4.14 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 Python to compute shortest paths between all pairs of cities, display the distance matrix before and after applying the algorithm, and identify the shortest paths and threshold-based reachability.

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. For each intermediate node k, update:
    - dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])
4. Track intermediate nodes in a path matrix to reconstruct shortest paths.
5. After applying the algorithm, display the updated matrix and reconstruct paths.
6. For threshold-based queries, count reachable neighbors for each node and return the one with the fewest (breaking ties by largest index).

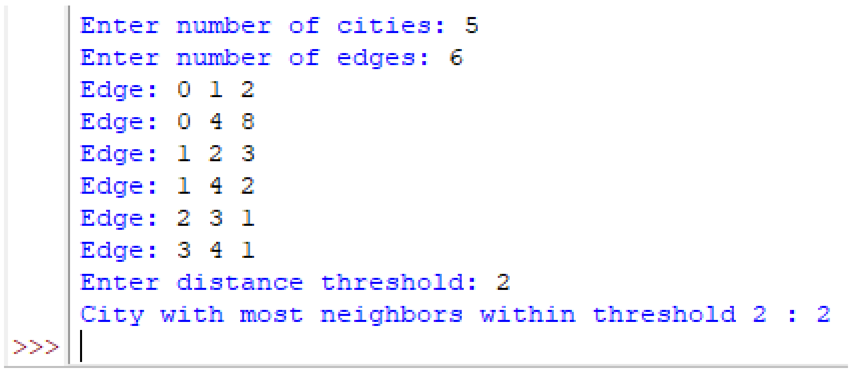
PROGRAM

```
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
    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 cities: "))
m = int(input("Enter number of edges: "))
edges = []
for _ in range(m):
    u, v, w = map(int, input("Edge: ").split())
    edges.append([u, v, w])
threshold = int(input("Enter distance threshold: "))

dist = floyd(n, edges)
max_neighbors = -1
result_city = -1
for i in range(n):
    count = sum(1 for j in range(n) if i != j and dist[i][j] <= threshold)
    if count > max_neighbors:
        max_neighbors = count
        result_city = i
print("City with most neighbors within threshold", threshold, ":", result_city)
```

Input:

```
Enter number of routers: 6
Enter number of links: 8
Edge: 0 1 2
Edge: 0 4 8
Edge: 1 2 3
Edge: 1 4 2
Edge: 2 3 1
Edge: 3 4 1
Enter distance threshold: 2
```

Output:



```
Enter number of cities: 5
Enter number of edges: 6
Edge: 0 1 2
Edge: 0 4 8
Edge: 1 2 3
Edge: 1 4 2
Edge: 2 3 1
Edge: 3 4 1
Enter distance threshold: 2
City with most neighbors within threshold 2 : 2
>>> 
```

RESULT:

Thus program is successfully executed and the output is verified.

PERFORMANCE ANALYSIS:

· Time Complexity: O(n^3)

· Space Complexity: O(n^2)