TUGAS KECIL 3 IF2122 STRATEGI ALGORITMA
IMPLEMENTASI ALGORITMA A* UNTUK MENENTUKAN LINTASAN TERPENDEK

LAPORAN TUGAS

Diajukan sebagai laporan dari tugas kecil tiga mata kuliah Strategi Algoritma IF2122 pada Semester II Tahun Akademik 2020-2021

oleh

Epata Tuah	13519120
Habibina Arif Muzayyan	13519125

TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2021

A. Algoritma A*

Algoritma A* merupakan algoritma yang sering digunakan dalam penentuan rute terpendek. Tujuan utama algoritma ini adalah menghindari jalur yang memiliki bobot terbesar. Fungsi algoritma A* adalah f(n) = g(n) + h(n).

- g(n) adalah bobot yang dibutuhkan untuk mencapai simpul n
- h(n) adalah perkiraan bobot dari simpul n ke simpul tujuan
- f(n) adalah perkiraan total jalur melalui simpul n ke simpul tujuan

B. Kode Program

1. Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Starif
{
    static class Program
    {
        /// <summary>
        ///  The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.SetHighDpiMode(HighDpiMode.SystemAware);
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Starif());
        }
    }
}
```

2. Starif.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Globalization;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using Microsoft.Msagl.Drawing;
```

```
namespace Starif
{
    public partial class Starif : Form
    {
        private OpenFileDialog browse = new OpenFileDialog();
        private string[] input;
        private Graf graf;
        public Starif()
        {
            InitializeComponent();
        }

        //browse
        private void button1_Click(object sender, EventArgs e)
        {
            comboBox2.Items.Clear();
            comboBox3.Items.Clear();
            //string directory;
            Microsoft.Msagl.Drawing.Graph grafVisualization = new
Microsoft.Msagl.Drawing.Graph("graf");
            browse.Filter = "*.txt (file berekstensi txt) | *.txt";
            if (browse.ShowDialog() == DialogResult.OK)
            {
                int i;
                string fileDirectory = browse.FileName;
                string filename = browse.SafeFileName;
                input = File.ReadAllLines(fileDirectory);
                label4.Text = filename;
                graf = new Graf(input);
                //menambahkan simpul simpul pada comboBox
                for (i = 1; i <= graf.getGraphNode().Count; i++)
                {
                    comboBox2.Items.Add(graf.getGraphNode()[i-1].getNamaNode());
                    comboBox3.Items.Add(graf.getGraphNode()[i-1].getNamaNode());
                }
                //visualisasi graf awal
                grafVisualization = FrontEndUtility.grafVisualization(graf);
                gViewer1.Graph = grafVisualization;
            }
        }

        //submit
        private void button2_Click(object sender, EventArgs e)
        {
            if (comboBox2.Text != "awal" && comboBox3.Text != "tujuan")
            {
                label7.Text = "";
                //label4.Text = "Jalur";
                Microsoft.Msagl.Drawing.Graph grafVisualization = new
Microsoft.Msagl.Drawing.Graph("graf");
                int i, j;
                // input simpul awal dan tujuan
                string awal = comboBox2.Text;
                string tujuan = comboBox3.Text;
```

```
                graf = new Graf(input);
                Node nodeAwal = graf.searchNode(awal);
                Node nodeTujuan = graf.searchNode(tujuan);

                if (!graf.searchNode(nodeAwal) || !graf.searchNode(nodeTujuan))
                {
                    //Console.WriteLine("Simpul tidak ada");
                }
                else
                {
                    //ALGORITMA A*
                    List<Edge> rute = new List<Edge>();
                    List<Node> ruteNode = new List<Node>();
                    Node nodeAwalAlgo = nodeAwal;


                    j = 0;
                    while (nodeAwalAlgo != nodeTujuan) //selama nodeAwalAlgo bukan
nodeTujuan
                    {
                        ruteNode.Add(nodeAwalAlgo);
                        Dictionary<Node, double> results = new Dictionary<Node,
double>();
                        int jumlahEdges = nodeAwalAlgo.getEdges().Count;
                        i = 0;
                        j += 1;
                        while (i < jumlahEdges)
                        {
                            //g(n) = distance dari nodeAwalAlgo ke n
                            double sumG = 0;
                            if (rute.Count>0){
                                foreach (Edge edge in rute)
                                {
                                    sumG += edge.getBobot();
                                }
                            }
                            double g = nodeAwalAlgo.getEdges()[i].getBobot() + sumG;
                            //h(n) = straight line distance from n ke nodeTujuan
                            double h =
Edge.euclideanDistance(nodeAwalAlgo.getEdges()[i].getNext().getKoordinatNode(),
nodeTujuan.getKoordinatNode());
                            //f(n) = g(n) + h(n)
                            double f = g + h;
                            if
(!ruteNode.Contains(nodeAwalAlgo.getEdges()[i].getNext()))
                            {
                                results.Add(nodeAwalAlgo.getEdges()[i].getNext(),
f);
                            }

                            i += 1;
                        }
                        Node tempNodeAwal = nodeAwalAlgo;
                        if (results.Count > 0)
                        {
```

```
                            nodeAwalAlgo = results.OrderBy(KeyValuePair =>
KeyValuePair.Value).First().Key;
                            Edge edgeRute = new Edge(tempNodeAwal, nodeAwalAlgo);
                            rute.Add(edgeRute);
                        }
                        else
                        {
                            break;
                        }
                    }

                    //visualisasi graf hasil
                    grafVisualization =
FrontEndUtility.colorizedGrafVisualization(graf, rute);
                    gViewer1.Graph = grafVisualization;

                    //hitung jarak rute yang dihasilkan
                    double sumBobot = 0;
                    foreach (Edge edge in rute)
                    {
                        sumBobot += edge.getBobot() * 100000;
                    }
                    label7.Text = sumBobot.ToString() + " m";

                    //kasus tidak ada tujuan
                    if (nodeAwalAlgo != nodeTujuan)
                    {
                        grafVisualization = FrontEndUtility.grafVisualization(graf);
                        gViewer1.Graph = grafVisualization;
                        label7.Text = "Tidak ada jalur";
                    }
                }
            }
            else
            {
                label4.Text = "Tidak ada jalur";
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void label5_Click(object sender, EventArgs e)
        {

        }

        private void label8_Click(object sender, EventArgs e)
```

```
        {

        }
    }
}
```

3. FrontEndUtility.cs

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
//using System.Threading.Tasks

namespace Starif
{
    class FrontEndUtility
    {
        //Menghapus edge duplikat pada frontend
        public static Graf deletedDuplicatedEdgesGraph(Graf graf, int N)
        {
            List<Edge> noDuplicateEdges = new List<Edge>();
            List<Node> newNodes = new List<Node>();
            Graf newGraph = new Graf(N);
            foreach (Node node in graf.getGraphNode())
            {
                List<Edge> newEdges = new List<Edge>();
                foreach (Edge edge in node.getEdges())
                {
                    string namaNodeTestedEdge = edge.getNode().getNamaNode();
                    string namaNextNodeTestedEdge = edge.getNext().getNamaNode();
                    bool exist = noDuplicateEdges.Any(x => x.getNode().getNamaNode()
== namaNextNodeTestedEdge && x.getNext().getNamaNode() == namaNodeTestedEdge);
                    if (!exist)
                    {
                        noDuplicateEdges.Add(edge);
                        newEdges.Add(edge);
                    }
                }
                Node newNode = new Node(node.getNamaNode(),
node.getKoordinatNode());
                newNode.setEdges(newEdges);
                newNodes.Add(newNode);
            }
            newGraph.setListNodes(newNodes);
            return newGraph;

        }

        //Memvisualisasikan graf awal
        public static Microsoft.Msagl.Drawing.Graph grafVisualization(Graf graf)
        {
            Microsoft.Msagl.Drawing.Graph grafVisualization = new
```

```
Microsoft.Msagl.Drawing.Graph("graf");
            int N = graf.getGraphNode().Count;
            //gViewer1.Controls.Clear();
            Graf frontEndGraph = FrontEndUtility.deletedDuplicatedEdgesGraph(graf,
graf.getGraphNode().Count);
            foreach (Node node in frontEndGraph.getGraphNode())
            {
                if (node.getEdges().Count == 0)
                {
                    grafVisualization.AddNode(node.getNamaNode());
                }
                foreach (Edge edge in node.getEdges())
                {
                    double roundedBobot = Math.Round(edge.getBobot() * 100000, 2);
                    var garis =
grafVisualization.AddEdge(edge.getNode().getNamaNode(), roundedBobot.ToString(),
edge.getNext().getNamaNode());
                    garis.Attr.ArrowheadAtTarget =
Microsoft.Msagl.Drawing.ArrowStyle.None;
                    garis.Attr.ArrowheadAtSource =
Microsoft.Msagl.Drawing.ArrowStyle.None;
                }
            }
            return grafVisualization;
        }

        //Memvisualisasikan graf hasil
        public static Microsoft.Msagl.Drawing.Graph colorizedGrafVisualization(Graf
graf, List<Edge> rute)
        {
            int i;
            Microsoft.Msagl.Drawing.Graph grafVisualization = new
Microsoft.Msagl.Drawing.Graph("graf");
            int N = graf.getGraphNode().Count;
            Graf frontEndGraph = FrontEndUtility.deletedDuplicatedEdgesGraph(graf,
graf.getGraphNode().Count);
            foreach (Node node in frontEndGraph.getGraphNode())
            {
                if (node.getEdges().Count == 0)
                {
                    grafVisualization.AddNode(node.getNamaNode());
                }
                foreach (Edge edge in node.getEdges())
                {
                    double roundedBobot = Math.Round(edge.getBobot() * 100000, 2);
                    var garis =
grafVisualization.AddEdge(edge.getNode().getNamaNode(), roundedBobot.ToString(),
edge.getNext().getNamaNode());
                    garis.Attr.ArrowheadAtTarget =
Microsoft.Msagl.Drawing.ArrowStyle.None;
                    garis.Attr.ArrowheadAtSource =
Microsoft.Msagl.Drawing.ArrowStyle.None;

                    for (i = 0; i < rute.Count; i++)
                    {
```

```
                        bool edgeFirstCheck = rute[i].getNode().getNamaNode() ==
edge.getNode().getNamaNode() && rute[i].getNext().getNamaNode() ==
edge.getNext().getNamaNode();
                        bool edgeSecondCheck = rute[i].getNode().getNamaNode() ==
edge.getNext().getNamaNode() && rute[i].getNext().getNamaNode() ==
edge.getNode().getNamaNode();
                        if (edgeFirstCheck || edgeSecondCheck)
                        {
                            garis.Attr.Color = Microsoft.Msagl.Drawing.Color.Blue;
                            if (i == 0)
                            {

grafVisualization.FindNode(rute[i].getNode().getNamaNode()).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.LightBlue;

grafVisualization.FindNode(rute[i].getNext().getNamaNode()).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.LightGreen;
                            }
                            else if (i == rute.Count - 1)
                            {

grafVisualization.FindNode(rute[i].getNode().getNamaNode()).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.Turquoise;

grafVisualization.FindNode(rute[i].getNext().getNamaNode()).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.LightGreen;
                            }
                            else
                            {

grafVisualization.FindNode(rute[i].getNode().getNamaNode()).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.Turquoise;

grafVisualization.FindNode(rute[i].getNext().getNamaNode()).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.Turquoise;
                            }
                        }
                    }
                }
            }
            return grafVisualization;
        }
    }
}
```

4. Graf.cs

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Globalization;
//using System.Linq;
using System.Text;
//using System.Threading.Tasks
```

```
namespace Starif
{
    class Graf
    {
        private List<Node> nodeGraf;

        //user defined graf constructor
        public Graf(string[] input)
        {
            int i, j;
            int N = Int32.Parse(input[0]);
            this.nodeGraf = new List<Node>(N);
            for (i = 1; i <= N; i++)
            {

                double x = Convert.ToDouble(input[i].Split(" ")[1],
CultureInfo.InvariantCulture);
                double y = Convert.ToDouble(input[i].Split(" ")[2],
CultureInfo.InvariantCulture);
                double z = x + y;

                Coordinate nodeCoordinate = new Coordinate(x, y);
                string namaNode = input[i].Split(" ")[0];
                //Node node = new Node(namaNode, nodeCoordinate);
                this.addNode(namaNode, nodeCoordinate);
            }
            for (i = 0; i < N; i++)
            {
                for (j = 0; j < N; j++)
                {
                    if (input[i + N + 1].Split("  ")[j] == "1")
                    {
                        this.nodeGraf[i].addEdge(this.nodeGraf[j]);
                    }
                }
            }
        }

        //user defined graf constructor
        public Graf(int N){
            this.nodeGraf = new List<Node>(N);
        }

        //user defined graf constructor
        public Graf(List<Node> nodeGraf){
            this.nodeGraf = nodeGraf;
        }

        //graf copy constructor
        public Graf(Graf G)
        {
            this.nodeGraf = G.nodeGraf;
        }
```

```
        //menambahkan node ke graf dan mengembalikan node tersebut
        public Node addNode(string namaNode, Coordinate koordinatNode){
            Node addedNode = new Node(namaNode, koordinatNode);
            nodeGraf.Add(addedNode);
            return addedNode;
        }

        //menambahkan node (prosedural)
        public void addNode(Node node){
            this.nodeGraf.Add(node);
        }

        //menghapus node
        public void removeNode(Node node){
            this.nodeGraf.Remove(node);
        }

        //getter list node graf
        public List<Node> getGraphNode(){
            return this.nodeGraf;
        }

        //mengecek apakah ada node pada graf
        public bool searchNode(Node node){
            bool ketemu = false;
            int i = 0;
            while (!ketemu && i < this.nodeGraf.Count){
                if (nodeGraf[i] == node){
                    ketemu = true;
                }
                i+=1;
            }
            return ketemu;
        }

        //mengecek apakah ada node pada graf dan mengembalikan node tersebut jika ada
        public Node searchNode(string namaNode){
            int i = 0;
            while (i < this.nodeGraf.Count){
                if (nodeGraf[i].getNamaNode() == namaNode){
                    return nodeGraf[i];
                }
                i+=1;
            }
            return null;
        }

        //setter list node graf
        public void setListNodes(List<Node> newNodes)
        {
            this.nodeGraf = newNodes;
        }
    }
}
```

5. Node.cs

```
using System;
using System.Collections.Generic;
//using System.Linq;
using System.Text;
//using System.Threading.Tasks

namespace Starif
{
    class Node
    {
        private string namaNode;
        private Coordinate koordinatNode;
        private List<Edge> edges = new List<Edge>();

        //default constructor
        public Node(){
            this.namaNode = "XXX";
            this.koordinatNode = new Coordinate();
        }

        //user defined constructor
        public Node(string namaNode, Coordinate koordinatNode){
            this.namaNode = namaNode;
            this.koordinatNode = koordinatNode;
        }

        //getter nama node
        public string getNamaNode(){
            return this.namaNode;
        }

        //getter koordinat node
        public Coordinate getKoordinatNode(){
            return this.koordinatNode;
        }

        //getter list edge node
        public List<Edge> getEdges(){
            return this.edges;
        }

        //setter nama node
        public void setNamaNode(string namaNode){
            this.namaNode = namaNode;
        }

        //setter koordinat node
        public void setKoordinatNode(Coordinate koordinatNode){
            this.koordinatNode = koordinatNode;
        }

        //menambahkan edge pada node dengan suatu next node
        public void addEdge(Node nextNode){
            int i;
```

```
            bool adaNext = false;
            bool adaCurr = false;
            i = 0;
            List<Edge> edgeCurrNode = this.getEdges();
            List<Edge> edgeNextNode = nextNode.getEdges();
            while (!adaCurr && i<edgeCurrNode.Count){
                if (edgeCurrNode[i].getNode()==this &&
edgeCurrNode[i].getNext()==nextNode){
                    adaCurr = true;
                }
                i+=1;
            }
            i = 0;
            while (!adaNext && i < edgeNextNode.Count){
                if (edgeNextNode[i].getNode()==nextNode &&
edgeNextNode[i].getNext()==this){
                    adaNext = true;
                }
                i +=1;
            }
            double bobot =
Edge.euclideanDistance(this.getKoordinatNode(),nextNode.getKoordinatNode());
            if (!adaCurr){
                this.edges.Add(new Edge(this,bobot,nextNode));
            }
            if (!adaNext){
                nextNode.getEdges().Add(new Edge(nextNode,bobot,this));
            }
        }

        //setter list edge node
        public void setEdges(List<Edge> newEdges)
        {
            this.edges = newEdges;
        }

        //menghapus edge node dengan suatu next node
        public void removeEdge(Node nextNode){
            int i = 0;
            foreach (Edge edge in this.edges)
            {
                if(edge.getNext() == nextNode){
                    break;
                }
                i += 1;
            }
            this.edges.RemoveAt(i);
        }
    }
}
```

6. Edge.cs

```
using System;
using System.Collections.Generic;
```

```
//using System.Linq;
using System.Text;
//using System.Threading.Tasks

namespace Starif
{
    class Edge
    {
        private Node currNode;
        private double bobot;
        private Node nextNode;

        //default constructor
        public Edge(){
            Node defaultNode = new Node();
            this.currNode = defaultNode;
            this.bobot = 0.0;
            this.nextNode = defaultNode;
        }

        //user defined constructor
        public Edge(Node currNode, double bobot, Node nextNode){
            this.currNode = currNode;
            this.bobot = bobot;
            this.nextNode = nextNode;
        }

        //user defined constructor
        public Edge(Node currNode, Node nextNode){
            this.currNode = currNode;
            this.bobot = euclideanDistance(currNode.getKoordinatNode(),
nextNode.getKoordinatNode());
            this.nextNode = nextNode;
        }

        //getter node mengembalikan node
        public Node getNode(){
            return this.currNode;
        }

        //getter bobot
        public double getBobot(){
            return this.bobot;
        }

        //getter next node
        public Node getNext(){
            return this.nextNode;
        }

        //setter node
        public void setNode(Node currNode){
            this.currNode = currNode;
        }
```

```
        //setter bobot
        public void setBobot(double bobot){
            this.bobot = bobot;
        }

        //setter next node
        public void setNext(Node next){
            this.nextNode = next;
        }

        //fungsi jarak euclidean antara dua koordinat
        public static double euclideanDistance(Coordinate A, Coordinate B){
            return(Math.Sqrt(Math.Pow((A.getX()-B.getX()),2) +
Math.Pow((A.getY()-B.getY()),2)));
        }
    }
}
```

7. Coordinate.cs

```
using System;
using System.Collections.Generic;
//using System.Linq;
using System.Text;
//using System.Threading.Tasks

namespace Starif
{
    class Coordinate
    {
        private double x;
        private double y;

        //default constructor
        public Coordinate(){
            this.x = 0.0;
            this.y = 0.0;
        }

        //user defined constructor
        public Coordinate(double x, double y){
            this.x = x;
            this.y = y;
        }

        //getter x
        public double getX(){
            return this.x;
        }

        //getter y
        public double getY(){
            return this.y;
        }
```

```
        //setter x
        public void setX(double x){
            this.x = x;
        }

        //setter y
        public void setY(double y){
            this.y = y;
        }
    }
}
```

C. Input

1. itb.txt

```
15
GerbangDepanITB -6.8932133744187505 107.61044212570688
GerbangDaysumITB -6.887397321822254 107.61146990240869
SimpangBonbin -6.893881521087048 107.60844773085253
SimpangDago -6.885223470684316 107.61368899017947
SimpangBorromeus -6.893769608067117 107.61297018362949
SimpangCiungwanara -6.893602175696701 107.61194080921429
SimpangDaysum -6.887403235761938 107.61356230552272
SimpangDaysumDago -6.887390161972454 107.61356928659616
BelokanBonbinSBM -6.887897127386889 107.60825836198674
SimpangTamansariGanyang -6.894898626762222 107.6088354979589
KantinGanyang -6.894795820919279 107.61040562750797
SimpangCiungwanaraGanyang -6.894750828889471 107.6117228492062
SimpangPelesiran -6.896862261756674 107.60965599692821
PDAMTirtawening -6.896613657179255 107.61054014180957
SimpangCiungwanaraBadakSinga -6.8976675030627685 107.61148299891029
0 0 1 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 1 0 1 0 0 0 0 0 0
1 0 0 0 0 0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 0 1 1 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1 0 0 0 0 0 0 0
1 0 0 0 1 0 0 0 0 0 0 1 0 0 0
0 1 0 1 0 0 0 1 0 0 0 0 0 0 0
0 0 0 1 1 0 1 0 0 0 0 0 0 0 0
0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 1 0 1 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0 0 0
0 0 0 0 0 1 0 0 0 0 1 0 0 0 1
0 0 0 0 0 0 0 0 0 1 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 1 0 1 0
```

2. alunalun.txt

```
13
SmpgAlunAlunBRI -6.921252168327341 107.60774492081428
SmpgAlunAlunPendopo -6.922574225785856 107.6075847615789
SmpgAlunAlunDalemKaum -6.92238751228416 107.6063652097867
SmpgKepatihanDewiSatrika -6.923447476977291 107.60630338474536
SmpgKepatihanKarangAnyar -6.923120258856476 107.60390660780551
SmpgCibadakDalemKaum -6.922097684596873 107.60401271422603
SmpgJSudirmanAsiaAfrika -6.92083173314835 107.60408157258428
SmpgAsiaAfrikaBraga -6.921506166599615 107.60981619288029
SmpgBragaNaripan -6.919702954233687 107.60996122486874
SmpgBanceuyABC -6.9189599130003705 107.60665694022833
SmpgABCOttoIskandarDinata -6.918299099834981 107.60428165497396
SmpgAsiaAfrikaTamblong -6.921729735354168 107.61199267445095
SmpgDalemKaumLengkongKecil -6.9230655011708855 107.61184851502033
0 1 0 0 0 0 0 1 0 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 0 1
0 1 0 1 0 1 0 0 0 0 0 0 0
0 0 1 0 1 0 0 0 0 0 0 0 0
0 0 0 1 0 1 0 0 0 0 0 0 0
0 0 1 0 1 0 1 0 0 0 0 0 0
1 0 0 0 0 1 0 0 0 0 1 0 0
1 0 0 0 0 0 0 0 1 0 0 1 0
0 0 0 0 0 0 0 0 1 0 1 0 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0 0
0 0 0 0 0 0 1 0 0 1 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 1
0 1 0 0 0 0 0 0 0 0 0 1 0
```

3. buahbatu.txt

```
10
MetroIndahMall -6.9400075995144705 107.65885487749868
SmpgSoettaRancabolang -6.9391067971791385 107.66386855584571
RSIAHumanaPrima -6.940808442782671 107.66375537150235
SmpgSDNRancabolang -6.950054571142093 107.66222472036802
SmpgRancabolangCiwastra -6.959009869645997 107.6598137945495
SmpgMargacintaIAdjie -6.954091691352396 107.64021230438665
SmpgSamsatBandungTimur -6.945481160335872 107.64188343866806
EdelweissHospital -6.943042881553332 107.64962154398019
PerumahanKiaraSariAsli -6.948699754349106 107.64600172123097
PerumahanMargahayuRaya -6.947389089198867 107.6586338593822
0 1 0 0 0 0 1 0 0 0
1 0 1 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 1
0 0 1 0 1 0 0 0 0 0
0 0 0 1 0 1 0 0 0 0
0 0 0 0 1 0 1 0 0 0
0 0 0 0 0 1 0 1 0 0
1 0 0 0 0 0 1 0 1 0
```

```
0 0 0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0 0 0
```

4. jambi.txt

```
12
SmpgAurduri -1.616914349341046 103.52905679076173
JembatanAurduri -1.5749869340062312 103.56814473982128
SmpgRimbo -1.6239058663835677 103.550862190936
SmpgPaal10 -1.6697007788807325 103.59557995331541
SmpgKawat -1.6169174175056973 103.60418266732414
SmpgTuguJuangJambi -1.6196807528521548 103.59027154493243
SmpgPulai -1.602238562885402 103.60387100010105
SmpgPalembang -1.6504094751775484 103.6241811061627
SmpgPalmerah -1.6406717206391925 103.63909109588666
SmpgAdipura -1.6241384867789346 103.63126934559735
SmpgJelutung -1.6023832183380788 103.61831634705392
SmpgRawasari -1.593715168713209 103.61507983652811
0 1 1 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0
1 0 0 1 0 1 0 0 0 0 0 0
0 0 1 0 1 0 0 1 0 0 0 0
0 0 0 1 0 1 1 0 0 0 0 0
0 0 1 0 1 0 1 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0 0 1
0 0 0 1 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 1 0 1 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 1 0 1
0 0 0 0 0 0 1 0 0 0 1 0
```

5. tambahan1.txt

```
10
Serang -6.117100268686004 106.1542902492578
Jakarta -6.179608982807102 106.84346239911925
Cikampek -6.3916431532957185 107.44333429570455
Bandung -6.916444793432949 107.616553826609
Cirebon -6.731470405325928 108.54815747503056
Semarang -7.00769658469111 110.43874413276158
Surakarta -7.576446556653895 110.82229567789823
Yogyakarta -7.784111621864054 110.36661188585784
Surabaya -7.256878916263042 112.75741601910968
Malang -7.966488947915297 112.63181837337143
0 1 0 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0
0 1 0 1 1 0 0 0 0 0
```

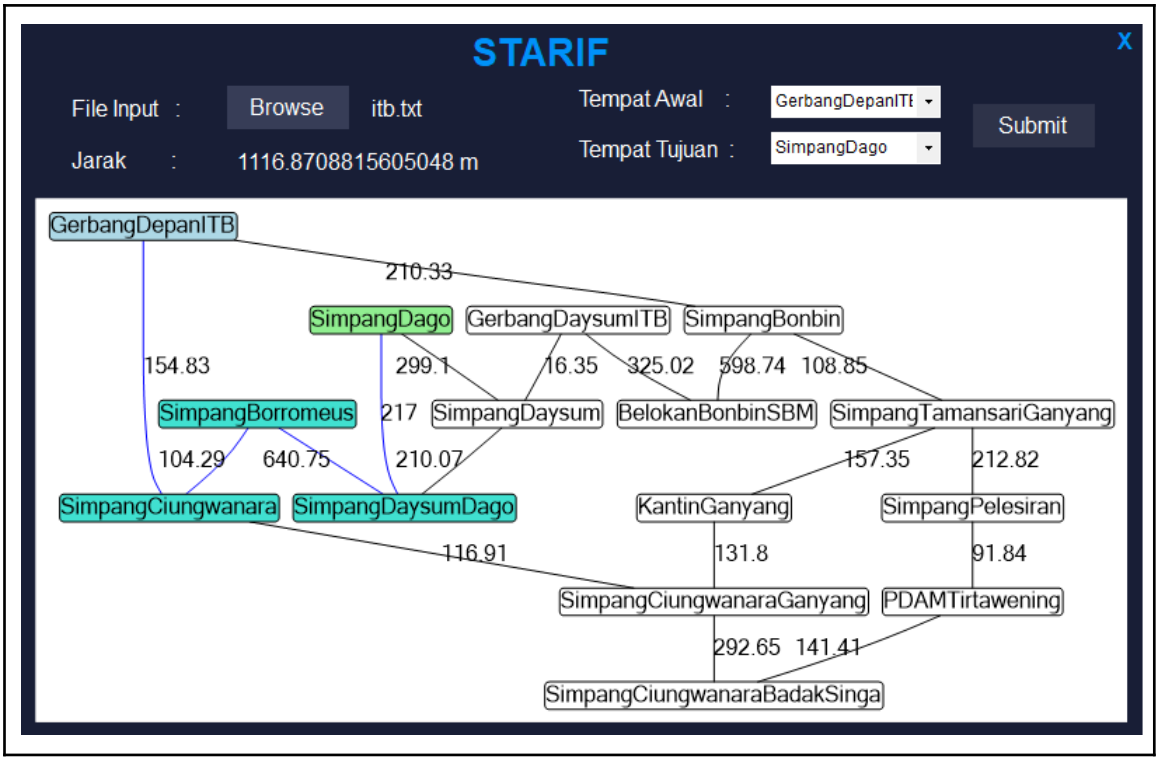
```
0 0 1 0 1 0 0 0 0 0
0 0 1 1 0 1 0 0 0 0
0 0 0 0 1 0 1 0 0 0
0 0 0 0 0 1 0 1 1 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 1 0 0 1
0 0 0 0 0 0 0 0 1 0
```

6. tambahan2.txt

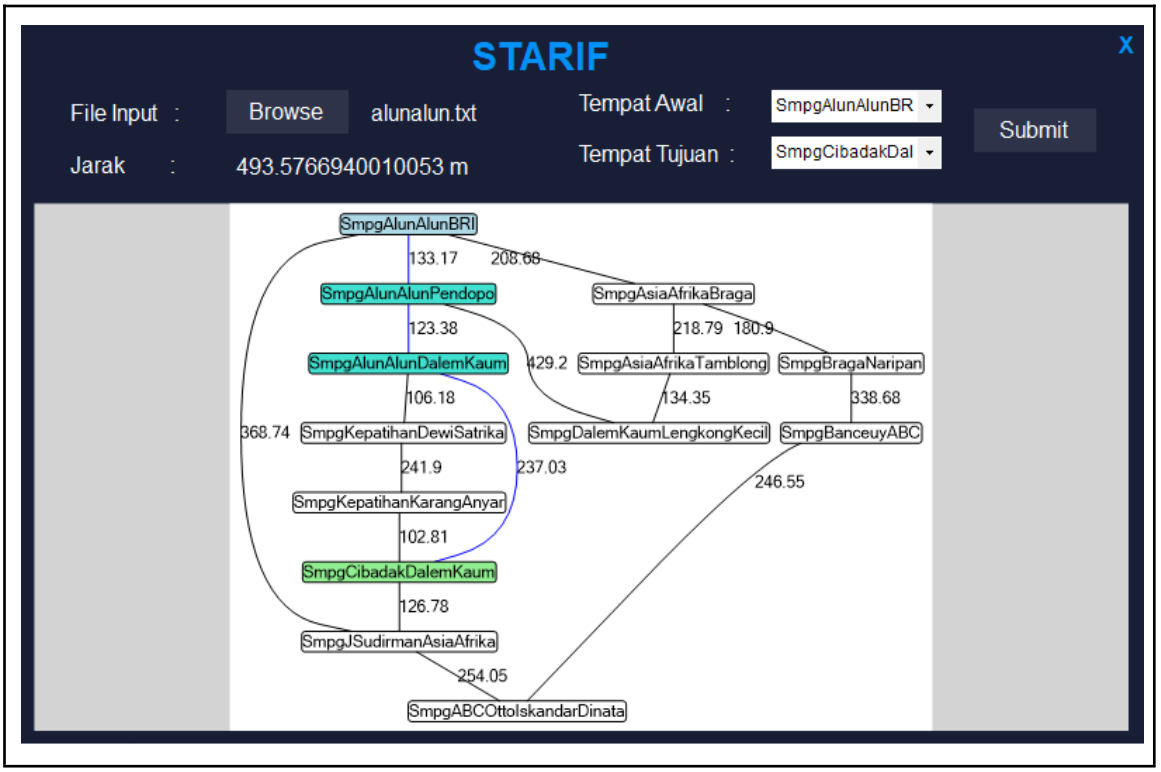
```
9
Bandar_Lampung -5.400751818589118 105.26367243509051
Palembang -2.9867973504530827 104.78161495594124
Bengkulu -3.7803862991908717 102.266354439598
Jambi -1.6051745628333463 103.60967504396503
Pekanbaru 0.5113643299197684 101.44086258856557
Padang -0.9515609756387432 100.35786398888106
Medan 3.5930914725778407 98.67251477209918
Banda_Aceh 5.549880204233839 95.3251422765239
Jakarta -6.179608982807102 106.84346239911925
0 1 0 0 0 0 0 0 0
1 0 1 1 0 0 0 0 0
0 1 0 0 0 1 0 0 0
0 1 0 0 1 0 0 0 0
0 0 0 1 0 1 1 0 0
0 0 1 0 1 0 0 0 0
0 0 0 0 1 0 0 1 0
0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0
```

D. Output

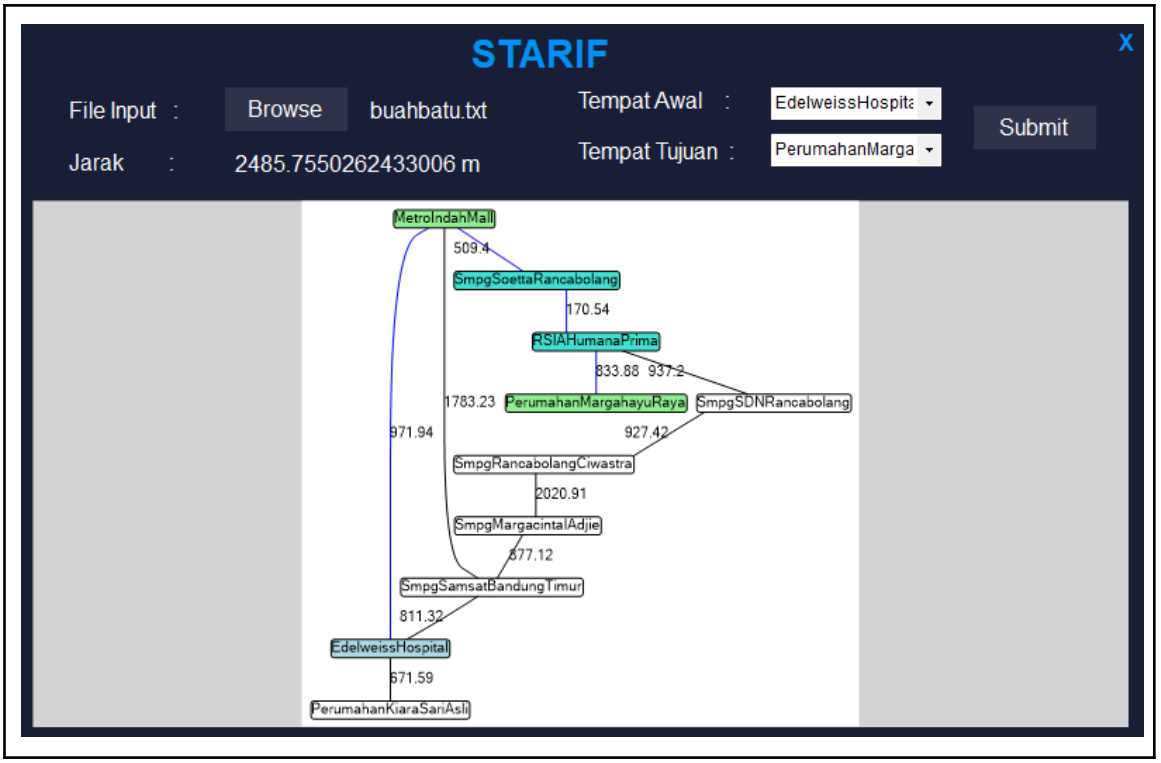
1. itb.txt (GerbangDepanITB → SimpangDago)



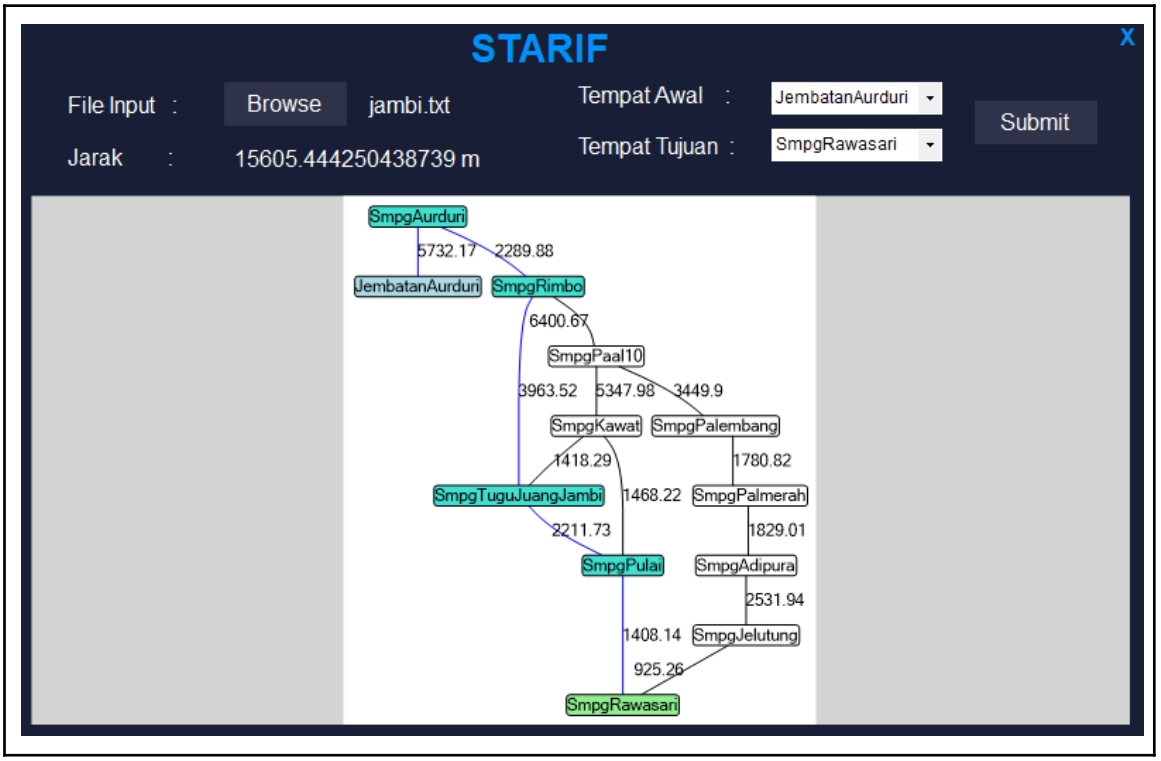
2. alunalun.txt (SmpgAlunAlunBRI → SmpgCibadakDalemKaum)



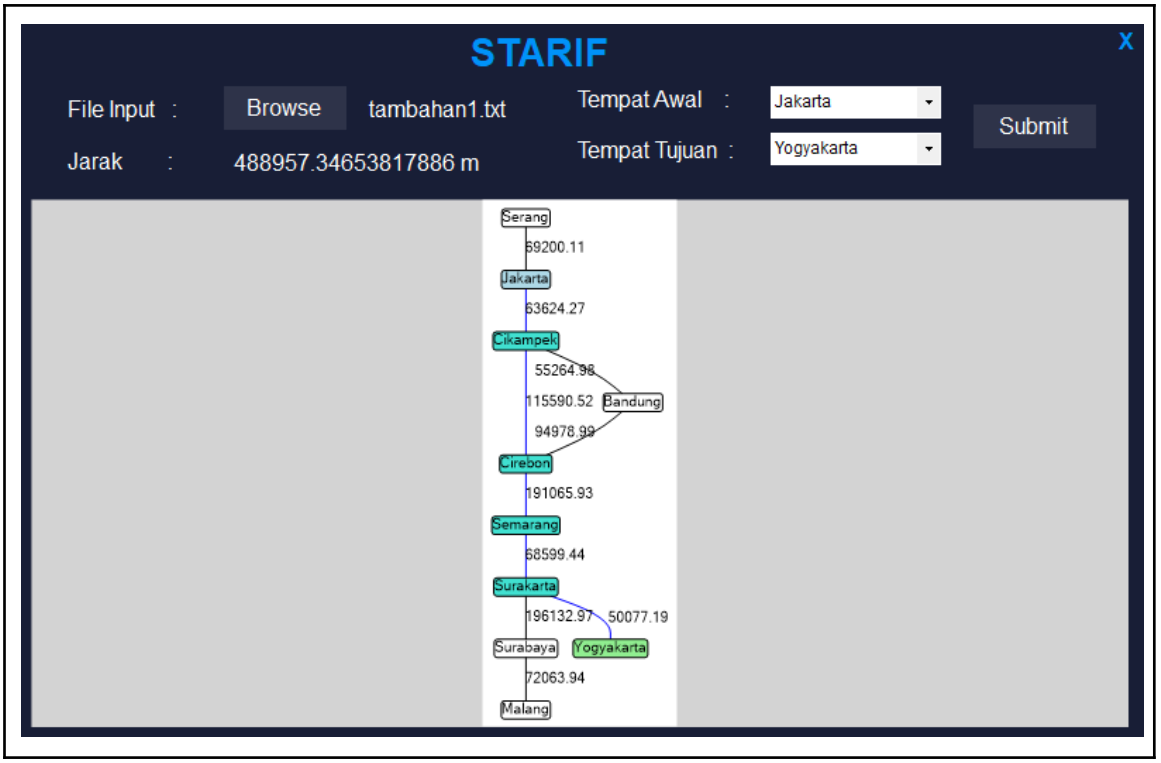
3. buahbatu.txt (EdelweissHospital → PerumahanMargahayuRaya



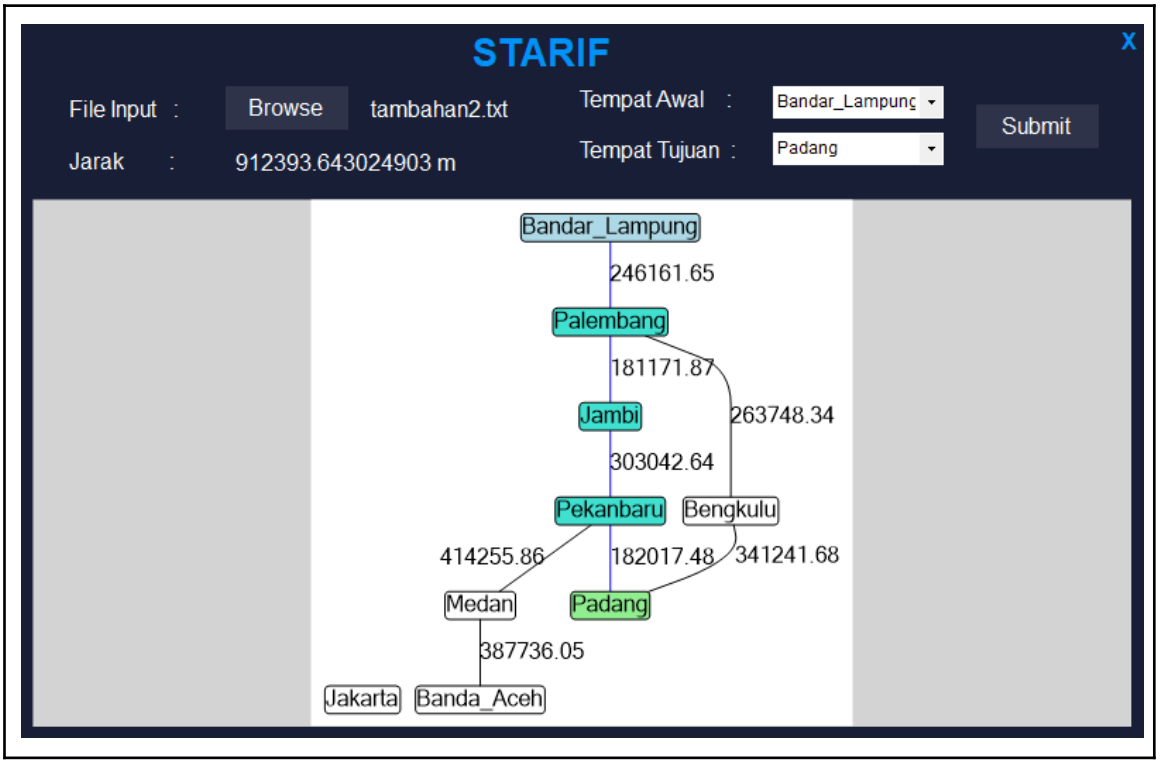
4. jambi.txt (JembatanAurduri → SmpgRawasari)



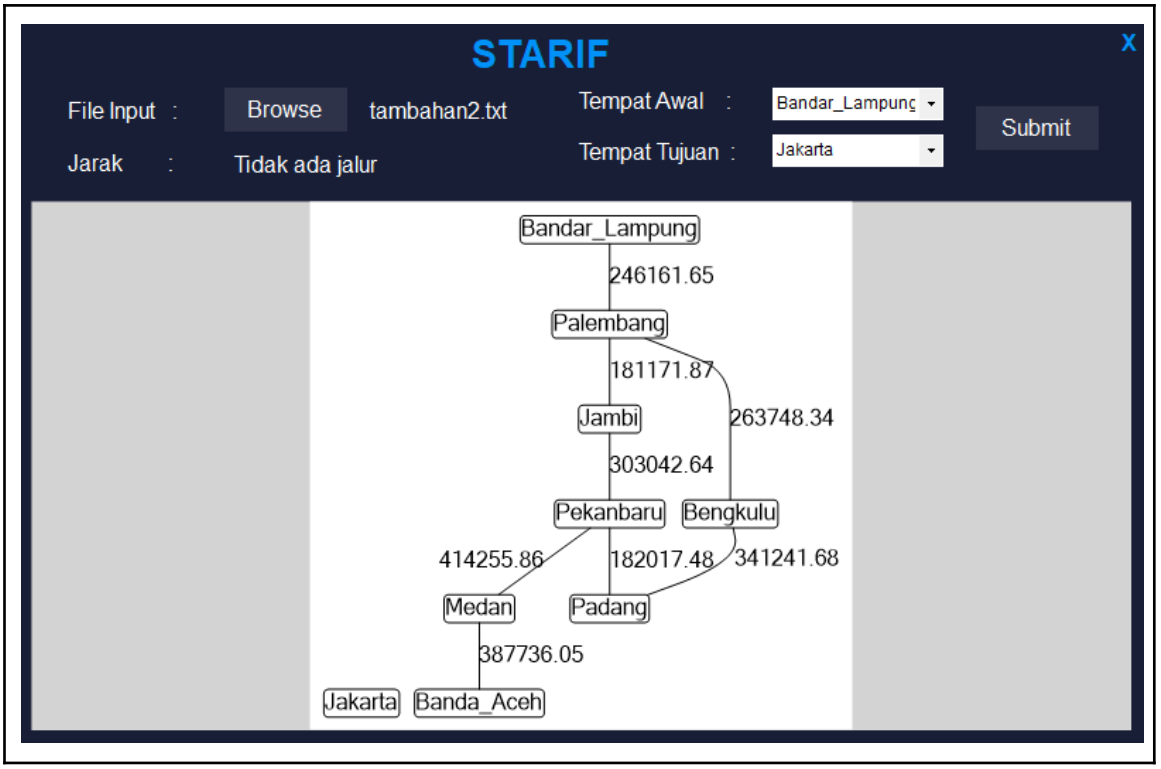
5. tambahan1.txt (Jakarta → Yogyakarta)



6. tambahan2.txt (Bandar_Lampung → Padang)



Kasus tidak ada jalur (Bandar_Lampung → Jakarta):



E. Alamat Kode Sumber Program

Alamat Alternatif 1:

https://github.com/epata/TucilSTIMA3

Alamat Alternatif 2:

https://drive.google.com/drive/folders/1m0M6C7KpRvwxQTbOOLib5T39oPPw3tCp?usp=sharing

Lampiran

1.	Program dapat menerima input graf	✓
2.	Program dapat menghitung lintasan terpendek	✓
3.	Program dapat menampilkan lintasan terpendek serta jaraknya	✓
4.	Bonus: Program dapat menerima input peta dengan Google Map API dan menampilkan peta	-