TUGAS KECIL 3

IF2211 STRATEGI ALGORITMA

Implementasi Algoritma A* untuk Menentukan Lintasan Terpendek

SEMESTER 2 TAHUN 2020/2021

oleh

Rafi Raihansyah Munandar 13519154

Irvin Andryan Pratomo 13519162

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2021

I. Kode Program

Graph.cs

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

namespace AStar
{
   class Point
   {
       public double X { get; set; }
       public double Y { get; set; }

       public Point(){
           X = 0;
           Y = 0;
       }

       public Point(double x, double y){
           X = x;
           Y = y;
       }

       public void printPoint()
       {
           Console.Write("(");
           Console.Write(this.X);
           Console.Write(",");
           Console.Write(this.Y);
           Console.WriteLine(")");
       }

       public double getEucledianDistance(Point other){
           return Math.Sqrt(Math.Pow(X - other.X, 2) + Math.Pow(Y - other.Y,
2));
           //return Math.sqrt(Math.Pow(this.X - other.X, 2) +
Math.Pow(this.Y - other.Y, 2));
       }
   }

   class Node : IComparable
   {
       public List<Edge> neighbors { get; }
       public String name { get; set; }
       public Point coordinate { get; set; }
```

```
        public int neighborCount => neighbors.Count;
        public bool isVisited { get; set; }
        public Node pred { get; set; }
        public double fn { get; set; }
        public double gn { get; set; }

        int IComparable.CompareTo(object obj)
        {
            Node n = (Node)obj;
            return fn.CompareTo(n.fn);
        }

        public Node(String n, double x, double y)
        {
            neighbors = new List<Edge>();
            name = n;
            coordinate = new Point(x, y);
            isVisited = false;
            pred = null;
            //fn = y;
        }

        public void addEdge(Node n, double d)
        {
            neighbors.Add(new Edge(n, d));
        }

        public void removeEdge(Node n)
        {
            foreach (Edge e in neighbors)
            {
                if (e.target.name == n.name)
                {
                    neighbors.Remove(e);
                    return;
                }
            }
        }

        public double hn(Node target)
        {
            return coordinate.getEucledianDistance(target.coordinate);
        }

        public void nodeInfo()
```

```
        {
            Console.Write($"Tetangga dari {name} adalah");
            foreach (var n in neighbors)
            {
                Console.Write($" {n.target.name}"); ;
            }
            Console.WriteLine("");
        }

        public void neighborInfo()
        {
            Console.WriteLine($"Tetangga dari {name} terdiri dari");
            foreach (var n in neighbors)
            {
                Console.WriteLine($"- {n.target.name} dengan jarak
{n.weight}"); ;
            }
            Console.WriteLine("");
        }
    }

    class Edge
    {
        public Node target { get; set; }
        public double weight { get; set; }

        public Edge(Node n, double w)
        {
            target = n;
            weight = w;
        }
    }

    class Graph
    {
        public List<Node> nodeList { get; }
        public int size => nodeList.Count;

        public Graph()
        {
            nodeList = new List<Node>();
        }

        public void readFromFile(string filename)
        {
```

```
        // Baca file...
        string[] lines = File.ReadAllLines($"../../../test/{filename}");
        // Baca jumlah node yang akan dibaca
        int nodeCount = int.Parse(lines[0]);

        // Iterasi pembacaan setiap node
        for (int i = 1; i < nodeCount + 1; i++)
        {
            string line = lines[i];
            string[] lineItem = line.Split(" ");

            string name = lineItem[0];
            double x = double.Parse(lineItem[1]);
            double y = double.Parse(lineItem[2]);

            nodeList.Add(new Node(name, x, y));
        }

        // Pembacaan matrix
        int j = 0;
        for (int i = nodeCount + 1; i < lines.Length; i++)
        {
            string line = lines[i];
            string[] lineItem = line.Split(" ");

            for (int k = j; k < lineItem.Length; k++)
            {
                if (double.Parse(lineItem[k]) != 0)
                {
                    addPair(nodeList[j], nodeList[k]);
                }
            }

            j++;
        }
    }

    public void addPair(Node n1, Node n2)
    {
        addToNodeList(n1);
        addToNodeList(n2);
        addNeighbor(n1, n2);
        addNeighbor(n2, n1);
    }
```

```
public void printNodeList()
{
    Console.WriteLine("Nama nodes yang tersedia pada graf:");
    foreach (var node in nodeList)
    {
        Console.WriteLine($"- {node.name}");
    }
}

public void printGraph()
{
    foreach (Node node in nodeList)
    {
        node.neighborInfo();
    }
}

public Node findNode(String nm)
{
    foreach (var node in nodeList)
    {
        if (node.name == nm)
        {
            return node;
        }
    }

    return null;
}

public Node AStar(Node start, Node target)
{
    unvisitAll();
    List<Node> openList = new List<Node>();
    start.gn = 0;
    start.fn = start.gn + start.hn(target);
    openList.Add(start);

    while (openList.Count != 0)
    {
        openList.Sort();
        Node current = openList[0];
        if (current.name == target.name)
        {
            return current;
```

```
            }

            foreach (Edge neighbor in current.neighbors)
            {
                Node nextNode = neighbor.target;
                double newGn = current.gn + neighbor.weight;

                if (!openList.Contains(nextNode) && !nextNode.isVisited)
                {
                    nextNode.pred = current;
                    nextNode.gn = newGn;
                    nextNode.fn = nextNode.gn + nextNode.hn(target);
                    openList.Add(nextNode);
                }
                else
                {
                    if (newGn < nextNode.gn)
                    {
                        nextNode.pred = current;
                        nextNode.gn = newGn;
                        nextNode.fn = nextNode.gn + nextNode.hn(target);

                        if (nextNode.isVisited)
                        {
                            nextNode.isVisited = false ;
                            openList.Add(nextNode);
                        }
                    }
                }
            }
            openList.Remove(current);
            current.isVisited = true;
        }

        return null;
    }

    public void printPath(Node target)
    {
        List<Node> path = new List<Node>();
        double totalGn = target.gn;
        path.Add(target);
        while (target.pred != null)
        {
            path.Add(target.pred);
```

```
                target = target.pred;
            }

            path.Reverse();
            for (int i = 0; i < path.Count; i++)
            {
                Console.Write(path[i].name);
                if (i != path.Count - 1)
                {
                    Console.Write(" -> ");
                }
            }

            Console.WriteLine($"\nDengan total jarak adalah {totalGn}");
        }

        /***** PRIVATE METHODS *****/
        private void addNeighbor(Node n1, Node n2)
        {
            foreach (var e in n1.neighbors)
            {
                if (e.target.name == n2.name)
                {
                    return;
                }
            }

            n1.addEdge(n2,
n1.coordinate.getEucledianDistance(n2.coordinate));
        }

        private void addToNodeList(Node n)
        {
            if (!nodeList.Contains(n))
            {
                nodeList.Add(n);
            }
        }

        private void unvisitAll()
        {
            foreach (var node in nodeList)
            {
                node.isVisited = false;
            }
```

```
        }
    }
}
```

Program.cs

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

namespace AStarStima
{
    class Program
    {
        static void Main(string[] args)
        {
            var g = new Graph();

            bool invalidFileName = true;
            while (invalidFileName)
            {
                Console.Write("Silakan masukkan nama file: ");
                string input = Console.ReadLine();
                try
                {
                    g.readFromFile($"{input}");
                    invalidFileName = false; ;
                }
                catch (Exception)
                {
                    Console.WriteLine("Nama file tidak ditemukan!");
                }
            }

            g.printNodeList();
            Node start = null;
            Node target = null;

            while (start == null)
            {
                Console.Write("Silakan masukkan start node: ");
                string input = Console.ReadLine();
                start = g.findNode(input);
                if (start == null)
                {
                    Console.WriteLine("Input invalid!");
```

```
                }
            }
            while (target == null)
            {
                Console.Write("Silakan masukkan target node: ");
                string input = Console.ReadLine();
                target = g.findNode(input);
                if (target == null)
                {
                    Console.WriteLine("Input invalid!");
                }
            }

            g.printGraph();

            Node path = g.AStar(start, target);
            if (path != null)
            {
                g.printPath(path);
            }
            else
            {
                Console.WriteLine("Ga ketemu");
            }
        }
    }
}
```

II. Graf Input

alunalun.txt

```
alunalun.txt
10
A -6.916382640280818 107.59821645430976
B -6.920089090766057 107.5984310310339
C -6.9268202712459574 107.5999759834478
D -6.915871403448246 107.604482094655
E -6.920898541598684 107.6041816872412
F -6.926990679631887 107.60370961844805
Alunalun -6.92123936258662 107.60770074551729
G -6.9171068915130745 107.60924569793121
H -6.921707991043214 107.6119064493107
I -6.931336072664229 107.61229268741417
0 1 0 1 0 0 0 0 0 0
1 0 1 0 1 0 0 0 0 0
0 1 0 0 0 1 0 0 0 0
1 0 0 0 1 0 0 1 0 0
0 1 0 1 0 1 1 0 0 0
0 0 1 0 1 0 0 0 0 1
0 0 0 0 1 0 0 0 1 0
0 0 0 1 0 0 0 0 1 0
0 0 0 0 0 0 1 1 0 1
0 0 0 0 0 1 0 0 1 0
```

bintaro.txt

```
bintaro.txt
13
1 -6.2451305520528315 106.78875286822782
2 -6.2465684548354234 106.79136069567708
3 -6.246583916134292 106.79192062681287
4 -6.243914258430079 106.79170287581563
5 -6.244357484396679 106.79018380338242
6 -6.244625481310769 106.78964461043682
7 -6.244878016738897 106.78920910844232
8 -6.246104615657203 106.7913140347491
9 -6.245635622292481 106.79128811201133
10 -6.245589238310509 106.791837674052
11 -6.245393750344293 106.79010575095998
12 -6.245254172270691 106.79182922350667
13 -6.244703936570421 106.79178387323083
0 1 0 0 0 0 1 0 0 0 0 0 0
1 0 1 0 0 0 0 1 0 0 0 0 0
0 1 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 1
0 0 0 1 0 1 0 0 0 0 1 0 0
0 0 0 0 1 0 1 0 0 0 1 0 0
1 0 0 0 0 1 0 1 0 0 0 0 0
0 1 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 1 0 0 0 0 0 0 1 0 0 1 0
0 0 0 0 1 1 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 1
0 0 0 1 0 0 0 0 0 0 0 1 0
```

buahbatu.txt

buahbatu.txt

```
9
A -6.948359699772665 107.6341083367845
B -6.954199889026712 107.63883609794311
C -6.961969313954817 107.6385734445454
D -6.940277175337379 107.64230312279278
E -6.942832310975645 107.64219806143367
F -6.945387432738373 107.64188287735645
G -6.9540434563321005 107.64035948764977
H -6.944396672885888 107.65154852239186
I -6.954929907579369 107.64750366006726
0 1 0 0 0 1 0 0 0
1 0 1 0 0 0 1 0 0
0 1 0 0 0 0 0 0 0
0 0 0 0 1 0 0 1 0
0 0 0 1 0 1 0 1 0
1 0 0 0 1 0 1 0 0
0 1 0 0 0 1 0 0 1
0 0 0 1 1 0 0 0 1
0 0 0 0 0 0 1 1 0
```

romania.txt

romania.txt

```
10
Arad 0 0
Zerind 1 2
Oraclea 2 4
Sibiu 5 -1
Timisosso 0 -3
Fagaras 8 -1
Rimnicu 6 -3
Pitesti 8 -5
Cralova 8 -8
Bucharest 11 -10
0 1 0 1 1 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0
1 0 1 0 0 1 1 0 0 0
1 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 1
0 0 0 1 0 0 0 1 1 0
0 0 0 0 0 0 1 0 1 1
0 0 0 0 0 0 1 1 0 0
0 0 0 0 0 1 0 1 0 0
```

itb.txt

itb.txt

```
13
A -6.893220 107.610454
B -6.892613 107.610428
C -6.892625 107.608842
D -6.891049 107.608705
E -6.891012 107.610386
F -6.890995 107.611570
G -6.892568 107.611680
H -6.889913 107.610374
I -6.889943 107.609006
J -6.889898 107.611563
K -6.888721 107.609086
L -6.888697 107.610370
M -6.888690 107.611537
0 1 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 1 0 1 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0 0 0 0
0 0 1 0 1 0 0 0 1 0 0 0 0
0 1 0 1 0 1 0 1 0 0 0 0 0
0 0 0 0 1 0 1 0 0 1 0 0 0
0 1 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 1 1 0 1 0
0 0 0 1 0 0 0 1 0 0 1 0 0
0 0 0 0 0 1 0 1 0 0 0 0 1
0 0 0 0 0 0 0 0 1 0 0 1 0
0 0 0 0 0 0 0 1 0 0 1 0 1
0 0 0 0 0 0 0 0 0 1 0 1 0
```

husein.txt

```
10
A -6.911183909954581 107.56953712792068
B -6.9172024956664275 107.57415630995298
C -6.929812616947645 107.57610502737286
D -6.913834963108213 107.57783722063498
E -6.917274145034489 107.57682677456542
F -6.918922077499522 107.58649818694556
G -6.926373526388681 107.58555991559524
H -6.916342702401879 107.59826266618408
I -6.920283408675999 107.59847919034185
J -6.926875061990689 107.5994896364114
0 1 0 1 0 0 0 0 0 0
1 0 1 0 1 0 0 0 0 0
0 1 0 0 0 0 1 0 0 0
1 0 0 0 1 0 0 1 0 0
0 1 0 1 0 1 0 0 0 0
0 0 0 0 1 0 1 0 1 0
0 0 1 0 0 1 0 0 0 1
0 0 0 1 0 0 0 0 1 0
0 0 0 0 0 1 0 1 0 1
0 0 0 0 0 0 1 0 1 0
```

III. Screenshot

Alun-alun

```
Silakan masukkan nama file: alunalun.txt
Nama nodes yang tersedia pada graf:
- A
- B
- C
- D
- E
- F
- Alunalun
- G
- H
- I
Silakan masukkan start node: A
Silakan masukkan target node: Alunalun
A -> B -> E -> Alunalun
Dengan total jarak adalah 0.01305552566176268
```

ITB

```
Silakan masukkan nama file: itb.txt
Nama nodes yang tersedia pada graf:
- A
- B
- C
- D
- E
- F
- G
- H
- I
- J
- K
- L
- M
Silakan masukkan start node: A
Silakan masukkan target node: K
A -> B -> E -> H -> L -> K
Dengan total jarak adalah 0.005808403763217684
```

Buah batu

```
Nama nodes yang tersedia pada graf:
- A
- B
- C
- D
- E
- F
- G
- H
- I
Silakan masukkan start node: A
Silakan masukkan target node: H
A -> F -> E -> H
Dengan total jarak adalah 0.02037823897020677
```

Bintaro

```
Silakan masukkan nama file: bintaro.txt
Nama nodes yang tersedia pada graf:
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
Silakan masukkan start node: 1
Silakan masukkan target node: 10
1 -> 2 -> 3 -> 10
Dengan total jarak adalah 0.19976642145850446
```

Husein

```
Nama nodes yang tersedia pada graf:
- A
- B
- C
- D
- E
- F
- G
- H
- I
- J
Silakan masukkan start node: A
Silakan masukkan target node: J
A -> B -> E -> F -> I -> J
Dengan total jarak adalah 0.03879582135704391
```

Romania

```
Nama nodes yang tersedia pada graf:
- Arad
- Zerind
- Oraclea
- Sibiu
- Timisosso
- Fagaras
- Rimnicu
- Pitesti
- Cralova
- Bucharest
Silakan masukkan start node: Arad
Silakan masukkan target node: Bucharest
Arad -> Sibiu -> Rimnicu -> Pitesti -> Bucharest
Dengan total jarak adalah 15.994466510684065
```

Program dapat menerima input graf	✓
Program dapat menghitung lintasan terpendek	✓
Program dapat menampilkan lintasan terpendek serta jaraknya	✓
Bonus: Program dapat menerima input peta dengan Google Map API dan menampilkan peta	

Link repository program :
https://github.com/irvinandryan/A-star-Stima