University of Bergen

INF237

Algorithm Engineering

Personal notes from the problem solving

Author:
Oskar Leirvåg (OLE006)

January 18, 2020

Contents

Abstract

This document is meant to give an insight in how the algorithms were designed. The intention is to give a deeper explanation to the thought process in the hopes of helping others understand while also helping myself to remember how it was done. It is important to mention that this course does require an oral test at the end of the semester and this text will help me in the preparation for this.
This course uses Kattis as a tool to publish the weekly challenges, and also test submissions. The algorithms that are submidted are thurougly tested over many hidden tests to affirm that they do fulfill all requirements. In order to pass this course a given number of problems must be completed each week, and they usually hold a theme in common.

Assignment 1: Ad-hoc

The first week was obviusly only a few simple problems as to get to know the Kattis tool. I used this time to experiment with some different programming languages like Java, C++ and Haskell.

Taks one: Oddities

Programing language: Java *Difficulty: Beginner* *Est: 0.5 hrs.*

This problem was really straight foreward. Given the input of any random number n of a given size, write if the given number is *odd* or *even*. So I guess there is little to say about the logic here. Could possibly have been solved even faster in Haskell than java.

Taks two: Different Problem

Programing language: Haskell *Difficulty: Beginner* *Est: 0.5 hrs.*

This problem simply gives you two numbers a and b and wants the output of the **absolute distance**. This is solved quite simply in Haskell with the recursive function

```
solve :: [Integer] -> [Integer]
solve [] = []
solve (a:b:xs) = abs(a - b):(solve xs)
```

Taks three: Guess the number

Programing language: C++ *Difficulty: Easy* *Est: 2 hrs.*

This is a simple simulation of the old guessing game. The computer decides a number $1 \leq n \leq 1000$, and your algorithm should be able to guess the correct number within 10 questions given an honest responce of higher/lower/correct. This has to be done by a binary search.

Taks four: Turtle master

Programing language: C++ *Difficulty: Easy* *Est: 3 hrs.*

In this problem we are given a game-board with a robot, a goal and some obsticles. Then we are given inputs (commands) for the robot to follow, and wee shall simply simulate the result. Then if all criteria is met we should return either *Diamond!* if correct, or *Bug!* if any error/illegal move occurs.
The short solution to this problem was simply a full on object oriented approach. I simply made a Position object and read from the map if any new positions was valid. Now finally if the robot was standing on the goal in the end, print *"Diamond!"*.