Name:_____ Date:_____

CSI 403 Homework 3

The deadline for this assignment is 11:59 PM, Tuesday, October 27th, 2020.

Very important note: You can organize your program in multiple java class files that can be compiled using a DriverMST.java file using javac *.java. You must turn in electronically in BB all your files and the video of your program compiling and running. Programs that do not compile will be assigned 0 points without exceptions.

The objective of this assignment is to implement two different algorithms for the minimum spanning tree problem and experimentally compare their computation times. The required background material is in Chapter 23, but in the following you will explicitly be given the necessary pseudo code.

An undirected graph consists of nodes and edges. Each edge has a cost associated with it.
An example of an undirected graph with edge costs is shown below.

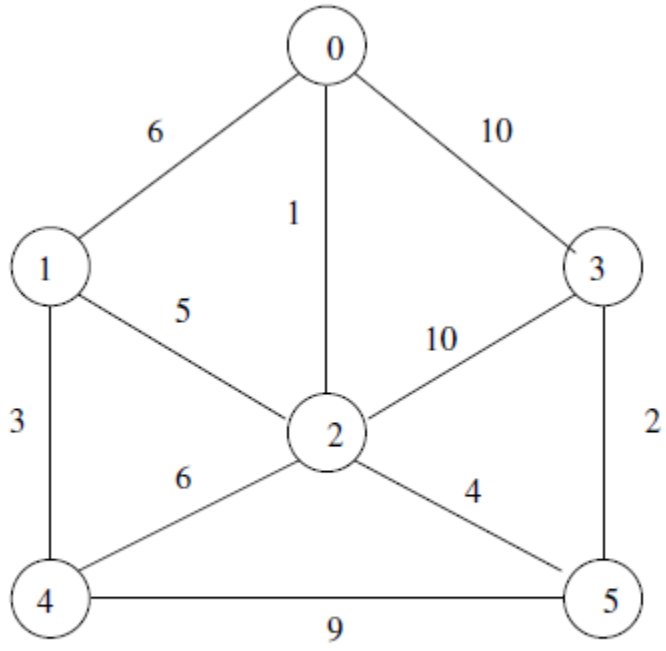

Figure 1: An undirected graph with edge costs

In the above figure, nodes are represented by circles and edges are represented by lines. The integer labeling each edge is the cost of that edge. (Note that nodes are numbered from 0.) The graph is connected; that is, we can go from any node to any other node using only the given edges.

A spanning tree for a connected graph is a tree that connects all the nodes. Such a tree uses a subset of the edges of the graph, and the cost of the tree is the sum of the costs of all the edges included in the tree. Figure 2 shows two spanning trees for the graph of Figure 1. Note that the

tree shown in Figure 2(b) has a lower cost than the tree shown in Figure 2(a).

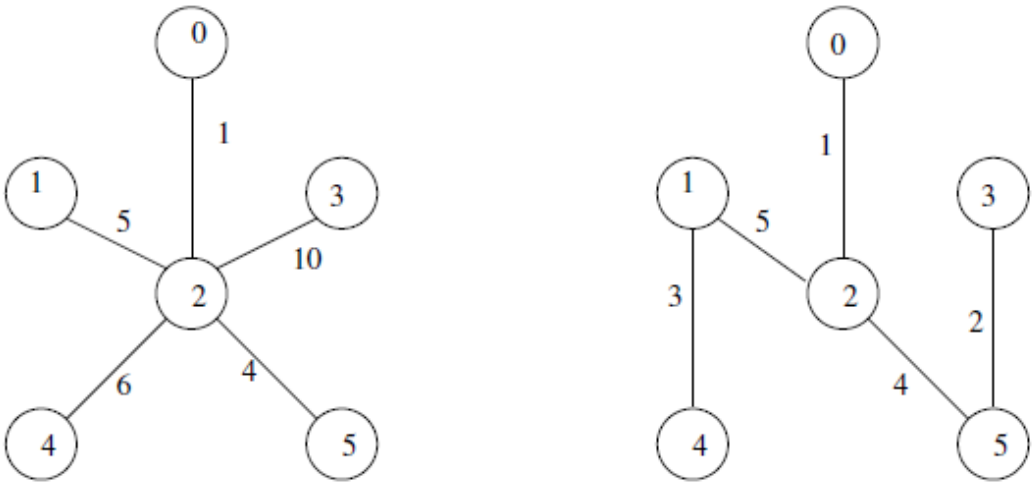

Figure 2: Two spanning trees for the graph of Figure 1

A **minimum spanning tree (MST)** for a graph is a spanning tree whose cost is minimum among all the spanning trees. The tree shown in Figure 2(b) is an MST for the graph of Figure 1.

Several algorithms for finding an MST of a connected graph are known. You are required to implement two algorithms whose pseudo-codes are given below in this assignment. Examples to illustrate these algorithms and some implementation details will be presented in class.

Command line details: In your video you should demonstrate that your program runs through a driver class named DriverMST.class. It should be tested using the command line command:

java DriverMST infile.txt

or using the command

java DriverMST infile.txt -e

where infile.txt represents the name of the input file which contains a description of the graph.

Note: additional arguments specifying the class path might also be added.

Name:_____ Date:_____

CSI 403 Homework 3

The format of the input file is discussed below. Your program must run both the algorithms on the input graph and write to System.out.println(), the following information for each algorithm (each piece of information on a separate line):

a) The execution time in ms (Please, refer back to programming assignment 1 to check how we timed parts of our programs.)
b) The cost of the minimum spanning tree.
c) If the -e option is specified on the command line, then the list of edges in the spanning tree along with the cost of each edge. (If the command line does not specify the -e option, then the list of edges should not be produced.)

Format of input file: The first line of the input file contains the number of nodes in the graph. This is followed by several lines, each specifying three positive integers separated by a space character. The first two of these integers specify the nodes joined by the edge and the third integer specifies the cost of the edge. For example, an input file to describe the graph of Figure 1 is as follows.

```
6
0 1 6
0 2 1
0 3 10
1 2 5
1 4 3
2 3 10
2 4 6
2 5 4
3 5 2
4 5 9
```

You may assume that each input graph is connected and that it has at most 1000 nodes and at most 10,000 edges.

Your program should be well-structured and documented with comments. It should be compatible with Java 1.7 and later. Some sample inputs and outputs that you can use to test your program can be found as part of the assignment on BB. Your program should correctly identify the MSTs for the sample inputs, but it will be tested with other inputs as well.

Grading policy:

1. If your program does not compile you will receive 0 points!
2. If you don't submit a video of your code compiling and running you receive a 0!
3. If your program fails to read the provided test files and produce an output, you will lose points!
4. Correct implementations of the two algorithms is required!
5. Programs that achieve the correct result on all test cases will get full points!

You should not hard code the answers for provided test cases!

This is an independent assignment. Similarity to code on the Internet or to that of other students will result in zero points for involved students and reporting for academic dishonesty.

Name:_____ Date:_____

CSI 403 Homework 3

Pseudo-codes for the two algorithms

In specifying the pseudo-codes, we have assumed that for a graph with n nodes, the nodes are numbered 0 through $n-1$. An edge between nodes i and j is denoted by (i, j).

Algorithm I – Kruskal's Algorithm: (In the following description, Steps 1, 6 and 7 should not included in measuring the computation time.)

1. Read in the number of nodes (n), the edges and their costs.
2. Sort the edges into non-decreasing order of their costs.
3. Initialize T to the empty set. (**Note:** At the end of the algorithm, T will contain all the edges in an MST.)
4. Initially, each of the n nodes is in a component by itself. (Thus, there are n components at this step.)
5. **while** (Number of components ≥ 2) {
   (a) Remove the first edge from the sorted list; let (i, j) denote this edge.
   (b) **if** (Nodes i and j are in different components) {
      (i) Add the edge (i, j) to T.
      (ii) Merge the components containing i and j into a single component.
      (iii) Decrement the number of components by 1.
   }
   } // End of **while**.
6. Output the total cost of the edges in T.
7. Output the edges in T, if required.

Name:_____ Date:_____

CSI 403 Homework 3

Algorithm II – Prim's Algorithm: (In the following description, Steps 1, 5 and 6 should not included in measuring the computation time.)

1. Read in the number of nodes (n), the edges and their weights.
2. Mark node 0 as "visited" and nodes 1 through $n-1$ as "unvisited."
3. Initialize T to the empty set. (**Note:** At the end of the algorithm, T will contain all the edges in an MST.)
4. **while** (there are nodes marked "unvisited") {
   (a) Let (i, j) be an edge of lowest cost such that node i is marked "visited" and node j is marked "unvisited."
   (b) Add the edge (i, j) to T.
   (c) Mark node j as "visited."

   } // End of **while**.
5. Output the total cost of the edges in T.
6. Output the edges in T, if required.

Name:_____ Date:_____

CSI 403 Homework 3

Rubric:

		Levels of Performances			
		UNSATISFACTORY 1	DEVELOPING 2	SATISFACTORY 3	EXEMPLARY 4
Performance Dimensions	Performance Indicator #1: The implementation of two different algorithms for the minimum spanning tree problem, as per the specification.	The program does not compile or program fails to read the provided test and produce an output.	Correct implementations of one of the two algorithms that achieve the correct result on the provided test cases.	Correct implementations of the two algorithms that achieve the correct result on the provided test cases.	Correct implementations of the two algorithms that achieve the correct result on all test cases.
	Performance Indicator #2: The complexity analysis of both algorithms.	Incorrect analysis or no analysis.	Correct analysis one of the two algorithms.	Correct analysis of both the algorithms.	Correct analysis of both the algorithms with a comparison with appropriate justification.
	Performance Indicator #3: The execution time analysis (in ms) of both algorithms.	Incorrect analysis or no analysis.	Correct analysis one of the two algorithms.	Correct analysis of both the algorithms.	Correct analysis of both the algorithms with a comparison with appropriate justification.
	Code reads from input files	Code does not use file analysis	Code uses only hard codes files and does not allow for parameters	Code uses file, but there are minor issues with the files	Code uses files and works as specified
	Overall look and feel of the assignment	Submission is not formatted and does not show understanding of the topic	Understanding of the topic is evident but the solution is not well organized	Understanding of the topic exists and there are minor issues with the presentation of the findings	Solution is well formatted and make proper sense for the assignment
	Code is compiled at the command line (-e option is implemented)	Code is not shown using the -e option while running at the command line	Code is shown but does not run the -e option properly from the command line	Code runs from the command line using the (-e) option but there are minor issues	Code runs from the command line using the -e option and there are no issues
	test cases exist	No test cases were shown	One test case was shown	Two or three test cases were shown	All four test cases were shown
	Video	No video is given	Video is given but it does not show the code or the compilation of the code	Video is given, shows the code compiling but doesn't test the entire program	Video is present and shows the required material

Name:_____ Date:_____

CSI 403 Homework 3