Sangeev Kumar Singh
1BM18CS093
B3

Sangeev classmate Date Page

Pg 1

10-12-20

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<>(c);
    for (int i=0; i<n ; i++) {
    adjList.add (new ArrayList());}

  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 Shortest path (Graph graph, int Src, int
Priority Queue <Node> minHeap ;
minHeap = new Priority dueue <> (Comparator.
                Comparing Int ( node → node. weight));
min Heap. add (new Node (Src, 0));
List <Integer> dist = new Array List <> (
                Collection. n (opies (n, Integer. Max))
dist. Set (Src, 0).
boolean [] done = new boolean [n];
done [src] = true;
int [] prev = new int [n];
prev [src] = -1;

List <Integer> route = new Array <> ();
while (!minHeap. isEmpty ()) {
Node node = minHeap poll ();
int u = node. Vertex.

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 (u, dist. get (v)));
}
```

Sanjeev Kumar Singh
1BM18CS093
Prgm 9
10-12-20
B3
Dijkstra's Algo

Sanjeev classmate
Date
Page
Pg 3

```
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.println ("Path (%d → %d): min Cost
    = %d and Route is %c", src, i,
        dist.get(i).route);
    route.clear();
}
}
```