Project 3: A Love Story

CmpE 250, Data Structures and Algorithms, Fall 2021

Instructor: H. B. Yılmaz
TA: Özlem Şimşek
SA: Leyla Yayladere, Berk Atıl, Hasan Baki Küçükçakıroğlu

Due: **10.12.2021 Friday, 23.55**

1 Introduction

As Shakespeare once said "i am to wait, though waiting so be hell...", it is an obvious fact that waiting is the worst thing in the world. Do not worry!!! Our hard-working team works really hard and saves you the burden of waiting for the next project. No thanks needed, we are here for you.

Mecnun and Leyla are two lovers who live in the Anatolian country. They try to come together but Leyla's father is preventing them from doing so. Recently Leyla has tried to convince her father and she eventually manages to do. But her father offers a condition for Mecnun that he has to reach their town in time less than or equal to a threshold. The father sets the threshold and writes it to a piece of paper and hides the paper. Mecnun will be informed about the challenge but he will not be provided any information about the time that was set. The countdown for Mecnun will start at the moment he departs from his city and if he can arrive in time Leyla's father will let them marry. Mecnun has a map of the Anatolian country which demonstrates all land routes over the country and their lengths. The country is divided into two regions by a river. Leyla and Mecnun are in the same region of the country but in different cities. Leyla's city is the only city in the country where there are bridges for crossing to the other side of the country. For this reason, anyone who wants to cross over to the other side should visit this city.

Two cities can be considered neighboring cities if there is a road between them. The roads of the country are one-way for the vehicles and two-ways for the pedestrians. Mecnun will take a bus while going to Leyla to be able to reach fast. Each road has a bus service of its own and there is always a bus service available at any given time. Every bus has the same constant velocity.

Mecnun and Leyla are planning a honeymoon in which they would be traveling on foot to all the cities of the other side of the country. But each pedestrian has to pay a sidewalk-tax which is equal to the length of the road. If they can't find a decent travel route for their plan,

they will have to stay in the city where Leyla lives.

According to the legend, if they can't get together, Mecnun will disappear.

2 Details

- For each case output consists of a list of integers and one integer, list1 and int1 . List1 indicates the output generated for Mecnun's path to reach Leyla. int1 indicates the output generated for the Honeymoon path.
- Mecnun does not know the time set by the father, so he must take the shortest possible route. The shortest route to be used in list1 should be containing the IDs of the cities passed starting from Mecnun's city to Leyla's city. If there is no route from Mecnun's city to Leyla's city, list1 should contain only -1
- When Mecnun tries to reach Leyla's city, he cannot get over the bridge. On the honeymoon, they can only visit Leyla's city + cities of other side of the country.(For the sample map illustrated in Schema 1, only c7, d1, d2, d3, and d4 cities can be visited on the honeymoon.)
- If Mecnun can reach Leyla, Leyla's father will compare the time he arrived with the time on the paper and come to a decision. Since the speed of the buses is 1br/second, the arrival time is equal to the total length of the path.
- Mecnun and Leyla are planning to walk around all the cities of the other side of the country and have the walkthrough honeymoon with the lowest sidewalk-tax. The roads traveled are subject to a sidewalk-tax proportional to their length, and after the tax is paid once, subsequent uses of the road are free of charge. In this case, the length of the path they walk does not matter. The sidewalk-tax paid must be the lowest and the route must include all cities of the other side of the country. The tax paid by Mecnun will be the int1 part of the output.
- If there is no such honeymoon route, Mecnun stays in Leyla's city and int1 should be -2.
- If they cannot get married, Mecnun disappears as per the legend. In this case int1 should be -1.

3 Input & Output

3.1 Input

- The first line represents the time set by Leyla's father.
- The second line represents the total number of cities in the Anatolian country.
- The third line represents the ID of the city where Mecnun lives and the ID of the city where Leyla lives, respectively.

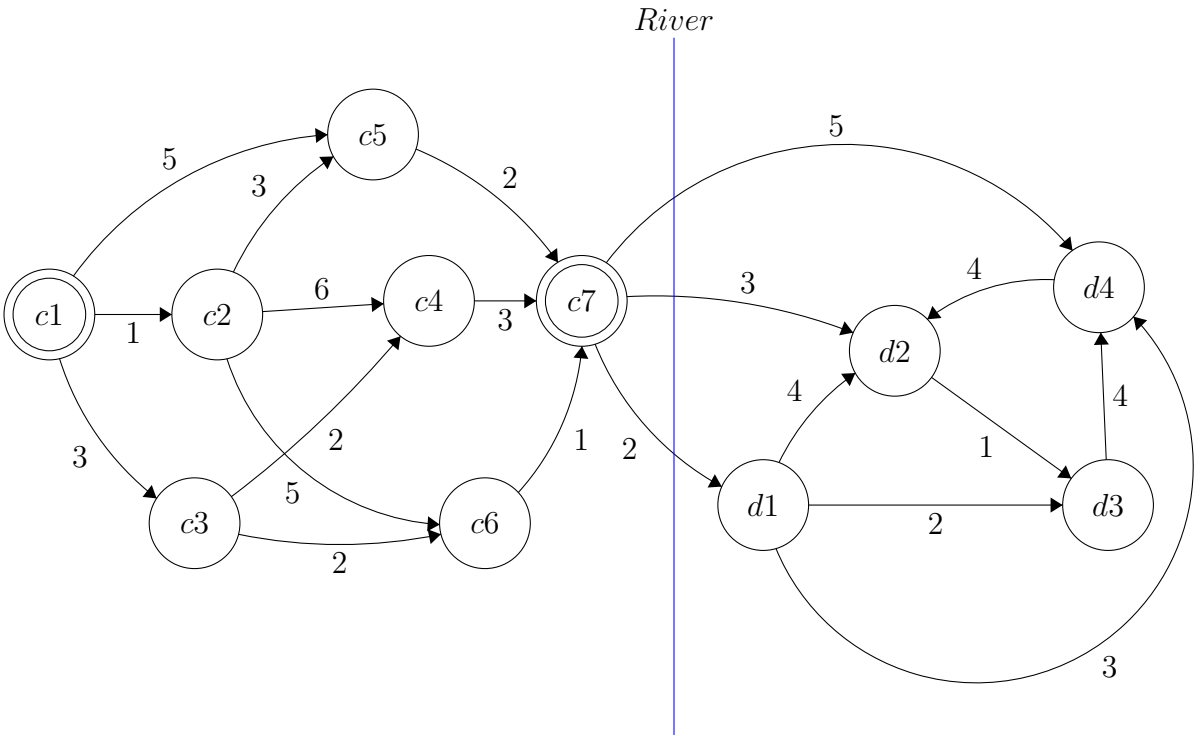
- Each next line will give the ID of a city of Anatolian country and the IDs and lengths of the cities that can be reached by vehicle from that city in pairs. For example:

  If there is a way from c1 city to c2, c3, and c4 cities of length 1, 2, and 3 respectively. Input line will be **c1 c2 1 c3 2 c4 3**

  If there is no way from c2, input line for c2 will be **c2**

Sample Input File	Explanations
8	Time limit determined by Leyla's father is 8.
11	The country has 11 cities. IDs start with c represents the cities of the side where Leyla and Mecnun lives and IDs start with d represents the cities of the other side of the country.
c1 c7	The ID of the Mecnun's city and the ID of the Leyla's city respectively.
c1 c2 1 c3 3 c5 5	The city with ID 1 has a way to c2, c3 and c5 with length of 1,3 and 4 respectively.
c2 c4 6 c5 3 c6 5	The city with ID 2 has a way to c4, c5 and c6 with length of 6,3 and 4 respectively.
c3 c4 2 c6 2	The city with ID c3 has a way to c4 and c6 with length of 2 and 2 respectively.
c4 c7 3	The city with ID c4 has a way to c7 with length of 3.
c5 c7 2	The city with ID c5 has a way to c7 with length of 2.
c6 c7 1	The city with ID c6 has a way to c7 with length of 1.
c7 d1 2 d2 3 d4 5	The city with ID c7 has a way to d1, d2 and d4 with length of 2,3 and 5 respectively.
d1 d2 4 d3 2 d4 3	The city with ID 8 has a way to d2, d3 and d4 with length of 4,2 and 3 respectively.
d2 d3 1	The city with ID d2 has a way to d3 with length of 1.
d3 d4 4	The city with ID d3 has a way to d4 with length of 4.
d4 d2 4	The city with ID d4 has a way to d2 with length of 4.

Table 1: Sample Input with Explanations



Schema 1: A schema of the sample Anatolian country

3.2 Output

1. For each case there will be two lines of output. First line will be list1 and second line will be int1.

2. List1, Mecnun's path of reaching Leyla (if he cannot reach -1 should be written) should be written.

3. Int1, the tax paid on the Honeymoon route if it is possible. (If Leyla and Mecnun don't marry -1 should be written but if they did not find a suitable honeymoon route -2 should be written.)

4. If there is more than one possible output for a test case, you can use any. The program used for grading will test their accuracy.

Output File	Explanation
c1 c2 c5 c7 or c1 c3 c6 c7	The path that Mecnun used to reach Leyla. Since there are two paths with the same minimum length of 6, one of them should be on the output file.
16	The side-walk tax paid on the Honeymoon route. (c7,d1), (d1,d4), (d1,d3) and (d2,d3) ways are used so (2+3+2+1)*2=16.

Table 2: Expected Output File and Explanation of each Statistic

3.3 Java Project Outline

Your java project will be named **Project3**. Your entry class for the project will be named **project3main**. All your .java files will be under folder **Project3/src**. Your project should be compatible with Java 16. Your program will be compiled with below command:

```
javac Project3/src/*.java -d Project3/bin –release 16
```

The input and output files can be at any folder. Design your code in order to accept full path for file arguments. Your program will be run with below command:

```
java project3main <inputfile> <outputfile>
```

Make sure that your final submission compiles and runs with these commands.

4 Grading

Grading of this project is based on the automatic compilation and run and the success of your code in test cases. If your code compiles and runs with no error, then you will get **10/100** of the project grade. The rest of your grade will be the sum of collected points from each test case. Maximum project grade is 100. If your submission is not auto-runnable, then **you will receive 0 at first and then have to issue an objection**.

5 Warnings

1. This is an individual project.
2. All source codes are checked automatically for similarity with other submissions and exercises from previous years. Make sure you write and submit your own code. Any sign of cheating will be penalized and you will get **-50** points for the project and you will get **F** grade in case of recurrence in any other project.
3. There will be time limit on test cases, so it is important to write your code in an efficient manner.
4. If the entire path you find is not suitable, you cannot get points for that route. Please don't object like "but my route is correct except for the last two cities"
5. Again, if the tax amount is not equal to correct tax amount you cannot get points. Please don't object like "but my amount is just one less than the correct one"
6. You can add as many files as you can as long as they are in the "src" folder and your project can be compiled as above. But the entry point of your program should be "project3main.java".
7. Make sure you document your code with necessary inline comments and use meaningful variable names. Do not over-comment, or make your variable names unnecessarily long. This is very important for partial grading.

6 Submission Details

You will zip up your project folder and submit on Moodle as a single .zip file. The name of the zip file is **Cmpe250_Project3_<studentid>.zip** No other type of submission will be accepted.

7 Some tips regarding the project

1. You need to know how the minimum weighted path on a directed graph is found for the first part of the project.
2. You should also know what a minimum spanning tree is to solve second part of the problem.
3. We do think that this project is one of the most important ones in this course. Therefore, we suggest starting this project early(not on the last day!).

Algorithm 2: Prim's Algorithm

```
Data: G: The given graph
      source: The node to start from
Result: Returns the cost of the MST
totalCost ← 0;
included ← {false };
Q.addOrUpdate(source, 0, Φ);
while ¬ Q.empty() do
    u ← Q.getNodeWithLowestWeight();
    totalCost ← totalCost + u.weight;
    if u.edge ≠ Φ then
        mst.add(u.edge);
    end
    included[u.node] ← true;
    for v ∈ G.neighbors(u.node) do
        if ¬included[v.node] then
            Q.addOrUpdate(v.node, weight(u.node, v.node),
             v.edge);
        end
    end
end
return totalCost, mst;
```

Figure 1: Pseudo code for implementation of MST

Figure 2: When someone says "I love you"