

**LAPORAN TUGAS KECIL 3 IF2211 STRATEGI ALGORITMA**  
**IMPLEMENTASI ALGORITMA A\* UNTUK MENENTUKAN LINTASAN**  
**TERPENDEK**

Disusun dalam rangka memenuhi tugas Strategi Algoritma (IF2211)



Disusun oleh :

Muhammad Bintang Pananjung      13519004

Muhammad Rayhan Ravianda      13519201

**PROGRAM STUDI TEKNIK INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**BANDUNG**

**2020**

## **BAB I**

### **PENJELASAN ALGORITMA**

Kami menggunakan bahasa C# dalam mengerjakan tugas kecil ini. Kami banyak mendaur ulang beberapa program dari tugas besar 2 milik kami dengan memodifikasi beberapa komponennya. Dalam program kami, terdapat kelas Graph, Edge, simpul, Vertex, ListVertex, simpulSumCost.cs, Form1.cs.

Algoritma yang kami buat mirip dengan algoritma A\* yang diajarkan dikelas, yaitu dengan mencari nilai minimum dari SumCost setiap simpul yang dilalui menggunakan algoritma BFS. SumCost adalah penjumlahan antara jarak dari simpul yang dilalui ke titik awalnya dengan jarak dari simpul yang dilalui ke titik destinasi. Untuk ini, kami membuat simpul dalam bentuk kelas yang memiliki atribut berupa nama, koordinat x, dan koordinat y. Koordinat ini nantinya akan digunakan untuk perhitungan SumCost dan perhitungan bobot dari masing-masing sisi. Perhitungan jaraknya menggunakan jarak euclidean.

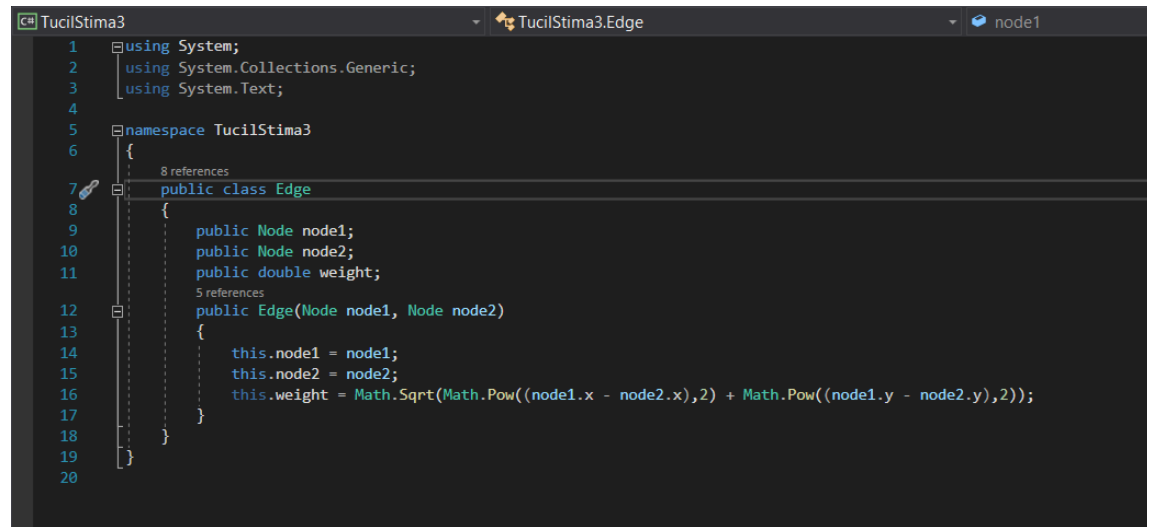
Mula-mula program akan menerima input dari pengguna berupa nama file, simpul awal, dan simpul tujuan. Kemudian, program akan mengolah file yang diinput menjadi graph. Dari simpul awal dan tujuan yang dimasukkan, akan diproses pencarian jalan terpendek menggunakan algoritma A\*. Algoritma A\* dimulai dengan membuat list NodeSumCost yang berisi simpul beserta SumCostnya. Kemudian, list ini akan dimasukkan simpul awalnya. Setelah itu, program akan iterasi sampai list tadi kosong atau simpul tujuan ketemu. List ini akan berfungsi seperti queue pada BFS, hanya saja tiap iterasi yang dihapus dari list adalah yang nilai SumCostnya minimum. Sebelum dihapus, simpul tadi disimpan ke dalam variabel yang kemudian akan disimpan kedalam ListVertex. ListVertex berguna untuk menandai jalur dari simpul-simpul yang telah dilalui. Kemudian, dicari tetangga dari simpul yang sedang diproses. Setiap tetangga yang diakses, jika belum dikunjungi (masuk kedalam list NodeSumCost), maka akan ditambahkan ke dalam ListVertex. Jika ketemu simpul yang dicari, maka ListVertex akan disimpan sebagai return value. ListVertex ini akan dibacktrack dari simpul tujuan yang berada ditetangga dari suatu simpul, sampai ketemu simpul awalnya. Kemudian hasil dari backtrack ini berupa shortestpath yang nanti akan divisualisasikan menggunakan MSAGL.

## BAB II

### SOURCE CODE PROGRAM

Pada bab kali ini akan ditampilkan source code program yang terdiri dari beberapa file sesuai classnya. Source code program adalah sebagai berikut,

#### 1. edge.cs



```
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4
5 namespace TucilStima3
6 {
7     public class Edge
8     {
9         public Node node1;
10        public Node node2;
11        public double weight;
12
13        public Edge(Node node1, Node node2)
14        {
15            this.node1 = node1;
16            this.node2 = node2;
17            this.weight = Math.Sqrt(Math.Pow((node1.x - node2.x),2) + Math.Pow((node1.y - node2.y),2));
18        }
19    }
20 }
```

#### 2. Form1.cs

```

TucilStima3 TucilStima3.Form1 LoadFileVis(string filename, Graph graph, List<Node> path, List<double> pathCost, Graph thisGraph)
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10
11 namespace TucilStima3
12 {
13     3 references
14     public partial class Form1 : Form
15     {
16         string iniNamaFile;
17
18     1 reference
19     public Form1()
20     {
21         InitializeComponent();
22
23     1 reference
24     public void LoadFileVis(string filename, Microsoft.Msagl.Drawing.Graph graph, List<Node> path, List<double> pathCost, Graph thisGraph)
25     {
26         string pathFile = @"..\..\..\test\" + filename;
27         string[] lines = System.IO.File.ReadAllLines(filename);
28         bool edge = false;
29         for (int idx = 0; idx < lines.Length; idx++)
30         {
31             string node1 = "";
32             string node2 = "";
33             bool parse = false;
34             int space = 0;
35             if (!edge)
36             {
37                 string xtemp = "";
38                 string ytemp = "";
39                 for (int i = 0; i < lines[idx].Length; i++)
40                 {
41                     if (lines[idx][i] == '~')

```

```

TucilStima3 TucilStima3.Form1 Form1()
41         edge = true;
42         //Console.WriteLine(edge);
43     }
44     else
45     {
46         if (lines[idx][i] != ' ' && space == 0)
47         {
48             node1 += lines[idx][i];
49         }
50         if (lines[idx][i] != ' ' && space == 1)
51         {
52             xtemp += lines[idx][i];
53         }
54         if (lines[idx][i] != ' ' && space == 2)
55         {
56             ytemp += lines[idx][i];
57         }
58         if (lines[idx][i] == ' ')
59         {
60             space++;
61         }
62     }
63     if (!edge)
64     {
65         graph.AddNode(node1);
66     }
67     }
68     else
69     {
70         for (int i = 0; i < lines[idx].Length; i++)
71         {
72             if (lines[idx][i] != ' ' && !parse)
73             {
74                 node1 += lines[idx][i];
75             }
76             if (lines[idx][i] != ' ' && parse)
77             {
78                 node2 += lines[idx][i];
79             }
80             if (lines[idx][i] == ' ')
81             {
82

```

```

TucilStima3 TucilStima3.Form1 LoadFileVis(string filename, Graph graph, List<Node> path
80
81 if (lines[idx][i] == ' ')
82 {
83     parse = true;
84 }
85 }
86 var e = graph.AddEdge(node1, node2);
87 e.Attr.ArrowheadAtSource = Microsoft.Msagl.Drawing.ArrowStyle.None;
88 e.Attr.ArrowheadAtTarget = Microsoft.Msagl.Drawing.ArrowStyle.None;
89 e.LabelText = (thisGraph.euclideanDistance(thisGraph.findNode(node1).x - thisGraph.findNode(node2).x, thisGraph.findNode(node1).y
90 int x;
91 for (x = 0; x < path.Count - 1; x++)
92 {
93     if ((node1 == path[x].name && node2 == path[x + 1].name) || (node2 == path[x].name && node1 == path[x + 1].name))
94     {
95         break;
96     }
97 }
98 if (x == path.Count - 1)
99 {
100 }
101 else
102 {
103     e.Attr.Color = Microsoft.Msagl.Drawing.Color.Red;
104 }
105 }
106 }
107 }
108 }
109 }
110 private void buttonBrowse_Click(object sender, EventArgs e)
111 {
112     openFileDialog1.Title = "Open file";
113     openFileDialog1.ShowDialog();
114     iniNamaFile = System.IO.Path.GetFileName(openFileDialog1.FileName);
115     labelFileName.Text = iniNamaFile;
116 }
117 }
118 }
119 private void startNode_TextChanged(object sender, EventArgs e)
120 {

```

```

TucilStima3 TucilStima3.Form1 LoadFileVis(string filename, Graph graph, List<Node> path
126
127 }
128 }
129 private void buttonSubmit_Click(object sender, EventArgs e)
130 {
131     panelGraf.Controls.Clear();
132     Graph filenyaNi = new Graph();
133     filenyaNi.LoadFile(iniNamaFile);
134     string startPlace = startNode.Text;
135     string destPlace = destNode.Text;
136     Node start = filenyaNi.findNode(startPlace);
137     Node destination = filenyaNi.findNode(destPlace);
138     ListVertex result = filenyaNi.shortestPath(start, destination);
139     List<Node> path = new List<Node>();
140     List<double> pathCost = new List<double>();
141     if (result.edgeList[(result.edgeList.Count - 1)].edges.Find(e => e.node2 == destination) != null)
142     {
143         result.findPath(path, start, destination);
144         for (int i = 0; i < path.Count - 1; i++)
145         {
146             pathCost.Add(filanyaNi.euclideanDistance(path[i].x - path[i + 1].x, path[i].y - path[i + 1].y));
147         }
148         //result.printPath(path, pathCost);
149     }
150     else
151     {
152         string msgtext = "Tidak ada jalur koneksi yang tersedia, Anda harus memulai koneksi baru itu sendiri.";
153         MessageBox.Show(msgtext);
154     }
155 }
156 }
157 Microsoft.Msagl.GraphViewerGdi.GViewer viewer = new Microsoft.Msagl.GraphViewerGdi.GViewer();
158 Microsoft.Msagl.Drawing.Graph graph = new Microsoft.Msagl.Drawing.Graph("graph");
159 LoadFileVis(iniNamaFile, graph, path, pathCost, filenyaNi);
160 viewer.Graph = graph;
161 panelGraf.SuspendLayout();
162 viewer.Dock = System.Windows.Forms.DockStyle.Fill;
163 panelGraf.Controls.Add(viewer);
164 viewer.ResumeLayout();
165 }
166 }

```

### 3. Graph.cs

```
TucilStima3 TucilStima3.Graph adjacencyList
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4
5 namespace TucilStima3
6 {
7     6 references
8     public class Graph
9     {
10         public List<Vertex> adjacencyList;
11
12         2 references
13         public Graph()
14         {
15             adjacencyList = new List<Vertex>();
16         }
17
18         0 references
19         public void AddEdge(Node node1, Node node2)
20         {
21             adjacencyList.Find(v => v.node.name == node1.name).edges.Add(new Edge(node1,node2));
22             adjacencyList.Find(v => v.node.name == node2.name).edges.Add(new Edge(node2,node1));
23         }
24
25         1 reference
26         public void AddEdge(string node1, string node2)
27         {
28             Node firstNode = adjacencyList.Find(v => v.node.name == node1).node;
29             Node secondNode = adjacencyList.Find(v => v.node.name == node2).node;
30             adjacencyList.Find(v => v.node.name == node1).edges.Add(new Edge(firstNode, secondNode));
31             adjacencyList.Find(v => v.node.name == node2).edges.Add(new Edge(secondNode, firstNode));
32         }
33
34         1 reference
35         public void AddNode(Node newNode)
36         {
37             if (adjacencyList.Find(v => v.node == newNode) == null)
38             {
39                 adjacencyList.Add(new Vertex(newNode));
40             }
41         }
42
43         0 references
44         public void PrintadjacencyList()
45         {
46             if (adjacencyList.Count != 0)
47             {
48                 // ...
49             }
50         }
51     }
52 }
```

```
TucilStima3 TucilStima3.Graph adjacencyList
38         foreach (var v in adjacencyList)
39             v.PrintEdge();
40     }
41 }
42 6 references
43 public Node findNode(string name)
44 {
45     return adjacencyList.Find(v => v.node.name == name).node;
46 }
47 2 references
48 public void LoadFile(string filename)
49 {
50     string pathFile = @"..\..\..\test" + filename;
51     string[] lines = System.IO.File.ReadAllLines(filename);
52     bool edge = false;
53     for (int idx = 0; idx < lines.Length; idx++)
54     {
55         string node1 = "";
56         string node2 = "";
57         bool parse = false;
58         int space = 0;
59         if (!edge)
60         {
61             string xtemp = "";
62             string ytemp = "";
63             for (int i = 0; i < lines[idx].Length; i++)
64             {
65                 if (lines[idx][i] == '~')
66                 {
67                     edge = true;
68                     //Console.WriteLine(edge);
69                 }
70                 else
71                 {
72                     if (lines[idx][i] != ' ' && space == 0)
73                     {
74                         node1 += lines[idx][i];
75                     }
76                     if (lines[idx][i] != ' ' && space == 1)
77                     {
78                         xtemp += lines[idx][i];
79                     }
80                     if (lines[idx][i] != ' ' && space == 2)
```

```
TucilStima3 TucilStima3.Graph LoadFile(strin
78 if (lines[idx][i] != ' ' && space == 2)
79 {
80     ytemp += lines[idx][i];
81 }
82 if (lines[idx][i] == ' ')
83 {
84     space++;
85 }
86 }
87 }
88 /*Console.WriteLine(lines[idx]);
89 Console.WriteLine(node1);
90 Console.WriteLine(xtemp);
91 Console.WriteLine(ytemp);
92 Console.WriteLine(space);*/
93 if (!edge)
94 {
95     AddNode(new Node(node1, Convert.ToDouble(xtemp), Convert.ToDouble(ytemp)));
96 }
97 }
98 else
99 {
100     for (int i = 0; i < lines[idx].Length; i++)
101     {
102         if (lines[idx][i] != ' ' && !parse)
103         {
104             node1 += lines[idx][i];
105         }
106         if (lines[idx][i] != ' ' && parse)
107         {
108             node2 += lines[idx][i];
109         }
110         if (lines[idx][i] == ' ')
111         {
112             parse = true;
113         }
114     }
115     AddEdge(node1, node2);
116 }
117 }
118 }
```



```
TuclIStima3
TuclIStima3.Graph
LoadFile(string filename)

117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156

    }
    4 references
    public double euclideanDistance(double a, double b)
    {
        return Math.Sqrt(Math.Pow(a, 2) + Math.Pow(b, 2));
    }
    2 references
    public double SumCostCount(Node tempStart, Node tempAdjacent, Node destination)
    {
        return euclideanDistance(tempStart.x - tempAdjacent.x, tempStart.y - tempAdjacent.y) + euclideanDistance(tempAdjacent.x - destination.x,
    }
    0 references
    public void printListNodeSumCost(List<NodeSumCost> listsum)
    {
        for(int i = 0; i < listsum.Count; i++)
        {
            Console.WriteLine(listsum[i].tempNode.name + "(" + (listsum[i].sumCost).ToString() + ") ");
        }
        Console.WriteLine("");
        Console.WriteLine("");
    }
    1 reference
    public void BFSSearch(List<NodeSumCost> queue, Queue<Node> visited, Node start, Node destination, ListVertex result)
    {
        Node node;
        bool found = false;
        Node adjacentnode;
        while (queue.Count != 0 && !found)
        {
            queue.Sort((a, b) => Convert.ToInt32(a.sumCost) - Convert.ToInt32(b.sumCost));
            //printlnListNodeSumCost(queue);
            node = queue[0].tempNode;
            queue.RemoveAt(0);
            result.addNodeList(node);
            for (int i = 0; i < adjacencyList.Find(v => v.node.name == node.name).edges.Count; i++)
            {
                adjacentnode = adjacencyList.Find(v => v.node.name == node.name).edges[i].node2;
                if (!visited.Contains(adjacentnode) )
                {
                    queue.Add(new NodeSumCost(adjacentnode, SumCostCount(start, adjacentnode, destination)));
                    if (adjacentnode == destination)
                    {
                        //printlnListNodeSumCost(queue);
                        result.addNodeList(adjacentnode);
                        return;
                    }
                }
            }
        }
    }
}
```

```

TucilStima3 TucilStima3.Graph BFSSearch(List<No
148     for (int i = 0; i < adjacencyList.Find(v => v.node.name == node.name).edges.Count; i++)
149     {
150         adjacentnode = adjacencyList.Find(v => v.node.name == node.name).edges[i].node2;
151         if (!visited.Contains(adjacentnode) )
152         {
153             queue.Add(new NodeSumCost(adjacentnode, SumCostCount(start, adjacentnode, destination)));
154             if (adjacentnode == destination)
155             {
156                 queue.Sort((a, b) => Convert.ToInt32(a.sumCost) - Convert.ToInt32(b.sumCost));
157                 if (queue[0].tempNode == destination)
158                 {
159                     visited.Enqueue(adjacentnode);
160                     result.addEdgeList(node, adjacentnode);
161                     found = true;
162                     break;
163                 }
164             }
165             else
166             {
167                 visited.Enqueue(adjacentnode);
168                 result.addEdgeList(node, adjacentnode);
169             }
170         }
171     }
172 }
173 }
174 }
2 references
175 public ListVertex shortestPath(Node start, Node destination)
176 {
177     List<NodeSumCost> queue = new List<NodeSumCost>();
178     ListVertex result = new ListVertex();
179     Queue<Node> visited = new Queue<Node>();
180     queue.Add(new NodeSumCost(start, SumCostCount(start, start, destination)));
181     BFSSearch(queue, visited, start, destination, result);
182     return result;
183 }
184 }
185 }
186 }

```

#### 4. ListVertex.cs

```

TucilStima3 TucilStima3.ListVertex printPat
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4
5 namespace TucilStima3
6 {
7     6 references
8     public class ListVertex
9     {
10         public List<Vertex> edgeList;
11         1 reference
12         public ListVertex()
13         {
14             edgeList = new List<Vertex>();
15         }
16         2 references
17         public void addEdgeList(Node node1, Node node2)
18         {
19             edgeList.Find(v => v.node.name == node1.name).edges.Add(new Edge(node1,node2));
20         }
21         1 reference
22         public void addNodeList(Node newNode)
23         {
24             if (edgeList.Find(v => v.node.name == newNode.name) == null)
25             {
26                 edgeList.Add(new Vertex(newNode));
27             }
28         }
29
30         1 reference
31         public void findPath(List<Node> path, Node name, Node other)
32         {
33             Node tempPath = other;
34             for (int i = edgeList.Count - 1; i >= 0; i--)
35             {
36                 if (edgeList[i].edges.Find(v => v.node2.name==tempPath.name)!=null)
37                 {
38                     path.Add(tempPath);
39                     tempPath = edgeList[i].node;
40                 }
41             }
42             path.Add(name);
43         }
44     }
45 }

```

```

TucilStima3 TucilStima3.ListVertex print
33 path.Add(tempPath);
34 tempPath = edgeList[i].node;
35 }
36 }
37 path.Add(name);
38 }
39 0 references
40 public void printPath(List<Node> path, List<double> pathCost)
41 {
42     int i;
43     for (i = path.Count - 1; i > 0; i--)
44     {
45         Console.Write(path[i].name + "-" + pathCost[(i-1)].ToString() + ">");
46     }
47     Console.WriteLine(path[i].name);
48 }
49 }
50

```

## 5. Node.cs

```
TucilStima3 TucilStima3.Node
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4
5 namespace TucilStima3
6 {
7     43 references
8     public class Node
9     {
10         public string name;
11         public double x;
12         public double y;
13         1 reference
14         public Node(string name, double xCor, double yCor)
15         {
16             this.name = name;
17             this.x = xCor;
18             this.y = yCor;
19         }
20     }
```

6. NodeSumCost.cs

```
TucilStima3 TucilStima3.NodeSumCost
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4
5 namespace TucilStima3
6 {
7     7 references
8     public class NodeSumCost
9     {
10         public Node tempNode;
11         public double sumCost;
12         2 references
13         public NodeSumCost(Node tempNode, double sumCost)
14         {
15             this.tempNode = tempNode;
16             this.sumCost = sumCost;
17         }
18     }
```

7. Program.cs

```
TucilStima3 TucilStima3.Program
7
8 namespace TucilStima3
9 {
10     0 references
11     class Program
12     {
13         [STAThread]
14         0 references
15         static public void Main()
16         {
17             Application.SetHighDpiMode(HighDpiMode.SystemAware);
18             Application.EnableVisualStyles();
19             Application.SetCompatibleTextRenderingDefault(false);
20             Application.Run(new Form1());
21         }
22     }
23 }
```

8. Vertex.cs

```
TucilStima3 TucilStima3.Vertex
1 using System;
2 using System.Collections.Generic;
3
4 namespace TucilStima3
5 {
6     7 references
7     public class Vertex
8     {
9         public List<Edge> edges;
10        public Node node;
11
12        2 references
13        public Vertex(Node node)
14        {
15            this.node = node;
16            this.edges = new List<Edge>();
17        }
18
19        1 reference
20        public void PrintEdge()
21        {
22            Console.Write(node.name + " | ");
23            foreach (var e in edges)
24                Console.Write(e.node2.name + "(" + e.weight + ") " + " ");
25            Console.WriteLine();
26        }
27    }
28 }
```



## BAB III

### HASIL PERCOBAAN

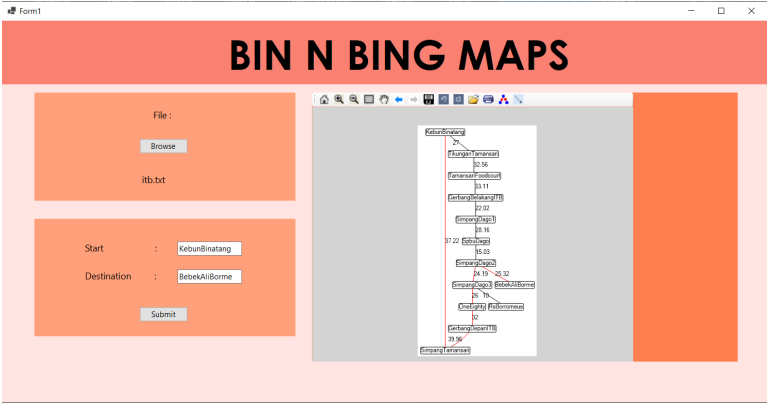
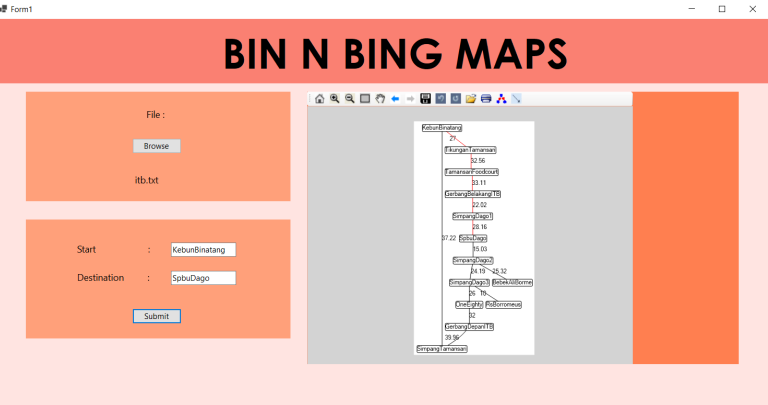
Pada percobaan kali ini dibuat sebanyak 4 file eksternal untuk melakukan testing pada program. Hasil dari percobaan semua test adalah sebagai berikut,

#### 1. itb.txt



```
itb - Notepad
File Edit Format View Help
KebunBinatang -2,3 0
TikunganTamansari -2,3 2,7
TamansariFoodcourt -1 3
GerbangBelakangITB 1,3 3,3
SimpangDago1 3,5 3,2
SpbuDago 3,2 0,4
SimpangDago2 3,1 -1,1
BebekAliBorme 6 -1,5
SimpangDago3 2,8 -3,5
RsBorromeus 2,8 -4,5
OneEighty 2 -3,5
GerbangDepanITB 0,2 -3
SimpangTamansari -1,9 -3,7
~
KebunBinatang TikunganTamansari
TikunganTamansari TamansariFoodcourt
TamansariFoodcourt GerbangBelakangITB
GerbangBelakangITB Sim pangDago1
SimpangDago1 SpbuDago
SpbuDago Sim pangDago2
SimpangDago2 BebekAliBorme
SimpangDago2 Sim pangDago3
SimpangDago3 RsBorromeus
SimpangDago3 OneEighty
OneEighty GerbangDepanITB
GerbangDepanITB Sim pangTamansari
SimpangTamansari KebunBinatang
```

Input	Output
-------	--------

<p>Start : KebunBinatang</p> <p>Destination : BebekAliBorme</p>	
<p>Start : KebunBinatang</p> <p>Destination : SpbuDago</p>	

2. alunalun.txt



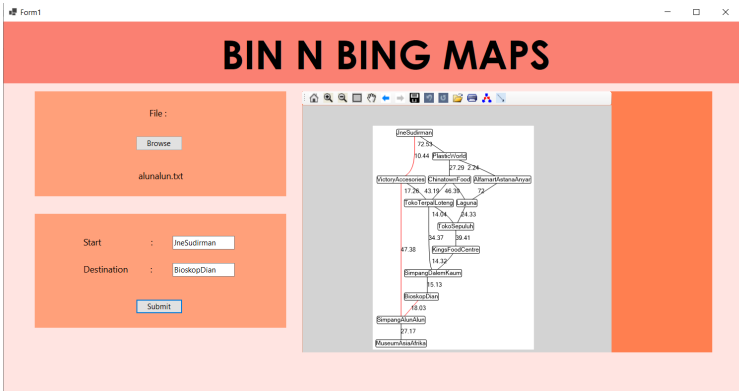
alunalun - Notepad

File Edit Format View Help

```

JneSudirman -8 1,5
PlasticWorld -7,8 -0,4
AlfamartAstanaAnyar -7,7 -2
ChinatownFood -5,1 -0,8
Laguna -5 -2
VictoryAccesories -0,5 0,5
TokoTerpalLoteng -0,8 -1,2
TokoSepuluh -0,9 -2,6
KingsFoodCentre 2,3 -3
SimpangDalemKaum 2,6 -1,7
BioskopDian 4,1 -1,9
SimpangAlunAlun 4,2 -0,1
MuseumAsiaAfrika 6,9 -0,4
~
JneSudirman PlasticWorld
JneSudirman VictoryAccesories
PlasticWorld ChinatownFood
PlasticWorld AlfamartAstanaAnyar
AlfamartAstanaAnyar Laguna
ChinatownFood Laguna
ChinatownFood TokoTerpalLoteng
Laguna TokoSepuluh
VictoryAccesories TokoTerpalLoteng
TokoTerpalLoteng TokoSepuluh
VictoryAccesories SimpangAlunAlun
TokoTerpalLoteng SimpangDalemKaum
TokoSepuluh KingsFoodCentre
KingsFoodCentre SimpangDalemKaum
SimpangDalemKaum BioskopDian
BioskopDian SimpangAlunAlun
SimpangAlunAlun MuseumAsiaAfrika

```

Input	Output
<p>Start : JneSudirman</p> <p>Destination : BioskopDian</p>	 <p>The screenshot shows a web application window titled 'Form1'. The main heading is 'BIN N BING MAPS'. On the left, there is a file upload section with a 'File:' label, a 'Browse' button, and the filename 'alunalun.txt'. Below this is a form with 'Start' and 'Destination' labels, each followed by a text input field containing 'JneSudirman' and 'BioskopDian' respectively, and a 'Submit' button. On the right, a map is displayed with a red line indicating a route between various locations. The locations on the map include JneSudirman, PlasticWorld, VictoryAccesories, ChinatownFood, AlfamartAstanaAnyar, Laguna, TokoTerpalLoteng, TokoSepuluh, KingsFoodCentre, SimpangDalemKaum, BioskopDian, SimpangAlunAlun, and MuseumAsiaAfrika. The map also shows coordinates for several points.</p>

Destination : MuseumAsiaAfrika

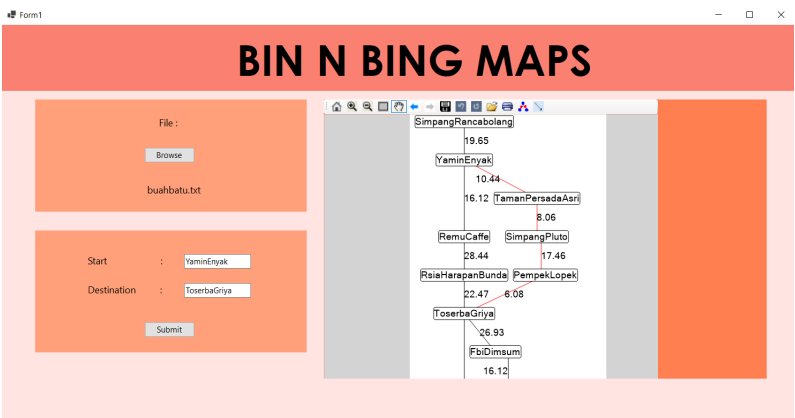
## BIN N BING MAPS

3. buahbatu.txt

```

buahbatu - Notepad
File Edit Format View Help
SimpangRancabolang -3,4 1,4
YaminEnyak -3,9 -0,5
RemuCaffe -4,1 -2,1
SimpangPluto -2,8 0
TamanPersadaAsri -2,9 -0,8
PempekLopek -1,1 -0,4
RsiaHarapanBunda -1,3 -2,6
ToserbaGriya -0,5 -0,5
FbiDimsum 0,2 2,1
PegadaianSyariah 1,8 1,9
RayisCollection 3,3 1,6
FourwheelCoffee 1,8 -0,9
MasjidAlMuhajirin 4,6 -1,4
SimpangSaturnusSel 4,9 1,3
~
SimpangRancabolang YaminEnyak
YaminEnyak TamanPersadaAsri
YaminEnyak RemuCaffe
TamanPersadaAsri SimpangPluto
SimpangPluto PempekLopek
PempekLopek ToserbaGriya
RemuCaffe RsiaHarapanBunda
RsiaHarapanBunda ToserbaGriya
ToserbaGriya FbiDimsum
ToserbaGriya FourwheelCoffee
FbiDimsum PegadaianSyariah
PegadaianSyariah RayisCollection
RayisCollection SimpangSaturnusSel
RayisCollection FourwheelCoffee
SimpangSaturnusSel MasjidAlMuhajirin
FourwheelCoffee MasjidAlMuhajirin

```

Input	Output
<p>Start : YaminEnyak</p> <p>Destination : ToserbaGriya</p>	 <p>The screenshot shows the 'BIN N BING MAPS' application. On the left, there is a file upload section with a 'Browse' button and the filename 'buahbatu.txt'. Below it, there is a form with 'Start' set to 'YaminEnyak' and 'Destination' set to 'ToserbaGriya', with a 'Submit' button. On the right, a map is displayed showing a route between the two locations. The route is marked with a red line and includes several intermediate points with their coordinates: SimpangRancabolang (19.65, 10.44), YaminEnyak (16.12, 8.06), TamanPersadaAsri (28.44, 17.46), RemuCaffe (22.47, 6.08), RsiaHarapanBunda (26.93, 16.12), PempekLopek, and FbiDimsum.</p>

```
Start      : YaminEnyak
Destination : RayisCollection
```

4. `jababeka.txt`

jababeka - Notepad

File Edit Format View Help

```
AganPos -5,4 -1,1
WarungKuning -3,9 -2,3
FactoryOutlet -2,6 -3,2
Alfamidi -1,2 -4,1
TempatMakanKita -3,7 0,8
MasjidAlJihad -2,5 -0,2
KlinikNurFajar -1,1 -1
Seafood94 0,2 -2
Bakbul -1,9 0,9
TbMandiriJaya -0,1 0,4
SimpangTapirRaya 1,3 -0,5
WarungSegitiga 0,2 1,1
SpbuJababeka 0,9 1,9
BebekKaleyo 2,2 0,9
WarunkUpnormal 2,1 3,3
~
AganPos TempatMakanKita
AganPos WarungKuning
WarungKuning MasjidAlJihad
WarungKuning FactoryOutlet
FactoryOutlet KlinikNurFajar
FactoryOutlet Alfamidi
Alfamidi Seafood94
Seafood94 SimpangTapirRaya
Seafood94 KlinikNurFajar
KlinikNurFajar TbMandiriJaya
KlinikNurFajar MasjidAlJihad
MasjidAlJihad Bakbul
MasjidAlJihad TempatMakanKita
TempatMakanKita Bakbul
Bakbul WarungSegitiga
Bakbul TbMandiriJaya
TbMandiriJaya WarungSegitiga
TbMandiriJaya SimpangTapirRaya
SimpangTapirRaya BebekKaleyo
WarungSegitiga SpbuJababeka
SpbuJababeka BebekKaleyo
SpbuJababeka WarunkUpnormal
BebekKaleyo WarunkUpnormal
```

Input	Output
-------	--------

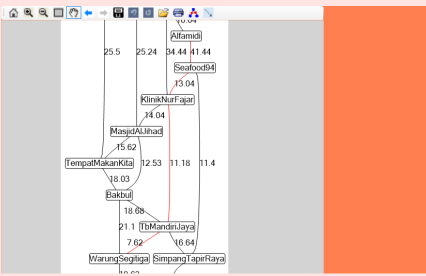
Start : Alfamidi  
Destination : WarungSegitiga

Form1

## BIN N BING MAPS

File :  
  
jababeka.txt

Start :   
Destination :



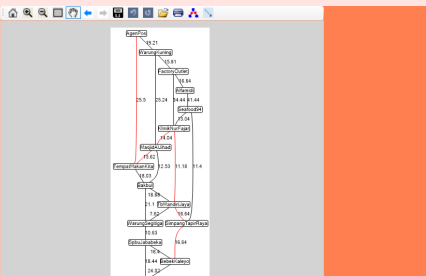
Start : Alfamidi  
Destination : WarungSegitiga

Form1

## BIN N BING MAPS

File :  
  
jababeka.txt

Start :   
Destination :



## BAB IV

### LAMPIRAN

#### A. Tautan Source Code

Semua file terkait Tugas Kecil 3 IF2211 Strategi Algoritma dapat ditemui di tautan <https://github.com/bintangpananjung/Tucil3Stima>

#### B. Tabel Laporan

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		√