Textbook Proposal – Cambridge University Press

Dan S. Myers

Contact

Dan S. Myers
Rollins College—Dept. Of Mathematics and Computer Science
dmyers@rollins.edu
407-646-2151

Proposed Title

Data Structures and Algorithms: A Project-Based Approach

Course

This book is aimed at the standard undergraduate "CS2" course that presents the core content of data structures and algorithms (sorting, lists, trees, complexity analysis, etc.) to students who have already completed introductory programming. The CS2 course—usually titled "Data Structures and Algorithms"—forms the heart of the intermediate course work in the CS major and bridges the gap between the introductory CS1 course and the upper-level electives that focus on specific areas of computer science. The course covers a standard set of topics, with an emphasis on the implementation of fundamental data structures, an introduction to computational complexity analysis, and standard algorithms for searching, sorting, and other classic problems.

I have taught Data Structures and Algorithms at Rollins College more than half a dozen times since 2014. In developing my course, I faced the following challenges, which I think are relevant to instructors at a broad range of institutions:

- **Review of CS1 Material**. Many universities expect students to take CS2 as their second programming-focused course, typically after having one semester covering the fundamentals of programming. Therefore, the CS2 course typically needs to begin with a review of fundamental concepts and a deeper presentation of object-oriented programming before introducing new material. This problem has become more acute in recent years as departments (including ours) have shifted their introductory class to Python while keeping the data structures course in Java.

- **Developing Students' Project Portfolios**. Job-market demand for CS graduates remains high, but competition for the best internships and full-time jobs is intense. Students—particularly those at smaller or regional programs with lower name recognition—are under pressure to develop a portfolio of significant projects that they can showcase in job and internship applications. Because students now seek internships earlier, there is often a need to include significant portfolio-ready projects earlier in the curriculum. The CS2 course is an ideal place to incorporate creative projects for early CS majors that go beyond simply implementing standard data structures.

- **Technical Interview Prep**. Similar to the previous point, there is now increased pressure on students to prepare for technical interviews earlier in the curriculum. Answering "whiteboard" questions is a skill, and students who wait until their senior year to being practicing may find themselves at a disadvantage when applying to top-tier tech companies. It can be challenging, however, to find the right balance between interview prep questions and software development projects.

Over the next five years, we will see the current emphasis on student success and persistence from the CS1 course to the rest of the curriculum. In addition to a greater emphasis on active learning, we'll see a strong desire for "relevance," interdisciplinary applications, and social awareness throughout the CS major. These changes, which have already been part of CS1 pedagogy for the past decade, are likely to move into the CS2 course before filtering further into upper-level electives.

Book Description

Summary and Need. My proposed book, tentatively titled *Data Structures and Algorithms: A Project-Based Approach*, takes a novel, active-learning approach to the CS2 course. The text will present data structures and algorithms as an active and relevant discipline with connections to important applications in both software development and other fields.

The relevance of algorithms may seem obvious, but the reality is that many popular CS2 texts take a primarily theoretical and self-contained approach. Even good students often finish their CS2 course with no idea how the material applies to actual software development work. Further, professional software developers often associate the subject solely with preparing for whiteboard interviews

My book will offer instructors an engaging and carefully curated textbook that presents all of the fundamental material of the CS2 course while emphasizing practical applications to engaging interdisciplinary problems—including several resume-worthy projects—in CS, the arts and the sciences. For example,

- Creating a HashMap-based search engine for words and phrases in the plays of William Shakespeare [1].

- Using a circular buffer to implement the Karplus-Strong algorithm, a simple sound synthesis program that simulates a plucked guitar string [2].

- Implementing a recommendation system—based on the Pixie system developed at Pinterest—that uses random walks on a bipartite graph as its underlying model [3].

This approach is in line with recent developments in computer science pedagogy. For the past decade, educators have been developing novel approaches to both non-majors' and introductory programming classes that bridge the gaps between CS and other disciplines. The annual SIGCSE conference features a popular panel of "nifty assignments," many of which are aimed at the CS2 course. I believe there is an opportunity for a new book that brings together the best of this recent research into an accessible form.

Level. The primary audience is undergraduate students with at least one semester of programming experience and some exposure to object-oriented programming.

Features. The outstanding features of the book are as follows:

- Thorough coverage of fundamental data structures and algorithms topics, including lists, stacks, trees, graphs, sorting and complexity analysis.

- Motivating interdisciplinary examples that emphasize the role of data structures and algorithms in solving practical interdisciplinary problems. Connections to other computer science topics will be featured.

- Project-based chapters focusing on the use of each topic in significant software projects, of the kind that could be major assignments or featured in a student's resume and online portfolio. Many of these projects have been tested in my own classes at Rollins College.

- Examples of interesting application areas that have become relevant since 2010, including hashing in cryptocurrencies, tree search in DeepMind's AlphaGo, and procedural content generation in games.

- Early review of important programming topics and object-oriented programming for recent CS1 students.

- Coverage of standard technical interview questions related to each topic. Although not the primary focus of the book, this is an important topic for contemporary CS2 classes that is not covered in other popular texts.

- Active learning using reflective questions and solutions integrated throughout the text, informed by extensive undergraduate teaching experience.

Choice of Language. Java is currently the most popular language for CS2 courses, but many departments have recently transitioned their CS1 course to Python, which may lead to an increasing preference for that language in the second course. The book could be written in either language. I am currently planning to prepare the book in Java unless there is a strong reviewer interest in Python.

Supplementary Material. In addition to the finished text, I am committed to developing resources that will meet the needs of instructors and encourage them to adopt the book for their classes. Supplemental materials will include:

- Source code for the book's projects and larger examples, distributed through Cambridge's website, GitHub, or another location.

- Online solutions to end-of-chapter exercises for instructors who adopt the text.

- Lecture slides for each chapter.

- If reviewer feedback supports it, an online bank of test questions.

A major question that would need to be addressed early in the writing process is the use of supplemental supporting programs for features like images or sounds. Some CS1 texts—for example, Sedgwick and Wayne's *Computer Science: An Interdisciplinary Approach*—make extensive use of custom libraries to simplify their examples. Creating some pre-written code to provide simple 2-D graphics would simplify the projects in Chapter 3

(Conway's Life), Chapter 8 (recursive art), and Chapter 22 (mazes and procedural content generation). Chapter 13 (sound synthesis) would benefit from a small audio library with methods to play buffered sound samples, since audio programming can be complex.

On the other hand, implementing the complete projects without using pre-written custom libraries would allow us to examine the Java Graphics and Audio APIs and use them to illustrate object-oriented programming concepts. That approach would require making Chapters 8, 13, and 22 a little longer to explain the relevant Java features but would make the book self-contained. At this time, I lean towards incorporating the Graphics and Audio APIs directly into the text, without relying on additional external classes, but would defer to the guidance of reviewers.

Professional Readership. Working software developers are an important secondary audience for the book, particularly those who entered the field through bootcamps or other non-traditional paths. Currently, working developers who want to learn data structures and algorithms typically turn to upper-level theoretical texts like Cormen et al.'s *Introduction to Algorithms*, which assume the reader already has a strong background in CS fundamentals. Other texts aimed at this market, like *Grokking Algorithms* by Aditya Bhargava (Manning Publications, 2016), lack comprehensive coverage.

Proposed Chapter Outline

The following outline presents a general overview of the structure and content of the book. I have chosen, in general, to separate "theory" topics and their related projects into separate chapters, but they could be combined to reduce the total number of chapters if necessary. In the current draft, all theoretical chapters end with a section of example interview questions and a section of exercises. In addition to modifying the specific examples and projects, additional chapters on advanced topics—for example, string algorithms, dynamic programming, or complexity theory—could be added if reviewers feel they are valuable.

1. Algorithms + Data Structures = Programs
    - 1.1 Three good reasons to study this book (and two bad ones)
    - 1.2 A preview of the upcoming topics
    - 1.3 Why do data structures and algorithms matter to real-world programmers?
    - 1.3 A brief history of algorithms
    - 1.4 How to use this book if you're teaching yourself

Comparable Books

There are a number of texts aimed at the CS2 course available on the market, although few that I see as directly comparable to the proposed book.

***Data Structures and Algorithms in Java 6e* (Wiley, 2014) by Michael Goodrich, Roberto Tamassia, and Michael Goldwasser (GTG).** A strength of this book is its emphasis on combining the theory of data structures with their implementations in the Java standard library. The list of topics includes two full chapters reviewing Java and object-oriented programming (useful for bridging the gap from CS1) and a dedicated chapter on iterators. Despite its strengths, I see three areas in which my proposed book can distinguish itself from the GTG text:

- GTG takes a breadth-oriented approach, covering more material than can realistically fit into a one-semester course. Overall, there is a fair amount of coverage of niche data structures and algorithm variations (e.g. (2, 4) trees, skip lists, splay trees) that are not typically covered in a standard CS2 course. I would prefer to narrow the breadth of the book and cover fewer minor variations in favor of more in-depth discussion of applications.

- The presentation is clear but verbose. Based on my experience with ZyBooks (discussed below), I have seen students respond well to a text that combines concise sections with integrated active learning questions. GTG, like the other books in this section, features a combination of discussion, examples, and end-of-chapter exercises, but does not incorporate an active, collaborative approach to teaching throughout the text.

- Many of GTG's examples are artificial and abstract, particularly in the early sections on Java and object-oriented programming. As discussed previously, I think examples drawn from real-world applications are both more engaging and illuminating for students.

***Data Structures and Algorithm Analysis in Java 3e* by Mark Weiss (Pearson, 2012)**. This book sits somewhere between a text for the undergraduate CS2 class and an upper-level elective on algorithm design and advanced data structures. Most of the coverage of core data structures is compressed—for example, lists, stacks and queues are covered in a single chapter and the chapter on trees discusses binary search trees, AVL trees, splay trees, and B-trees. As with the GTG book, a fair amount of the text is given to niche data structures and variations on the core concepts. However, a strength of the book is its thorough coverage of sorting and hashing techniques, including theoretically interesting variations like cuckoo hashing. This book could be a good choice for a curriculum with a late data structures and algorithms class taken by students that have already had multiple courses in programming and relevant math topics. The overall theoretical emphasis and concise presentation, though, isn't a good fit for students taking their data structures course shortly following CS1.

***Data Structures and Abstractions with Java* by Frank Carrano and Timothy Henry (Pearson, 5th ed., 2018).** This book is similar in style to the GTG text. Carrano and Henry's text covers the least material of any of the books discussed in this section, with a focus on only the standard CS2 topics (including balanced search trees and graphs). It does include sections covering generics, inheritance, and polymorphism in Java, which are interspersed throughout the book rather than being collected into one chapter.

My comments on this text are similar to the GTG volume, with similar points of distinction relative to my proposed book. It covers the basics in a good style with some relevant information about Java programming to help bridge the gap from CS1. As discussed earlier, I think there is an opening for a book with greater contemporary relevance and a bigger vision for how the data structures and algorithms course fits into the CS curriculum, which neither book really provides.

ZyBooks' *Data Structures and Algorithms*. I have used this text in my own courses. All of ZyBooks' offerings include autograded questions that are integrated with the text—"participation" questions that assess reading comprehension and "challenge" questions that can be used as full homework assignments. I use the ZyBook with participation questions to reinforce the topics I discuss in class and to provide coverage of small language features or special cases that are useful but too minor for class discussions. The autograded questions allow me to easily verify that the students have actually completed the assigned reading (at least to the extent that can be measured by the questions).

I would describe the style of the ZyBook as “serviceable but uninspiring.” Students do like the fact that sections are short and that answers are never far from their associated questions. Students’ engagement with the book is utilitarian—the goal of the text is to provide answers to the questions in each section. As an instructor, I’ve chosen the ZyBook as a practical solution to the problem of getting students to do out-of-class reading, but I don’t base my topic coverage around it or refer to it in class. The presentations are bare-bones and suitable for review, but not for actually learning new content.

One major advantage of the ZyBook is the price: it’s less than half the cost of most traditional textbooks, although structured as a rental for one semester rather than the purchase of a new copy.

Other texts. There are a few other texts that I have considered while preparing this proposal:

- *Algorithms* by Robert Sedgwick and Kevin Wayne (Addison-Wesley Professional, 4th ed., 2011) is comprehensive (800+ pages) and includes sections on string matching, advanced tree structures, and other topics. It discusses applications and context for the topics it presents, but the overall presentation style is theoretical with a focus on data structure implementation and mathematical analysis.

- Eric Roberts’ *Programming Abstractions in Java: A Client-First Approach* (Pearson, 2017) is an important comparison text, though not one of the most popular in the current market. It takes the interesting approach of presenting how to use the Java standard library implementation of each data structure before delving into the internal details of those structures. Although not a “project-based” book per se, it does include larger examples that overlap with my own choices, notably the application of backtracking search to games and a simple text editor (I developed my example projects before reviewing Roberts’ book as a point of comparison). The book does include a larger amount of material on Java programming and additional theoretical topics like induction and lambda expressions.

- *Grokking Algorithms* by Aditya Bhargava (Manning Publications, 2016) has achieved success as an accessible introduction to algorithms for working programmers but isn't comprehensive enough to be assigned as a textbook for a university course. Its main feature is its jargon-free style and emphasis on where different algorithms might be used in software development, but it does not include extensive examples.

Finally, there are multiple successful texts aimed at introductory programming courses that have already adopted interdisciplinary project-based approaches, which suggests that a conceptually similar book aimed at CS2 may find a ready audience. Two books of note are *Java Illuminated: An Active Learning Approach* by my former Rollins colleague Julie Anderson and Hervé Franceschi (Jones and Bartlett, 5th ed., 2019) and Sedgwick and Wayne's *Computer Science: An Interdisciplinary Approach* (Addison-Wesley Professional, 1st ed., 2016), the introductory counterpart to their *Algorithms* book. Both books teach programming through an active learning framework. The Sedgwick and Wayne text makes heavy use of supplemental programs for working with graphics, images, and sound.

Author Information

I am an Associate Professor and Chair of the Computer Science program at Rollins College in Winter Park, FL. I have been at Rollins since 2014, and teach classes across the computer science curriculum, including data structures, simulation, and analytics. My work on CS education topics has appeared at the ACM SIGCSE conference and at the Consortium of Computing Science in Colleges and I have presented and led discussions on the role of community engagement in CS education at the ACM Richard Tapia Celebration of Diversity in Computing and Ashoka U.

In addition to teaching, I work with our undergraduate students in a research group that performs impact assessment and data analytics projects with Central Florida nonprofits. My Rollins colleagues honored me with a 2019 distinguished faculty fellowship for excellence in teaching, scholarship, and community engagement. Before coming to Rollins, I received my PhD in Computer Sciences from the University of Wisconsin-Madison in 2014. Prior to attending UW, I served as a Senior Member of Technical Staff at Sandia National Laboratories in Albuquerque, NM, working on computer vision algorithms for national security problems.

Additional Information and Specifications

I aspire to bring the final text in at 600 pages, although that is only an estimate and would be influenced heavily by the number of full-length example programs (as opposed to excerpts) included in the text. If necessary, some of the larger project-based chapters could be converted to shorter application discussions (or even cut) to reduce the length or free up space for other topics.

My approximate schedule for the delivery of the finished manuscript is as follows.

- Completion of an initial chapter: August 2021
- Completion of one-half of chapters: December 2021
- Completion of three-quarters of chapters: May 2022
- Completion of all chapters: Late summer or early fall 2022
- Finalized manuscript: December 2022

The majority of the work would be completed in the summer and fall of 2021 during my scheduled sabbatical. I have developed preliminary material for half of the projects listed in the chapter summary to use as part of my own classes.

References

1. D. Myers. “A Compleat Shakespearean Search Engine”. Nifty Assignment, Consortium for Computing Sciences Southeastern Regional Conference (CCSC: SE) 2018. Roanoke College, Salem, VA, 11/2018.

2. K. Wayne. “Guitar Heroine”. Nifty Assignment, *SIGCSE ’12: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*. Raleigh, NC. 2/2012.

3. C. Eksombatchai, P. Jindal, J. Z. Liu, Y. Liu, R. Sharma, C. Sugnet, M. Ulrich, and J. Leskovec. "Pixie: A system for recommending 3+ billion items to 200+ million users in real-time." In *Proceedings of the 2018 world wide web conference*, pp. 1775-1784. 2018.