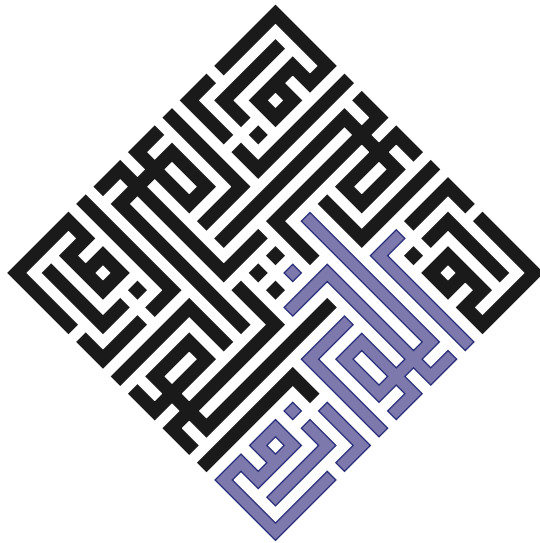


Algorithms



Jeff Erickson

0th edition (pre-publication draft) — December 30, 2018
½th edition (pre-publication draft) — April 9, 2019
1st paperback edition — June 13, 2019

1 2 3 4 5 6 7 8 9 — 27 26 25 24 23 22 21 20 19

ISBN: 978-1-792-64483-2 (paperback)

© Copyright 2019 Jeff Erickson



This work is available under a Creative Commons Attribution 4.0 International License.
For license details, see <http://creativecommons.org/licenses/by/4.0/>.

Download this book at <http://jeffe.cs.illinois.edu/teaching/algorithms/>
or <http://algorithms.wtf>
or <https://archive.org/details/Algorithms-Jeff-Erickson>

Please report errors at <https://github.com/jeffgerickson/algorithms>

Portions of our programming are mechanically reproduced,
and we now begin our broadcast day.

*For Kim, Kay, and Hannah
with love and admiration*

*And for Erin
with thanks
for breaking her promise*

Incipit prologus in libro algoarismi de practica arismetrice.

— Ioannis Hispalensis [John of Seville?],
Liber algorismi de pratica arismetrice (c.1135)

*Shall I tell you, my friend, how you will come to understand it?
Go and write a book upon it.*

— Henry Home, Lord Kames (1696–1782),
in a letter to Sir Gilbert Elliot

*The individual is always mistaken. He designed many things, and drew in other
persons as coadjutors, quarrelled with some or all, blundered much, and
something is done; all are a little advanced, but the individual is always mistaken.
It turns out somewhat new and very unlike what he promised himself.*

— Ralph Waldo Emerson, “Experience”, *Essays, Second Series* (1844)

*What I have outlined above is the content of a book the realization of whose basic
plan and the incorporation of whose details would perhaps be impossible; what I
have written is a second or third draft of a preliminary version of this book*

— Michael Spivak, preface of the first edition of
Differential Geometry, Volume I (1970)

Preface

About This Book

This textbook grew out of a collection of lecture notes that I wrote for various algorithms classes at the University of Illinois at Urbana-Champaign, which I have been teaching about once a year since January 1999. Spurred by changes of our undergraduate theory curriculum, I undertook a major revision of my notes in 2016; this book consists of a subset of my revised notes on the most fundamental course material, mostly reflecting the algorithmic content of our new required junior-level theory course.

Prerequisites

The algorithms classes I teach at Illinois have two significant prerequisites: a course on discrete mathematics and a course on fundamental data structures. Consequently, this textbook is probably not suitable for most students as a *first*

course in data structures and algorithms. In particular, I assume at least passing familiarity with the following specific topics:

- **Discrete mathematics:** High-school algebra, logarithm identities, naive set theory, Boolean algebra, first-order predicate logic, sets, functions, equivalences, partial orders, modular arithmetic, recursive definitions, trees (as abstract objects, not data structures), graphs (vertices and edges, not function plots).
- **Proof techniques:** direct, indirect, contradiction, exhaustive case analysis, and induction (especially “strong” and “structural” induction). Chapter 0 uses induction, and whenever Chapter $n-1$ uses induction, so does Chapter n .
- **Iterative programming concepts:** variables, conditionals, loops, records, indirection (addresses/pointers/references), subroutines, recursion. I do not assume fluency in any particular programming language, but I do assume experience with at least one language that supports both indirection and recursion.
- **Fundamental abstract data types:** scalars, sequences, vectors, sets, stacks, queues, maps/dictionaries, ordered maps/dictionaries, priority queues.
- **Fundamental data structures:** arrays, linked lists (single and double, linear and circular), binary search trees, at least one form of *balanced* binary search tree (such as AVL trees, red-black trees, treaps, skip lists, or splay trees), hash tables, binary heaps, and most importantly, the difference between this list and the previous list.
- **Fundamental computational problems:** elementary arithmetic, sorting, searching, enumeration, tree traversal (preorder, inorder, postorder, level-order, and so on).
- **Fundamental algorithms:** elementary algorithm, sequential search, binary search, sorting (selection, insertion, merge, heap, quick, radix, and so on), breadth- and depth-first search in (at least binary) trees, and most importantly, the difference between this list and the previous list.
- **Elementary algorithm analysis:** Asymptotic notation (o , O , Θ , Ω , ω), translating loops into sums and recursive calls into recurrences, evaluating simple sums and recurrences.
- **Mathematical maturity:** facility with abstraction, formal (especially recursive) definitions, and (especially inductive) proofs; writing and following mathematical arguments; recognizing and avoiding syntactic, semantic, and/or logical nonsense.

The book *briefly* covers some of this prerequisite material when it arises in context, but more as a reminder than a good introduction. For a more thorough overview, I strongly recommend the following freely available references:

- Margaret M. Fleck. *Building Blocks for Theoretical Computer Science*. Version 1.3 (January 2013) or later available from <http://mfleck.cs.illinois.edu/building-blocks/>.
- Eric Lehman, F. Thomson Leighton, and Albert R. Meyer. *Mathematics for Computer Science*. June 2018 revision available from <https://courses.csail.mit.edu/6.042/spring18/>. (I strongly recommend searching for the most recent revision.)
- Pat Morin. *Open Data Structures*. Edition 0.1G β (January 2016) or later available from <http://opendatastructures.org/>.
- Don Sheehy. *A Course in Data Structures and Object-Oriented Design*. February 2019 or later revision available from <https://donsheehy.github.io/datastructures/>.

Additional References

Please do not restrict yourself to this or any other single reference. Authors and readers bring their own perspectives to any intellectual material; no instructor “clicks” with every student, or even with every very strong student. Finding the author that most effectively gets *their* intuition into *your* head takes some effort, but that effort pays off handsomely in the long run.

The following references have been particularly valuable sources of intuition, examples, exercises, and inspiration; this is not meant to be a complete list.

- Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974. (I used this textbook as an undergraduate at Rice and again as a masters student at UC Irvine.)
- Boaz Barak. *Introduction to Theoretical Computer Science*. Textbook draft, most recently revised June 2019. (Not your grandfather’s theoretical CS textbook, and so much the better for it; the fact that it’s free is a delightful bonus.)
- Thomas Cormen, Charles Leiserson, Ron Rivest, and Cliff Stein. *Introduction to Algorithms*, third edition. MIT Press/McGraw-Hill, 2009. (I used the first edition as a teaching assistant at Berkeley.)
- Sanjoy Dasgupta, Christos H. Papadimitriou, and Umesh V. Vazirani. *Algorithms*. McGraw-Hill, 2006. (Probably the closest in content to this book, but considerably less verbose.)
- Jeff Edmonds. *How to Think about Algorithms*. Cambridge University Press, 2008.
- Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

- Michael T. Goodrich and Roberto Tamassia. *Algorithm Design: Foundations, Analysis, and Internet Examples*. John Wiley & Sons, 2002.
- Jon Kleinberg and Éva Tardos. *Algorithm Design*. Addison-Wesley, 2005. Borrow it from the library if you can.
- Donald Knuth. *The Art of Computer Programming*, volumes 1–4A. Addison-Wesley, 1997 and 2011. (My parents gave me the first three volumes for Christmas when I was 14. Alas, I didn't actually read them until *much* later.)
- Udi Manber. *Introduction to Algorithms: A Creative Approach*. Addison-Wesley, 1989. (I used this textbook as a teaching assistant at Berkeley.)
- Ian Parberry. *Problems on Algorithms*. Prentice-Hall, 1995 (out of print). Downloadable from <https://larc.unt.edu/ian/books/free/license.html> after you agree to make a small charitable donation. Please honor your agreement.
- Robert Sedgwick and Kevin Wayne. *Algorithms*. Addison-Wesley, 2011.
- Robert Endre Tarjan. *Data Structures and Network Algorithms*. SIAM, 1983.
- Class notes from my own algorithms classes at Berkeley, especially those taught by Dick Karp and Raimund Seidel.
- Lecture notes, slides, homeworks, exams, video lectures, research papers, blog posts, StackExchange questions and answers, podcasts, and full-fledged MOOCs made freely available on the web by innumerable colleagues around the world.

About the Exercises

Each chapter ends with several exercises, most of which I have used at least once in a homework assignment, discussion/lab section, or exam. The exercises are *not* ordered by increasing difficulty, but (generally) clustered by common techniques or themes. Some problems are annotated with symbols as follows:

- ♥ Red hearts indicate particularly challenging problems; many of these have appeared on qualifying exams for PhD students at Illinois. A small number of *really* hard problems are marked with ♥ large hearts.
- ♦ Blue diamonds indicate problems that require familiarity with material from later chapters, but thematically belong where they are. Problems that require familiarity with *earlier* material are not marked, however; the book, like life, is cumulative.
- ♣ Green clubs indicate problems that require familiarity with material outside the scope of this book, such as finite-state machines, linear algebra, probability, or planar graphs. These are rare.
- ♠ Black spades indicate problems that require a significant amount of grunt work and/or coding. These are rare.

- ★ Orange stars indicate that you are eating Lucky Charms that were manufactured before 1998. Ew.

These exercises are designed as opportunities to practice, not as targets for their own sake. The goal of each problem is not to solve that specific problem, but to exercise a certain set of skills, or to practice solving a certain *type* of problem. Partly for this reason, I don't provide solutions to the exercises; the solutions are not the point. In particular, there is no "instructor's manual"; if you can't solve a problem yourself, you probably shouldn't assign it to your students. That said, you can probably find solutions to whatever homework problems I've assigned *this* semester on the web page of whatever course I'm teaching. And nothing is stopping *you* from writing an instructor's manual!

Steal This Book!

This book is published under a Creative Commons Licence that allows you to use, redistribute, adapt, and remix its contents *without my permission*, as long as you point back to the original source. A complete electronic version of this book is freely available at any of the following locations:

- The book web site: <http://jeffe.cs.illinois.edu/teaching/algorithms/>
- The mnemonic shortcut: <http://algorithms.wtf>
- The bug-report site: <https://github.com/jeffgerickson/algorithms>
- The Internet Archive: <https://archive.org/details/Algorithms-Jeff-Erickson>

The book web site also contains several hundred pages of additional lecture notes on related and more advanced material, as well as a near-complete archive of past homeworks, exams, discussion/lab problems, and other teaching resources. Whenever I teach an algorithms class, I revise, update, and sometimes cull my teaching materials, so you may find more recent revisions on the web page of whatever course I am currently teaching.

Whether you are a student or an instructor, you are more than welcome to use any subset of this textbook or my other lecture notes in your own classes, without asking my permission—that's why I put them on the web! However, please also cite this book, either by name or with a link back to <http://algorithms.wtf>; this is *especially* important if you are a student, and you use my course materials to help with your homework. (Please also check with your instructor.)

However, if you are an instructor, I strongly encourage you to supplement these with additional material *that you write yourself*. Writing the material yourself will strengthen your mastery and in-class presentation of the material, which will in turn improve your students' mastery of the material. It will also get you past the frustration of dealing with the parts of this book that you don't like. All textbooks are ~~erap~~ imperfect, and this one is no exception.

Finally, **please make whatever you write freely, easily, and globally available on the open web**—not hidden behind the gates of a learning management system or some other type of paywall—so that students and instructors elsewhere can benefit from your unique insights. In particular, if you develop useful resources that directly complement this textbook, such as slides, videos, or solution manuals, please let me know so that I can add links to your resources from the book web site.

Acknowledgments

This textbook draws heavily on the contributions of countless algorithms students, teachers, and researchers. In particular, I am immensely grateful to more than three thousand Illinois students who have used my lecture notes as a primary reference, offered useful (if sometimes painful) criticism, and suffered through some truly awful early drafts. Thanks also to many colleagues and students around the world who have used these notes in their own classes and have sent helpful feedback and bug reports.

I am particularly grateful for the feedback and contributions (especially exercises) from my amazing teaching assistants:

Aditya Ramani, Akash Gautam, Alex Steiger, Alina Ene, Amir Nayyeri, Asha Seetharam, Ashish Vulimiri, Ben Moseley, Brad Sturt, Brian Ensink, Chao Xu, Charlie Carlson, Chris Neihengen, Connor Clark, Dan Bullok, Dan Cranston, Daniel Khashabi, David Morrison, Ekta Manaktala, Erin Wolf Chambers, Gail Steitz, Gio Kao, Grant Czajkowski, Hsien-Chih Chang, Igor Gammer, Jacob Laurel, John Lee, Johnathon Fischer, Junqing Deng, Kent Quanrud, Kevin Milans, Kevin Small, Konstantinos Koiliaris, Kyle Fox, Kyle Jao, Lan Chen, Mark Idleman, Michael Bond, Mitch Harris, Naveen Arivazhagen, Nick Bachmair, Nick Hurlburt, Nirman Kumar, Nitish Korula, Patrick Lin, Phillip Shih, Rachit Agarwal, Reza Zamani-Nasab, Rishi Talreja, Rob McCann, Sahand Mozaffari, Shalan Naqvi, Shripad Thite, Spencer Gordon, Srihita Vatsavaya, Subhro Roy, Tana Wattanawaroon, Umang Mathur, Vipul Goyal, Yasu Furakawa, and Yipu Wang.

I've also been helped tremendously by many discussions with faculty colleagues at Illinois: Alexandra Kolla, Cinda Heeren, Edgar Ramos, Herbert Edelsbrunner, Jason Zych, Kim Whittlesey, Lenny Pitt, Madhu Parasarathy, Mahesh Viswanathan, Margaret Fleck, Shang-Hua Teng, Steve LaValle, and especially Chandra Chekuri, Ed Reingold, and Sariel Har-Peled.

Of course this book owes a great debt to the people who taught me this algorithms stuff in the first place: Bob Bixby and Michael Pearlman at Rice; David Eppstein, Dan Hirschberg, and George Lueker at Irvine; and Abhiram Ranade, Dick Karp, Manuel Blum, Mike Luby, and Raimund Seidel at Berkeley.

I stole the first iteration of the overall course structure, and the idea to write up my own lecture notes in the first place, from Herbert Edelsbrunner; the idea of turning a subset of my notes into a book from Steve LaValle; and several components of the book design from Robert Ghrist.

Caveat Lector!

Of course, none of those people should be blamed for any flaws in the resulting book. Despite many rounds of revision and editing, this book contains several mistakes, bugs, gaffes, omissions, snafus, kludges, typos, mathos, grammaros, thinkos, brain farts, poor design decisions, historical inaccuracies, anachronisms, inconsistencies, exaggerations, dithering, blather, distortions, oversimplifications, redundancy, logorrhea, nonsense, garbage, cruft, junk, and outright lies, **all of which are entirely Steve Skiena's fault.**

I maintain an issue tracker at <https://github.com/jeffgerickson/algorithms>, where readers like you can submit bug reports, feature requests, and general feedback on the book. Please let me know if you find an error of any kind, whether mathematical, grammatical, historical, typographical, cultural, or otherwise, whether in the main text, in the exercises, or in my other course materials. (Steve is unlikely to care.) Of course, all other feedback is also welcome!

Enjoy!

— Jeff

It is traditional for the author to magnanimously accept the blame for whatever deficiencies remain. I don't. Any errors, deficiencies, or problems in this book are somebody else's fault, but I would appreciate knowing about them so as to determine who is to blame.

— Steven S. Skiena, *The Algorithm Design Manual* (1997)

No doubt this statement will be followed by an annotated list of all textbooks, and why each one is crap.

— Adam Contini, *MetaFilter*, January 4, 2010

Table of Contents

Preface	i
About This Book	i
Prerequisites	i
Additional References	iii
About the Exercises	iv
Steal This Book!	v
Acknowledgments	vi
Caveat Lector!	vii
Table of Contents	ix
o Introduction	1
o.1 What is an algorithm?	1
o.2 Multiplication	3

	Lattice Multiplication • Duplation and Mediation • Compass and Straight-edge	
0.3	Congressional Apportionment	8
0.4	A Bad Example	10
0.5	Describing Algorithms	11
	Specifying the Problem • Describing the Algorithm	
0.6	Analyzing Algorithms	14
	Correctness • Running Time	
	Exercises	17
1	Recursion	21
1.1	Reductions	21
1.2	Simplify and Delegate	22
1.3	Tower of Hanoi	24
1.4	Mergesort	26
	Correctness • Analysis	
1.5	Quicksort	29
	Correctness • Analysis	
1.6	The Pattern	31
1.7	Recursion Trees	31
	♥Ignoring Floors and Ceilings Is Okay, Honest	
1.8	♥Linear-Time Selection	35
	Quickselect • Good pivots • Analysis • Sanity Checking	
1.9	Fast Multiplication	40
1.10	Exponentiation	42
	Exercises	44
2	Backtracking	71
2.1	N Queens	71
2.2	Game Trees	74
2.3	Subset Sum	76
	Correctness • Analysis • Variants	
2.4	The General Pattern	79
2.5	Text Segmentation (<i>Interpunctio Verborum</i>)	80
	Index Formulation • ♥Analysis • Variants	
2.6	Longest Increasing Subsequence	86
2.7	Longest Increasing Subsequence, Take 2	89
2.8	Optimal Binary Search Trees	91
	♥Analysis	
	Exercises	93
3	Dynamic Programming	97

3.1	Mātrāvṛtta	97
	Backtracking Can Be Slow • Memo(r)ization: Remember Everything • Dynamic Programming: Fill Deliberately • Don't Remember Everything After All	
3.2	♥Aside: Even Faster Fibonacci Numbers	103
	Whoa! Not so fast!	
3.3	<i>Interpunctio Verborum Redux</i>	105
3.4	The Pattern: Smart Recursion	105
3.5	Warning: Greed is Stupid	107
3.6	Longest Increasing Subsequence	109
	First Recurrence: Is This Next? • Second Recurrence: What's Next?	
3.7	Edit Distance	111
	Recursive Structure • Recurrence • Dynamic Programming	
3.8	Subset Sum	116
3.9	Optimal Binary Search Trees	117
3.10	Dynamic Programming on Trees	120
	Exercises	123
4	Greedy Algorithms	159
4.1	Storing Files on Tape	159
4.2	Scheduling Classes	161
4.3	General Pattern	164
4.4	Huffman Codes	165
4.5	Stable Matching	170
	Some Bad Ideas • The Boston Pool and Gale-Shapley Algorithms • Running Time • Correctness • Optimality!	
	Exercises	176
5	Basic Graph Algorithms	187
5.1	Introduction and History	187
5.2	Basic Definitions	190
5.3	Representations and Examples	192
5.4	Data Structures	195
	Adjacency Lists • Adjacency Matrices • Comparison	
5.5	Whatever-First Search	199
	Analysis	
5.6	Important Variants	201
	Stack: Depth-First • Queue: Breadth-First • Priority Queue: Best-First • Disconnected Graphs • Directed Graphs	
5.7	Graph Reductions: Flood Fill	205
	Exercises	207

6	Depth-First Search	225
6.1	Preorder and Postorder	227
	Classifying Vertices and Edges	
6.2	Detecting Cycles	231
6.3	Topological Sort	232
	Implicit Topological Sort	
6.4	Memoization and Dynamic Programming	234
	Dynamic Programming in Dags	
6.5	Strong Connectivity	237
6.6	Strong Components in Linear Time	238
	Kosaraju and Sharir’s Algorithm • ♥Tarjan’s Algorithm	
	Exercises	244
7	Minimum Spanning Trees	257
7.1	Distinct Edge Weights	257
7.2	The Only Minimum Spanning Tree Algorithm	259
7.3	Borůvka’s Algorithm	261
	This is the MST Algorithm You Want	
7.4	Jarník’s (“Prim’s”) Algorithm	263
	♥Improving Jarník’s Algorithm	
7.5	Kruskal’s Algorithm	265
	Exercises	268
8	Shortest Paths	273
8.1	Shortest Path Trees	274
8.2	♥Negative Edges	274
8.3	The Only SSSP Algorithm	276
8.4	Unweighted Graphs: Breadth-First Search	278
8.5	Directed Acyclic Graphs: Depth-First Search	282
8.6	Best-First: Dijkstra’s Algorithm	284
	No Negative Edges • ♥Negative Edges	
8.7	Relax ALL the Edges: Bellman-Ford	289
	Moore’s Improvement • Dynamic Programming Formulation	
	Exercises	297
9	All-Pairs Shortest Paths	309
9.1	Introduction	309
9.2	Lots of Single Sources	310
9.3	Reweighting	311
9.4	Johnson’s Algorithm	312
9.5	Dynamic Programming	313
9.6	Divide and Conquer	315

9.7	Funny Matrix Multiplication	316
9.8	(Kleene-Roy-)Floyd-Warshall(-Ingerman)	318
	Exercises	320
10	Maximum Flows & Minimum Cuts	327
10.1	Flows	328
10.2	Cuts	329
10.3	The Maxflow-Mincut Theorem	331
10.4	Ford and Fulkerson's augmenting-path algorithm	334
	♥ Irrational Capacities	
10.5	Combining and Decomposing Flows	336
10.6	Edmonds and Karp's Algorithms	340
	Fattest Augmenting Paths • Shortest Augmenting Paths	
10.7	Further Progress	343
	Exercises	344
11	Applications of Flows and Cuts	353
11.1	Edge-Disjoint Paths	353
11.2	Vertex Capacities and Vertex-Disjoint Paths	354
11.3	Bipartite Matching	355
11.4	Tuple Selection	357
	Exam Scheduling	
11.5	Disjoint-Path Covers	360
	Minimal Faculty Hiring	
11.6	Baseball Elimination	363
11.7	Project Selection	366
	Exercises	368
12	NP-Hardness	379
12.1	A Game You Can't Win	379
12.2	P versus NP	381
12.3	NP-hard, NP-easy, and NP-complete	382
12.4	♥ Formal Definitions (<i>HC SVNT DRACONES</i>)	384
12.5	Reductions and SAT	385
12.6	3SAT (from CIRCUITSAT)	388
12.7	Maximum Independent Set (from 3SAT)	390
12.8	The General Pattern	392
12.9	Clique and Vertex Cover (from Independent Set)	394
12.10	Graph Coloring (from 3SAT)	395
12.11	Hamiltonian Cycle	398
	From Vertex Cover • From 3SAT • Variants and Extensions	
12.12	Subset Sum (from Vertex Cover)	402

Caveat Reductor!	
12.13 Other Useful NP-hard Problems	404
12.14 Choosing the Right Problem	407
12.15 A Frivolous Real-World Example	408
12.16 ♥On Beyond Zebra	412
Polynomial Space • Exponential Time • Excelsior!	
Exercises	415
Index	442
Index of People	446
Index of Pseudocode	449
Image Credits	451
Colophon	453