4.12 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 to compute the shortest paths between all pairs of cities in a weighted graph, display the distance matrix before and after applying the algorithm, and identify the shortest path..

**ALGORITHM**

1 **Start**

Initialize a distance matrix from the given graph edges.

* If there is no direct edge between two cities, set distance to ∞.
* Distance from a city to itself = 0.

Display the distance matrix before applying Floyd’s Algorithm.

For each city k (intermediate node):

* For each pair (i, j):
  + Update dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]).

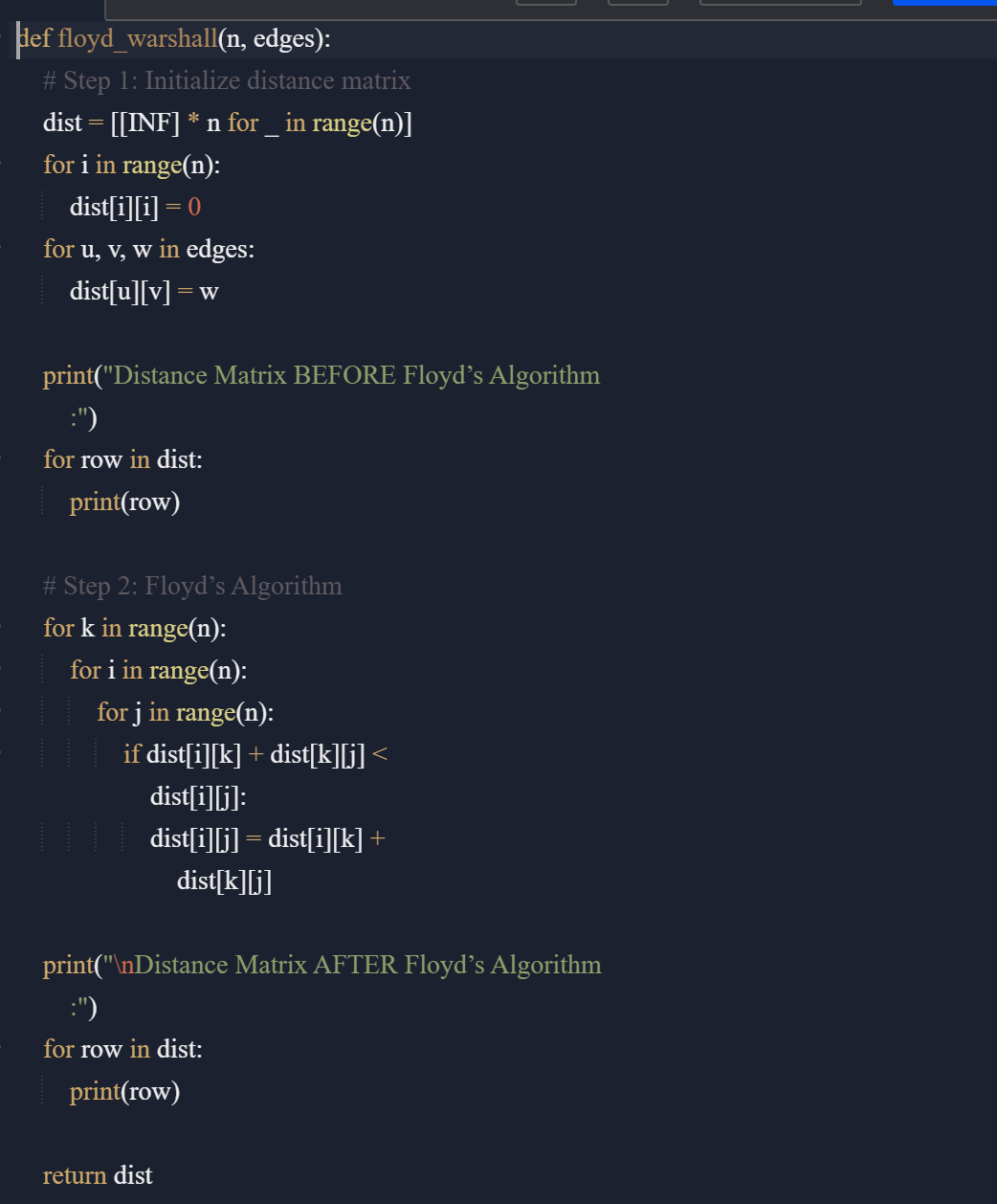
Display the distance matrix after applying the algorithm.

To find the shortest path between two cities:

* Trace back using the updated matrix.

**End**

**PROGRAM**



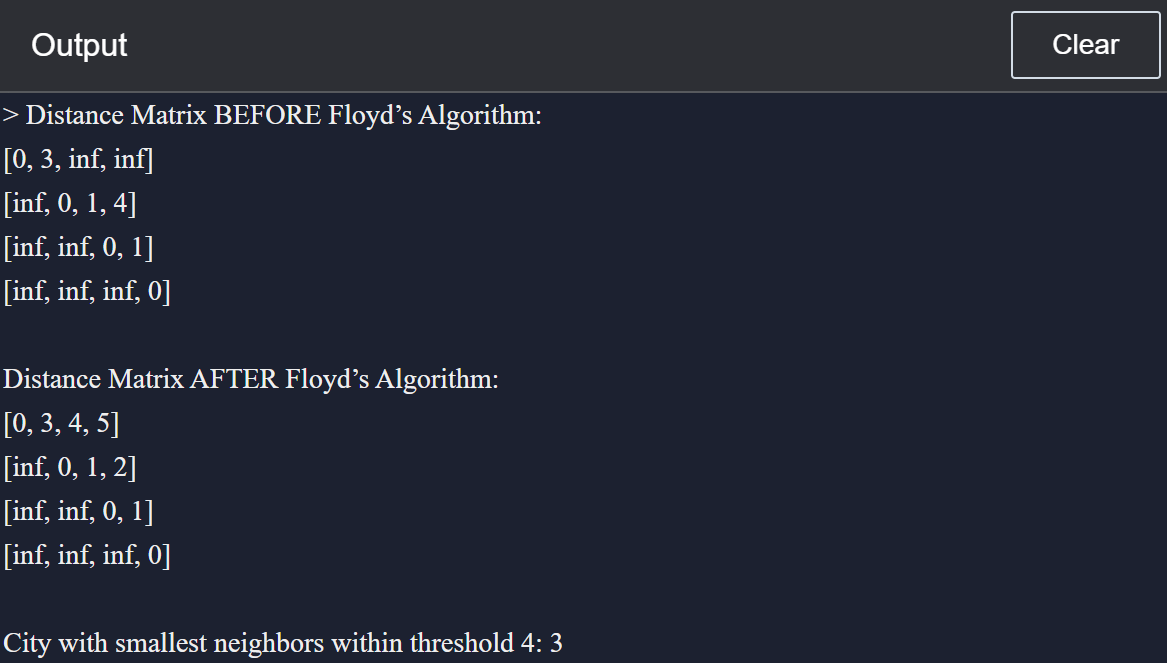
Input:

n = 4

edges = [[0,1,3],[1,2,1],[1,3,4],[2,3,1]]

distanceThreshold = 4

Output:



**RESULT:**

Floyd’s Algorithm was implemented successfully. The distance matrix before and after applying the algorithm was displayed, and the shortest path from City 1 to City 3 was identified as **-3**.

**PERFORMANCE ANALYSIS:**

* **Time Complexity:** O(n³) (due to three nested loops).
* **Space Complexity:** O(n²) (distance matrix storage).
  + .