

ALGORITHMIC THINKING

A Problem-Based Introduction

by Daniel Zingaro



**no starch
press**

San Francisco

ALGORITHMIC THINKING. Copyright © 2021 by Daniel Zingaro.

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13: 978-1-7185-0080-8 (print)

ISBN-13: 978-1-7185-0081-5 (ebook)

Publisher: Bill Pollock

Executive Editor: Barbara Yien

Production Editor: Kassie Andreadis

Developmental Editor: Alex Freed

Interior Design: Octopod Studios

Cover Illustration: Rob Gale

Technical Reviewer: Larry Yueli Zhang

Copyeditor: David Couzens

Proofreader: James Fraleigh

For information on book distributors or translations, please contact No Starch Press, Inc. directly:

No Starch Press, Inc.

245 8th Street, San Francisco, CA 94103

phone: 1-415-863-9900; info@nostarch.com

www.nostarch.com

Library of Congress Cataloging-in-Publication Data

Names: Zingaro, Daniel, author.

Title: Algorithmic thinking : a problem-based introduction / by Daniel Zingaro.

Description: San Francisco : No Starch Press, [2021] | Includes bibliographical references and index.

Identifiers: LCCN 2020031510 (print) | LCCN 2020031511 (ebook) | ISBN

9781718500808 (paperback) | ISBN 1718500807 (paperback) | ISBN

9781718500815 (ebook)

Subjects: LCSH: Computer algorithms-Problems, exercises, etc. | Computer programming-Problems, exercises, etc.

Classification: LCC QA76.9.A43 Z56 2020 (print) | LCC QA76.9.A43 (ebook) | DDC 005.13-dc23

LC record available at <https://lcn.loc.gov/2020031510>

LC ebook record available at <https://lcn.loc.gov/2020031511>

No Starch Press and the No Starch Press logo are registered trademarks of No Starch Press, Inc. Other product and company names mentioned herein may be the trademarks of their respective owners. Rather than use a trademark symbol with every occurrence of a trademarked name, we are using the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The information in this book is distributed on an “As Is” basis, without warranty. While every precaution has been taken in the preparation of this work, neither the author nor No Starch Press, Inc. shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in it.

To Doyali

About the Author

Dr. Daniel Zingaro is an assistant teaching professor of computer science and award-winning teacher at the University of Toronto. His main area of research is computer science education research, where he studies how students learn (and sometimes don't learn) computer science material.

About the Technical Reviewer

Larry Yueli Zhang is an Assistant Professor, Teaching Stream of Computer Science at the University of Toronto Mississauga. His teaching and research interests include algorithms, data structures, operating systems, computer networks, social networks, and computing education. He has been in the program committees of ACM SIGCSE, ACM ITiCSE, and WCCCE. He holds a PhD in Computer Science from the University of Toronto.

BRIEF CONTENTS

Foreword	xv
Acknowledgments	xvii
Introduction	xix
Chapter 1: Hash Tables	1
Chapter 2: Trees and Recursion	31
Chapter 3: Memoization and Dynamic Programming	71
Chapter 4: Graphs and Breadth-First Search	119
Chapter 5: Shortest Paths in Weighted Graphs	165
Chapter 6: Binary Search	197
Chapter 7: Heaps and Segment Trees	243
Chapter 8: Union-Find	295
Afterword	339
Appendix A: Algorithm Runtime	341
Appendix B: Because I Can't Resist	347
Appendix C: Problem Credits	361
Index	365

CONTENTS IN DETAIL

FOREWORD	xv
-----------------	-----------

ACKNOWLEDGMENTS	xvii
------------------------	-------------

INTRODUCTION	xix
---------------------	------------

Online Resources	xx
Who This Book Is For	xx
The Programming Language	xx
Why Use C?	xxi
Static Keyword	xxi
Include Files	xxii
Freeing Memory	xxii
Topics	xxii
Judges	xxiii
Anatomy of a Problem Description	xxv
Problem: Food Lines	xxvi
The Problem	xxvi
Solving the Problem	xxvii
Notes	xxix

1	
HASH TABLES	1

Problem 1: Unique Snowflakes	1
The Problem	2
Simplifying the Problem	3
Solving the Core Problem	4
Solution 1: Pairwise Comparisons	7
Solution 2: Doing Less Work	11
Hash Tables	16
Hash Table Design	16
Why Use Hash Tables?	19
Problem 2: Compound Words	19
The Problem	19
Identifying Compound Words	20
Solution	20

Problem 3: Spelling Check: Deleting a Letter	24
The Problem	24
Thinking About Hash Tables	25
An Ad Hoc Solution	27
Summary	30
Notes.....	30

2

TREES AND RECURSION

31

Problem 1: Halloween Haul	31
The Problem	32
Binary Trees	33
Solving the Sample Instance	35
Representing Binary Trees	35
Collecting All the Candy	39
A Completely Different Solution	45
Walking the Minimum Number of Streets	50
Reading the Input	53
Why Use Recursion?	59
Problem 2: Descendant Distance	59
The Problem	59
Reading the Input	62
Number of Descendants from One Node	65
Number of Descendants from All Nodes	67
Sorting Nodes	67
Outputting the Information	68
The main Function	69
Summary	70
Notes.....	70

3

MEMOIZATION AND DYNAMIC PROGRAMMING

71

Problem 1: Burger Fervor	72
The Problem	72
Forming a Plan	72
Characterizing Optimal Solutions	74
Solution 1: Recursion	75
Solution 2: Memoization	80
Solution 3: Dynamic Programming	85
Memoization and Dynamic Programming	88
Step 1: Structure of Optimal Solutions	88
Step 2: Recursive Solution	89
Step 3: Memoization	90
Step 4: Dynamic Programming	90

5	SHORTEST PATHS IN WEIGHTED GRAPHS	165
Problem 1: Mice Maze		166
The Problem		166
Moving On from BFS		166
Shortest Paths in Weighted Graphs		168
Building the Graph		171
Implementing Dijkstra's Algorithm		173
Two Optimizations		175
Dijkstra's Algorithm		177
Runtime of Dijkstra's Algorithm		177
Negative-Weight Edges		178
Problem 2: Grandma Planner		180
The Problem		180
Adjacency Matrix		181
Building the Graph		182
Weird Paths		184
Task 1: Shortest Paths		187
Task 2: Number of Shortest Paths		189
Summary		196
Notes		196

6	BINARY SEARCH	197
Problem 1: Feeding Ants		197
The Problem		198
A New Flavor of Tree Problem		199
Reading the Input		201
Testing Feasibility		203
Searching for a Solution		205
Binary Search		206
Runtime of Binary Search		207
Determining Feasibility		208
Searching a Sorted Array		208
Problem 2: River Jump		209
The Problem		209
A Greedy Idea		210
Testing Feasibility		212
Searching for a Solution		216
Reading the Input		219
Problem 3: Living Quality		220
The Problem		220
Sorting Every Rectangle		221
Binary Search		224
Testing Feasibility		225
Testing Feasibility More Quickly		227

Problem 4: Cave Doors	233
The Problem	233
Solving a Subtask	234
Using a Linear Search	236
Using Binary Search	238
Summary	240
Notes	241

7

HEAPS AND SEGMENT TREES 243

Problem 1: Supermarket Promotion	243
The Problem	243
Solution 1: Maximum and Minimum in an Array	244
Max-Heaps	248
Min-Heaps	259
Solution 2: Heaps	261
Heaps	263
Two More Applications	264
Choosing a Data Structure	265
Problem 2: Building Treaps	265
The Problem	265
Recursively Outputting Treaps	267
Sorting by Label	268
Solution 1: Recursion	269
Range Maximum Queries	272
Segment Trees	273
Solution 2: Segment Trees	281
Segment Trees	282
Problem 3: Two Sum	283
The Problem	283
Filling the Segment Tree	284
Querying the Segment Tree	288
Updating the Segment Tree	290
The main Function	293
Summary	294
Notes	294

8

UNION-FIND 295

Problem 1: Social Network	296
The Problem	296
Modeling as a Graph	297
Solution 1: BFS	300
Union-Find	304
Solution 2: Union-Find	307

Optimization 1: Union by Size	310
Optimization 2: Path Compression	314
Union-Find	316
Relationships: Three Requirements	316
Choosing Union-Find	317
Optimizations	317
Problem 2: Friends and Enemies	317
The Problem	318
Augmentation: Enemies	319
The main Function	323
Find and Union	324
SetFriends and SetEnemies	325
AreFriends and AreEnemies	327
Problem 3: Drawer Chore	328
The Problem	328
Equivalent Drawers	329
The main Function	334
Find and Union	335
Summary	336
Notes	337

AFTERWORD

339

A

ALGORITHM RUNTIME

341

The Case for Timing ... and Something Else	341
Big O Notation	343
Linear Time	343
Constant Time	344
Another Example	345
Quadratic Time	345
Big O in This Book	346

B

BECAUSE I CAN'T RESIST

347

Unique Snowflakes: Implicit Linked Lists	347
Burger Fervor: Reconstructing a Solution	350
Knight Chase: Encoding Moves	352
Dijkstra's Algorithm: Using a Heap	354
Mice Maze: Tracing with Heaps	354
Mice Maze: Implementation with Heaps	357

Compressing Path Compression	358
Step 1: No More Ternary If	358
Step 2: Cleaner Assignment Operator	359
Step 3: Understand the Recursion	360

C	
PROBLEM CREDITS	361

INDEX	365
--------------	------------