Ahmad Siddiq Priaji (22/496854/PA/21370)

Activity 10

Dijkstra.java

public class Dijkstra {

    int nTown;

    double[][] map;

    double[] distance;

    int src;

    Dijkstra(double[][] map){

        this.map = map;

        nTown = map.length; // jumlah node (town)

    }

    public void solve(int src, int dst) {

        this.src = src; // set starting node

        boolean[] selected = new boolean[nTown]; // untuk mengecek apakah node sudah dipilih atau belum

        distance = new double[nTown]; // jarak terpendek ke tiap node

        for (int i=0; i<nTown; i++) {

            distance[i] = Double.MAX\_VALUE; // jarak terpendek = tak hingga

            selected[i] = false;

        }

        distance[src] = 0; // jarak ke starting node = 0

        while (true) {

            // dari node-node yang belum dipilih, pilih yang jaraknya terdekat

            int marked = minIndex(distance, selected);

            if (marked < 0) return; //seluruh node sudah dipilih

            if (distance[marked] == Double.MAX\_VALUE) return; //ada node yang tidak terhubung

            selected[marked] = true; // tandai node tersebut sebagai "sudah dipilih"

            if (marked == dst) return; //sudah sampai tujuan, selesai

            for (int j=0; j<nTown; j++) { // untuk tiap node yang terhubung dengan node yang baru saja dipilih

                if (map[marked][j]>0 && !selected[j] ) { //dan node tersebut belum dipilih

                    //hitung jaraknya

                    double newDistance = distance[marked] + map[marked][j];

                    // update jika nilainya lebih kecil

                    if (newDistance < distance[j]) distance[j] = newDistance;

                }

            }

        }

    }

    //menentukan idex node yang jaraknya terdekat

    private int minIndex(double[] distance, boolean[] selected) {

        double dist = Double.MAX\_VALUE;

        int idx = -1;

        for (int i=0; i<nTown; i++) {

            if (!selected[i] && distance[i]<dist) { //node dengan jarak terkecil yang belum dipilih

                dist = distance[i];

                idx = i;

            }

        }

        return idx;

    }

    public double getDistance(int dst) {

        return distance[dst];

    }

}

RunDijkstra.java

import java.util.\*;

public class RunDijkstra {

    static double[][] map;

    static int src;

    static int dst;

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("masukkan jumlah node");

        int nTown = sc.nextInt(); // jumlah node

        map = new double[nTown][nTown];

        // inisialisasi nilai tiap elemen dari array 2d map[][]

        for (int i=0; i<nTown; i++) {

            for (int j=0; j<nTown; j++) {

                if(i==j) map[i][j] = 0; // jarak dari node ke dirinya sendiri = 0

                else if(i>j) map[i][j] = map[j][i]; // jarak dari node i ke node j = jarak dari node j ke node i

                else {

                    System.out.println("masukkan jarak dari node " + i + " ke node " + j);

                    map[i][j] = sc.nextDouble();

                }

            }

        }

        Dijkstra dj = new Dijkstra(map);

        dj.solve(src, dst);

        System.out.println(dj.getDistance(dst));

        sc.close();

    }

}

Hasil Dijkstra dari a ke h (0 ke 7)

