IMPLEMENTASI ALGORITMA A*
UNTUK MENENTUKAN LINTASAN TERPENDEK

LAPORAN

Sebagai Bagian dari Tugas Kecil 3 mata kuliah Strategi Algoritma IF2211 pada semester II Tahun Akademik 2020/2021

Oleh

Isabella Handayani Sumantri 13519081

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2021

DAFTAR ISI

BAB I
TEORI DASAR

1.1. Algoritma A*

Algoritma A* termasuk algoritma *informed search* karena terdapat *heuristic* atau informasi tambahan yang memperkirakan *cost* hingga simpul tujuan. Algoritma ini dapat digunakan untuk menentukan lintasan terpendek dari suatu titik ke titik lain. Ide dari algoritma A* adalah menghindari melakukan *expand* pada node yang “mahal”. *Heuristic* pada algoritma A* akan melakukan estimasi nilai dari node

Heuristic Search

Adapun fungsi evaluasi *heuristic* pada algoritma A* dapat dinyatakan dengan

$$f(n) = g(n) + h(n)$$

Keterangan :
g(n) = *cost so far to reach n*
h(n) = *estimated cost from n to goal*
f(n) = *estimated total cost of path through n to goal*

Dalam penentuan rute terpendek, nilai *h(n)* yang digunakan berdasarkan *straight-line distance* dari *n* ke *goal*. Akan dilakukan perhitungan nilai *h(n)* menggunakan *haversine*.

BAB II

Kode Program

2.1. *Class* Graph

2.1.1. Atribut

```
private Microsoft.Msagl.Drawing.Graph MSAGLform;
private Dictionary<string, Point> nodeLst;
private Dictionary<string, List<string>> adjLst;
private Dictionary<(string, string), Microsoft.Msagl.Drawing.Edge> GUIEdge;
```

2.1.2. *Method*

2.1.2.1 *Method Graph*

```
//ctor
public Graph(String filepath)
{
    /* Kamus */
    /*
    file, nodeInfo, edgeInfo : array of string
    numNodes : int
    source, target : string
    edge : var
    */

    /* Algoritma */

    // init Attribute
    nodeLst = new Dictionary<string, Point>();
    MSAGLform = new Microsoft.Msagl.Drawing.Graph();
    adjLst = new Dictionary<string, List<string>>();
    GUIEdge = new Dictionary<(string, string),
Microsoft.Msagl.Drawing.Edge>();


    // Read file
    string [] file = System.IO.File.ReadAllLines(filepath);

    // get number of nodes
    int numNodes = int.Parse(file[0]);



    // saves all nodes and its point
    for(int i = 1; i < (numNodes + 1); i++)
    {
        string[] nodeInfo = file[i].Split(' ');
        nodeLst.Add (nodeInfo[2], new Point(Double.Parse(nodeInfo[0]),
Double.Parse(nodeInfo[1])));
        adjLst.Add(nodeInfo[2], new List<string>());
    }

    // Saves edge between node and create MSAGL Graph
    for(int j = nodeLst.Count + 1; j < file.Length; j++)
    {
        string[] edgeInfo = file[j].Split(' ');
        for(int k = 0; k < numNodes; k++)
        {
            if(edgeInfo[k].Equals("1"))
            {
                string source = nodeLst.ElementAt(j - numNodes -
1).Key;
```

```
                string target = nodeLst.ElementAt(k).Key;

                adjLst[source].Add(target);

                if(!adjLst[target].Contains(source))
                {
                    var edge = MSAGLform.AddEdge(source,
Math.Round(haversine(source, target), 5).ToString(), target);
                    GUIEdge.Add((source, target), edge);
                    edge.Attr.ArrowheadAtTarget =
Microsoft.Msagl.Drawing.ArrowStyle.None;
                }
            }
        }
    }
}
```

2.1.2.2. *Method Haversine*

```
// Distance by Haversine formula
public double haversine(string source, string target)
{
    /* KAMUS */
    /*
     *  lon, lat, a, d : double
     */

    /* ALGORITMA */

    // Haversine formula taken from
http://1.bp.blogspot.com/-eIVzIqcs_ik/U4xLXqpgBMI/AAAAAAAAQyw/vRrNAYU3
U2E/s1600/Haversine+method+bakhtyiar.png
    double lon = (Math.PI/180) * (nodeLst[target].getLon() -
nodeLst[source].getLon());
    double lat = (Math.PI/180) *(nodeLst[target].getLat() -
nodeLst[source].getLat());
    double a = Math.Pow(Math.Sin(lat / 2), 2) + Math.Cos((Math.PI /
180) * nodeLst[target].getLat()) * Math.Cos((Math.PI / 180) *
nodeLst[source].getLat()) * Math.Pow(Math.Sin(lon / 2), 2);
    double d = Math.Sqrt(a);
    return 2 * 6371 * Math.Asin(d);
}
```

2.1.2.3. *Method getMSAGLGraph*

```
public Microsoft.Msagl.Drawing.Graph getMSAGLGraph()
{
    return MSAGLform;
}
```

2.1.2.4. *Method getNodeLst*

```
public Dictionary<string, Point> getNodeLst()
{
    return this.nodeLst;
}
```

2.1.2.5. *Method getGUIEdge*

```
public Dictionary<string, List<string>> getAdjLst()
{
    return this.adjLst;
```

```
}
```

2.1.2.6. *Method filterEdge*

```
// find edge between prev and next
public Microsoft.Msagl.Drawing.Edge filterEdge(string prev, string
next)
{
    /*KAMUS*/
    /*
        edge = Microsoft.MSAGL.Drawing.Edge

    */

    /* ALGORITMA */

    Microsoft.Msagl.Drawing.Edge edge;

    if (GUIEdge.ContainsKey((prev,  next)))
    {
        edge = GUIEdge[(prev, next)];
    }
    else
    {
        edge = GUIEdge[(next, prev)];
    }

    return edge;
}
```

2.2. A*

2.2.1. Atribut

```
private Graph graph;
private Dictionary<string, bool> expanded;
private Dictionary<string, double> visitedCost;
private List<string> path;
private bool found;
private Dictionary<string, string> parentMap;
```

2.2.2. Method

2.2.2.1. *Method AStar*

```
//Ctor
public AStar(Graph graph, string source, string target)
{
    /* KAMUS */
    /*
    end : bool
    currNode : string
    currNodeAdj : Dictionary<string, List<string>>
    */

    /* ALGORTIMA */

    // Init atribbute
    this.expanded = new Dictionary<string, bool>();
    this.visitedCost = new Dictionary<string, double>();
    this.graph = graph;
```

```
    this.path = new List<string>();
    this.parentMap = new Dictionary<string, string>();

    //Initialize expanded as false
    initVisit();


    bool end = false;
    string currNode = source;
    double cost;


    Dictionary<string, List<string>> currNodeAdj = graph.getAdjLst();

    this.visitedCost.Add(currNode, 0);
    this.parentMap.Add(currNode, source);

    // while target not found or there are no path found
    while (!end)
    {
        this.path.Add(currNode);
        // Check for all node adjacent with currNode
        foreach (var x in currNodeAdj[currNode])
        {
            // Add parent
            if (!parentMap.ContainsKey(x))
            {
                parentMap.Add(x, currNode);
            }

            // Check if it's target
            if (x.Equals(target))
            {
                this.found = true;
                end = true;
            }

            // Calculate Visited cost using haversine
            cost = getVisitedCost(source, parentMap[x]) +
graph.haversine(parentMap[x], x) + graph.haversine(x, target);
            if (!visitedCost.ContainsKey(x))
            {
                visitedCost.Add(x, cost);
            }
            else
            {
                visitedCost[x] = cost;
            }
        }

        // Check if found
        if (!end)
        {

            expanded[currNode] = true;
            //find lowest cost node to be expanded
            currNode = findLowestCost();

            // Check if there are no node left
            if (currNode == null)
            {
                end = true;
                found = false;
            }
```

```
        }
    }
    this.path = cleanPath(source);

}
```

2.2.2.2. *Method initVisit*

```
public void initVisit()
{
    /* Assign expanded value false for all node */

    expanded = new Dictionary<string, bool>();
    foreach (var x in this.graph.getNodeLst())
    {
        expanded.Add(x.Key, false);
    }
}
```

2.2.2.3. *Method findLowestCost*

```
public string findLowestCost()
{
    /* Find nodes with lowest cost */

    /* KAMUS */
    /*
    minNode : string
    min : double
    */

    string minNode = null ;
    double min = Double.MaxValue;
    foreach (var x in visitedCost)
    {
        if (expanded[x.Key] != true && x.Value < min)
        {
            minNode = x.Key;
            min = x.Value;
        }
    }
    return minNode;
}
```

2.2.2.4. *Method getStatus*

```
public bool getStatus()
{
    return this.found;
}
```

2.2.2.5. *Method cleanPath*

```
public List<string> cleanPath(string source)
{
    // get lowest path

    /* KAMUS */
```

```
    /*
        *cleaned : List<string>
        *curr : string
        */

    /* ALGORITMA */

    List<string> cleaned = new List<string>();
    double cost = 0;

    // Reverse path
    path.Reverse();
    string curr = path[0];
    string prev = curr;

    // While reconstruction have not reached path
    while(!curr.Equals(source))
    {
        cost += graph.haversine(prev, curr);

        cleaned.Add(curr);
        prev = curr;
        curr = parentMap[curr];
    }


    // reverse
    cleaned.Reverse();
    return cleaned;
}
```

2.2.2.6. *Method getPath*

```
// get path
public List<string> getPath()
{
    return this.path;
}
```

2.2.2.7. *Method getVisitedCost*

```
// Get cost
public double getVisitedCost(string source, string target)
{
    double cost = 0;
    string curr = source;

    // iterate edge and count cost
    foreach (var x in path)
    {
        cost += graph.haversine(curr, x);
        curr = x;
    }

    cost += graph.haversine(curr, target);


    return cost;
}
```

2.3. Main

2.3.1. Atribut

```
private Graph currGraph;
```

2.3.2. Method

2.3.2.1. *Method Main*

```
public Main()
{
    InitializeComponent();
}
```

2.3.2.2. *Method button1_click*

```
// Browse Button
private void button1_Click(object sender, EventArgs e)
{
    /* KAMUS */
    /*
        * filename : string
        * x : Graph
        * graf :  Microsoft.Msagl.Drawing.Graph
        */

    // get filename
    openFileDialog1.ShowDialog();
    string filename = openFileDialog1.FileName;

    // Construct graph
    currGraph = new Graph(filename);

    // Print Graph
    Microsoft.Msagl.Drawing.Graph graf = currGraph.getMSAGLGraph();
    clear(graf);
    visualizeGraph(graf);

    // Hide first page show second page
    button1.Visible = false;
    label4.Visible = false;
    label5.Visible = false;
    button2.Visible = true;
    button3.Visible = true;
    label1.Visible = true;
    label2.Visible = true;
    comboBox1.Visible = true;
    comboBox2.Visible = true;



    // Add Nodes to Combobox
    addNode(currGraph, 1);
    addNode(currGraph, 2);
}
```

2.3.2.3. *Method visualizeGraph*

```
// Print Graph
private void visualizeGraph(Microsoft.Msagl.Drawing.Graph Graf)
{
    /* Kamus */
    /*
        * renderer : Microsoft.Msagl.GraphViewerGdi.GraphRenderer
        * width, height : int
        * bitmap : Bitmap
        */

    /* ALGORITMA */

    // clear picturebox if there are any image
    if (pictureBox1.Image != null) pictureBox1.Image = null;
    Microsoft.Msagl.GraphViewerGdi.GraphRenderer renderer = new
Microsoft.Msagl.GraphViewerGdi.GraphRenderer(Graf);

    // Calculate layout dimension
    renderer.CalculateLayout();

    int width;
    int height;

    if (Graf.Width > Graf.Height)
    {
        width = 506;
        height = (int)(Graf.Height * (width / Graf.Width));
    }
    else
    {
        height = 600;
        width = (int)(Graf.Width * (height / Graf.Height));
    }
    Graf.Attr.BackgroundColor =
Microsoft.Msagl.Drawing.Color.Transparent;
    // Add graph to picture box
    Bitmap bitmap = new Bitmap(width, height,
PixelFormat.Format32bppPArgb);
    renderer.Render(bitmap);
    pictureBox1.Image = bitmap;
}
```

2.3.2.4. *Method button2_Click*

```
/* Reset button*/
private void button2_Click(object sender, EventArgs e)
{
    /* KAMUS */
    /*
        * Main : Form
    */

    /* Algoritma */

    this.Hide();
    Form Main = new Main();
    Main.Show();

}
```

2.3.2.5. *Method addNode*

```
// Add node to combobox
private void addNode(Graph graph, int y)
{
    /* ALGORITMA */

    // Check Which combobox to clear
    if(y == 1)
    {
        comboBox1.Items.Clear();

    }

    else
    {
        comboBox2.Items.Clear();
    }

    /* add all node to combobox */
    foreach (var x in graph.getNodeLst())
    {
        if(y == 1)
        {
            comboBox1.Items.Add(x.Key);
        }
        else
        {
            comboBox2.Items.Add(x.Key);
        }
    }
}
```

2.3.2.6. *Method comboBox2_SelectedIndexChanged*

```
/* Change graph visualization based on node choice for combobox1 */
private void comboBox2_SelectedIndexChanged(object sender, EventArgs
e)
{
    /* KAMUS */
    /*
        * graf : Microsoft.Msagl.Drawing.Graph
        */

    /* ALGORITMA */

    Microsoft.Msagl.Drawing.Graph graf = currGraph.getMSAGLGraph();
    clear(graf);

    // Check if combobox1 has content to color
    if (comboBox1.GetItemText(comboBox1.SelectedItem).Length > 0)
    {

graf.FindNode(comboBox1.GetItemText(comboBox1.SelectedItem)).Attr.Fill
Color = Microsoft.Msagl.Drawing.Color.Green;
    }

    // color node

graf.FindNode(comboBox2.GetItemText(comboBox2.SelectedItem)).Attr.Fill
Color = Microsoft.Msagl.Drawing.Color.Magenta;
```

```
    visualizeGraph(graf);


}
```

2.3.2.7. *Method comboBox1_SelectedIndexChanged*

```
/* Change graph visualization based on node choice for combobox2 */
private void comboBox1_SelectedIndexChanged(object sender, EventArgs
e)
{
    /* KAMUS */

    /*
        * graf : Microsoft.Msagl.Drawing.Graph
        * combo2Content : string
        */

    /* ALGORITMA */

    Microsoft.Msagl.Drawing.Graph graf = currGraph.getMSAGLGraph();
    string combo2Content = "";

    clear(graf);

    // Check if combobox2 has content to color
    if (comboBox2.GetItemText(comboBox2.SelectedItem).Length > 0)
    {

graf.FindNode(comboBox2.GetItemText(comboBox2.SelectedItem)).Attr.Fill
Color = Microsoft.Msagl.Drawing.Color.Magenta;
        cmbo2Content = comboBox2.GetItemText(comboBox2.SelectedItem);
    }


graf.FindNode(comboBox1.GetItemText(comboBox1.SelectedItem)).Attr.Fill
Color = Microsoft.Msagl.Drawing.Color.Green;

    // color node
    comboBox2.Items.Clear();
    addNode(currGraph, 2);

comboBox2.Items.Remove(comboBox1.GetItemText(comboBox1.SelectedItem));

    // Remove selecteditem from combobox1 in combobox2;
    if(combo2Content.Length > 0)
    {
        comboBox2.SelectedItem = combo2Content;
    }
    visualizeGraph(graf);
}
```

2.3.2.8. *Method button3_Click*

```
/* Submit button */
private void button3_Click(object sender, EventArgs e)
{
    /* KAMUS */
    /*
        * source, target : string
        * search : AStar
```

```
        * graf : Microsoft.Msagl.Drawing.Graph
        * result : List <string>
        * prev : string
        * edge : Microsoft.Msagl.Drawing.Edge
        */

    /* Algoritma */


    // Check if source and target node has been chosen
    if(comboBox1.GetItemText(comboBox1.SelectedItem).Length > 0 &&
comboBox2.GetItemText(comboBox2.SelectedItem).Length > 0)
    {

        string source = comboBox1.GetItemText(comboBox1.SelectedItem);
        string target = comboBox2.GetItemText(comboBox2.SelectedItem);
        Microsoft.Msagl.Drawing.Edge edge;

        // Init search
        AStar search = new AStar(currGraph, source.ToString(),
target.ToString());

        // Check if there are path
        if (search.getStatus() == true)
        {
            Microsoft.Msagl.Drawing.Graph graf =
currGraph.getMSAGLGraph();
            // Get shortest path
            List<string> result = search.cleanPath(source, target);
            string prev = source;

            // Color node and edges
            for (int i = 0; i < result.Count; i++)
            {

if(!result[i].Equals(comboBox1.GetItemText(comboBox1.SelectedItem)) &&
!result[i].Equals(comboBox2.GetItemText(comboBox2.SelectedItem)))
                {
                    graf.FindNode(result[i]).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.DodgerBlue;
                }

                edge = currGraph.filterEdge(prev, result[i]);
                edge.Attr.Color = Microsoft.Msagl.Drawing.Color.Red;
                prev = result[i];
            }


            // Print results
            visualizeGraph(graf);
            label3.Visible = true;
            label3.Text = "Cost : " + search.getVisitedCost(source,
target).ToString();
        }
        else
        {
            MessageBox.Show("No Path Found");
        }
    }
    else
    {
        MessageBox.Show("Complete the field first!");
    }
}
```

2.3.2.9. *Method clear*

```
/* Clear Msagl Graf from customized attribute */
private void clear(Microsoft.Msagl.Drawing.Graph Graf)
{
    /* ALGORITMA */

    // Color all node to transparent
    foreach (var x in currGraph.getNodeLst())
    {
        Graf.FindNode(x.Key).Attr.FillColor =
Microsoft.Msagl.Drawing.Color.White;
    }

    // Color all edge to black
    foreach(var x in currGraph.getGUIEdge())
    {
        x.Value.Attr.Color = Microsoft.Msagl.Drawing.Color.Black;
    }
}
```

BAB III

Screenshot Uji

3.1. Uji 1

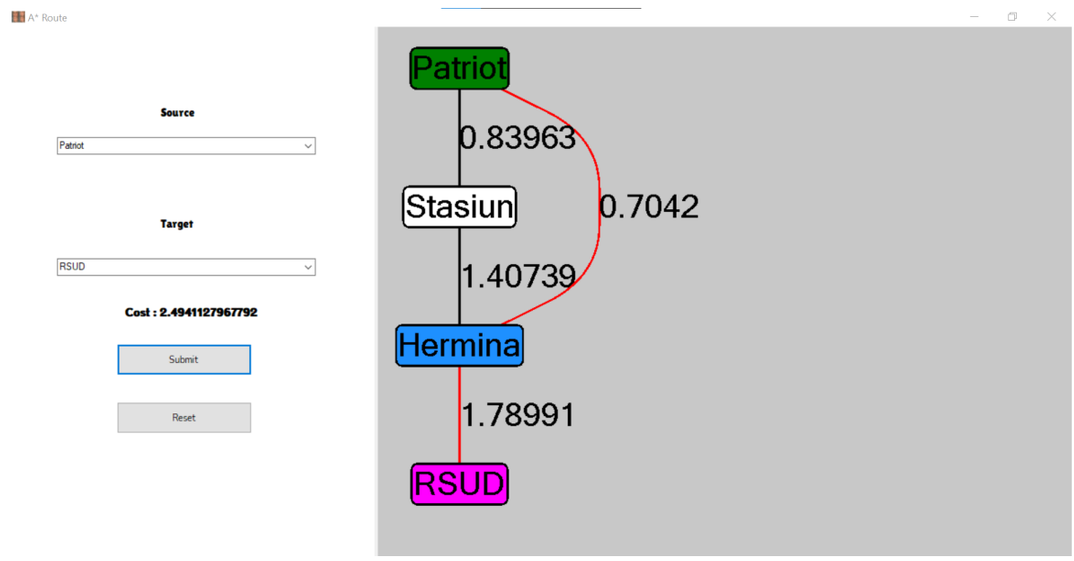
3.1.1. Input

```
4
-6.238393142992205 106.99183138081494 Patriot
-6.23447540694475 106.98682592794263 Hermina
-6.2370137614412675 106.9992995001439 Stasiun
-6.24200511367606 107.00113805383562 RSUD
0 1 1 0 0
1 0 0 1 1
1 1 0 0 0
0 1 0 0 0
```

3.1.2. Output

Graf awal

Source : Patriot, Target : RSUD



3.2. Uji 2

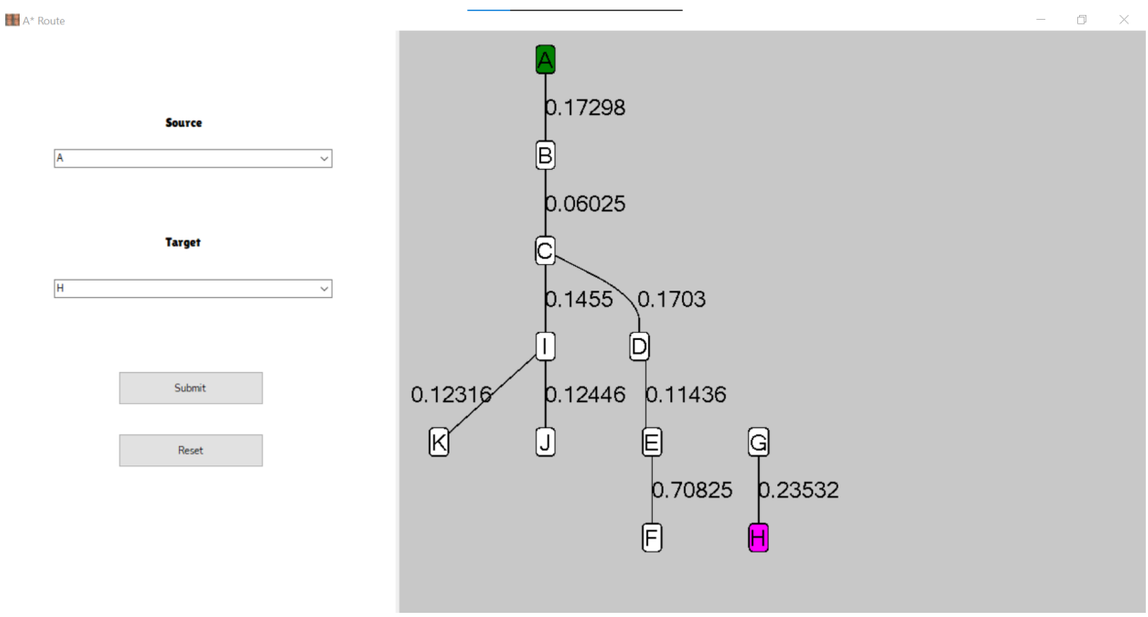
3.2.1. Input

```
11
```

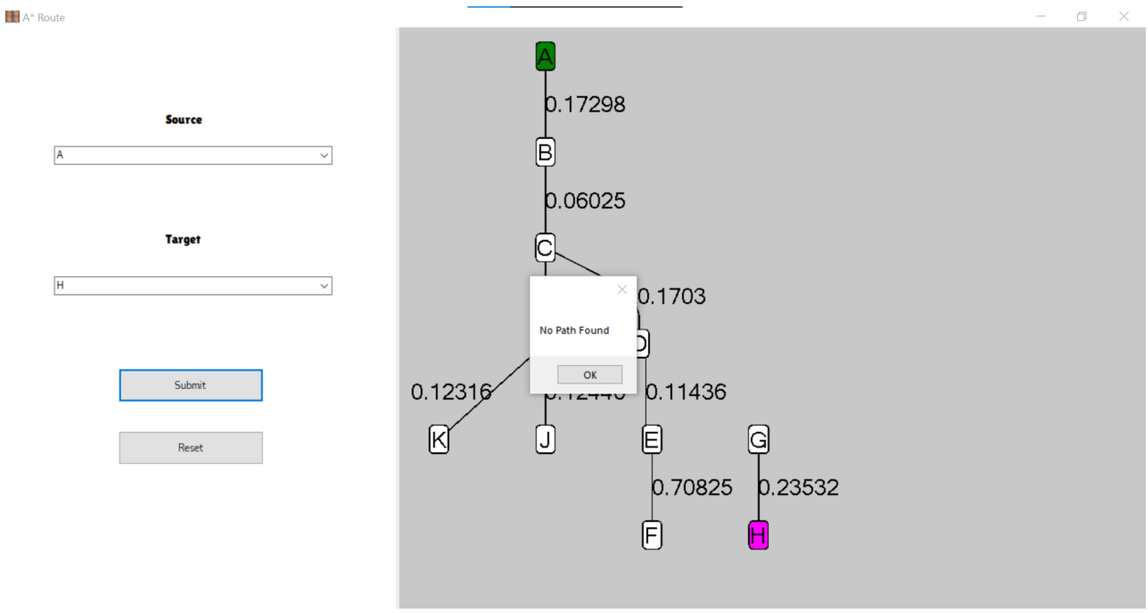
```
-6.893874 107.608454 A
-6.893294 107.609908 B
-6.893230 107.610450 C
-6.893592 107.611949 D
-6.893754 107.612972 E
-6.887412 107.613567 F
-6.887363 107.611469 G
-6.887703 107.609365 H
-6.891923 107.610386 I
-6.891034 107.609701 J
-6.891013 107.611022 K
0 1 0 0 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 1 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 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 1 0 0 0 0
0 0 1 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 0 0
```

3.2.2. Output

Source Node : A, Source Target : H



No Path Found



3.3. Uji 3

3.3.1. Input

```
8
-6.892674095579858 107.610456766119 ITB
-6.893685967570809 107.61119685795033 Salman
-6.894527417370569 107.6136852826586 Boromeus
-6.890295300177787 107.61281647346945 SPBU_Dago
-6.885090312494578 107.61351686089003 MCD
-6.886144806591044 107.61136736020688 Baksil
-6.886251321000536 107.60986572460708 Saraga
-6.891136755996797 107.60698154656727 Bonbin
0 1 1 0 0 0 0 1
1 0 1 0 0 0 0 0
1 1 0 1 0 0 0 0
0 0 1 0 1 0 0 0
0 0 0 1 0 1 1 0
0 0 0 0 1 0 1 0
0 0 0 0 1 1 0 1
1 0 0 0 0 0 1 0
```

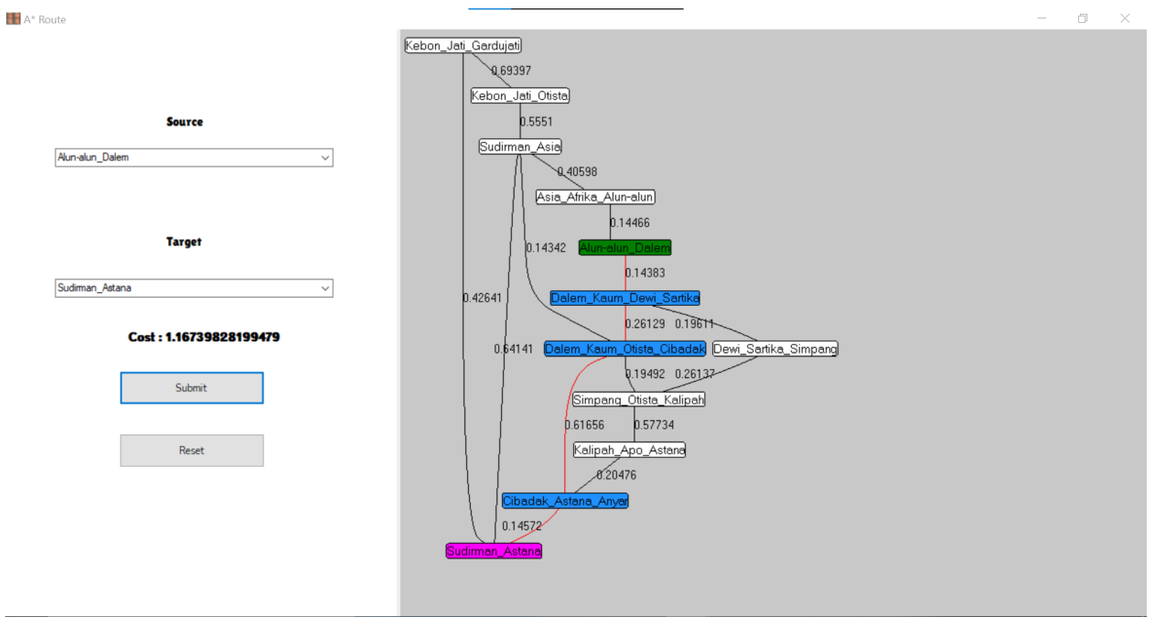
3.3.2. Output

3.4. Uji 4

3.4.1. Input

```
12
-6.916289558795759 107.59823109564569 Kebon_Jati_Gardujati
-6.915799623402846 107.60449846018152 Kebon_Jati_Otista
-6.920773509104472 107.60406918863798 Sudirman_Asia Afrika_Otista
-6.921231487380398 107.60771799675814 Asia_Afrika_Alun-alun
-6.9225308651860935 107.60765360602662 Alun-alun_Dalem Kaum
-6.92241370831757 107.6063559748914 Dalem_Kaum_Dewi_Sartika
-6.9220622375374745 107.60401552969505 Dalem_Kaum_Otista_Cibadak
-6.9241710582913045 107.6062057298512 Dewi_Sartika_Simpang
-6.9238089382260455 107.60386619993885 Simpang_Otista_Kalipah Apo
-6.9232540419560005 107.59866591318783 Kalipah_Apo_Astana Anyar
-6.921423199084892 107.59846719499465 Cibadak_Astana_Anyar
-6.9201238182298335 107.59829548637724 Sudirman_Astana Anyar
0 1 0 0 0 0 0 0 0 0 0 1
1 0 1 0 0 0 0 0 0 0 0 0
0 1 0 1 0 0 1 0 0 0 0 1
0 0 1 0 1 0 0 0 0 0 0 0
0 0 0 1 0 1 0 0 0 0 0 0
0 0 0 0 1 0 1 1 0 0 0 0
0 0 1 0 0 1 0 0 1 0 1 0
0 0 0 0 0 1 0 0 1 0 0 0
0 0 0 0 0 0 1 1 0 1 0 0
0 0 0 0 0 0 0 0 1 0 1 0
0 0 0 0 0 0 1 0 0 1 0 1
1 0 1 0 0 0 0 0 0 0 1 0
```

3.4.2. Output



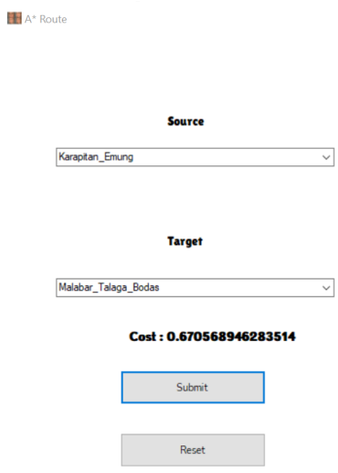
3.5. Uji 5

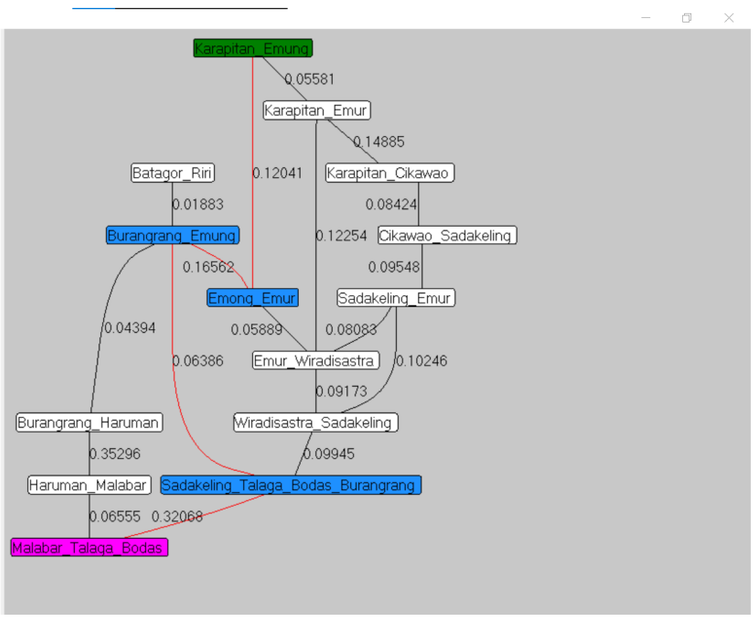
3.5.1. Input

```
14
-6.927562192722218 107.61959799634626 Batagor_Riri
-6.927025406721871 107.61690969806743 Karapitan_Emung
-6.927472728417712 107.61945313196297 Burangrang_Emung
-6.927312970717906 107.61796141334914 Emong_Emur
-6.927504679951173 107.61675945302719 Karapitan_Emur
-6.928793389996319 107.61639457221517 Karapitan_Cikawao
```

```
-6.92914485576096 107.61707067489624 Cikawao_Sadakeling
-6.928537778366568 107.61768238684581 Sadakeling_Emur
-6.927824195166772 107.61782190009745 Emur_Wiradisastra
-6.928217198601289 107.61855260357831 Wiradisastra_Sadakeling
-6.928046790655269 107.61943701951742
Sadakeling_Talaga_Bodas_Burangrang
-6.927077594291167 107.61945311720032 Burangrang_Haruman
-6.927871057396435 107.62254923820818 Haruman_Malabar
-6.928414232712092 107.62231850475352 Malabar_Talaga_Bodas
0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0 0 0 0 0 0 0
1 0 0 1 0 0 0 0 0 0 1 1 0 0
0 1 1 0 0 0 0 0 1 0 0 0 0 0
0 1 0 0 1 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 0 1 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 1 0 0 0
0 0 0 1 1 0 0 1 0 1 0 0 0 0
0 0 0 0 0 0 0 1 1 0 1 0 0 0
0 0 1 0 0 0 0 0 0 1 0 0 0 1
0 0 1 0 0 0 0 0 0 0 0 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 0 0 1 0 1 0
```

3.5.2. Output





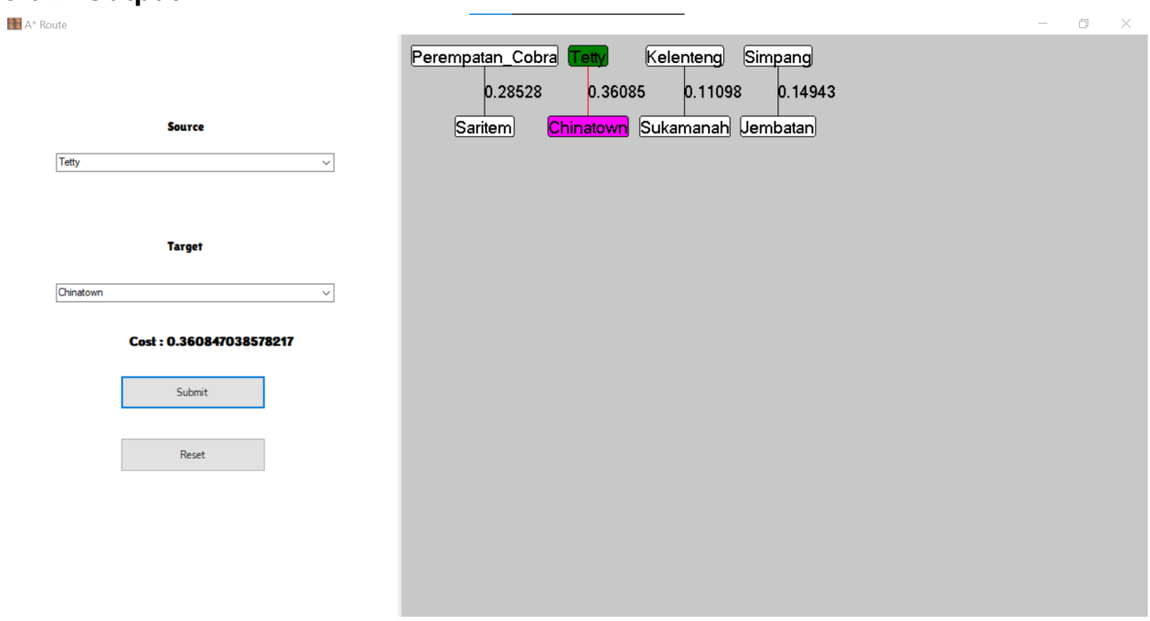
3.6. Uji 6

3.6.1. Input

```
8
-6.918470830235298 107.59625136852266 Tetty
-6.9171075367865145 107.59328484535219 Chinatown
-6.918193911572041 107.59314537048341 Kelenteng
-6.918353672359124 107.59413778781892 Sukamanah
-6.91946134566292 107.5929629802704 Perempatan_Cobra
-6.917192742742358 107.5941699743271 Saritem
-6.9170436323095394 107.59727597236635 Simpang
-6.917459011255302 107.5959885120392 Jembatan
0 1 0 0 0 0 0 0
1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 1 0 0 0
```

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

3.6.2. Output

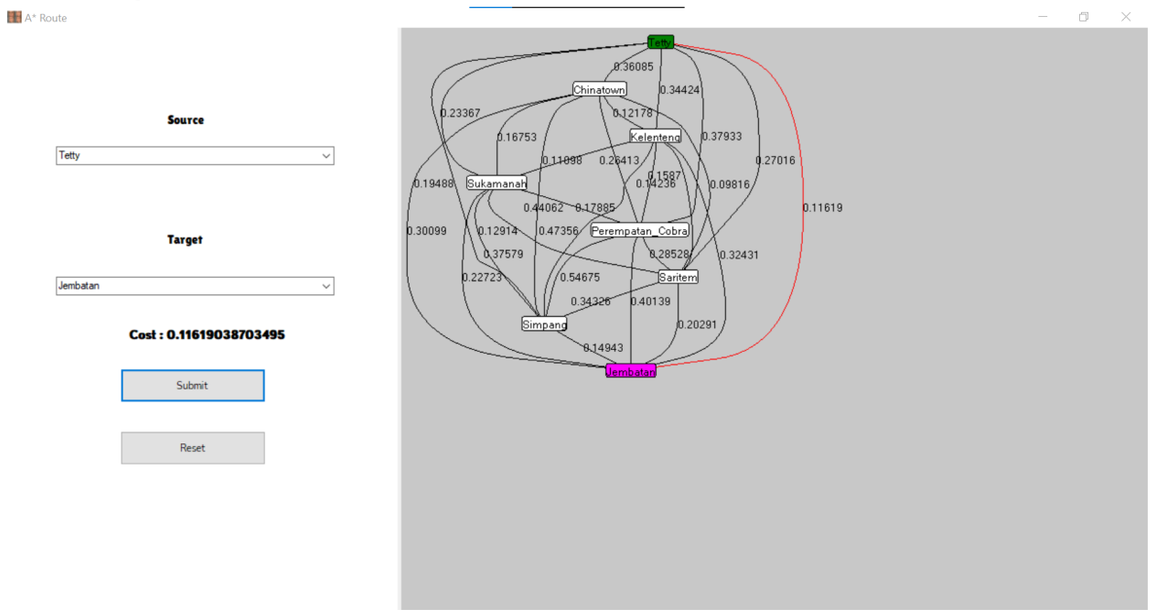


3.7. Uji 7

3.7.1. Input

```
8
-6.918470830235298 107.59625136852266 Tetty
-6.9171075367865145 107.59328484535219 Chinatown
-6.918193911572041 107.59314537048341 Kelenteng
-6.918353672359124 107.59413778781892 Sukamanah
-6.91946134566292 107.5929629802704 Perempatan_Cobra
-6.917192742742358 107.5941699743271 Saritem
-6.9170436323095394 107.59727597236635 Simpang
-6.917459011255302 107.5959885120392 Jembatan
0 1 1 1 1 1 1 1
1 0 1 1 1 1 1 1
1 1 0 1 1 1 1 1
1 1 1 0 1 1 1 1
1 1 1 1 0 1 1 1
1 1 1 1 1 0 1 1
1 1 1 1 1 1 0 1
1 1 1 1 1 1 1 0
```

3.7.2. Output


A* Route
Source
Tetty
Target
Jembatan
Cost : 0.11619038703495
Submit
Reset
Tetty
Chinatown
Kelenteng
Sukamanah
Perempatan_Cobra
Saritem
Simpang
Jembatan
0.36085
0.34424
0.23367
0.12178
0.16753
0.37933
0.11088
0.26413
0.27016
0.1587
0.19488
0.14236
0.09816
0.44062
0.17885
0.11619
0.30099
0.12914
0.47356
0.37579
0.28528
0.32431
0.22723
0.54675
0.34326
0.40139
0.20291
0.14943

BAB IV

Lampiran

4.1. Alamat *Repository*

https://github.com/isabellahandayani/AStar-Route

4.2. *Checklist*

Poin	Ya	Tidak
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		✓