
Preface to the Fourth Edition

This fourth edition is designed to provide an introduction to data structures and algorithms, including their design, analysis, and implementation. In terms of curricula based on the *IEEE/ACM 2001 Computing Curriculum*, this book is appropriate for use in the courses CS102 (I/O/B versions), CS103 (I/O/B versions), CS111 (A version), and CS112 (A/I/O/F/H versions). We discuss its use for such courses in more detail later in this preface.

The major changes, with respect to the third edition, are the following:

- Added new chapter on arrays, linked lists, and recursion.
- Added new chapter on memory management.
- Full integration with Java 5.0.
- Better integration with the Java Collections Framework.
- Better coverage of iterators.
- Increased coverage of array lists, including the replacement of uses of the class `java.util.Vector` with `java.util.ArrayList`.
- Update of all Java APIs to use generic types.
- Simplified list, binary tree, and priority queue ADTs.
- Further streamlining of mathematics to the seven most used functions.
- Expanded and revised exercises, bringing the total number of reinforcement, creativity, and project exercises to 670. Added exercises include new projects on maintaining a game's high-score list, evaluating postfix and infix expressions, minimax game-tree evaluation, processing stock buy and sell orders, scheduling CPU jobs, n -body simulation, computing DNA-strand edit distance, and creating and solving mazes.

This book is related to the following books:

- M.T. Goodrich, R. Tamassia, and D.M. Mount, *Data Structures and Algorithms in C++*, John Wiley & Sons, Inc., 2004. This book has a similar overall structure to the present book, but uses C++ as the underlying language (with some modest, but necessary pedagogical differences required by this approach). Thus, it could make for a handy companion book in a curriculum that allows for either a Java or C++ track in the introductory courses.
- M.T. Goodrich and R. Tamassia, *Algorithm Design: Foundations, Analysis, and Internet Examples*, John Wiley & Sons, Inc., 2002. This is a textbook for a more advanced algorithms and data structures course, such as CS210 (T/W/C/S versions) in the IEEE/ACM 2001 curriculum.

Use as a Textbook

The design and analysis of efficient data structures has long been recognized as a vital subject in computing, for the study of data structures is part of the core of every collegiate computer science and computer engineering major program we are familiar with. Typically, the introductory courses are presented as a two- or three-course sequence. Elementary data structures are often briefly introduced in the first programming or introduction to computer science course and this is followed by a more in-depth introduction to data structures in the following course(s). Furthermore, this course sequence is typically followed at a later point in the curriculum by a more in-depth study of data structures and algorithms. We feel that the central role of data structure design and analysis in the curriculum is fully justified, given the importance of efficient data structures in most software systems, including the Web, operating systems, databases, compilers, and scientific simulation systems.

With the emergence of the object-oriented paradigm as the framework of choice for building robust and reusable software, we have tried to take a consistent object-oriented viewpoint throughout this text. One of the main ideas of the object-oriented approach is that data should be presented as being encapsulated with the methods that access and modify them. That is, rather than simply viewing data as a collection of bytes and addresses, we think of data as instances of an *abstract data type (ADT)* that include a repertory of methods for performing operations on the data. Likewise, object-oriented solutions are often organized utilizing common *design patterns*, which facilitate software reuse and robustness. Thus, we present each data structure using ADTs and their respective implementations and we introduce important design patterns as means to organize those implementations into classes, methods, and objects.

For each ADT presented in this book, we provide an associated Java interface. Also, concrete data structures realizing the ADTs are provided as Java classes implementing the interfaces above. We also give Java implementations of fundamental algorithms (such as sorting and graph traversals) and of sample applications of data structures (such as HTML tag matching and a photo album). Due to space limitations, we sometimes show only code fragments in the book and make the full source code available on the companion Web site, <http://java.datastructures.net>.

The Java code implementing fundamental data structures in this book is organized in a single Java package, `net.datastructures`. This package forms a coherent library of data structures and algorithms in Java specifically designed for educational purposes in a way that is complementary with the Java Collections Framework.

Web Added-Value Education

This book is accompanied by an extensive Web site:

<http://java.datastructures.net>

Students are encouraged to use this site along with the book, to help with exercises and increase understanding of the subject. Instructors are likewise welcome to use the site to help plan, organize, and present their course materials.

For the Student

For all readers, and specifically for students, we include:

- All the Java source code presented in this book.
- The student version of the `net.datastructures` package.
- Slide handouts (four-per-page) in PDF format.
- A database of hints to *all* exercises, indexed by problem number.
- Java animations and interactive applets for data structures and algorithms.
- Hyperlinks to other data structures and algorithms resources.

We feel that the Java animations and interactive applets should be of particular interest, since they allow readers to interactively “play” with different data structures, which leads to better understanding of the different ADTs. In addition, the hints should be of considerable use to anyone needing a little help getting started on certain exercises.

For the Instructor

For instructors using this book, we include the following additional teaching aids:

- Solutions to over two hundred of the book’s exercises.
- A keyword-searchable database of additional exercises.
- The complete `net.datastructures` package.
- Additional Java source code.
- Slides in Powerpoint and PDF (one-per-page) format.
- Self-contained special-topic supplements, including discussions on convex hulls, range trees, and orthogonal segment intersection.

The slides are fully editable, so as to allow an instructor using this book full freedom in customizing his or her presentations.

A Resource for Teaching Data Structures and Algorithms

This book contains many Java-code and pseudo-code fragments, and over 670 exercises, which are divided into roughly 40% reinforcement exercises, 40% creativity exercises, and 20% programming projects.

This book can be used for courses CS102 (I/O/B versions), CS103 (I/O/B versions), CS111 (A version), and/or CS112 (A/I/O/F/H versions) in the IEEE/ACM 2001 Computing Curriculum, with instructional units as outlined in Table 0.1.

Instructional Unit	Relevant Material
PL1. Overview of Programming Languages	Chapters 1 & 2
PL2. Virtual Machines	Sections 14.1.1, 14.1.2, & 14.1.3
PL3. Introduction to Language Translation	Section 1.9
PL4. Declarations and Types	Sections 1.1, 2.4, & 2.5
PL5. Abstraction Mechanisms	Sections 2.4, 5.1, 5.2, 5.3, 6.1.1, 6.2, 6.4, 6.3, 7.1, 7.3.1, 8.1, 9.1, 9.3, 11.6, & 13.1
PL6. Object-Oriented Programming	Chapters 1 & 2 and Sections 6.2.2, 6.3, 7.3.7, 8.1.2, & 13.3.1
PF1. Fundamental Programming Constructs	Chapters 1 & 2
PF2. Algorithms and Problem-Solving	Sections 1.9 & 4.2
PF3. Fundamental Data Structures	Sections 3.1, 5.1–3.2, 5.3, , 6.1–6.4, 7.1, 7.3, 8.1, 8.3, 9.1–9.4, 10.1, & 13.1
PF4. Recursion	Section 3.5
SE1. Software Design	Chapter 2 and Sections 6.2.2, 6.3, 7.3.7, 8.1.2, & 13.3.1
SE2. Using APIs	Sections 2.4, 5.1, 5.2, 5.3, 6.1.1, 6.2, 6.4, 6.3, 7.1, 7.3.1, 8.1, 9.1, 9.3, 11.6, & 13.1
AL1. Basic Algorithmic Analysis	Chapter 3
AL2. Algorithmic Strategies	Sections 11.1.1, 11.7.1, 12.2.1, 12.4.2, & 12.5.2
AL3. Fundamental Computing Algorithms	Sections 8.1.4, 8.2.3, 8.3.5, 9.2, & 9.3.3, and Chapters 10, 11, & 12
DS1. Functions, Relations, and Sets	Sections 4.1, 8.1, & 11.6
DS3. Proof Techniques	Sections 4.3, 6.1.4, 7.3.3, 8.3, 10.2, 10.3, 10.4, 10.5, 11.2.1, 11.3, 11.6.2, 13.1, 13.3.1, 13.4, & 13.5
DS4. Basics of Counting	Sections 2.2.3 & 11.1.5
DS5. Graphs and Trees	Chapters 6, 7, 9, & 12
DS6. Discrete Probability	Appendix A and Sections 9.2.2, 9.4.2, 11.2.1, & 11.7

Table 0.1: Material for Units in the IEEE/ACM 2001 Computing Curriculum.

Chapter Listing

The chapters for this course are organized to provide a pedagogical path that starts with the basics of Java programming and object-oriented design, moves to concrete structures like arrays and linked lists, adds foundational techniques like recursion and algorithm analysis, and then presents the fundamental data structures and algorithms, concluding with a discussion of memory management (that is, the architectural underpinnings of data structures). Specifically, the chapters for this book are organized as follows:

1. **Java Programming Basics**
2. **Object-Oriented Design**
3. **Arrays, Linked Lists, and Recursion**
4. **Analysis Tools**
5. **Stacks and Queues**
6. **Lists and Iterators**
7. **Trees**
8. **Priority Queues**
9. **Maps and Dictionaries**
10. **Search Trees**
11. **Sorting, Sets, and Selection**
12. **Text Processing**
13. **Graphs**
14. **Memory**
- A. **Useful Mathematical Facts**

Prerequisites

We have written this book assuming that the reader comes to it with certain knowledge. That is, we assume that the reader is at least vaguely familiar with a high-level programming language, such as C, C++, or Java, and that he or she understands the main constructs from such a high-level language, including:

- Variables and expressions.
- Methods (also known as functions or procedures).
- Decision structures (such as if-statements and switch-statements).
- Iteration structures (for-loops and while-loops).

For readers who are familiar with these concepts, but not with how they are expressed in Java, we provide a primer on the Java language in Chapter 1. Still, this book is primarily a data structures book, not a Java book; hence, it does not provide a comprehensive treatment of Java. Nevertheless, we do not assume that the reader is necessarily familiar with object-oriented design or with linked structures, such as linked lists, for these topics are covered in the core chapters of this book.

In terms of mathematical background, we assume the reader is somewhat familiar with topics from high-school mathematics. Even so, in Chapter 4, we discuss the seven most-important functions for algorithm analysis. In fact, sections that use something other than one of these seven functions are considered optional, and are indicated with a star (★). We give a summary of other useful mathematical facts, including elementary probability, in Appendix A.

About the Authors

Professors Goodrich and Tamassia are well-recognized researchers in algorithms and data structures, having published many papers in this field, with applications to Internet computing, information visualization, computer security, and geometric computing. They have served as principal investigators in several joint projects sponsored by the National Science Foundation, the Army Research Office, and the Defense Advanced Research Projects Agency. They are also active in educational technology research, with special emphasis on algorithm visualization systems.

Michael Goodrich received his Ph.D. in Computer Science from Purdue University in 1987. He is currently a professor in the Department of Computer Science at University of California, Irvine. Previously, he was a professor at Johns Hopkins University. He is an editor for the *International Journal of Computational Geometry & Applications* and *Journal of Graph Algorithms and Applications*.

Roberto Tamassia received his Ph.D. in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign in 1988. He is currently a professor in the Department of Computer Science at Brown University. He is editor-in-chief for the *Journal of Graph Algorithms and Applications* and an editor for

Computational Geometry: Theory and Applications. He previously served on the editorial board of *IEEE Transactions on Computers*.

In addition to their research accomplishments, the authors also have extensive experience in the classroom. For example, Dr. Goodrich has taught data structures and algorithms courses, including Data Structures as a freshman-sophomore level course and Introduction to Algorithms as an upper level course. He has earned several teaching awards in this capacity. His teaching style is to involve the students in lively interactive classroom sessions that bring out the intuition and insights behind data structuring and algorithmic techniques. Dr. Tamassia has taught Data Structures and Algorithms as an introductory freshman-level course since 1988. One thing that has set his teaching style apart is his effective use of interactive hypermedia presentations integrated with the Web.

The instructional Web sites, datastructures.net and algorithmdesign.net, supported by Drs. Goodrich and Tamassia, are used as reference material by students, teachers, and professionals worldwide.

Acknowledgments

There are a number of individuals who have made contributions to this book.

We are grateful to all our research collaborators and teaching assistants, who provided feedback on early drafts of chapters and have helped us in developing exercises, programming assignments, and algorithm animation systems. In particular, we would like to thank Jeff Achter, James Baker, Ryan Baker, Benjamin Boer, Mike Boilen, Devin Borland, Lubomir Bourdev, Stina Bridgeman, Bryan Cantrill, Yi-Jen Chiang, Robert Cohen, David Emory, Jody Fanto, Ashim Garg, Natasha Gelfand, Mark Handy, Michael Horn, Benoît Hudson, Jovanna Ignatowicz, Seth Padowitz, James Piechota, Dan Polivy, Susannah Raub, Andy Schwerin, Michael Shapiro, Michael Shin, Christian Straub, Galina Shubina, Nikos Triandopoulos, and Luca Vismara. Lubomir Bourdev, Mike Demmer, Mark Handy, Michael Horn, and Scott Speigler developed a basic Java tutorial, which ultimately led to Chapter 1, Java Programming.

Special thanks go to Eric Zamore, who contributed to the development of the Java code examples in this book and to the design, implementation, and testing of the `net.datastructures` library of data structures and algorithms in Java.

Many students and instructors have used the two previous editions of this book and their experiences and responses have helped shape this fourth edition.

There have been a number of friends and colleagues whose comments have lead to improvements in the text. We are particularly thankful to Karen Goodrich, Art Moorshead, David Mount, Scott Smith, and Ioannis Tollis for their insightful comments. In addition, contributions by David Mount to Section 3.5 and to several figures are gratefully acknowledged.

We are also truly indebted to the outside reviewers and readers for their copious comments, emails, and constructive criticism, which were extremely useful in writing the fourth edition. We specifically thank the following reviewers for their comments and suggestions: Divy Agarwal, University of California, Santa Barbara; Terry Andres, University of Manitoba; Bobby Blumofe, University of Texas, Austin; Michael Clancy, University of California, Berkeley; Larry Davis, University of Maryland; Scott Drysdale, Dartmouth College; Arup Guha, University of Central Florida; Chris Ingram, University of Waterloo; Stan Kwasny, Washington University; Calvin Lin, University of Texas at Austin; John Mark Mercer, McGill University; Laurent Michel, University of Connecticut; Leonard Myers, California Polytechnic State University, San Luis Obispo; David Naumann, Stevens Institute of Technology; Robert Pastel, Michigan Technological University; Bina Ramamurthy, SUNY Buffalo; Ken Slonneger, University of Iowa; C.V. Ravishankar, University of Michigan; Val Tannen, University of Pennsylvania; Paul Van Aragon, Messiah College; and Christopher Wilson, University of Oregon.

The team at Wiley has been great. Many thanks go to Lilian Brady, Paul Crockett, Simon Durkin, Lisa Gee, Frank Lyman, Madelyn Lesure, Hope Miller, Bridget Morrisey, Ken Santor, Dan Sayre, Bruce Spatz, Dawn Stanley, and Jeri Warner.

The computing systems and excellent technical support staff in the departments of computer science at Brown University and University of California, Irvine gave us reliable working environments. This manuscript was prepared primarily with the \LaTeX typesetting package for the text and Adobe FrameMaker[®] and Microsoft Visio[®] for the figures.

Finally, we would like to warmly thank Isabel Cruz, Karen Goodrich, Giuseppe Di Battista, Franco Preparata, Ioannis Tollis, and our parents for providing advice, encouragement, and support at various stages of the preparation of this book. We also thank them for reminding us that there are things in life beyond writing books.

Michael T. Goodrich
Roberto Tamassia