Data Structures & Algorithms in Python

MICHAEL T. GOODRICH • ROBERTO TAMASSIA • MICHAEL H. GOLDWASSER

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 LaTeX by the authors. Printed and bound by Courier Kendallville.
The cover was printed by Courier Kendallville.

This book is printed on acid free paper.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

To Karen, Paul, Anna, and Jack
– *Michael T. Goodrich*

To Isabel
– *Roberto Tamassia*

To Susan, Calista, and Maya
– *Michael H. Goldwasser*

Preface

The design and analysis of efficient data structures has long been recognized as a vital subject in computing and is part of the core curriculum of computer science and computer engineering undergraduate degrees. *Data Structures and Algorithms in Python* provides an introduction to data structures and algorithms, including their design, analysis, and implementation. This book is designed for use in a beginning-level data structures course, or in an intermediate-level introduction to algorithms course. We discuss its use for such courses in more detail later in this preface.

To promote the development of 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 objects as instances of an ***abstract data type*** (*ADT*), which includes a repertoire of methods for performing operations on data objects of this type. We then emphasize that there may be several different implementation strategies for a particular ADT, and explore the relative pros and cons of these choices. We provide complete Python implementations for almost all data structures and algorithms discussed, and we introduce important object-oriented ***design patterns*** as means to organize those implementations into reusable components.

Desired outcomes for readers of our book include that:

- They have knowledge of the most common abstractions for data collections (e.g., stacks, queues, lists, trees, maps).
- They understand algorithmic strategies for producing efficient realizations of common data structures.
- They can analyze algorithmic performance, both theoretically and experimentally, and recognize common trade-offs between competing strategies.
- They can wisely use existing data structures and algorithms found in modern programming language libraries.
- They have experience working with concrete implementations for most foundational data structures and algorithms.
- They can apply data structures and algorithms to solve complex problems.

In support of the last goal, we present many example applications of data structures throughout the book, including the processing of file systems, matching of tags in structured formats such as HTML, simple cryptography, text frequency analysis, automated geometric layout, Huffman coding, DNA sequence alignment, and search engine indexing.

Book Features

This book is based upon the book *Data Structures and Algorithms in Java* by Goodrich and Tamassia, and the related *Data Structures and Algorithms in C++* by Goodrich, Tamassia, and Mount. However, this book is not simply a translation of those other books to Python. In adapting the material for this book, we have significantly redesigned the organization and content of the book as follows:

- The code base has been entirely redesigned to take advantage of the features of Python, such as use of generators for iterating elements of a collection.
- Many algorithms that were presented as pseudo-code in the Java and C++ versions are directly presented as complete Python code.
- In general, ADTs are defined to have consistent interface with Python's built-in data types and those in Python's collections module.
- Chapter 5 provides an in-depth exploration of the dynamic array-based underpinnings of Python's built-in list, tuple, and str classes. New Appendix A serves as an additional reference regarding the functionality of the str class.
- Over 450 illustrations have been created or revised.
- New and revised exercises bring the overall total number to 750.

Online Resources

This book is accompanied by an extensive set of online resources, which can be found at the following Web site:

www.wiley.com/college/goodrich

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. Included on this Web site is a collection of educational aids that augment the topics of this book, for both students and instructors. Because of their added value, some of these online resources are password protected.

For all readers, and especially for students, we include the following resources:

- All the Python source code presented in this book.
- PDF handouts of Powerpoint slides (four-per-page) provided to instructors.
- A database of hints to ***all*** exercises, indexed by problem number.

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

- Solutions to hundreds of the book's exercises.
- Color versions of all figures and illustrations from the book.
- Slides in Powerpoint and PDF (one-per-page) format.

The slides are fully editable, so as to allow an instructor using this book full freedom in customizing his or her presentations. All the online resources are provided at no extra charge to any instructor adopting this book for his or her course.

Contents and Organization

The chapters for this book are organized to provide a pedagogical path that starts with the basics of Python programming and object-oriented design. We then add foundational techniques like algorithm analysis and recursion. In the main portion of the book, we present 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. **Python Primer**
2. **Object-Oriented Programming**
3. **Algorithm Analysis**
4. **Recursion**
5. **Array-Based Sequences**
6. **Stacks, Queues, and Deques**
7. **Linked Lists**
8. **Trees**
9. **Priority Queues**
10. **Maps, Hash Tables, and Skip Lists**
11. **Search Trees**
12. **Sorting and Selection**
13. **Text Processing**
14. **Graph Algorithms**
15. **Memory Management and B-Trees**

A. **Character Strings in Python**
B. **Useful Mathematical Facts**

A more detailed table of contents follows this preface, beginning on page xi.

Prerequisites

We assume that the reader is at least vaguely familiar with a high-level programming language, such as C, C++, Python, or Java, and that he or she understands the main constructs from such a high-level language, including:

- Variables and expressions.
- Decision structures (such as if-statements and switch-statements).
- Iteration structures (for loops and while loops).
- Functions (whether stand-alone or object-oriented methods).

For readers who are familiar with these concepts, but not with how they are expressed in Python, we provide a primer on the Python language in Chapter 1. Still, this book is primarily a data structures book, not a Python book; hence, it does not give a comprehensive treatment of Python.

We delay treatment of object-oriented programming in Python until Chapter 2. This chapter is useful for those new to Python, and for those who may be familiar with Python, yet not with object-oriented programming.

In terms of mathematical background, we assume the reader is somewhat familiar with topics from high-school mathematics. Even so, in Chapter 3, 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 B.

Relation to Computer Science Curriculum

To assist instructors in designing a course in the context of the IEEE/ACM 2013 Computing Curriculum, the following table describes curricular knowledge units that are covered within this book.

Knowledge Unit	**Relevant Material**
AL/Basic Analysis	Chapter 3 and Sections 4.2 & 12.2.4
AL/Algorithmic Strategies	Sections 12.2.1, 13.2.1, 13.3, & 13.4.2
AL/Fundamental Data Structures and Algorithms	Sections 4.1.3, 5.5.2, 9.4.1, 9.3, 10.2, 11.1, 13.2, Chapter 12 & much of Chapter 14
AL/Advanced Data Structures	Sections 5.3, 10.4, 11.2 through 11.6, 12.3.1, 13.5, 14.5.1, & 15.3
AR/Memory System Organization and Architecture	Chapter 15
DS/Sets, Relations and Functions	Sections 10.5.1, 10.5.2, & 9.4
DS/Proof Techniques	Sections 3.4, 4.2, 5.3.2, 9.3.6, & 12.4.1
DS/Basics of Counting	Sections 2.4.2, 6.2.2, 12.2.4, 8.2.2 & Appendix B
DS/Graphs and Trees	Much of Chapters 8 and 14
DS/Discrete Probability	Sections 1.11.1, 10.2, 10.4.2, & 12.3.1
PL/Object-Oriented Programming	Much of the book, yet especially Chapter 2 and Sections 7.4, 9.5.1, 10.1.3, & 11.2.1
PL/Functional Programming	Section 1.10
SDF/Algorithms and Design	Sections 2.1, 3.3, & 12.2.1
SDF/Fundamental Programming Concepts	Chapters 1 & 4
SDF/Fundamental Data Structures	Chapters 6 & 7, Appendix A, and Sections 1.2.1, 5.2, 5.4, 9.1, & 10.1
SDF/Developmental Methods	Sections 1.7 & 2.2
SE/Software Design	Sections 2.1 & 2.1.3

Mapping *IEEE/ACM 2013 Computing Curriculum* knowledge units to coverage in this book.

About the Authors

Michael Goodrich received his Ph.D. in Computer Science from Purdue University in 1987. He is currently a Chancellor's Professor in the Department of Computer Science at University of California, Irvine. Previously, he was a professor at Johns Hopkins University. He is a Fulbright Scholar and a Fellow of the American Association for the Advancement of Science (AAAS), Association for Computing Machinery (ACM), and Institute of Electrical and Electronics Engineers (IEEE). He is a recipient of the IEEE Computer Society Technical Achievement Award, the ACM Recognition of Service Award, and the Pond Award for Excellence in Undergraduate Teaching.

Roberto Tamassia received his Ph.D. in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign in 1988. He is the Plastech Professor of Computer Science and the Chair of the Department of Computer Science at Brown University. He is also the Director of Brown's Center for Geometric Computing. His research interests include information security, cryptography, analysis, design, and implementation of algorithms, graph drawing and computational geometry. He is a Fellow of the American Association for the Advancement of Science (AAAS), Association for Computing Machinery (ACM) and Institute for Electrical and Electronic Engineers (IEEE). He is also a recipient of the Technical Achievement Award from the IEEE Computer Society.

Michael Goldwasser received his Ph.D. in Computer Science from Stanford University in 1997. He is currently a Professor in the Department of Mathematics and Computer Science at Saint Louis University and the Director of their Computer Science program. Previously, he was a faculty member in the Department of Computer Science at Loyola University Chicago. His research interests focus on the design and implementation of algorithms, having published work involving approximation algorithms, online computation, computational biology, and computational geometry. He is also active in the computer science education community.

Additional Books by These Authors

- M.T. Goodrich and R. Tamassia, *Data Structures and Algorithms in Java*, Wiley.
- M.T. Goodrich, R. Tamassia, and D.M. Mount, *Data Structures and Algorithms in C++*, Wiley.
- M.T. Goodrich and R. Tamassia, *Algorithm Design: Foundations, Analysis, and Internet Examples*, Wiley.
- M.T. Goodrich and R. Tamassia, *Introduction to Computer Security*, Addison-Wesley.
- M.H. Goldwasser and D. Letscher, *Object-Oriented Programming in Python*, Prentice Hall.

Acknowledgments

We have depended greatly upon the contributions of many individuals as part of the development of this book. We begin by acknowledging the wonderful team at Wiley. We are grateful to our editor, Beth Golub, for her enthusiastic support of this project, from beginning to end. The efforts of Elizabeth Mills and Katherine Willis were critical in keeping the project moving, from its early stages as an initial proposal, through the extensive peer review process. We greatly appreciate the attention to detail demonstrated by Julie Kennedy, the copyeditor for this book. Finally, many thanks are due to Joyce Poh for managing the final months of the production process.

We are truly indebted to the outside reviewers and readers for their copious comments, emails, and constructive criticism, which were extremely useful in writing this edition. We therefore thank the following reviewers for their comments and suggestions: Claude Anderson (Rose Hulman Institute of Technology), Alistair Campbell (Hamilton College), Barry Cohen (New Jersey Institute of Technology), Robert Franks (Central College), Andrew Harrington (Loyola University Chicago), Dave Musicant (Carleton College), and Victor Norman (Calvin College). We wish to particularly acknowledge Claude for going above and beyond the call of duty, providing us with an enumeration of 400 detailed corrections or suggestions.

We thank David Mount, of University of Maryland, for graciously sharing the wisdom gained from his experience with the C++ version of this text. We are grateful to Erin Chambers and David Letscher, of Saint Louis University, for their intangible contributions during many hallway conversations about the teaching of data structures, and to David for comments on early versions of the Python code base for this book. We thank David Zampino, a student at Loyola University Chicago, for his feedback while using a draft of this book during an independent study course, and to Andrew Harrington for supervising David's studies.

We also wish to reiterate our thanks to the many research collaborators and teaching assistants whose feedback shaped the previous Java and C++ versions of this material. The benefits of those contributions carry forward to this book.

Finally, we would like to warmly thank Susan Goldwasser, 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, and Calista and Maya Goldwasser for offering their advice regarding the artistic merits of many illustrations. More importantly, we thank all of these people for reminding us that there are things in life beyond writing books.

Michael T. Goodrich
Roberto Tamassia
Michael H. Goldwasser

Contents