Tugas Kecil 3 IF2211 Strategi Algoritma

Implementasi Algoritma A* untuk Menentukan Lintasan Terpendek

Nama : Jusuf Junior Athala
NIM : 13519174

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2021

A. Langkah-langkah Implementasi Algoritma A* untuk Menentukan Lintasan Terpendek

Algoritma A* (atau A star) dapat digunakan untuk menentukan lintasan terpendek dari suatu titik ke titik lain. Pada tugas kecil 3 ini, anda diminta menentukan lintasan terpendek berdasarakan peta Google Map jalan-jalan di kota Bandung. Dari ruas-ruas jalan di peta dibentuk graf. Simpul menyatakan persilangan jalan atau ujung jalan. Asumsikan jalan dapat dilalui dari dua arah. Bobot graf menyatakan jarak (m atau km) antar simpul. Jarak antar dua simpul dapat dihitung dari koordinat kedua simpul menggunakan rumus jarak Euclidean (berdasarkan koordinat) atau dapat menggunakan ruler di Google Map, atau cara lainnya yang disediakan oleh Google Map.

Langkah pertama di dalam program ini adalah membuat graf yang merepresentasikan peta (di area tertentu, misalnya di sekitar kampus ITB). Sisi diperoleh dari jalan antar dua simpul dan bobot sisi adalah jarak Euclidean. Berdasarkan graf yang dibentuk, lalu program A* menerima input simpul asal dan simpul tujuan, lalu menentukan lintasan terpendek antara keduanya. Lintasan terpendek dapat ditampilkan pada peta/graf. Nilai heuristik yang dipakai adalah jarak garis lurus dari suatu titik ke tujuan.

Spesifikasi program:

1. Program menerima input file graf (direpresentasikan sebagai matriks ketetanggaan berbobot), jumlah simpul minimal 8 buah.
2. Program dapat menampilkan peta/graf
3. Program menerima input simpul asal dan simpul tujuan.
4. Program dapat menampilkan lintasan terpendek beserta jaraknya antara simpul asal dan simpul tujuan

Berikut adalah langkah-langkah program Implementasi Algoritma A* untuk Menentukan Lintasan Terpendek :

1. Pisahkan matriks ketetanggan dan koordinat simpul dari text file.
2. Buat list simpul yang terdapat koordinat dan list nama untuk nama-nama yang muncul .
3. Buat graf berdasarkan matriks ketetanggaan.
4. Buat sebuah list Tuple <Stack<int>, double> dengan Stack adalah stack simpul yang diperiksa dan double adalah f(n) dari simpul yang sedang diperiksa.
5. Jika simpul yang diperiksa belum pernah dikunjungi, ubah array visited menjadi true jika simpul telah dikunjungi. Kemudian, simpul yang diperiksa akan dimasukkan ke stack. f(n) akan dihitung berdasarkan bobot / weight dari simpul yang diperiksa ditambah dengan jarak heuristik terhadap simpul tujuan berdasarkan jarak euclidean simpul yang diperiksa dengan simpul tujuan.
6. Stack-stack yang telah dihidupkan akan dimasukkan kedalam list Tuple.
7. List Tuple akan di Sort sehingga Tuple awal adalah Tuple dengan f(n) paling kecil.
8. Jika simpul tujuan belum ditemukan, akan diperiksa Tuple paling awal dan nilai f(n) Tuple tersebut akan dikurangi jarak heuristik terhadap simpul tujuan.
9. Lakukan langkah 5-8 sehingga simpul tujuan ditemukan.

B. Source Code Program

```
// Nama           : Jusuf Junior Athala
// NIM            : 13519174
// Mata kuliah    : IF2211 - Strategi Algoritma
// Kelas          : K-04
// Tugas          : Implementasi Algoritma A* untuk Menentukan Lintasan
//                  Terpendek
// Bahasa Program : C#
```

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

namespace Tucil3
{
    class Node
       {
              private int _v;
              private int _x;
              private int _y;
              public Node(int v, int x, int y)
              {
                     this._v = v;
                     this._x = x;
                     this._y = y;
              }

              public int V()
              {
                     return _v;
              }

              public int X()
              {
                     return _x;
              }

              public int Y()
              {
                     return _y;
              }

              public double EuclideanDistance(Node w)
              {
                     int x = this._x - w.X();
                     int y = this._y - w.Y();
                     return Math.Sqrt((x * x) + (y * y));
              }
       }
}
```

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

```
namespace Tucil3
{
    class Edge
       {
              private Node _v;
              private Node _w;
              private double _weight;
              public Edge(Node v, Node w)
              {
                     this._v = v;
                     this._w = w;
                     int x = v.X() - w.X();
                     int y = v.Y() - w.Y();
                     this._weight = Math.Sqrt((x * x) + (y * y));
              }

              public double Weight()
              {
                     return _weight;
              }

              public Node Source()
              {
                     return _v;
              }

              public Node Target(Node vertex)
              {
                     if (vertex == _v)
                            return _w;
                     else if (vertex == _w)
                            return _v;
                     else
                            throw new Exception("Error");
              }
       }
}
```

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

namespace Tucil3
{
    class WeightedGraph
       {
              private int _v;
              private int _e;
              private List<Edge>[] _adj;
              public WeightedGraph(int V)
              {
                     this._v = V;
                     this._e = 0;
                     this._adj = new List<Edge>[V];
                     for (int i = 0; i < this._adj.Length; i++)
                            this._adj[i] = new List<Edge>();
              }
```

```
            public int V()
            {
                  return _v;
            }

            public int E()
            {
                  return _e;
            }

            public void AddEdge(Edge e)
            {
                  Node v = e.Source();
                  Node w = e.Target(v);
                  this._adj[v.V()].Add(e);
                  this._adj[w.V()].Add(e);
                  this._e++;
            }

            public List<Edge> getAdjacency(int v)
            {
                  return this._adj[v];
            }
      }
}
```

```
using Microsoft.Msagl.Drawing;
using System;
using System.Collections.Generic;

using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;

using System.Windows.Forms;

namespace Tucil3
{
    public partial class Form1 : Form
    {

        private List<string> listNames;
        private List<Tuple<Stack<int>, double>> listTuples;
        private List<Node> listNodes;
        private WeightedGraph myGraph;
        //visited
        private bool[] visited;

        public Form1()
        {
            this.openFileDialog1 = new System.Windows.Forms.OpenFileDialog();
            OpenFileDialog openFileDialog1 = new OpenFileDialog();
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
```

```
        {

        }

        private void folderBrowserDialog2_HelpRequest(object sender, EventArgs
e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog1 = new OpenFileDialog
            {
                InitialDirectory = "./",
                Title = "Browse Text Files",

                CheckFileExists = true,
                CheckPathExists = true,

                DefaultExt = "txt",
                Filter = "txt files (*.txt)|*.txt",
                FilterIndex = 2,
                RestoreDirectory = true,

                ReadOnlyChecked = true,
                ShowReadOnly = true
            };

            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                textBox1.Text = openFileDialog1.FileName;
                FileName = textBox1.Text;
            }
        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {

        }

        private void viewer1_Load(object sender, EventArgs e)
        {

        }

        private void button2_Click(object sender, EventArgs e)
        {
            try
            {
                string text1 = File.ReadAllText(FileName);

                listNames = new List<string>();
                listTuples = new List<Tuple<Stack<int>, double>>();
                listNodes = new List<Node>();

                graph1 = new Microsoft.Msagl.Drawing.Graph("graph");
```

```
                char[] delimiterLine = { '\n' };
                char[] delimiterSlash = { '/' };
                char[] delimiterComma = { ',' };
                string[] delimiterDot = { "\n." };

                //pisahkan koordinat dan matrix
                string[] split = text1.Split(delimiterDot,
StringSplitOptions.RemoveEmptyEntries);

                string[] lines = split[0].Split(delimiterLine); //split setiap
newline

                for (int i = 1; i <= Int32.Parse(lines[0]); i++)
                {
                    string[] name = lines[i].Split(delimiterSlash);
                    string[] coord = name[1].Split(delimiterComma);
                    if (!listNames.Contains(name[0]))
                    {
                        //Menambahkan simpul ke list nama dan list simpul
                        listNames.Add(name[0]);
                        Node n = new Node(listNames.IndexOf(name[0]),
Int32.Parse(coord[0]), Int32.Parse(coord[1]));
                        listNodes.Add(n);
                    }
                }

                string[] matrix = split[1].Split(delimiterLine);
                myGraph = new WeightedGraph(Int32.Parse(lines[0]));

                //Menambahkan ketetanggaan simpul dan menggambar visualisasi
Graf
                for (int i = 1; i <= Int32.Parse(lines[0]); i++)
                {

                    string copy = matrix[i];
                    for (int j = i; j <= Int32.Parse(lines[0]); j++)
                    {
                        if (copy[j].Equals('1'))
                        {
                            Edge ed = new Edge(listNodes[i - 1], listNodes[j -
1]);
                            myGraph.AddEdge(ed);

                            string node1;
                            string node2;
                            node1 = matrix[0].Substring(i, 1);
                            node2 = matrix[0].Substring(j, 1);
                            var Edge = graph1.AddEdge(node1, node2);
                            Edge.Attr.ArrowheadAtTarget = ArrowStyle.None;
                            Edge.Attr.ArrowheadAtSource = ArrowStyle.None;

                            graph1.FindNode(node1).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.MistyRose;
                            graph1.FindNode(node2).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.MistyRose;
```

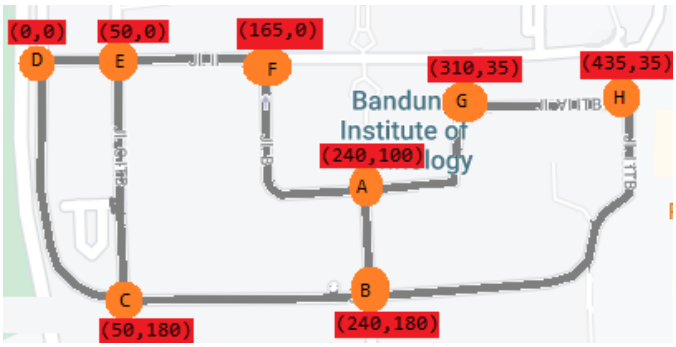
```
                        }
                    }
                }



                gViewer1.Graph = graph1;
                comboBox1.Items.Clear();
                comboBox2.Items.Clear();
                comboBox1.Items.AddRange(listNames.ToArray());
                comboBox2.Items.AddRange(listNames.ToArray());
                comboBox1.Enabled = true;
                comboBox2.Enabled = true;
            }
            catch
            {
                MessageBox.Show("Please Input an acceptable file", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }

        private void textBox3_TextChanged(object sender, EventArgs e)
        {

        }

        private void checkedListBox1_SelectedIndexChanged(object sender,
EventArgs e)
        {

        }

        private void textBox4_TextChanged(object sender, EventArgs e)
        {

        }

        private void button3_Click(object sender, EventArgs e)
        {
            try
            {
                if (comboBox1.Items.Count == 0)
                {
                    MessageBox.Show("Please click visualize first", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
                else if (comboBox1.Text.Equals("") ||
comboBox2.Text.Equals(""))
                {
                    MessageBox.Show("Please select node first", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
```

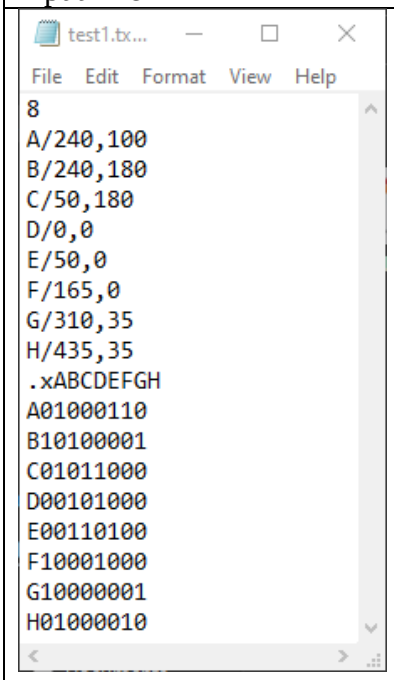
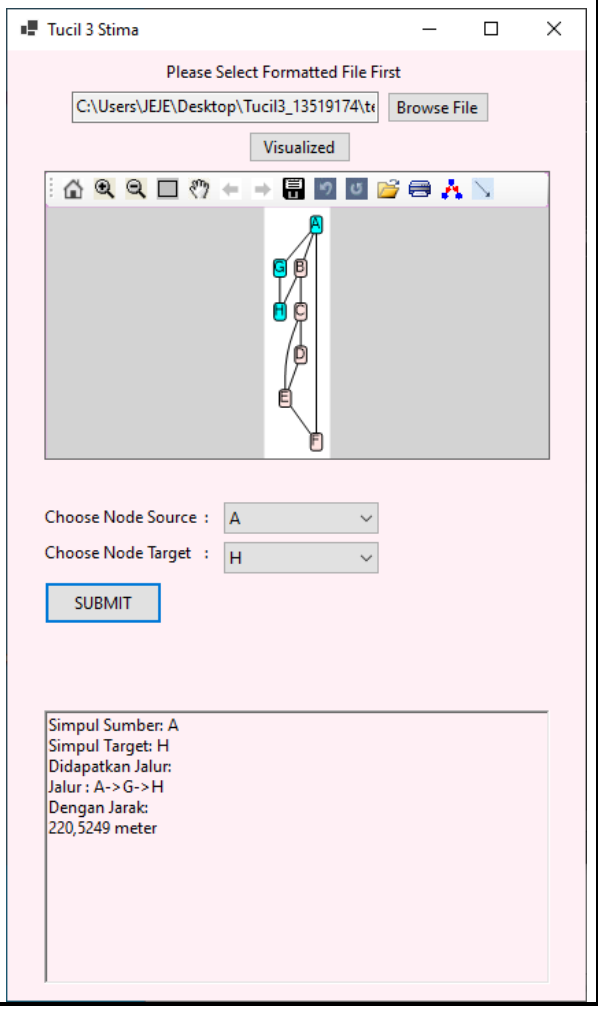
```
                    return;
                }
                else if (comboBox1.Text.Equals(comboBox2.Text))
                {
                    MessageBox.Show("Cannot select same node", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
                Size newSize;
                newSize = new Size(Size.Width, 733);
                richTextBox1.BorderStyle = BorderStyle.FixedSingle;
                this.Size = newSize;
                Point newLoc = new Point(0, 0);
                this.Location = newLoc;
                string richboxtext1 = "Simpul Sumber: " +
comboBox1.SelectedItem + "\n" + "Simpul Target: " + comboBox2.SelectedItem +
"\n";

                //Array visited
                visited = new bool[myGraph.V()];
                for (int i = 0; i < myGraph.V(); i++)
                {
                    visited[i] = false;
                }

                //isource adalah index simpul sumber dan ifind adalah index
simpul tujuan
                int isource = comboBox1.SelectedIndex;
                int ifind = comboBox2.SelectedIndex;

                //Push simpul sumber
                Stack<int> start = new Stack<int>();
                start.Push(isource);

                listTuples.Clear();
                listTuples.Add(Tuple.Create(start, Convert.ToDouble(0)));

                int v = listTuples[0].Item1.Peek();
                visited[v] = true;
                int visitedcount = 1;

                while (listTuples.Count > 0 && visitedcount < myGraph.V() && v
!= ifind)
                {
                    //jika bukan memeriksa simpul awal akan mengganti tuple
awal dengan
                    //tuple yang memiliki f(n) dikurangi jarak heuristik
                    if (listTuples[0].Item2 != 0)
                    {
                        Stack<int> temp1 = new
Stack<int>(listTuples[0].Item1.Reverse());
                        double temp2 = listTuples[0].Item2 -
listNodes[listTuples[0].Item1.Peek()].EuclideanDistance(listNodes[ifind]);
                        listTuples.Insert(1, Tuple.Create(temp1, temp2));
                        listTuples.RemoveAt(0);
                    }
```

```
                    //memeriksa simpul yang terhubung dengan simpul sumber
                    foreach (var edge in myGraph.getAdjacency(v))
                    {
                        if (!visited[edge.Target(listNodes[v]).V()])
                        {
                            visited[edge.Target(listNodes[v]).V()] = true;
                            visitedcount++;
                            Stack<int> s = new
Stack<int>(listTuples[0].Item1.Reverse());
                            s.Push(edge.Target(listNodes[v]).V());

                            double fn = 0;
                            double eucli =
edge.Target(listNodes[v]).EuclideanDistance(listNodes[ifind]);
                            fn = listTuples[0].Item2 + edge.Weight() + eucli;

                            listTuples.Add(Tuple.Create(s, fn));
                        }
                    }


                    //melakukan sort list tuple
                    listTuples.Sort((x, y) => x.Item2.CompareTo(y.Item2));
                    if (listTuples[0].Item1.Peek() != ifind)
                    {
                        listTuples.RemoveAt(0);
                    }

                    v = listTuples[0].Item1.Peek();

                }

                richboxtext1 = richboxtext1 + "Didapatkan Jalur: " + "\n";
                richboxtext1 = richboxtext1 + printStack(listTuples[0].Item1,
listNames);
                richboxtext1 = richboxtext1 + "Dengan Jarak: " + "\n";
                richboxtext1 = richboxtext1 + string.Format("{0:N4}",
listTuples[0].Item2) + " meter\n";

                Stack<int> stackPath = new Stack<int>(listTuples[0].Item1);

                RefreshGraphColor(listNames);

                //memberi warna pada simpul graf
                System.Drawing.Color colorname =
System.Drawing.Color.FromName("Cyan");
                graph1.FindNode(listNames.ElementAt(isource)).Attr.FillColor =
new Microsoft.Msagl.Drawing.Color(colorname.R, colorname.G, colorname.B);

                while (stackPath.Count > 0)
                {

graph1.FindNode(listNames.ElementAt(stackPath.Pop())).Attr.FillColor = new
Microsoft.Msagl.Drawing.Color(colorname.R, colorname.G, colorname.B);
                }
                //refresh graph
                gViewer1.Graph = graph1;
```

```
                richTextBox1.Text = richboxtext1;
            }
            catch
            {
                MessageBox.Show("Some error happened.", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

        }
        private void RefreshGraphColor(List<string> listNames)
        {
            for (int i = 0; i < listNames.Count; i++)
            {
                graph1.FindNode(listNames[i]).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.MistyRose;
            }
        }
        private static string printStack(Stack<int> stack, List<string>
listNames)
        {
            StringBuilder build = new StringBuilder();
            build.Append("Jalur : ");
            Stack<int> temp = new Stack<int>(stack);
            while (temp.Count > 0)
            {
                build.Append(listNames[temp.Pop()]);
                if (temp.Count != 0)
                {
                    build.Append("->");
                }
                else
                {
                    build.Append("\n");
                }
            }
            return build.ToString();
        }



        private void richTextBox2_TextChanged(object sender, EventArgs e)
        {

        }
    }
}
```

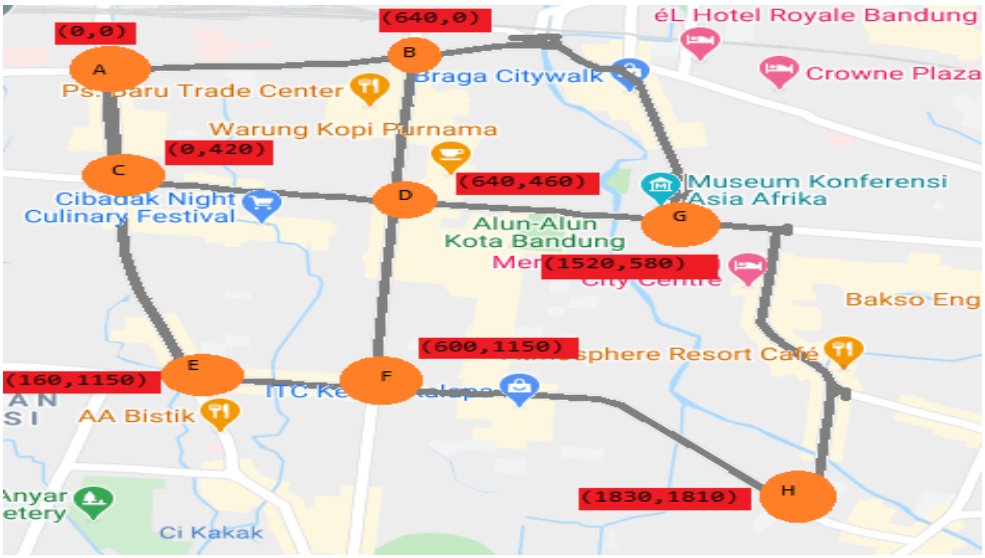
C. Contoh Eksekusi Program (Input dan Output)

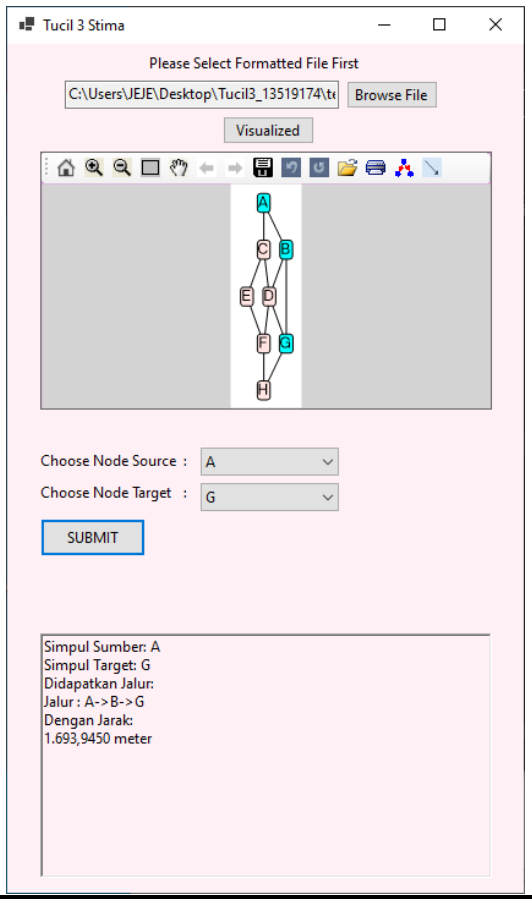
<table>
<tr><td>No</td><td>Peta</td></tr>
<tr><td>1</td><td>Kawasan ITB
 

<table>
<tr><td>Input File</td><td>Output</td></tr>
<tr><td> 
</td><td>Misal dari simpul A ke H
 
</td></tr>
</table></td></tr>
</table>

2

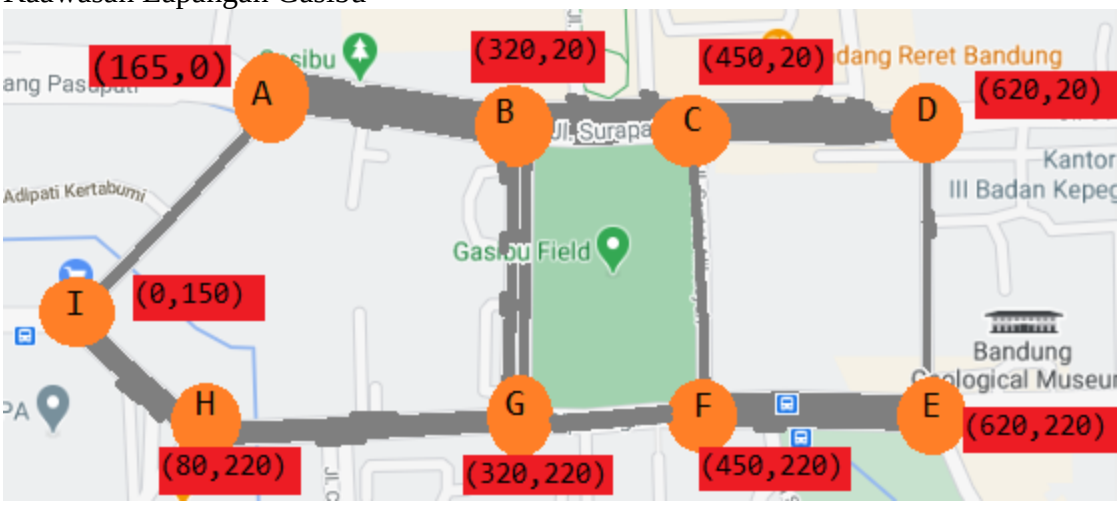
Kawasan Alun-Alun Bandung



Input File	Output
test2.txt -... File Edit Format View Help 8 A/0,0 B/640,0 C/0,420 D/640,460 E/160,1150 F/600,1150 G/1520,580 H/1830,1810 .xABCDEFGH A01100000 B10010010 C10011000 D01100110 E00100100 F00011001 G01010001 H00000110	Misal dari simpul A ke G 

3.

Kaawasan Lapangan Gasibu



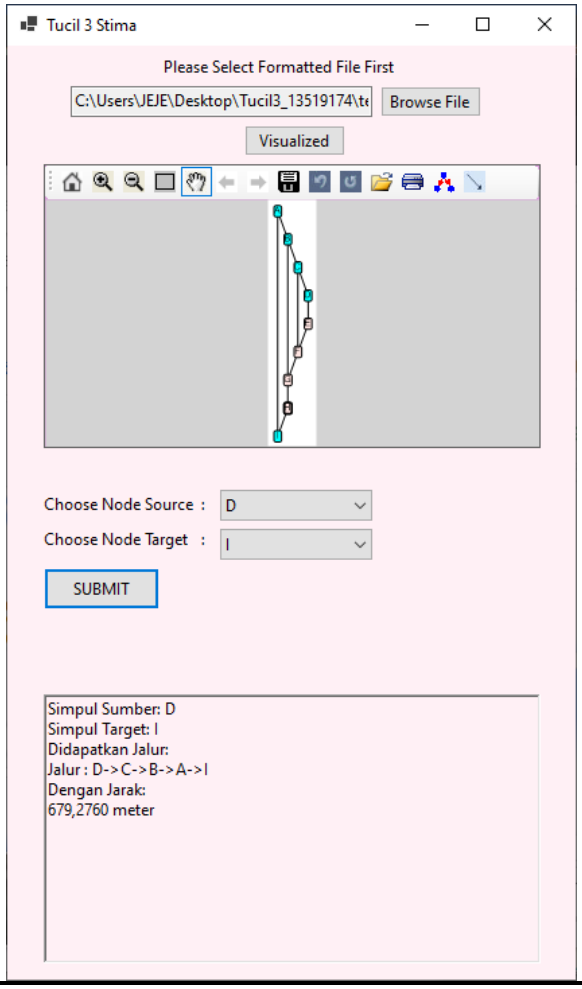
Input File	Output

test3.txt -...

File Edit Format View Help

```
9
A/165,0
B/320,20
C/450,20
D/620,20
E/620,220
F/450,220
G/320,220
H/80,220
I/0,150
.xABCDEFGHI
A010000001
B101000100
C010101000
D001010000
E000101000
F001010100
G010001010
H000000101
I100000010
```

Misal dari simpul D ke I



D. Checklist

Berikut adalah link untuk file program :
https://github.com/jusufjathala/Tucil3_13519174

Poin	Ya
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.	