**Ex:10 - DIJKSTRA**

**Name:** Athithraja R

**Reg.no:** 2022503702

**dijkstra.java**

import java.util.\*;

public class Dijkstra {

static class Graph {

int vertices;

List<List<Edge>> adjList;

public Graph(int vertices) {

this.vertices = vertices;

adjList = new ArrayList<>();

for (int i = 0; i < vertices; i++) {

adjList.add(new ArrayList<>());

}

}

public void addEdge(int u, int v, int weight) {

adjList.get(u).add(new Edge(v, weight));

adjList.get(v).add(new Edge(u, weight)); // If the graph is undirected

}

}

static class Edge {

int vertex;

int weight;

public Edge(int vertex, int weight) {

this.vertex = vertex;

this.weight = weight;

}

}

public static int[] dijkstra(Graph graph, int src) {

int[] dist = new int[graph.vertices];

Arrays.fill(dist, Integer.MAX\_VALUE);

dist[src] = 0;

PriorityQueue<Edge> pq = new PriorityQueue<>(Comparator.comparingInt(e -> e.weight));

pq.add(new Edge(src, 0));

while (!pq.isEmpty()) {

Edge current = pq.poll();

int u = current.vertex;

for (Edge neighbor : graph.adjList.get(u)) {

int v = neighbor.vertex;

int weight = neighbor.weight;

if (dist[u] + weight < dist[v]) {

dist[v] = dist[u] + weight;

pq.add(new Edge(v, dist[v]));

}

}

}

return dist;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of vertices: ");

int vertices = scanner.nextInt();

Graph graph = new Graph(vertices);

System.out.print("Enter the number of edges: ");

int edgesCount = scanner.nextInt();

System.out.println("Enter the edges in the format: <u> <v> <weight>");

for (int i = 0; i < edgesCount; i++) {

int u = scanner.nextInt();

int v = scanner.nextInt();

int weight = scanner.nextInt();

graph.addEdge(u, v, weight);

}

System.out.print("Enter the source vertex: ");

int source = scanner.nextInt();

System.out.print("Enter the destination vertex: ");

int destination = scanner.nextInt();

int[] distances = dijkstra(graph, source);

if (distances[destination] == Integer.MAX\_VALUE) {

System.out.println("Destination vertex " + destination + " is unreachable from source vertex " + source);

} else {

System.out.println("Shortest distance from vertex " + source + " to vertex " + destination + ": " + distances[destination]);

}

}

}

**Output:**

