Experiment No. 07

Title: Implementation of Routing Algorithm

Batch: B3 **Roll No.:16010420049** **Experiment No.07**

Aim: To write a program to implement Shortest path Algorithm

Resources Used: Java / Turbo C

Theory:
Network layer is responsible for routing the data from source to destination machine. Routing can have many attributes to be considered while routing the data from source to destination such as delay, distance, no of hops. Data transfer should be done as fast as possible. There can be many routes from source to destination. Routing algorithms are mainly divided into two parts

1. Static algorithms
2. Dynamic Algorithms.

After considering the network topology and architecture the routing table is prepared from the parameter to be considered like delay, distance or no of hops. Form this table, the shortest distance is calculated and the packets are routed through that path. Every router does this. In static methodology the table is constructed only once but in dynamic after some regular interval the table is updated.

There are many algorithms
Static –

- Flooding
- Shortest path algorithm
- Flow based algorithm.

Dynamic

- Distance vector
- Link state routing.

The idea behind the shortest path is to build a graph of the subnet, with each node of the graph representing a router and each arc of the graph representing a communication line (often called a link). To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.

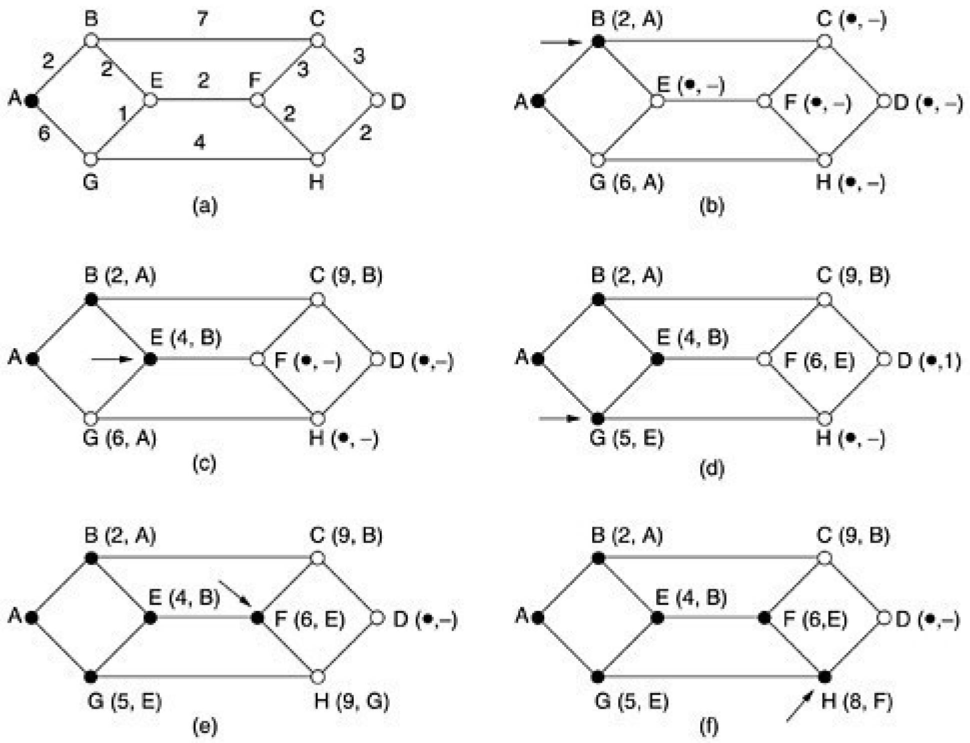
Several algorithms for computing the shortest path between two nodes of a graph are known.

Algorithm:

Dijkstra's Algorithm: Each node is labeled (in parentheses) with its distance from the source node along the best known path. Initially, no paths are known, so all nodes are labeled with infinity. As the algorithm proceeds and paths are found, the labels may change, reflecting better paths. A label may be either tentative or permanent. Initially, all labels are tentative.

When it is discovered that a label represents the shortest possible path from the source to that node, it is made permanent and never changed thereafter.

Figure1. The first five steps used in computing the shortest path from A to D. The arrows indicate the working node.



To illustrate how the labeling algorithm works, look at the weighted, undirected graph of Fig.1(a), where the weights represent, for example, distance. We want to find the shortest path from A to D. We start out by marking node A as permanent, indicated by a filled-in circle. Then we examine, in turn, each of the nodes adjacent to A (the working node), relabeling each one with the distance to A. Whenever a node is relabeled, we also label it with the node from which the probe was made so that we can reconstruct the final path later. Having examined each of the nodes adjacent to A, we examine all the tentatively labeled nodes in the whole graph and make the one with the smallest label permanent, as shown in Fig.1(b). This one becomes the new working node.

We now start at B and examine all nodes adjacent to it. If the sum of the label on B and the distance from B to the node being considered is less than the label on that node, we have a shorter path, so the node is relabeled.

After all the nodes adjacent to the working node have been inspected and the tentative labels changed if possible, the entire graph is searched for the tentatively-labeled node with the smallest value. This node is made permanent and becomes the working node for the next round. Fig 1 shows the first five steps of the algorithm.

To see why the algorithm works, look at Fig(c). At that point we have just made E permanent. Suppose that there were a shorter path than ABE, say AXYZE. There are two possibilities: either node Z has already been made permanent, or it has not been. If it has, then E has already been probed (on the round following the one when Z was made permanent), so the AXYZE path has not escaped our attention and thus cannot be a shorter path.

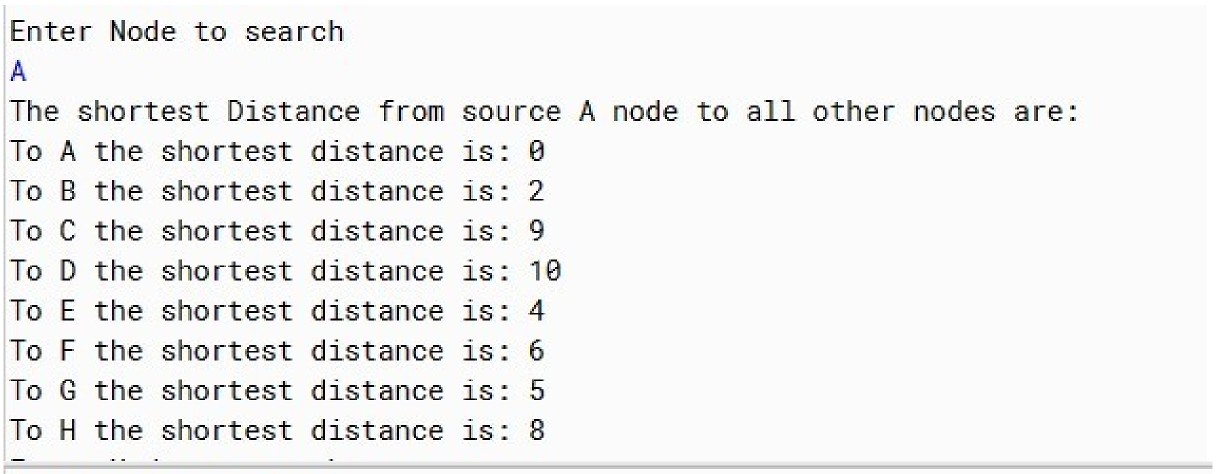
Now **consider t**he case where Z is still tentatively labeled. Either the label at Z is greater than or equal to that at E, in which case AXYZE cannot be a shorter path than ABE, or it is less than that of E, in which case Z and not E will become permanent first, allowing E to be probed from Z.

Program and Output:

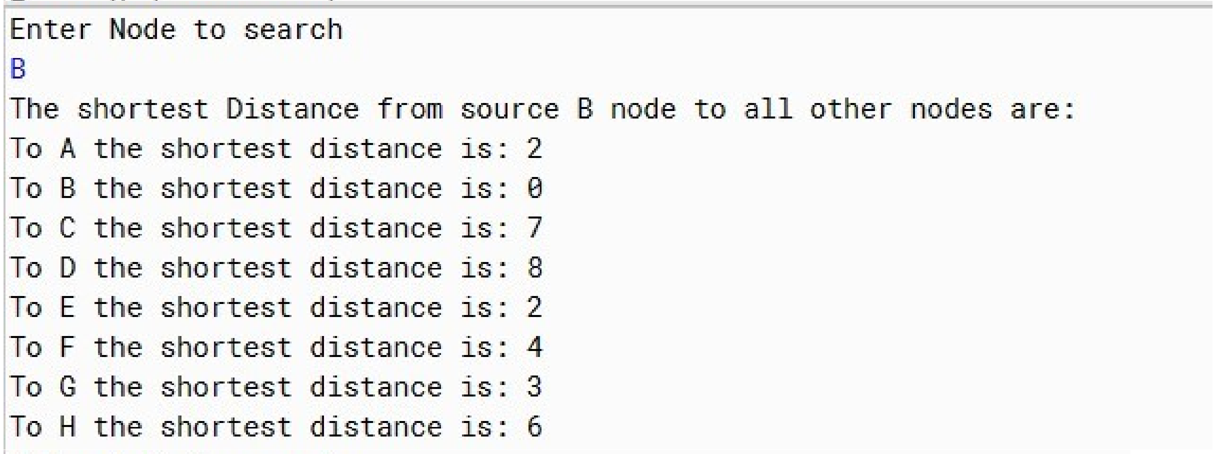
Program:

```
import java.util.*;

public class A
{
   static final int tv = 8;
   int minimumDistance(int d[], Boolean sp[])
   {
      int m = Integer.MAX_VALUE, m_index = -1;

      for (int vx = 0; vx < tv; vx++)
      {
         if (sp[vx] == false && d[vx] <= m)
         {
            m = d[vx];
            m_index = vx;
         }
      }
      return m_index;

   }
   void printSolution(int d[], int n,int p)
   {
      char a[]={'A','B','C','D','E','F','G','H'};
      System.out.println("The shortest Distance from source "+a[p]+" node to all other
nodes are: ");
      for (int j = 0; j < n; j++)
         System.out.println("To " + a[j] + " the shortest distance is: " + d[j]);
   }
   void dijkstra(int graph[][], int s)
   {
      int d[] = new int[tv];
      Boolean sp[] = new Boolean[tv];
      for (int j = 0; j < tv; j++)
      {
         d[j] = Integer.MAX_VALUE;
         sp[j] = false;
      }
      d[s] = 0;
      for (int cnt = 0; cnt < tv - 1; cnt++)
      {
         int ux = minimumDistance(d, sp);
         sp[ux] = true;
         for (int vx = 0; vx < tv; vx++)
            if (!sp[vx] && graph[ux][vx] != -1 && d[ux] != Integer.MAX_VALUE &&
d[ux] + graph[ux][vx] < d[vx])
            {
               d[vx] = d[ux] + graph[ux][vx];
            }
      }
      printSolution(d,tv,s);
   }

   public static void main()
   {
      // arr[x][y] = - 1 means, there is no any edge that connects the nodes x and y directly
      int grph[][] = new int[][] {
            { -1, 2, -1, -1, -1, -1, 6,-1 },
            {  2, -1, 7, -1, 2, -1, -1,-1 },
            { -1, 7, -1, 3, -1, 3, -1, -1 },
            { -1, -1, 3, -1, -1, -1, -1, 2},
            { -1, 2, -1, -1, -1, 2, 1, -1 },
            { -1, -1, 3, -1, 2, -1, 4, 2 },
            { 6, -1, -1, -1, 1, -1, -1, 4 },
            { -1, -1, -1, 2, -1, 2, 4, -1 }};
      A obj = A();
      Scanner sc=new Scanner(System.in);
      System.out.println("Enter Node to search");
      char m=sc.next().charAt(0);
      String ms="ABCDEFGH";
      obj.dijkstra(grph, ms.indexOf(m));
   }
}
```

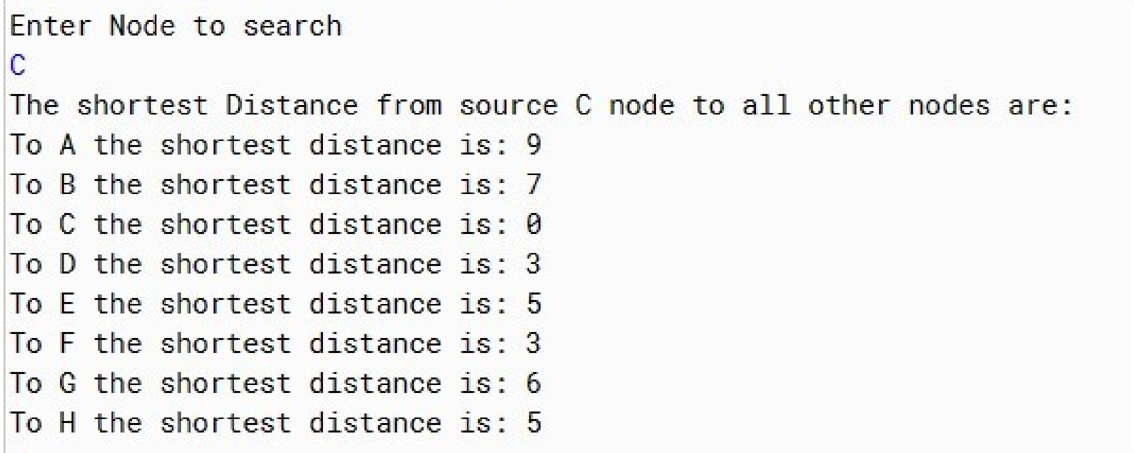
Output:



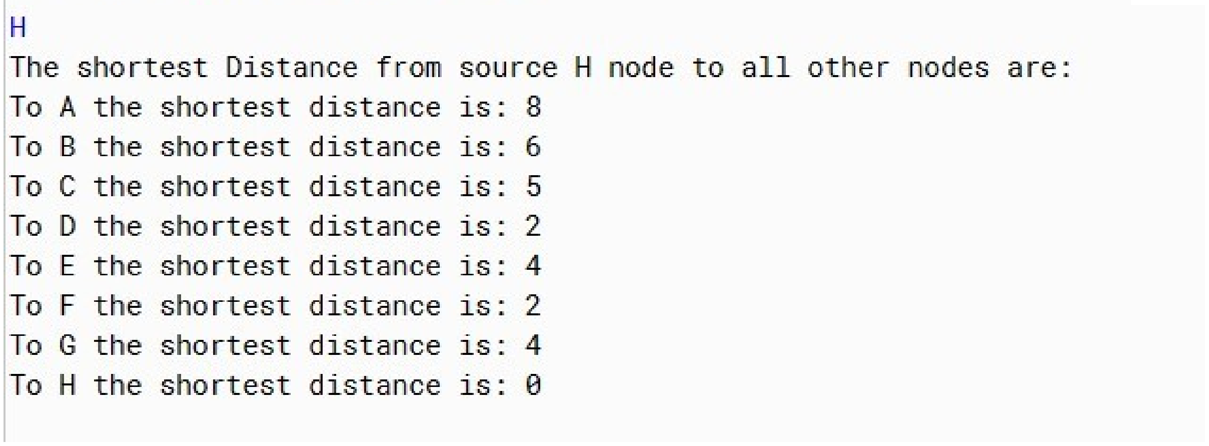
```
Enter Node to search
A
The shortest Distance from source A node to all other nodes are:
To A the shortest distance is: 0
To B the shortest distance is: 2
To C the shortest distance is: 9
To D the shortest distance is: 10
To E the shortest distance is: 4
To F the shortest distance is: 6
To G the shortest distance is: 5
To H the shortest distance is: 8
```



```
Enter Node to search
B
The shortest Distance from source B node to all other nodes are:
To A the shortest distance is: 2
To B the shortest distance is: 0
To C the shortest distance is: 7
To D the shortest distance is: 8
To E the shortest distance is: 2
To F the shortest distance is: 4
To G the shortest distance is: 3
To H the shortest distance is: 6
```



```
Enter Node to search
C
The shortest Distance from source C node to all other nodes are:
To A the shortest distance is: 9
To B the shortest distance is: 7
To C the shortest distance is: 0
To D the shortest distance is: 3
To E the shortest distance is: 5
To F the shortest distance is: 3
To G the shortest distance is: 6
To H the shortest distance is: 5
```



```
H
The shortest Distance from source H node to all other nodes are:
To A the shortest distance is: 8
To B the shortest distance is: 6
To C the shortest distance is: 5
To D the shortest distance is: 2
To E the shortest distance is: 4
To F the shortest distance is: 2
To G the shortest distance is: 4
To H the shortest distance is: 0
```

Questions:

1) The shortest path in routing can refer to

a) The least expensive path
b) The least distant path
c) The path with the smallest number of hops
d) Any or a combination of the above

Ans: **The least expensive path**

2) In Distance Vector Routing each router receives vectors from

a) Every router in the network
b) Every router less than two units away
c) A table stored by the software
d) Its neighbors only

Ans: **Its neighbors only**

3) Link State routing is a dynamic routing algorithm

a) Static
b) Dynamic
c) Both
d) Any

Ans: **Dynamic**

4) In the network layer the packet is frequently called as **Datagram**

a) Message
b) Frame
c) Datagram
d) None of the Above

Ans: **Datagram**

5) What is Traffic Shaping?

Ans: Traffic shaping (also known as packet shaping) is bandwidth management technique that delays the flow of certain types of network packets in order to ensure network performance for higher priority applications. Traffic shaping essentially limits the amount of bandwidth that can be consumed by certain types of applications. It is primarily used to ensure a high quality of service for business-related network traffic.

The most common type of traffic shaping is application-based traffic shaping. Fingerprinting tools are first used to identify the application associated with a data packet. Based on this, specific traffic shaping policies are applied. For example, you might want to use application-based traffic-shaping to throttle peer-to-peer file sharing, while giving maximum bandwidth to a business-critical application such as Voice-over-IP (VoIP), which is especially sensitive to latency.

Many application protocols use encryption to circumvent application-based traffic shaping. To prevent applications from bypassing traffic shaping policies, route-based traffic shaping can be used. Route-based traffic shaping applies packet regulation policies based on the source and intended destination of the previous address a packet.

Outcomes:
Implement Backtracking and Branch-and-bound algorithms

Conclusion:
We have successfully understood and implemented the shortest path algorithm using the program

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:

Books/ Journals/ Websites:

- B. A. Forouzan and Firouz Mosharraf, "Computer Networks ", A Top-Down Approach, Special Indian Edition 2012, Tata McGraw Hill.
- Behrouz A Forouzan, Data Communication and Networking, Tata Mc Graw Hill, India, 4th Edition