Software Engineering SS18

Course Exercises and Material

May 4, 2018

1 Introduction

The purpose of this problem set is to improve your understanding of Abstract Data Types(ADT). In the previous exercise you saw an example of an ADT, namely the (faulty) implementation of a directed graph.

Designing an abstract data type involves determining what services should be provided and what their behaviour should be. More precisely it involves writing a specification.

Note that our graph from the last exercise is dependent of a specific implementation. The concept of a graph however does not limit itself to such implementation details. The goal of this exercise is to design (and use), a graph interface along with its specification as well one or more implementations of said interface.

2 Problem

2.1 Interface Design

Design an interface for a graph, that does not assume a specific implementation of said graph. You can take inspiration from the specification of the last exercise, but note that it specifically targets directed multi-graphs.

2.2 Implementing and Testing Graphs

Implement a specific implementation of your graph interface, and create tests to ensure that it complies with your specification. It is up to you to decide if your graph edges will be directed/undirected, if your graph will be strored as an adjacency list/matrix etc.

2.3 Use your graph implementation

Write a program that given a graph representing train routes, gives you directions from one city to another. To this end, you should implement a function

`getDirections(String fileName, String origin, String destination)`, which takes an input file name and two city names and returns a String containing directions from the origin to the destination.

To read the input file, you should additionally implement a parser that can read the following format:

```
city1 city2
city2 city3
```

Meaning, there are 3 cities, and there goes a train from city1 to city2 and city2 to city3. Note that if you implemented your graph as undirected, it also means that every train also goes in the opposite direction.

To get directions between cities, a simple breadth first search is sufficient.