
Undergraduate Topics in Computer Science

Undergraduate Topics in Computer Science (UTiCS) delivers high-quality instructional content for undergraduates studying in all areas of computing and information science. From core foundational and theoretical material to final-year topics and applications, UTiCS books take a fresh, concise, and modern approach and are ideal for self-study or for a one- or two-semester course. The texts are all authored by established experts in their fields, reviewed by an international advisory board, and contain numerous examples and problems. Many include fully worked solutions.

More information about this series at <http://www.springer.com/series/7592>

Kent D. Lee · Steve Hubbard

Data Structures and Algorithms with Python

Kent D. Lee
Steve Hubbard
Luther College
Decorah, IA
USA

Series editor
Ian Mackie

Advisory Board

Samson Abramsky, University of Oxford, Oxford, UK
Karin Breitman, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil
Chris Hankin, Imperial College London, London, UK
Dexter Kozen, Cornell University, Ithaca, USA
Andrew Pitts, University of Cambridge, Cambridge, UK
Hanne Riis Nielson, Technical University of Denmark, Kongens Lyngby, Denmark
Steven Skiena, Stony Brook University, Stony Brook, USA
Iain Stewart, University of Durham, Durham, UK

ISSN 1863-7310 ISSN 2197-1781 (electronic)
ISBN 978-3-319-13071-2 ISBN 978-3-319-13072-9 (eBook)
DOI 10.1007/978-3-319-13072-9

Library of Congress Control Number: 2014953918

Springer Cham Heidelberg New York Dordrecht London

© Springer International Publishing Switzerland 2015

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law. The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Thanks for choosing *Data Structures and Algorithms with Python*. This text was written based on classroom notes for two courses, an introductory data structures and algorithms course and an advanced data structures and algorithms course. The material contained in this text can be taught in two semesters. The early chapters in this text are intended as an introductory text for data structures and algorithms, while the later chapters cover advanced topics that are suitable for the second course in data structures and algorithms. The Python language is used throughout the text and some familiarity with Python or some other object-oriented language is assumed. However, the first chapter contains a Python primer for those coming from a different language background.

This text serves well as a follow-on text to *Python Programming Fundamentals* by Kent D. Lee and published by Springer, but does not require you to have read that text. In this text the next steps are taken to teach you how to handle large amounts of data efficiently. A number of algorithms are introduced and the need for them is motivated through examples that bring meaning to the problems we face as computer programmers. An algorithm is a well-defined procedure for accomplishing a task. Algorithms are an important part of Computer Science and this text explores many algorithms to give you the background you need when writing programs of your own. The goal is that having seen some of the sorts of algorithms presented in this text, you will be able to apply these techniques to other programs you write in the future.

Another goal of this text is to introduce you to the idea of computational complexity. While there are many unique and interesting algorithms that we could explore, it is important to understand that some algorithms are more efficient than others. While computers are very good at doing calculations quickly, an inefficient algorithm can make the fastest computer seem very slow or even make it appear to come to a halt. This text will show you what can and cannot be computed efficiently. The text builds this idea of efficiency from the most basic of facts giving you the tools you will need to determine just how efficient any algorithm is so you can make informed judgements about the programs you write.

The text assumes that you have some prior experience in computer programming, probably from an introductory programming course where you learned to break simple problems into steps that could be solved by a computer. The language you used may have been Python, but not necessarily. Python is an excellent language for a text on data structures and algorithms whether you have used it before or not. Python is an object-oriented programming language with operator overloading and dynamic typing. Whether this is your first exposure to Python or you used it in your first course, you will learn more about the language from this text. The first chapter of the text reviews some of the fundamentals of computer programming along with the basic syntax of Python to get you up to speed in the language. Then subsequent chapters dive into more advanced topics and should be read in sequence.

At the beginning of every chapter the goals of the chapter are stated. At the end of every chapter is a set of review questions that reinforce the goals of the chapter. These review questions are followed in each chapter by a few programming problems that relate to the chapter goals by asking you to use the things you learned in the chapter and apply them to a computer program. You can motivate your reading of a chapter by first consulting the review questions and then reading the chapter to answer them. Along the way, there are lots of examples to illustrate the concepts being introduced.

We hope you enjoy the text! If you have any questions or comments please send them to kentlee@luther.edu.

Kent D. Lee
Steve Hubbard

For Teachers

A typical introductory data structures course covers the first seven chapters of this text. Chapter 1 introduces Python programming and the Tkinter module which is used in various places in the text. Tkinter comes with Python, so no special libraries need be installed for students to use it. Tkinter is used to visualize many of the results in this text.

Chapter 2 introduces complexity analysis and depending on your needs, some of the material in Chap. 2 could be skipped in an introductory data structures course. In particular, the material on Θ notation and amortized complexity can be skipped. Big-Oh notation is enough for the first seven chapters. Typically, Chap. 7 is covered lightly and near the end of a semester course. It seems there is generally not enough time in a semester to cover graph theory in much detail.

Advanced courses in data structures and algorithms should start with Chap. 1 if students are unfamiliar with Python or Tkinter. A brief refresher may not be bad even for those that have programmed using Python before. Chapter 2 should be covered in detail including the material on Θ notation and amortized complexity.

Some review of hashing as it is used in sets and maps in Chap. 5 may be good review earlier in the advanced course along with a brief discussion of binary search trees and tree traversals in Chap. 6. Depending on your needs, Chap. 7 would be a good chapter to cover next including the material on depth first search of a graph.

Chapter 8 is where the advanced material begins with assumptions made that students understand the concepts presented in the earlier chapters. The two introductory chapters along with Chaps. 8–12 make a seven-chapter sequence that will fill a semester in an advanced course nicely.

This text is very project oriented. Solutions for all projects are available from Kent D. Lee. You can contact Kent at kentlee@luther.edu for instructor solutions. You must provide proof (through a website or other reference) that you are an instructor at an educational institution to get access to the instructor materials.

If you have any suggestions or find any errors in the text, please let us know by emailing Kent at kentlee@luther.edu. Thanks and we hope you enjoy using the text in your course!

Kent D. Lee
Steve Hubbard

Credits

Connect Four is referenced in Chaps. 4, 12 and Appendix H. Connect Four is a trademark of the Milton Bradley Company in the United States and other countries. Chapter 2 references Mac OS X. Mac and Mac OS are registered trademarks of Apple Inc., registered in the U.S. and other countries. Microsoft Windows is also referenced in Chap. 2. Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

Contents

1	Python Programming 101	1
1.1	Chapter Goals	3
1.2	Creating Objects	3
1.3	Calling Methods on Objects	5
1.4	Implementing a Class	6
1.5	Operator Overloading	8
1.6	Importing Modules	10
1.7	Indentation in Python Programs	11
1.8	The <i>Main</i> Function	12
1.9	Reading from a File	13
1.10	Reading Multi-line Records from a File	16
1.11	A Container Class	20
1.12	Polymorphism	20
1.13	The Accumulator Pattern	22
1.14	Implementing a GUI with Tkinter	24
1.15	XML Files	32
1.16	Reading XML Files	35
1.17	Chapter Summary	37
1.18	Review Questions	38
1.19	Programming Problems	39
2	Computational Complexity	41
2.1	Chapter Goals	41
2.2	Computer Architecture	42
2.3	Accessing Elements in a Python List	44
2.4	Big-Oh Notation	48
2.5	The PyList Append Operation	50
2.6	A Proof by Induction	51

2.7	Making the PyList Append Efficient	53
2.8	Commonly Occurring Computational Complexities	55
2.9	More Asymptotic Notation	56
2.10	Amortized Complexity	58
2.11	Chapter Summary	62
2.12	Review Questions	63
2.13	Programming Problems	64
3	Recursion	67
3.1	Chapter Goals	68
3.2	Scope	69
3.3	The Run-Time Stack and the Heap	72
3.4	Writing a Recursive Function	75
3.5	Tracing the Execution of a Recursive Function	78
3.6	Recursion in Computer Graphics	82
3.7	Recursion on Lists and Strings	83
3.8	Using Type Reflection	86
3.9	Chapter Summary	87
3.10	Review Questions	88
3.11	Programming Problems	88
4	Sequences	91
4.1	Chapter Goals	91
4.2	Lists	92
4.3	Cloning Objects	99
4.4	Item Ordering	100
4.5	Selection Sort	102
4.6	Merge Sort	105
4.7	Quicksort	109
4.8	Two-Dimensional Sequences	112
4.9	The Minimax Algorithm	116
4.10	Linked Lists	117
4.11	Stacks and Queues	123
4.12	Chapter Summary	135
4.13	Review Questions	135
4.14	Programming Problems	136
5	Sets and Maps	139
5.1	Chapter Goals	139
5.2	Playing Sudoku	140
5.3	Sets	142
5.4	Hashing	144
5.5	The HashSet Class	145
5.6	Solving Sudoku	151

5.7	Maps	153
5.8	Memoization	156
5.9	Correlating Two Sources of Information	158
5.10	Chapter Summary	159
5.11	Review Questions	159
5.12	Programming Problems	160
6	Trees	163
6.1	Chapter Goals.	163
6.2	Abstract Syntax Trees and Expressions	164
6.3	Prefix and Postfix Expressions	166
6.4	Parsing Prefix Expressions	167
6.5	Binary Search Trees	170
6.6	Search Spaces.	176
6.7	Chapter Summary	179
6.8	Review Questions	179
6.9	Programming Problems	180
7	Graphs.	185
7.1	Chapter Goals.	185
7.2	Graph Notation.	186
7.3	Searching a Graph.	188
7.4	Kruskal’s Algorithm	190
7.5	Dijkstra’s Algorithm	196
7.6	Graph Representations	199
7.7	Chapter Summary	201
7.8	Review Questions	202
7.9	Programming Problems	202
8	Membership Structures.	205
8.1	Chapter Goals.	205
8.2	Bloom Filters	206
8.3	The Trie Datatype	209
8.4	Chapter Summary	213
8.5	Review Questions	213
8.6	Programming Problems	214
9	Heaps.	215
9.1	Chapter Goals.	215
9.2	Key Ideas	215
9.3	Building a Heap	217
9.4	The Heapsort Algorithm Version 1	219
9.5	Analysis of Version 1 Phase I.	221

9.6	Phase II	225
9.7	Analysis of Phase II	228
9.8	The Heapsort Algorithm Version 2	229
9.9	Analysis of Heapsort Version 2	232
9.10	Comparison to Other Sorting Algorithms	233
9.11	Chapter Summary	234
9.12	Review Questions	235
9.13	Programming Problems	236
10	Balanced Binary Search Trees.	237
10.1	Chapter Goals.	237
10.2	Binary Search Trees	238
10.3	AVL Trees.	239
10.4	Splay Trees	250
10.5	Iterative Splaying	254
10.6	Recursive Splaying	256
10.7	Performance Analysis	257
10.8	Chapter Summary	258
10.9	Review Questions	259
10.10	Programming Problems	259
11	B-Trees	261
11.1	Chapter Goals.	261
11.2	Relational Databases	261
11.3	B-Tree Organization	270
11.4	The Advantages of B-Trees	272
11.5	B-Tree Implementation	274
11.6	B-Tree Insert	274
11.7	B-Tree Delete.	276
11.8	Chapter Summary	279
11.9	Review Questions	279
11.10	Programming Problems	280
12	Heuristic Search	281
12.1	Chapter Goals.	281
12.2	Depth First Search	282
12.3	Breadth First Search	285
12.4	Hill Climbing	286
12.5	Best First Search.	291
12.6	A* Search	292
12.7	Minimax Revisited	293
12.8	Chapter Summary	295
12.9	Review Questions	295
12.10	Programming Problems	296

13	Appendix A: Integer Operators	299
14	Appendix B: Float Operators	301
15	Appendix C: String Operators and Methods	303
16	Appendix D: List Operators and Methods	307
17	Appendix E: Dictionary Operators and Methods	309
18	Appendix F: Turtle Methods	311
19	Appendix G: TurtleScreen Methods	323
20	Appendix H: Complete Programs	331
20.1	The Draw Program	331
20.2	The Scope Program	338
20.3	The Sort Animation	339
20.4	The PlotData Program	346
20.5	The Tic Tac Toe Application	348
20.6	The Connect Four Front-End	353
	Bibliography	359
	Index	361