Practical No. : 3

Name : Mohan Kadambande
Roll No. : 13212
Aim : Implement Dijkstra's Aigorithm using Greedy Search Algorthm.
Code :

```
import sys
# Function to find the vertex with the minimum distance value that has not been included in the shortest path yet
def min_distance(dist, spt_set, V):
    min_val = sys.maxsize
    min_index = -1

    for v in range(V):
        if dist[v] < min_val and not spt_set[v]:
            min_val = dist[v]
            min_index = v

    return min_index

# Function to implement Dijkstra's Algorithm
def dijkstra(graph, src, V):
    dist = [sys.maxsize] * V    # Initialize distances to all vertices as infinity
    dist[src] = 0    # Distance to the source is 0
    spt_set = [False] * V    # Shortest path tree set (True means the vertex is included in the shortest path)

    # Find the shortest path for all vertices
    for _ in range(V):
        # Get the vertex with the minimum distance from the set of vertices not yet processed
        u = min_distance(dist, spt_set, V)

        # Include this vertex in the shortest path tree set
        spt_set[u] = True

        # Update the distance value of the adjacent vertices of the selected vertex
        for v in range(V):
            # Update dist[v] if and only if the vertex u is not in the shortest path tree set,
            # and there is an edge from u to v, and the total weight of path through u is smaller
            # than the current value of dist[v].
            if not spt_set[v] and graph[u][v] != 0 and dist[u] != sys.maxsize and dist[u] +
```

```
graph[u][v] < dist[v]:
                        dist[v] = dist[u] + graph[u][v]

    # Print the constructed distance array
    print("Vertex \tDistance from Source")
    for i in range(V):
        print(f"{i} \t{dist[i]}")

# Main function to take user input
def main():
    V = int(input("Enter the number of vertices: "))

    # Create an empty graph (adjacency matrix) with zeros
    graph = [[0 for _ in range(V)] for _ in range(V)]

    print("Enter the adjacency matrix (enter 0 for no edge between vertices):")
    for i in range(V):
        for j in range(i + 1, V):
            weight = int(input(f"Enter the weight of edge ({i}, {j}): "))
            graph[i][j] = weight
            graph[j][i] = weight

    src = int(input("Enter the source vertex: "))

    # Call Dijkstra's algorithm
    dijkstra(graph, src, V)

if __name__ == "__main__":
    main()
```

Output :

```
Enter the number of vertices: 5
Enter the adjacency matrix (enter 0 for no edge between vertices):
Enter the weight of edge (0, 1): 4
Enter the weight of edge (0, 2): 2
Enter the weight of edge (0, 3): 0
Enter the weight of edge (0, 4): 0
Enter the weight of edge (1, 2): 1
Enter the weight of edge (1, 3): 7
Enter the weight of edge (1, 4): 0
Enter the weight of edge (2, 3): 3
Enter the weight of edge (2, 4): 5
Enter the weight of edge (3, 4): 7
Enter the source vertex: 0
```

Vertex	Distance from Source
0	0
1	3
2	2
3	5
4	7