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

Springer

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

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

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 kentdlee@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 semeseter 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 kentdlee@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 kentdlee@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 Stated and other countries.

Contents