EXP-9: Implement Unicast Routing Algorithms for the following network using Distance Vector Routing and Dijstraw's Routing Algorithms in C++

I Problem Statement:

- Consider a network represented as an undirected graph where the nodes (A, B, C, D, E, F, G) represent routers, and the edges represent the direct communication links between them. The weight of each edge represents the cost (distance) associated with traversing that link. The objective is to determine the shortest path from a given source router to all other routers using:

  Distance Vector Algorithm: Using this distributed routing algorithm, compute the distance vectors for each router until convergence. Illustrate the step-by-step process of how each router updates its distance table based on information received from its neighbors.

  Dijkstra's Algorithm: Using this centralized algorithm, compute the shortest path from given source router (e.g., router A) to all other routers in the network. Provide the fin shortest path distances and the corresponding paths from the source router to eac destination router.

- **Distance Vector Algorithm**:
  - Start by initializing the distance table for each router with direct distances an infinite for all other destinations.
  - Show each iteration of the distance vector exchange between routers.
  - Continue until the distance tables converge (no further updates).
- **Dijkstra's Algorithm**:
  - Choose a source node (e.g., A) and compute the shortest path to all other nodes.
  - Illustrate each step of the algorithm, including the selection of the current node and the update of tentative distances.
- **Output**:
  - For Distance Vector Algorithm: The final distance table for each router after convergence.
  - For Dijkstra's Algorithm: The shortest path distances from the chosen source to all other routers and the path taken.

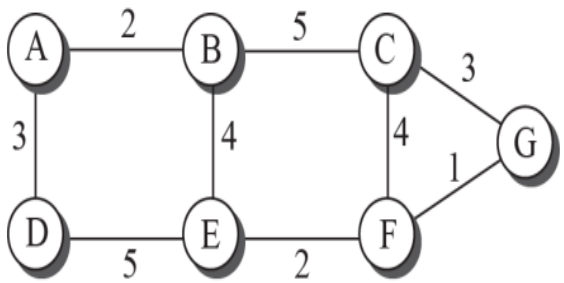

Fig. Network model.

II. Aim:

- Implement the Distance Vector Routing algorithm to compute the shortest path from each router to every other router in the network.
- Implement Dijkstra's algorithm to find the shortest path from node A to all other nodes in the network.

III. Components and Tools:

- System: Desktop Computer/Laptop
- Operating system: Windows/Linux
- Language: C++/Python

IV. Description of Distance Vector Routing algorithm and Dijkstra's algorithm:

V. Algorithms of Distance Vector Routing algorithm and Dijkstra's algorithm:

VI. Output Description/Analysis:

VII. Viva-Voce Questions:

1. Distance-Vector Routing Algorithm

```
1   Distance_Vector_Routing ( )
2   {
3       // Initialize (create initial vectors for the node)
4       D[myself] = 0
5       for (y = 1 to N)
6       {
7           if (y is a neighbor)
8               D[y] = c[myself][y]
9           else
10              D[y] = ∞
11      }
12      send vector {D[1], D[2], ..., D[N]} to all neighbors
13      // Update (improve the vector with the vector received from a neighbor)
14      repeat (forever)
15      {
16          wait (for a vector D_w from a neighbor w or any change in the link)
17          for (y = 1 to N)
18          {
19              D[y] = min [D[y], (c[myself][w] + D_w[y])]      // Bellman-Ford equation
20          }
21          if (any change in the vector)
22              send vector {D[1], D[2], ..., D[N]} to all neighbors
23      }
24  } // End of Distance Vector
```

2. Dijkstra's Algorithm

```
1   Dijkstra's Algorithm ( )
2   {
3       // Initialization
4       Tree = {root}                    // Tree is made only of the root

5       for (y = 1 to N)                 // N is the number of nodes
6       {
7           if (y is the root)
8               D[y] = 0                 // D[y] is shortest distance from root to node y
9           else if (y is a neighbor)
10              D[y] = c[root][y]        // c[x][y] is cost between nodes x and y in LSDB
11          else
12              D[y] = ∞
13      }
14      // Calculation
15      repeat
16      {
17          find a node w, with D[w] minimum among all nodes not in the Tree
18          Tree = Tree ∪ {w}            // Add w to tree
19          // Update distances for all neighbors of w
20          for (every node x, which is a neighbor of w and not in the Tree)
21          {
22              D[x] = min {D[x], (D[w] + c[w][x])}
23          }
24      } until (all nodes included in the Tree)
25  } // End of Dijkstra
```