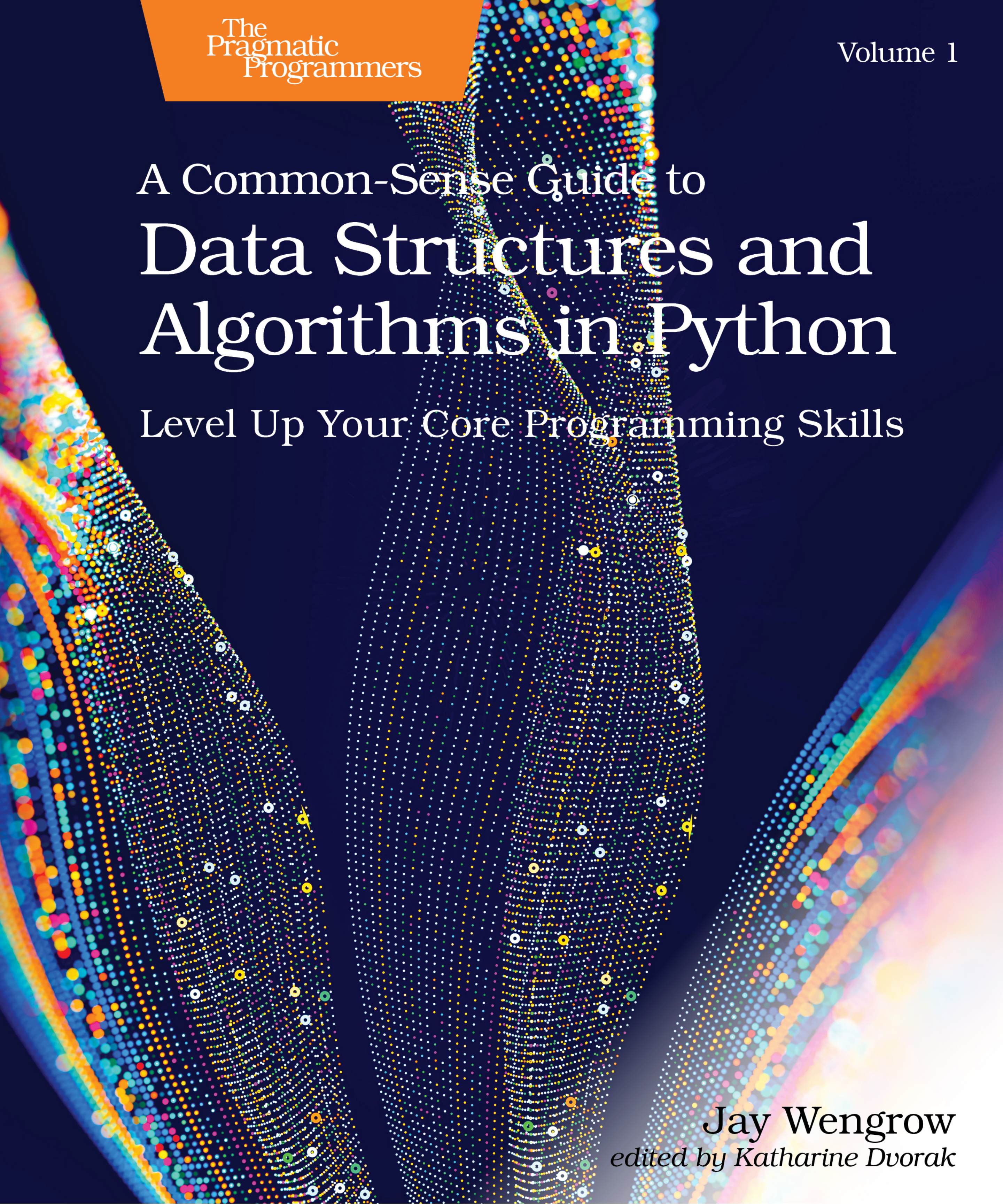
Extracted from:

A Common-Sense Guide to Data Structures and Algorithms in Python, Volume 1

Level Up Your Core Programming Skills

The Pragmatic Bookshelf
Dallas, Texas



The Pragmatic Programmers

Volume 1

A Common-Sense Guide to Data Structures and Algorithms in Python

Level Up Your Core Programming Skills

Jay Wengrow

edited by Katharine Dvorak

A Common-Sense Guide to Data Structures and Algorithms in Python, Volume 1

Level Up Your Core Programming Skills

Jay Wengrow

The Pragmatic Bookshelf

Dallas, Texas

The team that produced this book includes:

Publisher: Dave Thomas
COO: Janet Furlow
Managing Editor: Tammy Coron
Development Editor: Katharine Dvorak
Copy Editor: L. Sakhi MacMillan
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 979-8-88865-035-6
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—December 2023

Preface

Data structures and algorithms are more than abstract concepts. Mastering them enables you to write code that is *efficient*, leading to software that runs faster and consumes less memory. This is a big deal for today's software applications, which exist on increasingly mobile platforms and handle increasingly greater amounts of data.

The problem with most resources on these subjects, though, is that they are…well…obtuse. Most texts go heavy on the math jargon, and if you're not a mathematician, it can be difficult to grasp what on earth is going on. Even books that claim to make algorithms "easy" seem to assume that the reader has an advanced math degree. Because of this, too many people shy away from these concepts feeling that they're not "smart" enough to understand them.

The truth, though, is that everything about data structures and algorithms boils down to common sense. Mathematical notation itself is simply a particular language, and everything in math can also be explained with common-sense terminology. In this book, I use that common-sense language (plus a lot of diagrams!) to explain these concepts in simple and, dare I say, enjoyable ways.

Once you understand these concepts, you'll be equipped to write code that is efficient, fast, and elegant. You'll be able to weigh the pros and cons of various code alternatives and be able to make educated decisions as to which code is best for the given situation.

In this book, I go out of my way to make these concepts real and practical with ideas that you can make use of *today*. Sure, you'll learn some really cool computer science along the way. But this book is about taking that seemingly abstract stuff and making it directly practical. You'll be writing better code and faster software by the time you're done reading this book.

Who Is This Book For?

This book is ideal for several audiences:

- You're a computer science student who wants a text that explains data structures and algorithms in plain English. This book can serve as a supplement to whatever "classic" textbook you happen to be using.
- You're a beginning developer who knows basic programming but wants to learn the fundamentals of computer science to write better code and increase your programming knowledge and skills.
- You're a self-taught developer who has never studied formal computer science (or a developer who did but forgot everything!) and wants to leverage the power of data structures and algorithms to write more scalable and elegant code.

Whoever you may be, I tried to write this book so that it can be accessed and enjoyed by people of all skill levels.

The Python Edition

Pythonistas, rejoice! All the code in this book is written in Python. The original edition of this book, *A Common-Sense Guide to Data Structures and Algorithms* (the second edition of which was published in 2020), was language-agnostic; it was written using multiple programming languages. The idea behind this was that we didn't want to imply that the book would only be helpful for one language, given that data structures and algorithms are concepts that apply across all of computing in general.

Over the years, I received feedback from readers who expressed that they would love to see language-specific editions of the book. After all, if you're working with one particular programming language, wouldn't it be nice to have a book in which all of the code is in that language?

Accordingly, I set out to create new single-language versions of *A Common-Sense Guide to Data Structures and Algorithms*. The first of these is the Python version and is what you're reading now! And because there's so much to say about data structures and algorithms, this is the first volume in a progression of books. In this volume, I lay the foundation for these concepts and cover the most common data structures and algorithms. In the next volume, I'll build upon this knowledge to level up further with more advanced ideas and techniques.

A Note About the Code

I strived to follow PEP 8 standards (for the most part) and write the code in such a way so that it runs equivalently in both Python Version 2 and Python Version 3. You can now enjoy reading the code samples in your language of choice.

That being said, I still want to emphasize that the concepts in this book apply to virtually all coding languages, and I expect that some people not as familiar with Python will be reading this book. Because of this, sometimes I have avoided certain Python idioms where I thought that this would utterly confuse people coming from other languages. It's a tricky balance to keep the code Pythonic while also being welcoming to non-Python coders, but I hope that I've maintained an equilibrium that satisfies most readers.

Virtually all the code in this book can be downloaded online from the book's web page.[1] This code repository contains automated tests too! I encourage you to check that out since the tests don't appear in the actual book.

You'll find some longer code snippets under the headings that read "Code Implementation." I certainly encourage you to study these code samples, but you don't necessarily have to understand every last line to proceed to the next section of the book. If these long pieces of code are bogging you down, just skim (or skip) them for now.

Finally, it's important to note that the code in this book is not "production ready." My greatest focus has been to clarify the concept at hand, and while I did also try to make the code generally complete, I have not accounted for every edge case. There's certainly room for you to optimize the code further—so feel free to go crazy with that.

What's in This Book?

As you may have guessed, this book talks quite a bit about data structures and algorithms. More specifically, the book is laid out as follows:

In *Why Data Structures Matter* and *Why Algorithms Matter*, I explain what data structures and algorithms are and explore the concept of time complexity—which is used to determine how fast an algorithm is. In the process, I also talk a great deal about arrays, sets, and binary search.

1. https://pragprog.com/titles/jwpython

In *O Yes! Big O Notation*, I unveil Big O notation and explain it in terms that are easy to understand. We use this notation throughout the book, so this chapter is pretty important.

In *Speeding Up Your Code with Big O*, *Optimizing Code With and Without Big O*, and *Optimizing for Optimistic Scenarios*, we delve further into Big O notation and use it to make our day-to-day code faster. Along the way, I cover various sorting algorithms, including Bubble Sort, Selection Sort, and Insertion Sort.

In *Big O in Everyday Code*, you apply all that you learned about Big O notation and analyze the efficiency of code from the real world.

In *Blazing Fast Lookup with Hash Tables* and *Crafting Elegant Code*, I discuss a few additional data structures, including hash tables, stacks, and queues. I show how they impact the speed and elegance of our code and how we can use them to solve real-world problems.

Recursively Recurse with Recursion introduces recursion, an anchor concept in the world of computer science. We break it down in this chapter and see how it can be a great tool for certain situations. *Learning to Write in Recursive* teaches you how to write recursive code, which can be confusing if you're not familiar with it.

Dynamic Programming shows you how to optimize recursive code and prevent it from spiraling out of control. And *Recursive Algorithms for Speed* shows you how to use recursion as the foundation for turbo-fast algorithms like Quicksort and Quickselect, and then it takes your algorithm-development skills up a few notches.

The following chapters, *Node-Based Data Structures*, *Speeding Up All the Things*, *Keeping Your Priorities Straight with Heaps*, *It Doesn't Hurt to Trie*, and *Connecting Everything with Graphs*, explore node-based data structures including the linked list, the binary tree, the heap, the trie, and the graph and show how each is ideal for various applications.

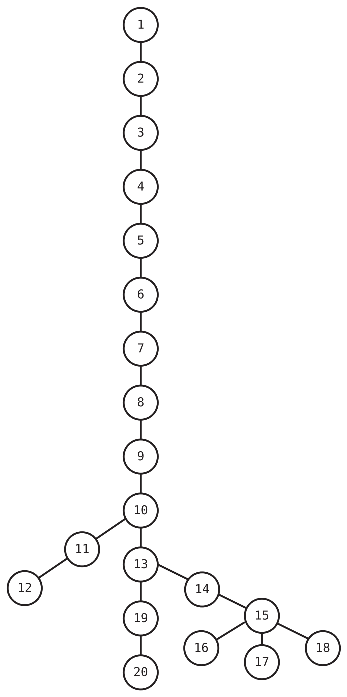
Dealing with Space Constraints explores space complexity, which is important when programming for devices with relatively small amounts of disk space or when dealing with big data.

The final chapter, *Techniques for Code Optimization*, walks you through various practical techniques for optimizing the efficiency of code and gives you new ideas for improving the code you write every day.

How to Read This Book

You've got to read this book in order. In some books out there you can read each chapter independently and skip around a bit, but *this isn't one of them.* Each chapter assumes you've read the previous ones, and the book is carefully constructed so you can ramp up your understanding as you proceed.

That being said, there are certain chapters in the latter half of the book that don't depend entirely on each other. The diagram below depicts which chapters are prerequisites for other chapters.



For example, you could technically skip from Chapter 10 to Chapter 13 if you wanted to. (Oh! And this diagram is based on a data structure called a *tree.* You're going to learn about it in Chapter 15.)

Another important note: to make this book easy to understand, I don't always reveal everything about a particular concept when I first introduce it. Sometimes, the best way to break down a complex concept is to reveal a small piece of it and only reveal the next piece once the first piece has sunk in. If I define a particular term as such-and-such, don't take that to be the textbook definition until you've completed the entire section on that topic.

It's a trade-off: to make the book digestible, I've chosen to oversimplify certain concepts at first and clarify them over time rather than ensure that every sentence is completely, academically, accurate. But don't worry too much, because by the end, you'll see the entire accurate picture.

Online Resources

This book has its own web page[2] on pragprog.com where you can find more information about the book, download the source code for the code examples, and help improve the book by reporting errata, typos, and content suggestions.

Additionally, I post updates about my writing at my website.[3] There you can find more information about my books as well as video tutorials made by my colleagues and me in which we use the "common-sense" approach to explaining all sorts of technologies and concepts. We can also train your employees! We cover all sorts of topics regarding code efficiency and leveling up their software development skills, including:

- Writing maintainable code
- Refactoring
- Unit testing
- And of course, writing efficient code!

My colleagues and I teach a variety of technologies, and we always develop curriculum using the "common-sense" way of explaining things. You can find more information at the aforementioned website.[4]

Connecting

I enjoy connecting with my readers and invite you to find me on LinkedIn.[5] I'd gladly accept your connection request—just send a message that you're a reader of this book. I look forward to hearing from you!

Jay Wengrow
December 2023

2. https://pragprog.com/titles/jwpython
3. https://commonsensedev.com
4. https://commonsensedev.com
5. https://www.linkedin.com/in/jaywengrow