CS 542 Project

Simulating Link State Routing Algorithm

Rohan Digambar Gawade

A20379951

Seat No 29

INDEX

|  |  |
| --- | --- |
| **Topic** | **Page Number** |
| Link State Routing Protocol | 3 |
| Project Design | 3-4 |
| Dijkstra Algorithm | 5 |
| Test Report | 6-10 |
| Execution Instructions | 11 |

Link State Routing Protocol

* The link state routing protocol is one of the two protocols used in packet switching network.
* Every router in the network performs link state routing algorithm to compute the forwarding table.
* The link state routing algorithm can be applied to the network only if the network topology and the cost of all the links are known.
* Dijkstra’s algorithm computes the minimum path cost from a source node to every other node in the topology.
* The property of the Dijkstra’s is that its iterative and after the kth iteration of the algorithm, the least cost path are found to the k destination nodes.

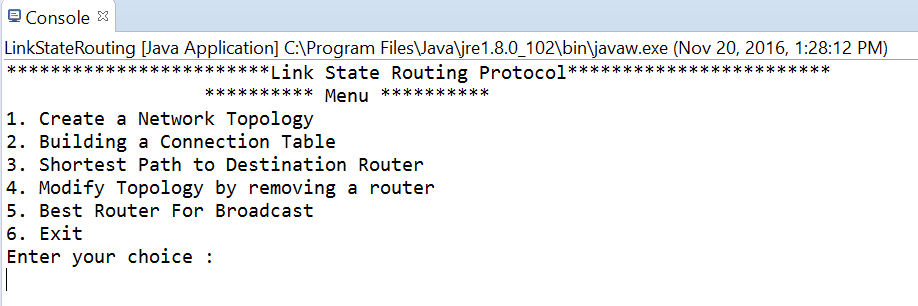
Project Design

The project consists of 5 java files which are used to solve the problem statement.

1. LinkStateRouting.java
2. CreateTopology.java
3. ConnectionTable.java
4. DijkstraShortestPath.java
5. ModifyTopology.java
6. **LinkStateRouting.java**

* This java code is used to display the menu from which the user selects the desire task to be done.
* When the user enters the choice, the respective function is called.

**Output**

****

1. **CreateTopology.java**

* When the user selects choice 1 from the menu options, the CreateTopology() constructor is called, which reads the file containing the network topology and stores the cost information in routerAdjMatrix.
* It also contains displayMatrix() function which is used to display the topology.

1. **ConnectionTable.java**

* The ConnectionTable file consist of three parts

1. ConnectionTable() constructor

* This constructor initializes all the variables used for computing the distance, interface Hop, visited router.
* It changes the value in adjacency matrix from 0 or -1 to 999.

1. getBestRouter() function.

* This function is used to calculate the Best Router for broadcasting, which is option 5

1. getInterfaceConnection()

* This function gets the source router and calculates the forwarding table for the source router to destination router

1. **DijkstraShortestPath.java**

* Dijsktra algorithm is implemented in this file.
* It takes the topology, source and destination as input.
* It calculates the minimum path from source to destination router and is displayed.

1. **ModifyTopology.java**

* This class contains ModifyTopology() which is used to modify the original network topology.
* The user can shutdown the router by selecting choice 5 and entering the router number.
* The corresponding value in the matrix for this router is replaced with -1 indicating the router is down.

Additional Features

* My project gives an option for the user to select whether to display the minimum distance along with the interface hop. If the user selects Y|y then the connection table for source router is displayed along with the distance from source router to other routers.
* When the topology is modified by removing the router, the new shortest path from source to destination is computed and displayed, If source and destination are already provided.
* The project runs fine for both directed as well as undirected graph.

**Dijkstra Algorithm.**

* Dijkstra Algorithm solves the single source shortest paths problem on weighted, directed graph.
* We use this algorithm to find the distance matrix which contains cost from source router to every other router in the graph.
* Its an Link State Routing Algorithm which is used to create the forwarding table by each router in the packet switch network.

**Pseudocode**

* 1 Initialization:
* 2 visitedNode[u] = 1
* 3 for all nodes v
* 4 if v is a neighbor of u
* 5 then D(v) = c(u,v)
* 6 else D(v) = ∞
* 7
* 8 Loop
* 9 find w such that visitedNode[w] !=1 and D(w) is a minimum
* 10 add w to visited list by making visitedNode[w] = 1
* 11 update D(v) for each neighbor v of w and not in N’:
* 12 D(v) = min( D(v), D(w) + c(w,v) )
* 13 /\* new cost to v is either old cost to v or known
* 14 least path cost to w plus cost from w to v \*/
* 15 until all nodes are visited

**Description:**

* The algorithm takes Source node u as input.
* In the Initialization step, the source node is marked as visited indicated by visitedNode[u] = 1.
* All the neighbors of u are processed and if there exists a cost between u and neighbor v then it is stored in D(v) else D(v) = ∞
* From step 8 to 15 the Dijkstra algorithm finds the shortest path from source node u to all other nodes.
* Node w is selected such that visitedNode[w]!=1, that is, it is not visited and has a minimum distance D(w).
* After the minimum distance node is found it is added to visited list by making visitedNode[w]=1’
* For each neighbor v of w and not visited find the minimum distance from u. The new cost will be D(u,v) or D(w)+D(w,v)
* Iterate until all the nodes are visited.

**Test Report**

In order to run the program successfully placed the topology.txt file(any input file) in the same folder as the jar file.

The network topology inputted through file is as follows

0 3 5 7 -1

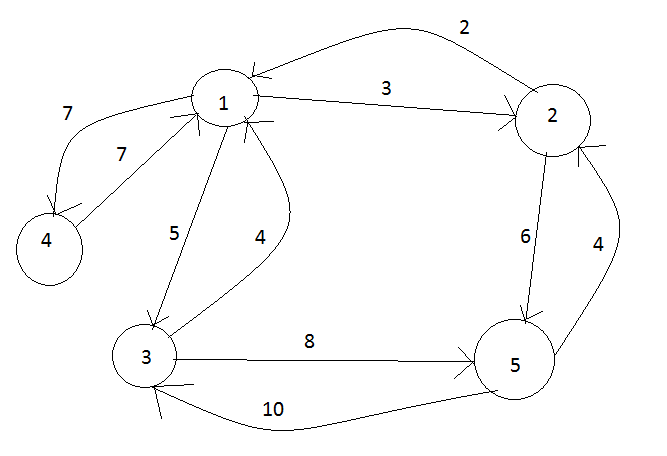
2 0 -1 -1 6

4 -1 0 -1 8

7 -1 -1 0 -1

-1 4 10 -1 0

**Graph**

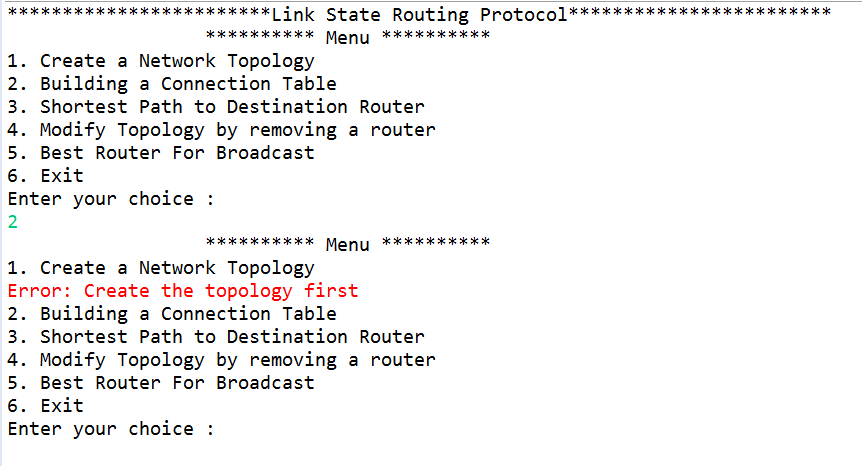


**Test Case 1:**

If the topology is not created that is if choice 1 is not selected prior to other selection an error is thrown.

**Expected:** Error

**Output** : Error: Create the topology first

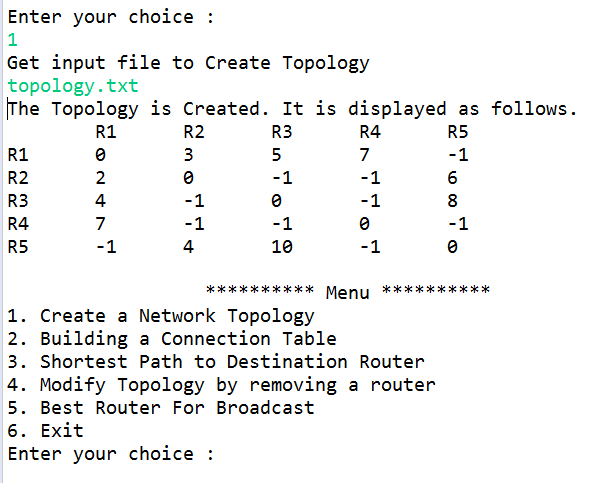


**Test Case 2**

**Input** choice 1, n\*n matrix topology is read through file. Any length topology can be read, only we need to enter the name of the file.

**Expectation:**Topology created successfully.

**Output: As Expected**

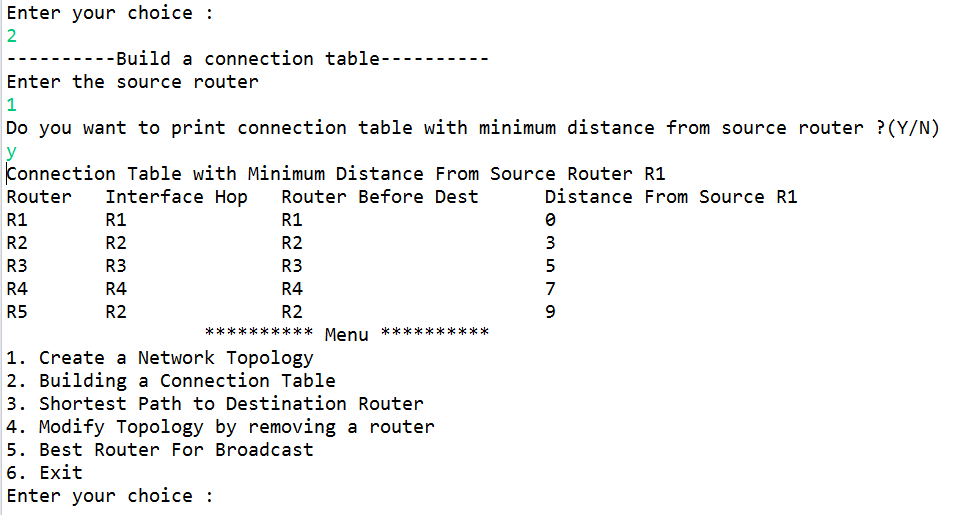


**Test Case 3:**

**Input**: Enter choice 3, enter the source router 1 and show distance = y

**Expected output:** Interface Hop from source router along with minimum cost/distance.

**Output: As Expected**

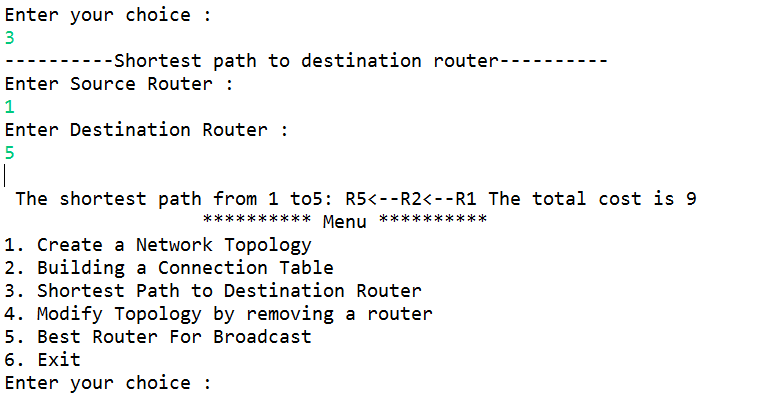
****

**Test Case 4**

**Input:** Enter choice 4, source = 1 destination = 5

**Expected:**  Path = R5 🡨 R2 🡨 R1

**Output: As Expected**

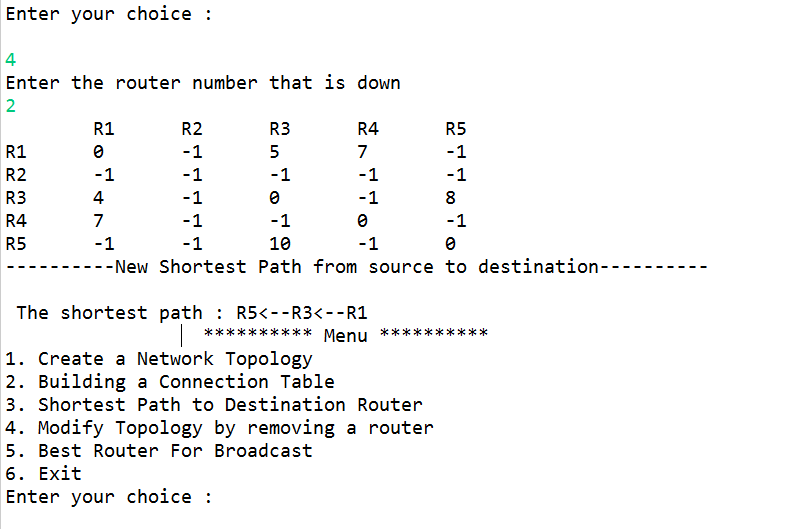


**Test Case 5**

**Input:** Enter 4 to Modify the topology. Enter router 2 to remove the router.

**Expected:** New topology is created and the shortest path for the previous source and destination value is computed which has new route.

**Output:** As expected. Router 3 is now selected as Router 2 is down.

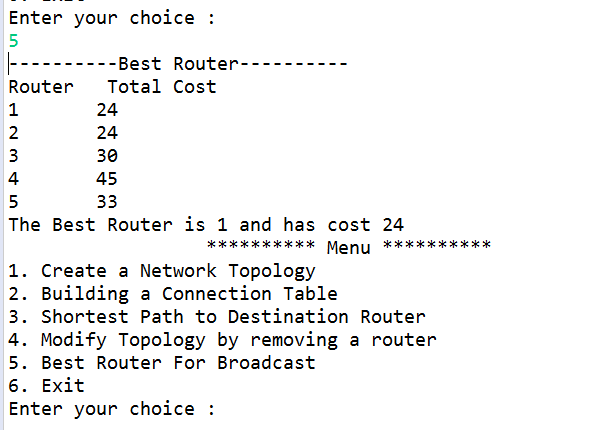


**Test Case 6**

**Input:** Enter choice 6(on original input)**.**

**Expected:** Router with best total minimum cost to all other router is displayed

**Output:** As expected



**Instructions on how to execute the project**

The zip folder contains source file, sample text file(topology.txt) which is input, CN\_Project.jar file, cnproject.bat file PPT and Design and Test Case document.

1. Keep the .jar file, .bat file and input text file in single folder.
2. Just double click the .bat file and you will be good to go.

If the .bat file fails to work.

* Place the jar and txt file in same folder.
* Execute command **java -jar CN\_Project.jar** in command prompt.