Programming Languages Report

Evelyn Lawrie
Chapman University

September 6, 2023

Abstract

This report is composed of weekly homework assignments for the Programming Languages course at Chapman University. First, an introduction describes the objectives and contents of this report. Then, further sections explore each assigned problem, week by week. The report ends in a conclusion that summarizes the goals of this project and the larger context it belongs to.

Contents

1 Introduction

This report's goal is to encourage problem solving and original thinking through the implementation of several self-contained weekly homework assignments. The task entails implementing a solution to each problem in a programming language of choice and then working out a given example of the algorithm's logic. By thinking through and thoroughly understanding each step of the algorithm, a deeper understanding of the logic behind why algorithms work the way that they do is cultivated.

2 Week 1

Week 1's task is implementing the GCD (greatest common divisor) algorithm.

2.1 GCD Introduction

The greatest common divisor algorithm can be understood from a geometric, algebraic, and programming perspective. The basis of this algorithm is the idea that whatever divides two numbers n and m where n > m must also divide n - m. This can be explored geometrically by visualizing small rectangles dividing up bigger rectangles until there is no smaller common divisor that will evenly fit in rectangles. This algorithm is also referred to as Euclid's algorithm, stemming from the Greek mathematician known as the "father of geometry" [GCD].

When exploring how algebra can solve the greatest common divisor problem, this entails breaking up the problem into three cases:

- n > m
- n < m
- n = m

Here is the algebraic definition of Euclid's algorithm using the three cases as described above [LTX]:

$$gcd(n,m) = \left\{ \begin{array}{ll} gcd(m, m-n), & \text{if } n > m \\ gcd(n, n-m), & \text{if } n < m \\ n, & \text{if } n = m \end{array} \right\} \tag{1}$$

2.2 GCD Code

Below is a recursive implementation of the GCD function in Java.

```
// recursive function to implement the greatest common divisor of two integers
int gcdCalc(int n, int m) {
       if (n > m) {
           return gcdCalc(m, n - m); // subtract m from n when n is bigger
       }
       else if (n < m) {
           return gcdCalc(n, m - n); // subtract n from m when m is bigger
       }
       else {
           return n; // return n when m and n are equal
       }
   }
```

2.3 Example Logic

Below is an implementation of the greatest common divisor algorithm using the example numbers 9 and 33:

$$\begin{aligned} gcd(9,33) &= gcd(9, 33-9) && (2)\\ &= gcd(9,24) && (3)\\ &= gcd(9, 24-9) && (4)\\ &= gcd(9,15) && (5)\\ &= gcd(9, 15-9) && (6)\\ &= gcd(9,6) && (7)\\ &= gcd(9-6, 6) && (8)\\ &= gcd(3,6) && (9)\\ &= gcd(3, 6-3) && (10)\\ &= gcd(3,3) && (11)\\ &= 3 && (12) \end{aligned}$$

3 Conclusion

This report details various logical problems and their solutions implemented throughout this semester. As illustrated in this report, these algorithms can be approached in various different ways. Exploring different

approaches to the same problem hones the skill set of novel solution creation, which allows programmers to complete tasks by thinking outside of the box in ways that seem less than straightforward. The thought processes behind the solutions to these problems can be applied to technical interviews, future computer science courses, and projects assigned in the workplace within the field of computer programming. The problem-solving techniques developed in this class and through this report serve as a basis of knowledge to assist students with our future endeavours.

References

[LTX] Cases and piecewise functions in LaTex, LaTex-Tutorial.com, 2023.

[GCD] Alexander Kurz, gcd and Euclid's algorithm, YouTube, 2020.