4.13 FLOYD'S ALGORITHM WITH LINK FAILURE SIMULATION

Question:

Write a program to implement Floyd's Algorithm to calculate the shortest paths between all pairs of routers. Simulate a change where the link between Router B and Router D fails. Update the distance matrix accordingly. Display the shortest path from Router A to Router F before and after the link failure.

AIM

To implement Floyd's Algorithm in Python, simulate a link failure, and observe its impact on the shortest path between two routers.

ALGORITHM

1. Initialize a distance matrix dist[n][n] with INF for unreachable pairs and 0 for diagonal entries.
2. Populate the matrix with given edge weights.
3. Use Floyd's Algorithm to update shortest paths:
    - For each intermediate router k, update dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])
4. Track intermediate routers in a path matrix to reconstruct shortest paths.
5. Simulate link failure by setting the weight of the failed link to INF.
6. Re-run Floyd's Algorithm and compare the updated shortest path.

PROGRAM

```
def floyd(n, edges):
    INF = float('inf')
    dist = [[INF] * n for _ in range(n)]
    for i in range(n):
        dist[i][i] = 0
    for u, v, w in edges:
        dist[u][v] = w
        dist[v][u] = w
    for k in range(n):
        for i in range(n):
            for j in range(n):
                if dist[i][k] + dist[k][j] < dist[i][j]:
                    dist[i][j] = dist[i][k] + dist[k][j]
    return dist


n = int(input("Enter number of routers: "))
m = int(input("Enter number of links: "))
edges = []
print("Enter each link as: from to cost (e.g., 0 1 5 for A-B)")
for _ in range(m):
    u, v, w = map(int, input("Link: ").split())
    edges.append([u, v, w])

dist1 = floyd(n, edges)
print(f"\nShortest path from Router A to Router F before failure: {dist1[0][5]}")


edges = [e for e in edges if not (e[0] == 1 and e[1] == 3 or e[0] == 3 and e[1] == 1)]
dist2 = floyd(n, edges)
print(f"Shortest path from Router A to Router F after failure: {dist2[0][5]}")
```

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:

```
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)
Link: 0 1 1
Link: 0 2 5
Link: 1 2 2
Link: 1 3 1
Link: 2 4 3
Link: 3 4 1
Link: 3 5 6
Link: 4 5 2

Shortest path from Router A to Router F before failure: 5
Shortest path from Router A to Router F after failure: 8
>>>
```

RESULT:

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

PERFORMANCE ANALYSIS:

- Time Complexity: $O(n^3)$
- Space Complexity: $O(n^2)$