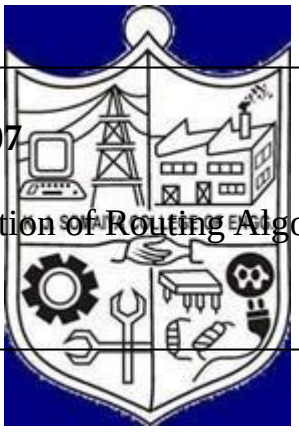


**Experiment No. 07**

**Title:** Implementation of Routing Algorithm

**Batch: B1** **Roll No.: 16010420133** **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.

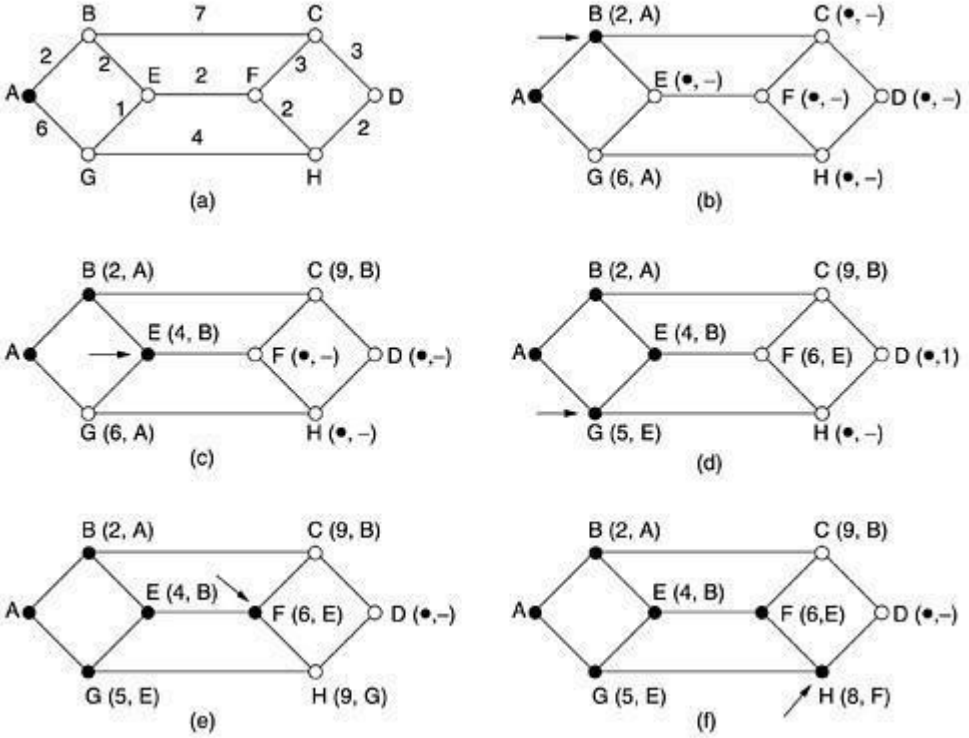
Several algorithms for computing the shortest path between two nodes of a graph are known.

**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.

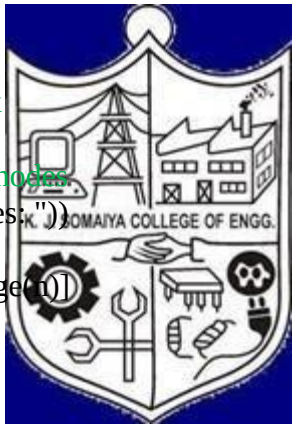
To see why the algorithm works, look at Fig( c). At that point we have just made E permanent. Suppose that there were a shorter path than ABE, say AXYZE. There are two possibilities: either node Z has already been made permanent, or it has not been. If it has, then E has already been probed (on the round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

Python Code:-




```
# Taking input for adjacency matrix
adj_mat = list()
# Taking input for total number of nodes
n = int(input("Enter number of nodes: "))
# 9999 is in place of INFINITY
adj_mat = [[9999 for column in range(n)]
           for row in range(n)]

for i in range(0, n):
    for j in range(i, n):

        if(i == j):
            adj_mat[i][j] = 9999
            continue

        c = int(input(f"Enter cost from node {i + 1} to node {j + 1} (Enter 0 for not connected): "))

        if(c == 0):
            adj_mat[i][j] = 9999
            adj_mat[j][i] = 9999
        else:
            adj_mat[i][j] = c
            adj_mat[j][i] = c

# Taking input for starting node
start = int(input(f"Enter starting node (1 to {n}): "))
start = start - 1

# dist is list which stores distance of each node from starting node
```

```
dist = [adj_mat[start][i] for i in range(0, n)]
# pre stores the predecessor of each node
pre = [start for i in range(0, n)]
# visited stores whether a node has become permanant or not
visited = [0 for i in range(0, n)]

# Setting all three for starting node
dist[start] = 0
visited[start] = 1
count = 1

while count < n-1:

    # Setting min_dist to INFINITY so that all other values are less than this
    min_dist = 9999

    for i in range(0, n):
        if i != start:
            # Setting the min_dist as the distance of node from starting node
            if (dist[i] < min_dist and visited[i] != 1):
                min_dist = dist[i]
                next_node = i
                v = next_node

            # Checking if a better path exists through next_node
            for i in range(0, n):
                if visited[i] != 1:
                    if (min_dist + adj_mat[next_node][i] < dist[i]):
                        dist[i] = min_dist + adj_mat[next_node][i]
                        pre[i] = next_node
    visited[v] = 1
    count += 1

# Printing the distance and the path of each node from starting node
for i in range(0, n):
    if i !=start:
        print(f"\nThe Distance of node {i+1} from {start+1} is: {dist[i]}")
        print(f"Path: {i+1}", end = " ")
        j = i
        j = pre[j]
        print(f"<- {j+1}", end = " ")
        while j != start:
            j = pre[j]
            print(f"<- {j+1}", end = " ")
```

Output:-

```
Enter number of nodes: 4
Enter cost from node 1 to node 2 (Enter 0 for not connected): 10
Enter cost from node 1 to node 3 (Enter 0 for not connected): 0
Enter cost from node 1 to node 4 (Enter 0 for not connected): 2
Enter cost from node 2 to node 3 (Enter 0 for not connected): 3
Enter cost from node 2 to node 4 (Enter 0 for not connected): 0
Enter cost from node 3 to node 4 (Enter 0 for not connected): 4
Enter starting node (1 to 4): 1

The Distance of node 2 from 1 is: 9
Path: 2 <- 3 <- 4 <- 1
The Distance of node 3 from 1 is: 6
Path: 3 <- 4 <- 1
The Distance of node 4 from 1 is: 2
Path: 4 <- 1
```

**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:** d) The least distant 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:** d) It's neighbors only

3) Link State routing is a_____routing algorithm

a) Static
b) Dynamic
c) Both

**Ans:** b) Dynamic

4) In the network layer the packet is frequently called as _____

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

**Ans:** a) Message

5) What is Traffic Shaping?

**Ans:** Traffic shaping is a congestion management method that regulates network data transfer by delaying the flow of less important or less desired packets. It is used to optimize network performance by prioritizing certain traffic flows and ensuring the traffic rate doesn't exceed the bandwidth limit. It is also known as packet shaping.

Regulating the flow of packets into a network is known as data transfer throttling. Regulation of the flow of packets out of a network is known as rate limiting.

---

**Outcome:**

**CO3:** Build the skills of subnetting and routing mechanism.

---

**Conclusion**:

We have implemented the python code to find the shortest path using Dijkstra's Algorithm and this experiment helps us understand this concept better. The program takes appropriate input from user and handles exceptions without errors.

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

**Signature of faculty in-charge with date**

---

s

## 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