PROJECT ASSIGNMENT 4

Issue Date : 10.05.2024 - Friday
Due Date : 24.05.2024 - Friday (23:00)
Advisor : R.A. Görkem AKYILDIZ, Berra Nur ÖZTÜRK
Programming Language : Java 8u401 (1.8.0_401)

1 Introduction

Road designers must cope with the trade-off between less resource to construct for builders of the road and less fuel to consume for users of the road, as both of them consume resources of our globe, they must be more than careful to find the best place between these two ends.

In this project, you do not need to cope with this optimization but you are supposed to analyze advantageous and disadvantageous sides of both ends. Your task consists of three core steps. The first step is designing an algorithm that finds fastest route in terms of distance in given road-map, the second step is designing an algorithm that designs the roads (by selecting the existing ones) such that points (such as cities, villages, countries etc.) are barely connected which means there is no cycle such that you can end at the point that you have been started (which means without using any road that is already passed). The last step is analysis of these two approaches, you have to report the differences as reporting the construction material amount difference between given map and the map that you have created; and fastest route difference between the given map and yours in terms of distance that traveler wend.

2 Calculating the Fastest Route

The algorithm must give a route that is the fastest in terms of distance, you will try to reach from the starting point to destination point and you have to find the fastest one. For

reaching to the destination in fastest way, you can use the following approach:

Firstly you have to create the following structures:

- An object that holds distance, road, and two points. Distance is the distance from the starting point, road is the road that connects two points that the object holds. The points are both ends of the road.
- A list that holds the object defined above. It must be sorted before each removal. You must sort items according to the distance to the starting point, if the distances are equal, preserve the append order which means sorting them according to the append order.
- A map (dictionary) that holds point as the key and the object defined above as the value.

Start from the starting point, sort all the roads (you must sort them according to their length, if they are equal in length, you must use their IDs as tiebreaker to sort the equal ones) that are connected to the starting point, create the object defined above for all of the points at the other end of the roads. Add them to the list. Select the smallest one from the list and append it to the map (dictionary). Sort all of the roads that are connected to the point that you just appended to the map (dictionary). Append all these roads by creating the object of them as in the previous step. Then, follow the same logic, as in the first step. Note that if a point is already in the map (dictionary), do not append it again as you already discovered the fastest route from starting point to it. Note that, do not append the roads that connects the points that are already in the map (dictionary) including the starting point even if it is not in the map (dictionary), only append the roads if they have one end in the point that is not appended to the map (dictionary) yet.

3 Calculating the Barely Connected Map

In this part, assume that all of the roads are broken due to a painful disaster and you have very limited resources. Your aim is, connecting all of the points on budget, which means, you must select the shortest ones and you must avoid from creating cycles as cycle means there are more than one way to go from the point A to B which is something you must avoid to cut the budget. Note that you can only rebuild the roads that existed before the disaster, you cannot create any new roads. For building this barely connected map, you can use the following approach: Sort the points alphabetically in increasing order. Create a list to hold roads. You must sort the list in increasing order according to their length (you must use their IDs for determining which one is smaller if the lengths are equal) before each removal. Start with the point that is smallest in context of alphabetical order, add all of its roads to the list. Select the smallest road, then append it to the barely connected map. Then append all the roads that are connected to the other end of the road that you just appended to the list. Select the smallest road again. Note that, you must add the roads such that the newly added road does not create any cycle, you may use a list to hold the points that are already appended to the barely connected map for cycle detection, if both ends of the road are already appended, it means that appending it causes a cycle. **Hint: Number of roads that are supposed to be added must be exactly one less than the number of points.**

4 Analysis

In this part, you are supposed to output the statistics, you will give two statistics, the first one is, the ratio of the fastest route from the start point to the end point in barely connected map and the fastest route in the original map; the second one is ratio of the construction material used in the barely connected map and the original map, you can assume that each part of road uses the same amount of material, which means one unit (such as kilometer) of road uses one unit of construction material.

5 Definition of Input & Output

Input is very straight forward, each line contains four information (except the first line as it will contain the start and end points -which are separated with tab character- that is requested to calculate the fastest route between them) that are separated each other with the tab character. The first information is one end of the road, the second information is the another end, the third information is the length where the fourth one is ID of the road.

Roads can be assumed as two-lane roads. For the sake of easiness, points will be described as strings, such as Ankara, İstanbul; lengths will be described as in kilometers as integers, ID will be non-negative and unique integer.

Output is a bit complicated than the input but it is also an easy one. Your output will consist of three parts:

1. Fastest route from the given start point to the end point, you will write the roads that your route contains in sorted order from start point to end point. Road representation format will be as in the input.
2. Roads that barely connected map contains, they must be sorted in ascending order according to their length, if they are equal in length, you must use their IDs as tiebreaker to sort the equal ones. Road representation format will be as in input.
3. Analysis of the both parts, you must show the ratios that are described at the analysis part. You must show at the most two digit at the fraction part, do not use any rounding you can benefit from `%.2f` formatting style.

Note that please analyze the given sample I/Os to see full description of the I/O format.

6 Restrictions

- Your code must be able to execute on our department's developer server (dev.cs.hacettepe.edu.tr).
- You must obey given submit hierarchy and get score (1 point) from the submit system.
- **Your code must use the same logic that is described which means your results must be same as the given even if there are more than one fastest route or barely connected map. Any other maps or routes that are not according to the given logic will not be accepted even if they are correct.**

- **You must benefit from OOP if there is such a need; any solution that does not contain OOP where there is such a need will not be accepted, moreover, partial point deductions may exist due to partially erroneous usage of it.**
- Your code must be clean, do not forget that main method is just a driver method that means it is just for making your code fragments run, not for using them as a main container, create classes and methods in necessary situations but use them as required. Moreover, use the four pillars of Object-Oriented Programming (Abstraction, Encapsulation, Inheritance, Polymorphism) if there is such a need, remember that your code must satisfy Object-Oriented Programming Principles, also you can benefit from exceptions and even if create your own exception class if you need any.
- You are encouraged to use lambda expressions which are introduced with **Java 8**.
- You must use JavaDoc commenting style for this project, and you must give brief information about the challenging parts of your code, do not over comment as it is against clean code approach. Design your comments so that if someone wants to read your code they should be able to easily understand what is going on. You can check here to access Oracle's own guide about JavaDoc Sytle.
- You can benefit from Internet sources for inspiration but do not use any code that does not belong to you.
- You can discuss high-level (design) problems with your friends but do not share any code or implementation with anybody.
- Do not miss the submission deadline.
- Source code readability is a great of importance. Thus, write READABLE SOURCE CODE, comments, and clear MAIN function. This expectation will be graded as "clean code".
- Use UNDERSTANDABLE names for your variables, classes, and functions regardless of the length. The names of classes, attributes and methods must obey to the Java naming convention. This expectation will be graded as "coding standards".
- You can ask your questions through course's Piazza group, and you are supposed to be aware of everything discussed in the Piazza group. General discussion of the problem is allowed, but **DO NOT SHARE** answers, algorithms, source codes and reports.
- All assignments must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating.
- Submit system for this homework will be opened a few days before deadline, so please be patient.

7 Execution and Test

Your code must be executed under **Java 8u401 (1.8.0_401)** at **dev.cs.hacettepe.edu.tr**. If your code does not run at developer server during the testing stage, then you will be graded as 0 for code part even if it works on your own machine. Sample run command is as follows:

- Either `javac8 MapAnalyzer.java` or `javac8 *.java` command for compilation.
- `java8 MapAnalyzer input.txt output.txt` command for run.

8 Grading

Task	Point
Output	80*
Comments in JavaDoc Style	20* **
Total	100

CRUCIAL: Even though the given algorithms are just for helping you which means you do not have to exactly use them, your code must either use the given algorithms or you must find a way to make your code act like the given algorithms, which means, your solution will not be accepted even if it is correct but not exactly matched. For example, there may be more than one fastest route or barely connected map but there is only one fastest route and barely connected map if you follow the algorithms that are offered. THE MOST CRUCIAL PARTS ARE THE WAY OF SELECTING AND SORTING THE ROADS.
*** Even though producing correct output and commenting seems like enough to get full credit, you must obey to the given rules in the PDF (for example using concept of OOP etc.) otherwise you may face with some point deductions which may result with a grade that is as low as zero. There will be two overall multipliers about quality of your OOP and clean code separately which will vary in between 0 and 1! Note that there may be any other multipliers or point deductions in case of violation of the rules.**
**** The score of your comments will be multiplied by your overall score (excluding the comments part) divided by the maximum score that can be taken from these parts. Say that you got 60 from all parts excluding the comments and 20 from the comments part, your score for report is going to be 20*(60/80) which is 15 and your overall score will be 75.**

9 Submit Format

File hierarchy must be zipped before submitted (Not .rar, only not compressed .zip files because the system just supports .zip files).

```
- b<StudentID>.zip
    - <src>
       - MapAnalyzer.java
       - *.java (Optional)
```