Homework 9 (100 points): Dijkstra's Algorithm, Due Date*: 11:59pm 04/29/2020, Cutoff Date**: 11:59pm 05/01/2020

Submission: (1) Upload all source code (.java) files to both Blackboard and the Virtual Server cs3700a, AND (2) set up and test java programs (.class files) on cs3700a (See Part II for details)
****Submission will NOT be accepted after the cutoff deadline**

An option of peer programming: You may choose to work on this assignment individually or in a team of two students. If you choose to work in a ***team of two students***, you **must** 1) **add** both team members' first and last names as the **comments** on Blackboard when submitting the .java files (both team members are required to submit the same .java files on Blackboard), and 2) put a ***team.txt*** file including both team members' names under your HW9/ on cs3700a (both team members are required to complete Task II under both home directories on cs3700a). For grading, I will randomly pick the submission in one of the two *home directories* of the team members on **cs3700a** and **cs3700b**, and then give both team members the same grade. If the team information is *missing* on Blackboard or cs3700a, two or more submissions with similar source codes with just variable name/comment changes will be considered to be a violation of the Integrity described in the course policies and both or all will be graded as 0.

Grading: Your programs will be graded via testing and points are associated with how much task it can complete. ~~A program that cannot be compiled or crashes while running will receive up to 5% of the total points.~~ ~~A submission of .java files that are similar to any online Java program with only variable name changes will receive 0% of the total points.~~

Programming in Java is highly recommended and NO library function(s) related to the algorithm/protocol may be used, since all requirements in this assignment are guaranteed to be supported in Java. If you choose to use any other language such as Python or C/C++, it is YOUR responsibility, before the cutoff deadline, to (2) set up the compiling and running environment on cs3700a, (2) make sure that I can run/test your programs in your home directory on cs3700a, and (3) provide a README file under HW9/ on cs3700a to include the commands that I need to use.

Part I (90%): Write a **Java program** to build up the forwarding table at **source router V0** using the Dijkstra's algorithm.

- The routers in the network are labeled as **V0**, **V1**, **V2**, …, etc (for display, such labels are **required** to use, please do NOT use 0, 1, 2, 3, … etc. However, it is perfectly okay for you to use indices 0, 1, …, and *n* – 1 internally in your program.)
- Your routing program needs to
  1. Display a prompt message to ask the user to input the total number of routers, ***n***, in the network. Validate ***n*** to make sure that it is greater than or equal to 2.
  2. Use a topo.txt file that contains costs of all links. Each line in this file provides the cost between a pair of routers as below, where **tab ('\t')** is used to separate the numbers in each line.
     ```
     <# of one router> <# of the other router> <link cost between these two routers>
     ...               ...                     ...
     ```
     where the first and the second numbers in each row need to be validated to be between 0 and ***n - 1***, the third number needs to be validated to be positive. For example, for the link (V0, V3) whose cost is 10, **only ONE of the following two lines needs to be included in the topo.txt file**.
     ```
     0    3    10    // only ONE of this or the next line needs to be included in the topo.txt file
     3    0    10    // only ONE of this or the above line needs to be included in the topo.txt file
     ```
     This topo.txt file can locate in the same directory where this program runs such that a path is not needed. Display a message saying that in which row the first invalid number is detected, close the txt file, and KEEP asking for the name of the cost input file until all numbers are checked to be valid. Record the cost information in the Cost matrix.
  3. Implement the Dijsktra's algorithm to build up the shortest-path tree rooted at source router V_0. As the intermediate results, at the end of **Initialization** and each iteration of the **Loop**, display
     - The set **N'**
     - The set **Y'**
     - The distance vector **D**, i.e., D(i) for each i between 1 and ***n* - 1**
     - The predecessor vector **P**, i.e., p(i) for each i between 1 and ***n* - 1**
  4. Use the shortest-path tree resulted from the Dijsktra's algorithm to build up the forwarding table for router V_0. Display the forwarding table in the following format:

Destination	Link
V1	(V0, …)
V2	(V0, …)
…	
Vn-1	(V0, …)

Part II (10%): Test your programs on the Virtual Server cs3700a.msudenver.edu

(more on next page!)

Warning: to complete this part, especially when you work at home, you must first (1) **connect to GlobalProtect** using your NetID account (please read "**how to connect to GlobalProtect …**" at https://msudenver.edu/vpn/); then (2) **connect to the virtual servers cs3700a** using ***sftp*** and ***ssh*** command on MAC/Linux or ***PUTTY*** and ***PSFTP*** on Windows.

ITS only supports GlobalProtect on MAC and Windows machines. If your home computer has a different OS, it is your responsibility to figure out how to connect to cs3700a for programming assignments and submit your work by the cutoff deadline. Such issues cannot be used as an excuse to request any extension.

1. MAKE a directory "**HW9**" under your home directory on **cs3700a**.msdenver.edu
2. UPLOAD and COMPILE ***your program*** under "**HW9**" on **cs3700a**.msdenver.edu
3. TEST ***your program*** running on **cs3700a**.msudenver.edu
4. SAVE a file named ***testResultsClient.txt*** under "**HW9**" on **cs3700a**.msudenver.edu, which captures the outputs of your program when you test it. You can use the following command to redirect the standard output (stdout) and the standard error (stderr) to a **file** on UNIX, Linux, or Mac, and view the contents of the file

```
java prog_name_args | tee testResultsClient.txt //copy stdout to the .txt file
      //if you want, you may also use "script" command instead of the "tee" command
      //to write both stdin and stdout into testResultsClient.txt while testing your
      //client program on cs3700a.  For how to use "script", see
      // https://www.geeksforgeeks.org/script-command-in-linux-with-examples/
      //or Google "script command in Linux" if the above link is broken.
cat file-name     //display the file's contents.
```