Codeless Data Structures and Algorithms

Learn DSA Without Writing a Single Line of Code

Armstrong Subero

Apress®

CODELESS DATA STRUCTURES AND ALGORITHMS

LEARN DSA WITHOUT WRITING A SINGLE LINE OF CODE

Armstrong Subero

Apress®

Codeless Data Structures and Algorithms: Learn DSA Without Writing a Single Line of Code

Armstrong Subero
Basse Terre, Moruga, Trinidad and Tobago

ISBN-13 (pbk): 978-1-4842-5724-1 ISBN-13 (electronic): 978-1-4842-5725-8
https://doi.org/10.1007/978-1-4842-5725-8

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Shiva Ramachandran
Development Editor: Rita Fernando
Coordinating Editor: Rita Fernando

Cover designed by eStudioCalamar

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, New York, NY 100043. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at www.apress.com/bulk-sales.

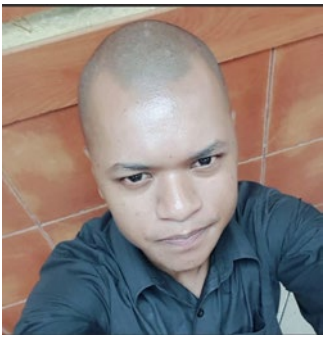
Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484257241. For more detailed information, please visit www.apress.com/source-code.

Printed on acid-free paper

To all those who tinker to keep the world running.

Contents

About the Author



Armstrong Subero started learning electronics at the age of 8. From then on he got into programming and embedded systems development. Within the realm of programming, he has a special interest in algorithms and data structures and enjoys implementing them in different programming languages and on different processor architectures, particularly resource-constrained systems. He currently works at the Ministry of National Security in his country, and he has degrees in computer science and liberal arts and sciences from Thomas Edison State University. He is the author of *Programming PIC Microcontrollers with XC8* (Apress, 2018).

Acknowledgments

I want to thank my family.

I want to thank *everyone* who ever said anything positive to me or taught me something. I heard it all, and it meant something.

I want to thank God most of all, because without God I wouldn't be able to do any of this.

Introduction

There is one question beginners and even good self-taught developers ask me all the time: "How do I learn data structures and algorithms?" I have had to answer this question so often that I thought I would write it down in a book so that anyone interested in this topic could follow along and understand the process.

The thing is, anyone can learn the basics of data structures and algorithms. There really isn't much to them. The hardest part of teaching someone this topic is that they are likely to be using any language from the current "zoo" of languages. They may be using Python, Java, C++, or even C or Rust. Whatever the language they are using, it is important to be able to understand data structures and algorithms at the fundamental level. As such, I decided to make my book "codeless." This is a book on algorithms and data structures that doesn't use a single line of code from any programming language.

As someone who must regularly change between different programing languages, take it from me, once you understand the concepts behind these algorithms in a codeless way, then you will be able to apply them to whatever language you are using.

There are also some people "devout" in their programming language who are unwilling to look at any material that isn't in their language of choice. As such, I have written this book without offending anyone's "beliefs." I think there are enough books on "data structures and algorithms in X language" that are thousands of pages detailing programs and all their nuances for the language you are using. I think such books would be fine complements to this one, as this book will give you the "big picture," and then you can look at whatever book you want for the rest. You can learn about a concept here and then go to that book to learn the programming details.

Everything is explained in plain English, and each chapter is short and to the point. You will learn a lot without burning your brain out.

Who Is This Book For?

This book is for people who want to understand data structures and algorithms but don't want unnecessary details about quirks of a language and don't have time to sit and read a massive tome on the subject. This book is

for people who want to understand the concepts of algorithms and data structures in plain English. I assume, though, that you have knowledge of at least one programming language, preferably a C-based one such as C, C++, Java, C#, or Python. The types and terminology used in this book are biased toward people who have used one of these languages.

I assume you are good at thinking abstractly and at least have basic knowledge of a programming language and of the basics of computer science (what a bit is, what a byte is, etc.). I also assume you know basic mathematics, at least algebra, though this book is by no means a "math-heavy" book. I use math concepts only where they are necessary and explain them clearly.

What Will I Need for This Book?

You won't need anything except an open mind and time to read and understand the concepts being presented. I wrote this book as if we were having a coffee and I was explaining these concepts to you from the top of my head. For that reason, you won't need any compiler or text editor, just a listening ear.

What Will I Learn in This Book?

You will learn quite a lot in a short space of time about data structures and algorithms. You will learn the concepts of data structures and algorithms with a focus on the most relevant ones in simple, plain language. After completing this book, you'll understand which data structures and algorithms can be used to solve which problems and will be able to confidently follow along with discussions involving algorithms and data structures.

PART I – Data Structures

Chapter 1 goes into what algorithms and data structures are, and we discuss primitive types and Big O notation.

Chapter 2 looks at the linear data structures of arrays and linked lists; we also discuss stacks and queues.

Chapter 3 delves into trees and tree-based data structures.

Chapter 4 introduces you to hash-type data structures.

Chapter 5 briefly covers the basics of graphs.

PART II – Algorithms

Chapter 6 introduces two common algorithms, that of linear search and binary search.

Chapter 7 explores sorting algorithms, including bubble sort, selection sort, insertion sort, merge sort, and quick sort.

Chapter 8 presents some search algorithms; we look at breath-first search, Dijkstra's algorithm, and the A* algorithm.

Chapter 9 introduces clustering algorithms, specifically the K-means algorithm and K-nearest neighbor and get a taste of machine learning and neural networks.

Chapter 10 teaches some basics about the concept of randomness.

Appendix A provides resources for going further.

Upon finishing this book, you will have a solid foundation in algorithms and data structures and will be able to use this knowledge when writing and designing your own programs.

Errata and Suggestions

If you find any errors in this text, have suggestions, or want to ask a question about your own project, you can contact me at `armstrongsubero@gmail.com`; please, no spam.

PART

I

Data Structures

CHAPTER

1

Intro to DSA, Types, and Big O

Every journey has a beginning. In this chapter, we will begin our journey by talking about what data structures and algorithms are. I will introduce basic types, and I will show you how easy Big O notation is to understand. If you have ever read a dull, overly complex book on data structures and algorithms (abbreviated to DSA), you will love how quickly this chapter teaches you things!

An Overview of Data Structures and Algorithms

> *"Talk is cheap. Show me the code."*
>
> —Linus Torvalds, Finnish software engineer and creator of Linux

Linus said this while replying on the Linux Kernel mailing list on August 25, 2000. This reply has become a famous quote among programmers. It is used by developers whenever they don't want to read anything and instead just jump into coding. This approach is particularly taken by novices and poorly self-taught programmers. They don't plan, they think they know more than everyone, and they think programming is all about the code. This couldn't be further from the truth. Code simply expresses your thoughts to solve a problem.

A. Subero, *Codeless Data Structures and Algorithms*,
https://doi.org/10.1007/978-1-4842-5725-8_1

Nothing more. Therefore, the more you know, the more you can apply to solve a problem.

Data structures and algorithms are simply more things to know to apply to solve your problems. Despite some people using them interchangeably, data structures and algorithms are actually very different things. It is possible to learn data structures and algorithms without touching a programming language. Programming essentially consists of thinking algorithmically and knowing the syntax of the programming language to solve problems. In this book, we will focus on thinking algorithmically and avoid learning the syntax of any programming language.

Before we discuss data structures and algorithms, I think we should talk a little about data. Data can mean different things depending on the discipline you are currently occupied with. However, *data* within the context of this book refers to any information that is stored on your machine or that is being handled or processed by it. Data should not be confused with *information*, which is data that has been processed; however, within the context of computing, many developers may use these terms independently to mean the same thing.

Data Structures

A *data structure* is a concept we use to describe ways to organize and store types of data. Data structures are important because they not only provide a way of storing and organizing data but also provide a way of identifying data within the structure; additionally, they show the relationships of the data within the structure. It is best to illustrate what a data structure is with an example.

For example, let's say we have some footwear, as depicted in Figure 1-1. We have two boots and two shoes arranged alternately.

Figure 1-1. Two types of footwear

We can think of each side of the shoe as a unit of data. If we needed a way to maintain this data arrangement, we would need a mechanism to provide some ordering and identification of these shoes; this is what we may call a *data structure*. A data structure may provide some mechanism to organize and store this data, for example, by separating boots and shoes as shown in Figure 1-2.

Figure 1-2. Separating boots and shoes

A data structure may also be able to take each item and, regardless of whether it's a shoe or boot, assign an identifier, for example, a number as shown in Figure 1-3.

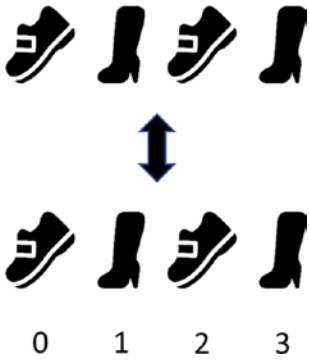


Figure 1-3. Assigning identifiers

So, if I wanted the "second shoe," instead of wondering if this means from the left or from the right, I can simply tell the data structure, "Give me the shoe at index 2," and I will get exactly what I wanted.

As basic as it may seem, this is all a data structure does, though there are much more complex methods of storing, identifying, and showing the relationships within data. This explanation provides a good grasp of what a data structure does. If you still aren't clear what a data structure is, it is best to think of a data structure as a container for data.

Algorithms

An *algorithm* is a method of solving a problem by applying a sequence of steps that will always work to solve the type of problem it was designed to solve. Another way of saying this is that an algorithm is simply a method of solving a

problem in an ordered manner. We can even shorten it further to say an algorithm is a procedure. Many people may have expanded or diminished perspectives to describe what an algorithm is, but this definition will work for our understanding.

One thing everyone will agree on is that algorithms are logical steps to accomplish a task. To accomplish this, an algorithm must be simple, precise, and unambiguous. Though some programmers focus on using esoteric features of programming languages that make an algorithm hard to read for other programmers, the simpler and more intuitive an algorithm is, the more powerful and useful it will be.

We can describe algorithms with natural languages such as English, pseudocode, or a programming language. We can discuss algorithms at great length; however, it is best to see how algorithms work with a good example. In our example, we can show how algorithms operate with a pure English description, with no code needed.

We can gain an understanding of algorithms by looking at one in action. Let's say you need to put some fruit on a plate. You would, of course, grab a plate and put some fruit on it, right? Well, imagine you were describing these same steps to a robot. How would you do it? You would tell the robot to do something like the following:

1. Go to the cupboard.
2. Open the door.
3. Take out a plate.
4. Go to the fruit bowl.
5. Put the fruit on the plate.

This seems like a logical sequence of steps, until you look around the kitchen and realize that the robot left the cupboard door open. So, you decide to add another step to the algorithm, as shown here:

1. Go to the cupboard.
2. Open the door.
3. Take out a plate.
4. Close the cupboard door.
5. Go to the fruit bowl.
6. Put the fruit on the plate.