Laporan Tugas Kecil 3 Strategi Algoritma IF2211

Implementasi Algoritma A* untuk Menentukan Lintasan Terpendek

Disusun oleh:

Alvin Wilta – 13519163 – K03

Leonard Matheus – 13519215 – K04

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

A. Algoritma A*

Algoritma A* adalah sebuah algoritma pencarian *path* atau jalan terpendek dari suatu titik ke titik lain dan merupakan algoritma yang optimal di antara algoritma pencarian jalan lainnya tetapi dengan biaya waktu yang cukup mahal dengan kompleksitas waktu berupa $O(b^d)$. Algoritma ini merupakan algoritma *Informed Search* yang artinya goal state dari sebuah masalah sudah diberikan terlebih dahulu untuk membantu dalam pencarian dan diformulasikan dalam bentuk graf berbobot. Pencarian ini juga didasari oleh pendekatan heuristik sehingga optimal.

Cara kerja algoritma ini dimulai dari node awal tertentu dari sebuah grafik, dengna tujuan untuk menemukan jalur ke node tujuan tertentu yang memiliki biaya terkecil (jarak tempuh terkecil, waktu terpendek, dll.). Ini dilakukan dengan memperhitungkan setiap sisi pada simpul awal dan memperluas jalur tersebut ke sisi lainnya pada satu waktu hingga *goal state* nya terpenuhi.

Pada setiap iterasi dari loop utamanya, A* perlu menentukan jalur mana yang akan diperpanjang. Hal tersebut dilakukan dengan mempertimbangkan bobot dari suatu sisi dan perkiraan biaya yang diperlukan untuk memperluas jalur sampai ke tujuan. Secara khusus, A * memilih jalur yang diminimalkan menggunakan rumus

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

dimana n adalah node berikutnya pada jalur, g(n) adalah biaya jalur dari node awal ke n, dan h(n) adalah fungsi heuristik yang memperkirakan biaya jalur terkecil dari n ke tujuan. Algoritma ini berhenti ketika jalur yang dipilih untuk diperluas adalah jalur dari awal ke tujuan atau jika tidak ada jalur yang memenuhi syarat untuk dilanjutkan pencariannya. Sedangkan untuk h(n) merupakan fungsi yang bergantung pada permasalahan yang ingin diselesaikan, jika sebuah permasalahan tidak dapat dicari pemecahan masalahnya menggunakan pendekatan heuristic maka h(n) adalah 0.

B. Source Code Program (C#)

Graph.cs

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace ABintang
{
    class Graph
    {
        private int V; // Jumlah Node pada suatu graph

        //Struktur data Array of List
        public List<double>[] adj;

        /// <summary>
        /// Konstruktor
        /// </summary>
        public Graph(int v)
        {
            V = v;
            adj = new List<double>[v];
            for (int i = 0; i < v; ++i)
            {
                adj[i] = new List<double>();
                for(int j = 0; j < v; j++)
                {
                    adj[i].Add(0);
                }
            }
        }
```

```
/// <summary>
/// Fungsi untuk membuat graph dari input dan sort secara alphabetical agar terurut
/// </summary>
public void InputGraph(List<List<Point>> DataNode, Dictionary<int, Point> Kamus)
{
    foreach (var line in DataNode)
    {
        AddEdge(Kamus.FirstOrDefault(x => x.Value == line[0]).Key, Kamus.FirstOrDefault(x => x.Value == line[1]).Key, Kamus);
    }
}

/// <summary>
/// Fungsi untuk menambah simpul pada graf
/// </summary>
void AddEdge(int v, int w, Dictionary<int, Point> Kamus)
{
    //JANGAN LUPA
    adj[v][w] = JarakEuclidian(TranslatetoPoint(Kamus,v), TranslatetoPoint(Kamus, w)); // Add w to v's list.
    adj[w][v] = JarakEuclidian(TranslatetoPoint(Kamus, v), TranslatetoPoint(Kamus, w)); // Add w to v's list.
}
/// <summary>
/// Menghitung jarak Haversine (penghitungan anti-flat earther)
/// </summary>
public double JarakEuclidian(Point a, Point b)
{
    //double dX = Math.Abs(a.Getlat() - b.Getlat());
    //double dY = Math.Abs(a.Getlongt() - b.Getlongt());
    //return (Math.Sqrt(dX * dX + dY * dY));

    double dLat = this.toRadian(b.Getlat() - a.Getlat());
    double dLon = this.toRadian(b.Getlongt() - a.Getlongt());
    double x = Math.Sin(dLat / 2) * Math.Sin(dLat / 2) +
```

```
            Math.Cos(this.toRadian(a.Getlat())) * Math.Cos(this.toRadian(b.Getlat())) *
            Math.Sin(dLon / 2) * Math.Sin(dLon / 2);
        double c = 2 * Math.Asin(Math.Min(1, Math.Sqrt(x)));
        double d = 6371 * c;

        return d;
    }
    /// <summary>
    /// Konversi double ke Radian
    /// </summary>
    private double toRadian(double val)
    {
        return (Math.PI / 180) * val;
    }
    /// <summary>
    /// Konversi Integer ke Pointnya sesuai Kamus
    /// </summary>
    public Point TranslatetoPoint(Dictionary<int, Point> kamus, int x)
    {
        return kamus.ElementAt(x).Value;
    }
    /// <summary>
    /// Konversi Point ke Integernya sesuai Kamus
    /// </summary>
    public int TranslatetoInt(Dictionary<int, Point> kamus, Point y)
    {
        return kamus.FirstOrDefault(x => x.Value == y).Key;
    }
    public Point TranslatetoName(Dictionary<int, Point> kamus, string name)
    {
        Point p = new Point();
        foreach (var entry in kamus)
        {
```

```
            if (entry.Value.Getname()==name)
            {
                p = entry.Value;
                return p;
            }
        }
        return p;
    }

    /// <summary>
    /// Algoritma utama A* shortest path
    /// </summary>
    public List<Point> ABintangShortestPath(Dictionary<int, Point> kamus, Point start, Point target)
    {
        Node start_node = new Node(9999, TranslatetoInt(kamus, start));
        Node end_node = new Node(9999, TranslatetoInt(kamus, target));
        //Inisialisasi Node Open dan Close
        List<Node> open = new List<Node>();
        List<Node> close = new List<Node>();
        List<Point> path2 = new List<Point>();
        open.Add(start_node);
        while (open.Count > 0)
        {
            Node current_node = open[0];
            foreach(Node n in open)
            {
                if (n.Getfn() < current_node.Getfn())
                {
                    current_node = n;
                }
            }

            close.Add(current_node);
```

```
            open.Remove(current_node);

            //Kalau targetnya ketemu
            if (current_node == end_node)
            {
                List<int> path = new List<int>();
                int current = current_node.Getposition();
                while (current != 9999)
                {
                    path.Add(current);
                    current = current_node.Getparent();
                    foreach (Node x in close)
                    {
                        //Console.WriteLine(x.Getposition());
                        if (x.Getposition() == current_node.Getparent() && current != 9999)
                        {
                            current_node = x;
                        }
                    }
                }
                Console.WriteLine("length path :" + path.Count);
                path.Reverse();
                for(int i=0; i < path.Count; i++)
                {
                     path2.Add(TranslatetoPoint(kamus, path[i]));
                }
                return path2;
            }

            List<Node> children = new List<Node>();
            for (int i = 0; i < V; i++)
            {
                if(adj[current_node.Getposition()][i] != 0){
```

```
                    Node new_node = new Node(current_node.Getposition(), i);
                    children.Add(new_node);
                }
            }

            foreach (var child in children)
            {
                if (close.Any(x => x.Getposition() == child.Getposition()))
                    continue;

                child.Setgn(adj[current_node.Getposition()][child.Getposition()]);
                child.Sethn(JarakEuclidian(TranslatetoPoint(kamus,child.Getposition()), target));
                child.Setfn(child.Getgn() + child.Gethn());

                //Apabila simpul anak ternyata sudah ada di list Open dan g(n) x lebih besar dari g(n) simpul anaknya
                if (open.Any(x => x.Getposition() == child.Getposition() && x.Getgn() > child.Getgn()))
                    continue;

                open.Add(child);
            }
        }
        //Tidak ketemu target
        return path2;
    }

    /// <summary>
    /// Algoritma untuk menghitung jarak dari simpul awal ke simpul tujuan A*
    /// </summary>
    public double HitungJarak(Dictionary<int,Point> kamus, List<Point> path)
    {
        double jarak = 0;
        for(int i = 0; i < path.Count-1; i++)
        {
```

```
                jarak += adj[TranslatetoInt(kamus, path[i])][TranslatetoInt(kamus, path[i+1])];
            }
            return jarak;
        }
    }
}
```

Point.cs

```
/// <summary>
/// Kelas untuk menampung nama dan posisi point
/// </summary>
class Point
{
    private double lat;
    private double longt;
    private string name;
    /// <summary>
    /// Konstruktor
    /// </summary>
    public Point()
    {
        lat = 0;
        longt = 0;
        name = "NONAME";
    }
    /// <summary>
    /// Konstruktor berparameter point
    /// </summary>
    public Point(double lat, double longt, string name)
    {
        this.lat = lat;
        this.longt = longt;
        this.name = name;
```

```
    }
    //getter
    public double Getlat(){return lat;}
    public double Getlongt(){return longt;}
    public string Getname(){return name;}
    //setter
    public void Setlat(double _lat){lat = _lat;}
    public void Setlongt(double _longt){longt = _longt;}
    public void Setname(string _name){name = _name;}
    //operator overloading
    public static bool operator == (Point p1, Point p2){return (p1.lat == p2.lat && p1.longt == p2.longt && p1.name == p2.name);}
    public static bool operator != (Point p1, Point p2){return (p1.lat!=p2.lat || p1.longt != p2.longt || p1.name != p2.name);}
    public bool Equals(Point p2){return (this.lat == p2.lat && this.longt == p2.longt && this.name == p2.name);}
}
```

Program.cs

```
static class Program
{
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new Form1());
    }
}
```

Input.cs

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

```
using System.Linq;
using System.Globalization;

namespace ABintang{
    class Input
    {
        public Dictionary<int, Point> Kamus;
        public List<List<Point>> DataNode;
        public int nodeinp;
        public List<List<String>> Data;
        public List<List<int>> hubungan;
        public List<List<string>> temp_bracket;
        /// <summary>
        /// Konstruktor berparameter filename
        /// </summary>
        public Input(string filename)
        {
            Data = FileInput(filename); //list berisi string point
            DataNode = FilePoint(Data); //list berisi point source ke point target
            Kamus = KamusData(DataNode); //Kamus berisi integer dan poinnya
            nodeinp = Kamus.Count;
        }
        public Input()
        {
            Data = new List<List<string>>();
            hubungan = new List<List<int>>();
            DataNode = new List<List<Point>>();
            nodeinp = 0;
            Kamus = new Dictionary<int, Point>();
        }
        /// <summary>
        /// Menghasilkan List of string berisi data point dalam string
        /// </summary>
        private List<List<string>> FileInput(string filename)
        {
            List<List<string>> bracket = new List<List<string>>();
            temp_bracket = new List<List<string>>();
```

```
            // Read each line of the file into a string array. Each element
            // of the array is one line of the file.
            string[] lines = System.IO.File.ReadAllLines(filename);
            hubungan = new List<List<int>>();
            // Display the file contents by using a foreach loop.
            int i = 0;
            int jumlah_node = 0;
            foreach (string line in lines)
            {
                if (i != 0)
                {
                    if (i > jumlah_node)
                    {
                        List<string> line2 = line.Split(' ').ToList();
                        List<int> baris = line2.Select(s => Convert.ToInt32(s)).ToList();
                        hubungan.Add(baris);
                        i++;
                    }
                    else
                    {
                        List<string> line2 = line.Split(' ').ToList();
                        temp_bracket.Add(line2);
                        i++;
                    }
                }
                else
                {
                    jumlah_node = Convert.ToInt32(line);
                    i++;
                }
                // Use a tab to indent each line of the file.
            }
            for (int a = 0; a < hubungan.Count; a++)
            {
                for (int b = 0; b < a; b++)
                {
                    if (hubungan[a][b] == 1)
```

```
                    {
                        var row = new List<string>();
                        row.Add(temp_bracket[b][0]);
                        row.Add(temp_bracket[b][1]);
                        row.Add(temp_bracket[b][2]);
                        row.Add(temp_bracket[a][0]);
                        row.Add(temp_bracket[a][1]);
                        row.Add(temp_bracket[a][2]);
                        bracket.Add(row);
                    }
                }
            }
            return bracket;
        }
        /// <summary>
        /// Menghasilkan List of Point berisi data point dalam Kelas Point
        /// </summary>
        private List<List<Point>> FilePoint(List<List<string>> fileinput)
        {
            List<List<Point>> bracket = new List<List<Point>>();
            foreach (var line in fileinput)
            {
                List<Point> listofpoint = new List<Point>();
                Point source = new Point(Convert.ToDouble(line[0], CultureInfo.InvariantCulture), Convert.ToDouble(line[1],
CultureInfo.InvariantCulture), line[2]);
                Point target = new Point(Convert.ToDouble(line[3], CultureInfo.InvariantCulture), Convert.ToDouble(line[4],
CultureInfo.InvariantCulture), line[5]);
                listofpoint.Add(source);
                listofpoint.Add(target);
                bracket.Add(listofpoint);
            }
            return bracket;
        }
        /// <summary>
        /// Mengecek apakah terdapat suatu Point di dalam list of Point
        /// </summary>
        private bool Contains(List<Point> bahanbaku, Point line)
```

```
        {
            foreach (var x in bahanbaku)
            {
                if (x == line)
                {
                    return true;
                }
            }
            return false;
        }
        /// <summary>
        /// Menghasilkan Map berisi korelasi antara integer dan Point
        /// </summary>
        private Dictionary<int, Point> KamusData(List<List<Point>> bracket)
        {
            List<Point> bahanbaku = new List<Point>();
            foreach (var line in bracket)
            {
                if (!Contains(bahanbaku, line[0]))
                {
                    //Cek apakah elemen ke 0 dari tiap line sudah ada di bahanbaku
                    bahanbaku.Add(line[0]);
                }
                if (!Contains(bahanbaku, line[1]))
                {
                    //Cek apakah elemen ke 1 dari tiap line sudah ada di bahanbaku
                    bahanbaku.Add(line[1]);
                }
            }
            //Himpunan bahanbaku sudah siap
            //Masukkan jumlah node ke variabel static global node

            /*Tahap Pembuatan Dictionary*/
            Dictionary<int, Point> kamus = new Dictionary<int, Point>();
            int i = 0;
            foreach (var elemen in bahanbaku)
            {
```

```
            kamus.Add(i, elemen);
            i++;
        }
        return kamus;
    }

    public Point GetMiddleCoords()
    {
        Double maxlat = Kamus.First().Value.Getlat();
        Double maxlongt = Kamus.First().Value.Getlongt();
        Double minlat = Kamus.First().Value.Getlat();
        Double minlongt = Kamus.First().Value.Getlongt();
        foreach (var entry in Kamus)
        {
            if (entry.Value.Getlat() > maxlat) { maxlat = entry.Value.Getlat(); }
            if (entry.Value.Getlongt() > maxlongt) { maxlongt = entry.Value.Getlongt(); }
            if (entry.Value.Getlat() < minlat) { minlat = entry.Value.Getlat(); }
            if (entry.Value.Getlongt() < minlongt) { minlongt = entry.Value.Getlongt(); }
        }
        Double lat = minlat + (maxlat - minlat) / 2;
        Double longt = minlongt + (maxlongt - minlongt) / 2;
        Point temp = new Point(lat, longt, "COORDS");
        return temp;
    }

    /// <summary>
    /// Check if there is duplicate of name in Kamus
    /// </summary>
    /// <param name="name"></param>
    /// <returns> True if exists duplicate, else false </returns>
    public Boolean CheckDuplName(string name)
    {
        if (Kamus.Count() > 0)
        {
            foreach (var entry in Kamus)
            {
                if (entry.Value.Getname() == name) { return true; }
```

```
            }
        }
        return false;
    }

    public void AddKamusData(double lat, double lng, string name)
    {
        Point temp = new Point(lat, lng, name);
        Kamus.Add(nodeinp, temp);
        if (nodeinp == 0)
        {
            List<int> temp2 = new List<int>();
            temp2.Add(0);
            hubungan.Add(temp2);
        }
        else //kalau node > 0
        {
            List<int> temp2 = new List<int>();
            for (int i = 0; i < nodeinp; i++)
            {
                hubungan[i].Add(0);
                temp2.Add(0);
            }
            temp2.Add(0);
            hubungan.Add(temp2);
        }
        nodeinp++;
    }
    /// <summary>
    /// Membersihkan class input
    /// </summary>
    public void ClearInputClass()
    {
        Kamus.Clear();
        DataNode.Clear();
        nodeinp = 0;
        Data.Clear();
```

```
            hubungan.Clear();
        }
        /// <summary>
        /// Print adjacency Matrix
        /// </summary>
        /// <returns> string with newline, contents of adj matrix </returns>
        public string CheckAdjMatrix()
        {
            string temp = "";
            foreach (var x in Kamus)
            {
                temp += x.Value.Getname();
                temp += " ";
            }
            temp += "\n";
            foreach (var x in hubungan)
            {
                foreach (var y in x)
                {
                    temp += y;
                    temp += " ";
                }
                temp += "\n";
            }
            return temp;
        }
        /// <summary>
        /// Mengecek apakah terdapat sisi diantara 2 node
        /// </summary>
        /// <param name="a">nama node 1</param>
        /// <param name="b">nama node 2</param>
        /// <returns>jika ada true, else false</returns>
        public Boolean IsSisiAda(string a, string b)
        {
            int n1 = Kamus.FirstOrDefault(x => x.Value.Getname() == a).Key;
            int n2 = Kamus.FirstOrDefault(x => x.Value.Getname() == b).Key;
            if (hubungan[n1][n2]==1) { return true; }
```

```
            return false;
        }
        public void AddSisi(string a, string b)
        {
            int n1 = Kamus.FirstOrDefault(x => x.Value.Getname() == a).Key;
            int n2 = Kamus.FirstOrDefault(x => x.Value.Getname() == b).Key;
            hubungan[n1][n2] = 1;
            hubungan[n2][n1] = 1;
        }
        public List<List<int>> IterasiAdjList()
        {
            List<List<int>> retval = new List<List<int>>();
            List<int> temp;
            for (int i=0; i<nodeinp; i++)
            {
                for (int j=nodeinp-1; j>i; j--)
                {
                    temp = new List<int>();
                    if (hubungan[i][j] == 1)
                    {
                        temp.Add(i);
                        temp.Add(j);
                        retval.Add(temp);
                    }
                }
            }
            return retval;
        }
    }
}
```

Node.cs

```
namespace ABintang
{
    /// <summary>
```

```
    /// Node untuk mengandung properti dari A* berupa g(n), h(n), dan f(n)
    /// </summary>
    class Node
    {
        private int parent;
        private int position;
        private double gn;
        private double hn;
        private double fn;

        /// <summary>
        /// Konstruktor
        /// </summary>
        public Node()
        {
            parent = 9999;
            position = 9999;
            gn = 0;
            hn = 0;
            fn = 0;
        }
        /// <summary>
        /// Konstruktor berparameter parent dan pointnya
        /// </summary>
        public Node(int _parent, int _position)
        {
            parent = _parent;
            position = _position;
            gn = 0;
            hn = 0;
            fn = 0;
        }
        //getter
        public int Getparent(){return parent;}
        public int Getposition(){return position;}
        public double Getgn(){return gn;}
        public double Gethn(){return hn;}
```

```
        public double Getfn(){return fn;}
        //setter
        public void Setparent(int _parent){parent = _parent;}
        public void Setposition(int _position){position = _position;}
        public void Setgn(double _gn){gn = _gn;}
        public void Sethn(double _hn){hn = _hn;}
        public void Setfn(double _fn){fn = _fn;}
        //operator overloading
        public static bool operator == (Node n1, Node n2){return (n1.position == n2.position);}
        public static bool operator !=(Node n1, Node n2){return (n1.position != n2.position);}
        public bool Equals (Node n2){return (this.position == n2.position);}
    }
}
```

GmapMarkerWithLabel.cs

```
using System.Drawing;
using GMap.NET.WindowsForms.Markers;
using GMap.NET.WindowsForms;
using System.Runtime.Serialization;
using GMap.NET;

namespace ABintang
{
    public class GmapMarkerWithLabel : GMapMarker, ISerializable
    {
        private Font font;
        private GMarkerGoogle innerMarker;

        public string Caption;

        public GmapMarkerWithLabel(PointLatLng p, string caption, GMarkerGoogleType type)
            : base(p)
```

```
        {
            font = new Font("Arial", 14);
            innerMarker = new GMarkerGoogle(p, type);

            Caption = caption;
        }

        public override void OnRender(Graphics g)
        {
            if (innerMarker != null)
            {
                innerMarker.OnRender(g);
            }

            g.DrawString(Caption, font, Brushes.Black, new PointF(0.0f, innerMarker.Size.Height));
        }

        public override void Dispose()
        {
            if (innerMarker != null)
            {
                innerMarker.Dispose();
                innerMarker = null;
            }

            base.Dispose();
        }

        #region ISerializable Members

        void ISerializable.GetObjectData(SerializationInfo info, StreamingContext context)
        {
            base.GetObjectData(info, context);
        }
```

```
        protected GmapMarkerWithLabel(SerializationInfo info, StreamingContext context)
            : base(info, context)
        {
        }

        #endregion
    }
}
```

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Globalization;
using System.Text;
using System.IO;
using System.Threading.Tasks;
using System.Windows.Forms;
using GMap.NET.WindowsForms.Markers;
using GMap.NET.WindowsForms;
using GMap.NET.MapProviders;
using GMap.NET;

namespace ABintang
{
```

```
public partial class Form1 : Form
{
    private List<PointLatLng> _points;
    //private Button currentButton;
    private Input input;
    private Graph g;
    // private Microsoft.Msagl.GraphViewerGdi.GraphRenderer renderer;
    //private Microsoft.Msagl.GraphViewerGdi.GViewer viewer;
    private string filename;
    //private string algorithm = "null";
    private List<string> nodelist1;
    private List<string> nodelist2;
    private double latitude;
    private double longitude;
    private double zoom;
    private double latitude2;
    private double longitude2;
    private double zoom2;
    //private GMapOverlay overlayTemp;
    //private GMarkerGoogle markerTemp;
    private PointLatLng tempPoint;
    public Form1()
    {
        InitializeComponent();
        _points = new List<PointLatLng>();
        map.MapProvider = GMapProviders.BingOSMap;
        map2.MapProvider = GMapProviders.BingOSMap;
        map.DragButton = MouseButtons.Left;
        map2.DragButton = MouseButtons.Left;
        map.MinZoom = 5;
        map.MaxZoom = 100;
        map2.MinZoom = 5;
        map2.MaxZoom = 100;
```

```
        nodelist1 = new List<string>();
        nodelist2 = new List<string>();
        tempPoint = new PointLatLng();
        input = new Input();
    }

    private void splitContainer1_Panel1_Paint(object sender, PaintEventArgs e)
    {

    }

    private void map_Load(object sender, EventArgs e)
    {
        map.ShowCenter = false;
    }


    private void btnGetRouteInfo_Click(object sender, EventArgs e)
    {
        if (!string.IsNullOrEmpty(comboStart.Text) && !string.IsNullOrEmpty(comboFinish.Text))
        {
            if (map.Overlays.Any(x => x.Id == "routes"))
            {
                GMapOverlay hapus = map.Overlays.First(x => x.Id == "routes");
                map.Overlays.Remove(hapus);
            }
            latitude = map.Position.Lat;
            longitude = map.Position.Lng;
            zoom = map.Zoom;

            label3.Text = "";
            AddPathSisi1();
            List<PointLatLng> points = new List<PointLatLng>();
```

```
            // buat variabel point, TranslatetoName() sama cek input

            labelErrorRoute.Text = "";
            List<Point> Solusi = g.ABintangShortestPath(input.Kamus, g.TranslatetoName(input.Kamus,
comboStart.SelectedItem.ToString()), g.TranslatetoName(input.Kamus, comboFinish.SelectedItem.ToString()));
            for (int i = 0; i < Solusi.Count; i++)
            {
                label3.Text += Solusi[i].Getname();
                points.Add(new PointLatLng(Solusi[i].Getlat(), Solusi[i].Getlongt()));
                if (i != Solusi.Count - 1)
                {
                    label3.Text += " -> ";
                }
            }
            label3.Text += "\nJarak =";
            label3.Text += Convert.ToString(g.HitungJarak(input.Kamus, Solusi));
            label3.Text += " km";
            GMapOverlay routes = new GMapOverlay("routes");
            GMapRoute route = new GMapRoute(points, "rute");
            route.Stroke = new Pen(Color.Red, 3);
            routes.Routes.Add(route);
            map.Overlays.Add(routes);

            //GMapProvider.GoogleMap.ApiKey = AppConfig.Key;
            map.DragButton = MouseButtons.Left;

            map.Position = new PointLatLng(latitude, longitude);
            map.MinZoom = 3;
            map.MaxZoom = 100;
            map.Zoom = zoom;
        }
        else
        {
```

```
            labelErrorRoute.Text = "Node pair incomplete";
        }

    }

    private void BtnInputDir_Click(object sender, EventArgs e)
    {
        if (ofd.ShowDialog() == DialogResult.OK)
        {
            filename = ofd.FileName;
            this.FileDir.Text = filename;
            // parse input
            input = new Input(filename);
            // initialize graph
            g = new Graph(input.nodeinp);
            g.InputGraph(input.DataNode, input.Kamus);
            //TODO
            //points.Add(_points[0]);
            //points.Add(_points[1]);
            map.Overlays.Clear();
            AddPathSisi1();
            foreach (var x in input.Kamus)
            {
                double lat2 = x.Value.Getlat();
                double longt2 = x.Value.Getlongt();
                PointLatLng point2 = new PointLatLng(lat2, longt2);
                GMarkerGoogle marker2 = new GMarkerGoogle(point2, GMarkerGoogleType.blue_dot);
                var addresses = GetAddress(point2);
                if (addresses != null)
                {
                    marker2.ToolTipText = x.Value.Getname() + String.Join(", ", addresses.ToArray());
                }
                else
```

```
                {
                    marker2.ToolTipText = x.Value.Getname();
                }
                //Create Overlay
                GMapOverlay markers2 = new GMapOverlay("markers");

                //Add all available markers to overlay
                markers2.Markers.Add(marker2);

                //Cover Map with Overlay
                map.Overlays.Add(markers2);
            }
            //GMapProvider.GoogleMap.ApiKey = AppConfig.Key;
            map.DragButton = MouseButtons.Left;
            //map.MapProvider = GMapProviders.BingOSMap;
            double lat = input.GetMiddleCoords().Getlat();
            double longt = input.GetMiddleCoords().Getlongt();
            map.Position = new PointLatLng(lat, longt);
            map.MinZoom = 3;
            map.MaxZoom = 100;
            map.Zoom = 16;

            //Create Combobox of nodes
            comboStart.Items.Clear();
            comboFinish.Items.Clear();
            foreach (var entry in input.Kamus)
            {
                comboStart.Items.Add(entry.Value.Getname());
                comboFinish.Items.Add(entry.Value.Getname());
            }
            //comboStart.DataSource = nodelist1;
            //comboFinish.DataSource = nodelist2;
        }
```

```
        }

        private void splitContainer1_Panel2_Paint(object sender, PaintEventArgs e)
        {

        }

        private void label3_Click(object sender, EventArgs e)
        {

        }

        private void label4_Click(object sender, EventArgs e)
        {

        }

        private void button2_Click(object sender, EventArgs e)
        {
            //map2.DragButton = MouseButtons.Left;
            labelExport.Text = "";
            if (checkInputValid(txtLatitude.Text) && checkInputValid(txtLongitude.Text))
            {
                labelError2.Text = "";
                double lat = Convert.ToDouble(txtLatitude.Text);
                double lng = Convert.ToDouble(txtLongitude.Text);
                map2.Position = new PointLatLng(lat, lng);
                map2.MinZoom = 3;
                map2.MaxZoom = 100;
                map2.Zoom = 15;
                // inisialisasi input lagi tapi input kosong
            }
            else
```

```
        {
            labelError2.Text = "Wrong input";
        }

    }

    private void label6_Click(object sender, EventArgs e)
    {

    }

    // default Position
    private void button3_Click(object sender, EventArgs e)
    {
        labelExport.Text = "";
        map2.Position = new PointLatLng(-6.891235014541753, 107.61071274138624);
        map2.Zoom = 15;
    }

    private void textBox1_TextChanged_1(object sender, EventArgs e)
    {

    }

    private void gMapControl1_Load(object sender, EventArgs e)
    {
        map2.ShowCenter = false;
    }
    private Boolean checkInputValid(String input)
    {
        Double d;
        return Double.TryParse(input, out d);
    }
```

```
private void comboStart_SelectedIndexChanged(object sender, EventArgs e)
{

}

private void comboFinish_SelectedIndexChanged_1(object sender, EventArgs e)
{

}

private List<string> GetAddress(PointLatLng point)
{
    List<Placemark> placemarks = null;
    var statusCode = GMapProviders.GoogleMap.GetPlacemarks(point, out placemarks);
    if(/*statusCode == GeoCoderStatusCode.G_GEO_SUCCESS &&*/ placemarks!= null)
    {
        List<string> addresses = new List<string>();
        foreach(var placemark in placemarks)
        {
            addresses.Add(placemark.Address);
        }
        return addresses;
    }
    return null;
}

private void map2_Click(object sender, EventArgs e)
{

}

private void map2_MouseDoubleClick(object sender, MouseEventArgs e)
```

```
        {
            if (e.Button == MouseButtons.Left)
            {
                //if (!newoverlay)
                //{

                //}

                latitude2 = map2.Position.Lat;
                longitude2 = map2.Position.Lng;
                zoom2 = map2.Zoom;
                map2.Overlays.Clear();
                AddPathSisi2();
                foreach (var x in input.Kamus)
                {
                    double lat2 = x.Value.Getlat();
                    double longt2 = x.Value.Getlongt();
                    PointLatLng point2 = new PointLatLng(lat2, longt2);
                    GMarkerGoogle marker2 = new GMarkerGoogle(point2, GMarkerGoogleType.blue_dot);
                    var addresses = GetAddress(point2);
                    if (addresses != null)
                    {
                        marker2.ToolTipText = x.Value.Getname() + String.Join(", ", addresses.ToArray());
                    }
                    else
                    {
                        marker2.ToolTipText = x.Value.Getname();
                    }
                    //Create Overlay
                    GMapOverlay markers2 = new GMapOverlay("markers");

                    //Add all available markers to overlay
                    markers2.Markers.Add(marker2);
```

```
            //Cover Map with Overlay
            map2.Overlays.Add(markers2);
        }

        var point = map2.FromLocalToLatLng(e.X, e.Y);
        tempPoint = new PointLatLng(point.Lat, point.Lng);
        GMarkerGoogle markerTemp = new GMarkerGoogle(map2.FromLocalToLatLng(e.X, e.Y), GMarkerGoogleType.red_dot);
        GMapOverlay overlayTemp = new GMapOverlay("markers");
        overlayTemp.Markers.Add(markerTemp);
        map2.Overlays.Add(overlayTemp);
        //newoverlay = true;
        //map2.DragButton = MouseButtons.Left;

        map2.Position = new PointLatLng(latitude2, longitude2);
        map2.MinZoom = 3;
        map2.MaxZoom = 100;
        map2.Zoom = zoom2;
    }
}

private void label12_Click(object sender, EventArgs e)
{

}

private void buttonExport_Click(object sender, EventArgs e)
{
    if (sfd.ShowDialog() == DialogResult.OK)
    {
        //Pass the filepath and filename to the StreamWriter Constructor
        StreamWriter sw = new StreamWriter(sfd.FileName);
        //Write a line of text
```

```
            sw.WriteLine(Convert.ToString(input.Kamus.Count));
            foreach (var a in input.Kamus)
            {
                sw.WriteLine(Convert.ToString(a.Value.Getlat() + " " + a.Value.Getlongt() + " " +
a.Value.Getname()).Replace(",", "."));

            }
            for(int x = 0; x < input.hubungan.Count; x++)
            {
                for (int y = 0; y < input.hubungan.ElementAt(x).Count; y++)
                {
                    sw.Write(Convert.ToString(input.hubungan.ElementAt(x).ElementAt(y)).Replace(",", "."));
                    if (y != input.hubungan.ElementAt(x).Count - 1)
                    {
                        sw.Write(" ");
                    }
                }
                if (x != input.hubungan.Count - 1)
                {
                    sw.WriteLine();
                }
            }
            sw.Close();
            labelExport.Text = "File created.";
            input.ClearInputClass();
            map2.Overlays.Clear();
            comboNode1.Items.Clear();
            comboNode2.Items.Clear();
        }
    }

    private void buttonAddPointManual_Click(object sender, EventArgs e)
    {
```

```
            labelExport.Text = "";
            if (!string.IsNullOrEmpty(textboxPointName.Text))
            {
                if (tempPoint == null)
                {
                    labelErrorPoint.Text = "No point selected!";
                }
                else
                {
                    if (input.CheckDuplName(textboxPointName.Text) )
                    {
                        labelErrorPoint.Text = "There are already point with that name!";
                    }
                    else
                    {
                        labelErrorPoint.Text = "";
                        input.AddKamusData(tempPoint.Lat, tempPoint.Lng, textboxPointName.Text);
                        comboNode1.Items.Add(textboxPointName.Text);
                        comboNode2.Items.Add(textboxPointName.Text);
                    }
                }
            }
            else
            {
                labelErrorPoint.Text = "Please insert a name";
            }
        }

        private void tabControl_SelectedIndexChanged(object sender, EventArgs e)
        {
            input.ClearInputClass();
            map2.Overlays.Clear();
            comboNode1.Items.Clear();
```

```
            comboNode2.Items.Clear();
        }

        private void check_Click(object sender, EventArgs e)
        {
            labelExport.Text = "";
            MessageBox.Show(input.CheckAdjMatrix());
        }

        private void checkAdjMat_Click(object sender, EventArgs e)
        {
            MessageBox.Show(input.CheckAdjMatrix());
        }

        private void buttonAddSisi_Click(object sender, EventArgs e)
        {
            labelExport.Text = "";
            if (!string.IsNullOrEmpty(comboNode1.SelectedItem.ToString()) &&
    !string.IsNullOrEmpty(comboNode2.SelectedItem.ToString()))
            {
                if (comboNode1.SelectedItem.ToString() != comboNode2.SelectedItem.ToString())
                {
                    if (!input.IsSisiAda(comboNode1.SelectedItem.ToString(), comboNode2.SelectedItem.ToString()))
                    {
                        labelErrorSisi.Text = "";
                        input.AddSisi(comboNode1.SelectedItem.ToString(), comboNode2.SelectedItem.ToString());
                        latitude2 = map2.Position.Lat;
                        longitude2 = map2.Position.Lng;
                        zoom2 = map2.Zoom;
                        map2.Overlays.Clear();
                        AddPathSisi2();
                        foreach (var x in input.Kamus)
                        {
```

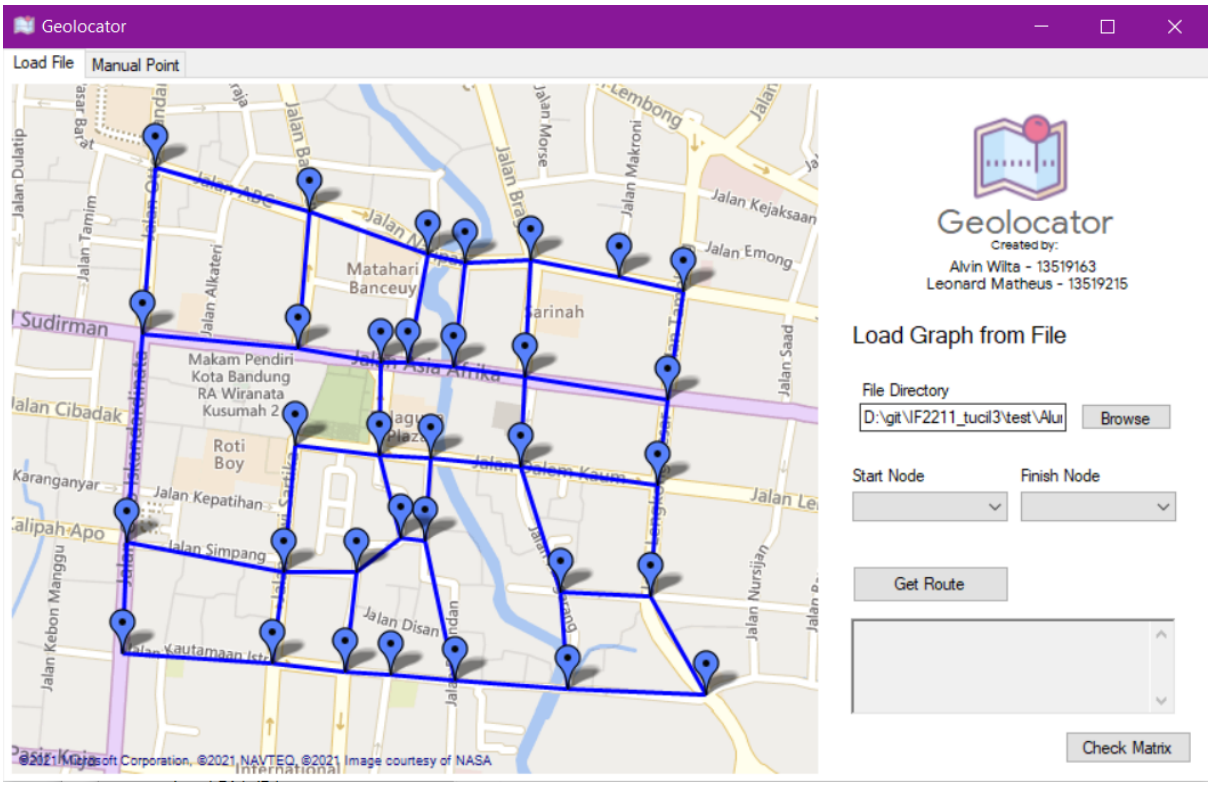
```
                    double lat2 = x.Value.Getlat();
                    double longt2 = x.Value.Getlongt();
                    PointLatLng point2 = new PointLatLng(lat2, longt2);
                    GMarkerGoogle marker2 = new GMarkerGoogle(point2, GMarkerGoogleType.blue_dot);
                    var addresses = GetAddress(point2);
                    if (addresses != null)
                    {
                        marker2.ToolTipText = x.Value.Getname() + String.Join(", ", addresses.ToArray());
                    }
                    else
                    {
                        marker2.ToolTipText = x.Value.Getname();
                    }
                    //Create Overlay
                    GMapOverlay markers2 = new GMapOverlay("markers");

                    //Add all available markers to overlay
                    markers2.Markers.Add(marker2);

                    //Cover Map with Overlay
                    map2.Overlays.Add(markers2);
                }
                map2.Position = new PointLatLng(latitude2, longitude2);
                map2.MinZoom = 3;
                map2.MaxZoom = 100;
                map2.Zoom = zoom2;
            }
            else
            {
                labelErrorSisi.Text = "Sisi sudah ada pada graf!";
            }
        }
        else
```

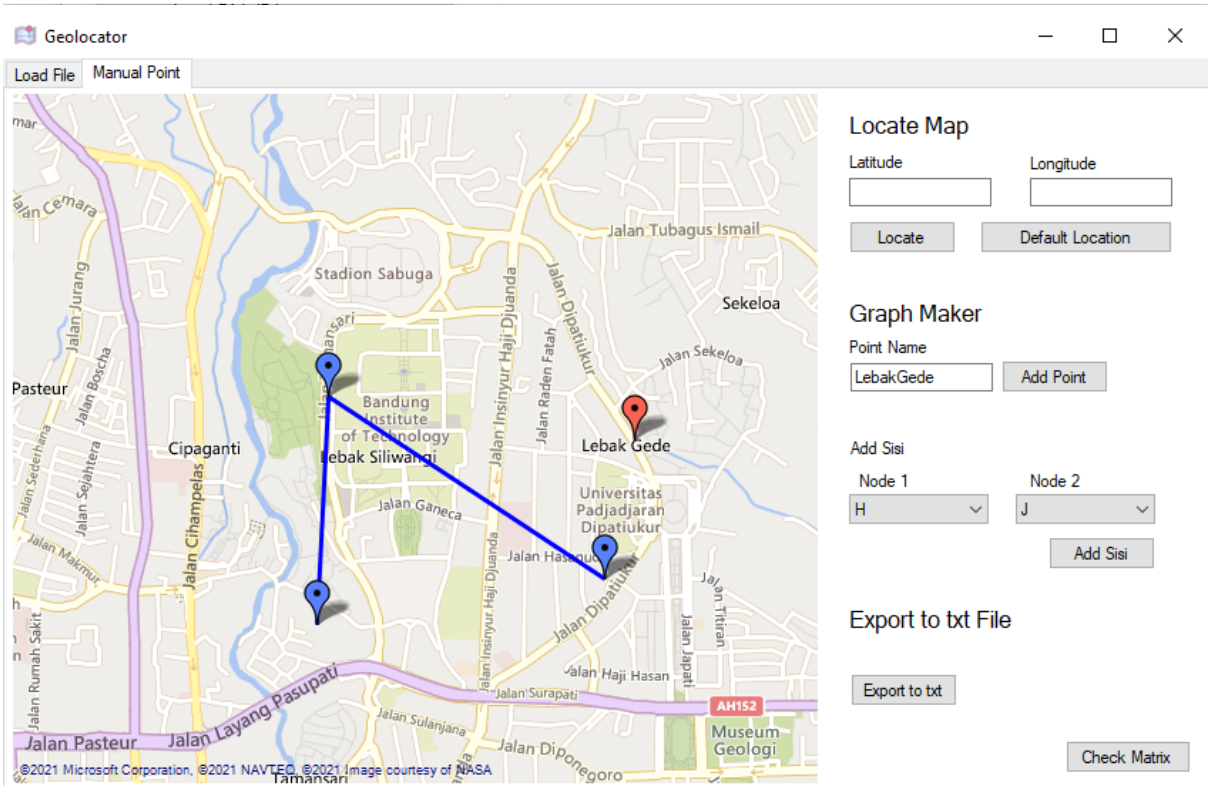
```
            {
                labelErrorSisi.Text = "Tidak bisa ke diri sendiri!";
            }
        }
        else
        {
            labelErrorSisi.Text = "Pasangan node tidak lengkap!";
        }
    }

    private void AddPathSisi1()
    {
        List<List<POINT>> solusi = input.DataNode;
        foreach (var sol in solusi)
        {
            List<PointLatLng> points = new List<PointLatLng>();
            points.Add(new PointLatLng(sol[0].Getlat(), sol[0].Getlongt()));
            points.Add(new PointLatLng(sol[1].Getlat(), sol[1].Getlongt()));
            GMapOverlay routes = new GMapOverlay("routes");
            GMapRoute route = new GMapRoute(points, "rute");
            route.Stroke = new Pen(Color.Blue, 3);
            routes.Routes.Add(route);
            map.Overlays.Add(routes);
        }

    }
    private void AddPathSisi2()
    {
        List<List<int>> solusi = input.IterasiAdjList();
        foreach (var sol in solusi)
        {
            List<PointLatLng> points = new List<PointLatLng>();
            points.Add(new PointLatLng(input.Kamus[sol[0]].Getlat(), input.Kamus[sol[0]].Getlongt()));
```

```
                points.Add(new PointLatLng(input.Kamus[sol[1]].Getlat(), input.Kamus[sol[1]].Getlongt()));
                GMapOverlay routes = new GMapOverlay("routes");
                GMapRoute route = new GMapRoute(points, "rute");
                route.Stroke = new Pen(Color.Blue, 3);
                routes.Routes.Add(route);
                map2.Overlays.Add(routes);
            }
        }
    }
}
```

C. Screenshot Input & Output

Tampilan penuh program:





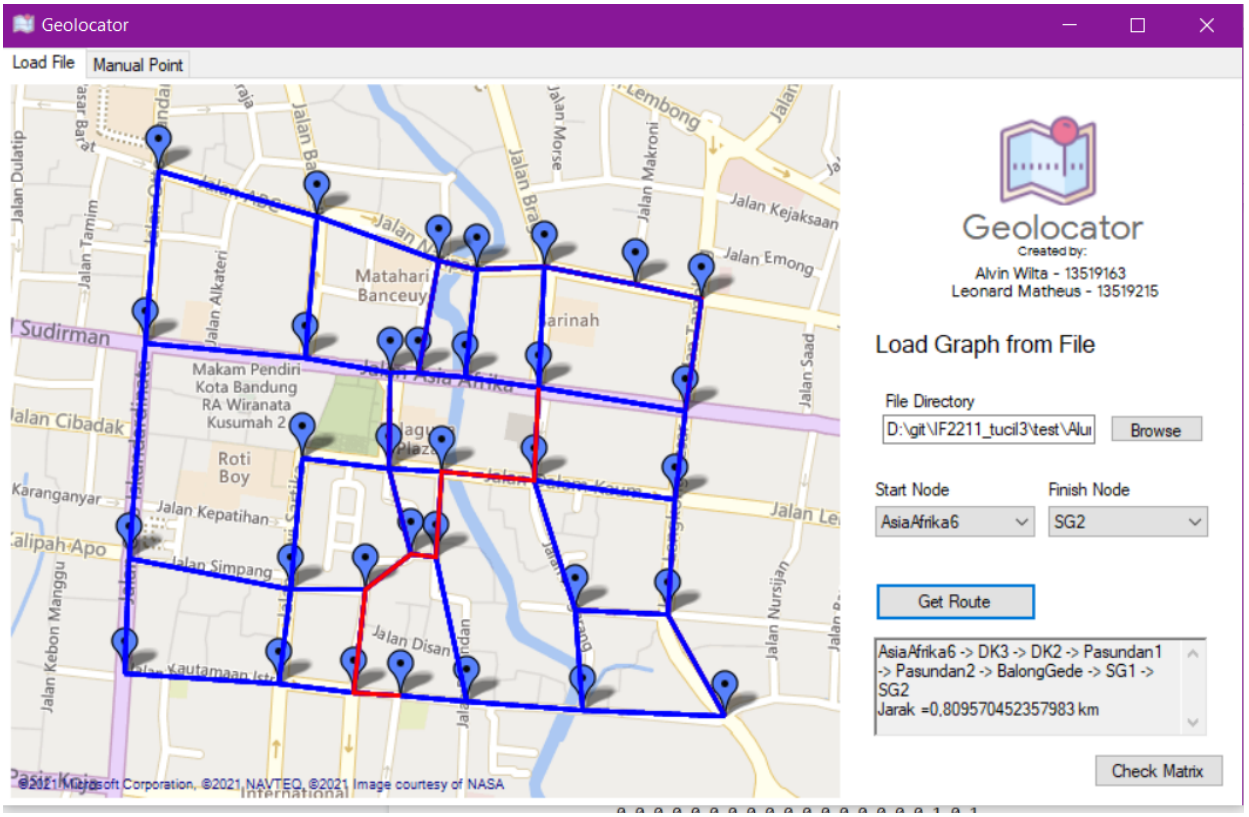
D. Output program:

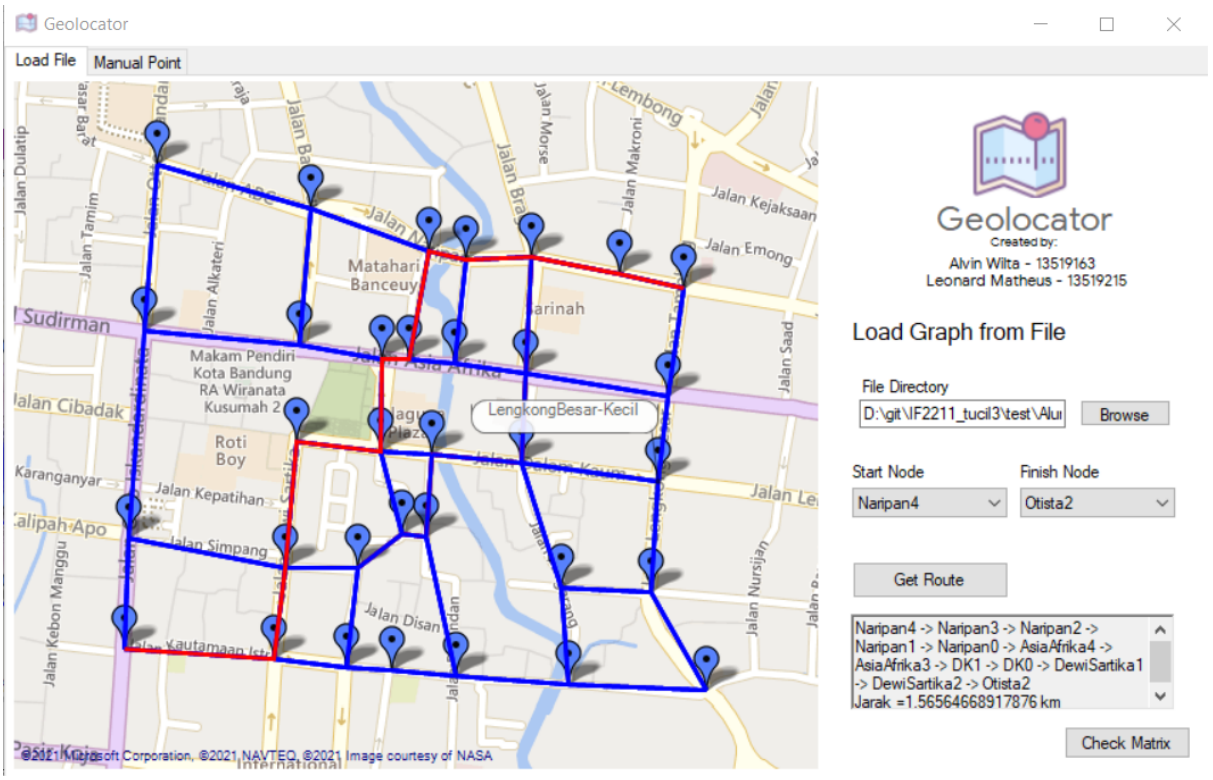
1) Masukan 1 (Alun-Alun Bandung)

a. isi file

```
20
47.0626384799543 21.90673828125 Oraclea
46.6305786805948 21.5194702148438 Zerind
46.1874367843254 21.3230895996094 Arad
45.7521933606311 21.2200927734375 Timisoara
45.7253564234102 21.8902587890625 Lugoj
44.3199181204774 23.7908935546875 Craiova
44.6334823448553 22.6483154296875 Dobreta
44.9103591745849 22.3599243164063 Mehadia
45.8019991666615 24.14794921875 Sibiu
45.8632375529644 24.9664306640625 Fagaras
45.1161766035749 24.3621826171875 Ramnicu Valcea
44.8597626880427 24.862060546875 Pitesti
44.4455460084355 26.092529296875 Bucharest
43.897892391258 25.960693359375 Giurgiu
44.7467332402468 26.630859375 Urziceni
44.0441673535722 28.6427307128906 Eforie
44.6916609229413 27.9557418823242 Hirsova
46.6381224623796 27.718505859375 Vaslui
47.1598400130443 27.564697265625 lasi
46.9323890469121 26.3732814788818 Neamt
0 1 0 0 0 0 0 0 1 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 0 0 0 0 0
0 1 0 1 0 0 0 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 0 0 0
0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 1 0 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
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 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 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 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 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 0 0 0 0 0 0 1 0
```

b. output


Geolocator
Load File
Manual Point
Geolocator
Created by:
Alvin Wilta - 13519163
Leonard Matheus - 13519215
Load Graph from File
File Directory
D:\git\IF2211_tucil3\test\Alu
Browse
Start Node
AsiaAfrika6
Finish Node
SG2
Get Route
AsiaAfrika6 -> DK3 -> DK2 -> Pasundan1 -> Pasundan2 -> BalongGede -> SG1 -> SG2
Jarak =0,809570452357983 km
Check Matrix
Matahari
Banceuy
Makam Pendiri Kota Bandung
RA Wiranata Kusumah 2
Roti Boy
Jalan Kepatihan
Jalan Simpang
Jalan Disan
Jalan Kejaksaan
Jalan Emong
Jalan Nursijan
Jalan Kebon Manggu
Karanganyar
Sudirman
Jalan Cibadak
Jalan Dulatip
Jalan Tamim
Jalan Alkateri
Jalan Makroni
Jalan Saad
Lembong
©2021 Microsoft Corporation, ©2021 NAVTEQ, ©2021 Image courtesy of NASA

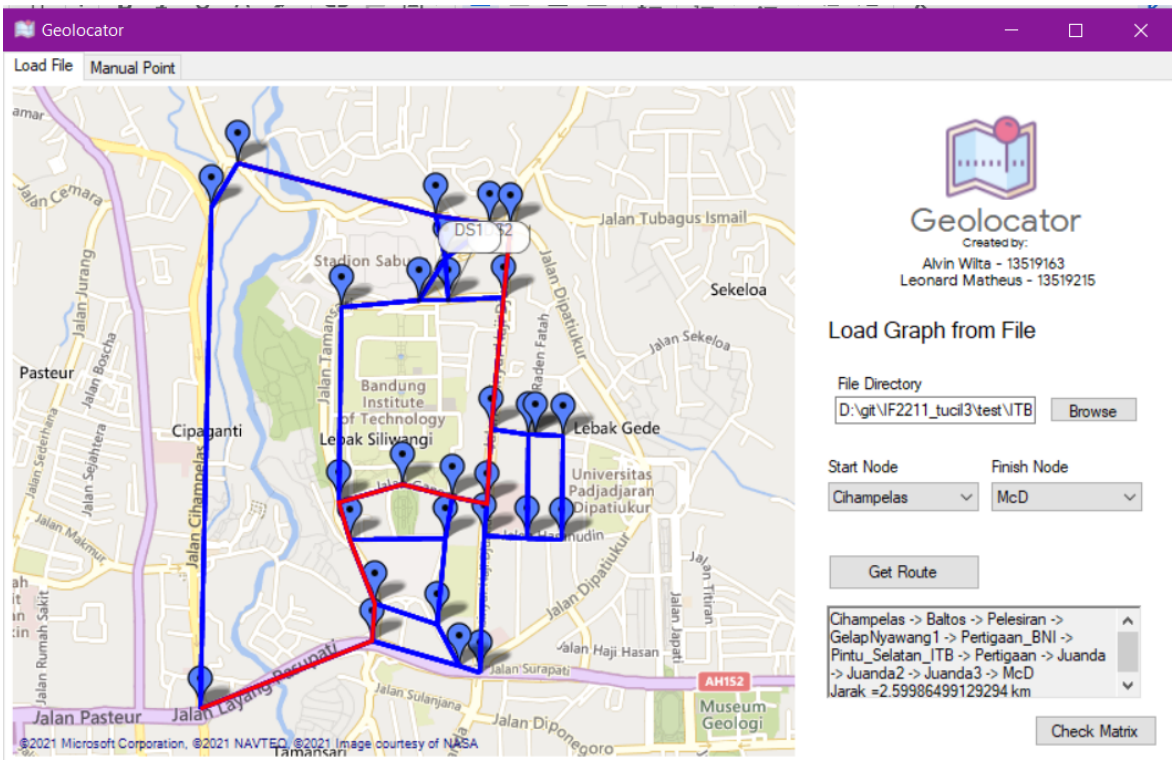

Geolocator
Load File
Manual Point
Geolocator
Created by:
Alvin Wilta - 13519163
Leonard Matheus - 13519215
Load Graph from File
File Directory
D:\git\IF2211_tucil3\test\Alu
Browse
Start Node
Naripan4
Finish Node
Otista2
Get Route
Naripan4 -> Naripan3 -> Naripan2 -> Naripan1 -> Naripan0 -> AsiaAfrika4 -> AsiaAfrika3 -> DK1 -> DK0 -> DewiSartika1 -> DewiSartika2 -> Otista2
Jarak =1,56564668917876 km
Check Matrix
LengkongBesar-Kecil
Matahari
Banceuy
Makam Pendiri Kota Bandung
RA Wiranata Kusumah 2
Roti Boy
Jalan Kepatihan
Jalan Simpang
Jalan Disan
Jalan Kejaksaan
Jalan Emong
Jalan Nursijan
Jalan Kebon Manggu
Karanganyar
Sudirman
Jalan Cibadak
Jalan Dulatip
Jalan Tamim
Jalan Alkateri
Jalan Makroni
Jalan Saad
Lembong
©2021 Microsoft Corporation, ©2021 NAVTEQ, ©2021 Image courtesy of NASA

2) Masukan 2 (ITB/Dago)

a. isi file

```
29
-6.89369661874186 107.608444690704 Pertigaan_BNI
-6.89314275224793 107.610397338867 Pintu_Selatan_ITB
-6.89354750090332 107.611920833588 Pertigaan Ciungwanara
-6.89373922382997 107.612993717194 Juanda
-6.8914491949148 107.613133192062 Juanda2
-6.88741232630974 107.613519430161 Juanda3
-6.88746558337517 107.611802816391 DS2
-6.88749753761157 107.610912322998 DS1
-6.88619806359563 107.611716985703 Separator
-6.88487728307963 107.611416578293 Siliwangi_1
-6.88513291830513 107.61311173439 Siliwangi_2
-6.88516487269865 107.613723278046 McD
-6.8876786115771 107.608487606049 Tamfest
-6.89159831341348 107.614291906357 TU_SuryaKencana
-6.89160896473301 107.614538669586 TU_Raden Fatah
-6.89167287264522 107.615364789963 Imam_Bonjol
-6.89483630352942 107.615343332291 Hasanuddin_ImamBonjol
-6.89479369853994 107.614259719849 Hasanuddin_SuryaKencana
-6.89470848854953 107.612929344177 Juanda4
-6.89485760602273 107.608766555786 GelapNyawang1
-6.89481500103518 107.611727714539 GelapNyawang2
-6.89681743130849 107.609539031982 Pelesiran
-6.89746715420608 107.611459493637 BadakSinga
-6.89893701583291 107.612800598145 Kolong_FlyOver
-6.89873464385076 107.612135410309 Kolong_FlyOver2
-6.89794645741172 107.609431743622 Baltos
-6.90005538575356 107.604131698608 Cihampelas
-6.88463229919262 107.604475021362 Cihampelas2
-6.88326890829364 107.605290412903 Cihampelas3
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 0 0 0
1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 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 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 0 0 1
0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0
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 0 0 0
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
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
0 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 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0
1 0 0 0 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 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
```

b. output

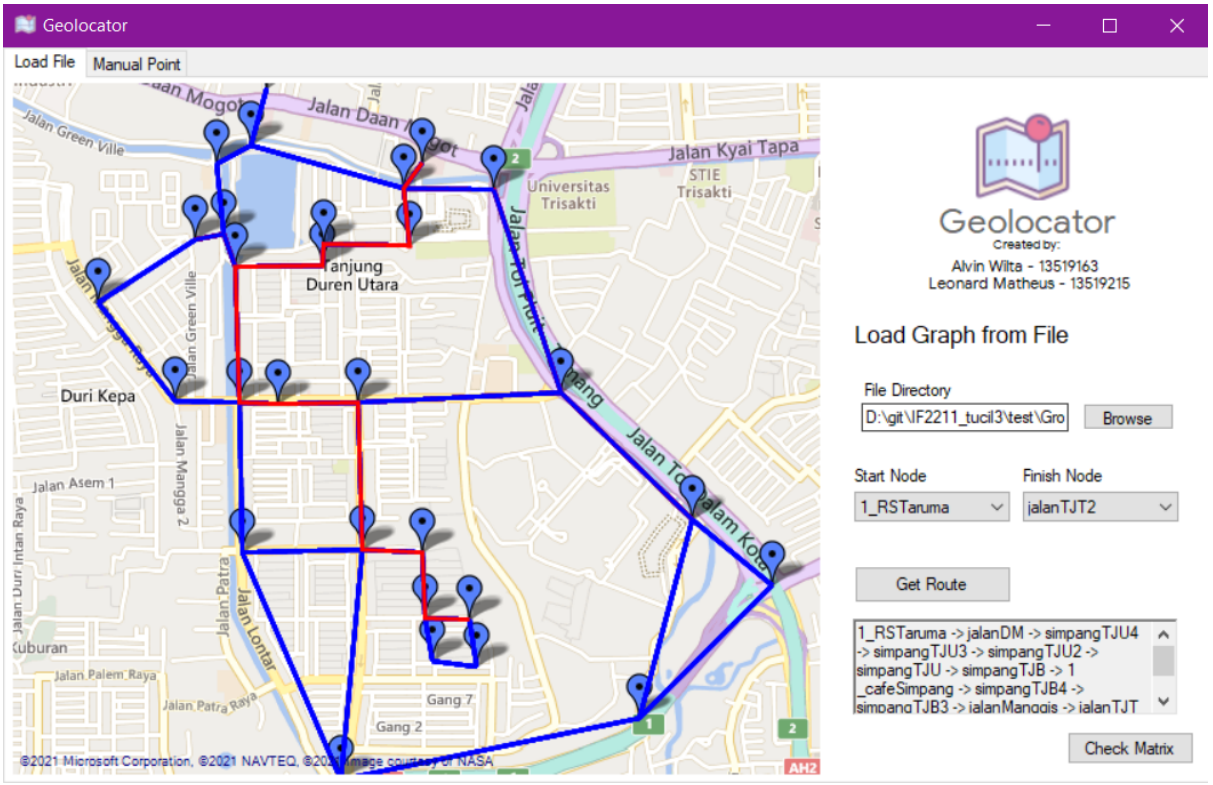

Geolocator
Load File
Manual Point
Jalan Tubagus Ismail
Sekeloa
Jalan Sekeloa
Pasteur
Cipaganti
Lebak Siliwangi
Lebak Gede
Universitas Padjadjaran Dipatiukur
Jalan Dipatiukur
Jalan Pasteur
Jalan Surapati
Jalan Haji Hasan
Museum Geologi
AH152
©2021 Microsoft Corporation, ©2021 NAVTEQ, ©2021 Image courtesy of NASA
Geolocator
Created by:
Alvin Wilta - 13519163
Leonard Matheus - 13519215
Load Graph from File
File Directory
D:\git\IF2211_tucil3\test\ITB
Browse
Start Node
Finish Node
Cihampelas
McD
Get Route
Cihampelas -> Baltos -> Pelesiran -> GelapNyawang1 -> Pertigaan_BNI -> Pintu_Selatan_ITB -> Pertigaan -> Juanda -> Juanda2 -> Juanda3 -> McD
Jarak =2.59986499129294 km
Check Matrix

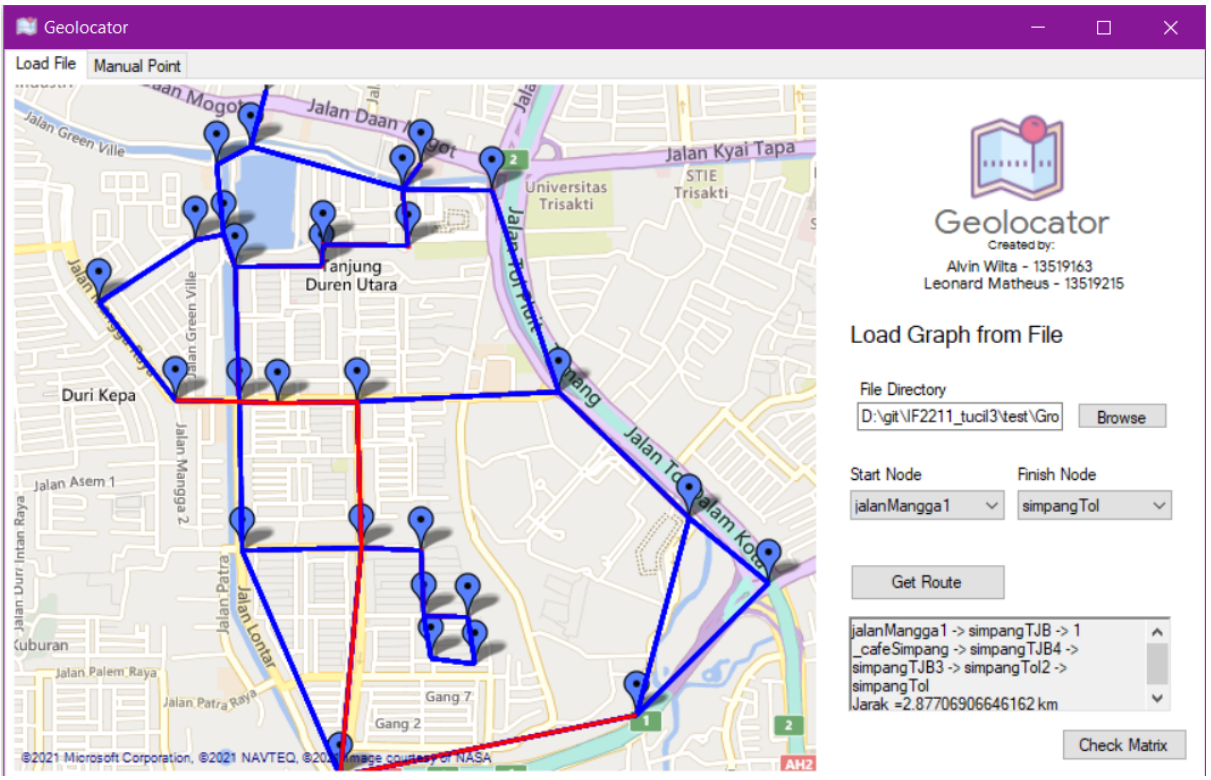
3) Masukan 3 (Grogol Petamburan)

a. isi file

```
29
-6.17089165883642 106.775736808777 jalanMangga
-6.16901432159139 106.778633594513 1_smakIPEKA
-6.17383565157741 106.778054237366 jalanMangga1
-6.17385698479842 106.779985427856 simpangTJB
-6.17835827521253 106.780071258545 simpangTJB2
-6.17825161000562 106.783654689789 simpangTJB3
-6.17385698479842 106.7835688591 simpangTJB4
-6.17735562141895 106.793578863144 1_TA
-6.1735583196259 106.789662837982 simpangTJR
-6.16749964692242 106.787602901459 simpangCL
-6.16747831344547 106.78493142128 jalanDM
-6.16620897002242 106.780339479446 jalanDM1
-6.16674230796525 106.779309511185 jalanDM2
-6.16884365423938 106.779491901398 jalanGV
-6.16447028460459 106.780779361725 1_RDAbadi
-6.18330751716816 106.792023181915 simpangTol
-6.18512080543113 106.783053874969 simpangTol2
-6.17389965123789 106.781144142151 1_cafeSimpang
-6.16668897419511 106.785500049591 1_RSTaruma
-6.17933959410746 106.796014308929 simpangTomang
-6.16980365737957 106.779835224152 simpangTJU
-6.16976099061029 106.782474517822 simpangTJU2
-6.16914232206932 106.78250670433 simpangTJU3
-6.1691636554793 106.785081624985 simpangTJU4
-6.17833694217286 106.785489320755 jalanManggis
-6.18032091118266 106.785553693771 jalanTJT
-6.18034224414234 106.786884069443 jalanTJT2
-6.18177155048168 106.787109375 1_rumahA
-6.1816008872412 106.785778999329 1_rumahB
0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 1 0 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 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 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 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 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 0 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 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0
```

b. output


Geolocator
Load File
Manual Point
Jalan Daan Mogot
Jalan Green Ville
Jalan Kyai Tapa
Universitas Trisakti
STIE Trisakti
Tanjung Duren Utara
Duri Kepa
Jalan Asem 1
Jalan Mangga 2
Jalan Patra
Jalan Lontar
Kuburan
Jalan Palem Raya
Jalan Patra Raya
Gang 7
Gang 2
©2021 Microsoft Corporation, ©2021 NAVTEQ, ©2021 Image courtesy of NASA
Geolocator
Created by:
Alvin Wilta - 13519163
Leonard Matheus - 13519215
Load Graph from File
File Directory
D:\git\IF2211_tucil3\test\Gro
Browse
Start Node
1_RSTaruma
Finish Node
jalanTJT2
Get Route
1_RSTaruma -> jalanDM -> simpangTJU4 -> simpangTJU3 -> simpangTJU2 -> simpangTJU -> simpangTJB -> 1 _cafeSimpang -> simpangTJB4 -> simpangTJB3 -> jalanMangga -> jalanTJT
Check Matrix

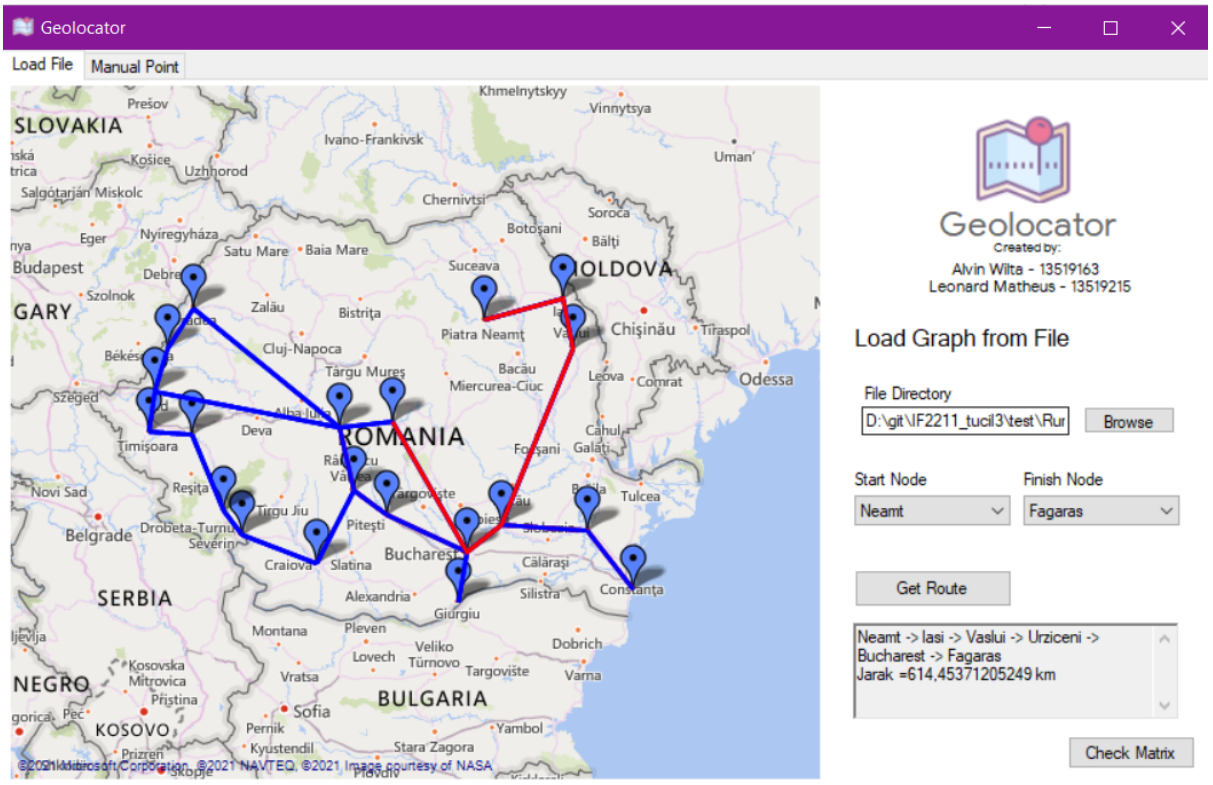

Geolocator
Load File
Manual Point
Jalan Daan Mogot
Jalan Green Ville
Jalan Kyai Tapa
Universitas Trisakti
STIE Trisakti
Tanjung Duren Utara
Duri Kepa
Jalan Asem 1
Jalan Mangga 2
Jalan Patra
Jalan Lontar
Kuburan
Jalan Palem Raya
Jalan Patra Raya
Gang 7
Gang 2
©2021 Microsoft Corporation, ©2021 NAVTEQ, ©2021 Image courtesy of NASA
Geolocator
Created by:
Alvin Wilta - 13519163
Leonard Matheus - 13519215
Load Graph from File
File Directory
D:\git\IF2211_tucil3\test\Gro
Browse
Start Node
jalanMangga1
Finish Node
simpangTol
Get Route
jalanMangga1 -> simpangTJB -> 1 _cafeSimpang -> simpangTJB4 -> simpangTJB3 -> simpangTol2 -> simpangTol
Jarak =2.87706906646162 km
Check Matrix

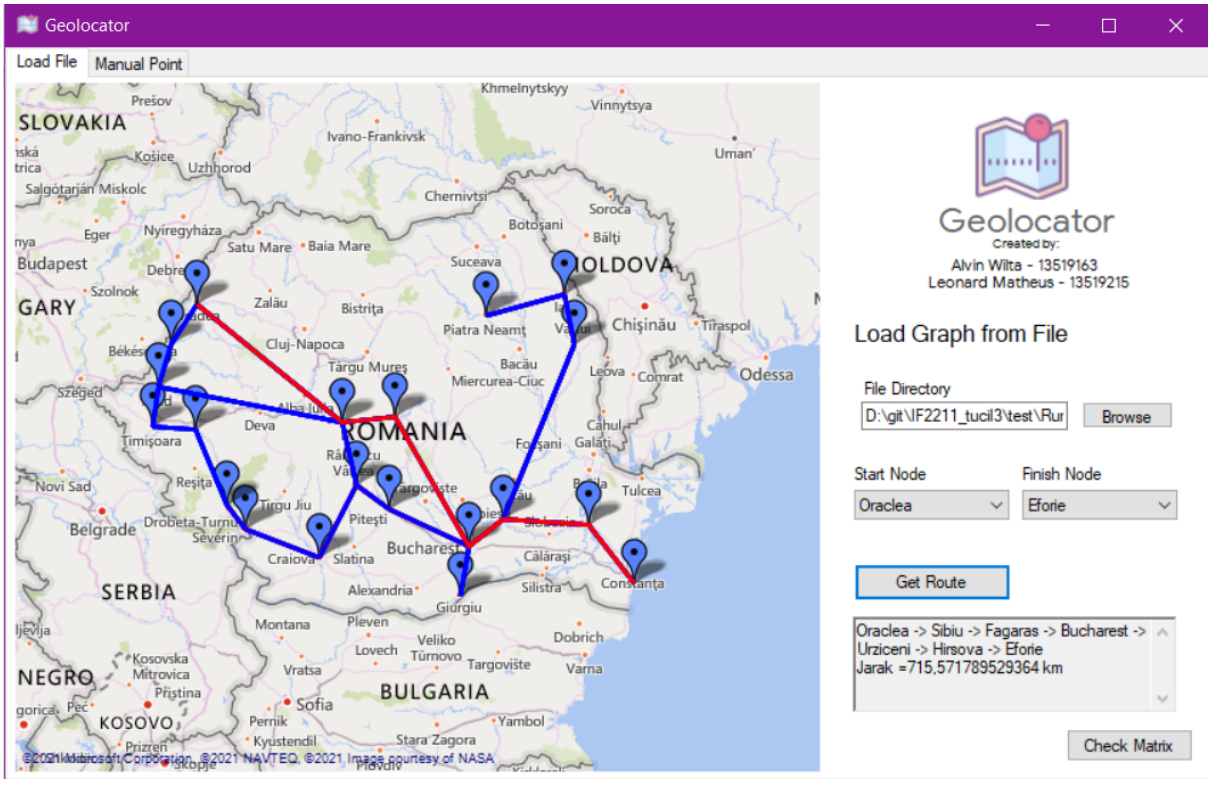
4) Masukan 4 (Romania)

a. isi file

```
20
47.0626384799543 21.90673828125 Oraclea
46.6305786805948 21.5194702148438 Zerind
46.1874367843254 21.3230895996094 Arad
45.7521933606311 21.2200927734375 Timisoara
45.7253564234102 21.8902587890625 Lugoj
44.3199181204774 23.7908935546875 Craiova
44.6334823448553 22.6483154296875 Dobreta
44.9103591745849 22.3599243164063 Mehadia
45.8019991666615 24.14794921875 Sibiu
45.8632375529644 24.9664306640625 Fagaras
45.1161766035749 24.3621826171875 Ramnicu Valcea
44.8597626880427 24.862060546875 Pitesti
44.4455460084355 26.092529296875 Bucharest
43.897892391258 25.960693359375 Giurgiu
44.7467332402468 26.630859375 Urziceni
44.0441673535722 28.6427307128906 Eforie
44.6916609229413 27.9557418823242 Hirsova
46.6381224623796 27.718505859375 Vaslui
47.1598400130443 27.564697265625 Iasi
46.9323890469121 26.3732814788818 Neamt
0 1 0 0 0 0 0 0 1 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 0 0 0 0 0
0 1 0 1 0 0 0 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 0 0 0
0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 1 0 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
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 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 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 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
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 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0
0 0 0 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 0 0 0 0 0 1 0
```

b. output


Geolocator
Load File
Manual Point
Geolocator
Created by:
Alvin Wilta - 13519163
Leonard Matheus - 13519215
Load Graph from File
File Directory
D:\git\IF2211_tucil3\test\Rur
Browse
Start Node
Finish Node
Neamt
Fagaras
Get Route
Neamt -> Iasi -> Vaslui -> Urziceni -> Bucharest -> Fagaras
Jarak =614,45371205249 km
Check Matrix

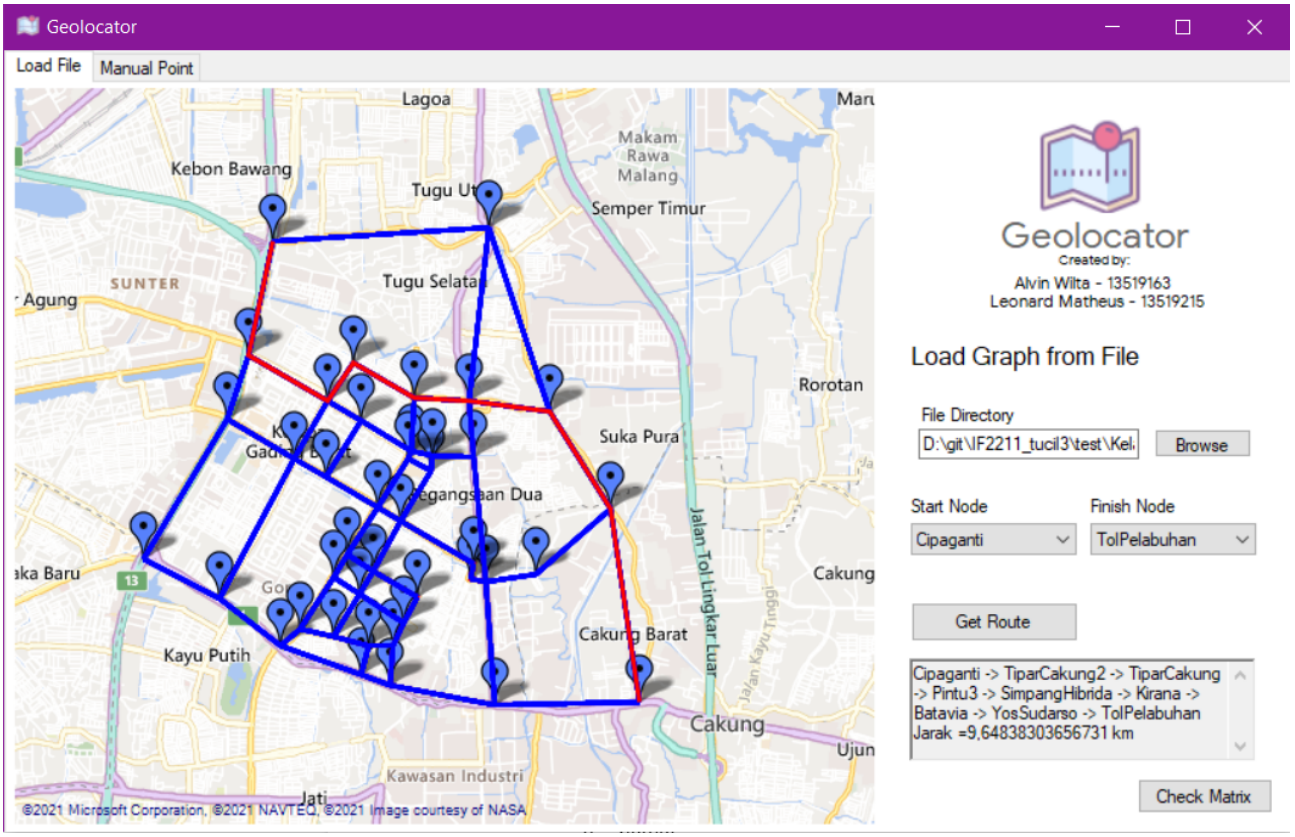

Geolocator
Load File
Manual Point
Geolocator
Created by:
Alvin Wilta - 13519163
Leonard Matheus - 13519215
Load Graph from File
File Directory
D:\git\IF2211_tucil3\test\Rur
Browse
Start Node
Finish Node
Oraclea
Eforie
Get Route
Oraclea -> Sibiu -> Fagaras -> Bucharest -> Urziceni -> Hirsova -> Eforie
Jarak =715,571789529364 km
Check Matrix

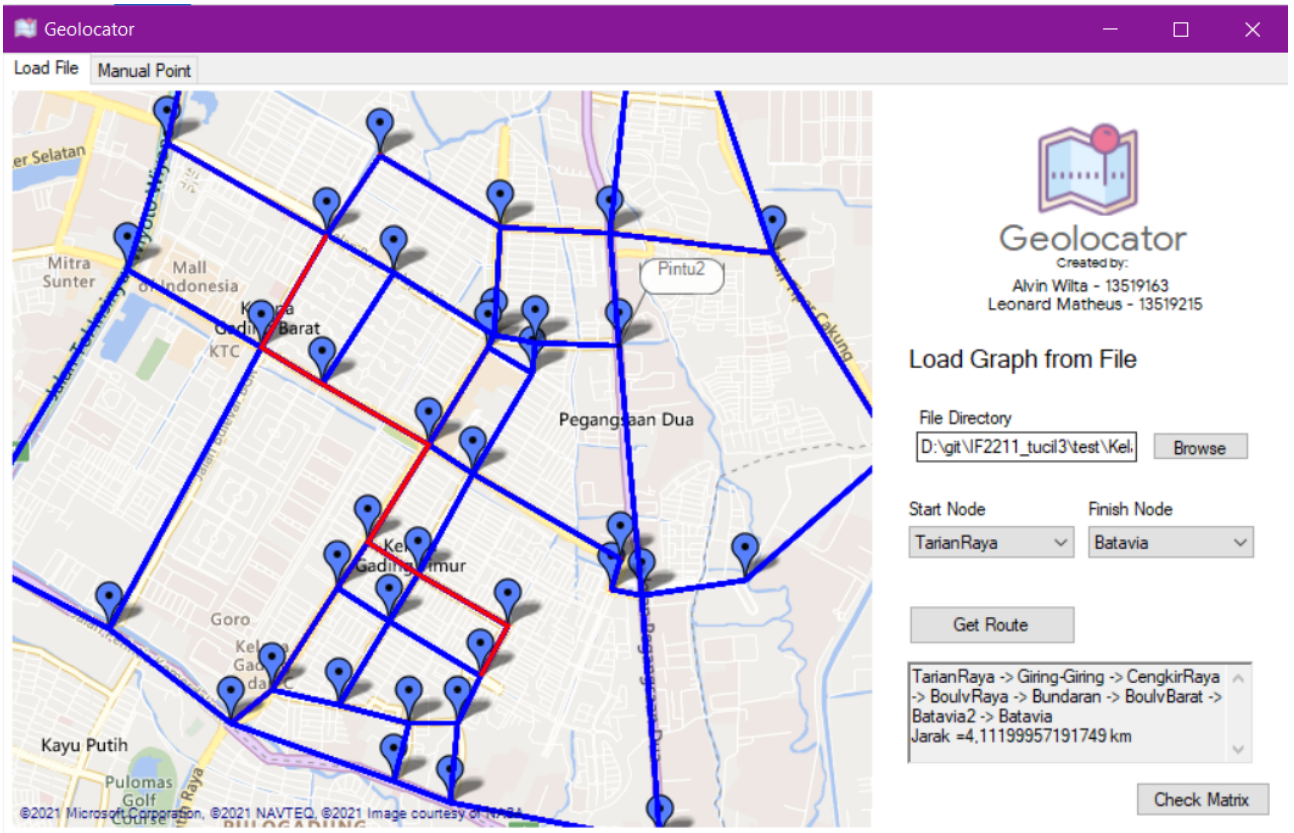
5) Masukan 5 (Kelapa Gading)

a. isi file

```
41
-6.16043821919196 106.905426979065 Bundaran
-6.15425139261342 106.909160614014 PerempatanMKG
-6.14861918439928 106.899762153625 Batavia
-6.15497674842409 106.895985603333 Batavia2
-6.1570461404409 106.89950466156 BoulvBarat
-6.15081660606629 106.903474330902 NiasRaya
-6.14823516803031 106.909482479095 SimpangHibrida
-6.14853384745236 106.915726661682 Pintu3
-6.15484874452946 106.916198730469 Pintu2
-6.16888632108254 106.917507648468 Pintu1
-6.16854498624119 106.915791034698 Kondominium
-6.16681697523444 106.916263103485 GadingPluit
-6.16203824881579 106.907980442047 BoulvTimur
-6.16767031470708 106.904890537262 CengkirRaya
-6.16587830022848 106.90203666687 BoulvRaya
-6.1760543019614 106.894354820251 PintuUtama
-6.17076365878096 106.887488365173 BellaTerra
-6.16634763793915 106.878969669342 CocaCola
-6.15064593286401 106.888453960419 SimpangMOI
-6.1434989433552 106.890749931335 YosSudarso
-6.17500897745774 106.900448799133 KelapaSawit
-6.17419831621797 106.896650791168 Kensington
-6.17030499166168 106.903259754181 KopyorRaya
-6.1684383190582 106.900341510773 LRTBoulvRaya
-6.18266753159088 106.918516159058 IGI
-6.17929692810638 106.903538703918 PerintisKemerdekaan
-6.17601163569552 106.904375553131 Inpres
-6.17603296882889 106.907143592834 BCS
-6.1804702418824 106.906714439392 Perintis2
-6.14968589508045 106.924824714661 TiparCakung
-6.18236887138978 106.934995651245 Cipaganti
-6.16069422425589 106.931691169739 TiparCakung2
-6.16803298356707 106.923322677612 SedayuCity
-6.1705716586398 106.909890174866 Giring-Giring
-6.17343032021468 106.908388137817 TarianRaya
-6.12911933650959 106.91782951355 Tugu
-6.13065545544828 106.893539428711 TolPelabuhan
-6.15644879099567 106.911327838898 GadingIndahUtara
-6.1547420745937 106.911478042603 Masjid
-6.15499808240354 106.908838748932 MKG
-6.14424564824266 106.902680397034 Kirana
0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
0 0 0 1 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 0 0 0 0 0 0 0 0 0 0 0 1
0 0 1 0 1 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 0 0 0 0 0 0 0 0 0 0
1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
1 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 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 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 0 0 0 0 0 0 0 0 0 0 0
0 0 0 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 0 0 0 0 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 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

b. output


Geolocator
Load File
Manual Point
Lagoa
Kebon Bawang
Tugu Utara
Makam Rawa Malang
Semper Timur
SUNTER
Agung
Tugu Selatan
Rorotan
Suka Pura
Pegangsaan Dua
Jalan Tol Lingkar Luar
Cakung
Cakung Barat
Kayu Putih
Kawasan Industri
Jati
©2021 Microsoft Corporation, ©2021 NAVTEQ, ©2021 Image courtesy of NASA
Geolocator
Created by:
Alvin Wilta - 13519163
Leonard Matheus - 13519215
Load Graph from File
File Directory
D:\git\IF2211_tucil3\test\Kel
Browse
Start Node
Finish Node
Cipaganti
TolPelabuhan
Get Route
Cipaganti -> TiparCakung2 -> TiparCakung -> Pintu3 -> SimpangHibrida -> Kirana -> Batavia -> YosSudarso -> TolPelabuhan
Jarak =9,64838303656731 km
Check Matrix

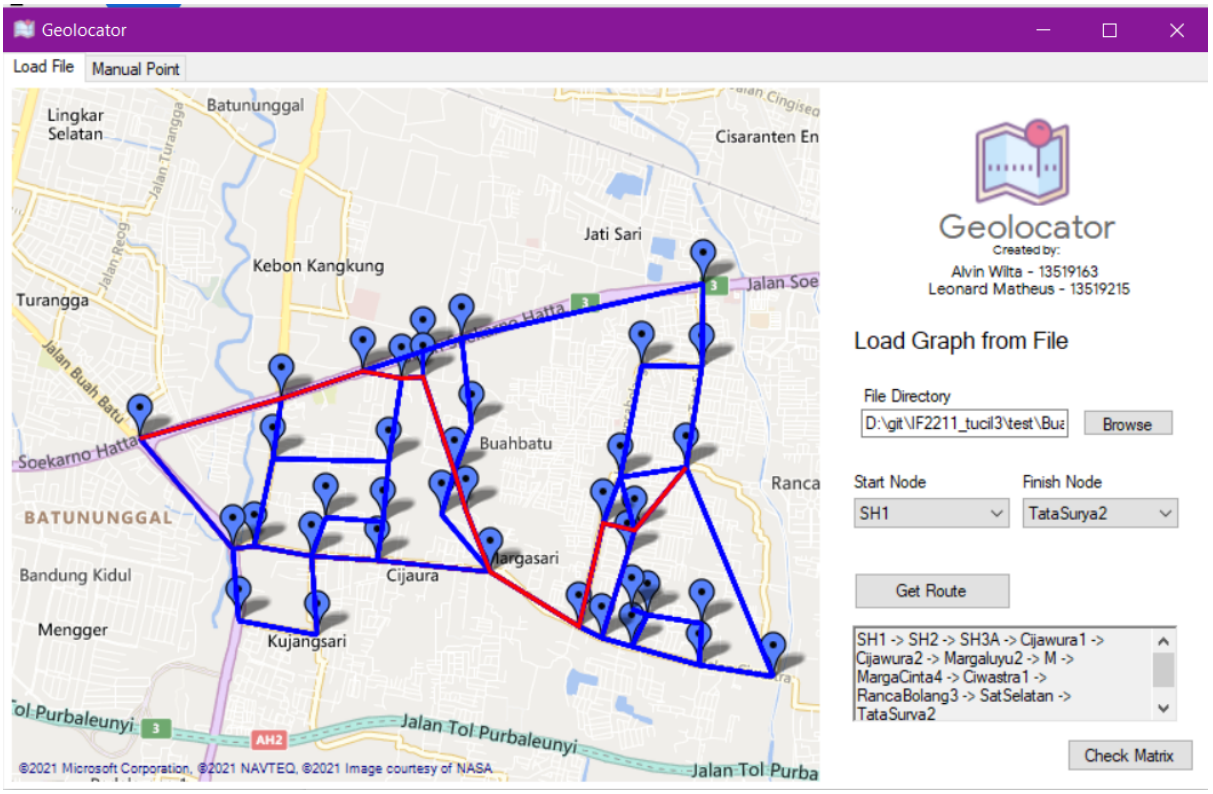

Geolocator
Load File
Manual Point
Mitra Sunter
Mall Indonesia
Pintu2
KTC
Pegangsaan Dua
Goro
Kayu Putih
Pulomas Golf
©2021 Microsoft Corporation, ©2021 NAVTEQ, ©2021 Image courtesy of NASA
Geolocator
Created by:
Alvin Wilta - 13519163
Leonard Matheus - 13519215
Load Graph from File
File Directory
D:\git\IF2211_tucil3\test\Kel
Browse
Start Node
Finish Node
TarianRaya
Batavia
Get Route
TarianRaya -> Giring-Giring -> CengkirRaya -> BoulvRaya -> Bundaran -> BoulvBarat -> Batavia2 -> Batavia
Jarak =4,11199957191749 km
Check Matrix

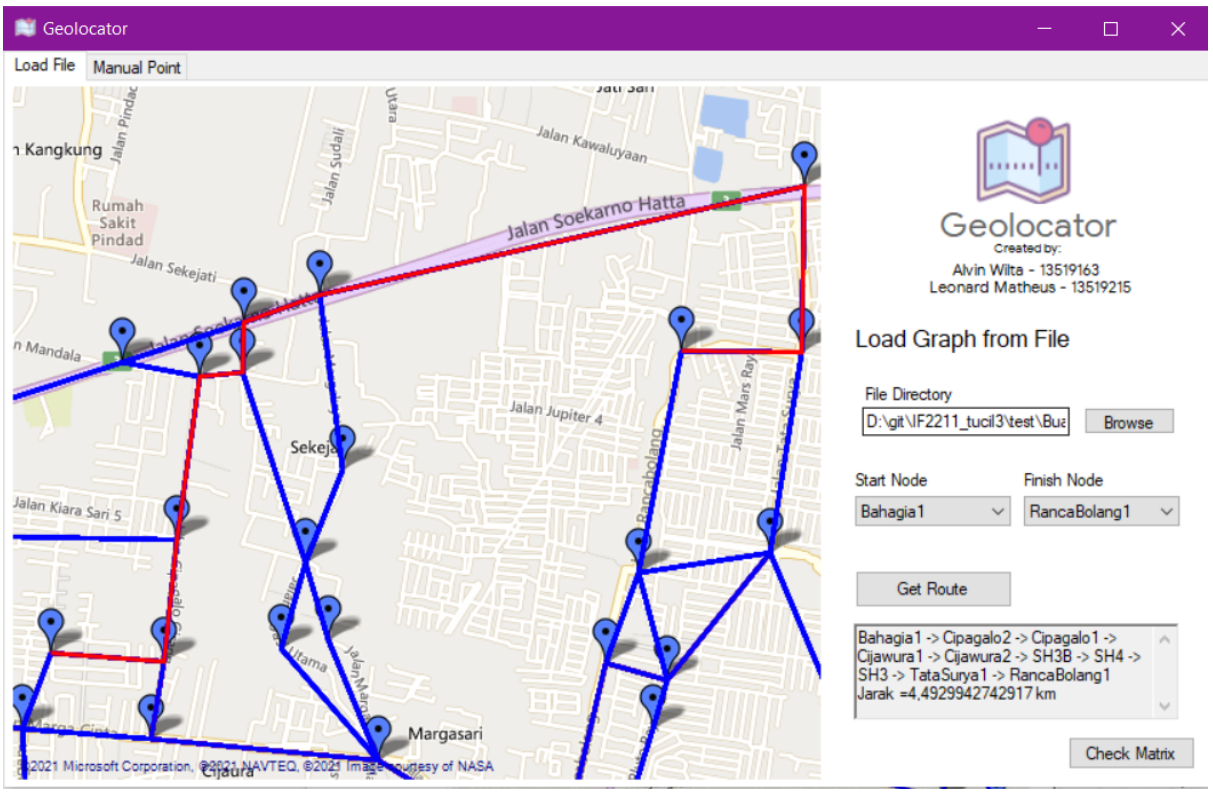
6) Masukan 6 (Buah Batu)

a) isi file

```
38
-6.95475641851452 107.64374256134 MargaCinta2
-6.9541600240662 107.640266418457 MargaCinta1
-6.95948494768901 107.644000053406 Logam
-6.95863296396762 107.639322280884 TBB2
-6.95437302217039 107.63897895813 TBB1
-6.94542701867184 107.641854286194 SH2
-6.94781263625141 107.633399963379 SH1
-6.93861090178387 107.667217254639 SH3
-6.94355259637047 107.66713142395 TataSurya1
-6.94949534137152 107.666187286377 TataSurya2
-6.95011304173966 107.662239074707 RancaBolang2
-6.94348869547855 107.663547992706 RancaBolang1
-6.95893115844601 107.659792900085 Ciwastra1
-6.96195569173516 107.671465873718 Ciwastra2
-6.94184856961591 107.652668952942 SH4
-6.94382950013515 107.646768093109 SH3A
-6.94263668275694 107.650415897369 SH3B
-6.94425550560875 107.649064064026 Cijawura1
-6.94414900427653 107.650351524353 Cijawura2
-6.94913324077832 107.648355960846 Cipagalo1
-6.95273293427701 107.648012638092 Cipagalo2
-6.95249863553914 107.644600868225 Bahagia1
-6.94900544050236 107.641382217407 IbrahimAjie
-6.94704583192068 107.653334140778 Margaluyu1
-6.94979354165059 107.6522397995 Margaluyu2
-6.95503331567954 107.647604942322 MargaCinta3
-6.95569360826046 107.654428482056 MargaCinta4
-6.95234953628149 107.651510238647 IU
-6.95209393744398 107.652947902679 M
-6.96016653355268 107.662968635559 Ciwastra1a
-6.96127410846998 107.666959762573 Ciwastra1b
-6.95878206123048 107.667152881622 HajiMukti
-6.95797267552 107.662947177887 SetiaGraha
-6.95824957078851 107.663869857788 SetiaGraha2
-6.95966599403057 107.661123275757 Ciwastra1c
-6.9547457686203 107.662678956985 Pluto1
-6.95329738075662 107.663097381592 SatSelatan
-6.95279683391252 107.661241292953 RancaBolang3
0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 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 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 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 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 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 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 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 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 1 1 0 0 0 0 0 0 0 0 0 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 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
```

b) output


Geolocator
Load File
Manual Point
Lingkar Selatan
Batununggal
Cisaranten En
Jati Sari
Kebon Kangkung
Turangga
Jalan Buah Batu
Soekarno Hatta
Buahbatu
Ranca
BATUNUNGGAL
Bandung Kidul
Cijaura
Margasari
Mengger
Kujangsari
Jalan Tol Purbaleunyi
©2021 Microsoft Corporation, ©2021 NAVTEQ, ©2021 Image courtesy of NASA
Geolocator
Created by:
Alvin Wilta - 13519163
Leonard Matheus - 13519215
Load Graph from File
File Directory
D:\git\IF2211_tucil3\test\Bua
Browse
Start Node
Finish Node
SH1
TataSurya2
Get Route
SH1 -> SH2 -> SH3A -> Cijawura1 ->
Cijawura2 -> Margaluyu2 -> M ->
MargaCinta4 -> Ciwastra1 ->
RancaBolang3 -> SatSelatan ->
TataSurva2
Check Matrix


Geolocator
Load File
Manual Point
Jalan Kawaluyaan
Jalan Soekarno Hatta
Rumah Sakit Pindad
Jalan Sekejati
Jalan Jupiter 4
Jalan Kiara Sari 5
Margasari
©2021 Microsoft Corporation, ©2021 NAVTEQ, ©2021 Image courtesy of NASA
Geolocator
Created by:
Alvin Wilta - 13519163
Leonard Matheus - 13519215
Load Graph from File
File Directory
D:\git\IF2211_tucil3\test\Bua
Browse
Start Node
Finish Node
Bahagia1
RancaBolang1
Get Route
Bahagia1 -> Cipagalo2 -> Cipagalo1 ->
Cijawura1 -> Cijawura2 -> SH3B -> SH4 ->
SH3 -> TataSurya1 -> RancaBolang1
Jarak =4,4929942742917 km
Check Matrix

E. Tabel Penilaian

Poin	Ya	Tidak
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	√	

F. Sumber Program

https://github.com/alvinwilta/IF2211_tucil3