CS542-COMPUTER NETWORKS FUNDAMENTALS-1

Karthikeyan Madhumitha-A20320012

About the Project:

We have a network with arbitrary number of routers. The network topology is given by a matrix, called the original topology (graph) matrix, which only indicates the costs of links between all directly connected routers. We assume each router only knows its own information and has no knowledge about others at the beginning.
To implement Link-State Routing Protocol, first the program creates the state of the links by each router after the input file containing the network information been loaded.By reading the topology matrix file a network graph can be determined. A Dijkstra's algorithm is be applied to find shortest path between two entities: source and destination nodes. Finally, the program outputs the connection table of any router, and output the optimal path between any two selected routers

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, the other being distance-vector routing protocols. Examples of link-state routing protocols include open shortest path first (OSPF) and intermediate system to intermediate system (IS-IS).

The link-state protocol is performed by every switching node in the network (i.e., nodes that are prepared to forward packets; in the Internet, these are called routers). 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.

Dijkstra's algorithm :

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph.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

Let the node at which we are starting be called the initial node. Let the distance of node Y be the distance from the initial node to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

1. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes.
2. Set the initial node as current. Mark all other nodes unvisited. Create a set of all the unvisited nodes called the unvisited set.
3. For the current node, consider all of its unvisited neighbors and calculate their tentative distances. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B (through

A) will be 6 + 2 = 8. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, keep the current value.

4. When we are done considering all of the neighbors of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again.
5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the unvisited set is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
6. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node", and go back to step 3.

PSEUDO-CODE OF THE ALGORITHM

The pseudo-code of Link State Routing using Dijkstras algorithm is as follows

Dijkstra's algorithm.

```
Procedure Dijsktra (V: set of vertices 1... n {Vertex 1 is the source}
        Adj[1…n] of adjacency lists;
        EdgeCost(u, w): edge – cost functions;)
Var: sDist[1…n] of path costs from source (vertex 1); {sDist[j] will be equal to
        the length of the shortest path to j}
Begin:
Initialize
{Create a virtual set Frontier to store i where sDist[i] is already fully solved}
                Create empty Priority Queue New Frontier;
                sDist[1]←0; {The distance to the source is zero}
                forall vertices w in V – {1} do {no edges have been explored yet}
                sDist[w]= infinity
                end for;
        Fill New Frontier with vertices w in V organized by priorities sDist[w];
endInitialize;
repeat
v←DeleteMin{New Frontier}; {v is the new closest; sDist[v] is already correct}
forall of the neighbors w in Adj[v] do
        if sDist[w]>sDist[v] +EdgeCost(v,w) then
                sDist[w]←sDist[v] +EdgeCost(v,w)
                update w in New Frontier {with new priority sDist[w]}
        endif
endfor
until New Frontier is empty
endDijkstra;
```

CS542-COMPUTER NETWORKS FUNDAMENTALS-1

PROGRAM EXECUTION:
The program is compiled and outputs the menu below.The user can give his choice from 1-5.

```
CS542 Link State Routing Simulator

 (1) Create a network topology

 (2) Build a Connection Table

 (3) Shortest Path to Destination Router

 (4) Modify a topology

 (5) Exit
Command:
```

When the user inputs 1, the topology.txt file is read and it is displayed to the user. Internally, it is stored in a matrix. Then it returns to the menu.

The input file, topology.txt has the original topology matrix

0 2 5 1 -1 3 2 1
2 0 8 7 9 1 2 3
5 8 0 -1 4 1 1 1
1 7 -1 0 2 1 2 3
-1 9 4 2 0 2 3 4
3 1 1 1 2 0 2 3
2 2 1 2 3 2 0 -1
1 3 1 3 4 3 -1 0

```
Command:
1

 Input original network topology matix data file:



Review original Topology matrix:
0 2 5 1 -1 3 2 1
2 0 8 7 9 1 2 3
5 8 0 -1 4 1 1 1
1 7 -1 0 2 1 2 3
-1 9 4 2 0 2 3 4
3 1 1 1 2 0 2 3
2 2 1 2 3 2 0 -1
1 3 1 3 4 3 -1 0
```

When the user tries to give options 2,3,4 to perform operations without reading the input. The program doesn't allow it and asks the user to choose option 1 to create the input topology first.

```
 (3) Shortest Path to Destination Router

 (4) Modify a topology

 (5) Exit
Command:

2
Create the Initial Network topology by choosing option 1 first.
CS542 Link State Routing Simulator

 (1) Create a network topology

 (2) Build a Connection Table

 (3) Shortest Path to Destination Router

 (4) Modify a topology

 (5) Exit
Command:

3
Create the Initial Network topology by choosing option 1 first.
```

```
Command:
4
Initial Network topology is not created! Choose option 1 first.
CS542 Link State Routing Simulator

 (1) Create a network topology

 (2) Build a Connection Table

 (3) Shortest Path to Destination Router

 (4) Modify a topology

 (5) Exit
Command:
```

If the user inputs anything other than a number or a number more than 5, the program indicates the user to pick the choice from 1-5 and displays the menu again

```
 (2) Build a Connection Table

 (3) Shortest Path to Destination Router

 (4) Modify a topology

 (5) Exit
Command:
a
You entered a bad input...please input a number between 1 to 5
CS542 Link State Routing Simulator

 (1) Create a network topology

 (2) Build a Connection Table

 (3) Shortest Path to Destination Router

 (4) Modify a topology

 (5) Exit
Command:
6
Please enter your choice between 1 and 5.
```

Extra credit: Minimum initial number of nodes: 8

If the input file has less than 8 routers, the program indicates the user about the minimum criteria and returns to menu again.

```
 (1) Create a network topology

 (2) Build a Connection Table

 (3) Shortest Path to Destination Router

 (4) Modify a topology

 (5) Exit
Command:
1

 Input original network topology matix data file:



Review original Topology 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
Sorry!, The minimum number of input routers must be 8
CS542 Link State Routing Simulator

 (1) Create a network topology

 (2) Build a Connection Table

 (3) Shortest Path to Destination Router

 (4) Modify a topology

 (5) Exit
Command:
```

Extra credit: Create a connection table of each node as default and display all

When the user inputs option 2, it outputs the routing table of all the routers using dijstra's algorithm by default and asks for a source router and displays the connection table for that router with the interface to the destination from that source.

```
(2) Build a connection table

 (3) Shortest Path to Destination Router

 (4) Modify a topology

 (5) Exit
Command:
2


 The routing table of all routers :
     |
R1   |  0       2       2       1       3       2       2       1
R2   |  2       0       2       2       3       1       2       3
R3   |  2       2       0       2       3       1       1       1
R4   |  1       2       2       0       2       1       2       2
R5   |  3       3       3       2       0       2       3       4
R6   |  2       1       1       1       2       0       2       2
R7   |  2       2       1       2       3       2       0       2
R8   |  1       3       1       2       4       2       2       0

 Select a source router:
2

The connection table for router R2 :
 Destination     Interface
=====================================
   R1            R1
   R2            -
   R3            R6
   R4            R6
   R5            R6
   R6            R6
   R7            R7
   R8            R8
CS542 Link State Routing Simulator

 (1) Create a network topology
```

When the user inputs option 3 , the source router is taken from the previous input and the destination router is got from the user. If the previous option is not selected, it gets fresh from the user.The shortest path is found out with dijkstra's algorithm and the cost from the source to destination is calculated and it is outputted to the user. Then it returns to the menu.

```
  R5            R6
  R6            R6
  R7            R7
  R8            R8
S542 Link State Routing Simulator

(1) Create a network topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit
ommand:

elect the destination router:


he shortest path between R2 and R4  is :


R4 <-- R6 <-- R2
Total cost is  : 2
S542 Link State Routing Simulator

(1) Create a network topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit
ommand:
```

If the user inputs a number more than the number of routers for source or router, the program indicates that to the user and allows them to input the desired router number.

```
2 2 1 2 3 2 0 -1
1 3 1 3 4 3 -1 0
CS542 Link State Routing Simulator

 (1) Create a network topology

 (2) Build a Connection Table

 (3) Shortest Path to Destination Router

 (4) Modify a topology

 (5) Exit
Command:
3
Select a source router:
2
Select the destination router:
9
Please input a number between 1 to 8
Select the destination router:
6

The shortest path between R2 and R6  is :


 R6 <-- R2
 Total cost is  : 1
CS542 Link State Routing Simulator
```

When the user gives the input option 4 , it takes asks for the router to be removed, if the user inputs a router more than the number of routers or a non integer, it doesn't allow and re asks it to the user. The source and destination routers are taken from the previous options and the new topology matrix , connection table from the source to destination, the new path and cost are displayed automatically.

```
Command:
4
 Select a router to be removed:
4
Modified topology
0 2 5 -1 3 2 1
2 0 8 9 1 2 3
5 8 0 4 1 1 1
-1 9 4 0 2 3 4
3 1 1 2 0 2 3
2 2 1 3 2 0 -1
1 3 1 4 3 -1 0


 The routing table of all routers :
     |
R1   |  0       2       2       5       3       2       1
R2   |  2       0       2       3       1       2       3
R3   |  2       2       0       3       1       1       1
R4   |  5       3       3       0       2       3       4
R5   |  3       1       1       2       0       2       2
R6   |  2       2       1       3       2       0       2
R7   |  1       3       1       4       2       2       0
The connection table for router R3 :
 Destination     Interface
====================================
   R1            R7
   R2            R5
   R3            -
   R4            R5
   R5            R5
   R6            R6
   R7            R7

The shortest path between R3 and R4  is :


 R4 <-- R5 <-- R3
```

```
 (5) Exit
Command:
4
 Select a router to be removed:
a
You entered a bad input...please input a number between 1 to 8
 Select a router to be removed:
9
We can remove routers only between 1 and 8
 Select a router to be removed:
2
Modified topology
0 5 1 -1 3 2 1
5 0 -1 4 1 1 1
1 -1 0 2 1 2 3
-1 4 2 0 2 3 4
3 1 1 2 0 2 3
2 1 2 3 2 0 -1
1 1 3 4 3 -1 0


 The routing table of all routers :
     |
R1   |  0       2       1       3       2       2       1
R2   |  2       0       2       3       1       1       1
R3   |  1       2       0       2       1       2       2
R4   |  3       3       2       0       2       3       4
R5   |  2       1       1       2       0       2       2
R6   |  2       1       2       3       2       0       2
R7   |  1       1       2       4       2       2       0

 Select a source router:
2


The connection table for router R2 :
 Destination     Interface
====================================
   R1            R7
   R2            -
   R3            R5
   R4            R5
   R5            R5
   R6            R6
   R7            R7
Select the destination router:
4

The shortest path between R2 and R4  is :


 R4 <-- R5 <-- R2
 Total cost is  : 3
CS542 Link State Routing Simulator

 (1) Create a network topology

 (2) Build a Connection Table

 (3) Shortest Path to Destination Router

 (4) Modify a topology

 (5) Exit
Command:
```

When the user selects option 5, the program outputs, Goodbye! and terminates.

```
CS542 Link State Routing Simulator

 (1) Create a network topology

 (2) Build a Connection Table

 (3) Shortest Path to Destination Router

 (4) Modify a topology

 (5) Exit
Command:
5
Good Bye!
```