## ART

\_ \_ . \_ . .

# ALGORITHM ANALYSIS

FROM FOUNDATIONS TO PRACTICE

 $\Theta(\log n)$ 

<sup>Ω(n²)</sup> 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 19, 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 19, 2025

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

#### **Contents**

Ti	itle P	age		i
C	onte	nts	i	iii
Ρı	refac	e	xv	/ii
			ımentsxx	
	го	unaa	tions	1
1	Mat	hema	tical Foundations I: Discrete Structures	3
	1.1	Logic	and Proof Theory	4
		1.1.1	Propositional Logic	4
		1.1.2	Predicate Logic (First-Order Logic)	5
		1.1.3	Proof Techniques and Strategies	5
		1.1.4	Common Proof Errors and How to Avoid Them	7
		1.1.5	Advanced Topics in Logic	7
		1.1.6	Exercises	7
	1.2	Set Th	neory, Functions, and Relations	7
		1.2.1	Axiomatic Set Theory (Informal)	8
		1.2.2	Basic Set Operations	8
		1.2.3	Cardinality and Counting	8
		1.2.4	Functions: Mappings Between Sets	8
		1.2.5	Relations	9
		1.2.6	Exercises	10
	1.3	Comb	inatorial Analysis and Enumeration	10
		1.3.1	Fundamental Counting Principles	10
		1.3.2	Permutations and Arrangements	10
		1.3.3	Combinations and Selections	11
		1.3.4	Inclusion-Exclusion Principle	12
		1.3.5	Pigeonhole Principle and Ramsey Theory	12
		1.3.6	Generating Functions	12
		1.3.7	Recurrence Relations	13
		1.3.8	Catalan Numbers	13
		120	Stirling Numbers	1 /

		1.3.10	Asymptotic Methods in Combinatorics	14
		1.3.11	Exercises	15
	1.4	Graph	Theory Foundations	15
		1.4.1	Basic Definitions and Representations	15
		1.4.2	Connectivity and Paths	16
		1.4.3	Trees and Forests	17
		1.4.4	Directed Acyclic Graphs (DAGs)	18
		1.4.5	Graph Coloring	18
		1.4.6	Planar Graphs	18
		1.4.7	Advanced Topics	18
		1.4.8	Exercises	19
	1.5	Algebr	aic Structures for Algorithm Design	19
		1.5.1	Groups	19
		1.5.2	Rings and Fields	20
		1.5.3	Applications to Cryptography	20
		1.5.4	Applications to Coding Theory	20
		1.5.5	Exercises	20
2	Mat	hemat	tical Foundations II: Analysis and Probability	21
			•	
II	Fo	ounda	ations of Algorithmic Analysis	22
	Fo	ounda oducti	ations of Algorithmic Analysis	<b>22</b> 23
II	Fo	ounda oducti What I	ations of Algorithmic Analysis	<b>22 23</b> 23
II	Fo	oducti What I	ations of Algorithmic Analysis	<b>22</b> 23
II	Fo	ounda oducti What I	ations of Algorithmic Analysisson to Algorithm Analysiss Algorithm Analysis?	<b>22 23</b> 23
II	Fo	oducti What I	ations of Algorithmic Analysis  ion to Algorithm Analysis  s Algorithm Analysis?  Correctness vs. Efficiency	<b>22 23</b> 23
II	Fo	oducti What I: 3.1.1 3.1.2 3.1.3	ations of Algorithmic Analysis  on to Algorithm Analysis  s Algorithm Analysis?  Correctness vs. Efficiency  Resource Measures: Time, Space, Energy, I/O	22 23 23 23 23
II	Fo Intro 3.1	oducti What I: 3.1.1 3.1.2 3.1.3	ations of Algorithmic Analysis  on to Algorithm Analysis  s Algorithm Analysis?  Correctness vs. Efficiency  Resource Measures: Time, Space, Energy, I/O  The Need for Mathematical Models	22 23 23 23 23 23
II	Fo Intro 3.1	oducti What I 3.1.1 3.1.2 3.1.3 The RA	ations of Algorithmic Analysis  fon to Algorithm Analysis  s Algorithm Analysis?  Correctness vs. Efficiency  Resource Measures: Time, Space, Energy, I/O  The Need for Mathematical Models  AM Model of Computation	22 23 23 23 23 23
II	Fo Intro 3.1	oducti What I 3.1.1 3.1.2 3.1.3 The RA 3.2.1	ations of Algorithmic Analysis  on to Algorithm Analysis  s Algorithm Analysis?  Correctness vs. Efficiency  Resource Measures: Time, Space, Energy, I/O  The Need for Mathematical Models  AM Model of Computation  Basic Operations and Unit-Cost Assumption	22 23 23 23 23 23 23 23 23
II	Fo Intro	oducti What I: 3.1.1 3.1.2 3.1.3 The Ri 3.2.1 3.2.2 3.2.3	ations of Algorithmic Analysis  on to Algorithm Analysis  s Algorithm Analysis?  Correctness vs. Efficiency  Resource Measures: Time, Space, Energy, I/O  The Need for Mathematical Models  AM Model of Computation  Basic Operations and Unit-Cost Assumption  Memory Access Model	22 23 23 23 23 23 23 23 23 23
II	3.1 3.2	Ounda oducti What II 3.1.1 3.1.2 3.1.3 The RA 3.2.1 3.2.2 3.2.3	Actions of Algorithmic Analysis  Son to Algorithm Analysis  Son Algorithm Analysis?  Correctness vs. Efficiency  Resource Measures: Time, Space, Energy, I/O  The Need for Mathematical Models  AM Model of Computation  Basic Operations and Unit-Cost Assumption  Memory Access Model  Limitations and Extensions of the RAM Model	22 23 23 23 23 23 23 23 23 23 23
II	3.1 3.2	oducti What II 3.1.1 3.1.2 3.1.3 The Ri 3.2.1 3.2.2 3.2.3 Measu	Ations of Algorithmic Analysis  Son to Algorithm Analysis  Son Algorithm Analysis  Son Algorithm Analysis  Son Algorithm Analysis  Son Algorithm Analysis  Correctness vs. Efficiency  Resource Measures: Time, Space, Energy, I/O  The Need for Mathematical Models  AM Model of Computation  Basic Operations and Unit-Cost Assumption  Memory Access Model  Limitations and Extensions of the RAM Model  Aring Algorithm Performance	22 23 23 23 23 23 23 23 23 23 23 23 23 2
II	3.1 3.2	Ounda oducti What I: 3.1.1 3.1.2 3.1.3 The Ri 3.2.1 3.2.2 3.2.3 Measu 3.3.1	Adjorithm Analysis  Ion to Algorithm Analysis  Is Algorithm Analysis?  Correctness vs. Efficiency  Resource Measures: Time, Space, Energy, I/O  The Need for Mathematical Models  AM Model of Computation  Basic Operations and Unit-Cost Assumption  Memory Access Model  Limitations and Extensions of the RAM Model  Iring Algorithm Performance  Input Size and Problem Instances	22 23 23 23 23 23 23 23 23 23 23 23 23 2
II	3.1 3.2	oducti What I: 3.1.1 3.1.2 3.1.3 The Ri 3.2.1 3.2.2 3.2.3 Measu 3.3.1 3.3.2 3.3.3	Adjorithm Analysis  Son to Algorithm Analysis  S	22 23 23 23 23 23 23 23 23 23 23 23 23 2

		3.4.1	P, NP, NP-Complete, and NP-Hard (Brief Introduction)	23
		3.4.2	Why We Focus on Polynomial-Time Algorithms	23
4	Asy	mptot	tic Notation	24
	4.1	The Ne	eed for Asymptotic Analysis	25
		4.1.1	Why Exact Counts Are Often Impractical	25
		4.1.2	Growth Rates and Scalability	25
	4.2	Big-O	Notation (O)	25
		4.2.1	Formal Definition	25
		4.2.2	Intuition: Upper Bounds	25
		4.2.3	Common Functions and Their Growth Rates	25
		4.2.4	Examples and Non-Examples	25
		4.2.5	Properties of Big-O	25
	4.3	Big-Or	mega Notation ( $\Omega$ )	25
		4.3.1	Formal Definition	25
		4.3.2	Intuition: Lower Bounds	25
		4.3.3	Examples and Applications	25
		4.3.4	Relationship Between $O$ and $\Omega$	25
	4.4	Big-Th	leta Notation ( $\Theta$ )	25
		4.4.1	Formal Definition	25
		4.4.2	Intuition: Tight Bounds	25
		4.4.3	When to Use $\Theta$ vs. $O$	25
		4.4.4	Examples of Tight Bounds	25
	4.5	Little-o	and Little-omega Notation $(o,\omega)$	25
		4.5.1	Formal Definitions	25
		4.5.2	Strict Asymptotic Bounds	25
		4.5.3	Applications in Analysis	25
	4.6	Comm	on Misconceptions and Pitfalls	25
		4.6.1	Confusing $O$ with $\Theta$	25
		4.6.2	Ignoring Constants in Practice	25
		4.6.3	Misapplying Asymptotic Notation to Small Inputs	25
	4.7	Compa	aring Functions	25
		4.7.1	L'Hôpital's Rule for Limits	25
		4.7.2	Logarithmic vs. Polynomial vs. Exponential Growth	25

		4.7.3	Hierarchy of Common Complexity Classes	25
	4.8	Exerci	ses	25
5	Rec	urren	ce Relations and Their Solutions	26
	5.1	Introdu	uction to Recurrence Relations	27
		5.1.1	What Are Recurrences?	27
		5.1.2	Why They Arise in Algorithm Analysis	27
		5.1.3	Examples from Divide-and-Conquer Algorithms	27
	5.2	The Su	ubstitution Method	27
		5.2.1	Guessing the Solution	27
		5.2.2	Proving by Induction	27
		5.2.3	Examples: Mergesort, Binary Search	27
		5.2.4	Strengthening the Inductive Hypothesis	27
	5.3	The R	ecursion-Tree Method	27
		5.3.1	Visualizing the Recurrence	27
		5.3.2	Summing Over Levels	27
		5.3.3	Examples and Illustrations	27
		5.3.4	Limitations and When to Use	27
	5.4	The M	aster Theorem	27
		5.4.1	Statement of the Master Theorem (Standard Form)	27
		5.4.2	Three Cases and Their Intuition	27
		5.4.3	Proof Sketch (Via Recursion Trees)	27
		5.4.4	Examples: $T(n) = aT(n/b) + f(n) \dots \dots \dots \dots \dots$	27
		5.4.5	Regularity Condition and Edge Cases	27
		5.4.6	Extended Master Theorem (Akra-Bazzi)	27
	5.5	The Al	kra-Bazzi Method	27
		5.5.1	Motivation: Unequal Subproblem Sizes	27
		5.5.2	Statement and Conditions	27
		5.5.3	Examples and Applications	27
		5.5.4	Proof Overview (Advanced)	27
	5.6	Linear	Recurrences with Constant Coefficients	27
		5.6.1	Homogeneous Linear Recurrences	27
		5.6.2	Characteristic Equations	27
		5.6.3	Solving Fibonacci-Type Recurrences	27

		5.6.4	Non-Homogeneous Recurrences and Particular Solutions	27
	5.7	Genera	ating Functions	27
		5.7.1	Introduction to Generating Functions	27
		5.7.2	Solving Recurrences with Generating Functions	27
		5.7.3	Examples: Catalan Numbers, Stirling Numbers	27
	5.8	Advan	ced Topics	27
		5.8.1	Full History Recurrences	27
		5.8.2	Recurrences with Variable Coefficients	27
		5.8.3	Probabilistic Recurrences (Preview)	27
	5.9	Exercis	ses	27
6	Bes	t-Case	e, Worst-Case, and Average-Case Analysis	28
	6.1	Definir	ng Input Classes	29
		6.1.1	What Constitutes an "Input"?	29
		6.1.2	Problem Instances and Instance Distributions	29
	6.2	Best-C	Case Analysis	29
		6.2.1	Definition and Purpose	29
		6.2.2	Examples: Insertion Sort, Linear Search	29
		6.2.3	When Best-Case Matters (and When It Doesn't)	29
	6.3	Worst-	Case Analysis	29
		6.3.1	Definition and Motivation	29
		6.3.2	Guarantees and Robustness	29
		6.3.3	Examples: Quicksort, Searching in Unsorted Arrays	29
		6.3.4	Lower Bounds and Optimality	29
	6.4	Averag	ge-Case Analysis	29
		6.4.1	Definition: Expected Running Time	29
		6.4.2	Assumptions About Input Distributions	29
		6.4.3	Probabilistic Models: Uniform, Gaussian, etc	29
		6.4.4	Examples: Quicksort, Hashing, Skip Lists	29
	6.5	Probak	pilistic Analysis vs. Randomized Algorithms	29
		6.5.1	Distinction Between the Two Concepts	29
		6.5.2	Randomized Quicksort: Expected $O(n \log n)$	29
		6.5.3	Las Vegas vs. Monte Carlo Algorithms	29
	6.6	Smoot	hed Analysis	29

		6.6.1	Motivation: Beyond Worst-Case Pessimism	29
		6.6.2	Introduction to Smoothed Analysis	29
		6.6.3	Case Study: Simplex Algorithm	29
	6.7	Exercis	ses	29
7	Pro	babilis	stic Analysis of Algorithms	30
	7.1	Found	ations of Probabilistic Analysis	31
		7.1.1	Random Variables in Algorithm Analysis	31
		7.1.2	Indicator Random Variables	31
		7.1.3	Linearity of Expectation	31
	7.2	Expect	ted Running Time	31
		7.2.1	Formal Definition	31
		7.2.2	Computing Expectations via Indicator Variables	31
		7.2.3	Examples: Hiring Problem, Randomized Quicksort	31
	7.3	Probak	pilistic Bounds	31
		7.3.1	Markov's Inequality	31
		7.3.2	Chebyshev's Inequality	31
		7.3.3	Chernoff Bounds	31
		7.3.4	Applications to Load Balancing and Hashing	31
	7.4	Rando	mized Algorithms	31
		7.4.1	Randomized Quicksort (Detailed Analysis)	31
		7.4.2	Randomized Selection (Quickselect)	31
		7.4.3	Hashing and Universal Hash Functions	31
		7.4.4	Bloom Filters and Probabilistic Data Structures	31
	7.5	Analys	sis of Randomized Data Structures	31
		7.5.1	Skip Lists	31
		7.5.2	Treaps	31
		7.5.3	Hash Tables with Chaining and Open Addressing	31
	7.6	High-P	Probability Results	31
		7.6.1	What Does "With High Probability" Mean?	31
		7.6.2	Concentration Inequalities	31
		7.6.3	Union Bound and Probabilistic Method	31
	77	Exercis	ses	31

Ш	Α	dvan	ced Analysis Techniques	32
8	Amo	ortized	d Analysis	33
	8.1	Introdu	oction to Amortized Analysis	34
		8.1.1	Motivation: Why Average Per-Operation Cost?	34
		8.1.2	Amortized vs. Average-Case Analysis	34
		8.1.3	When to Use Amortized Analysis	34
	8.2	Aggreg	gate Analysis	34
		8.2.1	Definition and Methodology	34
		8.2.2	Example: Dynamic Array (Vector) Resizing	34
		8.2.3	Example: Binary Counter Increment	34
		8.2.4	Example: Stack with Multipop	34
	8.3	The Ac	counting Method	34
		8.3.1	Conceptual Framework: Credits and Debits	34
		8.3.2	Defining Amortized Costs	34
		8.3.3	Example: Dynamic Array via Accounting	34
		8.3.4	Example: Splay Trees (Introduction)	34
		8.3.5	Ensuring Non-Negative Credit Balance	34
	8.4	The Po	otential Method	34
		8.4.1	Potential Functions: Definition and Intuition	34
		8.4.2	Relating Amortized Cost to Actual Cost	34
		8.4.3	Designing Good Potential Functions	34
		8.4.4	Example: Dynamic Array via Potential Method	34
		8.4.5	Example: Binary Counter via Potential Method	34
		8.4.6	Example: Fibonacci Heaps (Overview)	34
	8.5	Compa	aring the Three Methods	34
		8.5.1	Strengths and Weaknesses	34
		8.5.2	When to Choose Which Method	34
		8.5.3	Equivalence of Methods (Informal Discussion)	34
	8.6	Advand	ced Applications	34
		8.6.1	Splay Trees: Full Analysis	34
		8.6.2	Fibonacci Heaps	34
		8.6.3	Disjoint-Set Union (Union-Find)	34
	8.7	Exercis	ses	34

9	Spa	ce Co	mplexity Analysis	35
	9.1	Introdu	iction to Space Complexity	36
		9.1.1	Why Space Matters	36
		9.1.2	Types of Memory: Stack, Heap, Static	36
		9.1.3	In-Place vs. Out-of-Place Algorithms	36
	9.2	Measu	ring Space Usage	36
		9.2.1	Auxiliary Space vs. Total Space	36
		9.2.2	Recursive Call Stack Depth	36
		9.2.3	Implicit vs. Explicit Data Structures	36
	9.3	Examp	oles of Space Complexity Analysis	36
		9.3.1	Iterative Algorithms: Loops and Arrays	36
		9.3.2	Recursive Algorithms: Mergesort, Quicksort	36
		9.3.3	Dynamic Programming: Memoization vs. Tabulation	36
		9.3.4	Graph Algorithms: BFS, DFS, Shortest Paths	36
	9.4	Space-	-Time Tradeoffs	36
		9.4.1	Caching and Memoization	36
		9.4.2	Lookup Tables and Precomputation	36
		9.4.3	Compression and Succinct Data Structures	36
	9.5	Stream	ning and Online Algorithms	36
		9.5.1	Sublinear Space Algorithms	36
		9.5.2	Sketching and Sampling Techniques	36
		9.5.3	Examples: Distinct Elements, Heavy Hitters	36
	9.6	Space	Complexity Classes	36
		9.6.1	L, NL, PSPACE (Brief Overview)	36
		9.6.2	Savitch's Theorem	36
	9.7	Exercis	ses	36
10	) Cac	he-Aw	vare and I/O Complexity	37
			uction to the Memory Hierarchy	38
		10.1.1	Registers, Cache (L1, L2, L3), RAM, Disk	38
		10.1.2	Latency and Bandwidth Characteristics	38
		10.1.3	Why Algorithm Design Must Consider Memory	38
	10.2	The Ex	kternal Memory Model (I/O Model)	38
		10.2.1	Parameters: $N$ (data size), $M$ (memory size), $B$ (block size)	38
			,, , , , , , , , , , , , , , , , , , , ,	

		10.2.2	I/O Complexity: Counting Block Transfers	38
		10.2.3	Comparison with RAM Model	38
	10.3	I/O-Effi	cient Algorithms	38
		10.3.1	Scanning and Sorting	38
		10.3.2	Matrix Operations	38
		10.3.3	Graph Algorithms	38
	10.4	Cache-	Oblivious Algorithms	38
		10.4.1	Motivation: Optimal Without Knowing $M$ and $B$	38
		10.4.2	Cache-Oblivious Sorting (Funnelsort)	38
		10.4.3	Cache-Oblivious Matrix Multiplication	38
		10.4.4	Cache-Oblivious B-Trees (van Emde Boas Layout)	38
	10.5	Cache-	Aware Analysis	38
		10.5.1	Modeling Cache Behavior	38
		10.5.2	Locality of Reference: Temporal and Spatial	38
		10.5.3	Blocking and Tiling Techniques	38
	10.6	Real-W	Vorld Considerations	38
		10.6.1	Multi-Level Caches	38
		10.6.2	Cache Replacement Policies (LRU, LFU, etc.)	38
		10.6.3	Prefetching and Speculative Execution	38
		10.6.4	False Sharing and Cache Line Effects	38
	10.7	Case S	Studies	38
		10.7.1	Database Query Processing	38
		10.7.2	External Memory Sorting in Practice	38
		10.7.3	Scientific Computing and Large-Scale Simulations	38
	10.8	Exercis	ses	38
11	Cac	he-Aw	vare Scheduling and Analysis for Multicores	39
	11.1	Introdu	ction to Multicore and Parallel Computing	40
		11.1.1	Shared vs. Distributed Memory	40
		11.1.2	Parallel Models: PRAM, Fork-Join, Work-Stealing	40
		11.1.3	Performance Metrics: Work, Span, Parallelism	40
	11.2	Cache	Coherence and Consistency	40
		11.2.1	MESI and MOESI Protocols	40
		11.2.2	False Sharing in Multicore Systems	40

		11.2.3	Impact on Algorithm Design	40
	11.3	Cache-	Aware Parallel Algorithms	40
		11.3.1	Parallel Sorting with Cache Awareness	40
		11.3.2	Parallel Matrix Multiplication (Strassen, Coppersmith-Winograd)	40
		11.3.3	Load Balancing and Task Granularity	40
	11.4	Real-Ti	ime and Embedded Systems	40
		11.4.1	WCET Analysis in Cache-Aware Contexts	40
		11.4.2	Predictability vs. Average-Case Performance	40
		11.4.3	Cache Partitioning and Locking	40
	11.5	Schedu	uling Strategies	40
		11.5.1	Static vs. Dynamic Scheduling	40
		11.5.2	Work-Stealing Algorithms	40
		11.5.3	Affinity Scheduling for Cache Locality	40
	11.6	Analysi	is Techniques	40
		11.6.1	DAG-Based Analysis (Cilk Model)	40
		11.6.2	Brent's Theorem and Greedy Scheduling	40
		11.6.3	Cache Miss Analysis in Parallel Programs	40
	11.7	Case S	Studies from Research	40
		11.7.1	ECRTS 2007: Cache-Aware Real-Time Scheduling	40
		11.7.2	Cache-Aware Scheduling for Multicores (Embedded Systems)	40
		11.7.3	VLDB 2019: Concurrent Hash Tables and Cache Performance	40
	11.8	Exercis	ses	40
V	1	ower	Bounds and Optimality	41
•			•	
_			unds for Comparison-Based Algorithms	
	12.1		on Trees	43
		12.1.1	Modeling Algorithms as Decision Trees	43
		12.1.2	Height of Decision Trees and Worst-Case Complexity	43
	12.2		Lower Bound	43
		12.2.1	Information-Theoretic Argument	43
		12.2.2	$\Omega(n \log n)$ Lower Bound for Comparison Sorting	43
		12.2.3	Implications and Optimal Algorithms	43
	12.3		on and Searching Lower Bounds	43
		12.3.1	Finding the Minimum: $\Omega(n)$	43

		12.3.2	Finding Median: Adversary Arguments	43
		12.3.3	Searching in Sorted Arrays: $\Omega(\log n)$	43
	12.4	Advers	ary Arguments	43
		12.4.1	General Framework	43
		12.4.2	Examples: Merging, Element Uniqueness	43
	12.5	Exercis	ses	43
13	Alge	ebraic	and Non-Comparison Lower Bounds	44
	13.1	Algebra	aic Decision Trees	44
		13.1.1	Extending Beyond Comparisons	44
		13.1.2	Element Distinctness Lower Bound	44
	13.2	Comm	unication Complexity	44
		13.2.1	Models and Definitions	44
		13.2.2	Applications to Data Structures	44
	13.3	Cell-Pr	obe Model	44
		13.3.1	Lower Bounds for Data Structures	44
		13.3.2	Dynamic vs. Static Data Structures	44
	13.4	Exercis	ses	44
٧	Sp	necia	lized Topics and Applications	4 =
		Jour	nzeu Topics and Applications	45
14	Ana		of Specific Algorithm Paradigms	
14		lysis (	of Specific Algorithm Paradigms	46
14		lysis (	•	46
14		lysis (	of Specific Algorithm Paradigms	<b>46</b>
14		lysis of Divide-	of Specific Algorithm Paradigms	46 47 47
14	14.1	Divide- 14.1.1 14.1.2 14.1.3	of Specific Algorithm Paradigms  and-Conquer Algorithms	46 47 47 47
14	14.1	Divide- 14.1.1 14.1.2 14.1.3	of Specific Algorithm Paradigms  and-Conquer Algorithms	46 47 47 47 47
14	14.1	Divide- 14.1.1 14.1.2 14.1.3 Greedy	of Specific Algorithm Paradigms  and-Conquer Algorithms	46 47 47 47 47
14	14.1	Divide- 14.1.1 14.1.2 14.1.3 Greedy 14.2.1	and-Conquer Algorithms	46 47 47 47 47 47
14	14.1	Divide- 14.1.1 14.1.2 14.1.3 Greedy 14.2.1 14.2.2 14.2.3	of Specific Algorithm Paradigms  and-Conquer Algorithms	46 47 47 47 47 47 47
14	14.1	Divide- 14.1.1 14.1.2 14.1.3 Greedy 14.2.1 14.2.2 14.2.3	Algorithms Of Specific Algorithm Paradigms and-Conquer Algorithms General Framework and Recurrence Relations Examples: Mergesort, Quicksort, Strassen's Algorithm Optimality and Lower Bounds Algorithms Correctness via Exchange Arguments Matroid Theory (Brief Introduction) Examples: Huffman Coding, Kruskal's MST	46 47 47 47 47 47 47
14	14.1	Divide- 14.1.1 14.1.2 14.1.3 Greedy 14.2.1 14.2.2 14.2.3 Dynam	and-Conquer Algorithms  General Framework and Recurrence Relations  Examples: Mergesort, Quicksort, Strassen's Algorithm  Optimality and Lower Bounds  Algorithms  Correctness via Exchange Arguments  Matroid Theory (Brief Introduction)  Examples: Huffman Coding, Kruskal's MST	46 47 47 47 47 47 47 47
14	14.1	Divide- 14.1.1 14.1.2 14.1.3 Greedy 14.2.1 14.2.2 14.2.3 Dynam 14.3.1	and-Conquer Algorithms	46 47 47 47 47 47 47 47 47

	14.4	Backtra	acking and Branch-and-Bound	47
		14.4.1	Pruning the Search Space	47
		14.4.2	Worst-Case Exponential, Average-Case Better	47
		14.4.3	Examples: N-Queens, Traveling Salesman	47
	14.5	Exercis	ses	47
15	Onli	ine Al	gorithms and Competitive Analysis	48
	15.1	Introdu	iction to Online Algorithms	48
		15.1.1	Online vs. Offline Problems	48
		15.1.2	Competitive Ratio	48
	15.2	Examp	oles of Online Problems	48
		15.2.1	Paging and Caching (LRU, FIFO, LFU)	48
		15.2.2	Load Balancing	48
		15.2.3	Online Scheduling	48
	15.3	Compe	etitive Analysis Techniques	48
		15.3.1	Deterministic vs. Randomized Algorithms	48
		15.3.2	Lower Bounds via Adversary Arguments	48
	15.4	Exercis	ses	48
16	Арр	roxim	ation Algorithms	49
	16.1	Introdu	ection to Approximation	49
		16.1.1	NP-Hardness and Intractability	49
		16.1.2	Approximation Ratios	49
	16.2	Examp	oles of Approximation Algorithms	49
		16.2.1	Vertex Cover (2-Approximation)	49
		16.2.2	Set Cover (Greedy, $\log n$ -Approximation)	49
		16.2.3	Traveling Salesman Problem (Metric TSP)	49
	16.3	Analys	is Techniques	49
		16.3.1	Bounding Optimal Solutions	49
		16.3.2	Linear Programming Relaxations	49
	16.4	Exercis	ses	49
17	Para	amete	rized Complexity	50
			action to Parameterized Algorithms	50
		17.1.1	Fixed-Parameter Tractability (FPT)	50
		17.1.2	Kernelization	50

	17.2	Examp	les and Analysis	50
		17.2.1	Vertex Cover Parameterized by Solution Size	50
		17.2.2	Treewidth and Graph Algorithms	50
	17.3	W-Hiera	archy and Hardness	50
		17.3.1	W[1], W[2], and Beyond	50
	17.4	Exercis	es	50
۷I	Р	ractio	cal Considerations and Case Studies	51
18	Fror	n The	ory to Practice	52
	18.1	Hidden	Constants and Lower-Order Terms	52
		18.1.1	When $O(n \log n)$ Beats $O(n)$ in Practice	52
		18.1.2	Empirical Performance Measurements	52
	18.2	Algorith	nm Engineering	52
		18.2.1	Profiling and Benchmarking	52
		18.2.2	Tuning for Specific Hardware	52
		18.2.3	Libraries and Implementations (STL, Boost, etc.)	52
	18.3	Parallel	and Distributed Algorithm Analysis	52
		18.3.1	Scalability and Speedup	52
		18.3.2	Amdahl's Law and Gustafson's Law	52
	18.4	Energy	Efficiency	52
		18.4.1	Energy as a Resource	52
		18.4.2	Green Computing and Mobile Devices	52
	18.5	Exercis	es	52
19	Cas	e Stud	lies	53
	19.1	Sorting	Algorithms in Practice	53
		19.1.1	Timsort, Introsort, Radix Sort	53
		19.1.2	Comparison of Theoretical vs. Empirical Performance	53
	19.2	Graph A	Algorithms in Large-Scale Systems	53
		19.2.1	Web Graphs and PageRank	53
		19.2.2	Social Network Analysis	53
	19.3	Machin	e Learning and Data Science	53
		19.3.1	Complexity of Training Algorithms	53
		19.3.2	SGD, AdaGrad, Adam: Time and Space Analysis	53

	19.4	Database Systems	53
		19.4.1 Query Optimization	53
		19.4.2 Indexing Structures (B-Trees, LSM-Trees)	53
	19.5	Exercises	53
Α	Mat	hematical Background	54
	A.1	Summation Formulas	54
	A.2	Logarithms and Exponentials	54
	A.3	Recurrence Relations (Quick Reference)	54
	A.4	Probability Distributions	54
	A.5	Matrix Operations	54
В	Pse	udocode Conventions	55
В	Pse B.1		<b>55</b> 55
В		Notation and Style	
B	B.1 B.2	Notation and Style	55
	B.1 B.2 <b>Sol</b> 1	Notation and Style	55 55
С	B.1 B.2 Solu	Notation and Style	55 55 <b>56</b>
C D	B.1 B.2 Solu Glos	Notation and Style	55 55 <b>56</b> <b>57</b>
C D E	B.1 B.2 Solu Glos	Notation and Style  Common Data Structures  utions to Selected Exercises  ssary of Terms  ex of Algorithms  otated Bibliography	55 55 <b>56</b> <b>57</b>
C D E	B.1 B.2 Solu Glos Inde	Notation and Style  Common Data Structures  utions to Selected Exercises  ssary of Terms  ex of Algorithms  otated Bibliography  Foundational Texts	55 5 <b>5</b> <b>56</b> <b>57</b> <b>58</b>

#### **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
- 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 ART of algorithmic analysis demands both mathematical precision and computational intuition. This part provides the rigorous mathematical foundation upon which all subsequent analysis techniques rest.

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

#### What Makes This Different:

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

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

— JOHN VON NEUMANN

#### Chapter 1

### Mathematical Foundations I: Discrete Structures

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

#### **Chapter Organization:**

This chapter covers five fundamental areas:

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

#### Each section includes:

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

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

#### 1.1 Logic and Proof Theory

Before we can prove algorithms correct or establish complexity bounds, we must understand the nature of proof itself. This section develops propositional and predicate logic, formal proof systems, and the art of mathematical argumentation.

Why This Matters: Every correctness proof, every lower bound, every impossibility result rests on logical foundations. Mastery of proof techniques is not optional—it is the essence of computer science.

#### 1.1.1 Propositional Logic

**Syntax: Well-Formed Formulas** 

Formal Definition.

Parse Trees and Unique Readability.

Inductive Definition of Formulas.

**Semantics: Truth Tables and Valuations** 

**Boolean Valuations.** 

**Truth-Functional Connectives:**  $\land$ ,  $\lor$ ,  $\neg$ ,  $\rightarrow$ ,  $\leftrightarrow$ .

Tautologies, Contradictions, and Contingencies.

**Logical Equivalence and Normal Forms** 

Substitution and Equivalence Relations.

Conjunctive Normal Form (CNF).

Disjunctive Normal Form (DNF).

Theorem: Every Formula Has Equivalent CNF/DNF.

**Proof Systems for Propositional Logic** 

Natural Deduction.

Sequent Calculus.

Resolution Principle and SAT Solving.

#### 1.1.2 Predicate Logic (First-Order Logic)

Syntax: Terms, Predicates, Quantifiers

Variables, Constants, Function Symbols.

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

Free vs. Bound Variables.

**Semantics: Structures and Models** 

Interpretations and Domains.

Satisfaction and Truth in a Model.

Validity and Logical Consequence.

Prenex Normal Form

Quantifier Scope and Movement Rules.

Skolemization and Herbrand's Theorem.

**Decidability and Undecidability** 

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

Decidable Fragments and Applications to Verification.

#### 1.1.3 Proof Techniques and Strategies

**Direct Proof** 

Structure and Template.

Example: Properties of Even/Odd Integers.

Proof by Contradiction (Reductio ad Absurdum)

Strategy and When to Use.

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

Example: Infinitude of Primes (Euclid).

**Proof by Contrapositive** 

Logical Equivalence to Direct Proof.

When Contrapositive is Easier.

**Mathematical Induction** 

Weak Induction Principle.

Strong Induction (Complete Induction).

Well-Ordering Principle and Equivalence.

Structural Induction on Recursive Data Types.

**Example: Correctness of Merge Sort.** 

**Proof by Construction** 

Existence Proofs: Constructive vs. Non-Constructive.

Algorithmic Content of Constructive Proofs.

**Proof by Counterexample** 

Disproving Universal Statements.

Minimal Counterexamples.

Pigeonhole Principle

Simple Form: n + 1 Objects in n Boxes.

**Generalized Form: Distribution Arguments.** 

**Example: Birthday Paradox.** 

**Example: Lower Bounds for Sorting.** 

#### 1.1.4 Common Proof Errors and How to Avoid Them

Circular Reasoning (Begging the Question)

**Proof by Example** 

**Confusion of Necessary and Sufficient Conditions** 

**Quantifier Errors** 

Vacuous Truth

#### 1.1.5 Advanced Topics in Logic

Temporal Logic and Program Verification

Modal Logic and Epistemic Logic

Gödel's Incompleteness Theorems (Informal)

#### 1.1.6 Exercises

Warmup Problems (20 problems)

Standard Problems (30 problems)

Challenging Problems (25 problems)

Research-Level Problems (★) (10 problems)

#### 1.2 Set Theory, Functions, and Relations

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

#### 1.2.1 Axiomatic Set Theory (Informal)

**Zermelo-Fraenkel Axioms (Intuitive Overview)** 

Russell's Paradox and the Need for Axioms

The Axiom of Choice and Its Consequences

#### 1.2.2 Basic Set Operations

**Set Builder Notation and Comprehension** 

Union, Intersection, Difference, Symmetric Difference

**Power Set and Cartesian Product** 

De Morgan's Laws and Duality

#### 1.2.3 Cardinality and Counting

Finite, Countable, and Uncountable Sets

Cantor's Diagonal Argument

Schröder-Bernstein Theorem

**Continuum Hypothesis** 

#### 1.2.4 Functions: Mappings Between Sets

**Definition and Notation** 

Domain, Codomain, Range.

Function Composition and Associativity.

**Types of Functions** 

Injective (One-to-One) Functions.

Surjective (Onto) Functions.

Bijective Functions and Inverses.

Theorem: Composition of Bijections is Bijective.

**Special Functions** 

**Identity Function.** 

8|59

**Constant Functions.** 

**Projection Functions.** 

Image and Preimage

Properties of Image and Preimage.

Inverse Image and Set Operations.

#### 1.2.5 Relations

**Binary Relations** 

**Definition as Subsets of Cartesian Products.** 

Representation: Matrices, Graphs, Tables.

**Properties of Relations** 

Reflexivity, Irreflexivity.

Symmetry, Antisymmetry, Asymmetry.

Transitivity.

**Equivalence Relations** 

Definition and Examples.

**Equivalence Classes and Partitions.** 

Theorem: Equivalence Relations Induce Partitions.

**Quotient Sets.** 

**Partial Orders** 

Definition: Reflexive, Antisymmetric, Transitive.

Hasse Diagrams.

Maximal and Minimal Elements.

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

Lattices.

**Total Orders (Linear Orders)** 

Definition and Examples.

Well-Orderings.

Application: Topological Sorting.

#### 1.2.6 Exercises

Warmup Problems

**Standard Problems** 

**Challenging Problems** 

Research Problems (\*)

#### 1.3 Combinatorial Analysis and Enumeration

Combinatorics is the mathematics of counting. It answers fundamental questions: How many possible inputs exist? How many steps must an algorithm take? What is the size of the search space?

This section goes far beyond basic permutations and combinations, developing generating functions, recurrence relations, and asymptotic enumeration techniques essential for algorithm analysis.

#### 1.3.1 Fundamental Counting Principles

**Sum Rule (Addition Principle)** 

**Product Rule (Multiplication Principle)** 

**Division Principle (Quotient Principle)** 

**Bijection Principle** 

#### 1.3.2 Permutations and Arrangements

Permutations of *n* Distinct Objects

Formula: n!

#### Stirling's Approximation (Preview).

#### **Permutations with Repetition**

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

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

#### **Circular Permutations**

(n-1)! Arrangements.

#### Derangements

Permutations with No Fixed Points.

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

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

#### 1.3.3 Combinations and Selections

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

Definition and Pascal's Triangle.

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

#### Vandermonde's Identity.

#### **Binomial Theorem**

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

#### Applications to Probability.

#### **Multinomial Coefficients**

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

Multinomial Theorem.

**Combinations with Repetition** 

Stars and Bars Method.

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

#### 1.3.4 Inclusion-Exclusion Principle

**Two-Set Formula** 

General Formula for n Sets

**Applications** 

**Counting Surjections.** 

**Euler's Totient Function.** 

Derangement Formula (Alternative Derivation).

#### 1.3.5 Pigeonhole Principle and Ramsey Theory

Simple Pigeonhole Principle

Generalized Pigeonhole Principle

Ramsey's Theorem (Introduction)

R(3,3) = 6 — The Party Problem.

**Bounds on** R(m, n).

#### 1.3.6 Generating Functions

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

**Ordinary Generating Functions (OGF)** 

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

**Operations on Generating Functions.** 

Convolution and Product of GFs.

**Exponential Generating Functions (EGF)** 

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

When to Use EGF vs. OGF.

Solving Recurrences with Generating Functions

Example: Fibonacci Numbers.

Example: Catalan Numbers.

**Coefficient Extraction** 

Cauchy's Residue Theorem (Overview).

Asymptotic Analysis via Singularities.

#### 1.3.7 Recurrence Relations

**Linear Recurrences with Constant Coefficients** 

Homogeneous Recurrences.

Characteristic Equation Method.

Example: Fibonacci, Tribonacci.

Non-Homogeneous Recurrences

Method of Undetermined Coefficients.

Particular Solutions.

**Divide-and-Conquer Recurrences** 

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

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

#### 1.3.8 Catalan Numbers

**Definition and Recurrence** 

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

**Applications** 

**Balanced Parentheses.** 

**Binary Search Trees.** 

First Edition • 2025

Triangulations of Polygons.

#### 1.3.9 Stirling Numbers

Stirling Numbers of the First Kind

Unsigned: Number of Permutations with k Cycles.

Signed: Coefficients in Falling Factorial Expansion.

Stirling Numbers of the Second Kind

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

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

**Bell Numbers** 

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

#### 1.3.10 Asymptotic Methods in Combinatorics

Stirling's Approximation (Full Treatment)

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

Proof via Euler-Maclaurin Formula.

More Precise Expansions.

**Asymptotic Analysis of Binomial Coefficients** 

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

**Saddle-Point Method (Introduction)** 

#### 1.3.11 Exercises

Warmup Problems (30 problems)

Standard Problems (40 problems)

Challenging Problems (30 problems)

**Research Problems (★) (15 problems)** 

#### 1.4 Graph Theory Foundations

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

#### 1.4.1 Basic Definitions and Representations

**Graphs: Formal Definition** 

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

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

Multigraphs and Pseudographs.

**Graph Terminology** 

Vertices (Nodes) and Edges (Arcs).

Degree: In-Degree, Out-Degree.

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

Isolated, Pendant, and Universal Vertices.

**Special Types of Graphs** 

Complete Graphs:  $K_n$ 

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

First Edition • 2025

Cycle Graphs:  $C_n$ 

Wheel Graphs:  $W_n$ 

**Hypercubes:**  $Q_n$ 

**Graph Representations** 

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

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

**Edge List.** 

Incidence Matrix.

Trade-offs: Space vs. Query Time.

#### 1.4.2 Connectivity and Paths

Walks, Trails, Paths, and Cycles

**Definitions and Distinctions.** 

Simple Paths vs. Paths with Repetition.

**Connected Graphs** 

**Connected Components.** 

Strong and Weak Connectivity in Digraphs.

Distance and Diameter

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

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

Radius and Center.

**Eulerian Paths and Circuits** 

**Euler's Theorem: Necessary and Sufficient Conditions.** 

First Edition • 2025 16 | 59

Fleury's Algorithm.

Applications: DNA Sequencing, Route Planning.

**Hamiltonian Paths and Circuits** 

Definition and Examples.

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

NP-Completeness (Preview).

#### 1.4.3 Trees and Forests

**Definition of Trees** 

Acyclic Connected Graphs.

**Equivalent Characterizations of Trees.** 

**Properties of Trees** 

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

Leaves and Internal Nodes.

**Rooted Trees** 

Parent, Child, Sibling, Ancestor, Descendant.

Height and Depth.

Binary Trees, *k*-ary Trees.

**Spanning Trees** 

Definition and Existence.

Number of Spanning Trees: Cayley's Formula.

Minimum Spanning Tree Problem (Preview).

First Edition • 2025

#### 1.4.4 Directed Acyclic Graphs (DAGs)

**Definition and Properties** 

**Topological Sorting** 

Algorithm and Correctness.

Applications: Task Scheduling, Precedence Constraints.

**Transitive Closure** 

Warshall's Algorithm.

#### 1.4.5 Graph Coloring

**Vertex Coloring** 

**Chromatic Number:**  $\chi(G)$ 

Bounds: Brooks' Theorem.

**Edge Coloring** 

**Chromatic Index:**  $\chi'(G)$ 

Vizing's Theorem.

**Applications** 

Register Allocation in Compilers.

**Scheduling Problems.** 

#### 1.4.6 Planar Graphs

**Definition and Examples** 

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

Kuratowski's Theorem

Applications: Circuit Design, Geographic Maps.

#### 1.4.7 Advanced Topics

**Matchings and Covers** 

Maximum Matching.

First Edition • 2025 18 | 59

König's Theorem for Bipartite Graphs.

**Network Flows (Introduction)** 

Max-Flow Min-Cut Theorem (Statement).

**Expander Graphs** 

Applications to Derandomization.

#### 1.4.8 Exercises

Warmup Problems (25 problems)

Standard Problems (35 problems)

Challenging Problems (25 problems)

Research Problems (\*) (10 problems)

#### 1.5 Algebraic Structures for Algorithm Design

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

#### **1.5.1** Groups

**Definition and Examples** 

Axioms: Closure, Associativity, Identity, Inverses.

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

Subgroups

Definition and Lagrange's Theorem.

Cyclic Groups

Generators and Order.

**Permutation Groups** 

Symmetric Group  $S_n$ .

Cayley's Theorem.

First Edition • 2025

**Group Homomorphisms** 

Kernels and Images.

#### 1.5.2 Rings and Fields

Rings

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

Units and Zero Divisors.

**Integral Domains.** 

**Fields** 

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

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

**Polynomial Arithmetic** 

Division Algorithm for Polynomials.

**Greatest Common Divisor.** 

#### 1.5.3 Applications to Cryptography

**RSA** and Group Theory

Diffie-Hellman Key Exchange

Elliptic Curve Cryptography (Overview)

#### 1.5.4 Applications to Coding Theory

**Linear Codes** 

Cyclic Codes and BCH Codes

#### 1.5.5 Exercises

Warmup Problems

**Standard Problems** 

**Challenging Problems** 

Research Problems (\*)

First Edition • 2025 20 | 59

## Mathematical Foundations II: Analysis and Probability

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

#### **Chapter Organization:**

This chapter covers four major areas:

- 1. **Mathematical Analysis** Limits, series, asymptotics
- 2. **Probability Theory** Measure-theoretic foundations to applications
- 3. **Information Theory** Entropy, coding, lower bounds
- 4. **Number Theory** Modular arithmetic, primality, factorization

#### What Sets This Apart:

- Complete proofs of major theorems (Stirling, Central Limit, Prime Number)
- Asymptotic methods beyond Big-O: Euler-Maclaurin, saddle-point
- Rigorous probability with measure theory foundations
- Algorithmic number theory with complexity analysis

**Prerequisites:** Single-variable calculus (derivatives, integrals, series). Basic probability at the level of a first course.

## Part II

Foundations of Algorithmic Analysis

## **Introduction to Algorithm Analysis**

3.1	What Is Algorithm Analysis?
3.1.1	Correctness vs. Efficiency
3.1.2	Resource Measures: Time, Space, Energy, I/O
3.1.3	The Need for Mathematical Models
3.2	The RAM Model of Computation
3.2.1	Basic Operations and Unit-Cost Assumption
3.2.2	Memory Access Model
3.2.3	Limitations and Extensions of the RAM Model
3.3	Measuring Algorithm Performance
3.3.1	Input Size and Problem Instances
3.3.2	Counting Basic Operations
3.3.3	Exact vs. Asymptotic Analysis
3.4	Overview of Complexity Classes
3.4.1	P, NP, NP-Complete, and NP-Hard (Brief Introduction)
3.4.2	Why We Focus on Polynomial-Time Algorithms

#### **Asymptotic Notation**

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

- 4.1.1 Why Exact Counts Are Often Impractical
- 4.1.2 Growth Rates and Scalability
- 4.2 Big-O Notation (O)
- 4.2.1 Formal Definition
- 4.2.2 Intuition: Upper Bounds
- 4.2.3 Common Functions and Their Growth Rates
- 4.2.4 Examples and Non-Examples
- 4.2.5 Properties of Big-O

**Transitivity** 

Addition and Multiplication Rules

Reflexivity and Asymmetry

- 4.3 Big-Omega Notation ( $\Omega$ )
- 4.3.1 Formal Definition
- 4.3.2 Intuition: Lower Bounds
- 4.3.3 Examples and Applications
- 4.3.4 Relationship Between O and  $\Omega$

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

## Recurrence Relations and Their Solutions

51	Introdu	iction .	to Ra	curronce	a Ral	ations
.T. I	Introdu	iction :	TO K &	currenc	9 K (1)	lations

- 5.1.1 What Are Recurrences?
- 5.1.2 Why They Arise in Algorithm Analysis
- 5.1.3 Examples from Divide-and-Conquer Algorithms

#### 5.2 The Substitution Method

- 5.2.1 Guessing the Solution
- 5.2.2 Proving by Induction
- 5.2.3 Examples: Mergesort, Binary Search
- **5.2.4** Strengthening the Inductive Hypothesis

#### 5.3 The Recursion-Tree Method

- 5.3.1 Visualizing the Recurrence
- 5.3.2 Summing Over Levels
- **5.3.3** Examples and Illustrations
- 5.3.4 Limitations and When to Use

#### 5.4 The Master Theorem

5.4.1 Statement of the Master Theorem (Standard Form)

**27**|59

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

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

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

### **Probabilistic Analysis of Algorithms**

<b>7.1</b>	<b>Foundations</b>	of Probabilistic A	Analysis

- 7.1.1 Random Variables in Algorithm Analysis
- 7.1.2 Indicator Random Variables
- 7.1.3 Linearity of Expectation

#### 7.2 Expected Running Time

- 7.2.1 Formal Definition
- 7.2.2 Computing Expectations via Indicator Variables
- 7.2.3 Examples: Hiring Problem, Randomized Quicksort

#### 7.3 Probabilistic Bounds

- 7.3.1 Markov's Inequality
- 7.3.2 Chebyshev's Inequality
- 7.3.3 Chernoff Bounds
- 7.3.4 Applications to Load Balancing and Hashing

#### 7.4 Randomized Algorithms

- 7.4.1 Randomized Quicksort (Detailed Analysis)
- 7.4.2 Randomized Selection (Quickselect)
- 7:4:3 · 202 Hashing and Universal Hash Functions

# Part III Advanced Analysis Techniques

## **Amortized Analysis**

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

**36**|59

#### Chapter 9

## **Space Complexity Analysis**

9.1	Introduction to Space Complexity
9.1.1	Why Space Matters
9.1.2	Types of Memory: Stack, Heap, Static
9.1.3	In-Place vs. Out-of-Place Algorithms
9.2	Measuring Space Usage
9.2.1	Auxiliary Space vs. Total Space
9.2.2	Recursive Call Stack Depth
9.2.3	Implicit vs. Explicit Data Structures
9.3	<b>Examples of Space Complexity Analysis</b>
<ul><li>9.3</li><li>9.3.1</li></ul>	
9.3.1	
9.3.1 9.3.2	Iterative Algorithms: Loops and Arrays
9.3.1 9.3.2 9.3.3	Iterative Algorithms: Loops and Arrays Recursive Algorithms: Mergesort, Quicksort
9.3.1 9.3.2 9.3.3	Iterative Algorithms: Loops and Arrays Recursive Algorithms: Mergesort, Quicksort Dynamic Programming: Memoization vs. Tabulation Graph Algorithms: BFS, DFS, Shortest Paths
9.3.1 9.3.2 9.3.3 9.3.4 9.4	Iterative Algorithms: Loops and Arrays Recursive Algorithms: Mergesort, Quicksort Dynamic Programming: Memoization vs. Tabulation Graph Algorithms: BFS, DFS, Shortest Paths

9.5 Streaming and Online Algorithms

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

**38**|59

### Chapter 10

## Cache-Aware and I/O Complexity

10.1	Introduction to the Memory Hierarchy		
10.1.1	Registers, Cache (L1, L2, L3), RAM, Disk		
10.1.2	Latency and Bandwidth Characteristics		
10.1.3	Why Algorithm Design Must Consider Memory		
10.2	The External Memory Model (I/O Model)		
10.2.1	Parameters: $N$ (data size), $M$ (memory size), $B$ (block size)		
10.2.2	I/O Complexity: Counting Block Transfers		
10.2.3	Comparison with RAM Model		
10.3	I/O-Efficient Algorithms		
10.3.1	Scanning and Sorting		
Externa	External Merge Sort		
I/O Con	<b>aplexity:</b> $O((N/B)\log_{M/B}(N/B))$		
10.3.2	Matrix Operations		
Matrix Transposition			
Matrix I	Multiplication		
10.3.3	Graph Algorithms		
I/O-Effi	cient BFS and DFS		

10.4 Cache-Oblivious Algorithms

Minimum Spanning Tree

## Cache-Aware Scheduling and Analysis for Multicores

11.1	Introduction	to Multicore	and Parallel	Computing
------	--------------	--------------	--------------	-----------

- 11.1.1 Shared vs. Distributed Memory
- 11.1.2 Parallel Models: PRAM, Fork-Join, Work-Stealing
- 11.1.3 Performance Metrics: Work, Span, Parallelism
- 11.2 Cache Coherence and Consistency
- 11.2.1 MESI and MOESI Protocols
- 11.2.2 False Sharing in Multicore Systems
- 11.2.3 Impact on Algorithm Design
- 11.3 Cache-Aware Parallel Algorithms
- 11.3.1 Parallel Sorting with Cache Awareness
- 11.3.2 Parallel Matrix Multiplication (Strassen, Coppersmith-Winograd)
- 11.3.3 Load Balancing and Task Granularity
- 11.4 Real-Time and Embedded Systems
- 11.4.1 WCET Analysis in Cache-Aware Contexts
- 11.4.2.2 Predictability vs. Average-Case Performance

# Part IV Lower Bounds and Optimality

## Lower Bounds for Comparison-Based Algorithms

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

12.5 Exercises

12.4.1 General Framework

**Adversary Arguments** 

**Examples: Merging, Element Uniqueness** 

## Algebraic and Non-Comparison Lower Bounds

13.1	Algebraic Decision Trees
13.1.1	<b>Extending Beyond Comparisons</b>
13.1.2	<b>Element Distinctness Lower Bound</b>
13.2	<b>Communication Complexity</b>
13.2.1	Models and Definitions
13.2.2	Applications to Data Structures
13.3	Cell-Probe Model
13.3.1	Lower Bounds for Data Structures
13.3.2	Dynamic vs. Static Data Structures
13.4	Exercises

## Part V

**Specialized Topics and Applications** 

## **Analysis of Specific Algorithm Paradigms**

14.1	Divide-and-Conquer Algorithms
14.1.1	General Framework and Recurrence Relations
14.1.2	Examples: Mergesort, Quicksort, Strassen's Algorithm
14.1.3	Optimality and Lower Bounds
14.2	Greedy Algorithms
14.2.1	Correctness via Exchange Arguments
14.2.2	Matroid Theory (Brief Introduction)
14.2.3	Examples: Huffman Coding, Kruskal's MST
14.3	Dynamic Programming
14.3.1	Optimal Substructure and Overlapping Subproblems
14.3.2	Memoization vs. Tabulation
14.3.3	Time and Space Complexity Analysis
14.3.4	Examples: Knapsack, Edit Distance, Matrix Chain Multipli-

cation

## Online Algorithms and Competitive Analysis

<b>15.1</b>	Introduction to Online Algorithms
15.1.1	Online vs. Offline Problems
15.1.2	Competitive Ratio
15.2	<b>Examples of Online Problems</b>
15.2.1	Paging and Caching (LRU, FIFO, LFU)
15.2.2	Load Balancing
15.2.3	Online Scheduling
15.3	<b>Competitive Analysis Techniques</b>
15.3.1	Deterministic vs. Randomized Algorithms
15.3.2	Lower Bounds via Adversary Arguments
<b>15.4</b>	Exercises

## **Approximation Algorithms**

16.1	Introduction to Approximation
16.1.1	NP-Hardness and Intractability
16.1.2	Approximation Ratios
16.2	<b>Examples of Approximation Algorithms</b>
16.2.1	Vertex Cover (2-Approximation)
16.2.2	<b>Set Cover (Greedy,</b> $\log n$ <b>-Approximation)</b>
16.2.3	Traveling Salesman Problem (Metric TSP)
16.3	Analysis Techniques
16.3.1	<b>Bounding Optimal Solutions</b>
16.3.2	Linear Programming Relaxations
16.4	Exercises

### **Parameterized Complexity**

- 17.1 Introduction to Parameterized Algorithms
- 17.1.1 Fixed-Parameter Tractability (FPT)
- 17.1.2 Kernelization
- 17.2 Examples and Analysis
- 17.2.1 Vertex Cover Parameterized by Solution Size
- 17.2.2 Treewidth and Graph Algorithms
- 17.3 W-Hierarchy and Hardness
- 17.3.1 W[1], W[2], and Beyond
- 17.4 Exercises

#### Part VI

## **Practical Considerations and Case Studies**

## **From Theory to Practice**

18.1	Hidden Constants and Lower-Order Terms
18.1.1	When $O(n \log n)$ Beats $O(n)$ in Practice
18.1.2	<b>Empirical Performance Measurements</b>
18.2	Algorithm Engineering
18.2.1	Profiling and Benchmarking
18.2.2	Tuning for Specific Hardware
18.2.3	Libraries and Implementations (STL, Boost, etc.)
18.3	Parallel and Distributed Algorithm Analysis
	Parallel and Distributed Algorithm Analysis Scalability and Speedup
18.3.1	
18.3.1 18.3.2	Scalability and Speedup
18.3.1 18.3.2 18.4	Scalability and Speedup  Amdahl's Law and Gustafson's Law
18.3.1 18.3.2 18.4 18.4.1	Scalability and Speedup  Amdahl's Law and Gustafson's Law  Energy Efficiency

### **Case Studies**

19.1	Sorting Algorithms in Practice
19.1.1	Timsort, Introsort, Radix Sort
19.1.2	Comparison of Theoretical vs. Empirical Performance
19.2	<b>Graph Algorithms in Large-Scale Systems</b>
19.2.1	Web Graphs and PageRank
19.2.2	Social Network Analysis
19.3	Machine Learning and Data Science
19.3.1	Complexity of Training Algorithms
19.3.2	SGD, AdaGrad, Adam: Time and Space Analysis
19.4	Database Systems
19.4.1	Query Optimization
19.4.2	Indexing Structures (B-Trees, LSM-Trees)
19.5	Exercises

### Appendix A

## **Mathematical Background**

- A.1 Summation Formulas
- A.2 Logarithms and Exponentials
- A.3 Recurrence Relations (Quick Reference)
- A.4 Probability Distributions
- A.5 Matrix Operations

## Appendix B

#### **Pseudocode Conventions**

- **B.1** Notation and Style
- **B.2** Common Data Structures

## Appendix C

### **Solutions to Selected Exercises**

# Appendix D Glossary of Terms

# Appendix E Index of Algorithms

## Appendix F

## **Annotated Bibliography**

- F.1 Foundational Texts
- F.2 Research Papers by Topic
- F.3 Online Courses and Resources