UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS212: Practical 7

Release: Tuesday 23 April 2019, 07:00
Deadline: Wednesday 24 April 2019, 07:00

Objectives

The aim of this practical is to learn how to create Graphs and how to search and traverse them in different ways.

Instructions

Complete the tasks below. Certain classes have been provided for you in the *files* zip archive of the practical. You have also been given a main file which will test some code functionality, but it is by no means intended to provide extensive test coverage. You are encouraged to edit this file and test your code more thoroughly. Remember to test boundary cases. Upload **only** the given source files with your changes in a compressed archive before the deadline. Please comment your name **and** student number in at the top of each file.

Graphs - Shortest Path

A Graph is a collection of vertices and the connections between them. The connections are called edges. Each edge connects a pair of vertices. If the edges are not bi-directional, but directed from the start vertex to the end vertex, the graph is a directional graph or digraph. Each edge can be assigned a number that can represent values such as cost, distance, length or weight. Such a graph is then called a weighted digraph. You are required to implement a Shortest Path algorithm for a weighted digraph. Your algorithm must be able to handle cycles and self-cycles. It must also be able to deal with unreachable nodes. A distance of infinity must be returned in that case. You may assume that the weights of the digraph are always positive. You have been given functional `Vertex` and `Edge` classes and a partially implemented `Graph` class to use. Your task is to implement the following methods in the `Graph` class according to the given specification.

Task 1a: Shortest Path [20]

```
List<Vertex> getShortestPath(Vertex sourceVertex, Vertex targetVertex)
```

This function should return the shortest path from the given `sourceVertex` to the given `targetVertex`. The returned list should contain all the vertices from the source vertex to the target vertex in the order that describes the shortest path. Should the target vertex not be reachable, an empty list should be returned (**not null**). The predecessors can be stored in the provided field in the Vertex class. The pre-condition is that the graph is a positive weighted digraph.

Task 1b: Shortest Distance [20]

```
double getShortestPathDistance(Vertex sourceVertex, Vertex targetVertex)
```

This function should return the shortest total distance from the given `sourceVertex` to the given `targetVertex`. Should the target vertex not be reachable, infinity should be returned. Use the `Double.MAX_VALUE` Java constant as infinity. The calculated distances can be stored in the provided field in the Vertex class. The pre-condition is that the graph is a positive weighted digraph.

You should use your own helper functions to assist in implementing these methods as per specification. However, you may not modify any of the given method signatures.

Submission

You need to submit your source files on the Assignment website (assignments.cs.up.ac.za). All tasks need to be implemented (or at least stubbed) before submission. Place all the source files including a makefile in a zip or tar/gzip archive (you need to compress your tar archive) named uXXXXXXXX.zip or uXXXXXXXX.tar.gz where XXXXXXXX is your student number. There should be no folders in your archive. You have 24 hours to finish this practical, regardless of what practical session you attend. Upload your archive to the *2019 Prac7 - Tuesday* slot on the Assignment website. Submit your work before the deadline. **No late submissions will be accepted!**