**4.12 FLOYD’S ALGORITHM FOR ALL-PAIRS SHORTEST PATH**

**Question:**

Implement Floyd's Algorithm to find the shortest path between all pairs of cities. Display the distance matrix before and after applying the algorithm. Identify and print the shortest path.

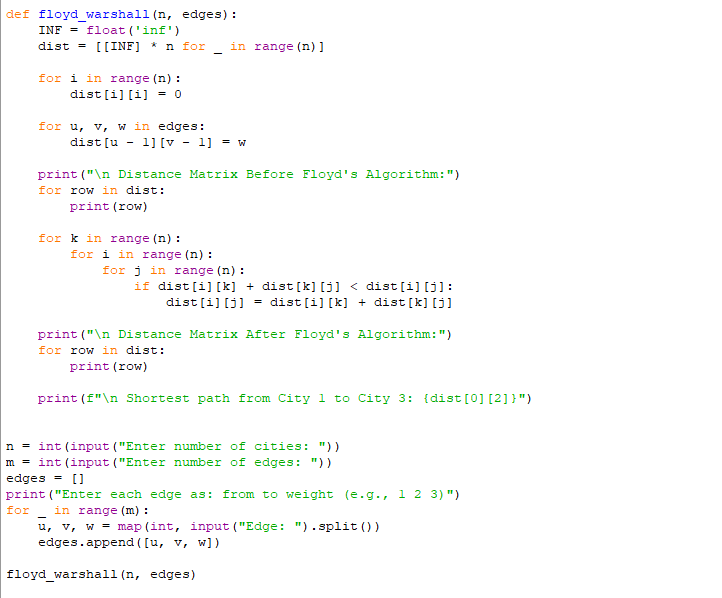
**AIM**

To implement Floyd’s Algorithm in C to compute the shortest paths between all pairs of cities and display the distance matrix before and after applying the algorithm.

**ALGORITHM**

1. Input the number of cities n and the distance matrix dist[n][n].
2. Initialize a path matrix path[n][n] to track intermediate nodes.
3. For each intermediate node k, update dist[i][j] as:
   * dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])
4. Update path[i][j] if a shorter path is found via k.
5. After the algorithm, print the updated distance matrix.
6. Use the path matrix to reconstruct and print the shortest path between each pair.

**PROGRAM**

a)

Input:

Enter number of cities: 4

Enter number of edges: 8

Enter each edge as: from to weight (e.g., 1 2 3)

Edge: 1 2 3

Edge:1 3 8

Edge: 1 4 -4

Edge: 2 4 1

Edge: 2 3 4

Edge: 3 1 2

Edge: 4 3 -5

Edge: 4 2 6

Output:

A screenshot of a computer code

AI-generated content may be incorrect.

**b)** **A screenshot of a computer program

AI-generated content may be incorrect.**

Input

Enter number of routers: 6

Enter number of links: 8

Enter each link as: from to cost (e.g., 0 1 5 for A-B)

Edge: 0 1 1

Edge: 0 2 5

Edge: 1 2 2

Edge: 1 3 1

Edge: 2 4 3

Edge: 3 4 1

Edge: 3 5 6

Edge: 4 5 2

Output

A screenshot of a computer code

AI-generated content may be incorrect.

**RESULT:**

Thus search program is successfully executed and the output is verified.

**PERFORMANCE ANALYSIS:**

· **Time Complexity:** O(n³)

· **Space Complexity:** O(n²)