CSCI203 ASSIGNMENT THREE

Due 23:55 Sunday 24/10/2021

Task:

This assignment involves extension to the single source-single destination shortest path problem.

Program:

Your program should:

1. Read the name of a text file from the console. (Not the command line)
2. Read an undirected weighted graph from the file.
3. Find the shortest path between the start and goal vertices specified in the file.
4. Print out the vertices on the shortest path, in order from start to goal.
5. Print out the length of the shortest path.
6. Find the second shortest path between the start and goal vertices.
7. Print out the vertices on the second shortest path, in order from start to goal.
8. Print out the length of the second shortest path.

The data files are constructed as follows:

- Two integers: $nVertices$ and $nEdges$, the number of vertices and edges in the graph.
- $nVertices$ triples consisting of the vertex label and the x- and y-coordinates of each vertex. (An int followed by two $doubles$)
- $nEdges$ triples consisting of the labels of the start and end vertices of each edge, along with its weight. Note: the weight associated with an edge will be greater than or equal to the Euclidean distance between its start and end vertices as determined by their coordinates. (Two ints followed by a $double$)
- Two vertex labels, the indicating the start and goal vertices for which the paths are required. (Two ints)

A possible solution to the second shortest path problem is as follows:

```
For each edge e_i on the shortest path:
     Find the shortest path on (V, E – {e_i}).
The shortest of these is the second shortest path.
```

Output to standard output will consist of the following data:

- The number of vertices and number of edges in the graph
- The start and goal vertices
- The Euclidean distance between the start and goal vertices calculated using their coordinates
- The vertices on the shortest path, in order from start to goal.
- The length of the shortest path.
- The vertices on the second shortest path, in order from start to goal.
- The length of the second shortest path.

Implementation Requirement:

1) Part of the purpose of this subject is to gain an in-depth understanding of data structures and algorithms. As such, all programming tasks in this subject require you to choose appropriate data structures and algorithms, and to implement them yourself.
2) You may use any data structures or algorithms that have been presented in class. If you use other data structures or algorithms appropriate references must be provided.
3) Programs must compile and run under `gcc` (C programs), `g++` (C++ programs), *java* or *python3*. Programs which do not compile and run may receive no marks.
4) Programs should be appropriately explained with comments.
5) All coding must be your own work.
6) Use of STL, Java Collection, collection framework and any third party libraries of data structures and algorithms in your language choice is disallowed.
7) If you use any references other than the lecture notes, ensue you cite them or you may receive 0 marks for plagiarism. A clear comment in your code is sufficient.
8) Code sourced from textbooks, the internet, etc may also not be used unless it is correctly credited. In the event that you use code sourced in this way you will not receive marks for that part of the program.
9) You may use dynamic memory allocation to create the data structures you need at the start of program execution.

Report:

A pdf file describing your solution and program output should be produced. This file should contain:

1. A high-level description of the overall solution strategy.
2. A list of all of the data structures used, where they are used and the reasons for their choice.
3. A list of any standard algorithms used, where they are used and why they are used.
4. The output produced by your program on the provided "`a3-sample.txt`" file. ***Use screenshot(s) to show the running and output of your program***.
5. The report should be no more than **3** pages. If it is more than **3** pages, marking will be only based on the first three pages.
6. The report pdf file should be called `<your login>_a3.pdf`.

Marking Guide:

- Programs submitted must work! A program which fails to compile or run may receive a mark of zero for the program.
- A program which produces the correct output, no matter how inefficient the code, will receive a minimum of 50% of the mark for the program.
- Additional marks beyond this will be awarded for the appropriateness, i.e. efficiency for this problem, of the algorithms and data structures you use.
- Programs which lack clarity, both in code and comments, will lose marks.
- The total mark will be determined based on both your code and the accompanying report.

Submission via Moodle:

- Source code for the assignment should be typed into a single file called `<your login>_a3.ext`, where `ext` is the appropriate file extension for the chosen language.
- Your submission will consist of two files:
  `<your login>_a3.ext` and `<your login>_a3.pdf`
  where `ext` is one of `c, cpp, java` or `py`. `<your login>` is your UOW login name