4.21 CITY WITH FEWEST REACHABLE NEIGHBORS WITHIN DISTANCE THRESHOLD

Question:

There are n cities numbered from 0 to n-1. Given the array edges where edges[i] = [fromi, toi, weighti] represents a bidirectional and weighted edge between cities fromi and toi, and given the integer distanceThreshold. Return the city with the smallest number of cities that are reachable through some path and whose distance is at most distanceThreshold, If there are multiple such cities, return the city with the greatest number. Notice that the distance of a path connecting cities i and j is equal to the sum of the edges' weights along that path.

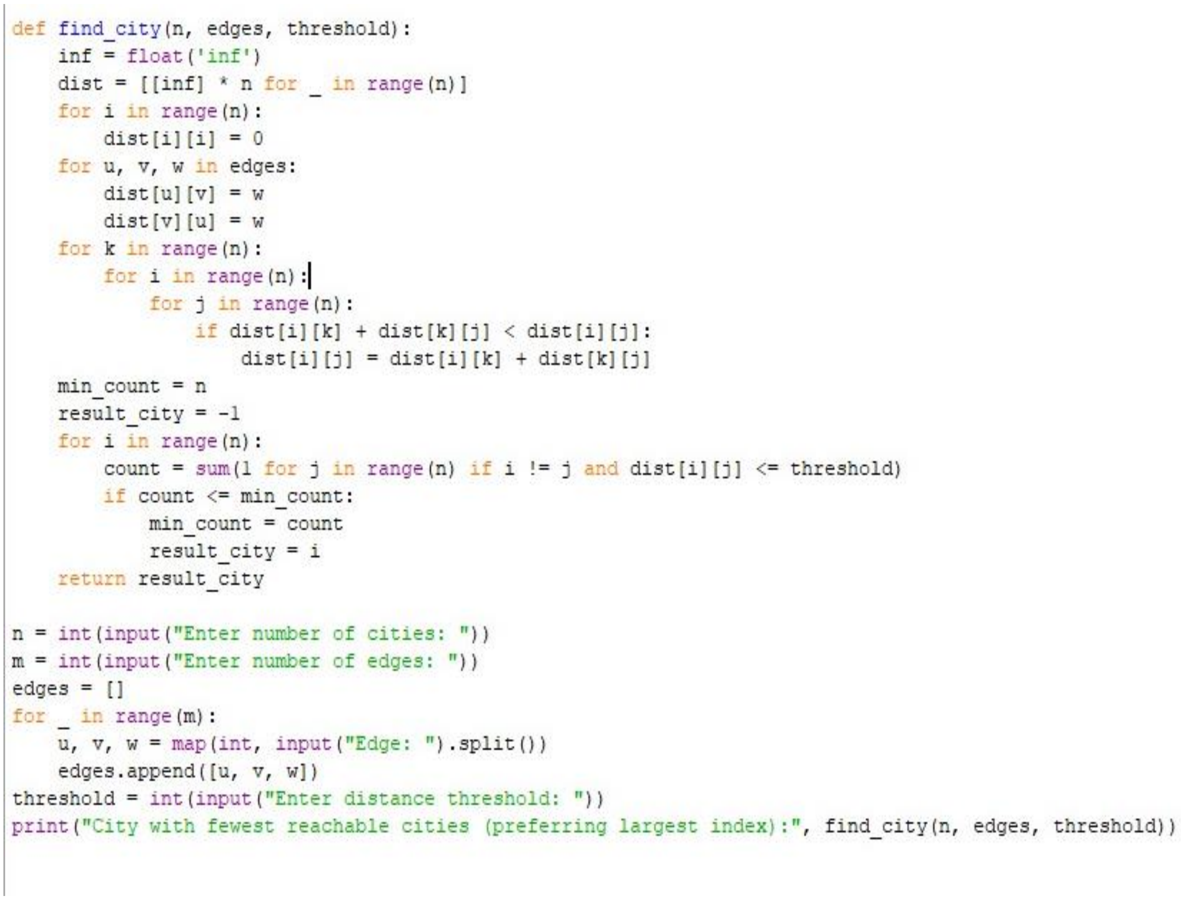
AIM

To implement Floyd's Algorithm to compute shortest paths between all pairs of cities and identify the city with the fewest reachable neighbors within a given distance threshold.

ALGORITHM

1. Initialize a distance matrix dist[n][n] with INF for all pairs except dist[i][i] = 0.
2. Populate the matrix with given edge weights.
3. Apply Floyd-Warshall Algorithm to compute shortest paths between all pairs.
4. For each city i, count the number of cities j ≠ i such that dist[i][j] ≤ distanceThreshold.
5. Track the city with the minimum count, breaking ties by choosing the greatest index.

PROGRAM



```
def find_city(n, edges, threshold):
    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]
    min_count = n
    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 <= min_count:
            min_count = count
            result_city = i
    return result_city

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: "))
print("City with fewest reachable cities (preferring largest index):", find_city(n, edges, threshold))
```

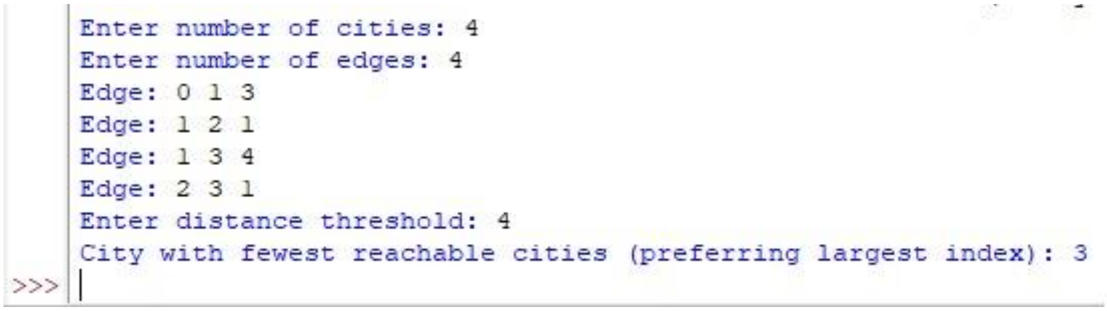
Input:

Enter cities: 4

Edges: 4
Edge: 0 1 3
Edge: 1 2 1
Edge: 1 3 4
Edge: 2 3 1

Enter distance threshold: 4

Output:



```
Enter number of cities: 4
Enter number of edges: 4
Edge: 0 1 3
Edge: 1 2 1
Edge: 1 3 4
Edge: 2 3 1
Enter distance threshold: 4
City with fewest reachable cities (preferring largest index): 3
>>> |
```

RESULT:

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

PERFORMANCE ANALYSIS:

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