

教育部高等教育司推荐
国外优秀信息科学与技术系列教学用书

数据结构与程序设计

—— C++ 语言描述

(影印版)

DATA STRUCTURES AND PROGRAM DESIGN IN C++

■ Robert L. Kruse
Alexander J. Ryba



高等教育出版社
Higher Education Press
Pearson Education
出版集团



教育部高等教育司推荐
国外优秀信息科学与技术系列教学用书

数据结构与程序设计

——C++语言描述

(影印版)

DATA STRUCTURES AND PROGRAM DESIGN IN C++

Robert L. Kruse
Alexander J. Ryba



高等教育出版社



Pearson Education 出版集团

图字：01-2001-1043 号

English Reprint Copyright © 2001 by Higher Education Press and Pearson Education North Asia Limited.

Data Structures and Program Design in C++

By Robert L. Kruse & Alexander J. Ryba

Copyright © 1999

All Rights Reserved

Published by arrangement with Prentice Hall, Inc., a Pearson Education company

This edition is authorized for sale only in the People's Republic of China (excluding the Special Administrative Regions of Hong Kong and Macau)

图书在版编目(CIP)数据

数据结构与程序设计：C++语言描述：英文/(美)克鲁斯
(Kruse, R.L.)等.—北京：高等教育出版社，2001(2002重印)
ISBN 7-04-010039-8

I. 数… II. 克… III. ① 数据结构—英文 ② C语言—程序
设计—英文 IV. TP311.12

中国版本图书馆 CIP 数据核字(2001)第 19797 号

数据结构与程序设计——C++语言描述

Robert L. Kruse 等

出版发行 高等教育出版社

社 址 北京市东城区沙滩后街 55 号

邮政编码 100009

传 真 010-64014048

经 销 新华书店北京发行所

印 刷 北京民族印刷厂

开 本 787×1092 1/16

印 张 46.00

字 数 1 026 000

购书热线 010-64054588

免费咨询 800-810-0598

网 址 <http://www.hep.edu.cn>
<http://www.hep.com.cn>

版 次 2001 年 5 月第 1 版

印 次 2002 年 10 月第 3 次印刷

定 价 39.00 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

版权所有 侵权必究

前 言

20 世纪末,以计算机和通信技术为代表的信息科学和技术,对世界的经济、军事、科技、教育、文化、卫生等方面的发展产生了深刻的影响,由此而兴起的信息产业已经成为世界经济发展的支柱。进入 21 世纪,各国为了加快本国的信息产业,加大了资金投入和政策扶持。

为了加快我国信息产业的进程,在我国《国民经济和社会发展第十个五年计划纲要》中,明确提出“以信息化带动工业化,发挥后发优势,实现社会生产力的跨越式发展。”信息产业的国际竞争将日趋激烈。在我国加入 WTO 后,我国信息产业将面临国外竞争对手的严峻挑战。竞争成败最终将取决于信息科学和技术人才的多少与优劣。

在 20 世纪末,我国信息产业虽然得到迅猛发展,但与国际先进国家相比,差距还很大。为了赶上并超过国际先进水平,我国必须加快信息技术人才的培养,特别要培养一大批具有国际竞争能力的高水平的信息技术人才,促进我国信息产业和国家信息化水平的全面提高。为此,教育部高等教育司根据教育部吕福源副部长的意见,在长期重视推动高等学校信息科学和技术的教学的基础上,将实施超前发展战略,采取一些重要举措,加快推动高等学校的信息科学和技术等相关专业的教学工作。在大力宣传、推荐我国专家编著的面向 21 世纪和“九五”重点的信息科学和技术课程教材的基础上,在有条件的高等学校的某些信息科学和技术课程中推动使用国外优秀教材的影印版进行英语或双语教学,以缩短我国在计算机教学上与国际先进水平的差距,同时也有助于强化我国大学生的英语水平。

为了达到上述目的,在分析一些出版社已影印相关教材,一些学校已试用影印教材进行教学的基础上,教育部高等教育司组织并委托高等教育出版社开展国外优秀信息科学和技术优秀教材及其教学辅助材料的引进研究与影印出版的试点工作。为推动用影印版教材进行教学创造条件。

本次引进的系列教材的影印出版工作,是在对我国高校的信息科学和技术专业的课程与美国高校的对比分析的基础上展开的;所影印出版的教材均由我国主要高

校的信息科学和技术专家组成的专家组，从国外近两年出版的大量最新教材中精心筛选评审通过的内容新、有影响的优秀教材；影印教材的定价原则上应与我国大学教材价格相当。

教育部高等教育司将此影印系列教材推荐给高等学校，希望有关教师选用，使用后有什么意见和建议请及时反馈。也希望有条件的出版社，根据影印教材的要求，积极参加此项工作，以便引进更多、更新、更好的外国教材和教学辅助材料。

同时，感谢国外有关出版公司对此项引进工作的配合，欢迎更多的国外公司关心并参与此项工作。

教育部高等教育司

二〇〇一年四月

Preface

THE APPRENTICE CARPENTER may want only a hammer and a saw, but a master builder employs many precision tools. Computer programming likewise requires sophisticated tools to cope with the complexity of real applications, and only practice with these tools will build skill in their use. This book treats structured problem solving, object-oriented programming, data abstraction, and the comparative analysis of algorithms as fundamental tools of program design. Several case studies of substantial size are worked out in detail, to show how all the tools are used together to build complete programs.

Many of the algorithms and data structures we study possess an intrinsic elegance, a simplicity that cloaks the range and power of their applicability. Before long the student discovers that vast improvements can be made over the naïve methods usually used in introductory courses. Yet this elegance of method is tempered with uncertainty. The student soon finds that it can be far from obvious which of several approaches will prove best in particular applications. Hence comes an early opportunity to introduce truly difficult problems of both intrinsic interest and practical importance and to exhibit the applicability of mathematical methods to algorithm verification and analysis.

Many students find difficulty in translating abstract ideas into practice. This book, therefore, takes special care in the formulation of ideas into algorithms and in the refinement of algorithms into concrete programs that can be applied to practical problems. The process of data specification and abstraction, similarly, comes before the selection of data structures and their implementations.

We believe in progressing from the concrete to the abstract, in the careful development of motivating examples, followed by the presentation of ideas in a more general form. At an early stage of their careers most students need reinforcement from seeing the immediate application of the ideas that they study, and they require the practice of writing and running programs to illustrate each important concept that they learn. This book therefore contains many sample programs, both short

functions and complete programs of substantial length. The exercises and programming projects, moreover, constitute an indispensable part of the book. Many of these are immediate applications of the topic under study, often requesting that programs be written and run, so that algorithms may be tested and compared. Some are larger projects, and a few are suitable for use by a small group of students working together.

Our programs are written in the popular object-oriented language C++. We take the view that many object-oriented techniques provide natural implementations for basic principles of data-structure design. In this way, C++ allows us to construct safe, efficient, and simple implementations of data-structures. We recognize that C++ is sufficiently complex that students will need to use the experience of a data structures courses to develop and refine their understanding of the language. We strive to support this development by carefully introducing and explaining various object-oriented features of C++ as we progress through the book. Thus, we begin Chapter 1 assuming that the reader is comfortable with the elementary parts of C++ (essentially, with the C subset), and gradually we add in such object-oriented elements of C++ as classes, methods, constructors, inheritance, dynamic memory management, destructors, copy constructors, overloaded functions and operations, templates, virtual functions, and the STL. Of course, our primary focus is on the data structures themselves, and therefore students with relatively little familiarity with C++ will need to supplement this text with a C++ programming text.

SYNOPSIS

Programming Principles

By working through the first large project (CONWAY's game of Life), Chapter 1 expounds principles of object-oriented program design, top-down refinement, review, and testing, principles that the student will see demonstrated and is expected to follow throughout the sequel. At the same time, this project provides an opportunity for the student to review the syntax of elementary features of C++, the programming language used throughout the book.

Introduction to Stacks

Chapter 2 introduces the first data structure we study, the stack. The chapter applies stacks to the development of programs for reversing input, for modelling a desk calculator, and for checking the nesting of brackets. We begin by utilizing the STL stack implementation, and later develop and use our own stack implementation. A major goal of Chapter 2 is to bring the student to appreciate the ideas behind information hiding, encapsulation and data abstraction and to apply methods of top-down design to data as well as to algorithms. The chapter closes with an introduction to abstract data types.

Queues

Queues are the central topic of Chapter 3. The chapter expounds several different implementations of the abstract data type and develops a large application program showing the relative advantages of different implementations. In this chapter we introduce the important object-oriented technique of inheritance.

Linked Stacks and Queues

Chapter 4 presents linked implementations of stacks and queues. The chapter begins with a thorough introduction to pointers and dynamic memory management in C++. After exhibiting a simple linked stack implementation, we discuss

destructors, copy constructors, and overloaded assignment operators, all of which are needed in the safe C++ implementation of linked structures.

Recursion

Chapter 5 continues to elucidate stacks by studying their relationship to problem solving and programming with recursion. These ideas are reinforced by exploring several substantial applications of recursion, including backtracking and tree-structured programs. This chapter can, if desired, be studied earlier in a course than its placement in the book, at any time after the completion of Chapter 2.

Lists and Strings

More general lists with their linked and contiguous implementations provide the theme for Chapter 6. The chapter also includes an encapsulated string implementation, an introduction to C++ templates, and an introduction to algorithm analysis in a very informal way.

Searching

Chapter 7, Chapter 8, and Chapter 9 present algorithms for searching, sorting, and table access (including hashing), respectively. These chapters illustrate the

Sorting

interplay between algorithms and the associated abstract data types, data structures, and implementations. The text introduces the “big- O ” and related notations for elementary algorithm analysis and highlights the crucial choices to be made

Tables and Information Retrieval

regarding best use of space, time, and programming effort. These choices require that we find analytical methods to assess algorithms, and producing such analyses is a battle for which combinatorial mathematics must provide the arsenal. At an elementary level we can expect students neither to be well armed nor to possess the mathematical maturity needed to hone their skills to perfection. Our goal, therefore, is to help students recognize the importance of such skills in anticipation of later chances to study mathematics.

Binary Trees

Binary trees are surely among the most elegant and useful of data structures. Their study, which occupies Chapter 10, ties together concepts from lists, searching, and sorting. As recursively defined data structures, binary trees afford an excellent opportunity for the student to become comfortable with recursion applied both to data structures and algorithms. The chapter begins with elementary topics and progresses as far as such advanced topics as splay trees and amortized algorithm analysis.

Multiway Trees

Chapter 11 continues the study of more sophisticated data structures, including tries, B-trees, and red-black trees.

Graphs

Chapter 12 introduces graphs as more general structures useful for problem solving, and introduces some of the classical algorithms for shortest paths and minimal spanning trees in graphs.

Case Study: The Polish Notation

The case study in Chapter 13 examines the Polish notation in considerable detail, exploring the interplay of recursion, trees, and stacks as vehicles for problem solving and algorithm development. Some of the questions addressed can serve as an informal introduction to compiler design. As usual, the algorithms are fully developed within a functioning C++ program. This program accepts as input an expression in ordinary (infix) form, translates the expression into postfix form, and evaluates the expression for specified values of the variable(s). Chapter 13 may be studied anytime after the completion of Section 10.1.

The appendices discuss several topics that are not properly part of the book’s subject but that are often missing from the student’s preparation.

Mathematical Methods

Appendix A presents several topics from discrete mathematics. Its final two sections, Fibonacci numbers and Catalan numbers, are more advanced and not

needed for any vital purpose in the text, but are included to encourage combinatorial interest in the more mathematically inclined.

Random Numbers Appendix B discusses pseudorandom numbers, generators, and applications, a topic that many students find interesting, but which often does not fit anywhere in the curriculum.

Packages and Utility Functions Appendix C catalogues the various utility and data-structure packages that are developed and used many times throughout this book. Appendix C discusses declaration and definition files, translation units, the utility package used throughout the book, and a package for calculating CPU times.

Programming Precepts, Pointers, and Pitfalls Appendix D, finally, collects all the Programming Precepts and all the Pointers and Pitfalls scattered through the book and organizes them by subject for convenience of reference.

COURSE STRUCTURE

prerequisite The prerequisite for this book is a first course in programming, with experience using the elementary features of C++. However, since we are careful to introduce sophisticated C++ techniques only gradually, we believe that, used in conjunction with a supplementary C++ textbook and extra instruction and emphasis on C++ language issues, this text provides a data structures course in C++ that remains suitable even for students whose programming background is in another language such as C, Pascal, or Java.

A good knowledge of high school mathematics will suffice for almost all the algorithm analyses, but further (perhaps concurrent) preparation in discrete mathematics will prove valuable. Appendix A reviews all required mathematics.

content This book is intended for courses such as the ACM Course CS2 (*Program Design and Implementation*), ACM Course CS7 (*Data Structures and Algorithm Analysis*), or a course combining these. Thorough coverage is given to most of the ACM/IEEE knowledge units¹ on data structures and algorithms. These include:

- AL1 Basic data structures, such as arrays, tables, stacks, queues, trees, and graphs;
- AL2 Abstract data types;
- AL3 Recursion and recursive algorithms;
- AL4 Complexity analysis using the big Oh notation;
- AL6 Sorting and searching; and
- AL8 Practical problem-solving strategies, with large case studies.

The three most advanced knowledge units, AL5 (complexity classes, NP-complete problems), AL7 (computability and undecidability), and AL9 (parallel and distributed algorithms) are not treated in this book.

¹ See *Computing Curricula 1991: Report of the ACM/IEEE-CS Joint Curriculum Task Force*, ACM Press, New York, 1990.

Most chapters of this book are structured so that the core topics are presented first, followed by examples, applications, and larger case studies. Hence, if time allows only a brief study of a topic, it is possible, with no loss of continuity, to move rapidly from chapter to chapter covering only the core topics. When time permits, however, both students and instructor will enjoy the occasional excursion into the supplementary topics and worked-out projects.

two-term course

A two-term course can cover nearly the entire book, thereby attaining a satisfying integration of many topics from the areas of problem solving, data structures, program development, and algorithm analysis. Students need time and practice to understand general methods. By combining the studies of data abstraction, data structures, and algorithms with their implementations in projects of realistic size, an integrated course can build a solid foundation on which, later, more theoretical courses can be built. Even if it is not covered in its entirety, this book will provide enough depth to enable interested students to continue using it as a reference in later work. It is important in any case to assign major programming projects and to allow adequate time for their completion.

SUPPLEMENTARY MATERIALS

A CD-ROM version of this book is anticipated that, in addition to the entire contents of the book, will include:

- All packages, programs, and other C++ code segments from the text, in a form ready to incorporate as needed into other programs;
- Executable versions (for DOS or Windows) of several demonstration programs and nearly all programming projects from the text;
- Brief outlines or summaries of each section of the text, suitable for use as a study guide.

These materials will also be available from the publisher's internet site. To reach these files with ftp, log in as user anonymous to the site <ftp.prenhall.com> and change to the directory

`pub/esm/computer_science.s-041/kruse/cpp`

Instructors teaching from this book may obtain, at no charge, an instructor's version on CD-ROM which, in addition to all the foregoing materials, includes:

- Brief teaching notes on each chapter;
- Full solutions to nearly all exercises in the textbook;
- Full source code to nearly all programming projects in the textbook;
- Transparency masters.

BOOK PRODUCTION

This book and its supplements were written and produced with software called PreTeX, a preprocessor and macro package for the TeX typesetting system.² PreTeX, by exploiting context dependency, automatically supplies much of the typesetting markup required by TeX. PreTeX also supplies several tools that greatly simplify some aspects of an author's work. These tools include a powerful cross-reference system, simplified typesetting of mathematics and computer-program listings, and automatic generation of the index and table of contents, while allowing the processing of the book in conveniently small files at every stage. Solutions, placed with exercises and projects, are automatically removed from the text and placed in a separate document.

For a book such as this, PreTeX's treatment of computer programs is its most important feature. Computer programs are not included with the main body of the text; instead, they are placed in separate, secondary files, along with any desired explanatory text, and with any desired typesetting markup in place. By placing tags at appropriate places in the secondary files, PreTeX can extract arbitrary parts of a secondary file, in any desired order, for typesetting with the text. Another utility removes all the tags, text, and markup, producing as its output a program ready to be compiled. The same input file thus automatically produces both typeset program listings and compiled program code. In this way, the reader gains increased confidence in the accuracy of the computer program listings appearing in the text. In fact, with just two exceptions, all of the programs developed in this book have been compiled and successfully tested under the g++ and Borland C++ compilers (versions 2.7.2.1 and 5.0, respectively). The two exceptions are the first program in Chapter 2 (which requires a compiler with a full ANSI C++ standard library) and the last program of Chapter 13 (which requires a compiler with certain Borland graphics routines).

ACKNOWLEDGMENTS

Over the years, the Pascal and C antecedents of this book have benefitted greatly from the contributions of many people: family, friends, colleagues, and students, some of whom are noted in the previous books. Many other people, while studying these books or their translations into various languages, have kindly forwarded their comments and suggestions, all of which have helped to make this a better book.

We are happy to acknowledge the suggestions of the following reviewers, who have helped in many ways to improve the presentation in this book: KEITH VANDER LINDEN (Calvin College), JENS GREGOR (University of Tennessee), VICTOR BERRY (Boston University), JEFFERY LEON (University of Illinois at Chicago), SUSAN

² TeX was developed by DONALD E. KNUTH, who has also made many important research contributions to data structures and algorithms. (See the entries under his name in the index.)

HUTT (University of Missouri–Columbia), FRED HARRIS (University of Nevada), ZHI-LI ZHANG (University of Minnesota), and ANDREW SUNG (New Mexico Institute of Technology).

ALEX RYBA especially acknowledges the helpful suggestions and encouraging advice he has received over the years from WIM RUITENBURG and JOHN SIMMS of Marquette University, as well as comments from former students RICK VOGEL and JUN WANG.

It is a special pleasure for ROBERT KRUSE to acknowledge the continuing advice and help of PAUL MAILHOT of PreTeX, Inc., who was from the first an outstanding student, then worked as a dependable research assistant, and who has now become a valued colleague making substantial contributions in software development for book production, in project management, in problem solving for the publisher, the printer, and the authors, and in providing advice and encouragement in all aspects of this work. The CD-ROM versions of this book, with all their hypertext features (such as extensive cross-reference links and execution of demonstration programs from the text), are entirely his accomplishment.

Without the continuing enthusiastic support, faithful encouragement, and patience of the editorial staff of Prentice Hall, especially ALAN APT, Publisher, LAURA STEELE, Acquisitions Editor, and MARCIA HORTON, Editor in Chief, this project would never have been started and certainly could never have been brought to completion. Their help, as well as that of the production staff named on the copyright page, has been invaluable.

ROBERT L. KRUSE
ALEXANDER J. RYBA

Contents

Preface xi

Synopsis xiii
Course Structure xiv
Supplementary Materials xv
Book Production xvi
Acknowledgments xvi

1 Programming Principles 1

1.1 Introduction 2

1.2 The Game of Life 4

- 1.2.1 Rules for the Game of Life 4
- 1.2.2 Examples 5
- 1.2.3 The Solution: Classes, Objects, and Methods 7
- 1.2.4 Life: The Main Program 8

1.3 Programming Style 10

- 1.3.1 Names 10
- 1.3.2 Documentation and Format 13
- 1.3.3 Refinement and Modularity 15

1.4 Coding, Testing, and Further Refinement 20

- 1.4.1 Stubs 20
- 1.4.2 Definition of the Class Life 22
- 1.4.3 Counting Neighbors 23
- 1.4.4 Updating the Grid 24
- 1.4.5 Input and Output 25
- 1.4.6 Drivers 27

- 1.4.7 Program Tracing 28
- 1.4.8 Principles of Program Testing 29

1.5 Program Maintenance 33

- 1.5.1 Program Evaluation 34
- 1.5.2 Review of the Life Program 35
- 1.5.3 Program Revision and Redevelopment 38

1.6 Conclusions and Preview 39

- 1.6.1 Software Engineering 39
- 1.6.2 Problem Analysis 40
- 1.6.3 Requirements Specification 41
- 1.6.4 Coding 41

Pointers and Pitfalls 45

Review Questions 46

References for Further Study 47

- C++ 47
- Programming Principles 47
- The Game of Life 47
- Software Engineering 48

2 Introduction to Stacks 49

2.1 Stack Specifications 50

- 2.1.1 Lists and Arrays 50
- 2.1.2 Stacks 50
- 2.1.3 First Example: Reversing a List 51
- 2.1.4 Information Hiding 54
- 2.1.5 The Standard Template Library 55

2.2 Implementation of Stacks	57
2.2.1 Specification of Methods for Stacks	57
2.2.2 The Class Specification	60
2.2.3 Pushing, Popping, and Other Methods	61
2.2.4 Encapsulation	63
2.3 Application: A Desk Calculator	66
2.4 Application: Bracket Matching	69
2.5 Abstract Data Types and Their Implementations	71
2.5.1 Introduction	71
2.5.2 General Definitions	73
2.5.3 Refinement of Data Specification	74
Pointers and Pitfalls	76
Review Questions	76
References for Further Study	77

3 Queues 78

3.1 Definitions	79
3.1.1 Queue Operations	79
3.1.2 Extended Queue Operations	81
3.2 Implementations of Queues	84
3.3 Circular Implementation of Queues in C++	89
3.4 Demonstration and Testing	93
3.5 Application of Queues: Simulation	96
3.5.1 Introduction	96
3.5.2 Simulation of an Airport	96
3.5.3 Random Numbers	99
3.5.4 The Runway Class Specification	99
3.5.5 The Plane Class Specification	100
3.5.6 Functions and Methods of the Simulation	101
3.5.7 Sample Results	107
Pointers and Pitfalls	110
Review Questions	110
References for Further Study	111

4 Linked Stacks and Queues 112

4.1 Pointers and Linked Structures	113
4.1.1 Introduction and Survey	113
4.1.2 Pointers and Dynamic Memory in C++	116
4.1.3 The Basics of Linked Structures	122
4.2 Linked Stacks	127
4.3 Linked Stacks with Safeguards	131
4.3.1 The Destructor	131
4.3.2 Overloading the Assignment Operator	132
4.3.3 The Copy Constructor	135
4.3.4 The Modified Linked-Stack Specification	136
4.4 Linked Queues	137
4.4.1 Basic Declarations	137
4.4.2 Extended Linked Queues	139
4.5 Application: Polynomial Arithmetic	141
4.5.1 Purpose of the Project	141
4.5.2 The Main Program	141
4.5.3 The Polynomial Data Structure	144
4.5.4 Reading and Writing Polynomials	147
4.5.5 Addition of Polynomials	148
4.5.6 Completing the Project	150
4.6 Abstract Data Types and Their Implementations	152
Pointers and Pitfalls	154
Review Questions	155

5 Recursion 157

5.1 Introduction to Recursion	158
5.1.1 Stack Frames for Subprograms	158
5.1.2 Tree of Subprogram Calls	159
5.1.3 Factorials:	
A Recursive Definition	160
5.1.4 Divide and Conquer:	
The Towers of Hanoi	163
5.2 Principles of Recursion	170
5.2.1 Designing Recursive Algorithms	170
5.2.2 How Recursion Works	171
5.2.3 Tail Recursion	174
5.2.4 When Not to Use Recursion	176
5.2.5 Guidelines and Conclusions	180

5.3 Backtracking: Postponing the Work	183
5.3.1 Solving the Eight-Queens Puzzle	183
5.3.2 Example: Four Queens	184
5.3.3 Backtracking	185
5.3.4 Overall Outline	186
5.3.5 Refinement: The First Data Structure and Its Methods	188
5.3.6 Review and Refinement	191
5.3.7 Analysis of Backtracking	194
5.4 Tree-Structured Programs: Look-Ahead in Games	198
5.4.1 Game Trees	198
5.4.2 The Minimax Method	199
5.4.3 Algorithm Development	201
5.4.4 Refinement	203
5.4.5 Tic-Tac-Toe	204
Pointers and Pitfalls	209
Review Questions	210
References for Further Study	211

6 Lists and Strings 212

6.1 List Definition	213
6.1.1 Method Specifications	214
6.2 Implementation of Lists	217
6.2.1 Class Templates	218
6.2.2 Contiguous Implementation	219
6.2.3 Simply Linked Implementation	221
6.2.4 Variation: Keeping the Current Position	225
6.2.5 Doubly Linked Lists	227
6.2.6 Comparison of Implementations	230
6.3 Strings	233
6.3.1 Strings in C++	233
6.3.2 Implementation of Strings	234
6.3.3 Further String Operations	238
6.4 Application: A Text Editor	242
6.4.1 Specifications	242
6.4.2 Implementation	243
6.5 Linked Lists in Arrays	251
6.6 Application: Generating Permutations	260
Pointers and Pitfalls	265
Review Questions	266
References for Further Study	267

7 Searching 268

7.1 Searching: Introduction and Notation	269
7.2 Sequential Search	271
7.3 Binary Search	278
7.3.1 Ordered Lists	278
7.3.2 Algorithm Development	280
7.3.3 The Forgetful Version	281
7.3.4 Recognizing Equality	284
7.4 Comparison Trees	286
7.4.1 Analysis for $n = 10$	287
7.4.2 Generalization	290
7.4.3 Comparison of Methods	294
7.4.4 A General Relationship	296
7.5 Lower Bounds	297
7.6 Asymptotics	302
7.6.1 Introduction	302
7.6.2 Orders of Magnitude	304
7.6.3 The Big-O and Related Notations	310
7.6.4 Keeping the Dominant Term	311
Pointers and Pitfalls	314
Review Questions	315
References for Further Study	316

8 Sorting 317

8.1 Introduction and Notation	318
8.1.1 Sortable Lists	319
8.2 Insertion Sort	320
8.2.1 Ordered Insertion	320
8.2.2 Sorting by Insertion	321
8.2.3 Linked Version	323
8.2.4 Analysis	325
8.3 Selection Sort	329
8.3.1 The Algorithm	329
8.3.2 Contiguous Implementation	330
8.3.3 Analysis	331
8.3.4 Comparisons	332
8.4 Shell Sort	333
8.5 Lower Bounds	336

8.6 Divide-and-Conquer Sorting	339
8.6.1 The Main Ideas	339
8.6.2 An Example	340
8.7 Mergesort for Linked Lists	344
8.7.1 The Functions	345
8.7.2 Analysis of Mergesort	348
8.8 Quicksort for Contiguous Lists	352
8.8.1 The Main Function	352
8.8.2 Partitioning the List	353
8.8.3 Analysis of Quicksort	356
8.8.4 Average-Case Analysis of Quicksort	358
8.8.5 Comparison with Mergesort	360
8.9 Heaps and Heapsort	363
8.9.1 Two-Way Trees as Lists	363
8.9.2 Development of Heapsort	365
8.9.3 Analysis of Heapsort	368
8.9.4 Priority Queues	369

8.10 Review: Comparison of Methods	372
---	------------

Pointers and Pitfalls	375
------------------------------	------------

Review Questions	376
-------------------------	------------

References for Further Study	377
-------------------------------------	------------

9 Tables and Information Retrieval — 379

9.1 Introduction: Breaking the $\lg n$ Barrier	380
9.2 Rectangular Tables	381
9.3 Tables of Various Shapes	383
9.3.1 Triangular Tables	383
9.3.2 Jagged Tables	385
9.3.3 Inverted Tables	386
9.4 Tables: A New Abstract Data Type	388
9.5 Application: Radix Sort	391
9.5.1 The Idea	392
9.5.2 Implementation	393
9.5.3 Analysis	396
9.6 Hashing	397
9.6.1 Sparse Tables	397
9.6.2 Choosing a Hash Function	399
9.6.3 Collision Resolution with Open Addressing	401
9.6.4 Collision Resolution by Chaining	406
9.7 Analysis of Hashing	411

9.8 Conclusions: Comparison of Methods	417
9.9 Application: The Life Game Revisited	418
9.9.1 Choice of Algorithm	418
9.9.2 Specification of Data Structures	419
9.9.3 The Life Class	421
9.9.4 The Life Functions	421

Pointers and Pitfalls	426
------------------------------	------------

Review Questions	427
-------------------------	------------

References for Further Study	428
-------------------------------------	------------

10 Binary Trees — 429

10.1 Binary Trees	430
10.1.1 Definitions	430
10.1.2 Traversal of Binary Trees	432
10.1.3 Linked Implementation of Binary Trees	437
10.2 Binary Search Trees	444
10.2.1 Ordered Lists and Implementations	446
10.2.2 Tree Search	447
10.2.3 Insertion into a Binary Search Tree	451
10.2.4 Treesort	453
10.2.5 Removal from a Binary Search Tree	455
10.3 Building a Binary Search Tree	463
10.3.1 Getting Started	464
10.3.2 Declarations and the Main Function	465
10.3.3 Inserting a Node	466
10.3.4 Finishing the Task	467
10.3.5 Evaluation	469
10.3.6 Random Search Trees and Optimality	470
10.4 Height Balance: AVL Trees	473
10.4.1 Definition	473
10.4.2 Insertion of a Node	477
10.4.3 Removal of a Node	484
10.4.4 The Height of an AVL Tree	485
10.5 Splay Trees: A Self-Adjusting Data Structure	490
10.5.1 Introduction	490
10.5.2 Splaying Steps	491
10.5.3 Algorithm Development	495

- 10.5.4 Amortized Algorithm Analysis: Introduction 505
- 10.5.5 Amortized Analysis of Splaying 509

Pointers and Pitfalls 515

Review Questions 516

References for Further Study 518

11 Multiway Trees 520

- 11.1 Orchards, Trees, and Binary Trees 521
 - 11.1.1 On the Classification of Species 521
 - 11.1.2 Ordered Trees 522
 - 11.1.3 Forests and Orchards 524
 - 11.1.4 The Formal Correspondence 526
 - 11.1.5 Rotations 527
 - 11.1.6 Summary 527

11.2 Lexicographic Search Trees: Tries 530

- 11.2.1 Tries 530
- 11.2.2 Searching for a Key 530
- 11.2.3 C++ Algorithm 531
- 11.2.4 Searching a Trie 532
- 11.2.5 Insertion into a Trie 533
- 11.2.6 Deletion from a Trie 533
- 11.2.7 Assessment of Tries 534

11.3 External Searching: B-Trees 535

- 11.3.1 Access Time 535
- 11.3.2 Multiway Search Trees 535
- 11.3.3 Balanced Multiway Trees 536
- 11.3.4 Insertion into a B-Tree 537
- 11.3.5 C++ Algorithms: Searching and Insertion 539
- 11.3.6 Deletion from a B-Tree 547

11.4 Red-Black Trees 556

- 11.4.1 Introduction 556
- 11.4.2 Definition and Analysis 557
- 11.4.3 Red-Black Tree Specification 559
- 11.4.4 Insertion 560
- 11.4.5 Insertion Method Implementation 561
- 11.4.6 Removal of a Node 565

Pointers and Pitfalls 566

Review Questions 567

References for Further Study 568

12 Graphs 569

12.1 Mathematical Background 570

- 12.1.1 Definitions and Examples 570
- 12.1.2 Undirected Graphs 571
- 12.1.3 Directed Graphs 571

12.2 Computer Representation 572

- 12.2.1 The Set Representation 572
- 12.2.2 Adjacency Lists 574
- 12.2.3 Information Fields 575

12.3 Graph Traversal 575

- 12.3.1 Methods 575
- 12.3.2 Depth-First Algorithm 577
- 12.3.3 Breadth-First Algorithm 578

12.4 Topological Sorting 579

- 12.4.1 The Problem 579
- 12.4.2 Depth-First Algorithm 580
- 12.4.3 Breadth-First Algorithm 581

12.5 A Greedy Algorithm: Shortest Paths 583

- 12.5.1 The Problem 583
- 12.5.2 Method 584
- 12.5.3 Example 585
- 12.5.4 Implementation 586

12.6 Minimal Spanning Trees 587

- 12.6.1 The Problem 587
- 12.6.2 Method 589
- 12.6.3 Implementation 590
- 12.6.4 Verification of Prim's Algorithm 593

12.7 Graphs as Data Structures 594

Pointers and Pitfalls 596

Review Questions 597

References for Further Study 597

13 Case Study: The Polish Notation 598

13.1 The Problem 599

- 13.1.1 The Quadratic Formula 599

13.2 The Idea 601

- 13.2.1 Expression Trees 601
- 13.2.2 Polish Notation 603