CNT4007 Computer Network Fundamentals, Spring 2020

Programming Assignment 3

Date assigned:	**Tuesday March 31, 2020**
Date due:	**Wednesday April 15, 2020 (10:40am Eastern)**
	NO LATE submission will be accepted for grading!
How to submit:	Please submit through CANVAS

Introduction

In this programming assignment, we are going to implement the Link-State Routing Algorithm, known as Dijkstra's algorithm. The detail of this algorithm is in Section 4.5 (6th edition) or Section 5.1 (7th edition) of your textbook, so please make sure you understand the material clearly before starting this project. We also follow the terms defined in the textbook, such as D( ), p( ) and N'.

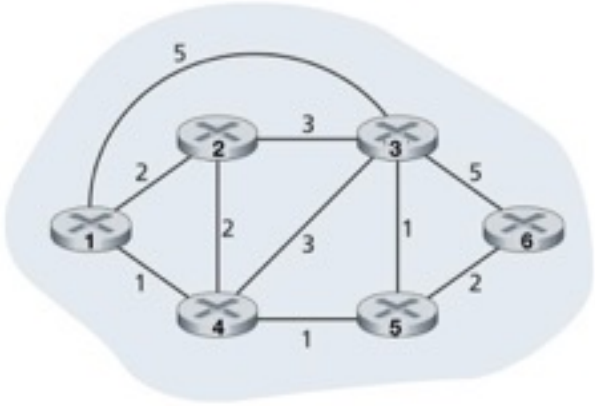
Notice that this assignment is a simulation of the routing algorithm and can be implemented on a standalone computer, so **it does not require socket programming**.

Problem Outline

This assignment requires one single program that will read an input file defining a network topology and runs Dijkstra's algorithm to get the routing table from Node 1 (just like the Node u in the textbook example) to all other nodes.

Network Graph

The network will have many nodes labeled 1, 2, 3, …and the links between those nodes will each have an associated weight (or cost, or distance).



Following is a simple example.
However, a **larger network** will be used when testing your program.

```
1 1 0
1 2 2
1 3 5
1 4 1
2 1 2
2 2 0
2 3 3
2 4 2
3 1 5
3 2 3
3 3 0
```

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

This specifies the exact network in the figure. To be more specific, each line gives a single edge connecting two nodes with the distance between them. So: `3 6 5` gives the edge from node 3 to node 6 weight 5. Note that we will provide you with both the weights from 3 to 6 and from 6 to 3, but those will **always** be the same---in other words the network is *undirected*. If an edge is omitted (e.g. 2 to 5 is not in the list), then the edge does not exist and the two nodes cannot communicate without first traversing another node (in this case: 4 or 3).

Note that in grading we will use a **larger network** (20+ nodes). See appendix for example.

Algorithm and Output

The Dijkstra's Algorithm is described in the textbook. Basically in each iteration we find a node with the least weight and update its route, until eventually all the nodes are included in our routing table , like Table 4.3 (6th edition) or Table 5.1 (7th edition) of the textbook. Please read the book carefully to understand the algorithm.

You will be outputting your results in comma-separated format. When the program starts, you should determine the number of nodes in the network and then print a line as follows:

```
Step,D1,P1,D2,P2,D3,P3,D4,P4,…
```

Where the D(istance) and P(arent) columns repeat for each node in the network. Then, from each node you should use Dijkstra's algorithm to construct the routing table. The output should look like:

```
0,0,1,2,1,5,1,1,1,…
1,0,1,2,1,4,4,1,1,…
```

You may print out whatever you want on lines that **start with** a #. You can use this for debugging your program or including extra output that you find helpful. For example, outputting the line:

```
Path: 1,4,5
```

would cause you to lose points, while the line:

```
# Path: 1,4,5
```

would not.

Running the System

The program should be able to run on CISE machines, read the input file, perform the Dijkstra's algorithm from Node 1 and print out all the steps in detail. After that, the program will exit. Here is an example on how to run the system:

```
java linkstate network.txt
```

Do not hard-code the file name. Your code will be run on multiple different networks, and you will **lose points** if you only read from network.txt.

Important Note

- The network file name should be command line input, do NOT hard code it.
- Print out clearly all the steps, **including** the header and the dashed lines.
- Test your program with **large network**, run it on CISE Linux machines.

Report

Submitted report file should be named "report.pdf" and include the following:

- Your personal information: Full name, UFID, and Email
- How to compile and run your code under which environment.
- Description of your code structure.
- Show some of the execution results.
- Discuss the results you got with your program.
- Explain about the bugs, missing items and limitations if there is any. Honesty is valued here.

Submission Guidelines:

1. The main code file should be named "linkstate.java". You may include other source files as long as the command `javac *.java` compiles the project successfully on storm/thunder.
2. Include the report in .pdf format. Name it "report.pdf".
3. Zip all your files into a packet: Firstname_Lastname_ID.zip
4. Submit to CANVAS before deadline.

Grading Criteria:

Correct Implementation	80%
Report / Code Style	20%
Total:	100%

Appendix

Here is a larger sample network for you to test on. This is not necessarily the same network we will use for grading.

```
1 1 0
1 2 3
1 3 2
2 1 3
2 2 0
2 5 5
2 8 5
2 9 3
3 1 2
3 3 0
3 4 1
3 5 4
3 6 5
4 3 1
4 4 0
4 7 2
5 2 5
5 3 4
5 5 0
5 9 4
5 14 7
6 3 5
6 6 0
6 7 2
6 14 6
7 4 2
7 6 2
7 7 0
7 15 5
8 2 5
8 8 0
8 9 7
8 10 1
9 2 3
9 5 4
9 8 7
9 9 0
9 11 2
9 13 6
10 8 1
10 10 0
10 11 9
10 12 3
11 9 2
11 10 9
11 11 0
11 19 6
11 20 4
12 10 3
```

12 12 0
12 20 5
13 9 6
13 13 0
13 14 4
13 17 3
13 19 5
14 5 7
14 6 6
14 13 4
14 14 0
14 15 8
14 17 7
15 7 5
15 14 8
15 15 0
15 16 2
15 18 6
16 15 2
16 16 0
16 18 3
17 13 3
17 14 7
17 17 0
17 18 8
18 15 6
18 16 3
18 17 8
18 18 0
18 23 5
18 26 3
19 11 6
19 13 5
19 19 0
19 22 8
19 23 2
20 11 4
20 12 5
20 20 0
20 21 4
20 22 7
21 20 4
21 21 0
21 24 9
22 19 8
22 20 7
22 22 0
22 24 9
23 18 5
23 19 2
23 23 0
23 24 4
23 25 5
24 21 9

```
24 22 9
24 23 4
24 24 0
24 25 2
25 23 5
25 24 2
25 25 0
25 26 9
26 18 3
26 25 9
26 26 0
```