5.5 DIJKSTRA'S ALGORITHM USING ADJACENCY MATRIX

Question:

Given a graph represented by an adjacency matrix, implement Dijkstra's Algorithm to find the shortest path from a given source vertex to all other vertices in the graph. The graph is represented as an adjacency matrix where graph[i][j] denote the weight of the edge from vertex i to vertex j. If there is no edge between vertices i and j, the value is Infinity (or a very large number).

AIM

To compute the shortest path from a source vertex to all other vertices in a weighted graph using Dijkstra's algorithm and an adjacency matrix.

ALGORITHM

1. Initialize a distance array dist[] with INF, and set dist[source] = 0.
2. Create a boolean array visited[] to track processed vertices.
3. Repeat for all vertices:
    - Select the unvisited vertex u with the smallest dist[u].
    - Mark u as visited.
    - For each neighbor v of u, if graph[u][v] is not INF and v is unvisited:
        - Update dist[v] = min(dist[v], dist[u] + graph[u][v])
4. After all vertices are processed, dist[] contains the shortest distances from the source.

PROGRAM

```
def dijkstra(graph, src):
    n = len(graph)
    dist = [float('inf')] * n
    visited = [False] * n
    dist[src] = 0
    for _ in range(n):
        u = -1
        for i in range(n):
            if not visited[i] and (u == -1 or dist[i] < dist[u]):
                u = i
        visited[u] = True
        for v in range(n):
            if graph[u][v] != 999999 and dist[u] + graph[u][v] < dist[v]:
                dist[v] = dist[u] + graph[u][v]
    return dist

n = int(input("Enter number of vertices: "))
graph = []
for i in range(n):
    row = list(map(int, input(f"Row {i}: ").split()))
    graph.append(row)
src = int(input("Enter source vertex: "))
print("Shortest distances from source:", dijkstra(graph, src))
```

Input:

Enter number of vertices: 5
Row 0: 0 10 3 999999 999999
Row 1: 999999 0 1 2 999999
Row 2: 999999 4 0 8 2
Row 3: 999999 999999 999999 0 7
Row 4: 999999 999999 9999999 9 0
Enter source vertex: 0

Output:

```
Enter number of vertices: 5
Row 0: 0 10 3 999999 999999
Row 1: 999999 0 1 2 999999
Row 2: 999999 4 0 8 2
Row 3: 999999 9999999 999999 0 7
Row 4: 999999 9999999 9999999 9 0
Enter source vertex: 0
Shortest distances from source: [0, 7, 3, 9, 5]
>>>
```

RESULT:

Thus the program is successfully executed and the output is verified.

PERFORMANCE ANALYSIS:

- Time Complexity: $O(n^2)$
- Space Complexity: O(n)