Algorithm Paradigms	. 39
	13.1	Divide-	and-Conquer Algorithms	40
		13.1.1	General Framework and Recurrence Relations	40
		13.1.2	Examples: Mergesort, Quicksort, Strassen's Algorithm	40
		13.1.3	Optimality and Lower Bounds	40
	13.2	Greedy	Algorithms	40
		13.2.1	Correctness via Exchange Arguments	40
		13.2.2	Matroid Theory (Brief Introduction)	40
		13.2.3	Examples: Huffman Coding, Kruskal's MST	40
	13.3	Dynam	ic Programming	40
		13.3.1	Optimal Substructure and Overlapping Subproblems	40
		13.3.2	Memoization vs. Tabulation	40
		13.3.3	Time and Space Complexity Analysis	40
		13.3.4	Examples: Knapsack, Edit Distance, Matrix Chain Multiplication	40
	13.4	Backtra	acking and Branch-and-Bound	40
		13.4.1	Pruning the Search Space	40
		13.4.2	Worst-Case Exponential, Average-Case Better	40
		13.4.3	Examples: N-Queens, Traveling Salesman	40
	13.5	Exercis	es	40
14	Onli	ine Ald	gorithms and Competitive Analysis	41

	14.1	Introdu	ction to Online Algorithms	41
		14.1.1	Online vs. Offline Problems	41
		14.1.2	Competitive Ratio	41
	14.2	Examp	les of Online Problems	41
		14.2.1	Paging and Caching (LRU, FIFO, LFU)	41
		14.2.2	Load Balancing	41
		14.2.3	Online Scheduling	41
	14.3	Compe	titive Analysis Techniques	41
		14.3.1	Deterministic vs. Randomized Algorithms	41
		14.3.2	Lower Bounds via Adversary Arguments	41
	14.4	Exercis	ses	41
15	Арр	roxim	ation Algorithms	42
			ction to Approximation	42
		15.1.1	NP-Hardness and Intractability	42
		15.1.2	Approximation Ratios	42
	15.2	Examp	les of Approximation Algorithms	42
		15.2.1	Vertex Cover (2-Approximation)	42
		15.2.2	Set Cover (Greedy, $\log n$ -Approximation)	42
		15.2.3	Traveling Salesman Problem (Metric TSP)	42
	15.3	Analysi	is Techniques	42
		15.3.1	Bounding Optimal Solutions	42
		15.3.2	Linear Programming Relaxations	42
	15.4	Exercis	ses	42
16	Para	ametei	rized Complexity	43
			ction to Parameterized Algorithms	43
		16.1.1	Fixed-Parameter Tractability (FPT)	43
		16.1.2	Kernelization	43
	16.2	Examp	les and Analysis	43
		16.2.1	Vertex Cover Parameterized by Solution Size	43
		16.2.2	Treewidth and Graph Algorithms	43
	16.3	W-Hier	archy and Hardness	43
	-		W[1], W[2], and Beyond	43
	16.4		ses	43
	~			

VI	P	ractio	cal Considerations and Case Studies	44
17	Froi	n The	ory to Practice	45
	17.1	Hidder	Constants and Lower-Order Terms	45
		17.1.1	When $O(n \log n)$ Beats $O(n)$ in Practice	45
		17.1.2	Empirical Performance Measurements	45
	17.2	Algorith	nm Engineering	45
		17.2.1	Profiling and Benchmarking	45
		17.2.2	Tuning for Specific Hardware	45
		17.2.3	Libraries and Implementations (STL, Boost, etc.)	45
	17.3	Paralle	I and Distributed Algorithm Analysis	45
		17.3.1	Scalability and Speedup	45
		17.3.2	Amdahl's Law and Gustafson's Law	45
	17.4	Energy	Efficiency	45
		17.4.1	Energy as a Resource	45
		17.4.2	Green Computing and Mobile Devices	45
	17.5	Exercis	ses	45
18	Cas	e Stud	dies	46
	18.1	Sorting	Algorithms in Practice	46
		18.1.1	Timsort, Introsort, Radix Sort	46
		18.1.2	Comparison of Theoretical vs. Empirical Performance	46
	18.2	Graph	Algorithms in Large-Scale Systems	46
			rigoritimo in Large Geale Gystems	
		18.2.1	Web Graphs and PageRank	46
		18.2.1 18.2.2	· · · · · · · · · · · · · · · · · · ·	46 46
	18.3	18.2.2	Web Graphs and PageRank	
	18.3	18.2.2	Web Graphs and PageRank	46
	18.3	18.2.2 Machir	Web Graphs and PageRank	46 46
		18.2.2 Machir 18.3.1 18.3.2	Web Graphs and PageRank	46 46 46
		18.2.2 Machir 18.3.1 18.3.2	Web Graphs and PageRank	46 46 46
		18.2.2 Machir 18.3.1 18.3.2 Databa	Web Graphs and PageRank Social Network Analysis The Learning and Data Science Complexity of Training Algorithms SGD, AdaGrad, Adam: Time and Space Analysis ase Systems	46 46 46 46
	18.4	18.2.2 Machir 18.3.1 18.3.2 Databa 18.4.1 18.4.2	Web Graphs and PageRank Social Network Analysis The Learning and Data Science Complexity of Training Algorithms SGD, AdaGrad, Adam: Time and Space Analysis ase Systems Query Optimization	46 46 46 46 46
A	18.4	18.2.2 Machir 18.3.1 18.3.2 Databa 18.4.1 18.4.2 Exercis	Web Graphs and PageRank Social Network Analysis The Learning and Data Science Complexity of Training Algorithms SGD, AdaGrad, Adam: Time and Space Analysis ase Systems Query Optimization Indexing Structures (B-Trees, LSM-Trees)	46 46 46 46 46 46

	A.2	Logarithms and Exponentials	47				
	A.3	Recurrence Relations (Quick Reference)	47				
	A.4	Probability Distributions	47				
	A.5	Matrix Operations	47				
В	B Pseudocode Conventions						
	B.1	Notation and Style	48				
	B.2	Common Data Structures	48				
С	Sol	utions to Selected Exercises	49				
D	Glo	ssary of Terms	50				
Ε	Inde	ex of Algorithms	51				
F	Annotated Bibliography						
	F.1	Foundational Texts	52				
	F.2	Research Papers by Topic	52				
	F.3	Online Courses and Resources	52				

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

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

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

A Note on Rigor

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

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

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

Final Thoughts

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

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

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

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

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

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

Mahdi 2025-2026

Acknowledgments

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

Part I Foundations

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

Chapter 1

Mathematical and Algorithmic Prerequisites

HIS CHAPTER establishes the mathematical foundation required for rigorous algorithm analysis. We cover discrete mathematics, probability theory, mathematical analysis, linear algebra, and number theory—each selected for its direct relevance to algorithmic reasoning.

Why This Chapter Matters:

- **Discrete Mathematics** provides the language of algorithms: sets, functions, graphs, and proof techniques
- Probability Theory enables average-case and randomized algorithm analysis
- Mathematical Analysis gives us tools for asymptotic reasoning and limits
- Linear Algebra underpins graph algorithms, Markov chains, and numerical methods
- Number Theory appears in cryptography, hashing, and randomization

How to Use This Chapter: If you're already comfortable with a topic, skim it for notation and conventions. Each section includes self-assessment problems to verify your understanding.

1.1 Discrete Mathematics: The Language of Algorithms

Discrete mathematics forms the grammatical structure of algorithm design. Unlike continuous mathematics, we work with countable, distinct objects: integers, graphs, logical propositions. This section covers the foundational concepts that appear repeatedly throughout algorithm analysis.

1.1.1 Sets, Functions, and Relations

Set Theory Fundamentals

Operations on Sets: Union, Intersection, Difference, Cartesian Product

Functions: Injective, Surjective, Bijective Mappings

Relations and Equivalence Classes

Order Relations: Partial Orders, Total Orders, Lattices

1.1.2 Combinatorics: Counting and Arrangements

Combinatorics answers "how many?" questions essential to algorithm analysis: How many possible inputs exist? How many comparisons are necessary?

Fundamental Counting Principles: Sum and Product Rules

Permutations: With and Without Repetition

Combinations and the Binomial Theorem

Pigeonhole Principle and Its Applications

Inclusion-Exclusion Principle

Generating Functions for Combinatorial Sequences

Combinatorial Identities and Proof Techniques

1.1.3 Graph Theory Essentials

Graphs model relationships between entities. They appear everywhere: networks, dependencies, state machines, data structures.

Graph Definitions: Vertices, Edges, Degree

Representations: Adjacency Matrix, Adjacency List, Edge List

Paths, Walks, Cycles, and Connectivity

Trees: Definitions, Properties, and Rooted Trees

Directed Acyclic Graphs (DAGs) and Topological Ordering

Special Graph Classes: Bipartite, Complete, Planar

Preview: BFS and DFS as Fundamental Traversals

1.1.4 Proof Techniques: The Art of Rigorous Reasoning

Algorithm correctness and complexity bounds require proofs. This subsection teaches you how to construct and verify arguments.

Direct Proof: Structure and Examples

Proof by Contradiction: Assuming the Negation

Proof by Contrapositive

Mathematical Induction: Weak Form

Strong Induction and the Well-Ordering Principle

Structural Induction for Recursive Definitions

Common Proof Pitfalls and How to Avoid Them

1.2 Probability Theory for Algorithm Analysis

Randomized algorithms and average-case analysis require probability. This section builds intuition and formal tools for probabilistic reasoning.

First Edition • 2025 5

1.2.1 Foundations: Sample Spaces and Events

Sample Spaces: The Universe of Outcomes

Probability Axioms: Kolmogorov's Framework

Equally Likely Outcomes: The Discrete Uniform Distribution

Conditional Probability: Updating Beliefs

Law of Total Probability

Bayes' Theorem: Inverting Conditional Probabilities

1.2.2 Random Variables and Expectations

Random variables map outcomes to numbers, allowing numerical analysis. Expectation is the most important summary statistic in algorithm analysis.

Discrete Random Variables: Definition

Probability Mass Functions (PMF)

Expected Value: Definition and Intuition

Functions of Random Variables

Properties of Expectation

1.2.3 Common Distributions in Algorithms

Bernoulli Distribution: Single Trial

Binomial Distribution: Multiple Independent Trials

Geometric Distribution: Waiting Times

Poisson Distribution: Rare Events

Uniform Distribution: Equally Likely Values

1.2.4 Linearity of Expectation: The Most Powerful Tool

Linearity of expectation works even when random variables are dependent—this makes it extraordinarily useful in algorithm analysis.

Statement and Proof

Why Independence Is Not Required

Indicator Random Variables

Applications in Algorithm Analysis

Case Study: Expected Number of Comparisons in QuickSort

1.2.5 Independence and Conditional Expectation

Independence of Events

Independence of Random Variables

Conditional Expectation: E[X|Y]

Applications to Randomized Algorithms

1.2.6 Variance and Concentration

Variance: Measuring Spread

Standard Deviation

Properties of Variance

Chebyshev's Inequality: Probabilistic Bounds

Preview: Chernoff Bounds (Covered in Part 2)

1.2.7 Moment Generating Functions (Brief Preview)

Definition and Intuition

Using MGFs to Compute Moments

Sum of Independent Random Variables

1.3 Mathematical Analysis: Limits and Asymptotics

Algorithm analysis is fundamentally about asymptotic behavior as input size grows. This section provides the calculus and analysis tools needed for rigorous reasoning.

1.3.1 Limits and Continuity

Limits of Sequences: $\lim_{n\to\infty} a_n$

Limit Laws: Addition, Multiplication, Composition

L'Hôpital's Rule: Resolving Indeterminate Forms

Asymptotic Equivalence: $f \sim g$

Landau Notation: Informal Preview

1.3.2 Sequences and Series

Convergence of Sequences

Infinite Series: $\sum_{n=1}^{\infty} a_n$

Geometric Series: $\sum_{k=0}^{n} r^k$

Harmonic Series and Generalizations

Power Series and Radius of Convergence

1.3.3 Summations and Closed Forms

Converting sums to closed forms is essential for exact complexity analysis.

Basic Summation Formulas

Arithmetic and Geometric Progressions

Techniques for Finding Closed Forms

Perturbation Method

Repertoire Method (Concrete Mathematics Approach)

Euler-Maclaurin Formula: Sum-Integral Connection

1.3.4 Integration and Differentiation

Fundamental Theorem of Calculus

Integration Techniques: Substitution, Parts, Partial Fractions

Approximating Sums with Integrals

Growth Rates via Derivatives

1.3.5 Taylor Series and Asymptotic Expansions

Taylor Series: Definition and Motivation

Common Taylor Series

Using Taylor Series for Approximation

Asymptotic Expansions

1.3.6 Stirling's Approximation

Factorials grow faster than polynomials but slower than exponentials. Stirling's formula provides precise asymptotic behavior.

Statement of Stirling's Formula

Proof Sketch via Integral Approximation

Applications in Combinatorics

More Precise Versions and Error Bounds

1.4 Linear Algebra for Algorithmic Applications

Linear algebra appears in graph algorithms, Markov chains, numerical methods, and machine learning. We focus on computational and algorithmic aspects.

1.4.1 Vectors, Matrices, and Linear Systems

Vector Spaces and Subspaces

Linear Independence, Span, and Basis

Matrix Operations: Addition, Multiplication, Transpose

Matrix Multiplication as Linear Transformation Composition

Systems of Linear Equations: Gaussian Elimination

1.4.2 Eigenvalues and Eigenvectors

Definitions: $Av = \lambda v$

Characteristic Polynomial: $det(A - \lambda I) = 0$

Diagonalization: $A = PDP^{-1}$

Spectral Theorem for Symmetric Matrices

Applications to Algorithm Analysis

1.4.3 Markov Chains and Random Walks

Markov chains model random processes with memoryless transitions. They appear in randomized algorithms and probabilistic analysis.

First Edition • 2025 10 | 52

Stochastic	Matrices:	Definition	and Pro	perties

Steady-State Distribution: $\pi P = \pi$

Random Walks on Graphs

Case Study: PageRank Algorithm

1.4.4 Matrix Operations and Complexity

Standard Matrix Multiplication: $O(n^3)$

Strassen's Algorithm: $O(n^{\log_2 7}) \approx O(n^{2.81})$

Coppersmith-Winograd and Beyond

Matrix Inversion Complexity

Solving Linear Systems: LU Decomposition, Cholesky

1.5 Number Theory Essentials

Number theory provides tools for cryptography, hashing, pseudorandom generation, and modular arithmetic—all crucial in modern algorithms.

1.5.1 Divisibility and Modular Arithmetic

Division Algorithm: a = bq + r

Modular Arithmetic: $a \equiv b \pmod{m}$

Properties of Modular Operations

Modular Inverses: $ax \equiv 1 \pmod{m}$

Chinese Remainder Theorem

1.5.2 Prime Numbers and Factorization

Definition of Prime Numbers

Fundamental Theorem of Arithmetic

Prime Distribution: Prime Number Theorem

Sieve of Eratosthenes: $O(n \log \log n)$

Primality Testing: Fermat, Miller-Rabin

1.5.3 Greatest Common Divisor and Euclidean Algorithm

The Euclidean algorithm is one of the oldest and most elegant algorithms. Its analysis demonstrates beautiful connections between number theory and complexity.

GCD Definition and Properties

Euclidean Algorithm

Extended Euclidean Algorithm

Complexity Analysis: $O(\log \min(a, b))$

Lamé's Theorem: Connection to Fibonacci Numbers

1.5.4 Applications to Cryptography and Hashing

Modular Exponentiation: Fast Algorithm

RSA Cryptosystem: Overview

Hash Functions: Collision Resistance

Universal Hashing (Preview)

1.6 Additional Mathematical Tools

1.6.1 Logarithms and Exponentials

Properties of Logarithms

Change of Base Formula

Natural vs. Binary Logarithms in CS

Exponential Growth and Decay

Comparing Growth Rates

1.6.2 Floor and Ceiling Functions

Definitions: $\lfloor x \rfloor$ and $\lceil x \rceil$

Useful Identities and Properties

Applications in Algorithm Analysis

1.6.3 Asymptotic Notation: Informal Preview

We provide intuition here; formal definitions appear in Part 2, Chapter 2.

First Edition • 2025 13 | 52

Intuition Behind O, Ω , Θ

Why We Need Formal Definitions

Common Growth Rate Hierarchy

1.7 Self-Assessment and Further Study

Test your understanding with these exercises before proceeding to Part 2. If you struggle with a section, revisit the material or consult the recommended resources.

1.7.1 Self-Assessment Exercises

Discrete Mathematics Problems

Probability Problems

Analysis Problems

Linear Algebra Problems

Number Theory Problems

1.7.2 Further Reading and Resources

Recommended Textbooks

Online Resources and Video Lectures

Practice Problem Collections

1.7.3 Notation and Conventions Used in This Book

Standard Mathematical Symbols

Asymptotic Notation

Probability Notation

Graph Notation

First Edition • 2025

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	Relations

- 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

4.4.2

4.4.1 Statement of the Master Theorem (Standard Form)

Three Cases and Their Intuition

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

5.1 Defining Input 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

0.1	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
	Markov's Inequality Chebyshev's Inequality
6.3.2	•
6.3.2 6.3.3	Chebyshev's Inequality
6.3.2 6.3.3	Chebyshev's Inequality Chernoff Bounds
6.3.26.3.36.3.46.4	Chebyshev's Inequality Chernoff Bounds Applications to Load Balancing and Hashing
6.3.26.3.36.3.46.4	Chebyshev's Inequality Chernoff Bounds Applications to Load Balancing and Hashing Randomized Algorithms Randomized Quicksort (Detailed Analysis)

Part III Advanced Analysis Techniques

Amortized Analysis

7.1	Introduction	to Amortiz	zed 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

8.1

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 1	202Compression and Succinct Data Structures

Cache-Aware and I/O Complexity

9.1 Introduction 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 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

31 | 52

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

4	4	-	-	•	•			
•		.1		001	CI	On.	Tre	Δc
_			·	יכנו	21	UIL	116	C 3

- 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

12.1	Algebraic Decision Irees
12.1.1	Extending Beyond Comparisons
12.1.2	Element Distinctness Lower Bound
12.2	Communication Complexity
12.2.1	Models and Definitions
12.2.2	Applications to Data Structures
12.3	Cell-Probe Model
12.3.1	Lower Bounds for 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