COMP - 4433 WA
Assignment 5
Due Date: April 03, 2024 @ 23:55
Total Marks: 100

Dijkstra's Algorithm in Pathfinding: Navigation System Implementation

Problem Statement:

- Define the problem of route planning in a city navigation system where a user needs to find the shortest path from their current location to a destination while considering the distance.
- You are required to implement a graph using adjacency list and adjacency matrix.

Working with Partner:

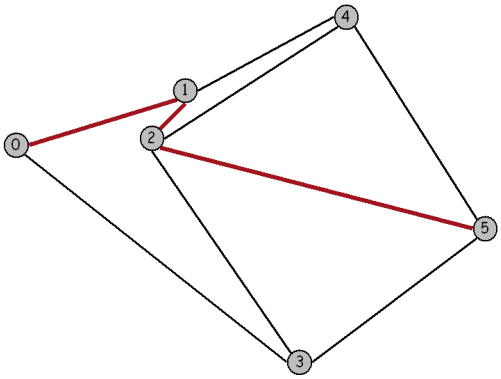
- You are strongly encouraged, but not required, to work with a partner.
- You write the name of both partners at the top of the files when you turn it in. No more than two students may form a group.
- You may divide the work however you wish, but place the names of both partners at the top of all files.
- If you work in a group, you should:
    - Turn in only *ONE* submission including per group.
    - Work together and make sure you both understand the answers to all questions.
    - Test all of your code together to be sure it properly integrates.

Design:

- For this assignment you will be working with *maps*, or graphs whose vertices are points in the plane and are connected by edges whose weights are Euclidean distances. Think of the vertices as cities and the edges as roads connected to them. To represent a map in a file, we list the number of vertices and edges, then list the vertices (index followed by its x and y coordinates) and then list the edges (pairs of vertices). For example, **input-graph.txt** represents the map below:

```
6 9
0 1000 2400
1 2800 3000
2 2400 2500
3 4000    0
4 4500 3800
5 6000 1500

0 1
0 3
1 2
1 4
2 4
2 3
2 5
3 5
4 5
```



- **[30 marks]** Your program should start by prompting the user for the name of an input file that specifies a map. After reading in the map from this file, your program should prompt the user for a pair of numbers representing the source and destination vertex, and will compute the corresponding shortest path. For instance, the red edges in the figure represent the shortest path from vertex 0 to vertex 5.

- **[30 marks]** The C++ code of Adjacency List implementation is provided in **adjacency-matrix.cpp**. This file will help you to create a data structure for your graphs first and then apply Dijkstra Algorithm. Adapt the adjacency matrix code and implement graphs using adjacency list too.

What is Expected?

- Your program should prompt the user to provide several source-destination pairs before they decide to quit the program. After quitting, your program should be able to provide the time used to find the shortest path.
- **[20 marks]** The source-destination pair should be processed by following Adjacency List and Adjacency Matrix Data structure following Dijkstra Algorithm and at the end of execution, it should print the time taken by both data structures and let us know which one is better in terms of time complexity.

Pseudo Code of Dijkstra Algorithm in C++

```
function dijkstra(G, S)
    for each vertex V in G
        dist[V] <- infinite
        prev[V] <- NULL
        If V != S, add V to Priority Queue Q
    dist[S] <- 0

    while Q IS NOT EMPTY
        U <- Extract MIN from Q
        for each unvisited neighbour V of U
            temperoryDist <- dist[U] + edgeWeight(U, V)
            if temperoryDist < dist[V]
                dist[V] <- temperoryDist
                prev[V] <- U
    return dist[], prev[]
```

What to submit?

- Use the name map.cpp for the program that reads the vertices in **canada.txt**, builds an edge-weighted graph, and then interacts with the user to obtain source-destination pairs to find the shortest path between by selecting the vertices from a given list. Submit the files map.cpp, and any other C++ file that may support the execution.
- The canada.txt file contains information of weighted graph starting from **a.**
- Create a READ_ME file and list the steps to follow for correct evaluation of your program.

Testing and Validation: [10 marks]

- The implementation will be tested by providing the vertices to compute the minimum distance. You are required to save all vertex information in an array, remove duplicates and show the list of cities (vertices) to the user before prompting the user to provide a source-destination pair.
- **input-Graph.txt** will provide help to check the correctness of your program on a smaller file.
- The file canada.txt should pass to the program once it is presenting results on **input-Graph.txt**.
- **Adjacency-Matrix.cpp** is provided.

Submission Guidelines [10 marks]:

- Submit map.cpp, READ_ME.txt or any other related file.
- Ensure that the code is properly commented and follows best coding practices.
- Provide clear instructions on how to compile and execute the code.
- Copy all files in a folder and give it a name 4433-WA-A5-[STUDENT-FIRST-NAME] compress it and upload it on D2L by the specified deadline .
- Submit the assignment by the specified deadline.
- **Not adherence** to the submission guidelines will result in marks deduction by 20%.

Late Penalty

The late penalty for assignment is [2.5^i] (2.5 to the i-th power, rounded to the nearest integer), where i>0 is the number of days you are late. So, if you hand-in your assignment 1 day late, you will be penalized 3%, a delay of 2 days will decrease your grade by 6%, 3 days is penalized 16% and 4 days takes 39% off your grade. You cannot be more than 4 days late, Extensions will be granted only by the course instructor. If you have serious medical or compassionate grounds for an extension, you should take supporting documentation to the office of the Dean of your faculty, who will contact the instructor.

Important Note:

- Plagiarism will lead to serious consequences and a grade of zero for the assignment. The similarity check will be used to detect possible cheating cases where both students will be awarded **ZERO** if found guilty.
- No support from Chat GPT or any other source is allowed and, **IF** will be detected by in place sophisticated softwares **THEN** will be dealt with by University prescribed rules.
- Feel free to seek assistance from the instructor or teaching assistants if you encounter any difficulties during the assignment.

********** **END of ASSIGNMENT 5** **********