

Data Structures & Algorithms in Java

Java 数据结构和算法 (第二版·影印版)

[美] Robert Lafore 著



SAMS



中国电力出版社
www.infopower.com.cn

国外经典计算机科学教材

Data Structures & Algorithms in Java

Java 数据结构和算法

(第二版·影印版)

[美] Robert Lafore 著

江苏工业学院图书馆
藏书章



中国电力出版社

www.infopower.com.cn

Data Structures & Algorithms in Java, Second Edition (ISBN 0-672-32453-9)

Robert Lafore

Copyright © 2003 Sams Publishing

Original English Language Edition Published by Addison-Wesley.

All rights reserved.

Reprinting edition published by PEARSON EDUCATION NORTH ASIA LTD and CHINA ELECTRIC POWER PRESS, Copyright © 2007.

本书影印版由 Pearson Education 授权中国电力出版社在中国境内（香港、澳门特别行政区和台湾地区除外）独家出版、发行。

未经出版者书面许可，不得以任何方式复制或抄袭本书的任何部分。

本书封面贴有 Pearson Education 防伪标签，无标签者不得销售。

北京市版权局著作合同登记号 图字：01-2007-2186 号

图书在版编目（CIP）数据

Java 数据结构和算法 = Data Structures & Algorithms in Java: 第 2 版: 英文 / (美) 拉佛 (Lafore, R.)

著. 影印本. —北京: 中国电力出版社, 2007.6

国外经典计算机科学教材

ISBN 978-7-5083-5644-0

I. J… II. 拉… III. JAVA 语言—程序设计—教材—英文 IV. TP312

中国版本图书馆 CIP 数据核字 (2007) 第 066468 号

书 名: Java 数据结构和算法 (第二版·影印版)

编 著: (美) Robert Lafore

责任编辑: 牛贵华

出版发行: 中国电力出版社

地址: 北京市三里河路6号

邮政编码: 100044

电话: (010) 88515918

传 真: (010) 88518169

印 刷: 北京市铁成印刷厂

开本尺寸: 185×233

印 张: 48.25

书 号: ISBN 978-7-5083-5644-0

版 次: 2007 年 6 月北京第 1 版

2007 年 6 月第 1 次印刷

定 价: 69.80 元

敬 告 读 者

本书封面贴有防伪标签，加热后中心图案消失

本书如有印装质量问题，我社发行部负责退换

版权所有 翻印必究

About the Author



Robert Lafore has degrees in Electrical Engineering and Mathematics, has worked as a systems analyst for the Lawrence Berkeley Laboratory, founded his own software company, and is a best-selling writer in the field of computer programming. Some of his current titles are *C++ Interactive Course* and *Object-Oriented Programming in C++*. Earlier best-selling titles include *Assembly Language Primer for the IBM PC and XT* and (back at the beginning of the computer revolution) *Soul of CP/M*.

Dedication

This book is dedicated to my readers, who have rewarded me over the years not only by buying my books, but with helpful suggestions and kind words. Thanks to you all.

Acknowledgments to the Second Edition

My thanks to the following people at Sams Publishing for their competence, effort, and patience in the development of this second edition: Acquisitions Editor Carol Ackerman and Development Editor Sonja Gougeon. Sonja guided this edition through the complex production process. Project Editor Matt Purcell corrected a seemingly infinite number of grammatical errors and made sure everything made sense. Tech Editor Mike Kopak reviewed the programs and saved me from several problems. Last but not least, Dan Schiet, an old friend from a previous era, provides skilled management of my code and appears on the Sams Web site.

Acknowledgments to the First Edition

My gratitude for the following people (and many others) cannot be fully expressed in this short acknowledgment. As always, Mitch Waite had the Java thing figured out before anyone else. He also let me bounce the applets off him until they did the job, and extracted the overall form of the project from a miasma of speculation. My editor, Kurt Stephan, found great reviewers, made sure everyone was on the same page, kept the ball rolling, and gently but firmly ensured that I did what I was supposed to do. Harry Henderson provided a skilled appraisal of the first draft, along with many valuable suggestions. Richard S. Wright, Jr., as technical editor, corrected numerous problems with his keen eye for detail. Jaime Niño, Ph.D., of the University of New Orleans, attempted to save me from myself and occasionally succeeded, but should bear no responsibility for my approach or coding details. Susan Walton has been a staunch and much-appreciated supporter in helping to convey the essence of the project to the non-technical. Carmela Carvajal was invaluable in extending our contacts with the academic world. Dan Scherf not only put the CD-ROM together, but was tireless in keeping me up to date on rapidly evolving software changes. Finally, Cecile Kaufman ably shepherded the book through its transition from the editing to the production process.

Acknowledgments to the Second Edition

My thanks to the following people at Sams Publishing for their competence, effort, and patience in the development of this second edition. Acquisitions Editor Carol Ackerman and Development Editor Songlin Qiu ably guided this edition through the complex production process. Project Editor Matt Purcell corrected a semi-infinite number of grammatical errors and made sure everything made sense. Tech Editor Mike Kopak reviewed the programs and saved me from several problems. Last but not least, Dan Scherf, an old friend from a previous era, provides skilled management of my code and applets on the Sams Web site.

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

As an executive editor for Sams Publishing, I welcome your comments. You can email or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that I cannot help you with technical problems related to the *topic* of this book. We do have a User Services group, however, where I will forward specific technical questions related to the book.

When you write, please be sure to include this book's title and author as well as your name, email address, and phone number. I will carefully review your comments and share them with the author and editors who worked on the book.

Email: feedback@sampublishing.com

Mail: Michael Stephens
Executive Editor
Sams Publishing
800 East 96th Street
Indianapolis, IN 46240 USA

For more information about this book or another Sams Publishing title, visit our Web site at www.sampublishing.com. Type the ISBN (excluding hyphens) or the title of a book in the Search field to find the page you're looking for.

Contents at a Glance

	Introduction	1
1	Overview	9
1	2 Arrays	33
1	3 Simple Sorting	77
2	4 Stacks and Queues	115
2	5 Linked Lists	179
2	6 Recursion	251
3	7 Advanced Sorting	315
3	8 Binary Trees	365
4	9 Red-Black Trees	429
2	10 2-3-4 Trees and External Storage	463
2	11 Hash Tables	519
6	12 Heaps	579
6	13 Graphs	615
8	14 Weighted Graphs	669
9	15 When to Use What	717
9	Appendixes	
10	A Running the Workshop Applets and Example Programs	729
11	B Further Reading	735
11	C Answers to Questions	739

Table of Contents

Introduction	1
What's New in the Second Edition	1
Additional Topics	1
End-of-Chapter Questions	2
Experiments	2
Programming Projects	2
What This Book Is About	2
What's Different About This Book	3
Easy to Understand	3
Workshop Applets	4
Java Example Code	5
Who This Book Is For	5
What You Need to Know Before You Read This Book	5
The Software You Need to Use This Book	6
How This Book Is Organized	6
Enjoy Yourself!	8
1 Overview	9
What Are Data Structures and Algorithms Good For?	9
Real-World Data Storage	10
Programmer's Tools	11
Real-World Modeling	11
Overview of Data Structures	11
Overview of Algorithms	12
Some Definitions	13
Database	13
Record	13
Field	13
Key	14
Object-Oriented Programming	14
Problems with Procedural Languages	14
Objects in a Nutshell	15
A Runnable Object-Oriented Program	18
Inheritance and Polymorphism	21
Software Engineering	21

40	Java for C++ Programmers	22
50	No Pointers	22
53	Overloaded Operators	25
70	Primitive Variable Types	25
70	Input/Output	26
70	Java Library Data Structures	29
71	Summary	30
71	Questions	30
73	2 Arrays	33
74	The Array Workshop Applet	33
75	Insertion	35
76	Searching	36
77	Deletion	36
78	The Duplicates Issue	37
78	Not Too Swift	39
79	The Basics of Arrays in Java	39
79	Creating an Array	40
81	Accessing Array Elements	40
82	Initialization	41
88	An Array Example	41
88	Dividing a Program into Classes	44
89	Classes LowArray and LowArrayApp	46
89	Class Interfaces	46
90	Not So Convenient	47
92	Who's Responsible for What?	48
92	The highArray.java Example	48
92	The User's Life Made Easier	52
92	Abstraction	52
92	The Ordered Workshop Applet	52
97	Linear Search	53
99	Binary Search	54
103	Java Code for an Ordered Array	56
103	Binary Search with the find() Method	56
103	The OrdArray Class	58
104	Advantages of Ordered Arrays	61
107	Logarithms	62
107	The Equation	63
108	The Opposite of Raising Two to a Power	64

Storing Objects	64
The Person Class	65
The classDataArray.java Program	65
Big O Notation	70
Insertion in an Unordered Array: Constant	70
Linear Search: Proportional to N	70
Binary Search: Proportional to log(N)	71
Don't Need the Constant	71
Why Not Use Arrays for Everything?	72
Summary	73
Questions	74
Experiments	75
Programming Projects	76
3 Simple Sorting	77
How Would You Do It?	78
Bubble Sort	79
Bubble Sort on the Baseball Players	79
The BubbleSort Workshop Applet	81
Java Code for a Bubble Sort	85
Invariants	88
Efficiency of the Bubble Sort	88
Selection Sort	89
Selection Sort on the Baseball Players	89
The SelectSort Workshop Applet	90
Java Code for Selection Sort	92
Invariant	95
Efficiency of the Selection Sort	95
Insertion Sort	95
Insertion Sort on the Baseball Players	95
The InsertSort Workshop Applet	97
Java Code for Insertion Sort	99
Invariants in the Insertion Sort	103
Efficiency of the Insertion Sort	103
Sorting Objects	103
Java Code for Sorting Objects	104
Lexicographical Comparisons	107
Stability	107
Comparing the Simple Sorts	108
Summary	108

Questions	109
Experiments	111
Programming Projects	112
4 Stacks and Queues	115
A Different Kind of Structure	115
Programmer's Tools	115
Restricted Access	116
More Abstract	116
Stacks	116
The Postal Analogy	117
The Stack Workshop Applet	118
Java Code for a Stack	120
Stack Example 1: Reversing a Word	124
Stack Example 2: Delimiter Matching	127
Efficiency of Stacks	132
Queues	132
The Queue Workshop Applet	133
A Circular Queue	136
Java Code for a Queue	137
Efficiency of Queues	142
Dequeues	143
Priority Queues	143
The PriorityQ Workshop Applet	144
Java Code for a Priority Queue	147
Efficiency of Priority Queues	149
Parsing Arithmetic Expressions	149
Postfix Notation	150
Translating Infix to Postfix	151
Evaluating Postfix Expressions	167
Summary	173
Questions	174
Experiments	176
Programming Projects	176
5 Linked Lists	179
Links	179
References and Basic Types	180
Relationship, Not Position	182

109	The LinkedList Workshop Applet	183
111	The Insert Button	183
112	The Find Button	184
112	The Delete Button	184
112	A Simple Linked List	185
112	The Link Class	185
112	The LinkedList Class	186
116	The insertFirst() Method	187
116	The deleteFirst() Method	188
116	The displayList() Method	189
117	The linkList.java Program	190
118	Finding and Deleting Specified Links	193
120	The find() Method	196
124	The delete() Method	196
127	Other Methods	197
132	Double-Ended Lists	198
132	Linked-List Efficiency	202
133	Abstract Data Types	202
136	A Stack Implemented by a Linked List	203
137	A Queue Implemented by a Linked List	206
142	Data Types and Abstraction	210
144	ADT Lists	211
144	ADTs as a Design Tool	212
144	Sorted Lists	212
147	Java Code to Insert an Item in a Sorted List	213
149	The sortedList.java Program	215
149	Efficiency of Sorted Linked Lists	218
150	List Insertion Sort	218
151	Doubly Linked Lists	221
157	Traversal	222
173	Insertion	223
174	Deletion	225
176	The doublyLinked.java Program	226
176	Doubly Linked List as Basis for Deques	231
177	Iterators	231
179	A Reference in the List Itself?	232
180	An Iterator Class	232
180	Additional Iterator Features	233
182	Iterator Methods	234
182	The interIterator.java Program	235

208	Where Does the Iterator Point?	242
308	The atEnd() Method	242
308	Iterative Operations	243
310	Other Methods	244
312	Summary	244
312	Questions	245
312	Experiments	247
312	Programming Projects	247
312	6 Recursion	251
316	Triangular Numbers	251
317	Finding the nth Term Using a Loop	252
318	Finding the nth Term Using Recursion	253
321	The triangle.java Program	255
324	What's Really Happening?	257
324	Characteristics of Recursive Methods	259
325	Is Recursion Efficient?	259
325	Mathematical Induction	259
327	Factorials	260
328	Anagrams	262
328	A Recursive Binary Search	268
328	Recursion Replaces the Loop	268
328	Divide-and-Conquer Algorithms	272
328	The Towers of Hanoi	273
340	The Towers Workshop Applet	274
344	Moving Subtrees	275
345	The Recursive Algorithm	276
350	The towers.java Program	277
354	mergesort	279
355	Merging Two Sorted Arrays	280
357	Sorting by Merging	283
358	The MergeSort Workshop Applet	285
358	The mergeSort.java Program	287
359	Efficiency of the mergesort	291
359	Eliminating Recursion	294
361	Recursion and Stacks	294
363	Simulating a Recursive Method	294
363	What Does This Prove?	301
	Some Interesting Recursive Applications	303
	Raising a Number to a Power	303

.....	The Knapsack Problem	305
.....	Combinations: Picking a Team	306
.....	Summary	308
.....	Questions	310
.....	Experiments	312
.....	Programming Projects	312
7	Advanced Sorting	315
.....	Shellsort	315
.....	Insertion Sort: Too Many Copies	316
.....	N-Sorting	316
.....	Diminishing Gaps	317
.....	The Shellsort Workshop Applet	319
.....	Java Code for the Shellsort	321
.....	Other Interval Sequences	324
.....	Efficiency of the Shellsort	324
.....	Partitioning	325
.....	The Partition Workshop Applet	325
.....	The partition.java Program	327
.....	The Partition Algorithm	330
.....	Efficiency of the Partition Algorithm	332
.....	Quicksort	333
.....	The Quicksort Algorithm	333
.....	Choosing a Pivot Value	335
.....	The QuickSort1 Workshop Applet	340
.....	Degenerates to $O(N^2)$ Performance	344
.....	Median-of-Three Partitioning	345
.....	Handling Small Partitions	350
.....	Removing Recursion	354
.....	Efficiency of Quicksort	355
.....	Radix Sort	357
.....	Algorithm for the Radix Sort	358
.....	Designing a Program	358
.....	Efficiency of the Radix Sort	359
.....	Summary	359
.....	Questions	361
.....	Experiments	363
.....	Programming Projects	363

104	8 Binary Trees	365
403	Why Use Binary Trees?	365
404	Slow Insertion in an Ordered Array	365
403	Slow Searching in a Linked List	366
414	Trees to the Rescue	366
414	What Is a Tree?	366
417	Tree Terminology	367
418	Path	368
420	Root	368
421	Parent	369
422	Child	369
423	Leaf	369
424	Subtree	369
425	Visiting	369
429	Traversing	369
429	Levels	369
430	Keys	369
430	Binary Trees	370
430	An Analogy	370
430	How Do Binary Search Trees Work?	371
431	The Binary Tree Workshop Applet	371
432	Representing the Tree in Java Code	373
432	Finding a Node	376
434	Using the Workshop Applet to Find a Node	376
434	Java Code for Finding a Node	377
434	Tree Efficiency	378
434	Inserting a Node	378
434	Using the Workshop Applet to Insert a Node	379
436	Java Code for Inserting a Node	379
436	Traversing the Tree	381
436	Inorder Traversal	381
436	Java Code for Traversing	382
436	Traversing a Three-Node Tree	382
437	Traversing with the Workshop Applet	384
437	Preorder and Postorder Traversals	385
437	Finding Maximum and Minimum Values	388
437	Deleting a Node	389
438	Case 1: The Node to Be Deleted Has No Children	389
439	Case 2: The Node to Be Deleted Has One Child	391
	Case 3: The Node to Be Deleted Has Two Children	393

203	The Efficiency of Binary Trees	401
203	Trees Represented as Arrays	403
203	Duplicate Keys	404
203	The Complete tree.java Program	405
203	The Huffman Code	415
203	Character Codes	415
203	Decoding with the Huffman Tree	417
203	Creating the Huffman Tree	418
203	Coding the Message	420
203	Creating the Huffman Code	421
203	Summary	422
203	Questions	423
203	Experiments	425
203	Programming Projects	425
203	9 Red-Black Trees	429
203	Our Approach to the Discussion	429
203	Conceptual	430
203	Top-Down Insertion	430
203	Balanced and Unbalanced Trees	430
203	Degenerates to $O(N)$	431
203	Balance to the Rescue	432
203	Red-Black Tree Characteristics	432
203	Fixing Rule Violations	434
203	Using the RBTREE Workshop Applet	434
203	Clicking on a Node	435
203	The Start Button	435
203	The Ins Button	435
203	The Del Button	436
203	The Flip Button	436
203	The RoL Button	436
203	The RoR Button	436
203	The R/B Button	436
203	Text Messages	437
203	Where's the Find Button?	437
203	Experimenting with the Workshop Applet	437
203	Experiment 1: Inserting Two Red Nodes	437
203	Experiment 2: Rotations	438
203	Experiment 3: Color Flips	439