```java
class dijkstraAlgoUsingPQ
{
    //Class for edge
    public static class Edge{
        int src;
        int dst;
        int weight;
        Edge(int src,int dst, int weight){
            this.src = src;
            this.dst = dst;
            this.weight=weight;
        }
    }


    static int[] dijkstra(int V, ArrayList<ArrayList<ArrayList<Integer>>> adj, int S)
    {
        // Write your code here
        int inf = Integer.MAX_VALUE;
        int distance[] = new int[V]; // create a array to keep track of shortest distance from source S
        {
        ArrayList<ArrayList<Integer>> cl = adj.get(S);
        int len = cl.size();
        Arrays.fill(distance,inf);
        for(int i=0;i<len;i++){
            int dst = cl.get(i).get(0);
            int weight = cl.get(i).get(1);
            distance[dst]=weight; // asigning the distance to the childrens of the source but not marking as visted them.
        }                      // because your not sure wheater its the shortest distance but it sure that the children with least cost

        distance[S]=0;
        }


        /*always give the shortest distnace reachable, but  the queue makes sure thatvery fist visit to all other nodes which are not directly connected
          to source are shortest. some times when you visit direclty connected nodes to source thorugh other path is not shortest,for that we have constrain*/


        PriorityQueue<Edge> pq = new PriorityQueue<Edge>(new Comparator<Edge>(){
            public int compare(Edge e1, Edge e2){
                return e1.weight-e2.weight;
            }
        });

        //sp means shortest path
        boolean foundSP[] = new boolean[V];// keeps track of founded shortest paths
        int verticesVisited=1;
        foundSP[S]=true;
        addEdges(S,pq,adj,S,0);
        while(verticesVisited<V){  //run until shorted distance to all vertices are found.
            Edge edge = pq.poll();
            if(foundSP[edge.dst]==true){
                continue;//if already visited at the first then you dont get better at the secondtime
            }
            if(edge.weight<distance[edge.dst]){ // this statement somtimes passes but only for nodes
```

```
                                                //connected directly to the source but not alway.no other nodes pass this
            distance[edge.dst]=edge.weight;
            }
            foundSP[edge.dst]=true;
            verticesVisited++;
            addEdges(edge.dst,pq,adj,edge.src,distance[edge.dst]);
        }
        return distance;

    }

    public static void addEdges(int vertex,PriorityQueue<Edge> pq,ArrayList<ArrayList<ArrayList<Integer>>> adj,int parent,int parentDistance){
        ArrayList<ArrayList<Integer>> cl = adj.get(vertex);
        int len = cl.size();
        for(int i=0;i<len;i++){
            int dst = cl.get(i).get(0);
            int weight = cl.get(i).get(1);
            if(dst==parent){
                continue;
            }
            pq.add(new Edge(vertex,dst,weight+parentDistance));//add edge in such a way that destination has distance direclty form source/
        }
    }
}
```