CS 3353 Fall 2023

Program 2

Discovery of the extra "pipe"

Due: 10/27 (Fri) 11:59pm.

You are the Ministry of Energy of Timbuk 4. You are in charge of building an energy network to connect all the cities within Timbuk 4. This is done by building pipes between cities so that energy can flow from one city to another freely. However, there are a few things that need to be aware of:

- Not every pair of cities can be connected by a pipe.
- Every pipe has a maintenance cost. Your goal is to build the network to connect all cities while minimizing the sum of the maintenance costs of all the pipes being built.

Task 1 : Given a list of cities, and a list of pipes that can be build (each pipe connect two cities), and the cost of maintenance for each one, come up with a list of pipe to be build such that the total maintenance cost is minimized.

After you have come up with the list of pipes to be built, and before authorizing the plan to build it, you suddenly get a phone call from the Ministry of Engineering, telling you that we now have the ability to build one more pipe, and tell you the details about the pipe (which 2 cities it connects, and the maintenance). Now you have to decide whether you want to change your plan

Take 2 (Done after task 1): After you have find the solution for task 1, now you are given an extra pipe that can built (cities it connects, and the maintenance cost). You need to decide whether you want to replace one of your pipes in the solution for task 1 with the new pipe, and if so which pipe do you replace.

Problem specification

You will be given an input file that contain the following information:

- The first line contains 2 number (n), denoting the number of cities (the cities are labeled from 1 to n) and number of pipes (e) that can be built
- Then each of the next e lines denote a pipe and the maintanence The first two numbers denote the two cities (notice that the first number need not be smaller than the second), and the third number denotes the cost (as a positive floating point number).
- Follow that, there will be a line with one number (m), denoting the number of test cases
- Then the following m lines contain the test case, each test case is a pipe that is to be added to the original set of pipes. ***Note that the test cases are independent of one another. That means each pipe is added to the original problem separately. The pipes are NOT added one after the other.***

You are to write a program to find finish the two taskes mentioned above.

A sample input file is as below:

```
5 7
```

1 5 2.5
2 3 1.4
5 2 6.9
4 2 7
1 2 1.8
3 5 11.8
4 5 2.6
3
1 4 8
4 1 1.8
2 5 2.3

For this file:

- There are 5 houses, and 7 possible pipes that can be build
- A pipe can be built between city 1 and city 5, with cost 2.5
- A pipe can be built between city 2 and city 3, with cost 1.4
- The next 5 lines are similar.
- Then the following line (3) denote there are 3 test case
- The first case means you can add a pipe between city 1 and 4 with cost 8
- The second case means you can add a pipe between city 4 and 1 with cost 1.8
- The third case means you can add a pipe between 2 and 5 with cost 2.3
- Notice that each case is independent from the other. That mean for each case you are only adding the edge to the original case with 7 pipes.

Algorithm

You are to devise the algorithm to find the solution. Here are some hints to help you devise the algorithm:

- The first task is a straightforward MST algorithm. You will need to convert the input into a graph and find the MST.
- The second task basically is asking if I add an edge to the graph, will it change the MST? Obviously you can rerun the MST algorithm from scratch. However, if you may be able to speed up the algorithm if you are willing to store some extra information when you create the MST, you may be able to answer the question faster. Also remember the question I brought up in class: "Given a tree, if you add one edge, how many cycle can it form?"

Program Specification:

You are given a structure called Link, which is defined as follows:

```
struct Link {

  public:
```

```
  int v1, v2;
  float w;
};
```

This does not come with any methods. It will be used for parameter passing and return for the methods below. You are welcome (but not required) to add any methods to it. (I will provide an overloaded ostream << method). This structure is used mainly to store information about pipes.

You are to implement aclass called MyGraph. It represents an undirected graph with weights on each edge. You must implement the following methods:

Methods:

- Constructors:
    - MyGraph(int n): Create a graph with n vertices. The vertices are labelled **1..n**
    - MyGraph(const MyGraph& g): Construct a new graph that is a copy of g
- Methods
    - bool addEdge(int a, int b, float w): Add an edge between vertex a and b, with weight w. If the edge already exists or a vertex is not on the graph, do nothing and return false. Otherwise (addition is successful) return true.
    - void output(ostream& os): Output the graph to the ostream& specified. Your output should have the following format:
        - The first line print the number of vertices.
        - Each subsequent line prints an edge. It should print three numbers: the two vertices associated with the edge (print the vertex with the smaller number first), and the weight of the edge. You should have one space character between the numbers, and no space at the end. You can order the edges whatever way you want.
        - You must follow the format strictly. Otherwise points will be taken off and you are not eligible for extra credit
    - pair<bool, float> weight(int a, int b): if there is an edge between a and b, then the first item to be return is true, and the second item is the weight of the edge. Otherwise, the first value returned if false, and you can return anything for the second value.

You are also given a separate class called MyHelper. Currently the class has only one integer as its member, and one constructor that take no parameters and do nothing. You can modify it anyway you like (or you can choose not to use it at all).

You are to implement the following two functions:

- vector<Link> Task1(int n, vector<Link>& pipes, MyHelper& helper) : Given the number of cities (n), a vector of Links, which each Link corresponds to a pipe that can be built, return the list of pipes (as a vector of Links) that solve task 1. The Links in the vector can be of any order, but no Link can repeat itself in the list.
  Helper is the reference to a MyHelper object where you can choose (not) to modify to store any information. This info will be passed to task 2.

- pair<bool, Link> Task2(int n, vector<Link> pipes, Link newPipe, MyHelper helper): this function takes in the number of cities (n), the original set of pipes, a newPipe, and the helper object that is being passed, and return whether the new pipe will modify the original solution. If it does, the pair of return will contains true, follow by the Link that is displaced. Otherwise it will return false, and the Link can be anything (it will be ignored) by the main program.
  Notice that in this case, the helper object is passed by value. So you cannot modify it inside the function.

What to hand in

You are given 3 files:

- MyGraph.h : declaration of all the types/classes/functions
- Prog2test.cpp: The driver program that read a file "test2_1.txt" and call the function and output the result.
- MyGraph.cpp: contains the code for implemenration of all the classes and methods (right now there is no code in the methods (or are make to return empty vectors).

Your task is to modify MyGraph.h and MyGraph.cpp to implement the solution of the program.

You should well comment your code, and in your comments, you should describe your algorithm to solve the program.

You should upload your modified MyGraph.cpp to Canvas (you should upload ONLY that file)

Grading (Full mark is 100)

For programs that does not compile, the best you can get is a 40

For programs that compile but does not run, the best you can get is a 70

Otherwise, your program will be judged by whether you get the task 1 correct (10 points), and whether you get task 2 correct (10 points)

For all programs that get task 1 and task 2 correct (for ALL test cases that I have), their program will be timed, and the fastest of such program will get an extra bonus:

Rank	Bonus
1	60
2	50
3-4	35
5-8	30
9-12	25
13-18	20
19-24	15
25-30	10
31-38	5

If there are ties, the bonus will be averaged and split (e.g. if two students tied for first place, each will get a bonus of 55 points).