CS542 Computer Networks 1: Fundamentals

Project Report

LINK STATE ROUTING PROTOCOL SIMULATOR

Jayanth Vangari

(A20337867)

CONTENTS

Figures page no:

1. Objective:

The project simulates Link State Routing protocol, basically implementing 2 functions:

1. Creating a Forwarding Table for any selected node.

2. Calculate Optimal Cost Path between any 2 nodes, source and destination nodes namely.

2. Project Description:

The project includes a network topology matrix that consists of costs of links between any of the directly connected routers, the matrix is fed as an input file to the program implementing the Link State Routing Protocol. The input file can contain network information of arbitrary number of routers. It is assumed that each router knows its own information and has no knowledge about others routers in the network.

Link State Routing Protocol:

The link State Routing protocol is performed by every node i.e., routers in the network. In The Link-State routing, every node runs the algorithm and compute least cost paths to every other node in the network. The algorithm that is used in the project is Dijkstra's algorithm. Dijkstra's algorithm is essentially iterative and is such that after *kth* iteration, least cost paths to k destination nodes are known. The costs known are the smallest possible costs from the node called source node. The algorithm takes connectivity between all the nodes and all the link costs as input.

In the project, Dijkstra's algorithm is used to find shortest path between any two routers i.e., source and destination nodes namely. The project allows you to output the Forwarding Table for any selected node and also outputs the shortest cost path between the source and destination nodes.

3. Scope:

The scope of the project is to provide the shortest cost path for any two routers using the Dijkstra's algorithm taking Network Topology file as input.

The program works for any arbitrary number of routers.

The algorithm is efficient for finding shortest path between nodes with asymmetric links and for implementing functions of the algorithm, JAVA is used.

4. Design and Algorithm Description:

The project deals with link state routing protocol for finding the shortest cost path between source and destination nodes using the Dijkstra's algorithm. The goal is to output the forwarding table for any selected node, and output the least cost path between two selected nodes.

The Network topology is a graph with arbitrary number of routers(nodes) that are connected and each link connecting two nodes is associated with a cost, that determines which minimum cost path is to the destination node.

In this project, the assumption is that Every node in the network knows its own information and is unaware of the information about other nodes,

The project includes various modules:

1. Input Matrix

2. Forwarding Table

3. Least Cost Path

Input Matrix:

The Network topology graph is fed into the input matrix. The input matrix consists of only the costs of links between the nodes. If two nodes are directly connected, the value in the matrix is the cost between those two nodes, if the nodes are not connected the cost of the link is considered to be -1 and the cost of the link is 0 if the source and destination node are the same.

Forwarding Table:

The Dijkstra's algorithm gives a least cost path to every other node in the network from a single node called the source node. The algorithm is used to find the forwarding table for every node in the network. The Forwarding table for a node gives the interface chosen for the initial hop to reach every other possible destination node in the network. Each node is associated with its own forwarding table.

Least Cost Path:

The least Cost Path module provides an optimal cost path between any two selected nodes, using the Dijkstra's algorithm. The optimal cost path is a path traced from a source node to destination node with minimum cost.

4.1 Algorithm:

The algorithm applied in the project is below:

```
Initialization:
add x (source node) to M
for all nodes y
if y adjacent to x
then D(y) = cost(x,y)
else D(y) = -1
temp : =x
flag[x] =1 /* x is visited */
for all nodes y
        flag[y]= 0 /* node is unvisited */
while list M is not full /* all nodes are not added to M */
        For all nodes that are not visited yet
                min : = node y with minimum D[y]
                temp : = y
        add temp to M
        flag[temp]:= 1 /* node is visited */
        for each neighbor y of temp  that aren't visited :
                If min+cost(temp,y)<D(y)
        D(y) = min+cost(temp,y) /* new cost to y is either old cost to y or know
                        shortest path cost to r plus cost from r to y */
                        predecessor[y]: = temp
```

```
22                  end if
23            end for
24      end while
24 return predecessor [ ]
```

M is a list initially with only source node, and if least cost path is found to a node from source node x,

Then the node is added to M.

cost(u ,v) gives the cost of the link between the directly connected nodes ‘u’ and ‘v’.

D(y) gives the cost of the link from source node ‘x’ to the node ‘y’.

Predecessor[] is the predecessor array that gives the predecessor node for the destination node , that is used to trace back the path to the source node from the destination node.

Flag[] is a flag array, that keep tracks of nodes that are visited,

flag[]=1 indicates that the node is visited. If not visited flag[]=0

The Dijkstra’s algorithm computes a sink tree on the graph.

Each link in the graph is associated with a non-negative cost.

The algorithm begins with initialization with only the source node ‘x’ added to the List M.

For any node ‘y’ if it is adjacent to source node then D(y) =cost(x ,y) i.e., cost of the link x and y.

If ‘y’ is not directly connected to the source node ‘x’ D(y) is initialized to -1.

The next Step, involves finding least cost path from the node ‘x’ to all other nodes in the network. It involves updating the least cost path D(y).

D(y) =min (D(y), D(temp) +cost (temp,y)) updates the least cost path D(y) and gives the shortest cost path from node x to node ‘y’ in the network.

At *k*th iteration, the algorithm gives the shortest or least cost path for *k* nodes and add the nodes for which the least cost path has been found to the list M.

At the end of the algorithm, the minimum cost path is computed from node ‘x’ to every other possible destination node.

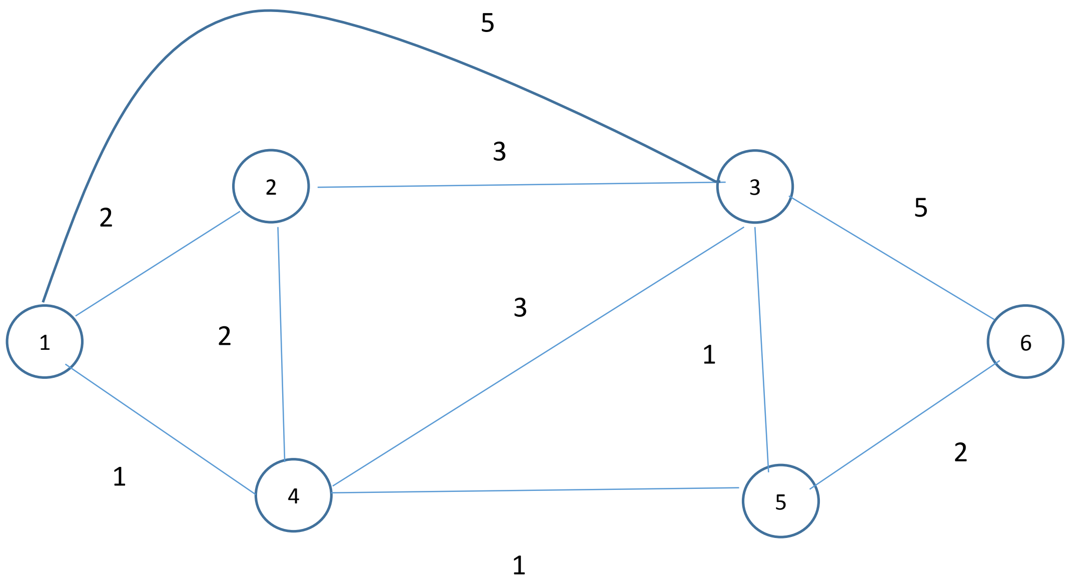


Figure 4.1.1 Network Topology graph

Consider node ‘1’ as source node for which the above algorithm is applied.

Initial Step:

M = {1}

temp = 1

flag[1]=1

For all nodes adjacent to ‘1’ D(y) = cost(1,y) otherwise D(y) = -1

D(2) = 2

D(3) =5

D(4) = 1

D(5) = -1

D(6) = -1

Step-2:

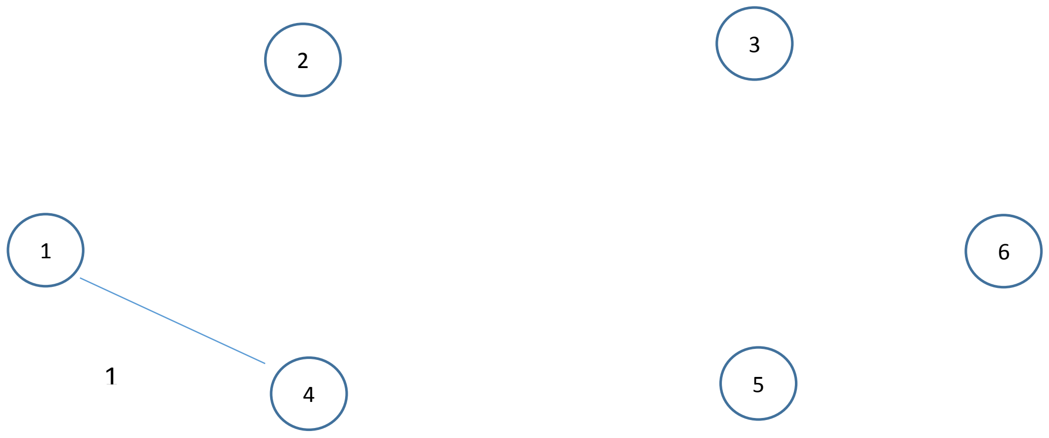
Finding the adjacent node that has least cost path from the node ‘1’ and incorporate that node to M.

D(4) = 1

temp =4

flag[4]=1

M = {1, 4}



Step-3:

Update the least-cost path and finding the next node with the least cost path from node ‘1’.

D(2) = min((D(2),D(4)+cost(4,2))

=2

D(3) = min((D(3),D(4)+cost(4,3))

=4

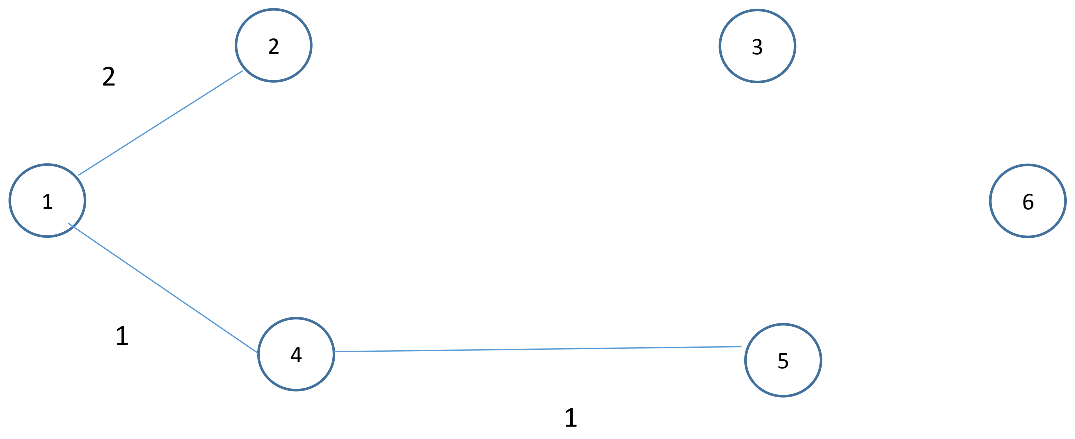
D(5) = min((D(5),D(4)+cost(4,5))

=2

M = {1, 2, 4, 5}

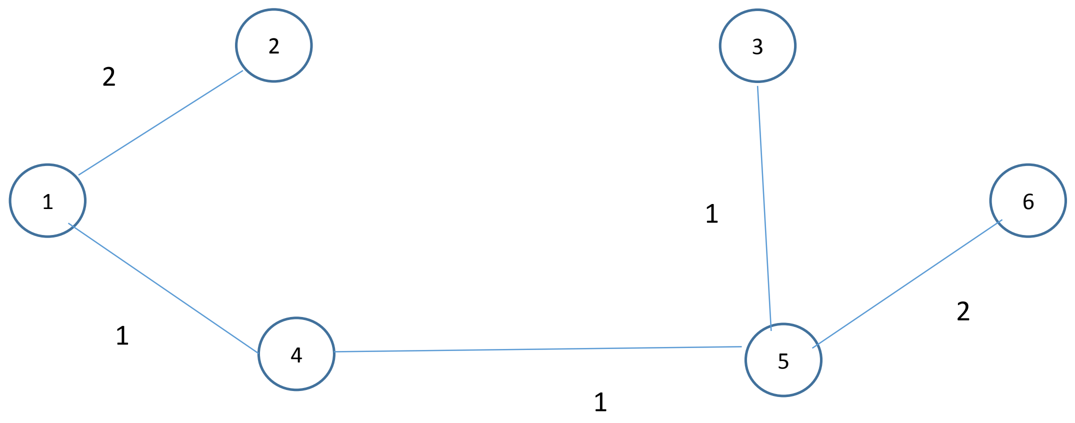
flag[2]=1

flag[5]=1



Repeating the loop until all the nodes are added to M and their least cost paths are found.

The algorithm produces a sink tree as below



The algorithm produces least cost paths from node '1' to all other possible nodes in the graph

The updated least cost values from node '1' are:

D(2) =2

D(3) =3

D(4) =1

D(5) =2

D(6) =4

5. Implementation:

The project simulating the link state routing protocol performing two functions of creating a forwarding table for selected node and finding the least cost path for any two nodes sis implemented in java.

The LinkStateAlgo class implements the link state routing protocol for finding the least cost path between two selected nodes and also create a forwarding table for any selected node

LinkStateAlgo class has 4 methods other than main() method

a. menu()

b. forwardingTable()

c. leastCostPath()

d. connection()

The main() method reads a file provided through console. The input file is a Input Matrix containing the costs of links of the routers (nodes). The contents of the file are read into an array router [ ] [ ] and then menu is displayed at the console.

a. menu() :

This method displays the menu of the functions to be carried out implementing link state routing algorithm. The menu displays two functions that are implemented in this project

1. Forwarding Table

2. Least Cost Path

b. forwardingTable():

This method creates a Forwarding table for the selected node given from the console. The method calls the connection() method implementing the dijkstra's algorithm and then displaying the forwarding table consisting of the all possible destination nodes and the interface chosen for the initial hop to get to that destination node.

c. leastCostPath():

This method traces an optimal cost path from the source to destination node. It takes the source node and destination node inputs from the console and computes the shortest or least cost path from the source node to destination node along with their cost.

d. connection():

This method is the core part, implementing the Dijkstra's Algorithm (it produces a shortest cost Path tree from a node to every other node) and is called by both methods forwardingTable() And leastCostPath(). The method returns predecessor array that is used by the methods forwardingTable() and leastCostPath() to create forwarding table and least cost path tracing the predecessor array back from destination node to source node

5.1 System Requirements

5.1.1 HARDWARE SPECIFICATION

- RAM -128 Mb or higher
- PROCESSOR-Pentium 133 MHz
- HARD DISK-10GB

5.1.2 SOFTWARE SPECIFICATION

- OPERATING SYSTEM-Windows XP/2000/Vista/Windows 7
- Java

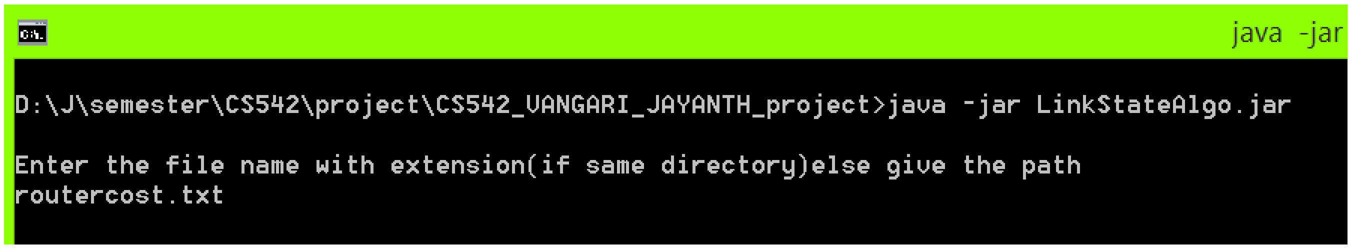
6. Execution steps and Results:

The program is an executable jar file. So, navigate to the java file location and run the jar file from the command prompt.

To run the jar file :

```
java  -jar LinkStateAlgo.jar
```

When the above command is given on the prompt, it asks for the file that is to be given as input as shown in the image below



```
D:\J\semester\CS542\project\CS542_VANGARI_JAYANTH_project>java -jar LinkStateAlgo.jar

Enter the file name with extension(if same directory)else give the path
routercost.txt
```

Figure 6.1 file input

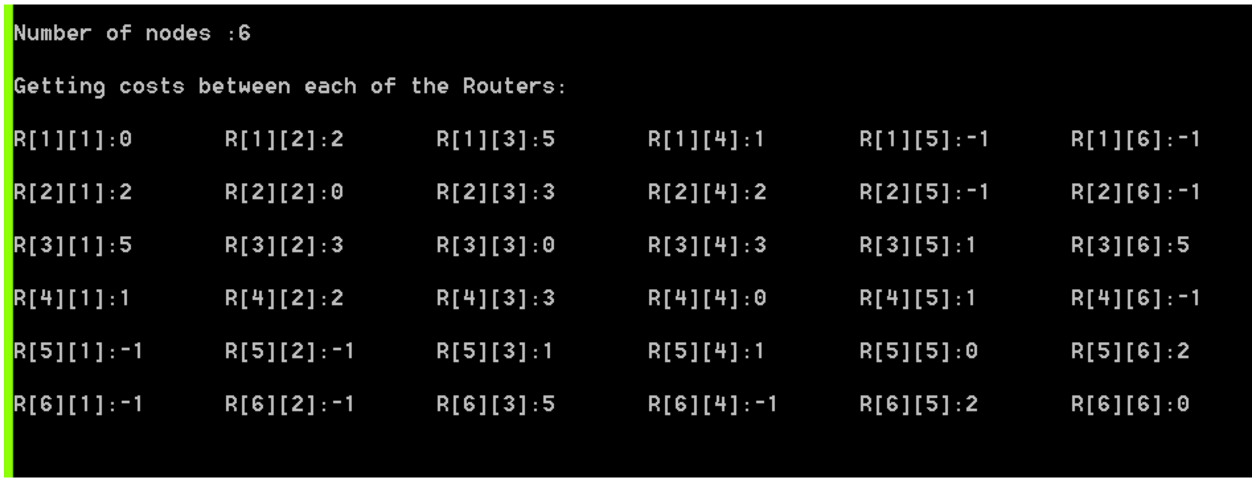
The sample input file, routercost.txt is a text with the contents as cost of links of the nodes, here is the contents of the routercost.txt

routercost.txt

0	2	5	1	-1	-1
2	0	3	2	-1	-1
5	3	0	3	1	5
1	2	3	0	1	-1
-1	-1	1	1	0	2
-1	-1	5	-1	2	0

After the input file is given, it displays the number of nodes in the file and the cost of links of the directly connected nodes.

As shown in the figure below, the cost of the link of 2 nodes is -1 if they are not directly connected. And cost is 0 if source and destination node is same.



```
Number of nodes :6

Getting costs between each of the Routers:

R[1][1]:0       R[1][2]:2       R[1][3]:5       R[1][4]:1       R[1][5]:-1      R[1][6]:-1

R[2][1]:2       R[2][2]:0       R[2][3]:3       R[2][4]:2       R[2][5]:-1      R[2][6]:-1

R[3][1]:5       R[3][2]:3       R[3][3]:0       R[3][4]:3       R[3][5]:1       R[3][6]:5

R[4][1]:1       R[4][2]:2       R[4][3]:3       R[4][4]:0       R[4][5]:1       R[4][6]:-1

R[5][1]:-1      R[5][2]:-1      R[5][3]:1       R[5][4]:1       R[5][5]:0       R[5][6]:2

R[6][1]:-1      R[6][2]:-1      R[6][3]:5       R[6][4]:-1      R[6][5]:2       R[6][6]:0
```

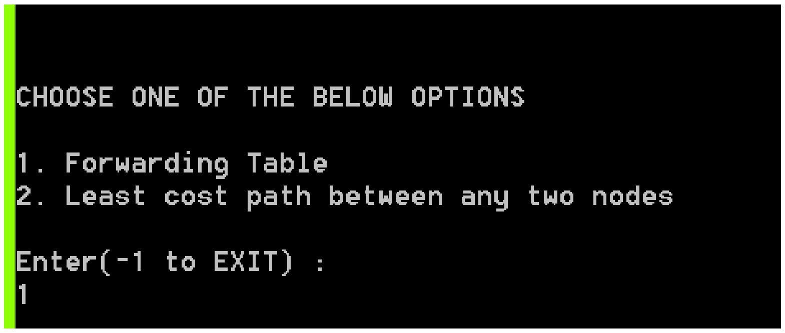
Figure 6.2 : node matrix

And also , the menu is displayed on the console to perform functions as required by the user.

If the user wants the forwarding table for a node, he is expected to enter the option 1.

If the user wants to find the least cost path between any two nodes , he is expected to enter the option 2.

And -1 to exit the console.

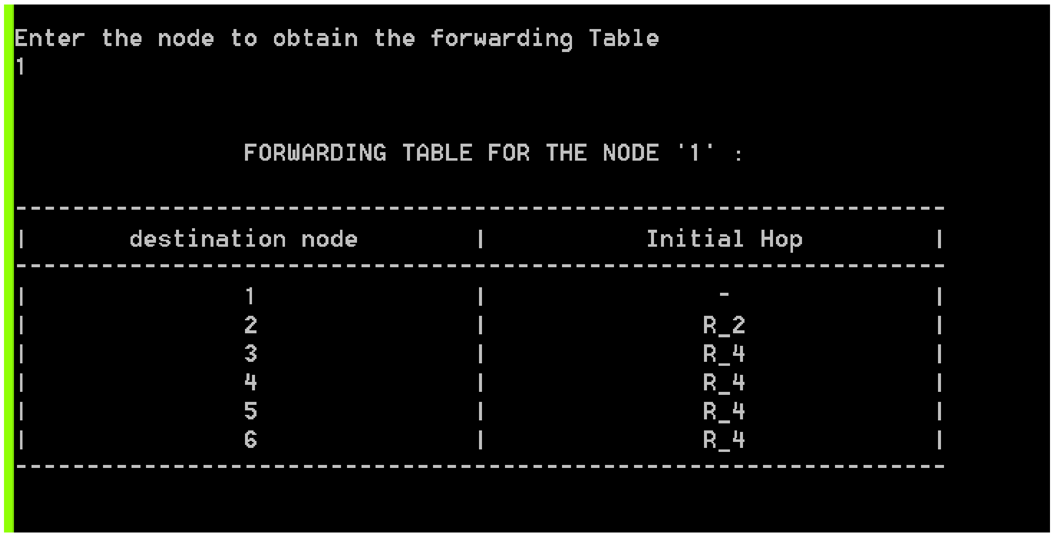


```
CHOOSE ONE OF THE BELOW OPTIONS

1. Forwarding Table
2. Least cost path between any two nodes

Enter(-1 to EXIT) :
1
```

Figure 6.3 menu

If the user chooses to know the forwarding table for any node .

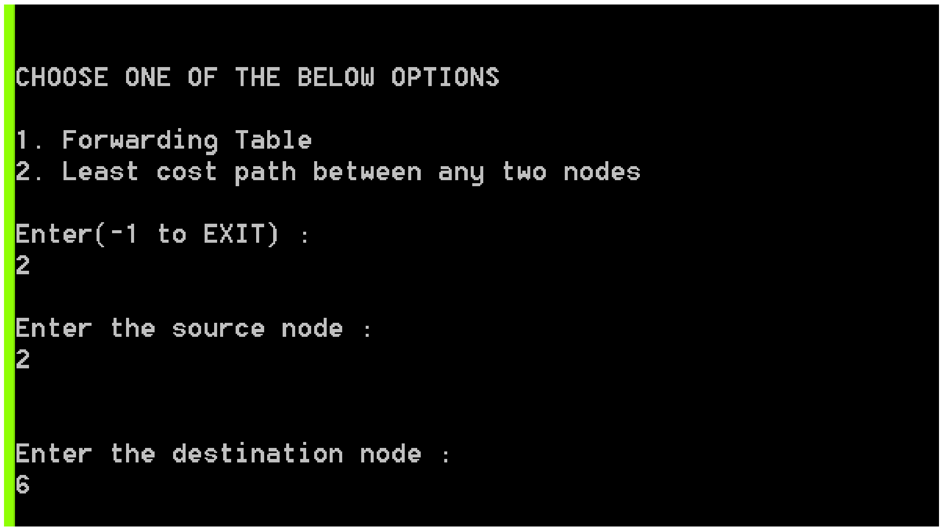
Then user is asked enter the node for which he wants to create the forwarding table and the forwarding table is displayed on the command prompt, as shown in the figure below.



```
Enter the node to obtain the forwarding Table
1


                FORWARDING TABLE FOR THE NODE '1' :

-----------------------------------------------------------------
|       destination node        |           Initial Hop         |
-----------------------------------------------------------------
|               1               |               -               |
|               2               |              R_2              |
|               3               |              R_4              |
|               4               |              R_4              |
|               5               |              R_4              |
|               6               |              R_4              |
-----------------------------------------------------------------
```

Figure 6.4 Forwarding Table

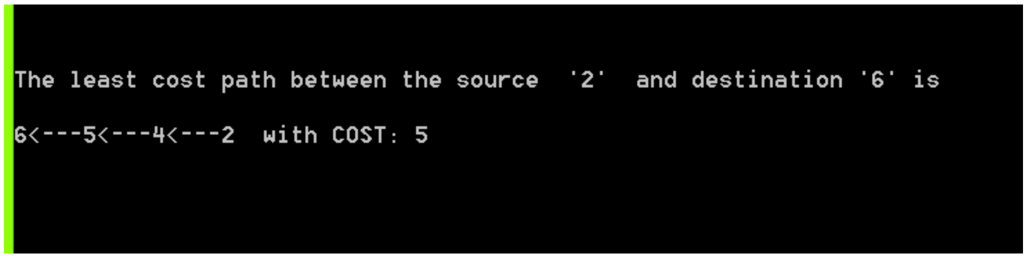
If the user is willing to know the least cost path between any two nodes , then he is expected to enter the source and destination node through the console.



```
CHOOSE ONE OF THE BELOW OPTIONS

1. Forwarding Table
2. Least cost path between any two nodes

Enter(-1 to EXIT) :
2

Enter the source node :
2


Enter the destination node :
6
```

Figure 6.5 Least Cost Path input

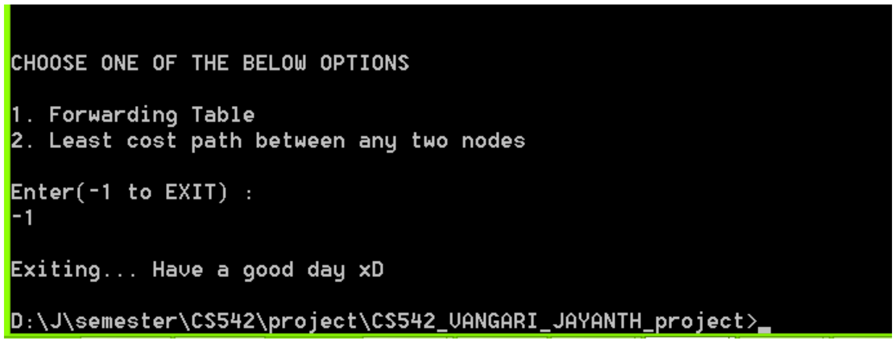
Then, the least cost path from the source node to destination node along with their cost is displayed.



```
The least cost path between the source  '2'  and destination '6' is

6<---5<---4<---2  with COST: 5
```

Figure 6.6 Least Cost Path between 2 nodes

And if user wants to exit, then he is expected to give -1 through the console.



```
CHOOSE ONE OF THE BELOW OPTIONS

1. Forwarding Table
2. Least cost path between any two nodes

Enter(-1 to EXIT) :
-1

Exiting... Have a good day xD

D:\J\semester\CS542\project\CS542_VANGARI_JAYANTH_project>_
```

Figure 6.7 : Exit menu

7. Testing :

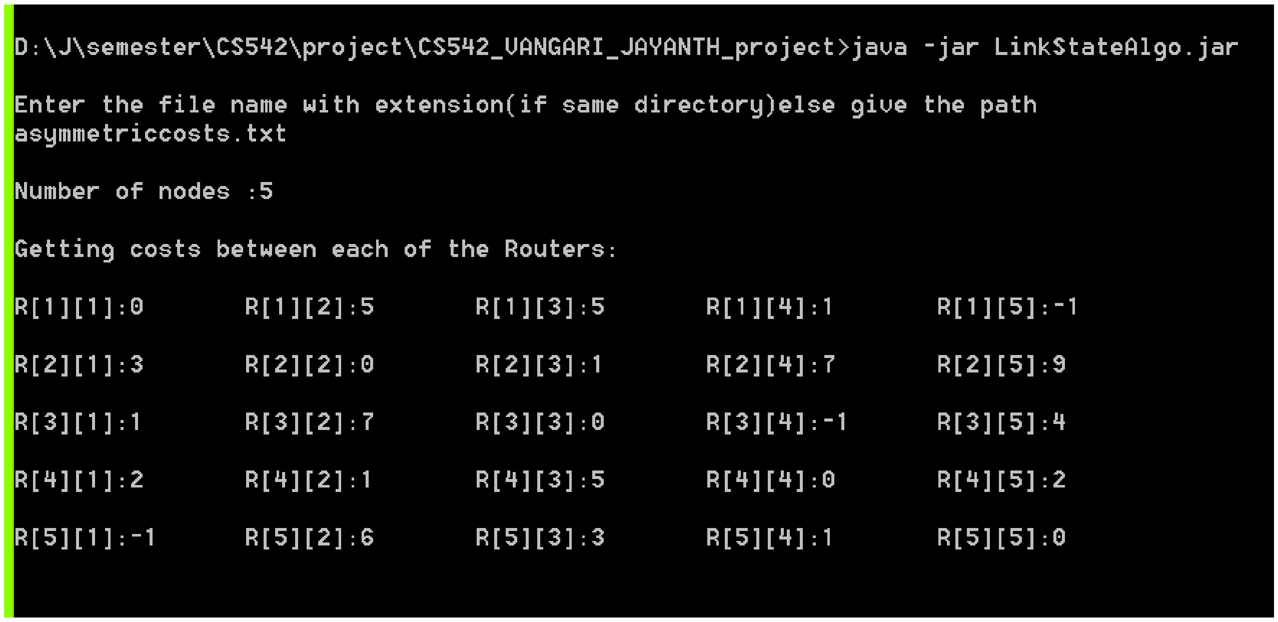
The program simulating the Link State routing protocol can also be used for the nodes with asymmetric links, i.e., the topology graph is directional and the costs between 2 nodes may vary depending on the chosen source and destination node

The input file to be considered for testing the program is

asymmetriccosts.txt

```
0    5    5    1    -1
3    0    1    7    9
1    7    0    -1   4
2    1    5    0    2
-1   6    3    1    0
```

Here's a sample output for the above input file as shown in the figures below



```
D:\J\semester\CS542\project\CS542_VANGARI_JAYANTH_project>java -jar LinkStateAlgo.jar

Enter the file name with extension(if same directory)else give the path
asymmetriccosts.txt

Number of nodes :5

Getting costs between each of the Routers:

R[1][1]:0       R[1][2]:5       R[1][3]:5       R[1][4]:1       R[1][5]:-1

R[2][1]:3       R[2][2]:0       R[2][3]:1       R[2][4]:7       R[2][5]:9

R[3][1]:1       R[3][2]:7       R[3][3]:0       R[3][4]:-1      R[3][5]:4

R[4][1]:2       R[4][2]:1       R[4][3]:5       R[4][4]:0       R[4][5]:2

R[5][1]:-1      R[5][2]:6       R[5][3]:3       R[5][4]:1       R[5][5]:0
```

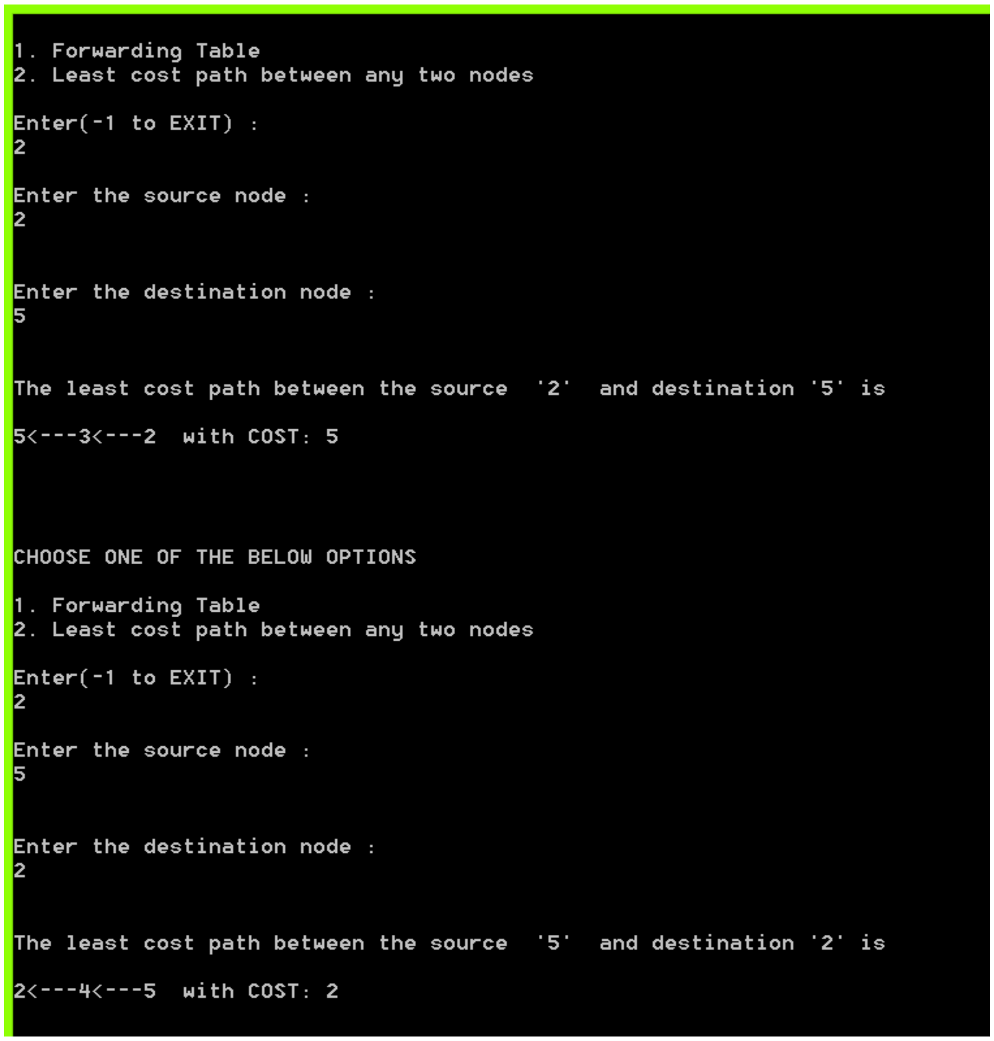
Figure 7.1 Asymmetriccosts.txt file input array

The cost between the node ‘2’ and node ‘5’ is checked, considering each of the nodes as source and destination nodes each time.

The least cost path between the nodes 2 and 5 with source node ‘2’ and destination node ‘5’

is of cost 5, while the least cost path with source node ‘5’ and destination node ‘2’ is of cost 2,with different least cost paths.

The outputs with least cost paths are as shown in the figure below.



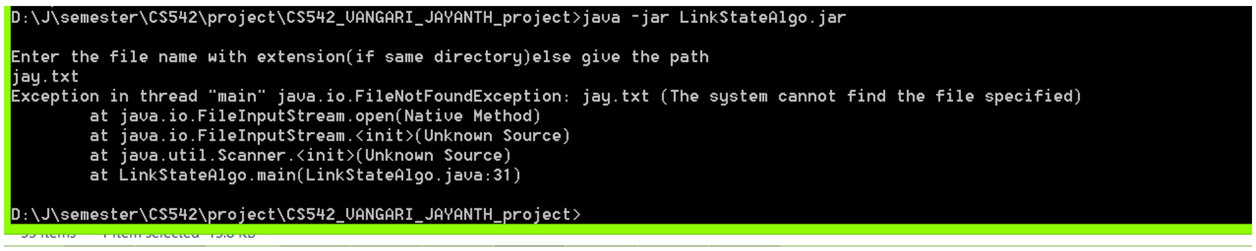
```
1. Forwarding Table
2. Least cost path between any two nodes

Enter(-1 to EXIT) :
2

Enter the source node :
2


Enter the destination node :
5


The least cost path between the source  '2'  and destination '5' is

5<---3<---2  with COST: 5



CHOOSE ONE OF THE BELOW OPTIONS

1. Forwarding Table
2. Least cost path between any two nodes

Enter(-1 to EXIT) :
2

Enter the source node :
5


Enter the destination node :
2


The least cost path between the source  '5'  and destination '2' is

2<---4<---5  with COST: 2
```

Figure 7.2 Asymmetric link Least Cost Path

The other test cases include :

1. Testing for the right input file:

If the user enters an invalid file or gives the wrong path then error is shown at the console as the one shown in the figure below

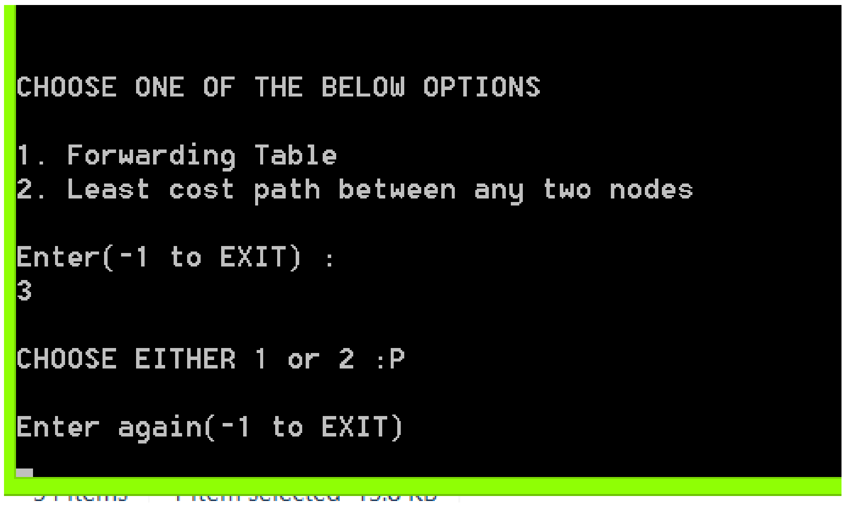


```
D:\J\semester\CS542\project\CS542_VANGARI_JAYANTH_project>java -jar LinkStateAlgo.jar

Enter the file name with extension(if same directory)else give the path
jay.txt
Exception in thread "main" java.io.FileNotFoundException: jay.txt (The system cannot find the file specified)
        at java.io.FileInputStream.open(Native Method)
        at java.io.FileInputStream.<init>(Unknown Source)
        at java.util.Scanner.<init>(Unknown Source)
        at LinkStateAlgo.main(LinkStateAlgo.java:31)

D:\J\semester\CS542\project\CS542_VANGARI_JAYANTH_project>
```

Figure 7.3 Input File error

2. Testing for the wrong input for menu :

The user is expected to enter one of the options displayed on the menu at the console, if the user enters an invalid option , he is asked to enter the right option.

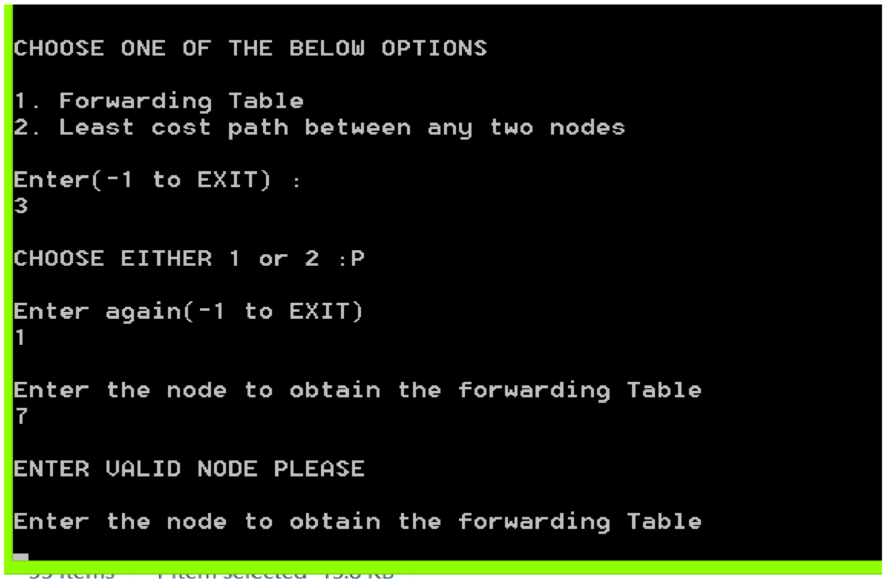


```
CHOOSE ONE OF THE BELOW OPTIONS

1. Forwarding Table
2. Least cost path between any two nodes

Enter(-1 to EXIT) :
3

CHOOSE EITHER 1 or 2 :P

Enter again(-1 to EXIT)
```

Figure 7.4 Wrong input at menu

3. Testing for the wrong input for the forwarding table:

The user is expected to enter the valid to node, to display the forwarding table at the console, if the user is to enter any valid node , he is asked to enter the valid node as shown in the figure below.

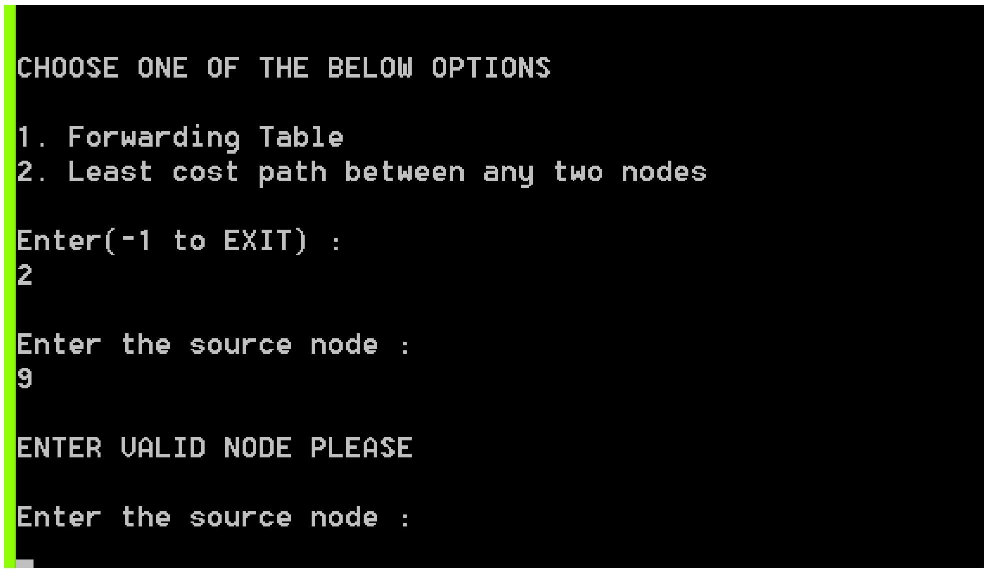


```
CHOOSE ONE OF THE BELOW OPTIONS

1. Forwarding Table
2. Least cost path between any two nodes

Enter(-1 to EXIT) :
3

CHOOSE EITHER 1 or 2 :P

Enter again(-1 to EXIT)
1

Enter the node to obtain the forwarding Table
7

ENTER VALID NODE PLEASE

Enter the node to obtain the forwarding Table
```

Figure 7.5 wrong input node for forwarding table

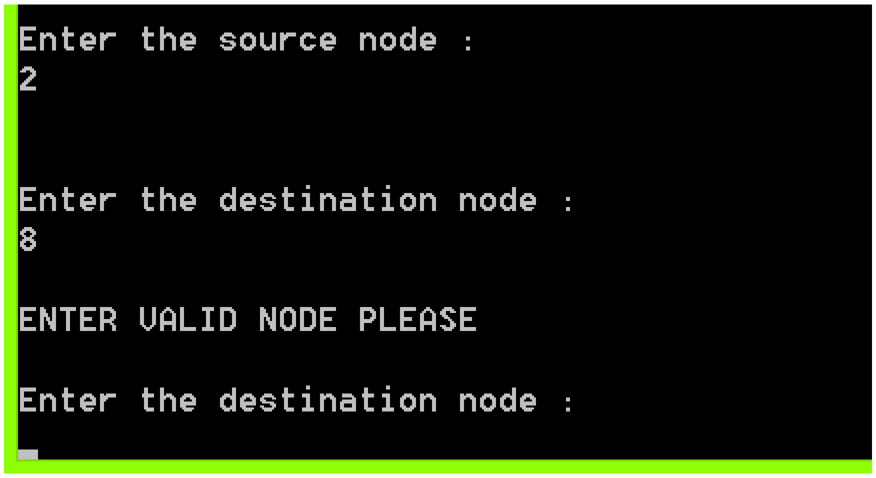
4. Testing for the wrong input for the least cost path :

The user is expected to enter valid nodes for the source and destination node to find the least cost path between any two nodes. If the user enters invalid nodes at the source or destination node, He is asked to enter the valid node again.



```
CHOOSE ONE OF THE BELOW OPTIONS

1. Forwarding Table
2. Least cost path between any two nodes

Enter(-1 to EXIT) :
2

Enter the source node :
9

ENTER VALID NODE PLEASE

Enter the source node :
```

Figure 7.6 Wrong input source node for least cost path

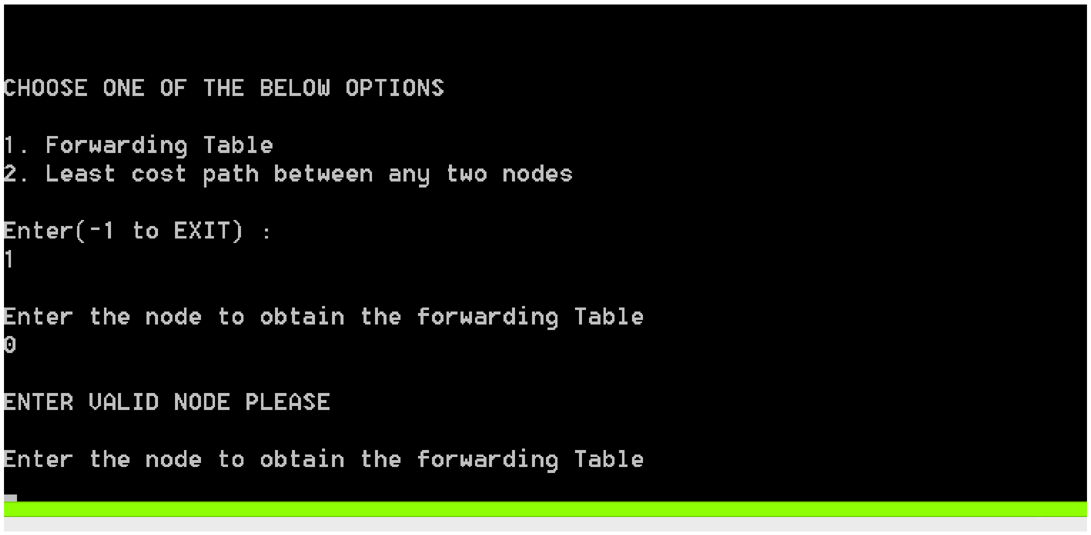


```
Enter the source node :
2


Enter the destination node :
8

ENTER VALID NODE PLEASE

Enter the destination node :
```

Figure 7.7 Wrong input destination node for least cost path

5. Testing for wrong input node '0':

The user is expected enter the nodes from 1 to number of nodes as learnt from the input file matrix. If the user is to give node 0 as input for finding the forwarding table or as a source or destination node for the least cost path. He is asked to enter a valid node at the console.



```
CHOOSE ONE OF THE BELOW OPTIONS

1. Forwarding Table
2. Least cost path between any two nodes

Enter(-1 to EXIT) :
1

Enter the node to obtain the forwarding Table
0

ENTER VALID NODE PLEASE

Enter the node to obtain the forwarding Table
```

Figure 7.8 Invalid input node '0'

8. Conclusion:

- The project simulating the Link state Routing protocol is effective in finding the least cost path between any two nodes and creating a Forwarding table for the selected node .
- The project runs effectively for nodes with asymmetric links.
- The project simulates Link State routing protocol for arbitrary number of routers.

9. References :

1. Computer Networking : A top – down approach by Kurose and Ross, Sixth Edition.

2. http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm