BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
MUNSHI NAGAR, ANDHERI (WEST), MUMBAI – 400 058.
(Autonomous College Affiliated to University of Mumbai)
MASTER OF COMPUTER APPLICATIONS

Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

UCID: 2024510001 BATCH: A NAME: Atharva Vasant Angre

EXPERIMENT NO: 08

EXPERIMENT TITLE: To implement Dijkstra Algorithm problem

5.1 To implement Dijkstra Algorithm problem.

Objective:

1. To understand the concept of shortest path in a weighted graph.
2. To implement Dijkstra's Algorithm to find the shortest path from a given source vertex to a target vertex.
3. To display the path taken and cumulative weights at each step.

Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

Program code: -

```
import java.util.*;

public class DijkstraAlgorithm {

    static class Edge {
        int target, weight;
        Edge(int target, int weight) {
            this.target = target;
            this.weight = weight;
        }
    }

    public static void dijkstra(List<List<Edge>> graph, int source, int
target) {
        int n = graph.size();
        int[] distance = new int[n];
        int[] parent = new int[n];
        boolean[] visited = new boolean[n];

        Arrays.fill(distance, Integer.MAX_VALUE);
        Arrays.fill(parent, -1);
        distance[source] = 0;

        PriorityQueue<int[]> pq = new
PriorityQueue<>(Comparator.comparingInt(a -> a[1]));
        pq.offer(new int[] {source, 0});

        while (!pq.isEmpty()) {
            int[] current = pq.poll();
            int node = current[0];

            if (visited[node]) continue;
            visited[node] = true;

            for (Edge edge : graph.get(node)) {
                if (distance[edge.target] > distance[node] + edge.weight) {
                    distance[edge.target] = distance[node] + edge.weight;
                    parent[edge.target] = node;
                    pq.offer(new int[] {edge.target,
distance[edge.target]});
                }
            }
        }

        if (distance[target] == Integer.MAX_VALUE) {
            System.out.println("No path found from " + source + " to " +
target);
```

Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

```
            return;
        }

        // Build path from source to target
        List<Integer> path = new ArrayList<>();
        for (int node = target; node != -1; node = parent[node]) {
            path.add(node);
        }
        Collections.reverse(path);

        // Print vertex path
        System.out.print("Path: ");
        for (int i = 0; i < path.size(); i++) {
            System.out.print(path.get(i));
            if (i < path.size() - 1) System.out.print(" -> ");
        }

        // Print cumulative weights
        System.out.print("\nWeighted Path: ");
        int sum = 0;
        System.out.print(sum);
        for (int i = 1; i < path.size(); i++) {
            int u = path.get(i - 1);
            int v = path.get(i);
            for (Edge edge : graph.get(u)) {
                if (edge.target == v) {
                    sum += edge.weight;
                    break;
                }
            }
            System.out.print(" -> " + sum);
        }

        System.out.println();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input number of vertices and edges
        System.out.print("Enter number of vertices: ");
        int v = scanner.nextInt();
        System.out.print("Enter number of edges: ");
        int e = scanner.nextInt();

        List<List<Edge>> graph = new ArrayList<>();
        for (int i = 0; i < v; i++) graph.add(new ArrayList<>());

        // Input edges
```

Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

```
        System.out.println("Enter edges in format: source target weight");
        int lastTarget = -1;
        for (int i = 0; i < e; i++) {
            int src = scanner.nextInt();
            int tgt = scanner.nextInt();
            int wgt = scanner.nextInt();
            graph.get(src).add(new Edge(tgt, wgt));
            lastTarget = tgt; // Keep updating to get the last entered
vertex
        }

        // Input source vertex
        System.out.print("Enter source vertex: ");
        int source = scanner.nextInt();

        System.out.println();
        dijkstra(graph, source, lastTarget);

        scanner.close();
    }
}
```

Output:

```
Enter number of vertices: 6
Enter number of edges: 8
Enter edges in format: source target weight
0 2 4
0 1 4
1 2 2
2 3 3
2 5 6
2 4 1
3 5 2
4 5 3
Enter source vertex: 0

Path: 0 -> 2 -> 4 -> 5
Weighted Path: 0 -> 4 -> 5 -> 8
```

UCID: 2024510001

Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

Conclusion:

In this experiment, we successfully implemented Dijkstra's Algorithm to find the shortest path from a source vertex to a target vertex in a weighted graph. We also displayed both the path and the cumulative weights at each step. This experiment helped us understand how greedy algorithms and priority queues can be effectively used to solve shortest path problems in real-world scenarios like network routing and map navigation.