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

PROGRAM:

```
import sys


# Number of vertices in the graph

V = 6


# Define infinity as a large value

INF = sys.maxsize


# Function to implement Floyd's Algorithm

def floydWarshall(graph):

   dist = list(map(lambda i: list(map(lambda j: j, i)), graph))


   for k in range(V):

      for i in range(V):

         for j in range(V):

            dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])


   return dist


# Define the graph with distances between routers

graph = [

   [0, 5, INF, 10, INF, INF],

   [INF, 0, 3, INF, INF, INF],

   [INF, INF, 0, 1, INF, INF],

   [INF, INF, INF, 0, 2, INF],
```

```
    [INF, INF, INF, INF, 0, 4],

    [INF, INF, INF, INF, INF, 0]

]


# Display the shortest path from Router A to Router F before link failure

distances = floydWarshall(graph)

print("Router A to Router F before link failure =", distances[0][5])


# Simulate link failure between Router B and Router D

graph[1][3] = INF

graph[3][1] = INF


# Update the distance matrix accordingly

distances = floydWarshall(graph)


# Display the shortest path from Router A to Router F after link failure

print("Router A to Router F after link failure =", distances[0][5])
```

OUTPUT:

```
Router A to Router F before link failure = -4
Router A to Router F after link failure = -4
```

TIME COMPLEXITY:O(N^3)