CS 3353 Fall 2024

Program 2

Building Internet on Misty Mountains

Due: 11/5 (Tue) 11:59pm

The infamous Misty Mountains in Middle Earth was a place of Goblins and Barlogs. But recently, some human are starting to settle in the mountains. However, because of the tough terrain, there has been no Internet services to its inhabitants until recently, where Elon Tusk has developed the Start-Link satellite system.

A group of human families has decided to work together to share Internet access. Each family has its own house, and each house can access the internet in one of the two ways:

- By installing a dish to connect directly to Start-Link. For each house, the cost of installing a dish will be different.
- By connecting the house (via cable) to another house that already has internet connection. Notice that the other house need not have a satellite, as long as it is connected to the Internet (via connection to another house) then it is fine. There is a cost of building a cable between the two houses, and it may be impossible to build a cable between two houses.

Notice that it is not necessary to connect all the houses to get Internet access (for example, each house can install its own satellite). On the other hand, it is possible to install just one satellite and build a network of cable to connect all the house together (in a graph) to connect every home to the Internet.

Your task is to write a program to determine the best way to connect all the houses to the Internet.

Problem specification

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

- The first line contains 1 number (n), denoting the number of houses (the houses are labeled from house 1 to house n)
- The second line contains n (floating point) numbers, the k-th number denotes the cost of installing a satellite at house k)
- The third line contain one number, m, denoting the number of pairs of houses such that a cable can be built between them
- Then each of the subsequent m lines denote a pair of houses and the cost to connect the two houses via a cable. The first two numbers denote the two houses (notice that the first number need not be smaller than the second), and the third number denotes the cost (in floating point number).

You are to write a program to find the least cost needed to connect all the houses to the Internet.

A sample input file is as below:

```
4
21.5 10.5 11.5 16
```

5
1 4 5
2 3 4.5
4 2 2.7
1 3 6.2
1 2 6.25

For this file:

- There are four houses
- For each house (1, 2, 3, 4), the cost of installing a satellite at that house is (21.5 10.5 11.5 16) respectively.
- There are five cables that can be build.
- Between house 1 and 4, building a cable cost 5
- Between house 2 and 3, building a cable cost 4,5
- Next three lines are similar.
  (Notice that one cannot build a cable between house 3 and 4)

Algorithm

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

- The algorithm requires you to find the minimum spanning tree of a certain graph G, where you need to convert the input data to.
- The key is to figure out how many vertices are there in the graph.
- A further hint for your graph: your graph should have weights on the edges, but no weights on the vertices.

Program Specification:

You are given a class called Link, who stored information about the links. You should look at the file Link.java to see the functionality about the class.

You are to implement following class 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(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 Print() Output the graph to standard output (using System.out.print() or System.out.println()). The output should be of 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
- ArrayList<Link> MST() This function return the MST of the current graph. The output should be list of links (represented by a linked list). Each edge on the MST should be represented by a link (which store both vertices and the weight). The edges can be in any order.

You are also provided a program, "prog2.java", where you are to implement the algorithm that solve the problem:

- Public static Prog2Return Prog2(float [] satcost, ArrayList<Link> l1)
  - This is the function for the main task of program 2. The two inputs are
    - Vector<float> satcost: the cost of installing a satellite at each house. Notice that the house is labelled 1 to n, so the vector has n+1 entries and satcost[0] is ignored.
    - ArrayList<Link> l1: a ArrayList of links such that each entry contains Link representing a possible connection between two houses and the cost of building a connection between them.
    - The output should contain be an object of Prog2Return class. It has 3 attributes:
      - Boolean [] sat: a Boolean array of n+1 elemenets, initialized to false. You should set sat[i] to true if it need
      - ArrayList<Link> links: a linked list of all links that you need to build cable with.
      - Int [] path: an array of n+1 integers, intiallized to -1 (except the first element is 1). You should return the path from house 1 to the satellite in your solution. You should list all the houses that you go through in order (from house 1) to the satellite.

      Notice that the members of Prog2Return object is public, so you can manipulate them directly if you would like. The arrays will be initialized when you create the object.

  - For example, for the test case about the output should be a pair
    - sat[2] should be set to true, the other values of the sat array should remain false
    - Link should contain the list [(1, 4, 5), (2, 4, 2.7), (2, 3, 4.5)] (Notice that the edges can be in any order)
    - The path array should be [1 4 2] (and leave the rest of the entries untouched)

What to hand in

You are given 4 files:

- Prog2Run.java : The driver program. It takes input from standard input. (Notice that when I grade your program I will have a separate file which I will test the MST() method also).
- Prog2Return.java: contains the code for the Prog2Return class where you need to store the result of Prog2().
- Link.java: Class corresponding to the links that is being stored.
- MyGraph.java: The MyGraph class
- Prog2.java: The prog2 class, contain the Prog2() function which you should implemne.t

Your task is to modify MyGraph.java and Prog2.java to solve the problem. You should zip those 2 files (NOTHING MORE!) and upload the zip file to Canvas.

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

Grading

For programs that contains syntax errors, the best you can get is a 40

For programs that have no syntax errors does not run, the best you can get is a 70

Otherwise, your program will be judged by whether you get the MST correct (10 points), and whether you get the answer of the problem correct (15 points)

Bonus

For all programs that get full marks, their program will be timed, and the fastest of such program will get an extra bonus:

Rank	Bonus
1	60
2	45
3-4	30
5-6	20
7-10	15
11-14	10

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 37.5 points).