Shubham Arya 1001650536

**CSE 4344 – 900 Project 2 README**

**Instructions to run the program:**

Download the zip file and unzip it. Open the terminal of your computer. Locate to the directory where this folder is saved on the terminal using ‘cd’ commands. Once you have reached the folder with the .java file of this assignment in the terminal, type the following commands:

*Javac Simulator.java*

*java Simulator*

This will run the program. It will ask the user to enter name of the file name that exists in that folder. Enter the name of the file and it will generate the initial distance vector table at the zeroth cycle. If the file does not exist, the program will simply end. A menu will appear, choose 1, 2, 3 or any other key to perform a function as mentioned in the menu.

**Instructions to generate .JAR file:**

In the terminal of your computer, use cd to reach the directory of this folder. Then in the terminal enter javac Simulator.java and then jar -cvf Simulator.jar Simulator.java. This will create a Simulator.jar file.

**Output:**

From the first step, we can see the nodes that are directly reachable to the other nodes. For instance, for node 1, we can see that node 1 can only reach node 1, 2 and 5 as they are the only rows present over there. Nodes that cannot reach other will have infinity distance between them. But from the assignment’s requirement, we have taken infinity to be 16. The nodes that have a link cost of 0 represents that they are the links that are directed to themselves. Since the shortest path for a node to reach itself is 0, we represent a 0 there.

After the cycle 0 is completed, we can either select 1 to run each step in the algorithm or we can select 2 to reach the final stable state directly or we can select 3 to change the link cost of the nodes.

**Graphical user interface, text

Description automatically generated**

On selecting 1, it will go through all the iterations from cycle 1 to cycle 4. In the screenshot below, only the final cycle is represented. But in the actual program, it should ask the user to select 1 until the final stable state is reached. This will also be the screenshot if the user selects 2 in the menu. The only difference here will be that on selecting 2, the program will directly arrive at the final cycle without going through the intermediate cycles. One thing to note in the below screenshot is that there are no infinite values (16) in the distance vector table. This is because every node is reachable to every other node through some intermediate node. So node 1 was not directly reachable to node 4 at the start, after reaching a stable state, node 1 can reach node 4 via node 5 with a link cost of 3.

**A picture containing graphical user interface

Description automatically generated**

Next, we can select 3 to change the link cost for specific link. Suppose we want to change the link cost between node 1 and node 2 to 1 from 7 in the original file. We can select 3 and it will ask us to enter the nodes for the link we want to change and the link cost. So our input will be of the form: 1 2 1. 1 is node 1, 2 is the second node and 1 is the link cost between node 1 and node 2. The output should indicate the link cost between 1 and 2 which was originally 7, to 1 now. This change in link will also lead to changes in values for other nodes too.

**A picture containing calendar

Description automatically generated**

For this project, I used some sources that are referred below. Some of these sources were used to get the logic and the algorithm needed for this code. Some functions were used from these sources directly or to crosscheck my code.

**Reference:**

1. <https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/>
2. <https://www.geeksforgeeks.org/distance-vector-routing-dvr-protocol/>
3. <http://campuscoke.blogspot.com/2015/01/distance-vector-routing-dvr-algorithm.html>
4. <https://www.javatpoint.com/distance-vector-routing-algorithm>
5. <https://ankurm.com/implementation-of-distance-vector-routing-dvr-algorithm-in-c/>
6. <https://github.com/BrandonChase/cse4344-ass2>
7. <https://www.gatevidyalay.com/distance-vector-routing-routing-algorithms/>