UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

Worksheet 4

Student Name: Rahul Saxena **UID:** 24MCI10204

Branch: MCA(AI&ML) **Section/Group:** 3-B

Semester: 1st semester **Date of Performance:** 13/09/2024

Subject Name: Design and Analysis of Algorithm **Subject Code:** 24CAP-612

Aim:

From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.

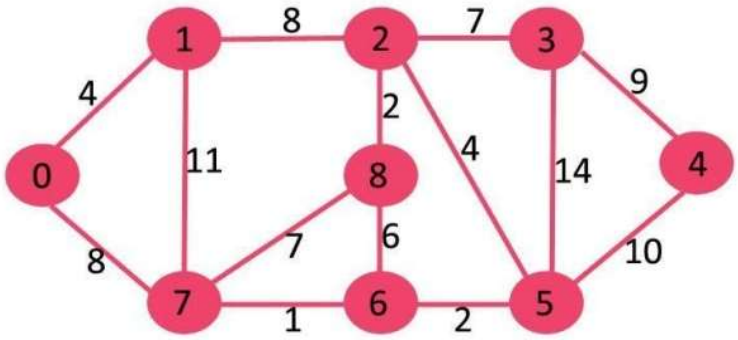

Figure 1: Graph-1

Task To be done:

- **Objective**: Implement Dijkstra's algorithm to find the shortest paths from a given source vertex (node) to all other vertices in the provided weighted connected graph.
- **Steps**:
    - Input the weighted graph (as seen in Figure 1).
    - Select a source vertex, e.g., vertex 0.
    - Use Dijkstra's algorithm to compute the shortest paths from the source vertex to every other vertex in the graph.
    - Display the shortest distances and paths for all vertices.

Source Code:

```
class DijkstraAlgorithm1{
   int minDistance(int[] dist, boolean[] sptSet, int V) {
      int min = Integer.MAX_VALUE, minIndex = -1;
      for (int v = 0; v < V; v++)
         if (!sptSet[v] && dist[v] <= min) {
```

```
                min = dist[v];
                minIndex = v;
            }
        return minIndex;
    }
    void dijkstra(int[][] graph, int src, int V) {
        int[] dist = new int[V];
        boolean[] sptSet = new boolean[V];
        for (int i = 0; i < V; i++) {
            dist[i] = Integer.MAX_VALUE;
            sptSet[i] = false;
        }
        dist[src] = 0;
        for (int count = 0; count < V - 1; count++) {
            int u = minDistance(dist, sptSet, V);
            sptSet[u] = true;
            for (int v = 0; v < V; v++)
                if (!sptSet[v] && graph[u][v] != 0 && dist[u] != Integer.MAX_VALUE && dist[u] +
graph[u][v] < dist[v])
                    dist[v] = dist[u] + graph[u][v];
        }
        printSolution(dist, V);
    }
    void printSolution(int[] dist, int V) {
        System.out.println("Vertex \t Distance from Source");
        for (int i = 0; i < V; i++)
            System.out.println(i + " \t\t " + dist[i]);
    }
    public static void main(String[] args) {
        int[][] graph = new int[][] {
                { 0, 4, 0, 0, 0, 0, 8, 0 },
                { 4, 0, 8, 0, 0, 0, 11, 0 },
                { 0, 8, 0, 7, 0, 4, 0, 2 },
                { 0, 0, 7, 0, 9, 14, 0, 0 },
                { 0, 0, 0, 9, 0, 10, 0, 0 },
                { 0, 0, 4, 14, 10, 0, 2, 0 },
                { 8, 11, 0, 0, 0, 2, 0, 1 },
                { 0, 0, 2, 0, 0, 0, 1, 0 }
        };
        DijkstraAlgorithm1 da = new DijkstraAlgorithm1();
        int src = 0;
        da.dijkstra(graph, src, graph.length);
    }
}
```

Output:

```
Vertex   Distance from Source
0        0
1        4
2        11
3        18
4        20
5        10
6        8
7        9

Process finished with exit code 0
```

Learning Outcome:

- **Understanding Dijkstra's Algorithm**: After implementing and executing this experiment, you will gain a solid understanding of how Dijkstra's algorithm works to find the shortest path in a weighted graph.
- **Graph Representation**: You will learn how to represent graphs using an adjacency matrix and manipulate graph data to implement graph algorithms.
- **Application in Real-World Problems**: You will understand how Dijkstra's algorithm can be applied to various real-world problems, such as finding the shortest path in a network of roads, telecommunications, or computer networks.
- **Optimization Techniques**: By studying Dijkstra's approach, you will develop skills to optimize the selection process of the shortest path in connected graphs.