CS 202: DATA STRUCTURES

ASSIGNMENT 4

GRAPH THEORY

Due Date: May 18, 2020, 11:00 pm

You have been hired by the internet service company **'Quarantino'**, with the motto "**social distance: the right way".** The company provides internet to several locations spread throughout the city. But recently, another competitor company has come up. Hence, Quarantino wishes to reduce its running costs and improve its client experience in order to stay in the market. You are required to help them achieve this.

When a new client arrives to get internet service, there are several factors to consider before you lay down internet cables to their location. You must check if the client location is reachable and then find the shortest path to it. Thanks to all the Data Structures you've been studying you know designing a system that takes in the locations and their connections (path), and outputs details, is not only a better solution, but can also make your tenure in Quarantino last long.

Task 1: Dataset Parsing 20 Marks

You first need to determine how exactly will you be representing your graph in terms of the underlying data structure.
You can opt for either of the two given data structures to store your **Graph**:

1. Adjacency Matrix
2. Adjacency List

For loading the graph from the file (inside the constructor), you **must use** the **addEdge** function (will make things easier for you). It will be graded.

Please note that your code for parsing should be generalized enough to work on all datasets based on the same format as that provided to you. Corresponding to this task you are required to build a **display** function which returns a **string**, meaning that the entire graph will be displayed as a single string. For reference check the example below:

For a graph having 4 locations and 5 connections each with format:

[Location1] [Location2] [Weight]

INPUT:

n 4
c 5
A B 60
A C 10
B D 25

C B 5
D A 12
OUPUT OF DISPLAY FUNCTION:

(A,B,60) (A,C,10) (A,D,12)
(B,A,60) (B,C,5) (B,D,25)
(C,A,10) (C,B,5)
(D,A,12) (D,B,25)

A tricky part will be to decide how to map the location name 'A' to the required entry in your graph data structure. You are encouraged to make a helper function that can help you with that.
Note: All edges (connections) are **undirected** which means for A B 60, you will store both (A,B,60) and (B,A,60).

Deliverables: Graph constructor (loads file), addEdge and display.

Task 2: Connectivity Check 20 Marks

Before you lay down cables for the connection, you need to check whether you can reach the client's location. You do not want to waste money on a location that you cannot even reach.

Once your graph is ready, it needs to check whether two locations are connected or not i.e. if a path exists from one location to another location through a series of connections.

You are, therefore, required to fill a **Reachable Function** which checks, if in a given graph, one location can be reached from another. The function should return a **Boolean** meaning that it only shows if such a path exists or not.

There are several ways to check reachability. You need to perform a simple traversal of the graph. As a hint, there is a traversal that checks depth and another for breadth. You need to implement either one. There you go, we just did half of the work for you.

Note: Do NOT use the Dijkstra's algorithm for reachability. You will score 0, if you do so. Choose another way – we gave you a hint above.

Deliverables: Reachable function.

Task 3: Shortest Path Between Locations 30 Marks

At Quarantino, the maintenance team is always hard at work, visiting clients location several times in a day. However, this travelling has increased their transportation costs.

Quarantino wants you determine the shortest path to the client's locations. From your data structures class, you remember your instructor talking about the **Dijkstra's shortest path algorithm** and decide to use that.

Dijkstra's shortest path algorithm is one of the most famous algorithms for finding the shortest path between two nodes, and in your case, two locations. If your supervisor tries to find a path between two nodes which are not connected the system should cater to it. The algorithm only needs to return **the weight** of the shortest path, and not the path itself. If no path exists, **return -1.**

Task 4: Least Cost Remapping 30 Marks

Before new clients are connected to the internet connection, Quarantino needs to lay out cable wires. It must do this by minimizing the total cable length cost and make sure every single client is connected. You decide to help your company with this task.

To remap keeping costs as the major factor in mind, you will need to make a graph with minimum cumulative weight (cable length cost). A **Minimum Spanning Tree** is one way which you can achieve this task. You can use either of the following algorithms:

1. Kruskal's MST algorithm
2. Prim's MST

Your function only needs to return the **cumulative weight of the MST**. You can assume the graph is connected for this task.

Test files

Use `g++ test*.cpp` , where * is task number, to compile the test case.

Caution

Quarantino does not appreciate instances of malpractice such as hardcoding your way through tasks, which is why your supervisors will be sure to check your algorithms on maps not made available to you. Ensure that you spend more time on completing the assignment than exploring a risky way out of It.

Submission Guideline

You will only be required to submit the **Graph.h** and **Graph.cpp** files in the form of a **zip** archive with the following naming convention: "PA4_ <your_roll_number>.zip".

Good Luck! 😊