PROJECT TITLE:

IMPLEMENTATION OF LINK-STATE ROUTING PROTOCOL

PROJECT OPERATIONS MANUAL

Name: Karthik Krishnamurthy

CWID: A20344464

Subj: CS-542 Computer Networks-1

Section: 02, Main Campus

Date: 11/21/2015

IMPLEMENTATION OF LINK – STATE ROUTING PROTOCOL

CS542 Term Project (Fall 2015)

Krishnamurthy, Karthik
[A20344464]
Main Campus : Section 02

INTRODUCTION

Link State is a routing protocol which is used in packet switching networks for computer communication. In the network that uses link state, every node will perform link - state; they construct a map of the connection between nodes in network by showing which nodes connected with which other nodes. After that each nodes can independently calculate the best path from it to other nodes in network.

Dijkstra's algorithm is called as single source shortest path Algorithm. It computes the least-cost path from one node (Source) to all other nodes in the network. Dijkstra's algorithm is iterative and has the property that after the kth iteration of the algorithm, the least-cost paths are known k-destination nodes, and among the least-cost paths to all destination nodes, these k-paths will have the k smallest costs.

In this project a program is developed to implement Link - State routing protocol. The program should simulate the process of generating routing tables for each router in a given network and compute optimal path with least cost between any two particular given routers. The Topology modification, by making a router down in the network is also implemented. Dijkstra's algorithm will be used to calculate the direction as well as the shortest path between two routers.

Java is used to implement the Dijkstra's algorithm. In the code, several two - dimensional arrays are used to store original routing table, the distance between routers, values of distance during shortest path calculation, final table after calculation. Some integer variables are used to keep a count on the routers. The program interface display a menu to user who can Create a Network Topology, by choosing to input matrix of routing table from file, the program also help users calculate the shortest path between any couple of

routers and display it to on the screen. Simple Command Line Interface is developed to calculate shortest path using Dijkstra's Algorithm.

Routing protocol and its classification:

Routing protocol is response to the demand for dynamic routing tables in network. A routing protocol is a combination of rules and procedures that let routers in the internet inform other routers of change. It helps routers in the network know about the internet of their neighbour by sharing information.

In routing protocol we have interior protocol which handles intra-domain routing and exterior protocol which handles inter-domain routing protocol. Link-state is one of the intra-domain routing protocol.

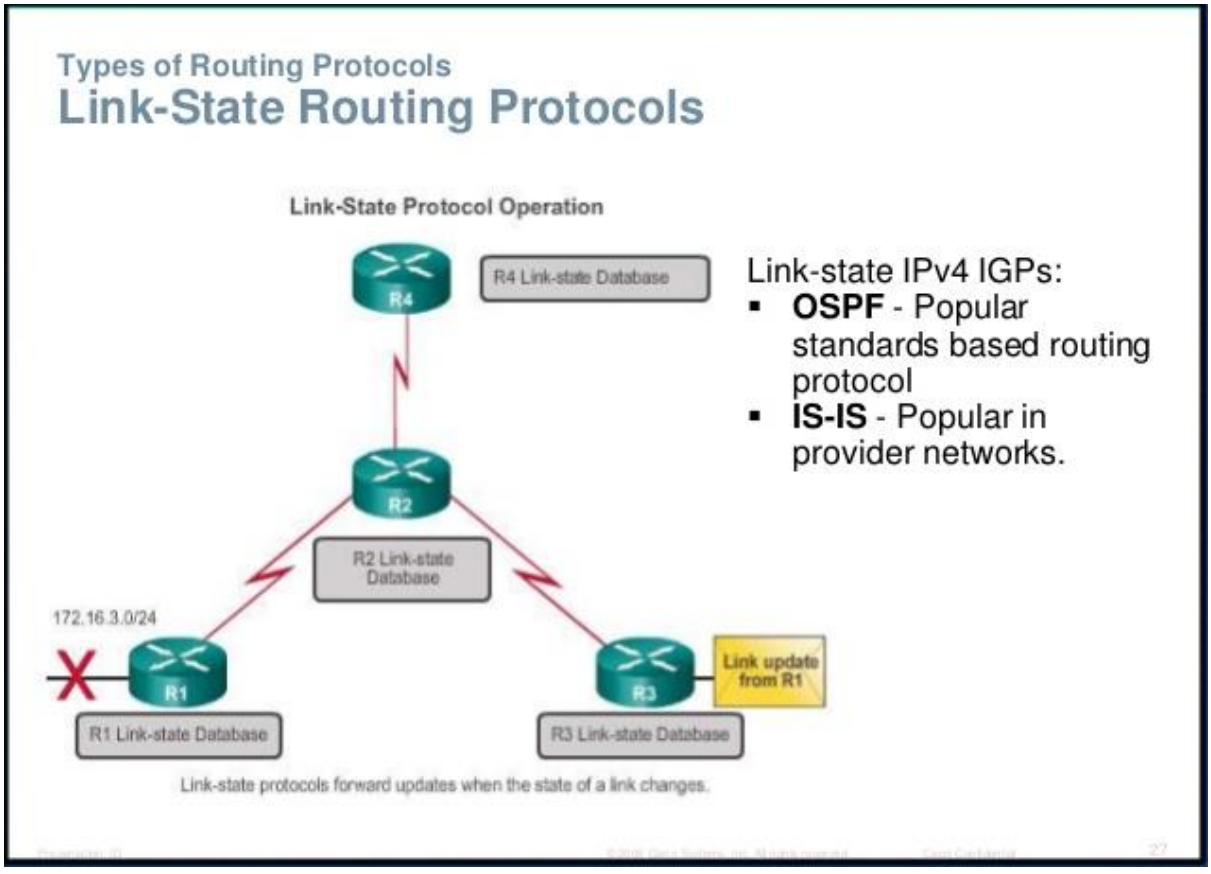


Figure: Representation of Link – State Routing Protocol

ALGORITHM DESCRIPTION:

The method to calculate the optimal routing table and thus the shortest path using Dijkstra's algorithm and link state in project can be explained as follows:

We have a matrix of routing table that present the original value of metric between a router and its neighbour. At very first stage, each router does not know about connectivity of all others routers, each one just knows about its neighbour and its link, we use non-negative integer to present cost of a link and -1 for indirect connection. Router will uses the link-state routing table through four steps:

(i) Creating the state of link by each nodes,
(ii) Disseminating the link-state packet to other nodes,
(iii) Formatting shortest path for each node,
(iv) Calculating the routing table based on shortest path tree.

After each node has connectivity of other nodes it still does not have shortest path from itself to other one. At this time we use the Dijkstra's algorithm to calculate the shortest path:

(i) From table of n routers we will have n shortest path tree, each node will become the root in its shortest path tree.
(ii) Set the shortest path distance for all root neighbours to the cost between root and neighbour.
(iii) We continue search the nodes in the path select one with minimum shortest distances and add to the path.
(iv) Use the operation: $D_i = minimum(D_i, D_j + c_{i,j})$, we repeat above step until all nodes are added to the path.
(v) While having the shortest path tree we can create the routing table, we use 0 to represent the distance from root to root i.e. for same router say R1 → R1 has cost = 0. On the other hand it has positive cost values for two different nodes.

==========

CODE

//Dijkstra.java

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.Scanner;
import javax.swing.JFileChooser;

//Dijkstra.java
public class Dijkstra {
static int Max_rooters;
static int DistGrph[][] = null;                    //Initialization and Declarations of
variables
static int DistRow[] = null;
static int source = -1,destination = -1;
class ConTableEntry {
boolean flg;
int length;
int []ids;
int depth;
}

static ConTableEntry []conTable = null;          //Connection Table initialization

//Main Method

public static void main(String[] args) {

Dijkstra psFrame = new Dijkstra();

}

public int nodeid(int in){

return (in+1);
}
```

/* ReadTextfileToBuildGraph Method */

```
public void ReadTextfileToBuildGraph() {

try

{
```

```
System.out.println("Enter Text File name to read Network Topology Matrix:");
Scanner in1 = new Scanner(System.in);                    //Takes from the user the
input file name
String fiename = in1.nextLine();

FileReader fr = new FileReader(fiename);          //Creation of FileReader object
String val=new String();

//Creation of BufferedReader Object

BufferedReader br = new BufferedReader(fr);

String[] temp;

val=br.readLine();

temp = val.split(" ");       //to find size of the matrics or number of rooters

DistGrph = new int [temp.length][temp.length];

//Close the BufferedReader Object

br.close();

//Close FileReader Object

fr.close();

System.out.println("<=======:Graph Size Read:=======>"+temp.length);

fr = new FileReader(fiename);//Read the content of file into the FileReader
object

val=new String();

//Creation of BufferedReader Object

br = new BufferedReader(fr);

int i = 0;

while((val=br.readLine())!=null)

{

String[] temp1;

temp1 = val.split(" ");           //Splitting with space as a Delimiter
```

```
for (int dist =0; dist < temp1.length; dist++ ){

DistGrph[i][dist] =  Integer.parseInt(temp1[dist]);
}
i++;
}
//Close the BufferedReader Object
br.close();
//Close FileReader Object
fr.close();
System.out.println("<=======:Graph Table Initialized:=======>");

Max_rooters = i;

String imageName = "%3d" ;              //Formatting to display the Content in the
Matrix format
System.out.println();
System.out.print("ID|");

for (int j =0; j < Max_rooters; j++ ){
System.out.print(String.format( imageName,nodeid(j)));

}
System.out.println();

System.out.println("----------------------------------------------------------------");

for (int j =0; j < Max_rooters; j++ ){

imageName = "%2d|" ;            //Formatting to display the Content in the Matrix
format

System.out.print(String.format( imageName,nodeid(j)));

imageName = "%3d" ;

for (int k =0; k < Max_rooters; k++ )
System.out.print( String.format( imageName, DistGrph[j][k]));
//System.out.print(" "+DistGrph[j][k] +"  ");
System.out.println();
}
System.out.println("----------------------------------------------------------------");
//}
} catch(Exception e){
//e.printStackTrace();

System.out.println("File Not Found, Please Enter a Valid File !");       //To
Handle File Not Found Exception
```

```
            }
        }
        /* ComputeConnectionTable Method*/
        public void ComputeConnectionTabel(){
        if (DistGrph[source][source] == 0) {
        conTable = new ConTableEntry[Max_rooters];          //Creates a ConTable
Object with the Number of Rooters
        for (int j = 0;j<Max_rooters;j++) {
        ConTableEntry ce = new ConTableEntry();
        ce.flg = true;
        ce.length = -1;                    //Initializing the variables to start processing
        ce.ids = new int[Max_rooters];
        ce.ids[0] = source ;
        ce.depth = 1;
        for (int i = 1;i<Max_rooters;i++) ce.ids[i] = -1;
        conTable[j] = ce;
        }
        //initializing source in working row
        int tmpsorce = source;
        conTable[tmpsorce].length = 0;
        conTable[tmpsorce].ids[0]=source;
        conTable[tmpsorce].flg = false;
        //int nodedepth = 1;

        for (int loopcnt = 0 ; loopcnt<Max_rooters; loopcnt++) {
```

```
for (int k = 0 ;  k< Max_rooters ; k++)

{

if (conTable[k].flg)

{

//System.out.println("k andDistGrph[tmpsorce][k] "+k+"
"+DistGrph[tmpsorce][k]);

if (DistGrph[tmpsorce][k]!= -1){



if ((conTable[k].length != -1) ) {

//System.out.println("k andDistGrph[tmpsorce][k] "+k+"
"+DistGrph[tmpsorce][k]);

//smaller ( selected node length+ tableentry,previous entry path)

if (conTable[k].length > conTable[tmpsorce].length + DistGrph[tmpsorce][k]) {

conTable[k].length = conTable[tmpsorce].length + DistGrph[tmpsorce][k];

for (int idx = 0; idx< conTable[tmpsorce].depth ;idx ++)

conTable[k].ids[idx] = conTable[tmpsorce].ids[idx];

conTable[k].depth = conTable[tmpsorce].depth ;

conTable[k].ids[conTable[k].depth] = k;

conTable[k].depth++;

//conTable[k].ids[conTable[k].depth] = -1;

}

}

else

{  //selected node length is added to length table entry for new length

//System.out.println("k  "+k+"  "+DistGrph[tmpsorce][k]);
```

```
conTable[k].length = conTable[tmpsorce].length + DistGrph[tmpsorce][k];

//System.out.println();


for (int idx = 0; idx< conTable[tmpsorce].depth ;idx ++){

//System.out.println("conTable[tmpsorce].ids[idx]"+conTable[tmpsorce].ids[idx
]+"id"+ idx);

conTable[k].ids[idx] = conTable[tmpsorce].ids[idx];

}
conTable[k].depth = conTable[tmpsorce].depth ;

conTable[k].ids[conTable[k].depth] = k;

conTable[k].depth++;

//conTable[k].ids[conTable[k].depth] = -1;

//System.out.println("finish");

}
}
}
}
for (int i = 0; i<Max_rooters; i++){

//System.out.print(" "+conTable[i].length);

}

System.out.println();

//initalize smallest dist
int small = 0;

int indx_small = 0;

for (int i = 0; i<Max_rooters; i++){

if (conTable[i].flg){

if(conTable[i].length !=-1 ){

small = conTable[i].length;

indx_small = i;
```

```
break;
}
}
}
//find source for next iteration
for (int i = 0; i<Max_rooters; i++){
//System.out.println("i and conTable[k].flg "+i +" " +conTable[i].flg);
if (conTable[i].flg){
if(conTable[i].length != -1 ){
if (small > conTable[i].length){
small = conTable[i].length;
indx_small = i;
}
}
}
}
}
//System.out.println("indx :"+indx_small + " Val:"+small );
tmpsorce = indx_small;
conTable[tmpsorce].flg = false;
}
System.out.println("Router [" + nodeid(source) + "] "+ "Connection Table:");
System.out.println("============================");
System.out.println("Destination Interface"); //Printing the
Connection Table
for (int i = 0; i<Max_rooters; i++){
```

```
String tmp = String.valueOf(conTable[i].ids[1]+1);

if (conTable[i].ids[1] == -1) tmp = "-1";              //Check the Router ID if it is -1

if (i == source) tmp = "-";                          //Source to Source Router will be "-"

System.out.print("     "+  nodeid(i) + "             "+ tmp);

System.out.println();

}

}

else {

System.out.println("Router [" + nodeid(source) + "] "+ "Connection Table:");

System.out.println("============================");

System.out.println("Destination       Interface");              //If there is no
Interface to the router then assign -1

for (int i = 0; i<Max_rooters; i++){

System.out.print("     "+  nodeid(i) + "             -1");

System.out.println();

}

}

}
```

/* PrintConnectionTable Method */

```
public void PrintConnectionTabel() {

//System.out.println("Enter  Id< 0 - :");

System.out.println("Enter Source Rooter Id< 1 - "+ (Max_rooters)+" >:");

Scanner in1 = new Scanner(System.in);     //Takes input from the User for the
Source Router ID

String str_source = in1.nextLine();
```

```
source = Integer.parseInt(str_source);

source--;                    //Decrements the Source ID by 1

ComputeConnectionTabel();          //Invoke ComputeConnectionTable Method

}
```

/* PrintShortPathToDestination Method */

```
public void PrintShortPathToDestination() {

System.out.println("Enter Destination Rooter Id< 1 - "+ (Max_rooters)+" >:");

Scanner in1 = new Scanner(System.in);              //Takes from the user the
Destination Router ID as input

String str_dest = in1.nextLine();

destination = Integer.parseInt(str_dest);

destination--;                    //Decrements Destination Router ID

if (DistGrph[source][source] == 0) {

if (DistGrph[destination][destination] == 0) {

System.out.print("Shortest Path from Rooter:["+nodeid(source) +"] to ["+
nodeid(destination) + "] is: ");

if (conTable[destination].length > 0) {

for (int n = 0;n< conTable[destination].depth; n++ ) {

if (-1 != conTable[destination].ids[n]) System.out.print(" "+
nodeid(conTable[destination].ids[n]));

}
System.out.println();

System.out.println("The total cost is "+ conTable[destination].length);

}  else System.out.println("Path Not Available");

}  else System.out.println("Destination Rooter is Down");      //If Destination
Router is down

} else System.out.println("Source Rooter is Down");          //If Source Rooter
is down
```

```
}

/* ChangeNetworkTopology */

public void ChangeNetworkTopology(){

System.out.println("Enter Rooter Id< 1 - "+ (Max_rooters)+" > to Down:");

Scanner in1 = new Scanner(System.in);                    //Takes from the user the
Router ID to Down as input

String str_delt = in1.nextLine();

int delid = Integer.parseInt(str_delt);

delid--;

for (int j =0; j < Max_rooters; j++ ){

//System.out.print(" R["+j+"] ");

DistGrph[j][delid] = -1 ;          //Assigns -1 to the Down Router row

}

for (int l =0; l < Max_rooters; l++ ){

//System.out.print(" R["+j+"] ");

DistGrph[delid][l] = -1 ;          //Assigns -1 to the Down Router column

}

//DistGrph[delid][delid] = 0;

//Max_rooters--;

System.out.println("Modified Topology:");

//insert

String imageName = "%3d" ;          //Formatting the content in Matrix format

System.out.println();

System.out.print("ID|");
```

```
for (int j =0; j < Max_rooters; j++ ){

System.out.print(String.format( imageName,nodeid(j)));

}

System.out.println();

System.out.println("------------------------------------------------------------------");

for (int j =0; j < Max_rooters; j++ ){

imageName = "%2d|" ;          //Formatting the content in Matrix format

System.out.print(String.format( imageName,nodeid(j)));

imageName = "%3d" ;

for (int k =0; k < Max_rooters; k++ )

System.out.print( String.format( imageName, DistGrph[j][k]));

//System.out.print(" "+DistGrph[j][k] +"  ");

System.out.println();

}

System.out.println("------------------------------------------------------------------");

}
```

/* MENU */

```
public Dijkstra() {

while (true){

//System.out.println("\n");

System.out.println("=============================================
============\n");

System.out.println("CS542-Link State Routing Simulator:");

System.out.println("==================================\n");
```

```
System.out.println("Enter The Option :\n==================\n1. Create a
Network Topology\n \n2. Build a Connection Table \n \n3. Shortest Path to
Destination Router \n \n4. Modify a topology \n \n5. Exit\n");

System.out.println("Command:");

Scanner in = new Scanner(System.in);

String regmessage = in.nextLine();
if (regmessage.equals("1")){

ReadTextfileToBuildGraph();    //ReadTextFiletoBuildGraph method call

for (int n = 0;n<Max_rooters;n++ ) {    //EXTRA FEATURE IMPLEMENTATION ->
TO DISPLAY CONNECTION TABLE FOR ALL NODES

source = n;

ComputeConnectionTabel();    //ComputeConnectionTable method call

System.out.println();
}
}
if (regmessage.equals("2")){

PrintConnectionTabel();                //PrintConnectionTable method call

}
if (regmessage.equals("3")){

PrintShortPathToDestination();          //PrintShortPathToDestination method
call
}
if (regmessage.equals("4")){

ChangeNetworkTopology();                //ChangeNetworkTopology method call

if ((source >-1) && (source < Max_rooters)){

ComputeConnectionTabel();        //ComputeConnectionTable method call

if (DistGrph[source][source] == 0) {

if ((destination >-1) && (destination < Max_rooters)){

if (DistGrph[destination][destination] == 0) {

System.out.print("Shortest Path from Rooter:["+nodeid(source) +"] to ["+
nodeid(destination) + "] is: ");
```

```
if (conTable[destination].length > -1) {

for (int n = 0;n<Max_rooters;n++ ) {

if (-1 != conTable[destination].ids[n]) System.out.print(" "+
nodeid(conTable[destination].ids[n]));
}
System.out.println();

System.out.println("The total cost is "+ conTable[destination].length);
}
else System.out.println("Not Available");
} else System.out.println("Destination Rooter is Down");
} else System.out.println("Destination node is not selected");
} else System.out.println("Source Rooter is Down");          //Router Check
conditions
}else System.out.println("Source node is not selected");
}
/*
if (regmessage.equals("5")){
for (int n = 0;n<Max_rooters;n++ ) {
source = n;
ComputeConnectionTabel();
System.out.println();
}
}
*/
if (regmessage.equals("5")){
System.out.println("Exit CS542 project. Good Bye!.");
System.exit(0);          //Exit system call
}
}
}
}
```

==========

DESIGN AND WORKFLOW

The Design of the Code is in Java. It has a Simple Command Line Interface.

The design of code is menu driven and works on user inputs. The Application consists of 1 Java Source file, which contains the implementation of all the functionalities of the program.

The detailed design of the source code as follows:

When Dijkstra.java is compiled and run, the application provides the user with the following functionalities, which the user has to select.

The Functionalities are:

1. Create a Network Topology

2. Build a Connection Table

3. Shortest Path to Destination Router

4. Modify a topology

5. Exit

The user is prompted to select his option. Upon selecting the corresponding options, the code executes the corresponding functionality in the program based on the user input.

→ When the user selects option 1, The **ReadTextfileToBuildGraph()** Method is invoked and it is used to read the matrix inside the input topology text file. The input Matrix which is read is then displayed on the Console.

→Extra Feature Implementation – >[3]
Along with this **"The Connection Table for all the Routers in the Network is also Displayed by Default" -> This is an Extra Feature Implementation.**

→Extra Feature Implementation – >[2]
Minimum initial number of nodes: 8 → This has also been implemented. Screenshots and Test Results for this matrix is also documented below.

→ When the user selects option 2, The **PrintConnectionTabel()** method is invoked and the user is prompted to "Enter the Source Router ID". Once the Source router ID is input from the User, it computes the shortest path to every

other node using Dijkstra's Algorithm using the **ComputeConnectionTabel()** method, and the Router Connection table is displayed on the console.

→ When the user select option 3, The **PrintShortPathToDestination()** method is invoked which implements the functionality of finding the Shortest path with the least cost available. This requires the user to "Enter the Destination Router ID", using which the method computes the cost and the complete shortest path using Dijkstra's Algorithm.

→ When the user selects option 4, This implements the "Modify Topology" feature, where in the user is prompted to "Enter the Router ID to remove or make it Down". When the user enters the Router ID to be made down, then **ChangeNetworkTopology()** method is invoked and this makes the selected router down. i.e it make it unreachable in the network to other nodes. The value of this router will be "-1" to signify that it is unreachable.

→ When user selects option 5, this is the Exit functionality, and it breaks the code flow and exits the application.

==========

USER MANUAL

Instructions:

The Folder Structure of the Project is as Follows:

The "**CS542Project_02_Krishnamurthy_Karthik.zip**" contains the Entire Project Folder "Dijkstra".

1) Dijkstra → This folder contains **3** subfolders →
a) "Source" → This contains the Source code file "**Dijkstra.java**" and the Compiled Java Bytecodes "**Dijkstra.class**" & "**DijkstraConTableEntry.class**" files within it. The Application can be Run either by Navigating into the Executable folder & running Executable JAR file using **"java –jar DijkstraExecutable.jar"** as mentioned in the Next Step (b) or by running "java Dijkstra" on the Command line from the Source Folder.

b) "Executable" → This folder contains the "DijkstraExecutable.jar" executable file. The Jar file should be executed from the command line using the command "**java –jar DijkstraExecutable.jar**" It also has "**input.txt**" test file which can be used for testing.
In both the above cases, the input files that needs to be tested must be present in the folder from where they are run. i.e → Either from Source folder or Executable folder.
c) "Test_Files" → This folder contains the input test files, for 5,10,12,15 and 20 Routers. i.e "5_routers.txt" and so on….
d) Link_State Routing_PPT→ Project Presentation File
e)Link_State_Routing_REPORT → Project Report and Operations manual.

Instructions to run:

- Keep the "**.txt" file which is the input file ready in the same directory of source file. Any input test file that needs to be tested should be copied into the Source directory.
- Ensure that a ".txt" file format is chosen and a square matrix is entered with the values separated by a single space.
- From the Command prompt/Terminal, Navigate into the "**Executable**" folder
- Run the Dijkstra java application or Jar through command line by typing following command **"java -jar DijkstraExecutable.jar"**
- For Compiling the Code → On the command prompt use the following command "javac Dijkstra.java". For running

directly without compiling, on command line use **"java Dijkstra"** from the Source folder –OR- "**java –jar DijkstraExecutable.jar**" from the Executable Folder.

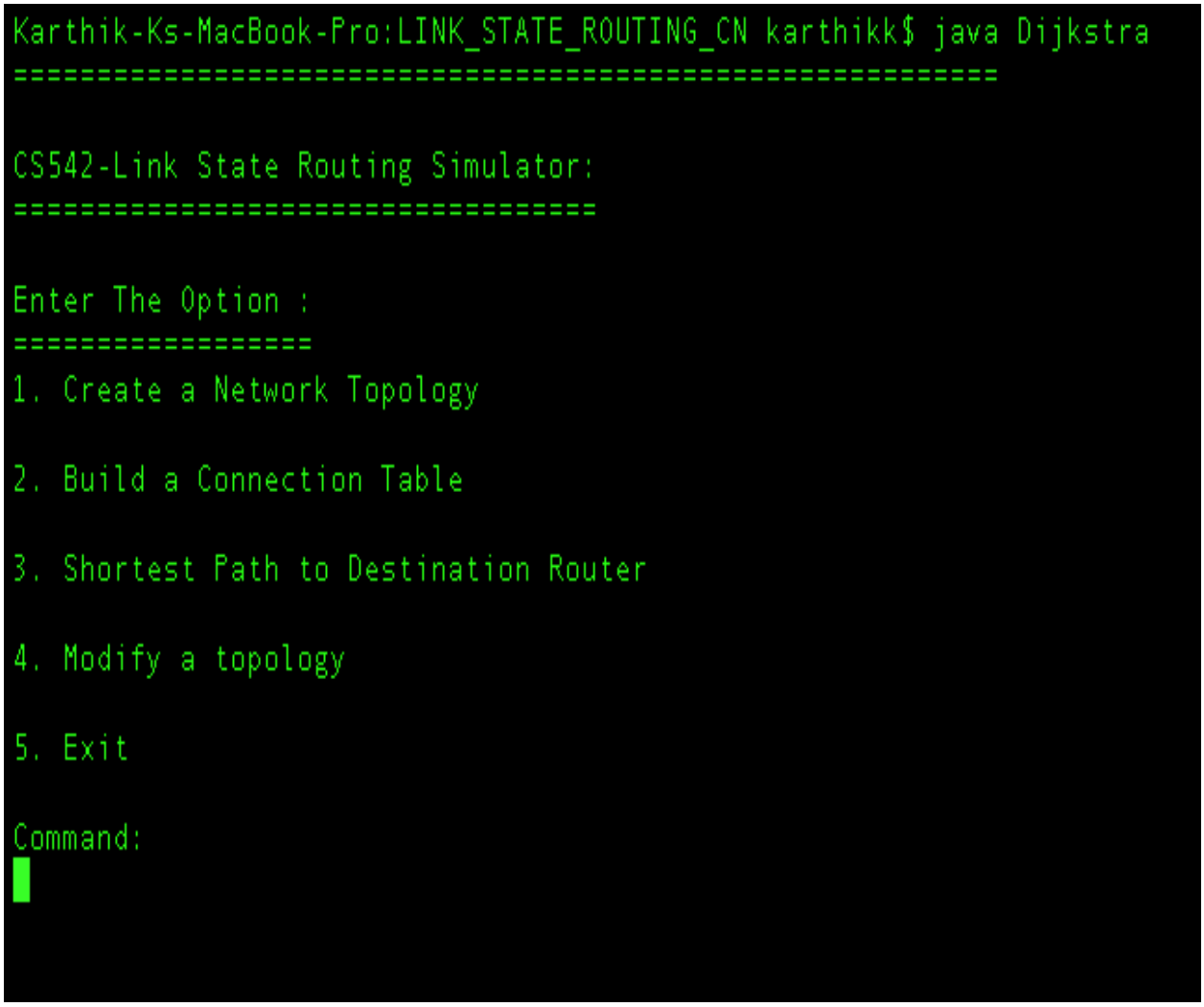
- The input file has to be in the same location as the source, ".class" and jar files are present. If new input file has to be tested, it has to be copied to the same location as the source, class and jar files.
- The application simulation will start running in the terminal

==========

SNAPSHOTS PRESENTED BELOW

SNAPSHOTS

INTERFACE DESIGN - INITIAL MENU SCREEN:



INPUT TO THE APPLICATION – COMMAND LINE

```
Karthik-Ks-MacBook-Pro:LINK_STATE_ROUTING_CN karthikk$ java Dijkstra
============================================================

CS542-Link State Routing Simulator:
===================================

Enter The Option :
==================
1. Create a Network Topology

2. Build a Connection Table

3. Shortest Path to Destination Router

4. Modify a topology

5. Exit

Command:
1
Enter Text File name to read Network Topalogy Matrix:
5_routers.txt
```

REVIEW OF INPUT AND CONNECTION TABLE

```
Karthik-Ks-MacBook-Pro:LINK_STATE_ROUTING_CN karthikk$ java Dijkstra
============================================================

CS542-Link State Routing Simulator:
===================================

Enter The Option :
==================
1. Create a Network Topology

2. Build a Connection Table

3. Shortest Path to Destination Router

4. Modify a topology

5. Exit

Command:
1
Enter Text File name to read Network Topalogy Matrix:
5_routers.txt
<=======:Graph Size Read:=======>5
<=======:Graph Table Initialized:=======>

ID|  1  2  3  4  5
-----------------------------------------------------------------
 1|  0  2  5  1 -1
 2|  2  0  8  7  9
 3|  5  8  0 -1  4
 4|  1  7 -1  0  2
 5| -1  9  4  2  0
-----------------------------------------------------------------
```

CONNECTION TABLE OF ALL ROUTERS

```
Router [1] Connection Table:
============================
Destination          Interface
        1                  -
        2                  2
        3                  3
        4                  4
        5                  4

Router [2] Connection Table:
============================
Destination          Interface
        1                  1
        2                  -
        3                  1
        4                  1
        5                  1

Router [3] Connection Table:
============================
Destination          Interface
        1                  1
        2                  1
        3                  -
        4                  5
        5                  5
```

```
Router [4] Connection Table:
============================
Destination          Interface
        1                 1
        2                 1
        3                 1
        4                 -
        5                 5

Router [5] Connection Table:
============================
Destination          Interface
        1                 4
        2                 4
        3                 3
        4                 4
        5                 -

============================
```

BUILDING A CONNECTION TABLE

```
CS542-Link State Routing Simulator:
===================================

Enter The Option :
==================
1. Create a Network Topology

2. Build a Connection Table

3. Shortest Path to Destination Router

4. Modify a topology

5. Exit

Command:
2
Enter Source Rooter Id< 1 - 5 >:
2




Router [2] Connection Table:
============================
Destination        Interface
      1                1
      2                -
      3                1
      4                1
      5                1
============================================================
```

DIJKSTRA – SHORTEST PATH

```
CS542-Link State Routing Simulator:
===================================

Enter The Option :
==================
1. Create a Network Topology

2. Build a Connection Table

3. Shortest Path to Destination Router

4. Modify a topology

5. Exit

Command:
3
Enter Destination Rooter Id< 1 - 5 >:
5
Shortest Path from Rooter:[2] to [5] is  2 1 4 5
The total cost is 5
============================================================
```

REMOVE A NODE FROM THE NETWORK

```
CS542-Link State Routing Simulator:
===================================

Enter The Option :
==================
1. Create a Network Topology

2. Build a Connection Table

3. Shortest Path to Destination Router

4. Modify a topology

5. Exit

Command:
4
Enter Rooter Id< 1 - 5 > to Down:
4
Modified Topology:

ID|  1  2  3  4  5
-------------------------------------------------------------------------
 1|  0  2  5 -1 -1
 2|  2  0  8 -1  9
 3|  5  8  0 -1  4
 4| -1 -1 -1 -1 -1
 5| -1  9  4 -1  0
-------------------------------------------------------------------------




Router [2] Connection Table:
============================
Destination        Interface
      1                1
      2                -
      3                1
      4                -1
      5                5
Shortest Path from Rooter:[2] to [5] is  2 5
The total cost is 9
============================================================
```

CLOSING THE SIMULATOR

```
CS542-Link State Routing Simulator:
===================================

Enter The Option :
==================
1. Create a Network Topology

2. Build a Connection Table

3. Shortest Path to Destination Router

4. Modify a topology

5. Exit

Command:
5
Exit CS542 project. Good Bye!.
```

==========

TESTING AND RESULTS:
8X8 Matrix – 8 Routers:

INPUT TO THE APPLICATION – COMMAND LINE

The Network topology is selected which has 8 Routers.

The Input matrix is read from the input file, and the Network topology is displayed on the Console.

Connection Table:

```
Karthik-Ks-MacBook-Pro:LINK_STATE_ROUTING_CN karthikk$ java Dijkstra
============================================================

CS542-Link State Routing Simulator:
===================================

Enter The Option :
==================
1. Create a Network Topology

2. Build a Connection Table

3. Shortest Path to Destination Router

4. Modify a topology

5. Exit

Command:
1
Enter Text File name to read Network Topalogy Matrix:
8_routers.txt
<=======:Graph Size Read:=======>8
<=======:Graph Table Initialized:=======>

ID|  1  2  3  4  5  6  7  8
------------------------------------------------------------------------
 1|  0 43 42 44 -1 27 31 32
 2| 43  0 42 40 38 28 30 34
 3| 42 42  0 43 36 29 29 -1
 4| 44 40 43  0 36 30 28 36
 5| -1 38 36 36  0 31 -1 38
 6| 27 28 29 30 31  0 27 40
 7| 31 30 29 28 -1 27  0 -1
 8| 32 34 -1 36 38 40 -1  0
------------------------------------------------------------------------
```

CONNECTION TABLE FOR ALL ROUTERS

```
Router [1] Connection Table:
==============================
Destination           Interface
          1                   -
          2                   2
          3                   3
          4                   4
          5                   6
          6                   6
          7                   7
          8                   8




Router [2] Connection Table:
==============================
Destination           Interface
          1                   1
          2                   -
          3                   3
          4                   4
          5                   5
          6                   6
          7                   7
          8                   8




Router [3] Connection Table:
==============================
Destination           Interface
          1                   1
          2                   2
          3                   -
          4                   4
          5                   5
          6                   6
          7                   7
          8                   6
```

```
Router [4] Connection Table:
==============================
Destination           Interface
          1                   1
          2                   2
          3                   3
          4                   -
          5                   5
          6                   6
          7                   7
          8                   8









Router [5] Connection Table:
==============================
Destination           Interface
          1                   6
          2                   2
          3                   3
          4                   4
          5                   -
          6                   6
          7                   6
          8                   8









Router [6] Connection Table:
==============================
Destination           Interface
          1                   1
          2                   2
          3                   3
          4                   4
          5                   5
          6                   -
          7                   7
          8                   8
```

```
Router [7] Connection Table:
============================
Destination          Interface
        1                 1
        2                 2
        3                 3
        4                 4
        5                 6
        6                 6
        7                 -
        8                 1









Router [8] Connection Table:
============================
Destination          Interface
        1                 1
        2                 2
        3                 6
        4                 4
        5                 5
        6                 6
        7                 1
        8                 -

=======================================
```

BUILDING A CONNECTION TABLE

The Source Router is selected to be 8.

The Connection table is built.

```
CS542-Link State Routing Simulator:
===================================

Enter The Option :
==================
1. Create a Network Topology

2. Build a Connection Table

3. Shortest Path to Destination Router

4. Modify a topology

5. Exit

Command:
2
Enter Source Rooter Id< 1 - 8 >:
8









Router [8] Connection Table:
============================
Destination        Interface
      1                1
      2                2
      3                6
      4                4
      5                5
      6                6
      7                1
      8                -
============================================================
```

DIJKSTRA – SHORTEST PATH

The Destination Router is selected to be 1. In this case the Dijkstra's algorithm is applied and the shortest path is found out.

```
CS542-Link State Routing Simulator:
===================================

Enter The Option :
==================
1. Create a Network Topology

2. Build a Connection Table

3. Shortest Path to Destination Router

4. Modify a topology

5. Exit

Command:
3
Enter Destination Rooter Id< 1 - 8 >:
1
Shortest Path from Rooter:[8] to [1] is  8 1
The total cost is 32
============================================================
```

There is a Direct Link from the Source Router 8 to the Destination Router 1. And the Cost is 32.

DIJKSTRA – SHORTEST PATH

The Destination Router is now selected to be 3.

```
CS542-Link State Routing Simulator:
===================================

Enter The Option :
==================
1. Create a Network Topology

2. Build a Connection Table

3. Shortest Path to Destination Router

4. Modify a topology

5. Exit

Command:
3
Enter Destination Rooter Id< 1 - 8 >:
3
Shortest Path from Rooter:[8] to [3] is  8 6 3
The total cost is 69
============================================================
```

In this case, the Shortest Path is computed from 8 to 3 via Router 6. This shows the Computation of Shortest path traversing through Multiple Nodes.

REMOVE A NODE FROM THE NETWORK

The Modify Topology → Can be used to make the Router Down. Once the Router is Down, it is indicated by "-1".

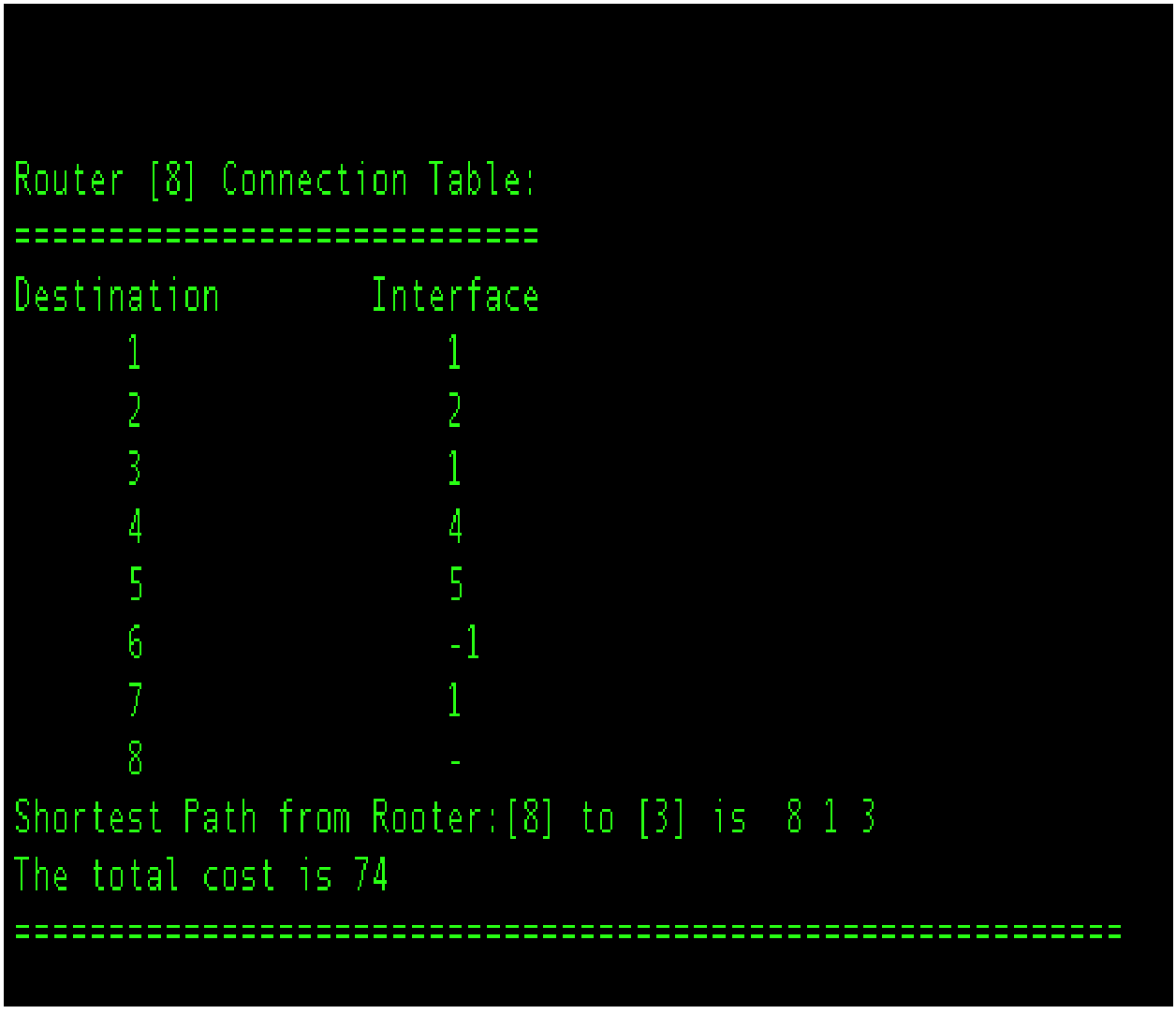
In the Screenshot, Router 6 is Made Down. So that Shortest path from Router 8 to Router 3 is computed via a different path, since Router 6 is down.

```
CS542-Link State Routing Simulator:
===================================

Enter The Option :
==================
1. Create a Network Topology

2. Build a Connection Table

3. Shortest Path to Destination Router

4. Modify a topology

5. Exit

Command:
4
Enter Rooter Id< 1 - 8 > to Down:
6
Modified Topology:

ID|  1  2  3  4  5  6  7  8
-------------------------------------------------------------------
 1|  0 43 42 44 -1 -1 31 32
 2| 43  0 42 40 38 -1 30 34
 3| 42 42  0 43 36 -1 29 -1
 4| 44 40 43  0 36 -1 28 36
 5| -1 38 36 36  0 -1 -1 38
 6| -1 -1 -1 -1 -1 -1 -1 -1
 7| 31 30 29 28 -1 -1  0 -1
 8| 32 34 -1 36 38 -1 -1  0
-------------------------------------------------------------------
```

The Modified Topology is Displayed with "-1" indicated in the Router 6 Row and Column, since Router 6 was made Down.

The Updated Connection Table for the Source Router is Displayed and Now the Shortest Path is computed From Router 8 to Router 3.

Since Router 6 which was the intermediate router for the shortest path from Router 8 to Router 3 was Down, It computes the next Shortest path from Router 8 to Router 3 via Router 1
The Total cost is 74.



Now if Connection Table for Router 6 is checked, the Table Displays "-1", Since the Router 6 was made Down.

```
CS542-Link State Routing Simulator:
===================================

Enter The Option :
==================
1. Create a Network Topology

2. Build a Connection Table

3. Shortest Path to Destination Router

4. Modify a topology

5. Exit

Command:
2
Enter Source Rooter Id< 1 - 8 >:
6
Router [6] Connection Table:
============================
Destination        Interface
      1                -1
      2                -1
      3                -1
      4                -1
      5                -1
      6                -1
      7                -1
      8                -1
============================================================
```

AFTER SELECTING THE SOURCE ROUTER, REMOVING THE SAME SOURCE ROUTER AND TRYING TO FIND THE SHORTEST PATH FROM THAT ROUTER

Here in this Scenario, The source Router is taken as 8. Then by modifying the topology, the Router 8 is made Down. Then If the Shortest Path is computed from the Source Router to any other Router "Source Router is Down" will be displayed.

```
CS542-Link State Routing Simulator:
===================================

Enter The Option :
==================
1. Create a Network Topology

2. Build a Connection Table

3. Shortest Path to Destination Router

4. Modify a topology

5. Exit

Command:
4
Enter Rooter Id< 1 - 8 > to Down:
8
Modified Topology:

ID|  1  2  3  4  5  6  7  8
------------------------------------------------------------------------
 1|  0 43 42 44 -1 27 31 -1
 2| 43  0 42 40 38 28 30 -1
 3| 42 42  0 43 36 29 29 -1
 4| 44 40 43  0 36 30 28 -1
 5| -1 38 36 36  0 31 -1 -1
 6| 27 28 29 30 31  0 27 -1
 7| 31 30 29 28 -1 27  0 -1
 8| -1 -1 -1 -1 -1 -1 -1 -1
------------------------------------------------------------------------
Router [8] Connection Table:
============================
Destination        Interface
      1                -1
      2                -1
      3                -1
      4                -1
      5                -1
      6                -1
      7                -1
      8                -1
Source Rooter is Down
============================================================
```

Similarly for Destination Router as well, this scenario is handled.

==========

TEST RESULTS

The Link – State Routing Protocol implemented in this project using Dijkstra's algorithm has been tested under various cases and is found to be successful. This project has been tested for various number of routers like 5 routers, 10 routers, 12 routers, 15 routers, 20 routers etc. The successful and correct results were obtained during testing and the results are tabulated as below.

- **Instance 1:** Number of routers = 5

  Input File: 5_routers.txt

  Original Matrix:

  0 2 5 1 -1

  2 0 8 7 9

  5 8 0 -1 4

  1 7 -1 0 2

  -1 9 4 2 0

  Routing or Connection Table:

ID	R1	R2	R3	R4	R5
R1	-	2	3	4	4
R2	1	-	1	1	1
R3	1	1	-	5	5
R4	1	1	1	-	5
R5	4	4	3	4	-

Shortest Path from Rooter: [1] to [5] is: 1 – 4 - 5
The total cost is 3

Shortest Path from Rooter: [3] to [1] is: 3 - 1
The total cost is 5

Shortest Path from Rooter: [5] to [4] is: 5 - 4
The total cost is 2

- **Instance 2:** Number of routers = 8

  Input File: 8_routers.txt

  Original Matrix:

0 43 42 44 -1 27 31 32
43 0 42 40 38 28 30 34
42 42 0 43 36 29 29 -1
44 40 43 0 36 30 28 36
-1 38 36 36 0 31 -1 38
27 28 29 30 31 0 27 40
31 30 29 28 -1 27 0 -1
32 34 -1 36 38 40 -1 0

  Routing or Connection Table:

	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**
R1	-	2	3	4	6	6	7	8
R2	1	-	3	4	5	6	7	8
R3	1	2	-	4	5	6	7	8
R4	1	2	3	-	5	6	7	8
R5	6	2	3	4	-	6	6	8
R6	1	2	3	4	5	-	7	8
R7	1	2	3	4	6	6	-	1
R8	1	2	6	4	5	6	1	-

Shortest Path from Rooter: [2] to [6] is: 2 - 6
The total cost is 28

Shortest Path from Rooter: [4] to [8] is: 4 - 8
The total cost is 36

Shortest Path from Rooter: [8] to [1] is: 8 - 1
The total cost is 32

- **Instance 3:** Number of routers = 10
  Input File: 10_routers.txt
  Original Matrix:

0 2 -1 -1 -1 -1 -1 -1 1 5
2 0 2 -1 -1 -1 -1 -1 -1 -1
-1 2 0 4 -1 -1 -1 -1 -1 -1
-1 -1 4 0 2 -1 -1 -1 4 2
-1 -1 -1 2 0 -1 4 -1 -1 -1
-1 -1 -1 -1 -1 0 2 -1 -1 5
-1 -1 -1 -1 4 2 0 6 -1 -1
-1 -1 -1 -1 -1 -1 6 0 8 -1
1 -1 -1 4 -1 -1 -1 8 0 -1
5 -1 -1 2 -1 5 -1 -1 -1

Routing or Connection Table:

	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
R1	-	2	2	9	9	10	9	9	9	10
R2	1	-	3	3	3	1	3	1	1	1
R3	2	2	-	4	4	4	4	2	2	4
R4	9	3	3	-	5	10	5	9	9	10
R5	4	4	4	4	-	7	7	7	4	4
R6	10	10	10	10	7	-	7	7	10	10
R7	5	5	5	5	5	6	-	8	5	6
R8	9	9	9	9	7	7	7	-	9	7
R9	1	1	1	4	4	1	4	8	-	1
R10	1	1	4	4	4	6	6	6	4	-

Shortest Path from Rooter: [4] to [10] is: 4 - 10
The total cost is 2

Shortest Path from Rooter: [2] to [8] is: 2 - 1 - 9 - 8
The total cost is 11

Shortest Path from Rooter: [10] to [1] is: 10 - 1
The total cost is 5

Similarly, tests have been performed for 10,12,15 and 20 Routers as well and the Test files have also been include in the Project Test_Files directory.

==========

NOTE: ADDITIONAL TWO FEATURES HAVE BEEN IMPLEMENTED FOR EXTRA CREDIT

Extra Credit Features implemented:

2) Minimum initial number of nodes: 8

3) Create a connection table of each node as default and display all

REFERENCES:

https://en.wikipedia.org/wiki/Link-state_routing_protocol

https://en.wikipedia.org/wiki/Dijkstra's_algorithm

https://en.wikipedia.org/wiki/Routing

http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/dijkstraAlgor.htm

http://www.danscourses.com/CCNA-2/link-state-routing-protocols.html

http://www.ciscopress.com/articles/article.asp?p=2180210&seqNum=7

==========