国外经典教材

PEARSON
Addison Wesley

Java算法（第3版，第2卷）

——图算法（影印版）

Algorithms in Java

Third Edition

Part 5

Graph Algorithms

（美）Robert Sedgewick 著

Pearson Education

清华大学出版社

国外经典教材

Java 算法（第 3 版，第 2 卷）

——图算法（影印版）

（美） Robert Sedgewick 著

清华大学出版社

北 京

内 容 简 介

本书深入介绍了图算法。书中分别对图属性和类型、图搜索、有向图、最小生成树、最短路径以及网络流的有关内容进行了透彻的讨论。在此不仅对基本内容做了全面的阐述，而且对经典算法也提供了详尽的分析，同时还涵盖了有关的高级主题。全书既强调了与实用有关的内容，在分析和理论研究上也很有深度。本书内容全面、论述清晰，适合于计算机科学和数学领域各个层次的人员使用。

图书在版编目（CIP）数据

Java 算法（第 3 版，第 2 卷）——图算法= Algorithms in Java (Third Edition), Part 5: Graph Algorithms/（美）塞吉维克著. —影印本. —北京：清华大学出版社，2003.9
（国外经典教材）
ISBN 7-302-07298-1
Ⅰ.J… Ⅱ.塞… Ⅲ.JAVA 语言—计算方法—教材—英文 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2003）第 085255 号

出 版 者: 清华大学出版社　　**地　址**: 北京清华大学学研大厦
http://www.tup.com.cn　　**邮　编**: 100084
社 总 机: 010-62770175　　**客户服务**: 010-62776969
责任编辑: 车立红
封面设计: 立日新设计公司
印 刷 者: 北京牛山世兴印刷厂
装 订 者: 三河市新茂装订有限公司
发 行 者: 新华书店总店北京发行所
开　本: 185×230　**印张**: 32.5
版　次: 2004 年 4 月第 1 版　2004 年 4 月第 1 次印刷
书　号: ISBN 7-302-07298-1/TP · 5298
印　数: 1～3000
定　价: 55.00 元

Preface

GRAPHS AND GRAPH algorithms are pervasive in modern computing applications. This book describes the most important known methods for solving the graph-processing problems that arise in practice. Its primary aim is to make these methods and the basic principles behind them accessible to the growing number of people in need of knowing them. The material is developed from first principles, starting with basic information and working through classical methods up through modern techniques that are still under development. Carefully chosen examples, detailed figures, and complete implementations supplement thorough descriptions of algorithms and applications.

Algorithms

This book is the second of three volumes that are intended to survey the most important computer algorithms in use today. The first volume (Parts 1–4) covers fundamental concepts (Part 1), data structures (Part 2), sorting algorithms (Part 3), and searching algorithms (Part 4); this volume (Part 5) covers graphs and graph algorithms; and the (yet to be published) third volume (Parts 6–8) covers strings (Part 6), computational geometry (Part 7), and advanced algorithms and applications (Part 8).

The books are useful as texts early in the computer science curriculum, after students have acquired basic programming skills and familiarity with computer systems, but before they have taken specialized courses in advanced areas of computer science or computer applications. The books also are useful for self-study or as a reference for people engaged in the development of computer systems or applications programs because they contain implementations of useful algorithms and detailed information on these algorithms' performance characteristics. The broad perspective taken makes the series an appropriate introduction to the field.

Together the three volumes comprise the *Third Edition* of a book that has been widely used by students and programmers around the world for many years. I have completely rewritten the text for this edition, and I have added thousands of new exercises, hundreds of new figures, dozens of new programs, and detailed commentary on all the figures and programs. This new material provides both coverage of new topics and fuller explanations of many of the classic algorithms. A new emphasis on abstract data types throughout the books makes the programs more broadly useful and relevant in modern object-oriented programming environments. People who have read previous editions will find a wealth of new information throughout; all readers will find a wealth of pedagogical material that provides effective access to essential concepts.

These books are not just for programmers and computer science students. Everyone who uses a computer wants it to run faster or to solve larger problems. The algorithms that we consider represent a body of knowledge developed during the last 50 years that is the basis for the efficient use of the computer for a broad variety of applications. From N-body simulation problems in physics to genetic-sequencing problems in molecular biology, the basic methods described here have become essential in scientific research; and from database systems to Internet search engines, they have become essential parts of modern software systems. As the scope of computer applications becomes more widespread, so grows the impact of basic algorithms. The goal of this book is to serve as a resource so that students and professionals can know and make intelligent use of graph algorithms as the need arises in whatever computer application they might undertake.

Scope

This book, ***Algorithms in Java, Third Edition, Part 5: Graph Algorithms***, contains six chapters that cover graph properties and types, graph search, directed graphs, minimal spanning trees, shortest paths, and networks. The descriptions here are intended to give readers an understanding of the basic properties of as broad a range of fundamental graph algorithms as possible.

You will most appreciate the material here if you have had a course covering basic principles of algorithm design and analysis and programming experience in a high-level language such as Java, C++, or C. *Algorithms in Java, Third Edition, Parts 1–4*, is certainly adequate preparation. This volume assumes basic knowledge about arrays, linked lists, and abstract data types (ADTs) and makes use of priority-queue, symbol-table, and union-find ADTs—all of which are described in detail in Parts 1–4 (and in many other introductory texts on algorithms and data structures).

Basic properties of graphs and graph algorithms are developed from first principles, but full understanding often can lead to deep and difficult mathematics. Although the discussion of advanced mathematical concepts is brief, general, and descriptive, you certainly need a higher level of mathematical maturity to appreciate graph algorithms than you do for the topics in Parts 1–4. Still, readers at various levels of mathematical maturity will be able to profit from this book. The topic dictates this approach: Some elementary graph algorithms that should be understood and used by everyone differ only slightly from some advanced algorithms that are not understood by anyone. The primary intent here is to place important algorithms in context with other methods throughout the book, not to teach all of the mathematical material. But the rigorous treatment demanded by good mathematics often leads us to good programs, so I have tried to provide a balance between the formal treatment favored by theoreticians and the coverage needed by practitioners, without sacrificing rigor.

Use in the Curriculum

There is a great deal of flexibility in how the material here can be taught, depending on the taste of the instructor and the preparation of the students. There is sufficient coverage of basic material for the book to be used to teach data structures to beginners, and there is sufficient detail and coverage of advanced material for the book to be used to teach the design and analysis of algorithms to upper-level students. Some instructors may wish to emphasize implementations and practical concerns; others may wish to emphasize analysis and theoretical concepts.

For a more comprehensive course, this book is also available in a special bundle with Parts 1–4; thereby instructors can cover fundamentals, data structures, sorting, searching, and graph algorithms in one consistent style.

The exercises—nearly all of which are new to this third edition—fall into several types. Some are intended to test understanding of material in the text, and simply ask readers to work through an example or to apply concepts described in the text. Others involve implementing and putting together the algorithms, or running empirical studies to compare variants of the algorithms and to learn their properties. Still others are a repository for important information at a level of detail that is not appropriate for the text. Reading and thinking about the exercises will pay dividends for every reader.

Algorithms of Practical Use

Anyone wanting to use a computer more effectively can use this book for reference or for self-study. People with programming experience can find information on specific topics throughout the book. To a large extent, you can read the individual chapters in the book independently of the others, although, in some cases, algorithms in one chapter make use of methods from a previous chapter.

The orientation of the book is to study algorithms likely to be of practical use. The book provides information about the tools of the trade to the point that readers can confidently implement, debug, and put algorithms to work to solve a problem or to provide functionality in an application. Full implementations of the methods discussed are included, as are descriptions of the operations of these programs on a consistent set of examples.

Because we work with real code, rather than write pseudo-code, you can put the programs to practical use quickly. Program listings are available from the book's home page. You can use these working programs in many ways to help you study algorithms. Read them to check your understanding of the details of an algorithm, or to see one way to handle initializations, boundary conditions, and other situations that pose programming challenges. Run them to see the algorithms in action, to study performance empirically and check your results against the tables in the book, or to try your own modifications.

Indeed, one practical application of the algorithms has been to produce the hundreds of figures throughout the book. Many algorithms are brought to light on an intuitive level through the visual dimension provided by these figures.

Characteristics of the algorithms and of the situations in which they might be useful are discussed in detail. Connections to the analysis of algorithms and theoretical computer science are developed in context. When appropriate, empirical and analytic results are presented to illustrate why certain algorithms are preferred. When interesting, the relationship of the practical algorithms being discussed to purely theoretical results is described. Specific information on performance characteristics of algorithms and implementations is synthesized, encapsulated, and discussed throughout the book.

Programming Language

The programming language used for all of the implementations is Java. The programs use a wide range of standard Java idioms, and the text includes concise descriptions of each construct.

Mike Schidlowsky and I developed a style of Java programming based on ADTs that we feel is an effective way to present the algorithms and data structures as real programs. We have striven for elegant, compact, efficient, and portable implementations. The style is consistent whenever possible, so programs that are similar look similar.

A goal of this book is to present the algorithms in as simple and direct a form as possible. For many of the algorithms, the similarities remain regardless of which language is used: Dijkstra's algorithm (to pick one prominent example) is Dijkstra's algorithm, whether expressed in Algol-60, Basic, Fortran, Smalltalk, Ada, Pascal, C, C++, Modula-3, PostScript, Java, Python, or any of the countless other programming languages and environments in which it has proved to be an effective graph-processing method. On the one hand, our code is informed by experience with implementing algorithms in these and numerous other languages (C and C++ versions of this book are also available); on the other hand, some of the properties of some of these languages are informed by their designers' experience with some of the algorithms and data structures that we consider in this book. In the end, we feel that the code presented in the book both precisely defines the algorithms and is useful in practice.

Acknowledgments

Many people gave me helpful feedback on earlier versions of this book. In particular, thousands of students at Princeton and Brown have suffered through preliminary drafts over the years. Special thanks are due to Trina Avery and Tom Freeman for their help in producing the first edition; to Janet Incerpi for her creativity and ingenuity in persuading our early and primitive digital computerized typesetting hardware and software to produce the first edition; to Marc Brown for his part in the algorithm visualization research that was the genesis of so many of the figures in the book; to Dave Hanson and Andrew Appel for their willingness to answer all of my questions about programming languages; and to Kevin Wayne, for patiently answering my basic questions about networks. Kevin urged me to include the network simplex algorithm in this book, but I was not persuaded that it would be possible to do so until I saw a presentation by Ulrich Lauther at Dagstuhl of the ideas on which the implementations in Chapter 22 are based. I would also like to thank the many readers who have provided me with comments about various editions, including Guy Almes, Jon Bentley, Marc Brown, Jay Gischer, Allan Heydon, Kennedy Lemke, Udi Manber, Michael Quinn, Dana Richards, John Reif, M. Rosenfeld, Stephen Seidman, and William Ward.

To produce this new edition, I have had the pleasure of working with Peter Gordon and Helen Goldstein at Addison-Wesley, who have patiently shepherded this project as it has evolved. It has also been my pleasure to work with several other members of the professional staff at Addison-Wesley. The nature of this project made the book a somewhat unusual challenge for many of them, and I much appreciate their forbearance.

I have gained three new mentors while writing this book and particularly want to express my appreciation to them. First, Steve Summit carefully checked early versions of the manuscript on a technical level and provided me with literally thousands of detailed comments, particularly on the programs. Steve clearly understood my goal of providing elegant, efficient, and effective implementations, and his comments not only helped me to provide a measure of consistency across the implementations, but also helped me to improve many of them substantially. Second, Lyn Dupré also provided me with thousands of de-

tailed comments on the manuscript, which were invaluable in helping me not only to correct and avoid grammatical errors, but also—more important—to find a consistent and coherent writing style that helps bind together the daunting mass of technical material here. Third, Chris Van Wyk, in a long series of spirited electronic mail exchanges, patiently defended the basic precepts of object-oriented programming and helped me develop a style of coding that exhibits the algorithms with clarity and precision while still taking advantage of what object-oriented programming has to offer. The approach that we developed for the C++ version of this book has substantially influenced the Java code here and will certainly influence future volumes in both languages (and C as well). I am extremely grateful for the opportunity to learn from Steve, Lyn, and Chris—their input was vital in the development of this book.

Much of what I have written here I have learned from the teaching and writings of Don Knuth, my advisor at Stanford. Although Don had no direct influence on this work, his presence may be felt in the book, for it was he who put the study of algorithms on the scientific footing that makes a work such as this possible. My friend and colleague Philippe Flajolet, who has been a major force in the development of the analysis of algorithms as a mature research area, has had a similar influence on this work.

I am deeply thankful for the support of Princeton University, Brown University, and the Institut National de Recherche en Informatique et Automatique (INRIA), where I did most of the work on the book; and of the Institute for Defense Analyses and the Xerox Palo Alto Research Center, where I did some work on the book while visiting. Many parts of the book are dependent on research that has been generously supported by the National Science Foundation and the Office of Naval Research. Finally, I thank Bill Bowen, Aaron Lemonick, and Neil Rudenstine for their support in building an academic environment at Princeton in which I was able to prepare this book, despite my numerous other responsibilities.

Robert Sedgewick
Marly-le-Roi, France, 1983
Princeton, New Jersey, 1990, 1992
Jamestown, Rhode Island, 1997, 2001
Princeton, New Jersey, 1998, 2003

Java Consultant's Preface

In the past decade, Java has become the language of choice for a variety of applications. But Java developers have found themselves repeatedly referring to references such as Sedgewick's *Algorithms in* C for solutions to common programming problems. There has long been an empty space on the bookshelf for a comparable reference work for Java; this series of books is here to fill that space.

We wrote the sample programs as utility methods to be used in a variety of contexts. To that end, we did not use the Java package mechanism. To focus on the algorithms at hand (and to expose the algorithmic basis of many fundamental library classes), we avoided the standard Java library in favor of more fundamental types. Proper error checking and other defensive practices would both substantially increase the amount of code and distract the reader from the core algorithms. Developers should introduce such code when using the programs in larger applications.

Although the algorithms we present are language independent, we have paid close attention to Java-specific performance issues. The timings throughout the book are provided as one context for comparing algorithms and will vary depending on the virtual machine. As Java environments evolve, programs will perform as fast as natively compiled code, but such optimizations will not change the performance of algorithms relative to one another. We provide the timings as a useful reference for such comparisons.

I would like to thank Mike Zamansky, for his mentorship and devotion to the teaching of computer science, and Daniel Chaskes, Jason Sanders, and James Percy, for their unwavering support. I would also like to thank my family for their support and for the computer that bore my first programs. Bringing together Java with the classic algorithms of computer science was an exciting endeavor for which I am very grateful. Thank you, Bob, for the opportunity to do so.

Michael Schidlowsky
Oakland Gardens, New York, 2003

Notes on Exercises

Classifying exercises is an activity fraught with peril because readers of a book such as this come to the material with various levels of knowledge and experience. Nonetheless, guidance is appropriate, so many of the exercises carry one of four annotations to help you decide how to approach them.

Exercises that *test your understanding* of the material are marked with an open triangle, as follows:

> ▷ 18.34 Consider the graph
>
> ```
> 3-7 1-4 7-8 0-5 5-2 3-8 2-9 0-6 4-9 2-6 6-4.
> ```
>
> Draw its DFS tree and use the tree to find the graph's bridges and edge-connected components.

Most often, such exercises relate directly to examples in the text. They should present no special difficulty, but working them might teach you a fact or concept that may have eluded you when you read the text.

Exercises that *add new and thought-provoking* information to the material are marked with an open circle, as follows:

> ○ 19.106 Write a program that counts the number of different possible results of topologically sorting a given DAG.

Such exercises encourage you to think about an important concept that is related to the material in the text, or to answer a question that may have occurred to you when you read the text. You may find it worthwhile to read these exercises, even if you do not have the time to work them through.

Exercises that are intended to *challenge you* are marked with a black dot, as follows:

> • 20.73 Describe how you would find the MST of a graph so large that only V edges can fit into main memory at once.

Such exercises may require a substantial amount of time to complete, depending on your experience. Generally, the most productive approach is to work on them in a few different sittings.

A few exercises that are *extremely difficult* (by comparison with most others) are marked with two black dots, as follows:

> •• 20.37 Develop a reasonable generator for random graphs with V vertices and E edges such that the running time of the heap-based PFS implementation of Dijkstra's algorithm is superlinear.

These exercises are similar to questions that might be addressed in the research literature, but the material in the book may prepare you to enjoy trying to solve them (and perhaps succeeding).

The annotations are intended to be neutral with respect to your programming and mathematical ability. Those exercises that require expertise in programming or in mathematical analysis are self-evident. All readers are encouraged to test their understanding of the algorithms by implementing them. Still, an exercise such as this one is straightforward for a practicing programmer or a student in a programming course, but may require substantial work for someone who has not recently programmed:

> • 17.74 Write a program that generates V random points in the plane, then builds a network with edges (in both directions) connecting all pairs of points within a given distance d of one another (see Program 3.20), setting each edge's weight to the distance between the two points that it connects. Determine how to set d so that the expected number of edges is E.

In a similar vein, all readers are encouraged to strive to appreciate the analytic underpinnings of our knowledge about properties of algorithms. Still, an exercise such as this one is straightforward for a scientist or a student in a discrete mathematics course, but may require substantial work for someone who has not recently done mathematical analysis:

> ○ 19.5 How many digraphs correspond to each undirected graph with V vertices and E edges?

There are far too many exercises for you to read and assimilate them all; my hope is that there are enough exercises here to stimulate you to strive to come to a broader understanding on the topics that interest you than you can glean by simply reading the text.

Contents

PART
FIVE

Graph Algorithms