106. Floyd algorithm

Program:

```
INF = float('inf')

def floyd_warshall(graph):
    n = len(graph)
    dist = [[INF for _ in range(n)] for _ in range(n)]

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

    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])

    return dist

# Example graph represented as an adjacency matrix
graph = [
    [0, 5, INF, 10],
    [INF, 0, 3, INF],
    [INF, INF, 0, 1],
    [INF, INF, INF, 0]
]
```

```
result = floyd_warshall(graph)

for row in result:

    print(row)
```

OUTPUT:

```
[0, 5, 8, 9]
[inf, 0, 3, 4]
[inf, inf, 0, 1]
[inf, inf, inf, 0]

=== Code Execution Successful ===
```

Time complexity: O(n^3)