Dokumentasi Tugas Program 1

Nama : Faizal Bima Prayudha

Nima : 1103120176

1. **Deskripsi masalah dan tujuan**

Sebuah permasalahan dimana terdapat kota yang saling terhubung dengan sungai, dan akan dicari jalur tercepat dari kota awal menuju titik akhir.namun setiap kota tersebut memiliki memiliki panjang jalur yang berbeda-beda ke setiap kotanya. program yang saya buat ini akan mencari jalur yang dapat menghubungkan kota awal ke kota tujuan, dengan jarak yang paling optimal.

1. **Desain Program dan metode algoritma**

* **Input**

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

n[i] = new Node();

n[i].setData("n-" + i);

}

n[0].setName("Ravenna");

n[1].setName("Rimini");

n[2].setName("ferrara");

n[3].setName("forli");

n[4].setName("Cesena");

n[5].setName("Faenza");

n[6].setName("Imola");

n[7].setName("Emilia");

n[8].setName("Terme");

n[9].setName("Carpi");

n[10].setName("Piacenza");

n[11].setName("Bobbia");

n[0].setH(0);

n[1].setH(0.5);

n[2].setH(5);

n[3].setH(2);

n[4].setH(4.5);

n[5].setH(4);

n[6].setH(5);

n[7].setH(6);

n[8].setH(7);

n[9].setH(8);

n[10].setH(10);

n[11].setH(10.5);

n[11].setNeighbors(n[10], 5);

n[11].setNeighbors(n[8], 3);

n[11].setNeighbors(n[4], 15);

n[10].setNeighbors(n[9], 3);

n[10].setNeighbors(n[8], 3);

n[9].setNeighbors(n[7], 2);

n[9].setNeighbors(n[5], 8);

n[8].setNeighbors(n[7], 2);

n[8].setNeighbors(n[5], 3);

n[7].setNeighbors(n[6], 2);

n[6].setNeighbors(n[5], 1);

n[6].setNeighbors(n[3], 3);

n[5].setNeighbors(n[3], 2);

n[5].setNeighbors(n[4], 6);

n[4].setNeighbors(n[1], 5);

n[3].setNeighbors(n[4], 2);

n[3].setNeighbors(n[0], 3);

n[2].setNeighbors(n[0], 6);

n[2].setNeighbors(n[6], 3);

n[1].setNeighbors(n[0], 1);

new AStar().traverse(n[11], n[0]);

* **Output**

private void printPath(Node node) {

System.out.println("kota yang ditempuh : ");

System.out.print(node.getName()+" <- ");

while (node.getParent() != null) {

node = node.getParent();

System.out.print(node.getName()+" <- ");

}

private void printOpen(PriorityQueue<Node> openList){

System.out.print("[");

for (Node openList1 : openList) {

System.out.print(openList1.getName() + " = " + fVals.get(openList1) + " , ");

}

System.out.print("]");

}

private void printClosed(ArrayList<Node> closedList){

System.out.print("[");

for (Node closedList1 : closedList) {

System.out.print(closedList1.getName() + " = " + fVals.get(closedList1) + " , ");

}

System.out.print("]");

}

* **Desain Proses**

public void traverse(Node start, Node end) {

openList.clear();

closedList.clear();

gVals.put(start, 0.0);

fVals.put(start, 0.0);

openList.add(start);

while (!openList.isEmpty()) {

Node current = openList.element();

System.out.println("open ketika best node akan di evaluasi :");

printOpen(openList);

System.out.println("");

System.out.println("closed ketika best node akan di evaluasi :");

printClosed(closedList);

System.out.println("");

System.out.println("Best Node yang akan di evaluasi ");

System.out.print(current.getName());

System.out.println("");

if (current.equals(end)) {

System.out.println("");

System.out.println("");

System.out.println("kota telah ditemukan");

System.out.println("Jarak yang ditempuh = "+gVals.get(current));

printPath(current);

return;

}

closedList.add(openList.poll());

ArrayList<Ketetanggaan> neighbors = current.getNeighbors();

for (Ketetanggaan neighbor : neighbors) {

double gScore = gVals.get(current) + neighbor.getJarak();

double fScore = gScore + neighbor.getTetangga().getH();

if (closedList.contains(neighbor)) {

if (gVals.get(neighbor) == null) {

gVals.put(neighbor.getTetangga(), gScore);

}

if (fVals.get(neighbor) == null) {

fVals.put(neighbor.getTetangga(), fScore);

}

if (fScore >= fVals.get(neighbor)) {

continue;

}

}

if (!openList.contains(neighbor) || fScore < fVals.get(neighbor)) {

neighbor.getTetangga().setParent(current);

gVals.put(neighbor.getTetangga(), gScore);

fVals.put(neighbor.getTetangga(), fScore);

if (!openList.contains(neighbor)) {

openList.add(neighbor.getTetangga());

}

}

}

System.out.println("open ketika best node telah di evaluasi :");

printOpen(openList);

System.out.println("");

System.out.println("closed ketika best node telah di evaluasi :");

printClosed(closedList);

System.out.println("");

System.out.println("");

System.out.println("");

}

System.out.println("FAIL");

}

1. **Screenshot program** 

