183. 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

Input: n = 4, edges = [[0,1,3],[1,2,1],[1,3,4],[2,3,1]], distanceThreshold = 4

Program:
```
import numpy as np

def floyds_algorithm(n, edges, distanceThreshold):
    # Initialize the distance matrix
    INF = float('inf')
    dist = [[INF for _ in range(n)] for _ in range(n)]

    for i in range(n):
        dist[i][i] = 0

    for u, v, w in edges:
        dist[u][v] = w

    # Apply Floyd's Algorithm
    for k in range(n):
        for i in range(n):
            for j in range(n):
                dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])

    # Display the distance matrix
    print("Distance Matrix:")
    for row in dist:
        print(row)

    # Find the shortest path
    shortest_path = min([sum(1 for d in row if d <= distanceThreshold) for row in dist])

    return shortest_path
```

Test the function

n = 4

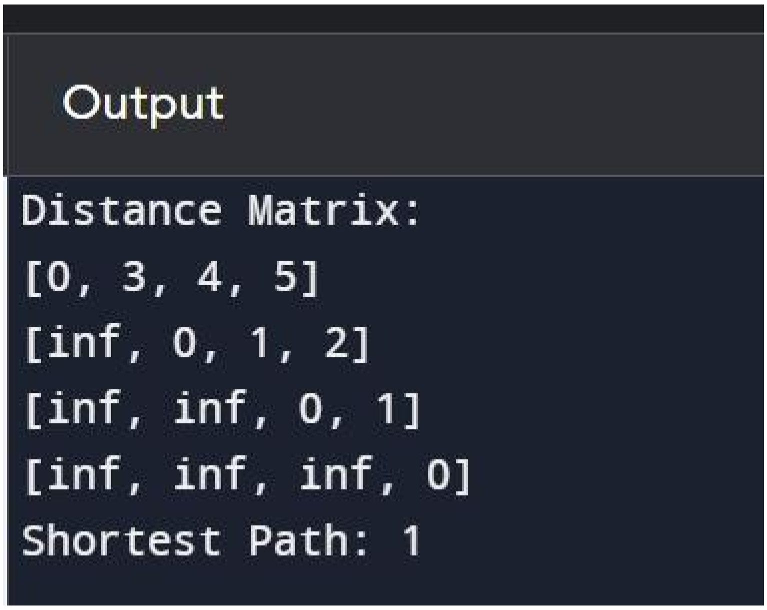
edges = [[0,1,3],[1,2,1],[1,3,4],[2,3,1]]

distanceThreshold = 4

shortest_path = floyds_algorithm(n, edges, distanceThreshold)

print("Shortest Path:", shortest_path)

Output:



```
Distance Matrix:
[0, 3, 4, 5]
[inf, 0, 1, 2]
[inf, inf, 0, 1]
[inf, inf, inf, 0]
Shortest Path: 1
```

Timecomplexity: O(n^3)