Homework 10 (100 points) Dijkstra's algorithm & Forwarding Table

Due Date*: 11:59pm 04/27/2024, Cutoff Deadline: 11:59pm 04/29/2024**

****Submission will NOT be accepted after the cutoff deadline**

Submission:
1) upload and submit all your **.java file(s)** for both programs in **Canvas** (email is NOT accepted). Only ONE attempt is allowed for each student. NO Paper/Hardcopy or Email submission please!
2) upload, compile, test, and set up your **program** (.java and .class files) on **cs3700a** under **HW10**, which will be used by the instructor for ***testing*** and ***grading*** your program on **cs3700a**. See **task II** for more details.
3) the file named "testResults.txt" under **HW10** on **cs3700a**. See **task II** for more details.

Recommended References:
[1] The lectures/classes ***before or on*** **the Due Date.**
[2] **Lab 1** under a "Weekly Learning Activities and Materials" module in Canvas

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** in Canvas when submitting the .java files (***both team members are required to submit*** the same .java files in Canvas), and 2) put a ***team.txt*** file including both team members' names under your **HW10/** on **cs3700a** (***both team members are required to complete Task II under their own home directories on cs3700a***). For grading, I will ***randomly*** pick the submission in ***one team member's home directory*** on **cs3700a**, and then give both team members the ***same grade***. If the team information is *missing* in Canvas or **cs3700a**, or two or more submissions from students *not all* working in a team contain similar source code with just variable name/comment changes, it will be a violation of the Academic Integrity and students involved will receive **0** as the grade. Here are a few reminders on "What IS NOT Allowed" to hopefully avoid such violation (see Course Policies for more details).
1. Other than the example programs provided by the instructor to all students fairly, you may NOT look at code created by another person or provided by any online/offline resource (including but not limited to tutors and online/offline tutoring service) for any programming assignment/project until after you have completed and submitted the assignment yourself. You may NOT show or share your code to/with anyone other than the instructor. You may NOT ask anyone else to debug your code.
2. Students absolutely may NOT turn in someone else's code or solution with simple cosmetic changes (say, changed variable names or changed order of statements) to any homework, project, or exam.
3. It is strictly forbidden for any student to refer to code or solutions from previous offerings of this course or from any online/offline resource including but not limited to tutors and tutoring services unless it is provided by the instructor to all students fairly.

Grading: Your programs will be graded ***via*** **testing under your home directory on cs3700a** and points are **completely associated with how much task your programs can complete while running on cs3700a**.
1. You are highly recommended **NOT to use any IDE**, *online* or *on your computer*, to compile, debug, and test your programs. Instead, you should compile, debug, and test your programs in the command-line Linux environment on **cs3700a**, which is part of the learning objectives in this upper-division CS course. Being able to make your programs only work in an IDE (online or on your computer) but NOT on **cs3700a** ***cannot*** be used as an excuse for re-grading or more points either.
2. ALL the ***output messages or files*** that are REQUIRED to be ***displayed or created*** by your program in Task I are the only way to demonstrate how much task your program can complete. So, whether your program can display those messages correctly with correct information is critical for your program to earn points for the work it may complete.
3. It is also extremely important for your program to be able to correctly READ and TAKE the information from the ***input file(s)*** or the ***user inputs*** that follow the format REQUIRED in this programming assignment, instead of some different format created by yourself. Otherwise, your program may crash or get incorrect information from the input file(s) or user inputs that follow the required format.
4. Grading is NOT based on reading/reviewing your source code although the source code will be used for plagiarism check. A program that cannot be compiled or crashes while running on **cs3700a** will receive up to 5% of the total points. A submission of source code files that are similar to any online source code with simply cosmetic changes will receive **0%** of the total points.

Programming in Java is highly recommended, 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 (1) 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 **HW10** on **cs3700a** to include the commands that I need to use.

Task I (90%): Write a **Java program** to build up the shortest-path tree using the **Dijkstra's algorithm** on the ***source*** **router V0** and then use the result of the Dijkstra's algorithm to build up the **forwarding table** at **source router V0**.
- 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,** ONE line for every link**. If there is NO link between two routers, i.e., the link cost between these two routers is infinite, NO line is included in the topo.txt file for these two routers**. 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, …)

Task II (10%): Test your programs on cs3700a.msudenver.edu

Warning: to complete this part, especially when you work at home, you must first (1) **connect to GlobalProtect** using your NetID account; then (2) **connect to the virtual servers cs3700a** using ***sftp*** and ***ssh*** commands on a MAC computer or using software like ***PUTTY*** and ***PSFTP*** on a Windows machine. (See Lab 1 on ***how to***.)
ITS only officially 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 and cs3700b for programming assignments and submit your work by the cutoff deadline. Such issues cannot be used as an excuse to request any extension.

1. CREATE a directory "**HW10**" under your *home directory* on **cs3700a**.msdenver.edu.
2. UPLOAD and COMPILE your program under "**HW10**" on **cs3700a**.
3. TEST/DEBUG your program while it runs on **cs3700a**.
4. SAVE a file named ***testResults.txt*** under "**HW10**" on **cs3700a**, which captures the output messages 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 testResults.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 testResults.txt while testing your
         // 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 a text file's contents.
   ```
5. If you work in a team of two students, you must put a ***team.txt*** file including both team members' names under "**HW10/**" on **cs3700a** (***both team members are required to complete Task II under their own home directories on cs3700a***). For testing and grading, I will ***randomly pick*** the submission in one of those two *home directories*, and then give both team members the ***same*** grade for Task I.