# TUGAS KECIL 3 IF2211 STRATEGI ALGORITMA

# SEMESTER II TAHUN 2020/2021

# Implementasi Algoritma A* untuk Menentukan Lintasan Terpendek

Oleh

Arsa Daris Gintara - 13519037

Muhammad Fawwaz Naabigh - 13519206

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2020/2021**

**Kode Program**

Class graph

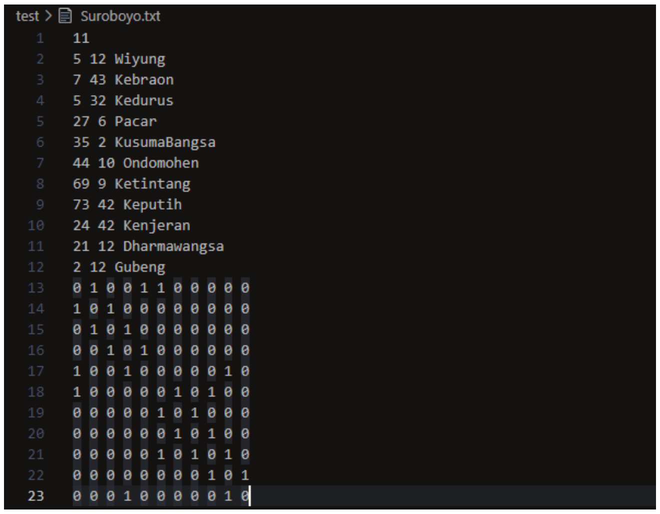
```
public class Graph
    {
        private Dictionary<string, string[]> adjacency;
        // [from, to] as key, distace as value
        private Dictionary<Tuple<string, string>, double> relationsDictionary;
        // nodes with its coordinate
        private Dictionary<string, Double[]> nodes;

        public Graph(List<string[]> relations, Dictionary<string, string[]>
adjacency, Dictionary<string, Double[]> nodes)
        {
            this.adjacency = adjacency;
            this.nodes = nodes;
            this.relationsDictionary = new Dictionary<Tuple<string, string>,
double>();
            foreach (var rel in relations)
            {
                var k1 = new Tuple<string, string>(rel[0], rel[1]);
                var k2 = new Tuple<string, string>(rel[1], rel[0]);
                if (!relationsDictionary.ContainsKey(k1))
this.relationsDictionary.Add(k1, Convert.ToDouble(rel[2]));
                if (!relationsDictionary.ContainsKey(k2))
this.relationsDictionary.Add(k2, Convert.ToDouble(rel[2]));
            }
        }
        public double G(String start, String n)
        {
            String[] from = { "0", start };
            double cost = 0;
            List<List<String>> simpulHidup = new List<List<String>>();
            List<string> dikunjungi = new List<string>();
            double distanceAtoN = 0;

            while (from[from.Count() - 1] != n && (simpulHidup.Any() ||
!dikunjungi.Any()))
            {
                dikunjungi.Add(from[from.Count() - 1]);

                String[] children = adjacency[from[from.Count() - 1]];
                foreach (var child in children)
                {
                    if (!dikunjungi.Contains(child))
                    {
                        var jalur = new Tuple<string, string>(child,
from[from.Count() - 1]);
                        String distance =
Convert.ToString(relationsDictionary[jalur] + distanceAtoN);
                        simpulHidup.Add(from.ToList<string>());
                        simpulHidup[simpulHidup.Count - 1][0] = distance;
                        simpulHidup[simpulHidup.Count - 1].Add(child);
                    }
                }
                var simpulMati = simpulHidup[0];
                double minDistance = 0;
                from = simpulMati.ToArray();
                distanceAtoN = Convert.ToDouble(simpulMati[0]);

                foreach (var node in simpulHidup)
                {
                    double dist = Convert.ToDouble(node[0]);
                    if (dist < minDistance || minDistance == 0)
                    {
                        minDistance = dist;
                        distanceAtoN = dist;
                        from = node.ToArray();
                        simpulMati = node;
                    }
                }
                simpulHidup.Remove(simpulMati);
            }

            cost = Convert.ToDouble(from[0]);
            return cost;
        }
        public double H(String n, string goal)
        {
            return euclideanDist(nodes[n][0], nodes[n][1], nodes[goal][0],
nodes[goal][1]);
        }
        public double F(String start, String n, String goal)
        {
            return G(start, n) + H(n, goal);
        }
        public List<string> Astar(string start, string goal)
        {
            List<string> dikunjungi = new List<string>();
            String[] from = { "0", start };
            List<List<String>> simpulHidup = new List<List<String>>();


            while (from[from.Count() - 1] != goal && (simpulHidup.Any() ||
!dikunjungi.Any()))
            {
                dikunjungi.Add(from[from.Count() - 1]);
                String[] children = adjacency[from[from.Count() - 1]];
                foreach (var child in children)
                {
                    if (!dikunjungi.Contains(child))
                    {
                        String bobot = Convert.ToString(F(start, child,
goal));
                        simpulHidup.Add(from.ToList<string>());
                        simpulHidup[simpulHidup.Count - 1][0] = bobot;
                        simpulHidup[simpulHidup.Count - 1].Add(child);
                    }
                }
                double bobotMin = Convert.ToDouble(simpulHidup[0][0]);
                var simpulMati = simpulHidup[0];
                from = simpulMati.ToArray();

                foreach (var node in simpulHidup)
                {
                    double dist = Convert.ToDouble(node[0]);
                    if (dist < bobotMin)
                    {
                        bobotMin = dist;
                        from = node.ToArray();
                        simpulMati = node;
                    }
                }

                simpulHidup.Remove(simpulMati);
            }
            if (from[from.Count() - 1] != goal) {
                String[] toReturn = { "0", start };
                return toReturn.ToList<string>();
            }
            return from.ToList<string>();
        }
        public double euclideanDist(Double x1, Double y1, Double x2, Double
y2) {
            return Math.Sqrt(Math.Pow(x1-x2,2)+Math.Pow(y1-y2,2));
        }
    }
```

Form1.cs

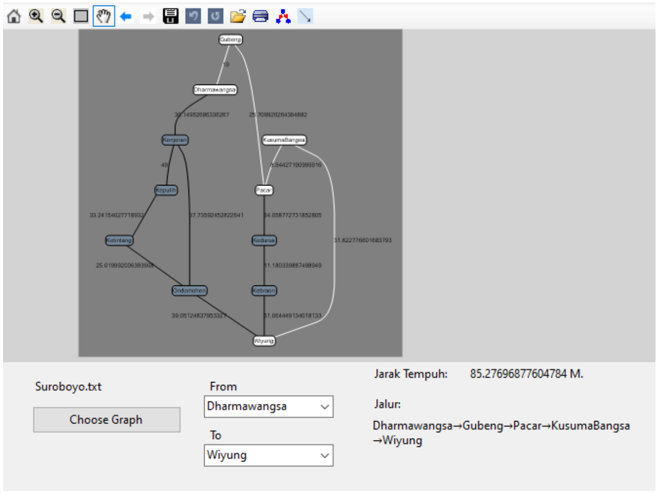
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using System.Diagnostics;


namespace Astar
{
    public partial class Form1 : Form
    {
        // num of relation
        int nRelations = 0;

        // contains all relations and its distance [from, to, distance]
        List<string[]> relations = new List<string[]>();

        // contains all unique nodes
        List<string> nodes = new List<string>();

        // nodes dictionary contains node's name as key, and its coordinat
        Dictionary<string, Double[]> nodesDictionary = new Dictionary<string,
Double[]>();

        // adjacency dict, contains parent as key, children as values
        Dictionary<string, string[]> adjacency = new Dictionary<string,
string[]>();

        // open file
        OpenFileDialog openFile = new OpenFileDialog();

        //visualizer
        Visualizer v = new Visualizer();

        Graph g;

        public Form1()
        {
            InitializeComponent();
            v.Initialize(graphVis); // initialize graph
        }

        private void chooseGraph_Click(object sender, EventArgs e)
        {
            if (openFile.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    // open .txt file
                    StreamReader sr = new StreamReader(openFile.FileName);
                    int lineNum = 0;
                    string line = "";

                    if (v.Viewer.Graph != null)
                    {
                        v.ClearScreen(nodes);
                    }

                    nodes.Clear();
                    adjacency.Clear();
                    nodesDictionary.Clear();
                    relations.Clear();
                    fromdropdown.Items.Clear();

                    while (line != null)
                    {
                        // read every line
                        line = sr.ReadLine();

                        // skip 1st line (num of relation)
                        if (line != null)
                        {
                            if (lineNum == 0)
                            {
                                nRelations = Convert.ToInt32(line);

                                // Debug.WriteLine("Banyak relasi {0}",
nRelations);
                            }
                            else if(lineNum <= nRelations)
                            {
                                // array of splitted line
                                String[] splitLine = new String[3];
                                // split every line read
                                splitLine = line.Split(' ');
                                Double[] koordinat = {
Convert.ToDouble(splitLine[0]), Convert.ToDouble(splitLine[1]) };

                                nodesDictionary.Add(splitLine[2], koordinat);
                                nodes.Add(splitLine[2]);

                                // Debug.WriteLine("Node ke {0}: {1}",
lineNum, nodes[lineNum-1]);
                                // Debug.WriteLine("Node Directory key {0},
koordinat({1}, {2})", nodesDictionary.Keys.ElementAt(lineNum - 1),
nodesDictionary.Values.ElementAt(lineNum-1)[0],
nodesDictionary.Values.ElementAt(lineNum - 1)[1]);

                            }
                            else    // adjacency
                            {
                                String[] splitLine = new String[nRelations];
                                splitLine = line.Split(' ');

                                int idxFrom = lineNum - 1 - nRelations;
                                // Debug.WriteLine("line ke {0}",
lineNum-nRelations);

                                List<String> child = new List<String>();

                                for (int i = 0; i<nRelations; i++)
                                {
                                    // Debug.WriteLine(i);
                                    int currentValue =
Convert.ToInt32(splitLine[i]);

                                    if ((i<= lineNum - nRelations - 1) &&
currentValue == 1)
                                    {
                                        int idxTo = i;

                                        String[] fromTo = { nodes[idxFrom],
nodes[idxTo], "" };

                                        double absis =
Convert.ToDouble(nodesDictionary[fromTo[0]][0]) -
Convert.ToDouble(nodesDictionary[fromTo[1]][0]);
                                        double ordinat =
Convert.ToDouble(nodesDictionary[fromTo[0]][1]) -
Convert.ToDouble(nodesDictionary[fromTo[1]][1]);

                                        double distance =
Math.Sqrt(Math.Pow(absis, 2.00) + Math.Pow(ordinat, 2.00));

                                        fromTo[2] =
Convert.ToString(distance);

                                        // Debug.WriteLine("idx from {0}",
idxFrom);
                                        // Debug.WriteLine("idx to {0}",
idxTo);

                                        relations.Add(fromTo);
                                        // Debug.WriteLine("Relation from {0}
to {1}. jaraknya {2}", relations[relations.Count-1][0],
relations[relations.Count - 1][1], relations[relations.Count - 1][2]);
                                    }

                                    if (currentValue == 1)
                                    {
                                        child.Add(nodes[i]);
                                    }
                                }
                                adjacency[nodes[idxFrom]] = child.ToArray();


                            }
                        }
                        lineNum++;
                    }

                    nodes.Sort();
                    foreach (var node in nodes)
           {
                        fromdropdown.Items.Add(node);
                    }

                    // display filename
                    label3.Text = Path.GetFileName(openFile.FileName);

                    g = new Graph(relations, adjacency, nodesDictionary);

                    v.Initialize(graphVis);

                    v.Start(nodes, relations);

                }
                catch (Exception error)
                {
                    MessageBox.Show(error.Message, "Failed to Open File",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                }

            }
            // failed to open file
            else
            {
                MessageBox.Show("Choose .txt File", "Failed to Open File",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // to filter only .txt file
            openFile.Filter = "Text Files (.txt) | *.txt";
        }

        private void fromdropdown_SelectedIndexChanged(object sender,
EventArgs e)
        {
            // clear "To dropdown
            todropdown.Items.Clear();
            // chosen point "from" dropdown
            String from = fromdropdown.Text;

            // add unselected account to "Explore Friends With" dropdown
            foreach (var item in fromdropdown.Items)
            {
                if (item.ToString() == from) continue;
                todropdown.Items.Add(item);
            }
        }

        private void todropdown_SelectedIndexChanged(object sender, EventArgs
e)
        {
            string start = fromdropdown.SelectedItem.ToString();
            string goal = todropdown.SelectedItem.ToString();
            List<string> path = g.Astar(start, goal);
            v.VisualizePath(nodes, relations, path);
            jaraktempuh.Text = path[0] + " M.";
            String teks = String.Join("→", path.Skip(1));
            richTextBox1.Text = teks;
            using (Graphics g = CreateGraphics())
            {
                richTextBox1.Height = (int)g.MeasureString(richTextBox1.Text, 
richTextBox1.Font, richTextBox1.Width).Height + 5;
            }
        }
    }
}
```

## Graf Input dan Screenshot Jalur

## 1.


test > Suroboyo.txt

```
11
5 12 Wiyung
7 43 Kebraon
5 32 Kedurus
27 6 Pacar
35 2 KusumaBangsa
44 10 Ondomohen
69 9 Ketintang
73 42 Keputih
24 42 Kenjeran
21 12 Dharmawangsa
2 12 Gubeng
0 1 0 0 1 1 0 0 0 0 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 1 0 1 0 0 0 0 0 0
1 0 0 1 0 0 0 0 0 1 0
1 0 0 0 0 0 1 0 1 0 0
0 0 0 0 0 1 0 1 0 0 0
0 0 0 0 0 0 1 0 1 0 0
0 0 0 0 0 1 0 1 0 1 0
0 0 0 0 0 0 0 0 1 0 1
0 0 0 1 0 0 0 0 0 1 0
```



Suroboyo.txt

From: Dharmawangsa

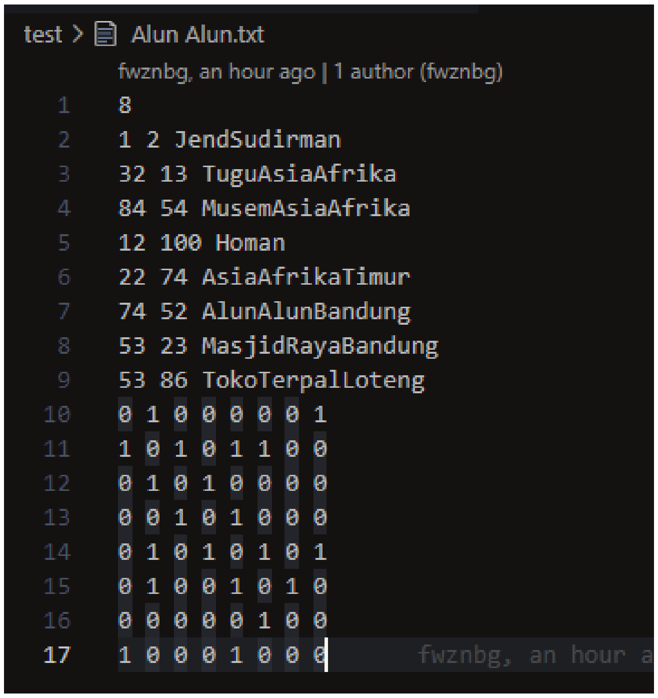
To: Wiyung

Jarak Tempuh: 85.27696877604784 M.

Jalur:

Dharmawangsa→Gubeng→Pacar→KusumaBangsa→Wiyung

**2.**

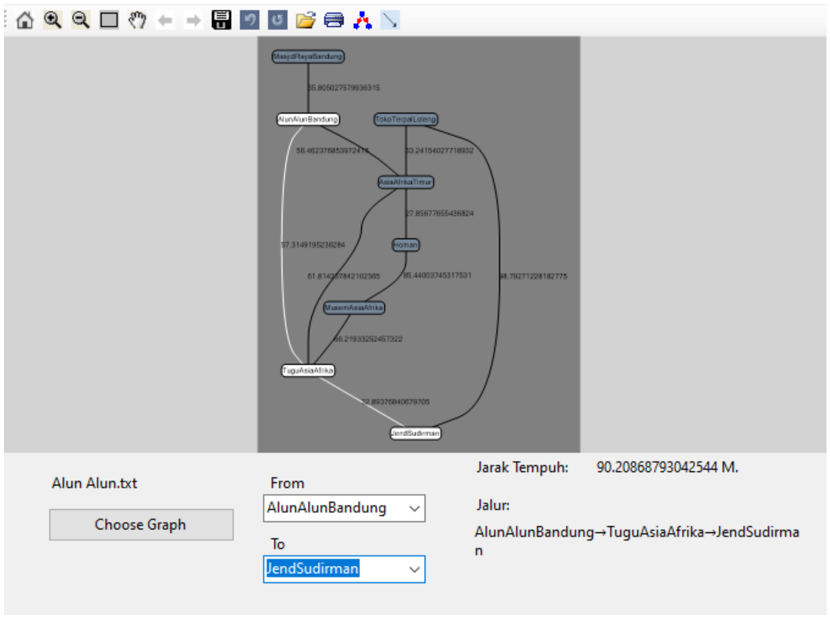


test > Alun Alun.txt

fwznbg, an hour ago | 1 author (fwznbg)

```
8
1 2 JendSudirman
32 13 TuguAsiaAfrika
84 54 MusemAsiaAfrika
12 100 Homan
22 74 AsiaAfrikaTimur
74 52 AlunAlunBandung
53 23 MasjidRayaBandung
53 86 TokoTerpalLoteng
0 1 0 0 0 0 0 1
1 0 1 0 1 1 0 0
0 1 0 1 0 0 0 0
0 0 1 0 1 0 0 0
0 1 0 1 0 1 0 1
0 1 0 0 1 0 1 0
0 0 0 0 0 1 0 0
1 0 0 0 1 0 0 0
```

fwznbg, an hour a



Alun Alun.txt

Choose Graph

From

AlunAlunBandung

To

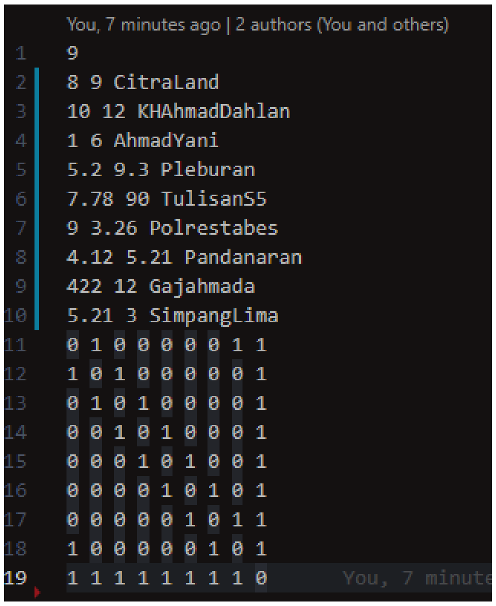
JendSudirman

Jarak Tempuh: 90.20868793042544 M.

Jalur:

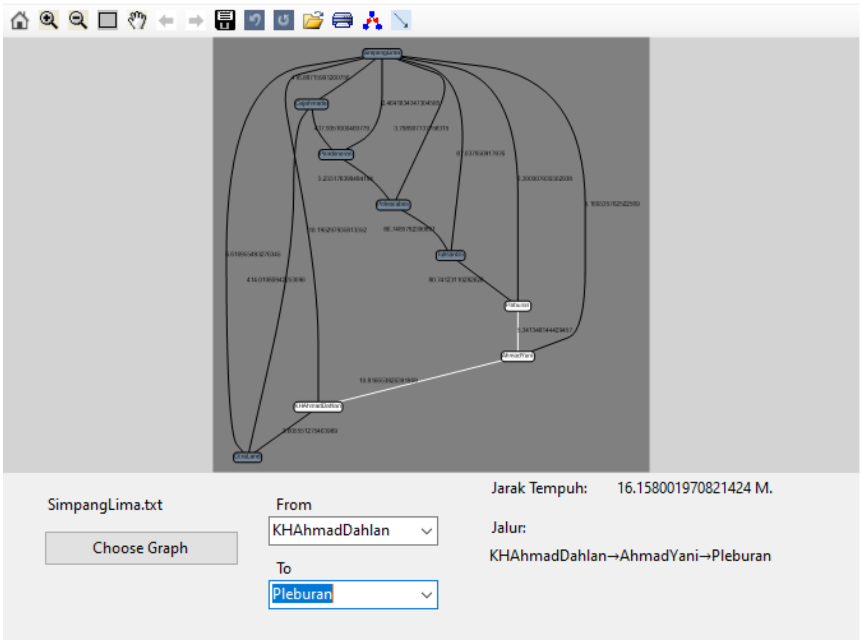
AlunAlunBandung→TuguAsiaAfrika→JendSudirma
n

**3.**



You, 7 minutes ago | 2 authors (You and others)

```
9
8 9 CitraLand
10 12 KHAhmadDahlan
1 6 AhmadYani
5.2 9.3 Pleburan
7.78 90 TulisanS5
9 3.26 Polrestabes
4.12 5.21 Pandanaran
422 12 Gajahmada
5.21 3 SimpangLima
0 1 0 0 0 0 0 1 1
1 0 1 0 0 0 0 0 1
0 1 0 1 0 0 0 0 1
0 0 1 0 1 0 0 0 1
0 0 0 1 0 1 0 0 1
0 0 0 0 1 0 1 0 1
0 0 0 0 0 1 0 1 1
1 0 0 0 0 0 1 0 1
1 1 1 1 1 1 1 1 0
```



SimpangLima.txt

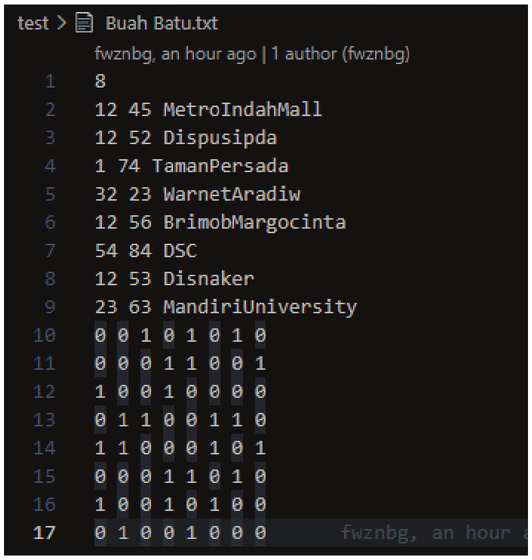
From: KHAhmadDahlan

To: Pleburan

Jarak Tempuh: 16.158001970821424 M.

Jalur:
KHAhmadDahlan→AhmadYani→Pleburan

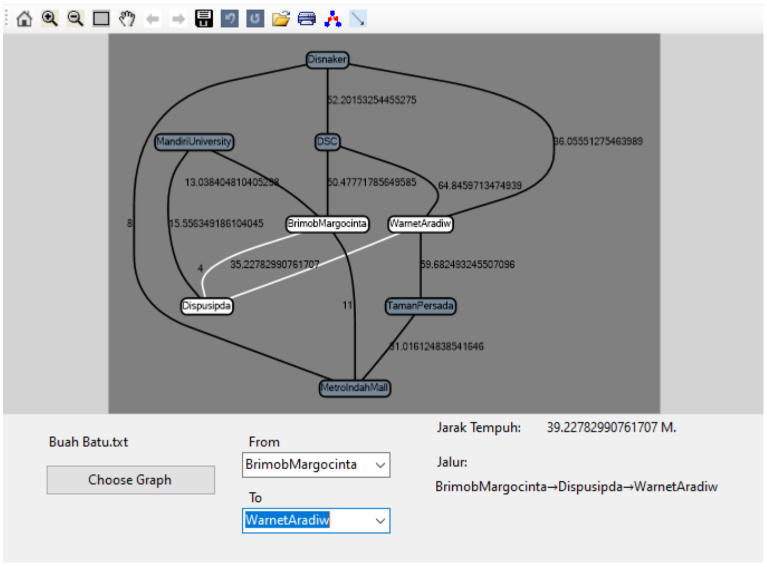
**4.**



test > Buah Batu.txt

fwznbg, an hour ago | 1 author (fwznbg)

```
8
12 45 MetroIndahMall
12 52 Dispusipda
1 74 TamanPersada
32 23 WarnetAradiw
12 56 BrimobMargocinta
54 84 DSC
12 53 Disnaker
23 63 MandiriUniversity
0 0 1 0 1 0 1 0
0 0 0 1 1 0 0 1
1 0 0 1 0 0 0 0
0 1 1 0 0 1 1 0
1 1 0 0 0 1 0 1
0 0 0 1 1 0 1 0
1 0 0 1 0 1 0 0
0 1 0 0 1 0 0 0        fwznbg, an hour a
```



Buah Batu.txt

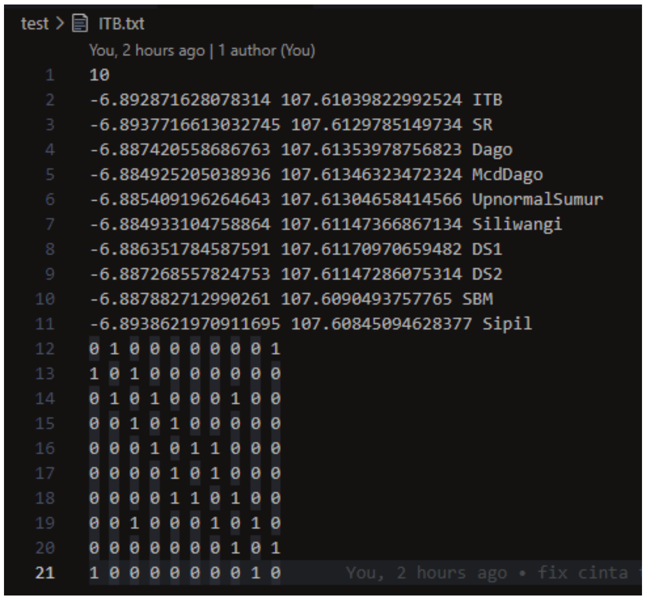
From: BrimobMargocinta

To: WarnetAradiw

Jarak Tempuh: 39.22782990761707 M.

Jalur:
BrimobMargocinta→Dispusipda→WarnetAradiw

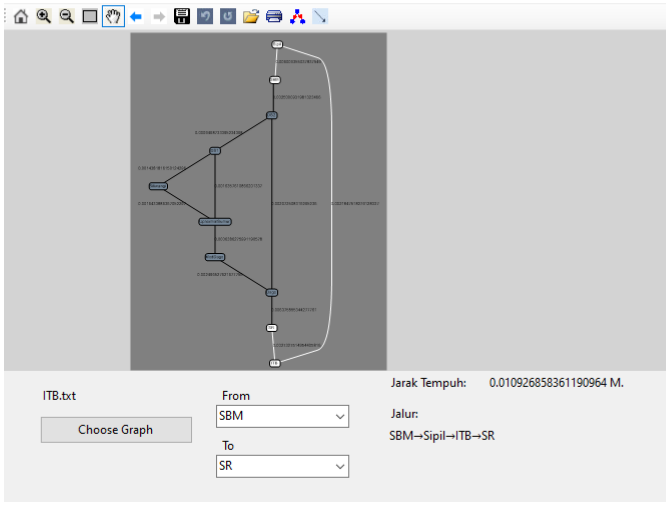
**5.**



test > ITB.txt

You, 2 hours ago | 1 author (You)

```
10
-6.892871628078314 107.61039822992524 ITB
-6.8937716613032745 107.6129785149734 SR
-6.887420558686763 107.61353978756823 Dago
-6.884925205038936 107.61346323472324 McdDago
-6.885409196264643 107.61304658414566 UpnormalSumur
-6.884933104758864 107.61147366867134 Siliwangi
-6.886351784587591 107.61170970659482 DS1
-6.887268557824753 107.61147286075314 DS2
-6.887882712990261 107.6090493757765 SBM
-6.8938621970911695 107.60845094628377 Sipil
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 1 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 1 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        You, 2 hours ago • fix cinta
```



ITB.txt

Choose Graph

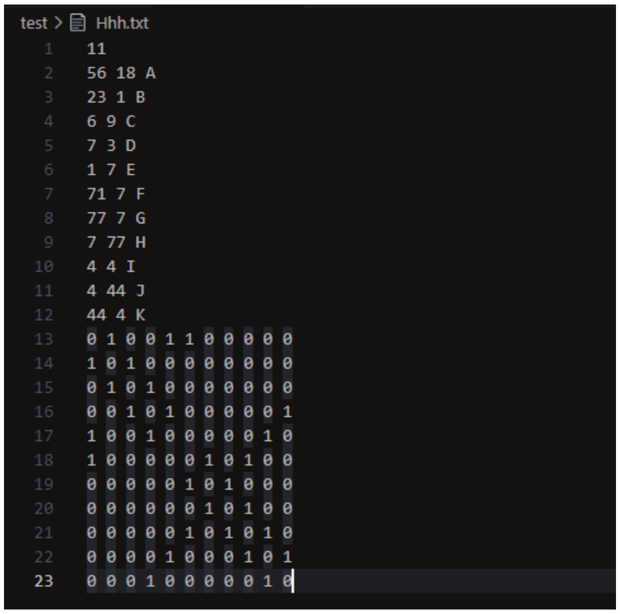
From: SBM

To: SR

Jarak Tempuh: 0.010926858361190964 M.

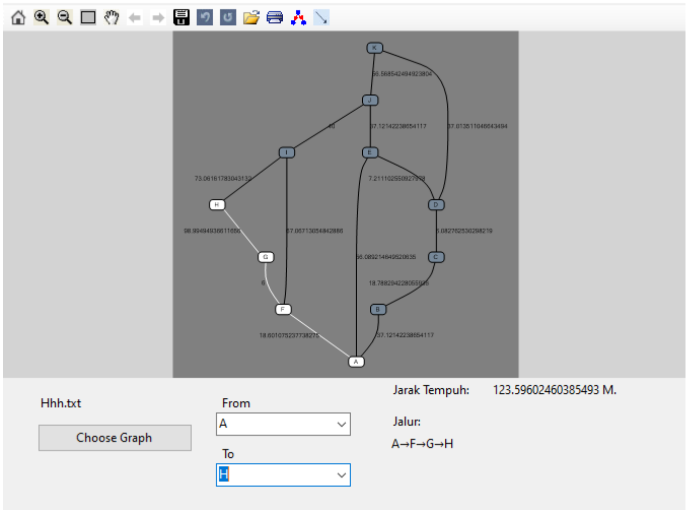
Jalur:
SBM→Sipil→ITB→SR

**6.**



test > Hhh.txt

```
11
56 18 A
23 1 B
6 9 C
7 3 D
1 7 E
71 7 F
77 7 G
7 77 H
4 4 I
4 44 J
44 4 K
0 1 0 0 1 1 0 0 0 0 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 1 0 1 0 0 0 0 0 1
1 0 0 1 0 0 0 0 0 1 0
1 0 0 0 0 0 1 0 1 0 0
0 0 0 0 0 1 0 1 0 0 0
0 0 0 0 0 0 1 0 1 0 0
0 0 0 0 0 1 0 1 0 1 0
0 0 0 0 1 0 0 0 1 0 1
0 0 0 1 0 0 0 0 0 1 0
```



Hhh.txt

Choose Graph

From
A

To
H

Jarak Tempuh: 123.59602460385493 M.

Jalur:
A→F→G→H

**Link Github**

https://github.com/fwznbg/ShortestPath

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