Amogh Karhadkar
1BM18CS014

Dijkstra's Algorithm

Code →

```
import java.util.*;
class Edge {
    int src,dst,w;
    public Edge(s,d,w){
      this.src=s;
      this.dst=d;
      this.w = w;
    }
}
class Node{
   int vertex, w;
   public Node(int v, int w){
      this.vertex=v;
      this.w=w;
   }
}
class Graph{
    List<List<Edge>> adjList =null;
    Graph(List<Edge> edges, int n){
       adjList = new ArrayList<>();
        for (int i=0; i<n; i++){
           [illegible]
           adjList.add(new ArrayList<>());
        }
        [illegible]
        for (Edge edge : edges){
            adjList.get(edge.src).add(edge);
        }
    }

  }

  class Dijkstra {

      Static void getRoute (int [] prev, int i, List <Integer> route)
      {
          if(i>0) {
              getRoute (prev, prev[i], route);
              route. add (i);
      }

      public static void shortestpath (Graph graph, int src, int n){
        PriorityQueue <Node> minHeap;
        minHeap = new PriorityQueue <>(Comparator. comparingInt(
                                       node→ node.weight));

        minHeap. add(new Node (src, 0));
        List <Integer> dist = new ArrayList<>(Collections. nCopies (n,
                                            Integer.MAX_ VALUE ))

        dist. set (src, 0) ;
        boolean[ ] done = new boolean [n];
         done [src] = true;
         int[] prev = new int[n];
         prev [src] = -1;
         List <Integer> route = new ArrayList <>();
         While ( !minHeap. isEmpty() ){
             Node node = minHeap. poll();
             int u = node. vertex;
```
[illegible]

```
for (Edge edge : graph.adjList.get(u)){

    int v= edge.dst ;

    int w = edge.w;

    if(!done[v]  && (dist.get(u) + w) < dist.get(v)){

        dist.set (v, dist.get (u) +w) ;

        prev [v] = u;

        minHeap . add (new Node(v, dist.get(v)));

    }

}

done[u] = true;

for (int i=1; i<n; i++){

    if( i!= src && dist.get(i) != Integer.MAX_VALUE){

        getRoute (prev, i, route);

        System.out.printf(" Path (%d → %d) : Min Cost
                     = %d  and Route is %c",
                   src, i, dist.get (i). route) ;

        route. clear();

    }

}
```

[illegible]