Assignment-14

Implement Dijkstra's algorithm in Java to find the shortest path in a weighted graph from a source vertex to all other vertices.

```
import java.util.Arrays;
import java.util.Comparator;
import java.util.LinkedList;
import java.util.PriorityQueue;

public class DijkstraAlgoDemo {
	private int vertices;
	private LinkedList<Node>[] adjList;

	class Node {
		int dest;
		int weight;

		Node(int d,int w){
			dest = d;
			weight = w;
		}
	}

	DijkstraAlgoDemo(int v){
		vertices = v;
		adjList = new LinkedList[v];
		for(int i = 0; i < v; i++) {
			adjList[i] = new LinkedList<>();
```

```
            }
        }

        void dijkstra(int src) {
            int dist[] = new int[vertices];
            boolean[] visited = new boolean[vertices];

            PriorityQueue<Node> pq = new
PriorityQueue<>(vertices,
                        Comparator.comparingInt(node ->
node.weight));

            Arrays.fill(dist, Integer.MAX_VALUE);
            dist[src] = 0;
            pq.add(new Node(src,0));

            while(!pq.isEmpty()) {
                Node node = pq.poll();
                int u = node.dest;

                if(!visited[u]) {
                    visited[u] = true;
                    for(Node neighbor: adjList[u]) {
                        int v = neighbor.dest;
                        int weight = neighbor.weight;

                        if(!visited[v] &&
                            dist[u] != Integer.MAX_VALUE
&&
                            weight < dist[v]) {
```

```
                                dist[v] = dist[u] + weight;
                                pq.add(new Node(v,dist[v]));

                            }
                        }
                    }
                }
                printSolution(dist);
        }


        private void printSolution(int[] dist) {
                // TODO Auto-generated method stub
                System.out.println("Vertex \t Distance from
Source");
                for(int i = 0; i < vertices ; i++) {
                        System.out.println(i +" \t\t " + dist[i]);
                }
        }

        void addEdge(int src,int dest, int weight) {
                adjList[src].add(new Node(dest,weight));
                adjList[dest].add(new Node(src,weight));
        }



        public static void main(String[] args) {
                // TODO Auto-generated method stub
                int V = 5;
                DijkstraAlgoDemo g = new DijkstraAlgoDemo(V);
```

```
        g.addEdge(0, 1, 10);
        g.addEdge(0, 2, 1);
        g.addEdge(0, 3, 4);
        g.addEdge(1, 2, 2);
        g.addEdge(1, 3, 5);
        g.addEdge(1, 4, 1);
        g.addEdge(2, 3, 2);
        g.addEdge(3, 4, 3);

        g.dijkstra(0);
    }

}
```