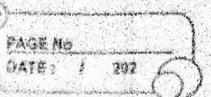


Sabrat Singh
1B MPCS094.



C N LAB
Dijkstra Algorithm

import java.util.*;

class Edge {

int src, dst, w;

public Edge (s, d, w) {

this.src = s;

this.dst = d;

this.w = w;

33

class Node {

int vertex, w;

public Node (int v, int w) {

this.vertex = v;

this.w = w;

33

class Graph {

list<Edge> adjList = null;

graph (list<Edge> edges, int n) {

adjList = new ArrayList<Edge>();

for (int i = 0; i < n; i++) {

adjList.add (new Edge (i, i));

}

for (Edge edge : edges) {

adjList.add (edge);

}

class Dijkstra {

static void getRoute (int [] prev, int i, list<Edge> adjList)

{ if (i > 0) {

last

getFronte (pren, prev[], fronte);
 fronte.add (x);

3

while (!stale) ~~while~~ Shortest Path (Graph graph, int s, int t, m){}

PriorityQueue<Node> minHeap;
 minHeap = new PriorityQueue<Node> (Comparator.comparingInt (node -> node.weight));
 minHeap.add (new Node (src, 0));
 List<Integer> dest = new ArrayList<> (Collections.synchronizedList (new Integer [m].MAX_VALUE));
 dist.set (src, 0);

boolean [] done = new boolean [m];
 done [src] = true;

int [] prev = new int [m];
 prev [src] = -1;

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

while (!minHeap.isEmpty()) {

Node mode = minHeap.poll();
 if (mode == null) continue;

for (Edge edge : graph.adjList[set(v)]) {

int w = edge.w;

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

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

prev[w] = v;

minHeap.add (new Node (w, dist.set(w)));

3

done[v] = true;

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

if (i == src && dist.set(i) == Integer.MAX_VALUE)

InSet

Set Pointe (pren, t, fronte);
System.out.println("Path (" + d + " → " + 1 - d + "): " + next);
if d and 1-d are 0 ("") (true, t, dist.set(1)).
fronte);
fronte.clear();
}

Good