CS 542
COMPUTER NETWORKS 1 – FUNDAMENTALS

Project
Link State routing Algorithm

SUBMITTED BY :

SACHIN KRISHNA MURTHY

CWID : A20354077

Contents :

Introduction

Routing :

Basically Routing is the process of selecting the best paths in a network. However, Routing is also described as forwarding. Routing is usually performed by a device called Router. In internetworking, the process of moving a packet of data from source to destination. Routing is a key feature of the Internet because it enables messages to pass from one computer to another and eventually reach the target machine. Each intermediary computer performs routing by passing along the message to the next computer. Part of this process involves analyzing a routing table to determine the best path. Some of the applications of Routing are :

- Telephone Network : Circuit Switching
- Electronic Data Network : Internet
- Transportation Networks

In case of considering Packet Switched Networks, routing directs packet forwarding through intermediate nodes which are typically network hardware devices such as routers, bridges, gateways, firewalls or switches. Routing tables maintains a record of the routes to various network destinations. Building Routing tables is a very important part for the routing process to be efficient. This is because the routing process usually directs the forwarding based on the routing tables. Most routing algorithms use only one network path at a time. Multipath routing techniques enable the use of multiple alternative paths.

Link State Routing Protocols :

Link State Routing Protocols are one of the two main classes of routing protocols used in packet switching networks for computer communications and the other being the distance vector routing protocols. Link-state routing protocols, such as OSPF and IS-IS create a topology of the network and place themselves at the root of the tree.

The link-state protocol is performed by every router in the network .The basic concept of link-state routing is that every node constructs a map of the connectivity to the network, in the form of a graph, showing which nodes are connected to which other nodes. Each node then independently calculates the next best logical path from it to every possible destination in the network. The collection of best paths will then form the node's routing table.

Link State Routing protocols are routing protocols whose algorithms calculate the best paths to networks in a different way when compared to Distance vector routing protocols. Distance vector routing protocols know the routes by measure of distance and direction by neighboring routers. Link state routing protocols calculate their network routes by building a complete topology of the entire network area and then calculating the best path from this topology of all the interconnected networks.

Characteristics of Link State Routing Protocols :

- Shortest Path First Algorithm : Link State routing protocols are designed around Shortest Path First algorithm in which the shortest path from point A to point B is build around a metric of cost.

- Cost Metric : SPF algorithm finds the shortest path based on the metric network link costs. Each router measures the cost of its own directly connected networks or links
- Hello Packets : Link State routing protocol establishes adjacencies with neighbouring routers using hello packets.
- Link State Packets : Initial flooding of link states to all routers in the network.
- Topology : Link State routing protocols build a complete topology or map of a network area.

- **Advantages of Link State Routing Protocols :**

- Faster Convergence : Link State Routing Protocols sends link updates to all the routers in the network before running route calculations.
- Triggered Updates : Link state routing protocols sends LSP's during router startup and also when a link changes it states like going up or down. If there are no changes in the network the protocol only sends hello packets to maintain adjacencies.
- Scalability : Link State routing protocols support the ability to configure multiple routing areas which allows an administrator to segment a routing protocol processes to defined areas which supports the expansion and troubleshooting of much larger networks.

- **Disadvantages of Link State Routing Protocols :**

- Greater Processing Requirements : Link State Routing protocols demand greater processing power and memory resources from the router.
- Greater administrator knowledge : Link state routing protocols can demand advances administrator knowledge to configure and troubleshoot the network area.

- **Dijkstra's Algorithm :**

Djikstra's algorithm solves the problem of finding the shortest path from a point in a graph (the source) to a destination. It turns out that one can find the shortest paths from a given source to all points in a graph in the same time, hence this problem is sometimes called the single-source shortest paths problem.

For a given source node in the graph, the algorithm finds the shortest path between that node and every other. It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined. For example, if the nodes of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities.

- **Pseudocode :**

Initialization:

Visited = {source}

for all nodes v

if v is neighbor of source

then Distance(v) = cost(source,v)

else Distance(v) = ∞

Loop

find node w not in Visited such that Distance(w) is a minimum

add w to Visited

update Distance(v) for each neighbor v of w and not in Visited:

Distance(v) = min(Distance(v), Distance(w) + cost(w,v))

/* new cost to v = either old cost to v

or known least path cost to w + cost (w,v)*/

until Visited = all nodes

- **<u>Working Procedure :</u>**

Consider a 5*5 matrix shown below :
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

Considering Source node=1, {according to matrix source =0}

Initialization :

Visited= [false, false, false, false, false]
Distance= matrix[source]= matrix[0]= [0, 2, 5, 1, ∞]

Visited[0]=true
➔ Visited= [true, false, false, false, false]
For each node v
V=1 :

for all the nodes w where visited[w]=false find minimum Distance
Minimum cost = 1 = Distance[3]
Next node=3
Visited[Next node]=true
➔ Visited= [true, false, false, true, false]

for all the nodes u where visited[u]=false
u=4
Minimum cost + matrix[Next node][u] < Distance[u]
Distance[u] = Minimum cost + matrix[Next node][u] = 1+2 =3

➔Distance = [0, 2, 5, 1, 3]

V=2 :

for all the nodes w where visited[w]=false find minimum Distance
Minimum cost = 2 = Distance[1]
Next node=1
Visited[Next node]=true
➔ Visited= [true, true, false, true, false]

for all the nodes u where visited[u]=false
for all value of u
Minimum cost + matrix[Next node][u] > Distance[u]
➔Distance = [0, 2, 5, 1, 3]

V=3 :

for all the nodes w where visited[w]=false find minimum Distance
Minimum cost = 3 = Distance[4]
Next node=4
Visited[Next node]=true
➔ Visited= [true, true, false, true, true]

for all the nodes u where visited[u]=false
for all value of u
Minimum cost + matrix[Next node][u] > Distance[u]
➔Distance = [0, 2, 5, 1, 3]

V=4 :

for all the nodes w where visited[w]=false find minimum Distance
Minimum cost = 5 = Distance[2]
Next node=2
Visited[Next node]=true
➔ Visited= [true, true, true, true, true]

for all the nodes u where visited[u]=false
Visited[All nodes] =true
Do nothing;

V=5 :

for all the nodes w where visited[w]=false find minimum Distance
Visited[All nodes] =true
Do nothing;

for all the nodes u where visited[u]=false
Visited[All nodes] =true
Do nothing;

Final shortest Distance:
Distance = [0, 2, 5, 1, 3]

Final shortest Distance from source node 1 :

Destination Node	Distance
1	0
2	2
3	5
4	1
5	3

- **<u>Instructions to Compile and Run the Program :</u>**

- Make sure you keep the input file in the same directory as of the source file.
- The program is compatible for .txt file format. So make sure that a .txt input file is chosen and it should contain numeric values in a square matrix with the values separated by a single space.
- Since the class name mentioned in the program is ProjectMain, you can compile the program in the command prompt using the option :
  javac ProjectMain.java

```
C:\Users\Sachin K\Desktop>javac ProjectMain.java
```

- After compilation you can run the program using the command :
  java ProjectMain

```
C:\Users\Sachin K\Desktop>java ProjectMain
```

- You can also run the program using jar from the folder where the input file is present as :
  java –jar ProjectMain.jar

- The application will start running and you can enter the choices as per your requirement.
- The zip folder CS542Project_02_Krishna Murthy_Sachin consists of :
  - CN Project Report
  - CN Project PPT
  - A folder Source Code which contains “ProjectMain.java” and “ProjectMain.class” files. It also has two input files used i,e “input.txt” and “Ten.txt”.
  - A folder Executable File contains “ProjectMain.jar” and two input files “input.txt” and “Ten.txt”.

Source Code

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 *
 * @author Sachin K
 *
 */

public class ProjectMain {

	public static void main(String[] args) {

		Scanner in = new Scanner(System.in);
		Scanner on = new Scanner(System.in);
		String op;
		int source = 0, dest;


		System.out.println("Input original network topology matix data
file: ");
		String fileName = in.nextLine();
		// reading filename containing topology matrix

		List<List<Integer>> matrix;
		List<Integer> removedNodes = new ArrayList<Integer>();

		matrix = readFileIntoMatrix(fileName);

		//Check if file is valid with topology matrix
		if (matrix == null || matrix.size() == 0) {
			if (matrix != null) {
				System.out.println("File Deosnt contain Topalogy
Matrix");
			}
			System.out.println("\nExit CS542 project.  Good Bye! ");
			return;
		}
		// to check if input file topology matrix deosnt have any error
		boolean isFileErrorfree = true;
		for (int i = 0; i < matrix.size(); i++) {
			for (int j = 0; j < matrix.size(); j++) {
				// check if there are errors n topology
				if (matrix.get(i).get(j) != matrix.get(j).get(i) &&
						(matrix.get(i).get(j)>= 0 ||
matrix.get(j).get(i)>=0)) {
					if (isFileErrorfree) {
						System.out.println("Input file error");
					}
```

```
                                    System.out.println("Cost(" + (i + 1) + "," + (j
+ 1)
                                                + ") != Cost(" + (j + 1) + "," + (i
+ 1) + ")");
                                    isFileErrorfree = false;
                              }
                              if ( i==j && matrix.get(i).get(j)!= 0) {
                                    if (isFileErrorfree) {
                                          System.out.println("Input file error");
                                    }
                                    System.out.println("Cost(" + (i + 1) + "," + (j
+ 1)
                                                + ") != 0 ");
                                    isFileErrorfree = false;
                              }
                        }
                  }

                  if (!isFileErrorfree) {
                        System.out.println("\nExit CS542 project.  Good Bye! ");
                        return;
                  }

                  int prevOption = 0;

                  while (true) {

                        List<Object[]> resultList = extractShortestPath(matrix,
                                    removedNodes);

                        System.out
                                    .println("\n CS542 Link State Routing Simulator
: \n"
                                                + "(1) Create a Network Topology
\n"
                                                + "(2) Build a Connection Table for
all nodes  \n"
                                                + "(3) Build a Connection Table for
given source node  \n"
                                                + "(4) Shortest Path to Destination
Router [choose only if last option was 3]  \n"
                                                + "(5) Modify a topology  \n" +
"(6) Exit \n"
                                                + "\n Command : ");

                        int choice;

                        //Making sure that source node is selected before
destination node
                        while (true) {
                              choice = on.nextInt();
                              if (choice == 4 && prevOption != 3) {
                                    System.out
                                                .println("you have to choose choice
3 before choice 4");
                                    continue;
                              } else {
```

```
                        break;
                    }
                }

                switch (choice) {
                case 1:
                    System.out.println("Review topology matrix: ");
                    printTopologyMatrix(matrix);
                    break;

                case 2:
                    System.out.println("Review Connection table: ");
                    printConnectionTable(resultList, removedNodes);
                    break;

                case 3:

                    System.out.println("Enter Source Node:");
                    for (;;) {
                        source = on.nextInt(); //read source node
                        //make sure source node is within the network
and is not removed
                        if (removedNodes.contains(source) || source >
matrix.size()) {
                            System.out.println(" node dosent exist in
network\n "
                                    + "Enter new source node");
                        } else {
                            break;
                        }
                    }
                    performSourceNodeOperations(source, resultList);
                    break;

                case 4:
                    System.out.println("Enter Destination Node");
                    for (;;) {
                        dest = on.nextInt(); //read destination node
                        //make sure destination node is within the
network and is not removed
                        if (removedNodes.contains(dest) || dest >
matrix.size()) {
                            System.out.println(" node dosent exist in
network\n "
                                    + "Enter new Destination
node");
                        } else {
                            break;
                        }
                    }
                    performDestinationNodeOperations(source, dest,
resultList);
                    break;

                case 5:

                    System.out.println("Enter Node to remove");
```

```
                        int node;

                        for (;;) {
                              node = on.nextInt(); //read node to remove
                              // make sure node is within network
                              if (node > matrix.size()) {
                                    System.out.println("Invalid node\nenter
another node");
                                    continue;
                              } else if (removedNodes.contains(node)) { //
make sure node is not removed
                                    System.out.println("Node already
removed\n");
                                    continue;
                              } else {
                                    break;
                              }
                        }
                        List<List<Integer>> tempMatrix =
performRemoveNode(matrix, node);
                        System.out.println(" Node"+node+" removed from
network");
                        //assign tempmatrix with node removed to original
matrix
                        matrix = tempMatrix;
                        //removed nodes list is maintained for further
reference
                        removedNodes.add(node);
                        break;

                  case 6:
                        break;

                  default:
                        System.out.println("Invalid option\n");
                        break;
                  }

                  if (choice == 6) {
                        break;
                  } else if (removedNodes.size() == matrix.size() - 1) {//
exit if matrix has only one node left in network
                        System.out.println("Review topology matrix: ");
                        printTopologyMatrix(matrix);
                        System.out.println("\n only 1 node in network");
                        System.out.println("No more operations possible");
                        break;
                  }
                  prevOption = choice;

            }

            System.out.println("\nExit CS542 project.  Good Bye! ");

      }
/**
 * The method takes topology matrix read from file as input and displays it
```

```
 * @param matrix
 */
	private static void printTopologyMatrix(List<List<Integer>> matrix) {

		System.out.print("   | ");
		for (int j = 1; j <= matrix.size(); j++) {
			if(j < 10){
				System.out.print(" ");
			}
			System.out.print(j + " ");
		}
		System.out.println("\n-------------------------------------");
		int rowCount = 1;
		//print topology matrix
		for (List<Integer> rows : matrix) {
			if(rowCount < 10){
				System.out.print(" ");
			}
			System.out.print(rowCount + " | ");
			rowCount++;
			for (Integer val : rows) {
				// condition to align display
				if(-1 < val && 10 > val)
					System.out.print(" ");
				System.out.print(val + " ");
			}
			System.out.println();
		}
	}

	/**
	 * Takes the file name as the input and reads file into a matrix and
returns the matrix
	 * @param fileName
	 * @return
	 */
	private static List<List<Integer>> readFileIntoMatrix(String fileName)
{

		List<List<Integer>> matrix = null;
		try {
			File file = new File(fileName);
			FileReader fr = new FileReader(file);
			BufferedReader br = new BufferedReader(fr);

			matrix = new ArrayList<List<Integer>>();
			String curLineString = br.readLine(); // reading first line
of the file

			while (null != curLineString) {

				// Splitting curLineString by space into array of
distances
				String[] distanceArray = curLineString.split(" ");

				List<Integer> row = new ArrayList<>();
				for (String value : distanceArray) {
```

```
                              row.add(Integer.valueOf(value)); // reading
values into row list
                        }
                        matrix.add(row); // reading rows into matrix
                        curLineString = br.readLine();// reading second to
last line of file
                  }
                  br.close();// close reader

            } catch (FileNotFoundException e) {
                  //if file is invalid execption is given
                  System.out.println("File not found");
            } catch (IOException e) {
                  //if file cannot be read execption is given
                  System.out.println("Cannot read file");
            }catch (NumberFormatException e){
                  //if file contents are string invalid execption is given
                  System.out.println("File contents invalid");
            }

            return matrix;
      }

      /**
       * It takes list of <object[]> result list as input along with the list
of removed nodes.
       * Prints topology matrix for nodes that are not in removed nodes.
       * @param resultList
       * @param removedNodes
       */
      private static void printConnectionTable(List<Object[]> resultList,
                  List<Integer> removedNodes) {
            for (int i = 0; i < resultList.size(); i++) {
                  if (removedNodes.contains(i + 1)) {
                        //connection table is not displayed for removed node
                        System.out.println("Router" + (i + 1) + " is
removed");
                        continue;
                  }
                  Object[] resultObj = resultList.get(i);
                  System.out.println("\nRouter" + (i + 1) + " connection
table");
                  System.out.println("================================");
                  System.out.println("Destination      Interface");
                  //Read path for nnodei+1
                  String[] path = (String[]) resultObj[1];
                  for (int j = 1; j <= path.length; j++) {
                        String interfac;
                        //display second node in the path as interface
                        if (path[j - 1].equalsIgnoreCase("-")) {
                              interfac = path[j - 1];
                        }else if (path[j - 1].length() == 3) {
                              interfac = path[j - 1].substring(2, 3);
                        }else if (path[j - 1].length() == 4) {
                              interfac = path[j - 1].substring(2, 4);
                        } else if (i < 10) {
                              interfac = path[j - 1].substring(2, 4);
```

```
                        } else {
                              interfac = path[j - 1].substring(3, 5);
                        }

                        if(interfac.length()>1 && interfac.substring(1,
2).equals(",")){
                              interfac = interfac.substring(0, 1);
                        } else if(interfac.length()>1 &&
interfac.substring(0, 1).equals(",")){
                              interfac = interfac.substring(1, 2);
                        }

                        System.out.println("  " + j + "           " + "     "
                                    + interfac);
                  }

            }
      }

      /**
       * This method takes List<Object[]> resultList and source as input.
       * For each node it displays the node,interface from the source, path
from source to node.
       * @param source
       * @param resultList
       */
      private static void performSourceNodeOperations(int source,
                  List<Object[]> resultList) {
            Object[] resultObj = resultList.get(source - 1);
            //read path from object
            String[] path = (String[]) resultObj[1];
            System.out.println("Router   Interface   Path");
            for (int i = 1; i <= path.length; i++) {
                  String interfac;
                  //display second node in the path as interface
                  if (path[i - 1].equalsIgnoreCase("-")) {
                        interfac = path[i - 1];
                  } else if (path[i - 1].length() == 3) {
                        interfac = path[i - 1].substring(2, 3);
                  } else if (path[i - 1].length() == 4) {
                        interfac = path[i - 1].substring(2, 4);
                  } else if (source < 10) {
                        interfac = path[i - 1].substring(2, 4);
                  } else {
                        interfac = path[i - 1].substring(3, 5);
                  }

                  if(interfac.length()>1 && interfac.substring(1,
2).equals(",")){
                        interfac = interfac.substring(0, 1);
                  } else if(interfac.length()>1 && interfac.substring(0,
1).equals(",")){
                        interfac = interfac.substring(1, 2);
                  }

                  System.out.println(i + "         " + interfac + "
"
```

```
                        + path[i - 1].replaceAll(",", "-->"));
            }
      }

      /**
       * This method displays distance and path between a given source and
destination nodes.
       * @param source
       * @param dest
       * @param resultList
       */
      private static void performDestinationNodeOperations(int source, int
dest,
                  List<Object[]> resultList) {
            Object[] resultObj1 = resultList.get(source - 1);
            //read distance from object
            int[] distance = (int[]) resultObj1[0];
            //read path from object
            String[] path1 = (String[]) resultObj1[1];
            //display distance and path
            System.out.println("Router" + source + " to " + "Router" + dest);
            System.out.println("Cost  = " + distance[dest - 1]);
            System.out.println("Path  = "
                        + path1[dest - 1].replaceAll(",", "-->"));
      }

      /**
       * This method creates a temporary matrix and and copies matrix value
to it.
       * For removed nodes -1 is assigned as cost and returns the temporary
matrix.
       * @param matrix
       * @param node
       * @return
       */
      private static List<List<Integer>> performRemoveNode(
                  List<List<Integer>> matrix, int node) {
            List<List<Integer>> tempMatrix = new ArrayList<List<Integer>>();
            //create temp matrisx
            for (int i = 0; i < matrix.size(); i++) {
                  List<Integer> templist = new ArrayList<Integer>();
                  for (int j = 0; j < matrix.size(); j++) {
                        if (i == node - 1 || j == node - 1) {
                              //assign cost of node to be removed as -1
                              templist.add(-1);
                        } else {
                              //assign cost of not removed node as that of
original matrix
                              templist.add(matrix.get(i).get(j));
                        }
                  }
                  tempMatrix.add(templist);
            }
            //return tempmatrix with node removed
            return tempMatrix;
      }
```

```
	/**
	 * It performs the dijkstars algorithm and finds the shortest distance
from source to all other nodes.
	 * It also finds path.Combines both distance and path array and returns
as an object.
	 * @param arr
	 * @param size
	 * @param source
	 * @param maxVal
	 * @param removedNodes
	 * @return
	 */
	// dijkstras algorithm used
	private static Object[] findShortestPathForGivenSource(int[][] arr,
			int size, int source, int maxVal, List<Integer>
removedNodes) {

		boolean[] visited = new boolean[size];

		int[] distance = new int[size];
		String[] path = new String[size];
		int min;
		int nextNode = 0;
		String prevPath;
		source--;

		for (int i = 0; i < size; i++) {
			visited[i] = false; // vsited nodes are set to false
			distance[i] = arr[source][i]; //distance array is source
row of array
		}
		distance[source] = 0;

		for (int i = 0; i < size; i++) {
			//set inital values for path
			path[i] = String.valueOf(source + 1) + "," +
String.valueOf(i + 1);
			if (arr[i][source] == maxVal) {
				path[i] = String.valueOf(source + 1);
			}
		}
		path[source] = "-";
		//visitid of source is set to false and it is not visited
		visited[source] = true;

		for (int i = 0; i < size; i++) {
			//visitid of removed nodes are set to false and are not
visited
			if (removedNodes.contains(i + 1)) {
				path[i] = "-";
				visited[i] = true;
			}
		}

		for (int i = 0; i < size; i++) {
			min = maxVal;
```

```
                  for (int j = 0; j < size; j++)
                        // for all unvisited nodes find minimum distance with
src
                        // and the node is noted as nextNode
                        if (min > distance[j] && visited[j] != true) {
                              min = distance[j];
                              nextNode = j;
                        }

                  visited[nextNode] = true; //This node is set to visited =
true
                  prevPath = path[nextNode]; //prevPath is the path of the
nod with min dist

                  for (int k = 0; k < size; k++) {
                        if (visited[k] != true) {
                              //for each unvisited node k
                              //if minimum dist + cost(nextNode,k) < current
distance of k with source
                              if (min + arr[nextNode][k] < distance[k]) {
                                    //distance is minvalue
                                    //path is minvalue node path,node k
                                    distance[k] = min + arr[nextNode][k];
                                    path[k] = prevPath + "," +
String.valueOf(k + 1);
                              }
                        }
                  }
            }
            Object[] obj = new Object[2];
            obj[0] = distance;
            obj[1] = path;
            // return distance and path array as object
            return obj;
      }

      /**
       * For a given matrix, it creates an array with cost-1 replaced with
maximum value.
       * Calls findshortestpath for given source() to find shortest path and
distance.
       * @param matrix
       * @param removedNodes
       * @return
       */
      private static List<Object[]> extractShortestPath(
                  List<List<Integer>> matrix, List<Integer> removedNodes) {
            List<Object[]> resultList = new ArrayList<Object[]>();
            int size = matrix.size();
            int[][] arr = new int[size][size];
            int maxVal = 999;

            Scanner in = new Scanner(System.in);
            // converting list matrix to two Dimension array
            for (int i = 0; i < size; i++) {
                  for (int j = 0; j < size; j++) {
                        arr[i][j] = matrix.get(i).get(j);
```

```
                        // When no connection, assigning high value to ignore
                        if (arr[i][j] <= -1)
                              arr[i][j] = maxVal;
                  }
            }
            for (int i = 0; i < matrix.size(); i++) {
                  Object[] resultObj = findShortestPathForGivenSource(arr,
size,
                              i + 1, maxVal, removedNodes);
                  resultList.add(resultObj);
            }
            return resultList;
      }


      }
```

User Manual

➢ Implementation Steps :

- **Step 1:** Initially the user is prompted to enter the name of the input file.

```
C:\Users\Sachin K\Desktop>java ProjectMain
Input original network topology matix data file:
input.txt
```

- After entering the input filename, the options available for CS542 Link State Routing Simulator are displayed. Also the user is prompted to enter the option.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
```

- **Step 2:** For option 1 i,e Create a Network Topology, the input topology matrix will be displayed. In this I am considering an example of 8*8 input matrix.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
1
Review topology matrix:
0 21 -1 -1 -1 -1 -1 14
21 0 9 -1 -1 -1 -1 3
-1 9 0 10 -1 18 -1 -1
-1 -1 10 0 20 3 -1 -1
-1 -1 -1 20 0 7 -1 -1
-1 -1 18 3 7 0 13 -1
-1 -1 -1 -1 -1 13 0 4
14 3 -1 -1 -1 -1 4 0
```

Again after displaying the results for the 1st option, the menu is again prompted to the user to continue his operations.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
```

- **Step 3:** For option 2, Build a connection table for all nodes, the connection table of all the routers existing in the input file will be displayed.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
2
```

Router1 connection table

Destination	Interface
1	-
2	8
3	8
4	8
5	8
6	8
7	8
8	8

Router2 connection table

Destination	Interface
1	8
2	-
3	3
4	3
5	8
6	8
7	8
8	8

Router3 connection table

Destination	Interface
1	2
2	2
3	-
4	4
5	4
6	4
7	2
8	2

Router4 connection table

Destination	Interface
1	6
2	3
3	3
4	-
5	6
6	6
7	6
8	6

Router5 connection table

Destination	Interface
1	6
2	6
3	6
4	6
5	-
6	6
7	6
8	6

Router6 connection table

Destination	Interface
1	7
2	7
3	4
4	4
5	5
6	-
7	7
8	7

Router7 connection table

Destination	Interface
1	8
2	8
3	8
4	6
5	6
6	6
7	-
8	8

Router8 connection table

Destination	Interface
1	1
2	2
3	2
4	7
5	7
6	7
7	7
8	-

- **Step 4:** For option 3, Build a connection table for a given source node, the user will prompted to enter the source node for which he wants to display the connection table.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
3
Enter Source Node:
```

For example, when the user enters the source node as 3, the connection table which includes the router, interface along with the path of the 3rd router will be displayed.

```
Enter Source Node:
3
Router   Interface   Path
1         2          3-->2-->8-->1
2         2          3-->2
3         -          -
4         4          3-->4
5         4          3-->4-->6-->5
6         4          3-->4-->6
7         2          3-->2-->8-->7
8         2          3-->2-->8
```

- **Step 5:** For option 4, Shortest path to the destination router, the user will be prompted to enter the destination node to which he wants to determine the shortest path from the source node mentioned in the option 3.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
4
Enter Destination Node:
```

For instance, the user enters the destination node as 7. Then the cost and the shortest path between the source node 3, given in the previous step to the destination node 7 will be displayed.

```
Enter Destination Node:
7
Router3 to Router7
Cost  = 16
Path  = 3-->2-->8-->7
```

- **Step 6:** For option 5, Modify a topology, the user will be prompted to enter the node which he wants to remove from the input file.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
5
Enter Node to remove:
```

After entering the node to be removed, that node will no longer exist in the input file.
For example, consider removing node 2.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
5
Enter Node to remove:
2
```

After removing node 2, when we use the option 1 to display the network topology, we can observe the value of -1 for Router 2.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
1
Review topology matrix:
0 -1 -1 -1 -1 -1 -1 14
-1 -1 -1 -1 -1 -1 -1 -1
-1 -1 0 10 -1 18 -1 -1
-1 -1 10 0 20 3 -1 -1
-1 -1 -1 20 0 7 -1 -1
-1 -1 18 3 7 0 13 -1
-1 -1 -1 -1 -1 13 0 4
14 -1 -1 -1 -1 -1 4 0
```

- **Step 7:** For option 6, Exit, indicates the end of operation and displays the message “Good Bye”.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
6

Exit CS542 project.  Good Bye!
```

Constraints Handled :

- **Constraint 1:** When a user enters a wrong input filename.
  For example, the input file name is “input.txt” but mistakenly the user enters it as “iinput.txt”, then a suitable error message “File not found” is displayed. Also the operation will be terminated displaying the exit message.

```
C:\Users\Sachin K\Desktop>java ProjectMain
Input original network topology matrix data file:
iinput.txt
File not found

Exit CS542 project.  Good Bye!
```

- **Constraint 2:** If the input file contains some junk data such as non numeric data then a suitable error message “File contents invalid” and “File does not contain Topology matrix” is displayed. Also the operation is terminated with a exit message.

```
Input original network topology matrix data file:
junkinput.txt
File contents invalid
File Does not contain Topology Matrix

Exit CS542 project.  Good Bye!
```

- **Constraint 3:** If the input topology matrix is incorrect, then the mistake in the topology matrix is identified and displayed. Also the operation is terminated and exit message is displayed.
  Consider the below input matrix. We can clearly observe that there is an error in the path from 1->10 , 10->1 and 2->10 ,10->2.

```
0 12 17 -1 -1 -1 -1 2 -1 7
 12 0 11 -1 -1 -1 5 2 -1 5
 17 11 0 8 -1 -1 -1 -1 8 7
 -1 -1 8 0 -1 10 -1 9 8 -1
-1 -1 -1 -1 0 3 -1 3 -1 -1
-1 -1 -1 10 3 0 -1 3 -1 -1
-1 5 -1 -1 -1 -1 0 7 -1 -1
  2 2 -1 9 3 3 7 0 -1 -1
-1 -1 8 8 -1 -1 -1 -1 0 -1
-1 6 7 -1 -1 -1 -1 -1 -1 0
```

```
Input original network topology matrix data file:
10
Input file error
Cost(1,10) != Cost(10,1)
Cost(2,10) != Cost(10,2)
Cost(10,1) != Cost(1,10)
Cost(10,2) != Cost(2,10)

Exit CS542 project.  Good Bye!
```

- **Constraint 4 :** We know that in an input matrix, the distance from 1->1 is 0, the distance from 2->2 is 0 and so on. But if the input matrix contains a non zero value for these cases, then a suitable error message is displayed and the operation is terminated.

  In the below case, we can observe that the distance from 3->3 is not 0. There fore the program displays the error as shown in the below screen shot.

  0 21 -1 -1 -1 -1 -1 14
  21 0 9 -1 -1 -1 -1 3
  -1 9 1 10 -1 18 -1 -1
  -1 -1 10 0 20 3 -1 -1
  -1 -1 -1 20 0 7 -1 -1
  -1 -1 18 3 7 0 13 -1
  -1 -1 -1 -1 -1 13 0 4
  14 3 -1 -1 -1 -1 4 0

```
Input original network topology matrix data file:
input.txt
Input file error
Cost(3,3) != 0

Exit CS542 project.  Good Bye!
```

- **Constraint 5:** For example, consider that the input file contains 8 nodes. While executing option 3, Building connection table for given source node, and consider that the user enters source node as 9. Since we have only 8 nodes in our input file, node 9 does not exist. This constraint is handled by displaying a suitable error message “Node does not exist in the network” and also prompts the user to enter a new source node.

  Also the same error message will be displayed when we use the option 4 to find the shortest path by entering the destination router that does not exist in the network.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
3
Enter Source Node:
9
 node dosent exist in network
 Enter new source node
```

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
4
Enter Destination Node:
9
 node dosent exist in network
 Enter new Destination node
```

- **Constraint 6:** It is necessary that the user needs to specify the source node first than the destination node inorder to find the shortest path between them. In case if the user tries to enter the destination node (option 4) before giving the source node (option 3) then the user is prompted to choose option 3 i,e the source node first and then choose option 4 for giving destination node.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
4
you have to choose option 3 before option 4
```

- **Constraint 7:** When the user chooses option 5 to modify a topology that is to remove a node from the matrix, and then if we chooses option 3 to build a connection table for the source node and if he enters the node which he has already removed then a suitable message is prompted like “Node does not exist in the network”.

  In the below screen shot node 3 has been removed.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
5
Enter Node to remove:
3
 Node3 removed from network
```

  Now if we try to find the shortest path using node 3 then the message node 3 doses not exist in the network will be displayed and also prompts the user to enter the new source node to find the shortest path.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
3
Enter Source Node:
3
 node dosent exist in network
 Enter new source node
```

  Also, when he chooses option 4 to enter the destination node and enters the node that has already been removed the same error message will be displayed.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
4
Enter Destination Node:
3
 node dosent exist in network
 Enter new Destination node
```

If the user tries to remove a node which has already been removed a suitable message "Node already removed" will be displayed. In this case the user is trying to remove node 3 which has already been removed.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
5
Enter Node to remove:
3
Node already removed
```

- **Constraint 8:** By using the option 5, if the user tries to remove all the nodes in the matrix, say for example the input consists of 8 nodes. Then the user is allowed to remove upto 7 nodes. After removing 7 nodes there are no more operations possible. So a suitable message is displayed for this constraint and the operation is terminated.

  In the below screen shot we can observe the input consists of -1's since all the nodes except a node has been removed.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
5
Enter Node to remove:
7
 Node7 removed from network
Review topology matrix:
-1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 0

 only 1 node in network
No more operations possible

Exit CS542 project.  Good Bye!
```

- **Constraint 9:** In this case, the link simulator displays 6 options as shown. If the user tries to enter a command more than 6, then the message “invalid option” is displayed.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
7
Invalid option
```

- **Constraint 10 :** If the user specifies the node to be removed using option 5 and enters the node which is not their in the input file, then a suitable message “Invalid Node” is displayed.

```
 Command :
5
Enter Node to remove
12
Invalid node
enter another node
```

Test Results :

- **Considering an 8*8 matrix :**

- Initially the input file 8.txt is taken and the options are displayed and prompts the user to enter the option.

```
Input original network topology matrix data file:
8.txt

 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
```

- **Option 1:** Create a Network Topology
  The input topology matrix will be displayed.

```
 Command :
1
Review topology matrix:
   |  1  2  3  4  5  6  7  8
-------------------------------------
 1 |  0 21 -1 -1 -1 -1 -1 14
 2 | 21  0  9 -1 -1 -1 -1  3
 3 | -1  9  0 10 -1 18 -1 -1
 4 | -1 -1 10  0 20  3 -1 -1
 5 | -1 -1 -1 20  0  7 -1 -1
 6 | -1 -1 18  3  7  0 13 -1
 7 | -1 -1 -1 -1 -1 13  0  4
 8 | 14  3 -1 -1 -1 -1  4  0
```

- **Option 2 :** Build a Connection table for all nodes
  Displays the connection tables for all the nodes of the input.

```
 Command :
2
Review Connection table:

Router1 connection table
================================
Destination      Interface
  1                -
  2                8
  3                8
  4                8
  5                8
  6                8
  7                8
  8                8

Router2 connection table
================================
Destination      Interface
  1                8
  2                -
  3                3
  4                3
  5                8
  6                8
  7                8
  8                8
```

```
Router3 connection table
================================
Destination      Interface
  1                2
  2                2
  3                -
  4                4
  5                4
  6                4
  7                2
  8                2

Router4 connection table
================================
Destination      Interface
  1                6
  2                3
  3                3
  4                -
  5                6
  6                6
  7                6
  8                6
```

```
Router5 connection table
================================
Destination      Interface
  1                6
  2                6
  3                6
  4                6
  5                -
  6                6
  7                6
  8                6

Router6 connection table
================================
Destination      Interface
  1                7
  2                7
  3                4
  4                4
  5                5
  6                -
  7                7
  8                7
```

```
Router7 connection table
================================
Destination      Interface
  1                8
  2                8
  3                8
  4                6
  5                6
  6                6
  7                -
  8                8

Router8 connection table
================================
Destination      Interface
  1                1
  2                2
  3                2
  4                7
  5                7
  6                7
  7                7
  8                -
```

- **Option 3 :** Build a connection table for the source node
  Here the source node is taken as 5.

```
 Command :
3
Enter Source Node:
5
Router   Interface   Path
1         6          5-->6-->7-->8-->1
2         6          5-->6-->7-->8-->2
3         6          5-->6-->4-->3
4         6          5-->6-->4
5         -          -
6         6          5-->6
7         6          5-->6-->7
8         6          5-->6-->7-->8
```

- **Option 4:** Shortest path to Destination router
  Here the shortest path from the source node given in the previous i,e 5 to the destination router given in this step I,e 8 is calculated.

```
 Command :
4
Enter Destination Node:
8
Router5 to Router8
Cost  = 24
Path  = 5-->6-->7-->8
```

If option 4 is chosen before choosing option 3, that is if the source node is not specified before giving the destination router then a suitable message is displayed.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
4
you have to choose choice 3 before choice 4
```

- **Option 5 :** Modify a topology

```
 Command :
5
Enter Node to remove:
1
 Node1 removed from network
```

If you try to remove the already removed router, then a message is displayed and it prompts to enter the another node to be removed.

```
 Command :
5
Enter Node to remove:
1
Node already removed
enter another node
```

If you try to display the connection table for the already removed node then the message “Node does not exist in the network is displayed and prompts the user to enter the new source node.

```
 Command :
3
Enter Source Node:
1
 node dosent exist in network
 Enter new source node
```

Also the same thing applies for finding the shortest path (for both source and destination).
If a node which has already been removed is used to find the shortest path by using option 3 or option 4 then the message is displayed.

```
 Command :
3
Enter Source Node:
1
 node dosent exist in network
 Enter new source node
```

The topology matrix after removing node 1:
We can observe the insertion of -1 value.

```
 Command :
1
Review topology matrix:
   |  1  2  3  4  5  6  7  8
-------------------------------------
 1 | -1 -1 -1 -1 -1 -1 -1 -1
 2 | -1  0  9 -1 -1 -1 -1  3
 3 | -1  9  0 10 -1 18 -1 -1
 4 | -1 -1 10  0 20  3 -1 -1
 5 | -1 -1 -1 20  0  7 -1 -1
 6 | -1 -1 18  3  7  0 13 -1
 7 | -1 -1 -1 -1 -1 13  0  4
 8 | -1  3 -1 -1 -1 -1  4  0
```

Since here there are 8 routers if the user enters a value more than the number routers in the input in option 3 or in option 4, then the message “Node does not exist in the network” is displayed and also the user is asked to enter the new node.
Here we have specified the router as 9.

```
 Command :
3
Enter Source Node:
9
 node dosent exist in network
 Enter new source node
```

```
 Command :
4
Enter Destination Node:
9
 node dosent exist in network
 Enter new Destination node
```

If the user tries to remove all the nodes in the matrix, here the input consists of 8 nodes. Then the user is allowed to remove upto 7 nodes. After removing 7 nodes there are no more operations possible. So a suitable message is displayed for this constraint and the operation is terminated.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
5
Enter Node to remove:
7
 Node7 removed from network
Review topology matrix:
-1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 0

 only 1 node in network
No more operations possible

Exit CS542 project.  Good Bye!
```

If the users tries to remove the node which is not their in the input file then the message “Invalid Node” is displayed. Here since we are considering 8 nodes, node 12 is not there in the input.

```
 Command :
5
Enter Node to remove
12
Invalid node
enter another node
```

- **Option 6:** Exit

```
 Command :
6

Exit CS542 project.  Good Bye!
```

The link simulator displays 6 options as shown. If the user tries to enter a command more than 6, then the message “invalid option” is displayed.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
7
Invalid option
```

- **Considering 10*10 Matrix :**

- **Option 1 :** Create a Network Topology

```
 Command :
1
Review topology matrix:
   |  1  2  3  4  5  6  7  8  9 10
-------------------------------------
 1 |  0  2 -1 -1 -1 -1 -1 -1  1  5
 2 |  2  0  2 -1 -1 -1 -1 -1 -1 -1
 3 | -1  2  0  4 -1 -1 -1 -1 -1 -1
 4 | -1 -1  4  0  2 -1 -1 -1  4  2
 5 | -1 -1 -1  2  0 -1  4 -1 -1 -1
 6 | -1 -1 -1 -1 -1  0  2 -1 -1  5
 7 | -1 -1 -1 -1  4  2  0  6 -1 -1
 8 | -1 -1 -1 -1 -1 -1  6  0  8 -1
 9 |  1 -1 -1  4 -1 -1 -1  8  0 -1
10 |  5 -1 -1  2 -1  5 -1 -1 -1  0
```

- **Option 2:** Build Connection table for all the nodes

```
 Command :
2
Review Connection table:

Router1 connection table
================================
Destination      Interface
  1                -
  2                2
  3                2
  4                9
  5                9
  6                10
  7                9
  8                9
  9                9
  10                10

Router2 connection table
================================
Destination      Interface
  1                1
  2                -
  3                3
  4                3
  5                3
  6                1
  7                3
  8                1
  9                1
  10                1
```

```
Router3 connection table
================================
Destination      Interface
  1                2
  2                2
  3                -
  4                4
  5                4
  6                4
  7                4
  8                2
  9                2
  10                4

Router4 connection table
================================
Destination      Interface
  1                9
  2                3
  3                3
  4                -
  5                5
  6                10
  7                5
  8                9
  9                9
  10                10
```

Router5 connection table

Destination	Interface
1	4
2	4
3	4
4	4
5	-
6	7
7	7
8	7
9	4
10	4

Router6 connection table

Destination	Interface
1	10
2	10
3	10
4	10
5	7
6	-
7	7
8	7
9	10
10	10

Router7 connection table

Destination	Interface
1	5
2	5
3	5
4	5
5	5
6	6
7	-
8	8
9	5
10	6

Router8 connection table

Destination	Interface
1	9
2	9
3	9
4	9
5	7
6	7
7	7
8	-
9	9
10	7

Router9 connection table

Destination	Interface
1	1
2	1
3	1
4	4
5	4
6	1
7	4
8	8
9	-
10	1

Router10 connection table

Destination	Interface
1	1
2	1
3	4
4	4
5	4
6	6
7	6
8	6
9	4
10	-

- **Option 3:** Build connection table for the given source node

```
 Command :
3
Enter Source Node:
1
Router   Interface   Path
1         -          -
2         2          1-->2
3         2          1-->2-->3
4         9          1-->9-->4
5         9          1-->9-->4-->5
6         10          1-->10-->6
7         9          1-->9-->4-->5-->7
8         9          1-->9-->8
9         9          1-->9
10         10          1-->10
```

- **Option 4:** Shortest path to the destination router

```
 Command :
4
Enter Destination Node
5
Router1 to Router5
Cost  = 7
Path  = 1-->9-->4-->5
```

If option 4 is chosen before choosing option 3

```
 Command :
4
you have to choose choice 3 before choice 4
```

- **Option 5:** Modify a Topology

```
 Command :
5
Enter Node to remove
2
 Node2 removed from network
```

If you try to remove the already removed router, then a message is displayed and it prompts to enter the another node to be removed.

```
 Command :
5
Enter Node to remove
2
Node already removed
```

If you try to display the connection table for the already removed node then the message “Node does not exist in the network is displayed and prompts the user to enter the new source node

```
 Command :
3
Enter Source Node:
2
 node dosent exist in network
 Enter new source node
```

Also the same thing applies for finding the shortest path (for both source and destination).
If a node which has already been removed is used to find the shortest path by using option 3 or option 4 then the message is displayed.

```
 Command :
4
Enter Destination Node
2
 node dosent exist in network
 Enter new Destination node
```

Since here there are 10 routers if the user enters a value more than the number routers in the input in option 3 or in option 4, then the message “Node does not exist in the network” is displayed and also the user is asked to enter the new node.
Here we have specified the router as 11.

```
 Command :
3
Enter Source Node:
11
 node dosent exist in network
 Enter new source node
```

```
 Command :
4
Enter Destination Node
11
 node dosent exist in network
 Enter new Destination node
```

If the user tries to remove all the nodes in the matrix, here the input consists of 10 nodes. Then the user is allowed to remove upto 9 nodes. After removing 9 nodes there are no more operations possible. So a suitable message is displayed for this constraint and the operation is terminated.

```
 Command :
5
Enter Node to remove
9
 Node9 removed from network
Review topology matrix:
   |  1  2  3  4  5  6  7  8  9 10
-------------------------------------
 1 | -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 2 | -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 3 | -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 4 | -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 5 | -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 6 | -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 7 | -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 8 | -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 9 | -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
10 | -1 -1 -1 -1 -1 -1 -1 -1 -1  0

 only 1 node in network
No more operations possible

Exit CS542 project.  Good Bye!
```

If the users tries to remove the node which is not their in the input file then the message “Invalid Node” is displayed. Here since we are considering 8 nodes, node 12 is not there in the input.

```
 Command :
5
Enter Node to remove
12
Invalid node
enter another node
```

- Option 6 : Exit

```
 Command :
6

Exit CS542 project.  Good Bye!
```

The link simulator displays 6 options as shown. If the user tries to enter a command more than 6, then the message “invalid option” is displayed.

```
 CS542 Link State Routing Simulator  :
(1) Create a Network Topology
(2) Build a Connection Table for all nodes
(3) Build a Connection Table for given source node
(4) Shortest Path to Destination Router [choose only if last option was 3]
(5) Modify a topology
(6) Exit

 Command :
7
Invalid option
```

- **References :**

- https://en.wikipedia.org/wiki/Routing
- http://www.webopedia.com/TERM/R/routing.html
- http://www.omnisecu.com/cisco-certified-network-associate-ccna/introduction-to-link-state-routing-protocols.php
- http://www.danscourses.com/CCNA-2/link-state-routing-protocols.html
- https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#Description
- https://www.cs.auckland.ac.nz/software/AlgAnim/dijkstra.html
- http://math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Melissa.pdf