SNU CSE: Data Structure | Due Date : 2 pm, June 7 2019

Assignment 5

Lecturer: Prof. Satti

1 - Write a program that generates a random weighted connected graph and saves that to a text file. The file should be constructed in following structure: Line 1: Number of vertices. Line 2: Number of edges. Line 3~: Information of edges. This includes starting vertex, ending vertex, and edge weight. Edges should be ordered in firstly ascending order of starting vertex, secondly ascending order of ending vertex. Each component is separated by a space, not a tab.

The number of vertices are at least $7(2^3-1)$ and at most $1{,}048{,}577(2^{20}+1)$.

Your code should be able to run if invoked `java ds5-step1`. No other filenames are allowed. This step constitutes 4 points in the code part.

<Sample Code Execution>

```
TCS# java ds5-step1 11 graph.txt

 A random graph having 11 vertices is generated.

Number of edge is 13.

Output saved as graph.txt.
```

<Sample File Output>

```
11

13

1 2 4

1 4 2

2 5 8

3 5 6

4 6 5

5 6 2

5 7 6

5 9 4

6 7 7

8 9 4

8 11 5

9 11 8

10 11 3
```

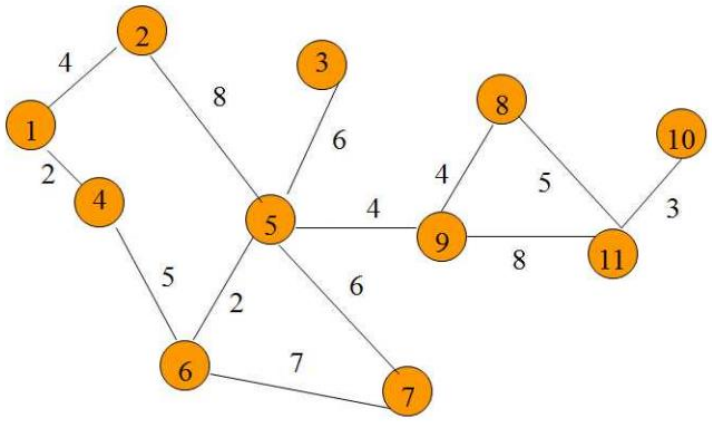

Figure 1 Example graph

2- Write a program that reads the file from [1] and computes a minimum spanning tree using three algorithms: Kruskal's method and Prim's method. Run each algorithm for 10,000 times, and display the total time. The file should be created in following structure:

Line 1: Name of algorithm.

Line 2: Total measured time in seconds. Save the result rounded to hundredths.

Line 3: Number of vertices.

Line 4: Number of edges.

Line 5: Total weight.

Line 6~: Information of edges.

This should be in same format as [1]. Saved data should include results of all three algorithms in same format, in the order of Kruskal andPrim. Your code should be able to run if invoked `java ds5-step2`. This step constitutes 6 points in the code part, 2 points for each algorithm.

<Sample Code Execution>

```
TCS# java ds5-step2 graph.txt result.txt
 Input graph.txt successfully read.
Trying Kruskal's Algorithm... Done.
Total measured time is *.**sec.
Trying Prim's Algorithm... Done.
Total measured time is *.**sec.
Output saved as result.txt.
```

<Sample File Output>

```
TCS# cat result.txt
Kruskal
*.**
10
9
41
1 2 4
1 4 2
(...)
11 8 5
```

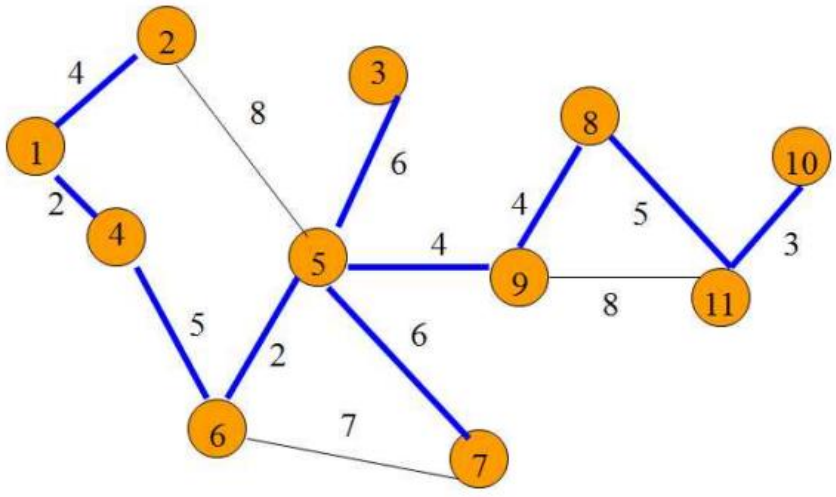

Figure 2 Example minimum spanning tree

```
11 10 3

Prim

(same format as Kruskal)
```

3- Write a program that reads the file similar to that of [1] (but has direction information) and finds the shortest path to all vertices from the vertex 1. Use Dijkstra's algorithm to solve the problem. You should save the solving process in a text file in following structure:

Line 1: Number of vertices.

Line 2: Current number of vertex.

Line 3: Result of d after applying algorithm. Infinite values should be marked as -. Each component is separated by a space, not a tab.

Line 4: Result of p after applying algorithm. Infinite values should be marked as -. Each component is separated by a space, not a tab. Sequence from line 2 to line 4 continues until the last vertex.

Your code should be able to run if invoked `java ds5-step3`.

You don't need to use any of the advanced data structures mentioned on the slides

but not discussed in detail (such as Fibonacci heap or union-find structure) in

implementing the above algorithms.

<Sample Code Execution>

```
TCS# java ds4-step3 graphn.txt dijkstra.txt

 Input graphn.txt successfully read.

Performing Dijkstra's algorithm... Done.

Output saved as dijkstra.txt.
```

<Sample File Output>

```
TCS# cat dijkstra.txt

7

1

d 0 6 2 16 - - 14

p - 1 1 1 - - 1

3

d 0 6 2 16 5 10 14

p  1 1 1 3 3 1 (...)

7
```

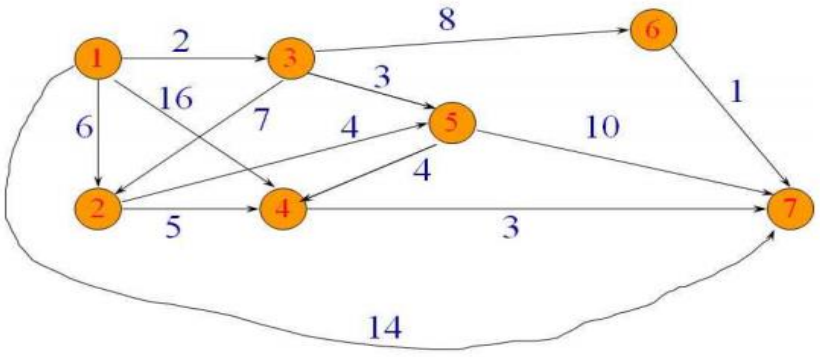

Figure 3 Example graph and shortest path

```
d 0 6 2 9 5 10 11

p  1 1 5 3 3 6
```

<Sample Code Execution>

```
7

13

1 2 6

(...)

3 2 7 //Comment: This is different from the output file of [1]; the ending vertex could be lower than the starting vertex, since this is a directed graph.

(...)

6 7 1
```

Assessment

Your codes will be judged on its correctness. Please make sure that your code runs on Linux terminal. TA will not accept any complain if your code doesn't run on Linux terminal correctly, even if it runs on other IDEs or working environment correctly.

Submission

Compress all your source and output files into a single file as your-full-name_SID.zip.