UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS212: Practical 8

Release: Monday 14 June 2021, 18:00
Deadline: Friday 18 June 2021, 18:00

PLAGIARISM POLICY

UNIVERSITY OF PRETORIA

The Department of Computer Science considers plagiarism as a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with consent) and copying material (such as text or program code) from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to `http://www.library.up.ac.za/plagiarism/index.htm` (from the main page of the University of Pretoria site, follow the *Library* quick link, and then choose the *Plagiarism* option under the *Services* menu). If you have any form of question regarding this, please ask one of the lecturers, to avoid any misunderstanding. Also note that the OOP principle of code re-use does not mean that you should copy and adapt code to suit your solution.

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 with positive and negative weights. Your algorithm must be able to deal with unreachable nodes. A distance of infinity must be returned in that case. Your algorithm must also be able to handle cycles and self-cycles. Negative cycles should be detected. 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 1: Shortest Path [20]

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

This function was also implemented in prac 7. It should be a bonus for you. The function should return the shortest path 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. If a negative cycle is detected, `null` should be returned. The pre-condition is that the graph is a weighted digraph.

Task 2: Shortest Distance [20]

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

This function should return the shortest total distance to the given `targetVertex`. Should the target vertex not be reachable, infinity should be returned. Use the Java constant `Double.POSITIVE_INFINITY`. The calculated distances can be stored in the provided field in the Vertex class. If a negative cycle is detected, negative infinity should be returned. Use the `Double.NEGATIVE_INFINITY` Java constant. The pre-condition is that the graph is a 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 `https://ff.cs.up.ac.za/`. All tasks need to be implemented (or at least stubbed) before submission. Place **ONLY Graph.java** file 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. You have 4 days to complete this practical, regardless of which practical session you attend. Upload your archive to the *Practical 8* slot on the Assignment website. Submit your work before the deadline. No late submissions will be accepted.