Algorithms and Data Structures

Graph Traversal, MST & SPT Algorithm Assignment

You are required to implement Prim's and Kruskal's algorithm for finding the minimum spanning tree for a weighted connected graph and Dijkstra's shortest path tree (SPT) algorithm. If the implementation is too difficult and/or you can't debug it, a paper based simulation of Prim/Dijkstra and Kruskal will suffice to pass the assignment.

The program when executed at the command line will prompt the user for the name of a text file which contains a sample graph and also will prompt the user for a staring vertex. The user will then enter this name and vertex (as a number). The graph will then be read from the text file and a graph data structure will be constructed. A method should then be called to compute the graph's SPT & MST after which the SPT & MST should be outputted to the console. While the algorithm is running, it should output some of its workings to the console so that you can see it working step by step.

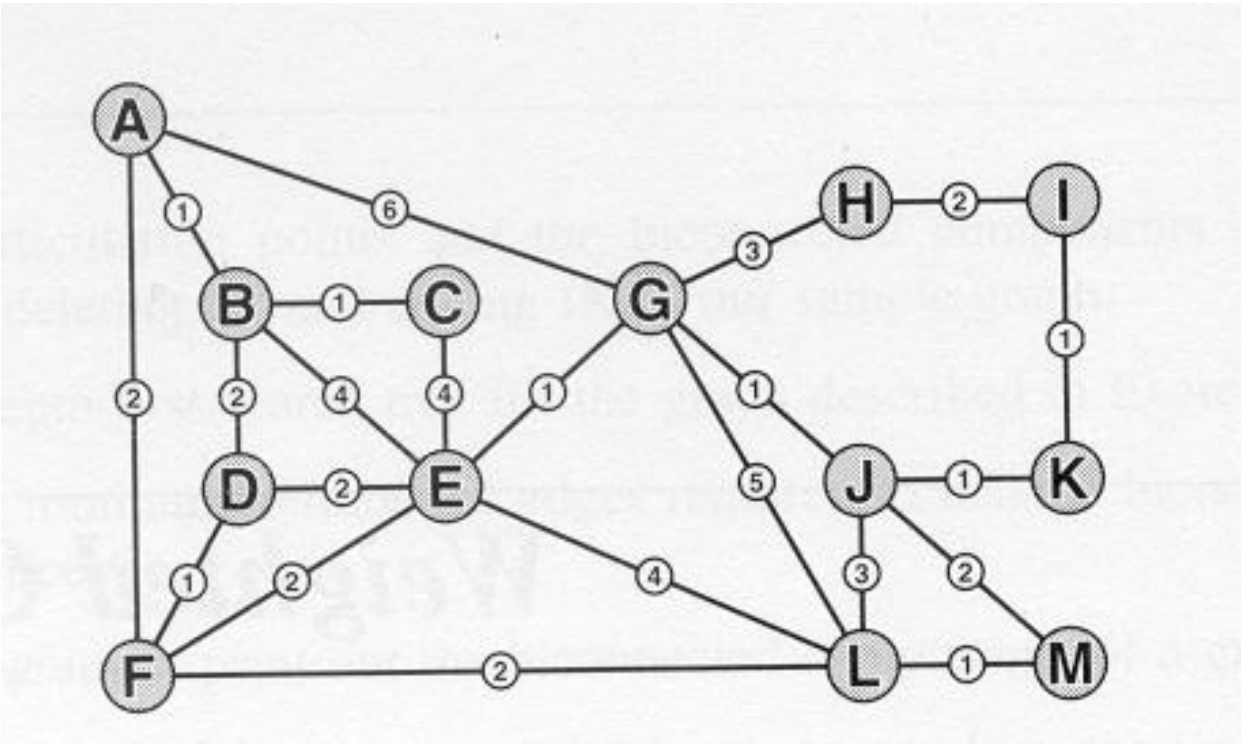
For Prim/Dijkstra, represent the graph using an adjacency lists data structure.
Also include:

- depth first traversal using recursion
- breadth first traversal using a queue (try to use Cormen's versio)

All the above 4 algorithms are all related and should be in one Java source code file.

Kruskal has only one version which uses an array of edges. In contrast Prim & Dijkstra with adjacency lists requires a priority queue or heap. You may use a heap for Kruskal too. The heap code is quite different for Kruskal. Alternatively with Kruskal you can use Quicksort or Heapsort to sort the edges instead of using a heap. Also two improvements on Kruskal that you might consider are are***: union by rank*** and ***path compression***.

You are to test your code on the graph below, save this graph in text file wGraph1.txt. For DF() BF(), Prim() and Dijkstra() start with vertex **L**. I have provided some of the Java code for the Graph and Heap classes.



Also, see if your code can find SPT/MST for one of the very large real world examples of the road graphs provided.

Make sure that

- your code is well commented and well structured
- messy code will lose marks, even if it works

Report

This is to be submitted to Brightspace in **PDF** format. It should include:

1. introduction/explanation.
2. an adjacency lists diagram showing the graph representation of the above sample graph.
3. a step by step construction of the MST using Prim's alg, starting from vertex **L**, for the above sample graph. You should show the contents of the heap, parent[] and dist[] arrays for each step in Prim.
4. a step by step construction of the SST using Dijkstra's alg, starting from vertex **L**, for the above sample graph. You should show the contents of the heap, parent[] and dist[] arrays for each step in Dijkstra.
5. a step by step construction of the MST for Kruskal and the union-find partition and set representations for Kruskal at each step.
6. a diagram showing the MST superimposed on the graph (use a highlighter if you want), easy to do using Microsoft Whiteboard.
7. a diagram showing the SST superimposed on the graph.
8. screen captures showing all your programs executing, especially the output of all implementations for the example graph.
9. a discussion/analysis/reflection on what you learned or found useful in the assignment.

For submission via Brightspace

1. Report in a file named something like **GraphAlgs-Smith-John-C18345678.pdf**. Make sure to use your name and student number in the report as shown as well as on the report cover page.
2. Two Java code files, one for Prim/Dijkstra/DF/BF and the other for Kruskal. Do **not include all the other Visual Studio** files. Also **do not submit compressed ZIP or RAR files**.

Marking

20% for graph traversal
40% for Prim & Dijkstra (including report)
40% for Kruskal (including report)