
Data Structures and Algorithms in Python

Michael T. Goodrich

Department of Computer Science
University of California, Irvine

Roberto Tamassia

Department of Computer Science
Brown University

Michael H. Goldwasser

Department of Mathematics and Computer Science
Saint Louis University

WILEY

VP & PUBLISHER	Don Fowley
EXECUTIVE EDITOR	Beth Lang Golub
EDITORIAL PROGRAM ASSISTANT	Katherine Willis
MARKETING MANAGER	Christopher Ruel
DESIGNER	Kenji Ngieng
SENIOR PRODUCTION MANAGER	Janis Soo
ASSOCIATE PRODUCTION MANAGER	Joyce Poh

This book was set in L^AT_EX by the authors. Printed and bound by Courier Westford.
The cover was printed by Courier Westford.

This book is printed on acid free paper.

Founded in 1807, John Wiley & Sons, Inc. has been a valued source of knowledge and understanding for more than 200 years, helping people around the world meet their needs and fulfill their aspirations. Our company is built on a foundation of principles that include responsibility to the communities we serve and where we live and work. In 2008, we launched a Corporate Citizenship Initiative, a global effort to address the environmental, social, economic, and ethical challenges we face in our business. Among the issues we are addressing are carbon impact, paper specifications and procurement, ethical conduct within our business and among our vendors, and community and charitable support. For more information, please visit our website: www.wiley.com/go/citizenship.

Copyright © 2013 John Wiley & Sons, Inc. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc. 222 Rosewood Drive, Danvers, MA 01923, website www.copyright.com. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030-5774, (201)748-6011, fax (201)748-6008, website <http://www.wiley.com/go/permissions>.

Evaluation copies are provided to qualified academics and professionals for review purposes only, for use in their courses during the next academic year. These copies are licensed and may not be sold or transferred to a third party. Upon completion of the review period, please return the evaluation copy to Wiley. Return instructions and a free of charge return mailing label are available at www.wiley.com/go/returnlabel. If you have chosen to adopt this textbook for use in your course, please accept this book as your complimentary desk copy. Outside of the United States, please contact your local sales representative.

Printed in the United States of America

10987654321

Preface	v
1 Python Primer	1
1.1 Python Overview	2
1.1.1 The Python Interpreter	2
1.1.2 Preview of a Python Program	3
1.2 Objects in Python	4
1.2.1 Identifiers, Objects, and the Assignment Statement	4
1.2.2 Creating and Using Objects	6
1.2.3 Python's Built-In Classes	7
1.3 Expressions, Operators, and Precedence	12
1.3.1 Compound Expressions and Operator Precedence	17
1.4 Control Flow	18
1.4.1 Conditionals	18
1.4.2 Loops	20
1.5 Functions	23
1.5.1 Information Passing	24
1.5.2 Python's Built-In Functions	28
1.6 Simple Input and Output	30
1.6.1 Console Input and Output	30
1.6.2 Files	31
1.7 Exception Handling	33
1.7.1 Raising an Exception	34
1.7.2 Catching an Exception	36
1.8 Iterators and Generators	39
1.9 Additional Python Conveniences	42
1.9.1 Conditional Expressions	42
1.9.2 Comprehension Syntax	43
1.9.3 Packing and Unpacking of Sequences	44
1.10 Scopes and Namespaces	46
1.11 Modules and the Import Statement	48
1.11.1 Existing Modules	49
1.12 Exercises	51

2	Object-Oriented Programming	56
2.1	Goals, Principles, and Patterns	57
2.1.1	Object-Oriented Design Goals	57
2.1.2	Object-Oriented Design Principles	58
2.1.3	Design Patterns	61
2.2	Software Development	62
2.2.1	Design	62
2.2.2	Pseudo-Code	64
2.2.3	Coding Style and Documentation	64
2.2.4	Testing and Debugging	67
2.3	Class Definitions	69
2.3.1	Example: CreditCard Class	69
2.3.2	Operator Overloading and Python's Special Methods	74
2.3.3	Example: Multidimensional Vector Class	77
2.3.4	Iterators	79
2.3.5	Example: Range Class	80
2.4	Inheritance	82
2.4.1	Extending the CreditCard Class	83
2.4.2	Hierarchy of Numeric Progressions	87
2.4.3	Abstract Base Classes	93
2.5	Namespaces and Object-Oriented	96
2.5.1	Instance and Class Namespaces	96
2.5.2	Name Resolution and Dynamic Dispatch	100
2.6	Shallow and Deep Copying	101
2.7	Exercises	103
3	Algorithm Analysis	109
3.1	Experimental Studies	111
3.1.1	Moving Beyond Experimental Analysis	113
3.2	The Seven Functions Used in This Book	115
3.2.1	Comparing Growth Rates	122
3.3	Asymptotic Analysis	123
3.3.1	The "Big-Oh" Notation	123
3.3.2	Comparative Analysis	128
3.3.3	Examples of Algorithm Analysis	130
3.4	Simple Justification Techniques	137
3.4.1	By Example	137
3.4.2	The "Contra" Attack	137
3.4.3	Induction and Loop Invariants	138
3.5	Exercises	141

4	Recursion	148
4.1	Illustrative Examples	150
4.1.1	The Factorial Function	150
4.1.2	Drawing an English Ruler	152
4.1.3	Binary Search	155
4.1.4	File Systems	157
4.2	Analyzing Recursive Algorithms	161
4.3	Recursion Run Amok	165
4.3.1	Maximum Recursive Depth in Python	168
4.4	Further Examples of Recursion	169
4.4.1	Linear Recursion	169
4.4.2	Binary Recursion	174
4.4.3	Multiple Recursion	175
4.5	Designing Recursive Algorithms	177
4.6	Eliminating Tail Recursion	178
4.7	Exercises	180
5	Array-Based Sequences	183
5.1	Python's Sequence Types	184
5.2	Low-Level Arrays	185
5.2.1	Referential Arrays	187
5.2.2	Compact Arrays in Python	190
5.3	Dynamic Arrays and Amortization	192
5.3.1	Implementing a Dynamic Array	195
5.3.2	Amortized Analysis of Dynamic Arrays	197
5.3.3	Python's List Class	201
5.4	Efficiency of Python's Sequence Types	202
5.4.1	Python's List and Tuple Classes	202
5.4.2	Python's String Class	208
5.5	Using Array-Based Sequences	210
5.5.1	Storing High Scores for a Game	210
5.5.2	Sorting a Sequence	214
5.5.3	Simple Cryptography	216
5.6	Multidimensional Data Sets	219
5.7	Exercises	224
6	Stacks, Queues, and Deques	228
6.1	Stacks	229
6.1.1	The Stack Abstract Data Type	230
6.1.2	Simple Array-Based Stack Implementation	231
6.1.3	Reversing Data Using a Stack	235
6.1.4	Matching Parentheses and HTML Tags	236

6.2	Queues	239
6.2.1	The Queue Abstract Data Type	240
6.2.2	Array-Based Queue Implementation	241
6.3	Double-Ended Queues	247
6.3.1	The Deque Abstract Data Type	247
6.3.2	Implementing a Deque with a Circular Array	248
6.3.3	Dequeues in the Python Collections Module	249
6.4	Exercises	250
7	Linked Lists	255
7.1	Singly Linked Lists	256
7.1.1	Implementing a Stack with a Singly Linked List	261
7.1.2	Implementing a Queue with a Singly Linked List	264
7.2	Circularly Linked Lists	266
7.2.1	Round-Robin Schedulers	267
7.2.2	Implementing a Queue with a Circularly Linked List	268
7.3	Doubly Linked Lists	270
7.3.1	Basic Implementation of a Doubly Linked List	273
7.3.2	Implementing a Deque with a Doubly Linked List	275
7.4	The Positional List ADT	277
7.4.1	The Positional List Abstract Data Type	279
7.4.2	Doubly Linked List Implementation	281
7.5	Sorting a Positional List	285
7.6	Case Study: Maintaining Access Frequencies	286
7.6.1	Using a Sorted List	286
7.6.2	Using a List with the Move-to-Front Heuristic	289
7.7	Link-Based vs. Array-Based Sequences	292
7.8	Exercises	294
8	Trees	299
8.1	General Trees	300
8.1.1	Tree Definitions and Properties	301
8.1.2	The Tree Abstract Data Type	305
8.1.3	Computing Depth and Height	308
8.2	Binary Trees	311
8.2.1	The Binary Tree Abstract Data Type	313
8.2.2	Properties of Binary Trees	315
8.3	Implementing Trees	317
8.3.1	Linked Structure for Binary Trees	317
8.3.2	Array-Based Representation of a Binary Tree	325
8.3.3	Linked Structure for General Trees	327
8.4	Tree Traversal Algorithms	328

8.4.1	Preorder and Postorder Traversals of General Trees . . .	328
8.4.2	Breadth-First Tree Traversal	330
8.4.3	Inorder Traversal of a Binary Tree	331
8.4.4	Implementing Tree Traversals in Python	333
8.4.5	Applications of Tree Traversals	337
8.4.6	Euler Tours and the Template Method Pattern ★	341
8.5	Case Study: An Expression Tree	348
8.6	Exercises	352
9	Priority Queues	362
9.1	The Priority Queue Abstract Data Type	363
9.1.1	Priorities	363
9.1.2	The Priority Queue ADT	364
9.2	Implementing a Priority Queue	365
9.2.1	The Composition Design Pattern	365
9.2.2	Implementation with an Unsorted List	366
9.2.3	Implementation with a Sorted List	368
9.3	Heaps	370
9.3.1	The Heap Data Structure	370
9.3.2	Implementing a Priority Queue with a Heap	372
9.3.3	Array-Based Representation of a Complete Binary Tree .	376
9.3.4	Python Heap Implementation	376
9.3.5	Analysis of a Heap-Based Priority Queue	379
9.3.6	Bottom-Up Heap Construction ★	380
9.3.7	Python's heapq Module	384
9.4	Sorting with a Priority Queue	385
9.4.1	Selection-Sort and Insertion-Sort	386
9.4.2	Heap-Sort	388
9.5	Adaptable Priority Queues	390
9.5.1	Locators	390
9.5.2	Implementing an Adaptable Priority Queue	391
9.6	Exercises	395
10	Maps, Hash Tables, and Skip Lists	401
10.1	Maps and Dictionaries	402
10.1.1	The Map ADT	403
10.1.2	Application: Counting Word Frequencies	405
10.1.3	Python's MutableMapping Abstract Base Class	406
10.1.4	Our MapBase Class	407
10.1.5	Simple Unsorted Map Implementation	408
10.2	Hash Tables	410
10.2.1	Hash Functions	411

10.2.2	Collision-Handling Schemes	417
10.2.3	Load Factors, Rehashing, and Efficiency	420
10.2.4	Python Hash Table Implementation	422
10.3	Sorted Maps	427
10.3.1	Sorted Search Tables	428
10.3.2	Two Applications of Sorted Maps	434
10.4	Skip Lists	437
10.4.1	Search and Update Operations in a Skip List	439
10.4.2	Probabilistic Analysis of Skip Lists ★	443
10.5	Sets, Multisets, and Multimaps	446
10.5.1	The Set ADT	446
10.5.2	Python's MutableSet Abstract Base Class	448
10.5.3	Implementing Sets, Multisets, and Multimaps	450
10.6	Exercises	452
11	Search Trees	459
11.1	Binary Search Trees	460
11.1.1	Navigating a Binary Search Tree	461
11.1.2	Searches	463
11.1.3	Insertions and Deletions	465
11.1.4	Python Implementation	468
11.1.5	Performance of a Binary Search Tree	473
11.2	Balanced Search Trees	475
11.2.1	Python Framework for Balancing Search Trees	478
11.3	AVL Trees	481
11.3.1	Update Operations	483
11.3.2	Python Implementation	488
11.4	Splay Trees	490
11.4.1	Splaying	490
11.4.2	When to Splay	494
11.4.3	Python Implementation	496
11.4.4	Amortized Analysis of Splaying ★	497
11.5	(2,4) Trees	502
11.5.1	Multiway Search Trees	502
11.5.2	(2,4)-Tree Operations	505
11.6	Red-Black Trees	512
11.6.1	Red-Black Tree Operations	514
11.6.2	Python Implementation	525
11.7	Exercises	528

12	Sorting and Selection	536
12.1	Why Study Sorting Algorithms?	537
12.2	Merge-Sort	538
12.2.1	Divide-and-Conquer	538
12.2.2	Array-Based Implementation of Merge-Sort	543
12.2.3	The Running Time of Merge-Sort	544
12.2.4	Merge-Sort and Recurrence Equations ★	546
12.2.5	Alternative Implementations of Merge-Sort	547
12.3	Quick-Sort	550
12.3.1	Randomized Quick-Sort	557
12.3.2	Additional Optimizations for Quick-Sort	559
12.4	Studying Sorting through an Algorithmic Lens	562
12.4.1	Lower Bound for Sorting	562
12.4.2	Linear-Time Sorting: Bucket-Sort and Radix-Sort	564
12.5	Comparing Sorting Algorithms	567
12.6	Python’s Built-In Sorting Functions	569
12.6.1	Sorting According to a Key Function	569
12.7	Selection	571
12.7.1	Prune-and-Search	571
12.7.2	Randomized Quick-Select	572
12.7.3	Analyzing Randomized Quick-Select	573
12.8	Exercises	574
13	Text Processing	581
13.1	Abundance of Digitized Text	582
13.1.1	Notations for Strings and the Python str Class	583
13.2	Pattern-Matching Algorithms	584
13.2.1	Brute Force	584
13.2.2	The Boyer-Moore Algorithm	586
13.2.3	The Knuth-Morris-Pratt Algorithm	590
13.3	Dynamic Programming	594
13.3.1	Matrix Chain-Product	594
13.3.2	DNA and Text Sequence Alignment	597
13.4	Text Compression and the Greedy Method	601
13.4.1	The Huffman Coding Algorithm	602
13.4.2	The Greedy Method	603
13.5	Tries	604
13.5.1	Standard Tries	604
13.5.2	Compressed Tries	608
13.5.3	Suffix Tries	610
13.5.4	Search Engine Indexing	612

13.6 Exercises	613
14 Graph Algorithms	619
14.1 Graphs	620
14.1.1 The Graph ADT	626
14.2 Data Structures for Graphs	627
14.2.1 Edge List Structure	628
14.2.2 Adjacency List Structure	630
14.2.3 Adjacency Map Structure	632
14.2.4 Adjacency Matrix Structure	633
14.2.5 Python Implementation	634
14.3 Graph Traversals	638
14.3.1 Depth-First Search	639
14.3.2 DFS Implementation and Extensions	644
14.3.3 Breadth-First Search	648
14.4 Transitive Closure	651
14.5 Directed Acyclic Graphs	655
14.5.1 Topological Ordering	655
14.6 Shortest Paths	659
14.6.1 Weighted Graphs	659
14.6.2 Dijkstra's Algorithm	661
14.7 Minimum Spanning Trees	670
14.7.1 Prim-Jarník Algorithm	672
14.7.2 Kruskal's Algorithm	676
14.7.3 Disjoint Partitions and Union-Find Structures	681
14.8 Exercises	686
15 Memory Management and B-Trees	697
15.1 Memory Management	698
15.1.1 Memory Allocation	699
15.1.2 Garbage Collection	700
15.1.3 Additional Memory Used by the Python Interpreter	703
15.2 Memory Hierarchies and Caching	705
15.2.1 Memory Systems	705
15.2.2 Caching Strategies	706
15.3 External Searching and B-Trees	711
15.3.1 (a, b) Trees	712
15.3.2 B-Trees	714
15.4 External-Memory Sorting	715
15.4.1 Multiway Merging	716
15.5 Exercises	717

<i>Contents</i>	xix
A Character Strings in Python	721
B Useful Mathematical Facts	725
Bibliography	732
Index	737