Implementation of Link-State Routing Protocol

A PROJECT REPORT

Submitted by

Divanji, Sughosh

CWID- A20276930

Bangalore Section

Towards the fulfillment of,

CS 542 Term Project (Fall 2013)

**SYNOPSIS**

**Link-state routing**

A Link-state routing protocol is one of the two main classes of routing protocols used in packet-switched networks for computer communications. Every router in the network performs the link-state protocol.

The basic concept of link-state routing is that every node receives a map of the connectivity of the network, in the form of a graph showing which nodes are connected to which other nodes.

**Dijkstra's algorithm**

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

**1.INTRODUCTION**

Dijkstra’s shortest path algorithm (also known as Link State Algorithm) is currently the most commonly used shortest path algorithm in GIS systems today. Not only is it a popular shortest path algorithm in GIS systems, but also is also widely used in programs made for network modeling. Edsger Dijkstra, who is currently a professor at the University of Texas, developed the algorithm. 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 path problem. It tends to be relatively fast and efficient compared to other algorithms solving the same problem.

**1.1Dijkstra’s Algorithm**

In a link state algorithm or Dijkstra’s algorithm, the network topology and all link costs are known; that is, available as input to the link state algorithm. In practice this is accomplished by having each node broadcast the identities and costs of its attached links to all other routers in the network. This link state broadcast can be accomplished without the nodes having to initially know the identities of all other nodes in the network. A node need only know the identities and costs to its directly attached neighbors; it will then learn about the topology of the rest of the network by receiving link state broadcasts from other nodes. The result of the nodes' link state broadcast is that all nodes have an identical

and complete view of the network. Each node can then run the link state algorithm and compute the same set of least-cost paths as every other node. Dijkstra's algorithm computes the least-cost path from one node (the source, which we will refer to as A) to all other nodes in the network. Dijkstra's algorithm is iterative and has the property that after the kth iteration of the algorithm, the least-cost paths are known to k destination nodes, and among the least-cost paths to all destination nodes, these k paths will have the k smallest costs. Let us define the following notation:

•

c(i,j): link cost from node i to node j. If nodes i and j are not directly connected, then c(i,j) = infinity . We will assume for simplicity that c(i,j) equals c(j,i).

•

D(v): cost of the path from the source node to destination v that has currently (as of this iteration of the algorithm) the least cost

•

p(v): previous node (neighbor of v) along the current least-cost path from the source to v

•

N: set of nodes whose least-cost path from the source is definitively known The link state algorithm consists of an initialization step followed by a loop. The number of times the loop is executed is equal to the number of nodes in the network. Upon termination, the algorithm will have calculated the shortest paths from the source node to every other node in the network.

**1.2 Algorithm for Link State Routing**

1 Initialization:

2 N = {A}

3 for all nodes v

4 if v adjacent to A

5 then D(v) = c(A,v)

6 else D(v) = infinity

7

8 Loop

9 find w not in N such that D(w) is a minimum

10 add w to N

11 update D(v) for all v adjacent to w and not in N:

12 D(v) = min( D(v), D(w) + c(w,v) )

13 /\* new cost to v is either old cost to v or known

14 shortest path cost to w plus cost from w to v \*/

15 until all nodes in N

As an example, compute the least-cost paths from A to all possible destinations. A tabular summary of the algorithm's computation , where each line in the table gives the values of the algorithm's variables at the end of the iteration.

Table 1.1 : Running the link state algorithm on the network in Example2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **step** | **N** | **D(B),p(B)** | **D(C),p(C)** | **D(D),p(D)** |
| 0 | A | 1,A | 7,A | 2,A |
| 1 | AB |  | 21,B | 9,B |
| 2 | ABD |  | 4,D |  |
| 3 | ABDC |  |  |  |

Let's consider the few first steps in detail:

•

In the initialization step, the currently known least-cost path from A to its directly attached neighbors, B, C, and D are initialized to 1, 7, and 2 respectively. Note in particular that the cost to C is set to 7 (even though we will soon see that a lesser-cost path does indeed exist) since this is the cost of the direct (one hop) link from A to C.

•

In the first iteration, we look among those nodes not yet added to the set N and find that node with the least cost as of the end of the previous iteration. That node is B, with a cost of 1, and thus B is added to the set N. Line 12 of the LS algorithm is then performed to update D(v) for all nodes v, yielding the results shown in the second line (step 1) in Table 1. The cost of the path to B is unchanged. The cost of the path to C (which was 7 at the end of the initialization) through node B is found to have a cost of 21. The cost of the path to D (which was 2 at the end of the initialization) through node B is found to have a cost of 9.

Hence the lower cost path i.e direct path from A to D is selected and D's predecessor along the shortest path from A is set to B.

•

In the second iteration, node C is found to have the least path costs (4), and we break the tie arbitrarily and add C to the set N so that N now contains A, B, and D. The cost to the remaining nodes not yet in N, that is, node C is updated via line 12 of the Link State algorithm, yielding the results shown in the third row in the Table.

When the Link State algorithm terminates, we have, for each node, its predecessor along the least-cost path from the source node. For each predecessor, we also have its predecessor, and so in this manner we can construct the entire path from the source to all destinations.

In the first iteration, we need to search through all n nodes to determine the node, w, not in N that has the minimum cost. In the second iteration, we need to check n - 1 nodes to determine the minimum cost; in the third iteration n - 2 nodes, and so on. Overall, the total number of nodes we need to search through over all the iterations is n(n + 1)/2, and thus we say that the above implementation of the link state algorithm has worst-case complexity of order n squared: O(n2).

**2.DIJKSTRA’S ALGORITHM**

Dijkstra's algorithm, named after its discoverer, Dutch computer scientist Edsger Dijkstra, is an algorithm that solves the single-source shortest path problem for a directed graph with nonnegative edge weights.

For example, if the vertices of the graph represent cities and edge weights represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between two cities.

The input of the algorithm consists of a weighted directed graph G and a source vertex s in G. We will denote V the set of all vertices in the graph G. Each edge of the graph is an ordered pair of vertices (u,v) representing a connection from vertex u to vertex v. The set of all edges is denoted E. Weights of edges are given by a weight function w: E . [0, 8]; therefore w(u,v) is the non-negative cost of moving directly from vertex u to vertex v.

The cost of an edge can be thought of as (a generalization of) the distance between those two vertices. The cost of a path between two vertices is the sum of costs of the edges in that path. For a given pair of vertices s and t in V, the algorithm finds the path from s to t with lowest cost (i.e. the shortest path). It can also be used for finding costs of shortest paths from a single vertex s to all other vertices in the graph.

The algorithm works by keeping for each vertex v the cost d[v] of the shortest path found so far between s and v. Initially, this value is 0 for the source vertex s (d[s]=0), and infinity for all other vertices, representing the fact that we do not know any path leading to those vertices (d[v]=infinity for every v in V, except s). When the algorithm finishes, d[v] will be the cost of the shortest path from s to v — or infinity, if no such path exists.

The basic operation of Dijkstra's algorithm is edge relaxation: if there is an edge from u to v, then the shortest known path from s to u (d[u]) can be extended to a path from s to v by adding edge (u,v) at the end. This path will have length d[u]+w(u,v). If this is less than the current d[v], we can replace the current value of d[v] with the new value. Edge relaxation is applied until all values d[v] represent the cost of the shortest path from s to

The algorithm is organized so that each edge (u,v) is relaxed only once, when d[u] has reached its final value. The notion of "relaxation" comes from an analogy between the estimate of the shortest path and the length of a helical tension spring, which is not designed for compression. Initially, the cost of the shortest path is an overestimate, likened to a stretched out spring.

As shorter paths are found, the estimated cost is lowered, and the spring is relaxed. Eventually, the shortest path, if one exists, is found and the spring has been relaxed to its resting length.

The algorithm maintains two sets of vertices S and Q. Set S contains all vertices for which we know that the value d[v] is already the cost of the shortest path and set Q contains all other vertices. Set S starts empty, and in each step one vertex is moved from Q to S. This vertex is chosen as the vertex with lowest value of d[u]. When a vertex u is moved to S, the algorithm relaxes every outgoing edge (u,v).

**2.1 Pseudocode**

In the following algorithm, u := Extract\_Min(Q) searches for the vertex u in the vertex set Q that has the least d[u] value. That vertex is removed from the set Q and returned to the user.

1 function Dijkstra(G, w, s)

2 for each vertex v in V[G] // Initializations

3 d[v] := infinity

4 previous[v] := undefined

5 d[s] := 0

6 S := empty set

7 Q := V[G]

8 while Q is not an empty set // The algorithm itself

9 u := Extract\_Min(Q)

10 S := S union {u}

11 for each edge (u,v) outgoing from u

12 if d[v] > d[u] + w(u,v) // Relax (u,v)

13 d[v] := d[u] + w(u,v)

14 previous[v] := u

If we are only interested in a shortest path between vertices s and t, we can terminate the search at line 9 if u = t.

Now we can read the shortest path from s to t by iteration:

1 S := empty sequence

2 u := t

3 while defined previous[u]

4 insert u to the beginning of S

5 u := previous[u]

Now sequence S is the list of vertices on the shortest path from s to t, or the empty sequence if no path exists.

**3. Design of the program.**

The program uses a C source file to read the original routing file from the input file, compute the routing table for each router and find the shortest path between any source and any destination.

The algorithm of the program is always

1. Display the options for the user to enter.
2. If the choice is 1, ask the user to enter the path and name of the input file.
3. Read a line from the file.
4. Check if the read line contains any invalid data.
5. If the line contains a “-” move to the next element of the string.
6. Check if the current element is a valid number.
7. If yes, then it is a negative number. Move on to the next element.
8. Else, then the string has invalid data. Return False.
9. Check if the string is empty.
10. If yes, then return false.
11. Else proceed.
12. Check if the string element is a valid number or a whitespace.
13. If yes, move on to the next element.
14. If no, return false.
15. Check if we have reached end of string.
16. If yes, return true.
17. Else go back to step a).
18. Convert the input file data from a string to an integer.
19. Split the string with space as the delimiter and store it in a new string. Lets call this string as pch.
20. Get the no. of elements in pch. Store it in size variable.
21. Remove the trailing ‘\n’ character.
22. Check if the first element of pch equals ‘-‘
23. If yes set nflag =1.
24. Check if current element of pch equals ‘\0’.
25. If yes, size=size-1. Break.
26. Convert the data in string format in pch into a valid integer number. Store it in res variable.
27. Check if nflag is set.
28. If yes res=res\*-1
29. Store res as a matrix element cost[j][k].
30. Increment pointer value of string pch.
31. Reset the res and nflag variables.
32. Split the line read from the file with space as delimiter.
33. If EOL, return.
34. Else, go back to step b.)
35. If EOF, return.
36. Else, go back to step A.
37. If the choice is 2,. Check if the file has not been read successfully.
38. If yes, go back to step 1.
39. Else, read the router number starting from 0, for which routing table has to be built.
40. Calculate the router for which the routing table has to be built.
41. For all nodes I in the set of nodes N flag=0, if I adjacent to source V then dist[I]=cost[V][I]. Else, dist[I]=Infinity. Count=1. //Initialization.
42. Loop
43. Min =99. We are assuming that all costs will be <=99.
44. Loop
45. Find a node U such that dist[U] <min and flag[U]=0.
46. Repeat for all nodes N.
47. Flag[U]=1. Increment count.
48. Loop
49. Dist[W]=min(Cost[U,W]+Dist[U],Dist[W])
50. Repeat for all nodes N.
51. Repeat till count<N.
52. Print the routing table for the selected source node.
53. Return.
54. If the option selected is 3, read the source and destination routers.
55. Find the shortest path from source to destination.
56. For all nodes I in the set of nodes N flag=0, if I adjacent to source V then dist[I]=cost[V][I]. Else, dist[I]=Infinity. Count=1. //Initialization.
57. Loop
58. Min =99. We are assuming that all costs will be <=99.
59. Loop
60. Find a node U such that dist[U] <min and flag[U]=0.
61. Repeat for all nodes N.
62. Flag[U]=1. Increment count.
63. Loop
64. Dist[W]=min(Cost[U,W]+Dist[U],Dist[W])
65. If W is equal to the destination node D
66. Insert U in the array path[]. Exit the loop
67. Else, continue.
68. Repeat for all nodes N.
69. Repeat till count<N.
70. Print the set of vertices along the shortest path from source to destination.
71. If option entered is between 1 and 3 go back to step 1.
72. Else, return.

**4. Test Report for sample network file.**

Input file,

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

**Menu**

**1-Load File**

**2-Build Routing Table for Each Router**

**3-Out Optimal Path and Minimum Cost**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

1

**Enter the file name with full path**

/Users/sdivanji/networksample.txt

**File opened succesfully**

**Original routing table is**

**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**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

0

**Routing table for router 0 is**

**0->1 cost =2**

**0->2 cost =5**

**0->3 cost =1**

**0->4 cost =3**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

1

**Routing table for router 1 is**

**1->0 cost =2**

**1->2 cost =7**

**1->3 cost =3**

**1->4 cost =5**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

2

**Routing table for router 2 is**

**2->0 cost =5**

**2->1 cost =7**

**2->3 cost =6**

**2->4 cost =4**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

3

**Routing table for router 3 is**

**3->0 cost =1**

**3->1 cost =3**

**3->2 cost =6**

**3->4 cost =2**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

4

**Routing table for router 4 is**

**4->0 cost =3**

**4->1 cost =5**

**4->2 cost =4**

**4->3 cost =2**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

0 4

**Source is 0 and destination is 4**

**0->3->4->and the cost is 3**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

3 2

**Source is 3 and destination is 2**

**3->0->2->and the cost is 6**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

2 3

**Source is 2 and destination is 3**

**2->4->3->and the cost is 6**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

1 2

**Source is 1 and destination is 2**

**1->0->2->and the cost is 7**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

4 3

**Source is 4 and destination is 3**

**4->3->and the cost is 2**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

4

Program ended with exit code: 0

**5. Test report for additional instances developed**

**A> 3 router topology**

Input file

0 2 -1

2 0 4

-1 4 0

Report:

**Menu**

**1-Load File**

**2-Build Routing Table for Each Router**

**3-Out Optimal Path and Minimum Cost**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

1

**Enter the file name with full path**

/Users/sdivanji/sample.txt

**File opened succesfully**

**Original routing table is**

**0 2 -1**

**2 0 4**

**-1 4 0**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

0

**Routing table for router 0 is**

**0->1 cost =2**

**0->2 cost =6**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

1

**Routing table for router 1 is**

**1->0 cost =2**

**1->2 cost =4**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

2

**Routing table for router 2 is**

**2->0 cost =6**

**2->1 cost =4**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

0 1

**Source is 0 and destination is 1**

**0->1->and the cost is 2**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

2

1

**Source is 2 and destination is 1**

**2->1->and the cost is 4**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

0 2

**Source is 0 and destination is 2**

**0->1->2->and the cost is 6**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

4

Program ended with exit code: 0

B. 4 router topology

Input file,

0 2 -1 6

2 0 4 -1

-1 4 0 1

6 -1 1 0

**Menu**

**1-Load File**

**2-Build Routing Table for Each Router**

**3-Out Optimal Path and Minimum Cost**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

1

**Enter the file name with full path**

/Users/sdivanji/sample.txt

**File opened succesfully**

**Original routing table is**

**0 2 -1 6**

**2 0 4 -1**

**-1 4 0 1**

**6 -1 1 0**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

0

**Routing table for router 0 is**

**0->1 cost =2**

**0->2 cost =6**

**0->3 cost =6**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

1

**Routing table for router 1 is**

**1->0 cost =2**

**1->2 cost =4**

**1->3 cost =5**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

2

**Routing table for router 2 is**

**2->0 cost =6**

**2->1 cost =4**

**2->3 cost =1**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

3

**Routing table for router 3 is**

**3->0 cost =6**

**3->1 cost =5**

**3->2 cost =1**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

0

**Routing table for router 0 is**

**0->1 cost =2**

**0->2 cost =6**

**0->3 cost =6**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

0 3

**Source is 0 and destination is 3**

**0->3->and the cost is 6**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

1

3

**Source is 1 and destination is 3**

**1->0->2->3->and the cost is 5**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

2

3

**Source is 2 and destination is 3**

**2->3->and the cost is 1**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

4

Program ended with exit code: 0

C. 5 router topology

Input file

0 11 14 -1 2

11 0 4 6 -1

14 4 0 -1 4

-1 6 -1 0 11

2 -1 4 11 0

**Menu**

**1-Load File**

**2-Build Routing Table for Each Router**

**3-Out Optimal Path and Minimum Cost**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

1

**Enter the file name with full path**

/Users/sdivanji/networksample.txt

**File opened succesfully**

**Original routing table is**

**0 11 14 -1 2**

**11 0 4 6 -1**

**14 4 0 -1 4**

**-1 6 -1 0 11**

**2 -1 4 11 0**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

0

**Routing table for router 0 is**

**0->1 cost =10**

**0->2 cost =6**

**0->3 cost =13**

**0->4 cost =2**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

1

**Routing table for router 1 is**

**1->0 cost =10**

**1->2 cost =4**

**1->3 cost =6**

**1->4 cost =8**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

2

**Routing table for router 2 is**

**2->0 cost =6**

**2->1 cost =4**

**2->3 cost =10**

**2->4 cost =4**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

3

**Routing table for router 3 is**

**3->0 cost =13**

**3->1 cost =6**

**3->2 cost =10**

**3->4 cost =11**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

4

**Routing table for router 4 is**

**4->0 cost =2**

**4->1 cost =8**

**4->2 cost =4**

**4->3 cost =11**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

0 4

**Source is 0 and destination is 4**

**0->4->and the cost is 2**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

1

4

**Source is 1 and destination is 4**

**1->2->4->and the cost is 8**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

0 3

**Source is 0 and destination is 3**

**0->4->3->and the cost is 13**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

4 3

**Source is 4 and destination is 3**

**4->3->and the cost is 11**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

4

Program ended with exit code: 0

D. 6 Router topology,

**Menu**

**1-Load File**

**2-Build Routing Table for Each Router**

**3-Out Optimal Path and Minimum Cost**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

1

**Enter the file name with full path**

/Users/sdivanji/networksample.txt

**File opened succesfully**

**Original routing table is**

**0 11 14 -1 2 1**

**11 0 4 6 -1 17**

**14 4 0 -1 4 3**

**-1 6 -1 0 11 -1**

**2 -1 4 11 0 4**

**1 17 3 -1 4 0**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

0

**Routing table for router 0 is**

**0->1 cost =8**

**0->2 cost =4**

**0->3 cost =13**

**0->4 cost =2**

**0->5 cost =1**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

1

**Routing table for router 1 is**

**1->0 cost =8**

**1->2 cost =4**

**1->3 cost =6**

**1->4 cost =8**

**1->5 cost =7**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

2

**Routing table for router 2 is**

**2->0 cost =4**

**2->1 cost =4**

**2->3 cost =10**

**2->4 cost =4**

**2->5 cost =3**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

3

**Routing table for router 3 is**

**3->0 cost =13**

**3->1 cost =6**

**3->2 cost =10**

**3->4 cost =11**

**3->5 cost =13**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

4

**Routing table for router 4 is**

**4->0 cost =2**

**4->1 cost =8**

**4->2 cost =4**

**4->3 cost =11**

**4->5 cost =3**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

5

**Routing table for router 5 is**

**5->0 cost =1**

**5->1 cost =7**

**5->2 cost =3**

**5->3 cost =13**

**5->4 cost =3**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

3

5

**Source is 3 and destination is 5**

**3->1->2->5->and the cost is 13**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

2 5

**Source is 2 and destination is 5**

**2->5->and the cost is 3**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

5

Program ended with exit code: 0

E. 10 Router topology

Input file,

0 11 14 -1 2 1 14 3 1 3

11 0 4 6 -1 17 5 -1 4 11

14 4 0 -1 4 3 11 20 7 5

-1 6 -1 0 11 -1 8 -1 3 18

2 -1 4 11 0 4 7 1 12 2

1 17 3 -1 4 0 2 2 -1 -1

14 5 11 8 7 2 0 5 6 24

3 -1 20 -1 1 2 5 0 -1 1

1 4 7 3 12 -1 6 -1 0 6

3 11 5 18 2 -1 24 1 6 0

**Menu**

**1-Load File**

**2-Build Routing Table for Each Router**

**3-Out Optimal Path and Minimum Cost**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

1

**Enter the file name with full path**

/Users/sdivanji/networksample.txt

**File opened succesfully**

**Original routing table is**

**0 11 14 -1 2 1 14 3 1 3**

**11 0 4 6 -1 17 5 -1 4 11**

**14 4 0 -1 4 3 11 21 7 5**

**-1 6 -1 0 11 -1 8 -1 3 18**

**2 -1 4 11 0 4 7 1 12 2**

**1 17 3 -1 4 0 2 2 -1 -1**

**14 5 11 8 7 2 0 5 6 25**

**3 -1 21 -1 1 2 5 0 -1 1**

**1 4 7 3 12 -1 6 -1 0 6**

**3 11 5 18 2 -1 25 1 6 0**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

0

**Routing table for router 0 is**

**0->1 cost =5**

**0->2 cost =4**

**0->3 cost =4**

**0->4 cost =2**

**0->5 cost =1**

**0->6 cost =3**

**0->7 cost =3**

**0->8 cost =1**

**0->9 cost =3**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

1

**Routing table for router 1 is**

**1->0 cost =5**

**1->2 cost =4**

**1->3 cost =6**

**1->4 cost =7**

**1->5 cost =6**

**1->6 cost =5**

**1->7 cost =8**

**1->8 cost =4**

**1->9 cost =8**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

2

**Routing table for router 2 is**

**2->0 cost =4**

**2->1 cost =4**

**2->3 cost =8**

**2->4 cost =4**

**2->5 cost =3**

**2->6 cost =5**

**2->7 cost =5**

**2->8 cost =5**

**2->9 cost =5**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

3

**Routing table for router 3 is**

**3->0 cost =4**

**3->1 cost =6**

**3->2 cost =8**

**3->4 cost =6**

**3->5 cost =5**

**3->6 cost =7**

**3->7 cost =7**

**3->8 cost =3**

**3->9 cost =7**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

4

**Routing table for router 4 is**

**4->0 cost =2**

**4->1 cost =7**

**4->2 cost =4**

**4->3 cost =6**

**4->5 cost =3**

**4->6 cost =5**

**4->7 cost =1**

**4->8 cost =3**

**4->9 cost =2**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

5

**Routing table for router 5 is**

**5->0 cost =1**

**5->1 cost =6**

**5->2 cost =3**

**5->3 cost =5**

**5->4 cost =3**

**5->6 cost =2**

**5->7 cost =2**

**5->8 cost =2**

**5->9 cost =3**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

6

**Routing table for router 6 is**

**6->0 cost =3**

**6->1 cost =5**

**6->2 cost =5**

**6->3 cost =7**

**6->4 cost =5**

**6->5 cost =2**

**6->7 cost =4**

**6->8 cost =4**

**6->9 cost =5**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

7

**Routing table for router 7 is**

**7->0 cost =3**

**7->1 cost =8**

**7->2 cost =5**

**7->3 cost =7**

**7->4 cost =1**

**7->5 cost =2**

**7->6 cost =4**

**7->8 cost =4**

**7->9 cost =1**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

8

**Routing table for router 8 is**

**8->0 cost =1**

**8->1 cost =4**

**8->2 cost =5**

**8->3 cost =3**

**8->4 cost =3**

**8->5 cost =2**

**8->6 cost =4**

**8->7 cost =4**

**8->9 cost =4**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

2

**Please select a router**

9

**Routing table for router 9 is**

**9->0 cost =3**

**9->1 cost =8**

**9->2 cost =5**

**9->3 cost =7**

**9->4 cost =2**

**9->5 cost =3**

**9->6 cost =5**

**9->7 cost =1**

**9->8 cost =4**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

5 8

**Source is 5 and destination is 8**

**5->0->8->and the cost is 2**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

5 9

**Source is 5 and destination is 9**

**5->0->7->9->and the cost is 3**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

3

**Please enter the source and destination router number**

9 5

**Source is 9 and destination is 5**

**9->7->5->and the cost is 3**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

4

Program ended with exit code: 0

1. If the input file contains invalid data

Input file,

0 2 -1 6

2 0 4 -1

-1 4a 0 1

6 -1 1 0

**Menu**

**1-Load File**

**2-Build Routing Table for Each Router**

**3-Out Optimal Path and Minimum Cost**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

1

**Enter the file name with full path**

/Users/sdivanji/sample.txt

**File opened succesfully**

**Input contains invalid data**

**Enter the option between 1 & 3 to execute the program. Enter any other value to exit.**

**6. Instructions on compiling and running the file,**

1. The C source code can be compiled and run from any C source compiler.
2. If the input file is not in the same path as the source code please provide the full file path.
3. In the input file, separate the elements of matrix only by a single space. There should not be any space at the end of line and no extra spaces between the matrix elements.
4. To build the routing table for a particular router enter the router number starting from 0. (0 for R0, 1 for R1 etc.)
5. Once routing table built is for all routers we can find the shortest path between any two routers. Enter the source and destination router numbers starting from 0 (0 for R0, 1 for R1 etc.)