Experiment 4: Distance Vector Routing

Aim: To generate routing tables for a network of routers using Distance Vector Routing

Objective: After carrying out this experiment, students will be able to:

- Generate routing tables for a given network using Distance Vector Routing
- Analyze the reasons why Distance Vector Routing is adaptive in nature

Problem statement: You are required to write a program that can generate routing tables for a network of routers. Take the number of nodes and the adjacency matrix as input from user. Your program should use this adjacency matrix and create routing tables for all the nodes in the network. The routing table should consist of one entry per destination. This entry should contain the total cost and the outgoing line to reach that destination.

Analysis: While analyzing your program, you are required to address the following points:

- Why is Distance Vector Routing classified as an adaptive routing algorithm?
- Limitations of Distance Vector Routing

MARKS DISTRIBUTION

Component	**Maximum Marks**	**Marks Obtained**
Preparation of Document	7	
Results	7	
Viva	6	
Total	**20**	

Submitted by: Harshit Kumar

Register No: 21ETMC412011

1. Algorithm/Flowchart

```
Algorithm DistanceVectorRouting
    Step. 1    Start
    Step. 2    repeat for all nodes in graph
    Step. 3    take a node as source
    Step. 4    initializes distances from source to all vertices as infinite and distance to source
               itself as 0.
    Step. 5    Create an array dist[] of size |V| with all values as infinite except dist[src]
    Step. 6    Do following |V|-1 times where |V| is the number of vertices in given graph.
            a.  Do following for each edge u-v
                    i.  If dist[v] > dist[u] + weight of edge uv, then update dist[v]
                    ii. dist[v] = dist[u] + weight of edge uv
    Step. 7    If dist[v] > dist[u] + weight of edge uv, then "Graph contains negative weight
               cycle"
    Step. 8    compute shortest path from source to all nodes
    Step. 9    Stop
```

2. Program

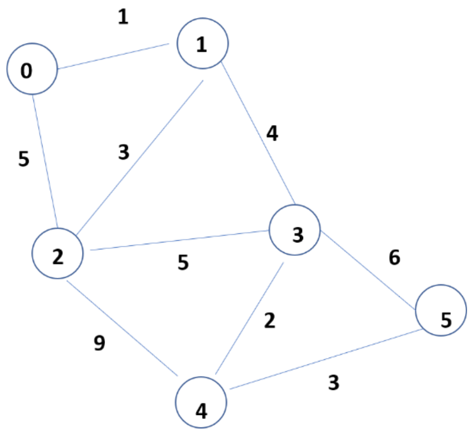
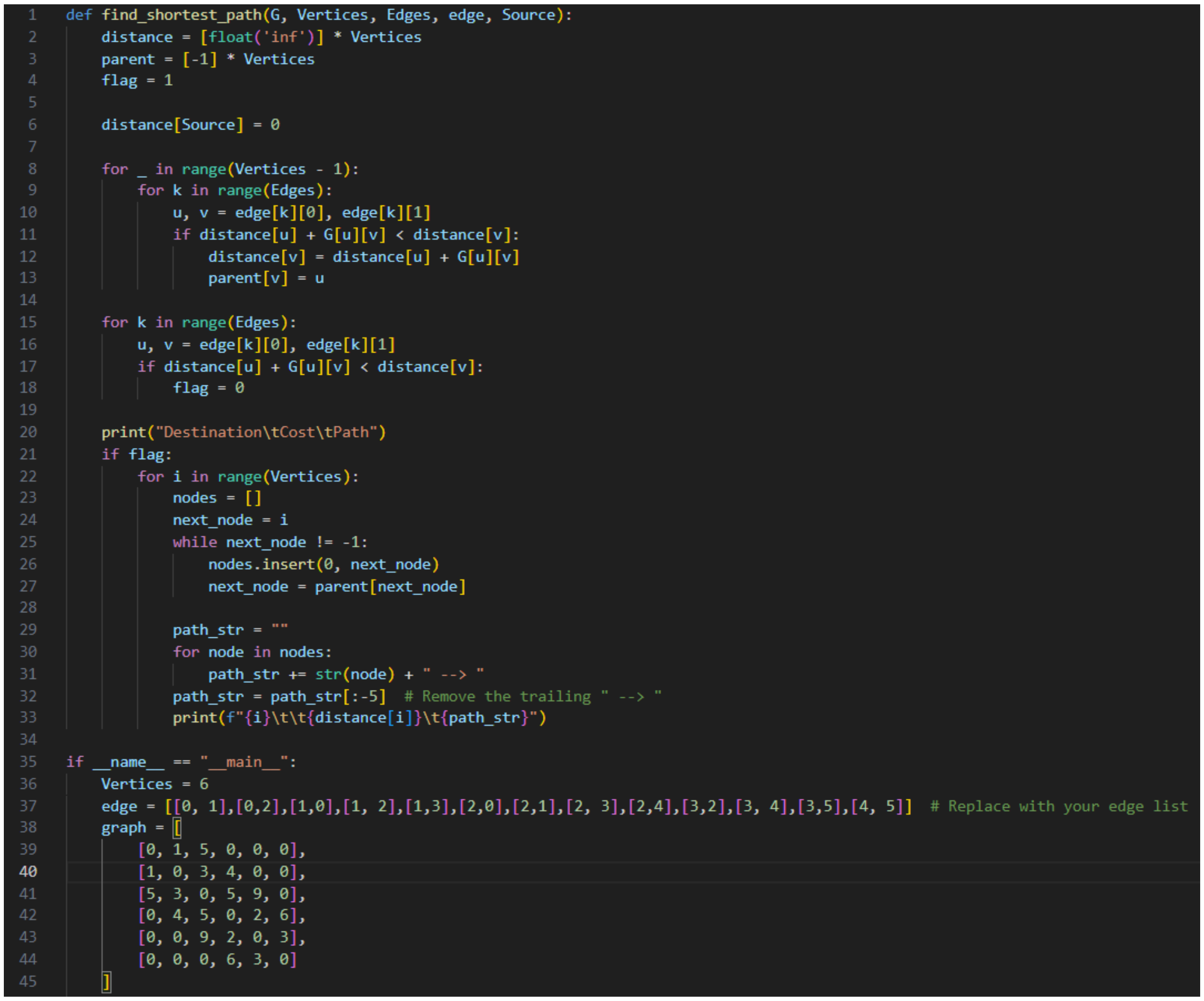


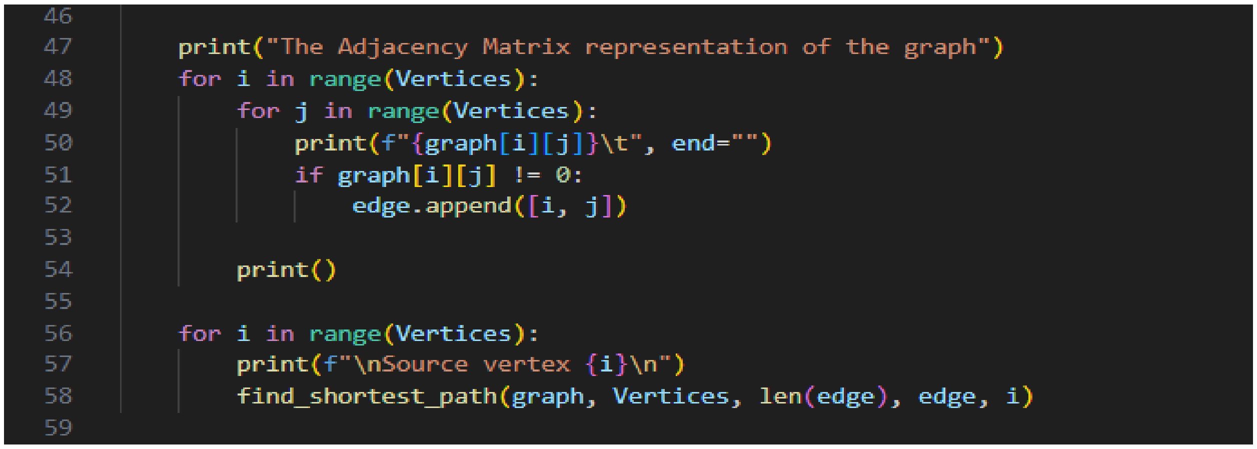
Figure 1: Network topology for which Distance Vector routing is applied.

3. Results

3.1.1 Program: C Program to implement Distance Vector Routing for Network in Figure 1.

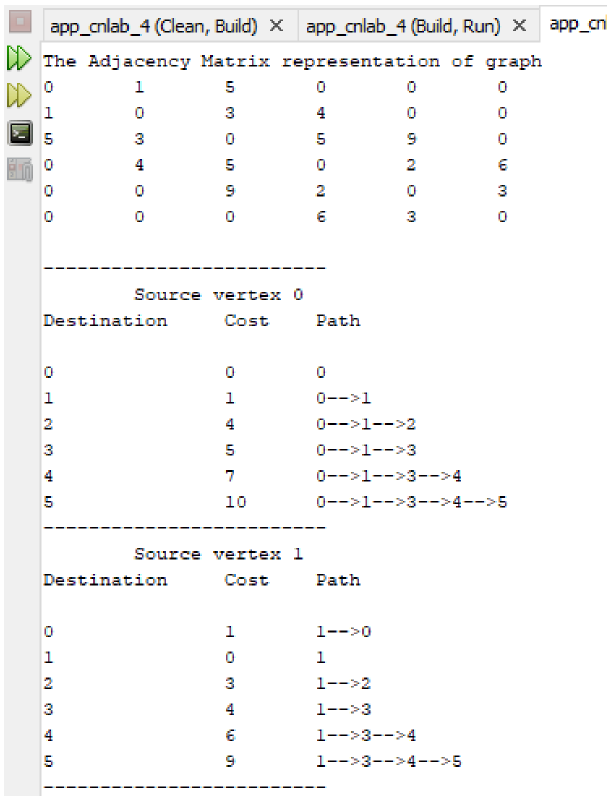


```python
def find_shortest_path(G, Vertices, Edges, edge, Source):
    distance = [float('inf')] * Vertices
    parent = [-1] * Vertices
    flag = 1

    distance[Source] = 0

    for _ in range(Vertices - 1):
        for k in range(Edges):
            u, v = edge[k][0], edge[k][1]
            if distance[u] + G[u][v] < distance[v]:
                distance[v] = distance[u] + G[u][v]
                parent[v] = u

    for k in range(Edges):
        u, v = edge[k][0], edge[k][1]
        if distance[u] + G[u][v] < distance[v]:
            flag = 0

    print("Destination\tCost\tPath")
    if flag:
        for i in range(Vertices):
            nodes = []
            next_node = i
            while next_node != -1:
                nodes.insert(0, next_node)
                next_node = parent[next_node]

            path_str = ""
            for node in nodes:
                path_str += str(node) + " --> "
            path_str = path_str[:-5]  # Remove the trailing " --> "
            print(f"{i}\t\t{distance[i]}\t{path_str}")

if __name__ == "__main__":
    Vertices = 6
    edge = [[0, 1],[0,2],[1,0],[1, 2],[1,3],[2,0],[2,1],[2, 3],[2,4],[3,2],[3, 4],[3,5],[4, 5]]  # Replace with your edge list
    graph = [
        [0, 1, 5, 0, 0, 0],
        [1, 0, 3, 4, 0, 0],
        [5, 3, 0, 5, 9, 0],
        [0, 4, 5, 0, 2, 6],
        [0, 0, 9, 2, 0, 3],
        [0, 0, 0, 6, 3, 0]
    ]
```

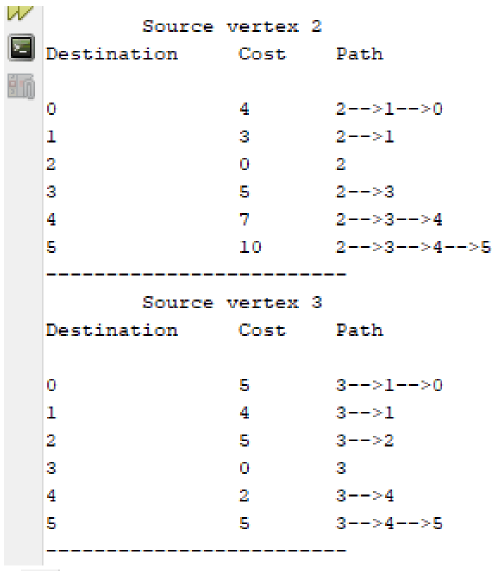


```python

    print("The Adjacency Matrix representation of the graph")
    for i in range(Vertices):
        for j in range(Vertices):
            print(f"{graph[i][j]}\t", end="")
            if graph[i][j] != 0:
                edge.append([i, j])

        print()

    for i in range(Vertices):
        print(f"\nSource vertex {i}\n")
        find_shortest_path(graph, Vertices, len(edge), edge, i)

```

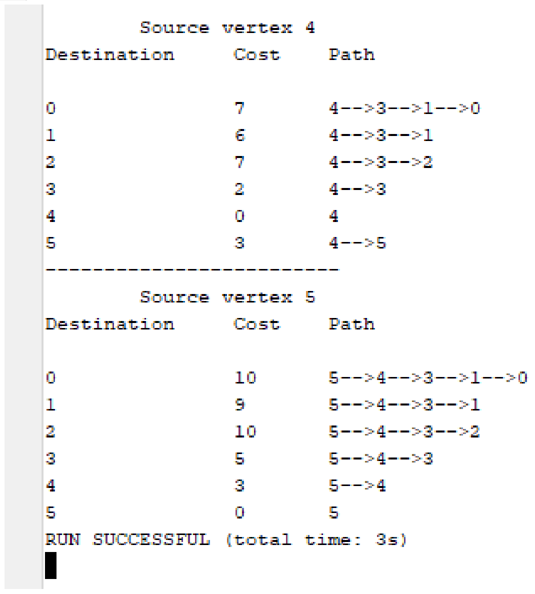
3.1.2Output: Result of the implemented program.



```
The Adjacency Matrix representation of graph
0        1        5        0        0        0
1        0        3        4        0        0
5        3        0        5        9        0
0        4        5        0        2        6
0        0        9        2        0        3
0        0        0        6        3        0


-------------------------
        Source vertex 0
Destination      Cost     Path

0                0        0
1                1        0-->1
2                4        0-->1-->2
3                5        0-->1-->3
4                7        0-->1-->3-->4
5                10       0-->1-->3-->4-->5
-------------------------
        Source vertex 1
Destination      Cost     Path

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



```
         Source vertex 2
Destination      Cost     Path

0                4        2-->1-->0
1                3        2-->1
2                0        2
3                5        2-->3
4                7        2-->3-->4
5                10       2-->3-->4-->5
-------------------------
         Source vertex 3
Destination      Cost     Path

0                5        3-->1-->0
1                4        3-->1
2                5        3-->2
3                0        3
4                2        3-->4
5                5        3-->4-->5
-------------------------
```



```
         Source vertex 4
Destination      Cost     Path

0                7        4-->3-->1-->0
1                6        4-->3-->1
2                7        4-->3-->2
3                2        4-->3
4                0        4
5                3        4-->5
-------------------------
         Source vertex 5
Destination      Cost     Path

0                10       5-->4-->3-->1-->0
1                9        5-->4-->3-->1
2                10       5-->4-->3-->2
3                5        5-->4-->3
4                3        5-->4
5                0        5
RUN SUCCESSFUL (total time: 3s)
```

4. Analysis and Discussions

Bellman Ford Basic idea: Each router maintains a Distance Vector table containing the distance between itself and ALL possible destination nodes. Distances, based on a chosen metric, are computed using information from the neighbors' distance vectors. Distance Metric: usually hops or delay

In Distance Vector Routing,

i) A router transmits its distance vector to each of its neighbors in a routing packet.

ii) Each router receives and saves the most recently received distance vector from each of its neighbors.

iii) A router recalculates its distance vector when:

a. It receives a distance vector from a neighbor containing different information than before.
b. It discovers that a link to a neighbor has gone down (i.e., a topology change). The DV calculation is based on minimizing the cost to each destination.

The adaptive nature of Distance Vector Routing is because of the following

i) Measurement of pertinent network data.
ii) Forwarding of information to where the routing computation will be done.
iii) Compute the routing tables.
iv) Convert the routing table information into a routing decision and then dispatch the data packet.

5. Conclusions

- Understanding, Developing and Analyzing generation of routing tables for a network of routers using Distance Vector Routing.

6. Comments

a. Limitations of the experiment

- It is slower to converge.
- It is at risk from the count-to-infinity problem.
- It creates more traffic since a hop count change must be propagated to all routers and processed on each router. Hop count updates take place on a periodic basis, even if there are no changes in the network topology, so bandwidth-wasting broadcasts still occur.
- For larger networks, Distance Vector routing results in larger routing tables than Link State since each router must know about all other routers. This can also lead to congestion on WAN links.

b. Limitations of the results obtained

- Result is specific to the network and does not take network topologies from user.
- The program is not tested for a graph with a negative cycle.

c. Learning

- Distance Vector Routing.

d. Recommendations

- none