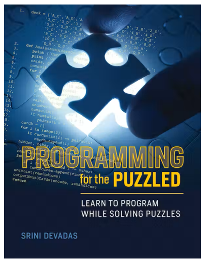
Review of[1]

Programming for the Puzzled:
Learn to Program While Solving Puzzles
Srini Devadas
The MIT Press, 2017
272 pages, Paperback, $25.00



Review by
S. V. Nagaraj (svnagaraj@acm.org)
Vellore Institute of Technology
Chennai Campus, India

1 Introduction

This book is about learning to program while solving puzzles. The motivation for the author to use this approach is to gain better attention from students, many of whom do not like to program just for the sake of programming. As mentioned in the informative preface, this book reflects the author's "attempt at teaching programming by building a bridge between the recreational world of algorithmic puzzles and the pragmatic world of computer programming," because the same analytical skills required for puzzle solving are also needed for "translating specifications into programming constructs, as well as discovering errors in early versions of code, called the debugging process."

Twenty one often-familiar puzzles are discussed in the book. Each chapter begins with a puzzle's description followed by a solution given in the form of specification of the code that needs to be written. Programming constructs and algorithmic paradigms needed to realize the code are made comprehensible. It is assumed that the readers have only a high school level understanding of programming concepts. Programming exercises are provided at the end of each chapter. These exercises vary in difficulty and the amount of code that needs to be written. Some of the puzzles are contrived to make the readers learn some algorithmic idea, while others, called Puzzle Exercises, involve more effort, and some of them may be considered as advanced puzzles. The code for puzzle solutions is downloadable from the book's website [2]. Ancillary material and the code for all puzzle solutions are available to instructors. Supplementary materials for the first eleven puzzles are available on YouTube [3] and as open courseware [4].

2 Summary

The first chapter is seemingly about how to give instructions to groups of people so that they wear hats the right way. However, it is actually about run-length encoding and data compression.

The second chapter is about finding the right hour to attend a party so as to maximize the number of celebrities with whom we can interact.

[1]©2023 S. V. Nagaraj

The third chapter is about encoding and communicating information, but cloaked up in the form of a card trick.

The fourth chapter is about the eight queens problem, which is a classic computer science problem.

The fifth chapter is about breaking a crystal ball. You have to drop the crystal ball from a floor in a high rise and determine the maximum floor in it from which it will not break. A ball can't be reused once it has been broken.

The sixth chapter is about detecting a fake coin in a set of coins that look identical.

The seventh chapter is about finding square roots.

The eighth chapter is about guessing who is not going to turn up for dinner. The maximum number of guests should be invited but no two of them should dislike each other.

The ninth chapter is about selecting candidates with varied talents. This puzzle is actually an instance of the set-covering problem, which has a large number of applications.

The tenth chapter is about the n-queens problem.

The eleventh chapter is about tiling a courtyard with L-shaped tiles.

The twelfth chapter is about the towers of Brahma puzzle with a twist.

The next chapter is about a disorganized handyman who has a whole collection of nuts and bolts of different sizes in a bag. Each nut is unique and has a matching unique bolt, but the handyman mixed them up. The objective is to find the right pairs of nuts and bolts. In this chapter, the author mentions the fact that built-in functions in Python may appear to be faster not because they are algorithmically better but because they have been carefully written in a low-level language.

The fourteenth chapter is related to the Sudoku puzzle.

The fifteenth chapter is about counting the ways change may be given using known denominations.

The sixteenth chapter is titled: "Greed is good." The problem is to maximize the number of courses that a student can take in any given semester.

The seventeenth chapter is about grouping anagrams.

The eighteenth chapter is about memoization. The problem introduced here requires us to pick a subset of a given set of coins so as to maximize their sum, although we are not allowed to pick two adjacent coins.

The nineteenth chapter is titled: "A weekend to remember." Your friends have to be invited for dinner, which you host on consecutive nights. There are conditions though: Each of your friends must attend exactly one of the two dinners. In addition, if A dislikes B or B dislikes A, they cannot both be in the same dinner party. This puzzle is related to bipartite graphs.

The twentieth chapter is about six degrees of separation, an observation that "everyone is six or fewer steps away, by way of introduction, from any other person in the world." The breadth-first search algorithm is introduced here.

The last chapter is related to the twenty questions game albeit with a twist. The binary search tree data structure and a greedy algorithm are the key concepts of this chapter.

3 Opinion

The title of the book could confuse some of those hunting for books. A better title would perhaps be: *Learning algorithms through programming and puzzle solving.* Perhaps an even more apt title might be: *Python programming for the puzzled.* It should be stated that not all the puzzles may be of interest to all the readers. Solving all the puzzles will not be easy for many. The author has taken effort to foster creative thinking along with programming. The puzzles in the book may lead the readers to solve even more mind-blowing puzzles in the future.

One may ask whether there is a positive correlation between puzzle-solving performance and coding experience, and between the puzzle difficulty for humans and non-human AI solvers. An interesting article by Schuster et al. studies this [1]. They have created an interesting database of programming puzzles in Python [5] that contains a dataset of Python programming puzzles for teaching and evaluating an AI's programming proficiency. Schuster and others state that puzzles are often used to teach and evaluate human programmers. Many classic puzzles such as the Tower of Hanoi teach fundamental concepts such as recursion, and programming competition problems, sometimes referred to as puzzles, evaluate a participant's ability to apply these concepts. Puzzles are also often used to judge programmers in job interviews, and some puzzles such as the RSA-factoring challenge test the limits of state-of-the-art algorithms.

This book should definitely help its readers improve their puzzle-solving as well as programming and algorithm skills that apply in any language not just Python. I believe this book is surely much more interesting and easier to follow compared to other programming books. However, it should be mentioned that the book does not really describe the basic technical steps for writing Python programs. Those who have no programming experience should invariably opt for a more traditional programming book. The plus side of this book is that it focuses more on teaching how to think as a programmer rather than just teaching syntax.

Although some readers may feel that coding puzzles may be a suitable way ahead if they are aimed at the beginner, they may also feel that some advanced puzzles tend to rely on techniques that are perhaps obscure. This in the worst scenario could lead to bad programming practices. Solving puzzles may be good for two reasons: (i) there will be focus on reasonable length of code for learning a new programming language, and (ii) the reader may learn by doing. The author follows this idea. However, this approach may not work if a reader is completely new to a programming language such as Python. Puzzles by themselves often do not teach a reader the techniques needed to design good quality software. As a matter of fact, there are often many language-specific rules of thumb for constructing utilitarian and expandable applications. Often a reader is unlikely to get exposed to these by merely solving just puzzles. A professional coder may also wonder whether coding puzzles is indeed reflective of real-world production code. One may even go to the extent of saying that solving puzzles does not teach you how to write beneficial code, or perhaps maintainable code.

I believe this book would not be worth buying for readers who are unfamiliar with Python, not interested in puzzles, and don't like programming. The readers should be reasonably good programmers interested in problem solving using Python to get maximum benefit from this book. I believe that the objective of this book is more to get the learner to think like a programmer rather than to teach the learner how to program. Programmers are often concerned with mundane chores rather than puzzles. Some of the chapters do not highlight the important algorithmic paradigms they use. For example, the fifteenth chapter on offering change does not state in the beginning that greedy algorithms are used. Nevertheless, the book should be of interest to those who want to learn how to solve puzzles and simultaneously learn programming using Python.

References

[1] T. Schuster and A. Kalyan and A. Polozov and A. Kalai. Programming Puzzles. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 1, 2021.

[2] `https://mitpress.mit.edu/books/programming-puzzled`

[3] `https://www.youtube.com/playlist?list=PLUl4u3cNGP62QumaaZtCCjkID-NgqrleA`

[4] `https://ocw.mit.edu/`

[5] `https://github.com/microsoft/PythonProgrammingPuzzles`