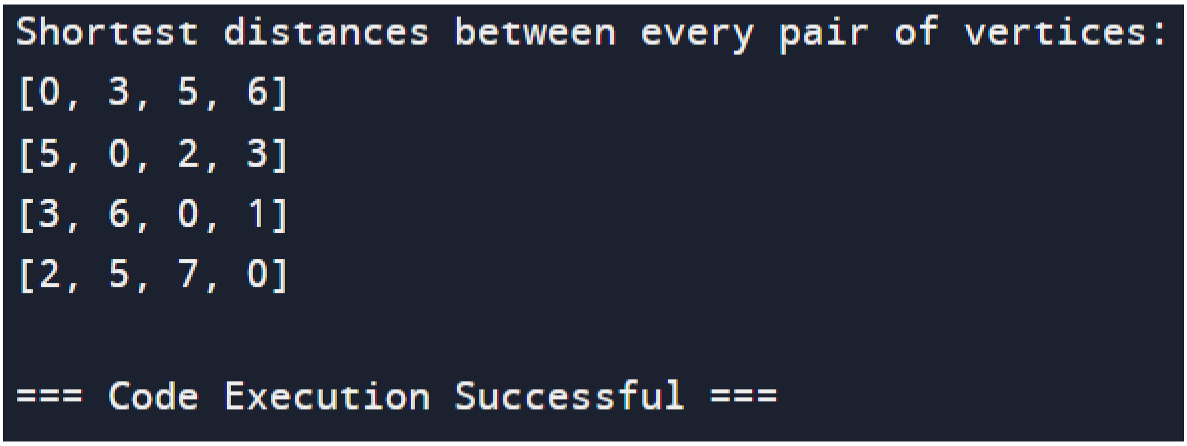
106. Floyd-Warshall Algorithm

PROGRAM:-

```
def floyd_warshall(graph):
    V = len(graph)
    # Initialize the solution matrix same as input graph matrix
    dist = [[float('inf')] * V for _ in range(V)]

    # Initialize the distance to the same as graph adjacency matrix
    for i in range(V):
        for j in range(V):
            dist[i][j] = graph[i][j]

    # Distance from a vertex to itself is always 0
    for i in range(V):
        dist[i][i] = 0

    # Update the solution matrix
    for k in range(V):
        for i in range(V):
            for j in range(V):
                # If vertex k is on the shortest path from i to j, then update the value of dist[i][j]
                if dist[i][k] + dist[k][j] < dist[i][j]:
                    dist[i][j] = dist[i][k] + dist[k][j]

    return dist

# Example usage:
# Graph represented as an adjacency matrix
graph = [
    [0, 3, float('inf'), 7],
    [8, 0, 2, float('inf')],
    [5, float('inf'), 0, 1],
    [2, float('inf'), float('inf'), 0]
]

dist = floyd_warshall(graph)

print("Shortest distances between every pair of vertices:")
for row in dist:
    print(row)
```

OUTPUT:-



```
Shortest distances between every pair of vertices:
[0, 3, 5, 6]
[5, 0, 2, 3]
[3, 6, 0, 1]
[2, 5, 7, 0]

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

TIME COMPLEXITY:-O(V^3)