07/12/20

Divyank Gupta

Write-Up

⊛ Dijsktra's Algorithm

```
class Main
{
    private 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 source, int N)
    {
        PriorityQueue <Node> minHeap;
        minHeap = new PriorityQueue <> (Comparator.comparingInt (node->node.
                                                    weight));
        minHeap.add (new Node (source, 0));

        List <Integer> dist = new ArrayList <> (Collections.nCopies (N, Integer.
                                                    MAX_VALUE));

        dist.set (source, 0);

        boolean[] done = new boolean [N];

        done [source] = true;

        int [] prev = new int [N];
        prev [source] = -1;

        List <Integer> route = new ArrayList <> ();
```


Divyank

07/12/20

Continued....

```
while ( !minHeap.isEmpty())
{
    Node node = minHeap.poll();
    int u = node.vertex;
    for ( Edge edge : graph.adjList.get (u))
    {
        int v = edge.dest ;
        int weight = edge.weight ;
        if ( ! done [v] && (dist.get(u) + weight) < dist.get(v))
        {
            dist.set (v, dist.get(u) + weight);
            prev[v] = u;
            minHeap.add (new Node (v, dist.get(v)));
        }
    }
    done[u] = true;
}

for (int i=1; i< N; i++)
{
    if (i != source && dist.get(i) != Integer.MAX_VALUE)
    {
        getRoute (prev, i, route);
        System.out.printf("Path (%d->%d): Min cost = %d and Route
                                                    is %s", source, i,
                                                    dist.get(i), route);
        route.clear();
    }
}
}
```


Dijkstra