CSCI203 ASSIGNMENT THREE

(Due 23:55 Sunday 22/10/2023)

Must be submitted via Moodle

Task:

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

Your program should:

1. Read the name of a text file from the console (Not the command line), which means that you should get file name from users' input.
2. Read a directed weighted graph from the file.
3. Find the shortest path between the start and the goal vertices specified in the file.
4. Print out the vertices on the shortest path, in order from the start to the goal.
5. Print out the length of the shortest path.
6. Find the longest path between the start and the goal vertices.
7. Print out the vertices on the longest path, in order from the start to the goal (each vertex can only be visited once maximally).
8. Print out the length of the longest path.

The data files are constructed as follows:

- There are 122 lines and each line contains one description of the graph.
- The 1st line contains **two integers** denoted as $\boldsymbol{nVertices}$ **and** $\boldsymbol{nEdges}$, which are the number of vertices and edges in the graph.
- The 2nd until $(nVertices+1)$st lines are nVertices triples of the form **k x y** (an *int* followed by two *doubles*),
  - k is the vertex label
  - x is the x-coordinates of the vertex
  - y is the y-coordinates of the vertex.
- Line $nVertices+2$ until line 121 are nEdges triples of the form **i j w** (Two *int*s followed by a *double*), where
  - i is the labels of the start vertex of the edge,
  - j is the labels of the end vertex of the edge,
  - w is the weight of the edge.

  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.
- The last line in the data file is two vertex labels indicating the start and the goal vertices for which the paths are required. (Two *int*s)

Output of the program will consist of the following data:

- The number of vertices and edges in the graph
- The start and the goal vertices
- The **Euclidean distance** between the start and the goal vertices is calculated using their coordinates
- The vertices on the shortest path, in order from the start to the goal.
- The length (weight) of the shortest path.
- The vertices on the longest path (each vertex can only be visited once maximally), in order from the start to the goal.
- The length (weight) of the longest path.

Implementation Requirement:

1) Part of the purpose of this subject is to gain an in-depth understanding of data structures and algorithms. Please choose appropriate data structures and algorithms, and to implement them all by 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.

4) Programs should be appropriately explained with comments.

5) All coding must be your own work.

6) You are only allowed to include or import input, output, and file streams in the beginning of your program. That is, you should only use built-in data structures and functions to implement your program. Do not use STL, Java Collection, collection framework and any third-party libraries of data structures and algorithms in your language choice.

7) Math libraries for calculating basic numeric operations such as the square root is allowed. For instance, you could use #include <cmath> in C++ to calculate the square root for Euclidean distance. Similar math libraries in Java and Python are also allowed.

8) If you use any references other than the lecture notes, ensure you cite them. Otherwise, it is plagiarism. A clear comment in your code is sufficient.

9) 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.

Report:

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

1. A high-level description (in pseudo-code) of the overall solution strategy.
2. A complexity analysis of your solution with big-O notation and sufficient justification.
3. A list of all of the data structures used, and the reasons for using them.
4. A snapshot of the compilation and the execution of your program on the provided "`a3-sample.txt`" file.
5. The outputs (the shortest and longest paths) are produced by your program on the provided "`a3-sample.txt`" file.
6. The report pdf file should be called `<your login>-a3.pdf`

Submission via Moodle:

- Please submit your source code and the pdf report as a zip file (named as `<your login>-a3.zip`) to CSCI203 Moodle site (Assignment 3 submission folder) before the deadline.
- Please note that the email submission will **NOT** be accepted.
- If an extension (maximally 1 week) is required, please submit an academic consideration via SOLS **before** the deadline.
- Late submission will receive 25% penalty per 24 hours and a zero mark after 4 days.

Marking Guide:

- Programs submitted must work (can be compiled and executed)! A program that fails to compile or run will lose marks.
- If your program produces different output from what is reported in the pdf file, a mark of zero will be graded.
- A program that produces the correct output, no matter how inefficient the code, will receive a minimum of 50% of the program component of the mark.

- 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 that lack clarity, both in code and comments, will lose marks. The total mark will be determined based on both your code and the accompanying design pdf document.