

## Competitive Programming in Python

Want to kill it at your job interview in the tech industry? Want to win that coding competition? Learn all the algorithmic techniques and programming skills you need from two experienced coaches, problem-setters, and judges for coding competitions. The authors highlight the versatility of each algorithm by considering a variety of problems and show how to implement algorithms in simple and efficient code. What to expect:

- \* Master 128 algorithms in Python.
- \* Discover the right way to tackle a problem and quickly implement a solution of low complexity.
- \* Understand classic problems like Dijkstra's shortest path algorithm and Knuth–Morris–Pratt's string matching algorithm, plus lesser-known data structures like Fenwick trees and Knuth's dancing links.
- \* Develop a framework to tackle algorithmic problem solving, including: Definition, Complexity, Applications, Algorithm, Key Information, Implementation, Variants, In Practice, and Problems.
- \* Python code included in the book and on the companion website.

**Christoph Dürr** is a senior researcher at the French National Center for Scientific Research (CNRS), affiliated with the Sorbonne University in Paris. After a PhD in 1996 at Paris-Sud University, he worked for one year as a postdoctoral researcher at the International Computer Science Institute in Berkeley and one year in the School of Computer Science and Engineering in the Hebrew University of Jerusalem in Israel. He has worked in the fields of quantum computation, discrete tomography and algorithmic game theory, and his current research activity focuses on algorithms and optimisation. From 2007 to 2014, he taught a preparation course for programming contests at the engineering school École Polytechnique, and acts regularly as a problem setter, trainer, or competitor for various coding competitions. In addition, he loves carrot cake.

**Jill-Jênn Vie** is a research scientist at Inria in machine learning. He is an alumnus from École normale supérieure Paris-Saclay, where he founded the algorithmic club of Paris-Saclay (CAPS) and coached several teams for the International Collegiate Programming Contest (ICPC). He published a book in theoretical computer science to help students prepare for prestigious French competitive exams such as *Grandes Écoles* or *agrégation*, and directed a television show “Blame the Algorithm” about the algorithms that govern our lives. He is part of the advisory board of the French Computer Science Society (SIF), itself a member of the International Federation for Information Processing (IFIP).

Cambridge University Press  
978-1-108-71682-6 — Competitive Programming in Python  
Christoph Dürr , Jill-Jênn Vie , Translated by Greg Gibbons , Danièle Gibbons  
Frontmatter  
[More Information](#)

---

# Competitive Programming in Python

128 Algorithms to Develop Your Coding Skills

CHRISTOPH DÜRR

CNRS, Sorbonne University

JILL-JÊNN VIE

Inria

Translated by Greg Gibbons and Danièle Gibbons



CAMBRIDGE  
UNIVERSITY PRESS

Cambridge University Press

978-1-108-71682-6 — Competitive Programming in Python

Christoph Dürr , Jill-Jënn Vie , Translated by Greg Gibbons , Danièle Gibbons

Frontmatter

[More Information](#)

## CAMBRIDGE UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre, New Delhi – 110025, India

79 Anson Road, #06–04/06, Singapore 079906

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning, and research at the highest international levels of excellence.

[www.cambridge.org](http://www.cambridge.org)

Information on this title: [www.cambridge.org/9781108716826](http://www.cambridge.org/9781108716826)

DOI: 10.1017/9781108591928

© Cambridge University Press 2021

Translation from the French language edition:

*Programmation efficace - 128 algorithmes qu'il faut avoir compris et codés en Python au cours de sa vie*

By Christoph Dürr & Jill-Jënn Vie

Copyright © 2016 Edition Marketing S.A.

[www.editions-ellipses.fr](http://www.editions-ellipses.fr)

All Rights Reserved

This publication is in copyright. Subject to statutory exception and to the provisions of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published 2021

Printed in the United Kingdom by TJ Books Limited, Padstow Cornwall

*A catalogue record for this publication is available from the British Library.*

*Library of Congress Cataloging-in-Publication Data*

Names: Dürr, Christoph, 1969– author. | Vie, Jill-Jënn, 1990– author. |

Gibbons, Greg, translator. | Gibbons, Danièle, translator.

Title: Competitive programming in Python : 128 algorithms to develop your coding skills / Christoph Dürr, Jill-Jënn Vie ; translated by Greg Gibbons, Danièle Gibbons.

Other titles: Programmation efficace. English

Description: First edition. | New York : Cambridge University Press, 2020.

| Includes bibliographical references and index.

Identifiers: LCCN 2020022774 (print) | LCCN 2020022775 (ebook) |

ISBN 9781108716826 (paperback) | ISBN 9781108591928 (epub)

Subjects: LCSH: Python (Computer program language) | Algorithms.

Classification: LCC QA76.73.P98 D8713 2020 (print) | LCC QA76.73.P98 (ebook) | DDC 005.13/3–dc23

LC record available at <https://lcn.loc.gov/2020022774>

LC ebook record available at <https://lcn.loc.gov/2020022775>

ISBN 978-1-108-71682-6 Paperback

Cambridge University Press has no responsibility for the persistence or accuracy of URLs for external or third-party internet websites referred to in this publication and does not guarantee that any content on such websites is, or will remain, accurate or appropriate.

Contents

	<i>Preface</i>	<i>page ix</i>
1	<b>Introduction</b>	1
	1.1 Programming Competitions	1
	1.2 Python in a Few Words	5
	1.3 Input-Output	13
	1.4 Complexity	17
	1.5 Abstract Types and Essential Data Structures	20
	1.6 Techniques	28
	1.7 Advice	37
	1.8 A Problem: ‘Frosting on the Cake’	39
2	<b>Character Strings</b>	42
	2.1 Anagrams	42
	2.2 T9—Text on 9 Keys	43
	2.3 Spell Checking with a Lexicographic Tree	46
	2.4 Searching for Patterns	48
	2.5 Maximal Boundaries—Knuth–Morris–Pratt	49
	2.6 Pattern Matching—Rabin–Karp	56
	2.7 Longest Palindrome of a String—Manacher	59
3	<b>Sequences</b>	62
	3.1 Shortest Path in a Grid	62
	3.2 The Levenshtein Edit Distance	63
	3.3 Longest Common Subsequence	65
	3.4 Longest Increasing Subsequence	68
	3.5 Winning Strategy in a Two-Player Game	70
4	<b>Arrays</b>	72
	4.1 Merge of Sorted Lists	73
	4.2 Sum Over a Range	74
	4.3 Duplicates in a Range	74
	4.4 Maximum Subarray Sum	75

4.5	Query for the Minimum of a Range—Segment Tree	75
4.6	Query the Sum over a Range—Fenwick Tree	77
4.7	Windows with $k$ Distinct Elements	80
<b>5</b>	<b>Intervals</b>	82
5.1	Interval Trees	82
5.2	Union of Intervals	85
5.3	The Interval Point Cover Problem	85
<b>6</b>	<b>Graphs</b>	88
6.1	Encoding in Python	88
6.2	Implicit Graphs	90
6.3	Depth-First Search—DFS	91
6.4	Breadth-First Search—BFS	93
6.5	Connected Components	94
6.6	Biconnected Components	97
6.7	Topological Sort	102
6.8	Strongly Connected Components	105
6.9	2-Satisfiability	110
<b>7</b>	<b>Cycles in Graphs</b>	113
7.1	Eulerian Tour	113
7.2	The Chinese Postman Problem	116
7.3	Cycles with Minimal Ratio of Weight to Length—Karp	117
7.4	Cycles with Minimal Cost-to-Time Ratio	120
7.5	Travelling Salesman	120
7.6	Full Example: Menu Tour	121
<b>8</b>	<b>Shortest Paths</b>	124
8.1	Composition Property	124
8.2	Graphs with Weights 0 or 1	126
8.3	Graphs with Non-negative Weights—Dijkstra	127
8.4	Graphs with Arbitrary Weights—Bellman–Ford	130
8.5	All Source–Destination paths—Floyd–Warshall	132
8.6	Grid	133
8.7	Variants	135
<b>9</b>	<b>Matchings and Flows</b>	138
9.1	Maximum Bipartite Matching	139
9.2	Maximal-Weight Perfect Matching—Kuhn–Munkres	145
9.3	Planar Matching without Crossings	151
9.4	Stable Marriages—Gale–Shapley	153

9.5	Maximum Flow by Ford–Fulkerson	155
9.6	Maximum Flow by Edmonds–Karp	158
9.7	Maximum Flow by Dinic	159
9.8	Minimum $s - t$ Cut	162
9.9	$s - t$ Minimum Cut for Planar Graphs	163
9.10	A Transport Problem	165
9.11	Reductions between Matchings and Flows	165
9.12	Width of a Partial Order—Dilworth	167
<b>10</b>	<b>Trees</b>	171
10.1	Huffman Coding	172
10.2	Lowest Common Ancestor	174
10.3	Longest Path in a Tree	178
10.4	Minimum Weight Spanning Tree—Kruskal	179
<b>11</b>	<b>Sets</b>	182
11.1	The Knapsack Problem	182
11.2	Making Change	184
11.3	Subset Sum	185
11.4	The $k$ -sum Problem	187
<b>12</b>	<b>Points and Polygons</b>	189
12.1	Convex Hull	190
12.2	Measures of a Polygon	193
12.3	Closest Pair of Points	195
12.4	Simple Rectilinear Polygon	198
<b>13</b>	<b>Rectangles</b>	200
13.1	Forming Rectangles	200
13.2	Largest Square in a Grid	201
13.3	Largest Rectangle in a Histogram	202
13.4	Largest Rectangle in a Grid	204
13.5	Union of Rectangles	205
13.6	Union of Disjoint Rectangles	212
<b>14</b>	<b>Numbers and Matrices</b>	214
14.1	GCD	214
14.2	Bézout Coefficients	214
14.3	Binomial Coefficients	215
14.4	Fast Exponentiation	216
14.5	Prime Numbers	217
14.6	Evaluate an Arithmetical Expression	218

viii	<b>Contents</b>	
	14.7 System of Linear Equations	221
	14.8 Multiplication of a Matrix Sequence	225
<b>15</b>	<b>Exhaustive Search</b>	227
	15.1 All Paths for a Laser	227
	15.2 The Exact Cover Problem	231
	15.3 Problems	237
	15.4 Sudoku	238
	15.5 Enumeration of Permutations	240
	15.6 Le Compte est Bon	243
<b>16</b>	<b>Conclusion</b>	245
	16.1 Combine Algorithms to Solve a Problem	245
	16.2 For Further Reading	245
	16.3 Rendez-vous on tryalgo.org	246
	<i>Debugging tool</i>	247
	<i>References</i>	248
	<i>Index</i>	251



## Preface

Algorithms play an important role in our society, solving numerous mathematical problems which appear in a broad spectrum of situations. To give a few examples, think of planning taxi routes given a set of reservations (see Section 9.12); assigning students to schools in a large urban school district, such as New York (see Section 9.4); or identifying a bottleneck in a transportation network (see Section 9.8). This is why job interviews in the IT (Information Technology) industry test candidates for their problem-solving skills. Many programming contests are organised by companies such as Google, Facebook and Microsoft to spot gifted candidates and then send them job offers. This book will help students to develop a culture of algorithms and data structures, so that they know how to apply them properly when faced with new mathematical problems.

Designing the right algorithm to solve a given problem is only half of the work; to complete the job, the algorithm needs to be implemented efficiently. This is why this book also emphasises implementation issues, and provides full source code for most of the algorithms presented. We have chosen Python for these implementations. What makes this language so enticing is that it allows a particularly clear and refined expression, illustrating the essential steps of the algorithm, without obscuring things behind burdensome notations describing data structures. Surprisingly, it is actually possible to re-read code written several months ago and even understand it!

We have collected here 128 algorithmic problems, indexed by theme rather than by technique. Many are classic, whereas certain are atypical. This work should prove itself useful when preparing to solve the wide variety of problems posed in programming contests such as ICPC, Google Code Jam, Facebook Hacker Cup, Prologin, France-ioi, etc. We hope that it could serve as a basis for an advanced course in programming and algorithms, where even certain candidates for the ‘agrégation de mathématiques option informatique’ (French competitive exam for the highest teacher’s certification) will find a few original developments. The website [tryalgo.org](http://tryalgo.org), maintained by the authors, contains links to the code of this book, as well as to selected problems at various online contests. This allows readers to verify their freshly acquired skills.

This book would never have seen the light of day without the support of the authors’ life partners. Danke, Hương. Merci, 智子. The authors would also like to thank the students of the École polytechnique and the École normale supérieure of Paris-Saclay, whose practice, often nocturnal, generated a major portion of the

material of this text. Thanks to all those who proofread the manuscript, especially René Adad, Evripidis Bampis, Binh-Minh Bui-Xuan, Stéphane Henriot, Lê Thành Dũng Nguyễn, Alexandre Nolin and Antoine Pietri. Thanks to all those who improved the programs on GitHub: Louis Abraham, Lilian Besson, Ryan Lahfa, Olivier Marty, Samuel Tardieu and Xavier Carcelle. One of the authors would especially like to thank his past teacher at the Lycée Thiers, Monsieur Yves Lemaire, for having introduced him to the admirable gem of Section 2.5 on page 52.

We hope that the reader will pass many long hours tackling algorithmic problems that at first glance appear insurmountable, and in the end feel the profound joy when a solution, especially an elegant solution, suddenly becomes apparent.

Finally, we would like to thank Danièle and Greg Gibbons for their translation of this work, even of this very phrase.

Attention, it's all systems go!