185. 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 = 5, edges = [[0,1,2],[0,4,8],[1,2,3],[1,4,2],[2,3,1],[3,4,1]], distanceThreshold = 2

Output: 0

Program:# Define the graph as an adjacency matrix

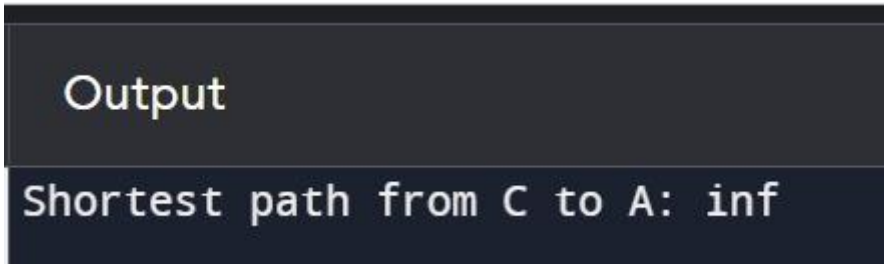
```
graph = [
   [0, 2, float('inf'), 6, float('inf')],
   [float('inf'), 0, 3, float('inf'), 7],
   [float('inf'), float('inf'), 0, 1, 1],
   [float('inf'), float('inf'), float('inf'), 0, 1],
   [float('inf'), float('inf'), float('inf'), float('inf'), 0]
]


# Implement Floyd's Algorithm
def floyd_algorithm(graph):
   n = len(graph)
   for k in range(n):
      for i in range(n):
         for j in range(n):
            if graph[i][k] + graph[k][j] < graph[i][j]:
               graph[i][j] = graph[i][k] + graph[k][j]


# Apply Floyd's Algorithm
floyd_algorithm(graph)


# Print the shortest path from C to A
print("Shortest path from C to A:", graph[2][0])
```

Output:



Timecomplexity: O(n^3)