

Laporan Tugas Kecil A* Algorithm

Strategi Algoritma IF2211



Anggota :
Cynthia Rusadi (13519118)
Kevin Ryan (13519191)

Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

BAB I

Kode Program

Kode program untuk tugas besar IF2211 Strategi Algoritma dikerjakan dengan bahasa Python dan memanfaatkan beberapa libraries yang tersedia oleh Python. Proses penulisan dilakukan secara modular untuk meningkatkan efektivitas dan efisiensi. Berikut modul-modul python yang diimplementasikan :

1. algorithm.py

Modul ini berisi A* Algorithm. Fungsi aStar di dalam program ini menerima input start point dan end point dan akan mengembalikan list of points jalur dari point start ke point end.

```
import point

def aStar(start, end):
    path = [start] # initialize path that will be used
    pathCost = []
    curr = start # initialize the current path to be the starting
point
    found = True # used if there's no path to be found
    visited = [start] # for backtracking purposes

    while (not curr.isSame(end)): # loop until the current point =
end point
        adjacent = [x for x in curr.route if x not in visited and x
not in path] # list of adjacent points
        findCost = [(99999*99999) for y in range(len(adjacent))] #
array that will be filled with the distance between 2 points, the
number 99999*99999 is used to avoid big distance numbers
        if not any(adjacent): # if adjacent is empty
            visited.append(path[len(path)-1]) # to state that the
point can no longer be used
            path.pop() # remove the path that can't be used anymore
            if len(path) != 0: # if there might still be a way
                pathCost.pop()
                curr = path[len(path)-1]
                continue # go back to the start of the loop
        found = False # to indicate there aren't any paths
```

```

        break # end the loop

    for adjacentPoints in adjacent: # get adjacent point in
adjacent
        tmpCost = adjacentPoints.euclideanDistance(end) +
adjacentPoints.euclideanDistance(curr) # temp cost = distance between
adjacent point and end point + distance between adjacent point and
current point
        findCost[adjacent.index(adjacentPoints)] = tmpCost # to
set the distances of temp cost based on the index of the adjacent
point that was used to calculate temp cost

        minIdx = findCost.index(min(findCost)) # to find the minimum
distance in findCost
        pathCost.append(findCost[minIdx])
        path.append(adjacent[minIdx]) # add point that has the
minimum distance to path
        curr = adjacent[minIdx] # set current point to point that has
the minimum distance

    if found: # if the path between start point and end point exists
        print("Jarak :", sum(pathCost), "km") # print the distance
between start point and end point
        return path # return the path
    print("Tidak ada jalan yang terbentuk")
    return [start] # return only start point

```

2. main.py

Modul ini berisi program utama. Modul ini berfungsi untuk memanggil modul-modul lain dan mengontrol alur jalannya program.

```

import point
import visualize
import algorithm
from geopy.geocoders import Nominatim

locator = Nominatim(user_agent="myGeocoder")

```

```

# Check if a point is in a list. Returns boolean
def pointInList(point, listPoint) : # Checks if point is in listPoint
    for elmt in listPoint : # Iterating every element in listPoint
        if point.x == elmt.x and point.y == elmt.y : # Returns True
            if found
                return True
    return False # Returns False by default

# Find a point in a list. Returns object point
def findPoint(point, listPoint) : # Find point in listPoint
    for elmt in listPoint : # Iterating every element in listPoint
        if point.x == elmt.x and point.y == elmt.y : # Returns elmnt
            if found
                return elmt
    return None # Returns None by default

# Read file and initialize listPoints (a list with elements of all
points in input file)
def read_file(file_name): # Read {file_name}.txt if located in test
folder
    listPoints=[] # Initializing empty list

    with open("../test/" + file_name, 'r') as file : # Opening file
        line = file.read() # Reads the content of the text file
        listLines = line.split("\n") # Make a list with elements of
each line in the text file

        lineCount = int(listLines[0]) # Initializing N

        for i in range(1, 1 + lineCount) : # Initializing every
points and store it in listPoints
            tmpString = listLines[i].replace(' ', '').split(",") #
Remove spaces and split by , to get coordinates
            tmpPoint = point.point(float(tmpString[0]),
float(tmpString[1])) # Construct point

```

```

        if (not(pointInList(tmpPoint, listPoints))): # Add point
to list if listPoints doesn't contain the point
            listPoints.append(tmpPoint)

    # Initialize all routes
    for i in range(1 + lineCount, 1 + 2 * lineCount) : #
Iterating every coordinate (2nd line to (2 + N)th line)
        tmpString = listLines[i].split(' ')

        for j in range(lineCount) : # Add to route if the element
in adjacency matrix == 1
            if tmpString[j] == "1" :
                if not(pointInList(listPoints[j], listPoints[i -
1 - lineCount].route)) : # Add to route if point hasn't been added
                    listPoints[i - 1 -
lineCount].addRoute(listPoints[j])
            return listPoints # Returns listPoints

# Main program
print("1. Dari titik yang sudah dipilih di web")
print("2. Dari file")

choice = input("Pilihlah input yang akan digunakan : ")
if choice == "1":
    file_name = "input.txt"
else:
    file_name = input("Input file name: ")

# Handling the case where user did not input .txt extension
if ".txt" not in file_name:
    file_name += ".txt"
listPoints = read_file(file_name)

# Opening file
with open("../test/" + file_name, 'r') as file :
    line = file.read() # Read the content of the file
    listLines = line.split("\n") # Splitting by line

```

```

linecount = int(listLines[0]) # Initializing N

for i in range(1, 1+linecount) : # Prints addresses
    location = locator.reverse(listLines[i])
    print("%d. %s" %(i, location.address))

start_index = int(input("Masukkan titik asal : ")) # Input start
coordinate
end_index = int(input("Masukkan titik tujuan : ")) # Input end
coordinate

choice = "0"
while (choice != "2") :
    print("\n1. Visualisasi graf dalam bidang kartesian")
    print("2. Jalankan A* dan visualisasi")
    choice = input("Pilihan : ")
    if choice == "1" :
        visualize.visualize(listPoints) # Visualize graph
    elif choice == "2" :
        path = algorithm.aStar(listPoints[start_index-1],
listPoints[end_index-1]) # Call the A* Algorithm
        visualize.visualizee(listPoints, path,
listPoints[start_index-1], listPoints[end_index-1]) # Visualizing
result into a map
        break
    else :
        print("Input salah")

```

3. makeRoute.py

Modul ini memberikan akses bagi user untuk membangun matriks ketetanggaan yang merepresentasikan ketersediaan jalan antara 2 titik koordinat yang telah dipilih melalui OpenLayers API. Hasil matriks akan secara otomatis dimasukkan ke dalam input.txt sehingga dapat langsung digunakan oleh user.

```

import geopy
from geopy.geocoders import Nominatim

```

```

locator = Nominatim(user_agent="myGeocoder")

with open("../test/input.txt", "w") as file: # Clear txt file
    file.writelines("")

with open("../coordinate/coordinate.txt", "r") as file : # Opening
coordinate.txt in read mode
    coordinates = file.read().split("\n") # Splitting coordinates.txt
by lines
    for i in range(len(coordinates)-1) : # Print the list of
coordinates
        location = locator.reverse(coordinates[i])
        print("%d. %s" % (i+1, location.address))

    # Initializing adjacency matrix of size N * N
    matriks = [[0 for i in range(len(coordinates)-1)] for j in
range(len(coordinates)-1)]

    # Loop to build routes
    while(True) :
        masukan = input("Masukkan route yang ingin dihubungkan
('Point1' 'Point2'): ") # Read Input

        if masukan == "-99" : # Stop condition
            break
        indexes = masukan.split(" ")

        try: # Handling error inputs (index out of bounds)
            matriks[int(indexes[0])-1][int(indexes[1])-1] = 1
            matriks[int(indexes[1])-1][int(indexes[0])-1] = 1
        except:
            print("Sepertinya ada yang salah dengan inputnya")

with open ("../test/input.txt", "a") as file: # Open input.txt in
append mode
    file.writelines("%d\n" % (len(coordinates)-1)) # Write N
    for i in range (len(coordinates)-1): # Write all the coordinates

```

```

        file.writelines("%s\n" % coordinates[i])

    for i in range(len(coordinates)-1) : # Write the adjacency matrix
        for j in range(len(coordinates)-1) :
            file.writelines("%d " % matriks[i][j])
        if i != len(coordinates)-2:
            file.writelines("\n")

print("Memasukkan matriks terbaru ke input.txt") # Success message

```

4. Point.py

Modul ini mengimplementasikan pemrograman berorientasi objek (OOP) yang berfungsi untuk membuat kelas point. Kelas point ini akan berisi titik-titik koordinat dan juga jalur dari titik tersebut ke titik lainnya. Program ini juga berisi methods yang dapat digunakan, seperti method haversine untuk menghitung jarak antara 2 titik.

```

import math

class point : # Initializing class point
    def __init__(self, x, y): # Constructor
        self.x = x
        self.y = y
        self.route = []

    def haversine(self, point2) :
        R = 6373.0 # Radius of the earth (in km)

        # Converting coordinates to radians
        lat1 = math.radians(float(self.x))
        lon1 = math.radians(float(self.y))
        lat2 = math.radians(float(point2.x))
        lon2 = math.radians(float(point2.y))

        # Calculate longitude and latitude distance
        dlon = lon2 - lon1
        dlat = lat2 - lat1

```



```

        # Executing Haversine Formula
        a = math.sin(dlat / 2)**2 + math.cos(lat1) * math.cos(lat2) *
math.sin(dlon / 2)**2
        c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
        distance = R * c

    # Return distance
    return distance

def addRoute(self, point) : # Add point to routes list
    self.route.append(point)

def printPoint(self) :
    print("(%f,%f)" % (self.x, self.y)) # Print coordinate

def isSame(self, point2):
    return self.x == point2.x and self.y == point2.y # Checks if
2 points are the same

```

5. server.py

Modul ini menggunakan library flask untuk berkomunikasi antara frontend dan backend. Modul ini berfungsi untuk mengeksekusi python script untuk file manipulating seperti clear text file, write text file, dll.

```

from flask import Flask, request, render_template
import json

# template
app = Flask(__name__)

# server on route '/' with both methods (get and post)
@app.route('/', methods=["GET", "POST"])
def writeText() :
    # if the map is clicked
    if request.method == "POST":

```

```

# get data from frontend and make it into json
data = request.data.decode("utf8")
jsonData = json.loads(data)
# get data from json
longitude = jsonData["longitude"]
latitude = jsonData["latitude"]
with open("./coordinate/coordinate.txt", 'a') as file :
    # write into coordinate.txt (in ./coordinate) the
coordinate that was clicked
    file.writelines("%f, %f\n" % (latitude, longitude))
# if the web is refreshed or just started
else:
    with open("./coordinate/coordinate.txt", "w") as file:
        # clear coordinate.txt (in ./coordinate)
        file.writelines("")
# show the map
return render_template("coordinate.html")

if __name__ == '__main__':
    app.run(debug=True)

```

6. visualize.py

Modul ini menggunakan library plotly untuk melakukan graphing. Fungsi visualize di dalam modul ini berfungsi untuk menggambarkan graf dalam bidang kartesian. Fungsi visualizer di dalam modul ini berfungsi untuk menggambarkan rute dari start point ke end point pada peta dunia.

```

import plotly.graph_objects as go
import point

def visualize(listPoints) : # This function visualizes the graph in
cartesian plane
    with open("../test/input1.txt", "r") as f :
        # Initializing empty list x and y for edge
        edge_x = []
        edge_y = []

```

```

# Appending edges with according to format
for elmt in listPoints :
    for elmt2 in elmt.route :
        edge_x.append(elmt.x)
        edge_x.append(elmt2.x)
        edge_x.append(None)
        edge_y.append(elmt.y)
        edge_y.append(elmt2.y)
        edge_y.append(None)

# Plotting with scatter
edge_trace = go.Scatter(
    x=edge_x, y=edge_y,
    line=dict(width=1, color='black'),
    hoverinfo='none',
    mode='lines')

# Initializing empty list x and y for node
node_x = []
node_y = []

# Appending x and y to list node
for elmt in listPoints :
    node_x.append(float(elmt.x))
    node_y.append(float(elmt.y))

# Make traces
node_trace = go.Scatter(
    x = node_x, y=node_y,
    mode = 'markers',
    marker = dict(
        line_width=5))

# Make figure
fig = go.Figure(data=[edge_trace, node_trace],
    layout = go.Layout( # Set layout
        title = 'Visualisasi Points',

```

```

        titlefont_size = 16,
        showlegend = False,
        margin = dict(b = 20, l = 5, r = 5, t = 40),
        xaxis = dict(showgrid = True, zeroline = True,
showticklabels = True),
        yaxis = dict(showgrid = True, zeroline = True,
showticklabels = True)
    ))
    fig.show()

```

```

def visualizee(listPoints, path, start, end) : # This functions
visualizes the routes in world map

```

```

    fig = go.Figure()

```

```

    for i in range(len(path)-1):

```

```

        fig.add_trace(go.Scattermapbox(
            mode = "markers+lines",
            lon = [path[i].y, path[i+1].y],
            lat = [path[i].x, path[i+1].x],
            marker = {"size" : 10, "color" : "black"},
            line_color = "blue",
        ))

```

```

    fig.add_trace(go.Scattermapbox(
        mode = "markers+text",
        lon = [start.y],
        lat = [start.x],
        marker = {"size" : 10, "color" : "red"},
        text = ["Start"],
        textposition = "top center",
        name = "Start"
    ))

```

```

    fig.add_trace(go.Scattermapbox(
        mode = "markers+text",
        lon = [end.y],
        lat = [end.x],

```

```

        marker = {"size" : 10, "color" : "red"},
        text = ["End"],
        textposition = "top center",
        name = "End"
    ))

fig.update_layout(
    margin = {'l':0, 't':0, 'b':0, 'r':0},
    mapbox = {
        'center' : {'lat':0, 'lon':50},
        'style' : "stamen-terrain",
        'zoom' : 1,
        'center' : {'lat':0, 'lon':50}
    }
)

fig.show()

```

7. Coordinate.html

File ini merupakan file html yang berupa frontend webpage yang akan menampilkan peta dunia. File ini menggunakan OpenLayers API yang memungkinkan user untuk mengclick peta untuk mendapatkan koordinat. File ini berhubungan dengan file server.py untuk mengeksekusi python script.

```

<!DOCTYPE HTML>

<html>
<head>
    <title>CLICK HANDLER</title>
    <script src="http://www.openlayers.org/api/OpenLayers.js"></script>
    <form action = "/" method = "post">
    <script>
        var map,vectorLayer,selectMarkerControl,selectedFeature;
        var lat = -6.891059;
        var lon = 107.610436;
        var zoom = 8;
        var curpos = new Array();
        var position;

```

```
    var fromProjection = new OpenLayers.Projection("EPSG:4326"); //  
Transform from WGS 1984
```

```
    var toProjection = new OpenLayers.Projection("EPSG:900913"); // to  
Spherical Mercator Projection
```

```
    var cntrposition = new OpenLayers.LonLat(lon,  
lat).transform(fromProjection, toProjection);
```

```
function init()  
{  
    map = new OpenLayers.Map("Map");  
    var mapnik = new OpenLayers.Layer.OSM("MAP");  
    var markers = new OpenLayers.Layer.Markers( "Markers" );  
  
    map.addLayers([mapnik,markers]);  
    map.addLayer(mapnik);  
    map.setCenter(cntrposition, zoom);  
  
    var click = new OpenLayers.Control.Click();  
    map.addControl(click);  
  
    click.activate();  
};
```

```
OpenLayers.Control.Click = OpenLayers.Class(OpenLayers.Control, {  
    defaultHandlerOptions: {  
        'single': true,  
        'double': false,  
        'pixelTolerance': 0,  
        'stopSingle': false,  
        'stopDouble': false  
    },  
    initialize: function(options) {
```

```
        this.handlerOptions = OpenLayers.Util.extend(  
            {}, this.defaultHandlerOptions  
        );  
        OpenLayers.Control.prototype.initialize.apply(
```

```

        this, arguments
    );
    this.handler = new OpenLayers.Handler.Click(
        this, {
            'click': this.trigger
        },
        this.handlerOptions
    );
},

trigger: function(e) {
    var lonlat = map.getLonLatFromPixel(e.xy);
    lonlat1= new
OpenLayers.LonLat(lonlat.lon,lonlat.lat).transform(toProjection,fromProjection);

    // send data to backend
    fetch("http://127.0.0.1:5000/", {
        method:"POST",
        headers: {
            'Accept': 'application/json',
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({
            "longitude" : lonlat1.lon,
            "latitude" : lonlat1.lat
        })
    })
    alert(lonlat1.lat + ", " +lonlat1.lon);
}
});
</script>
</form>
</head>

<body onload='init();'>
    <div id="Map" style="height: 100vh" ></div>
</body>

</html>

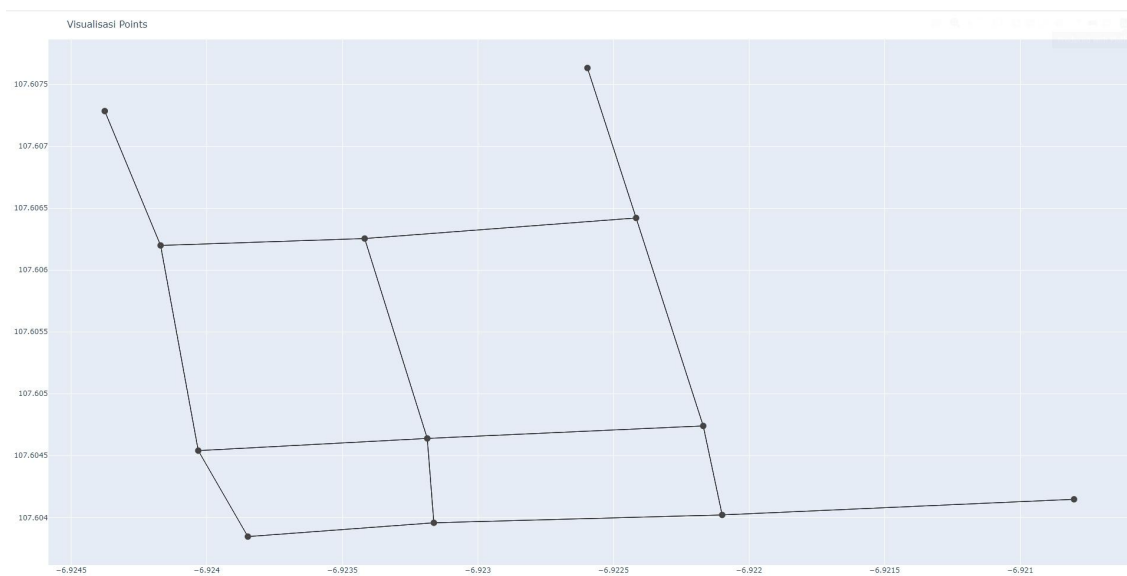
```

Bab II

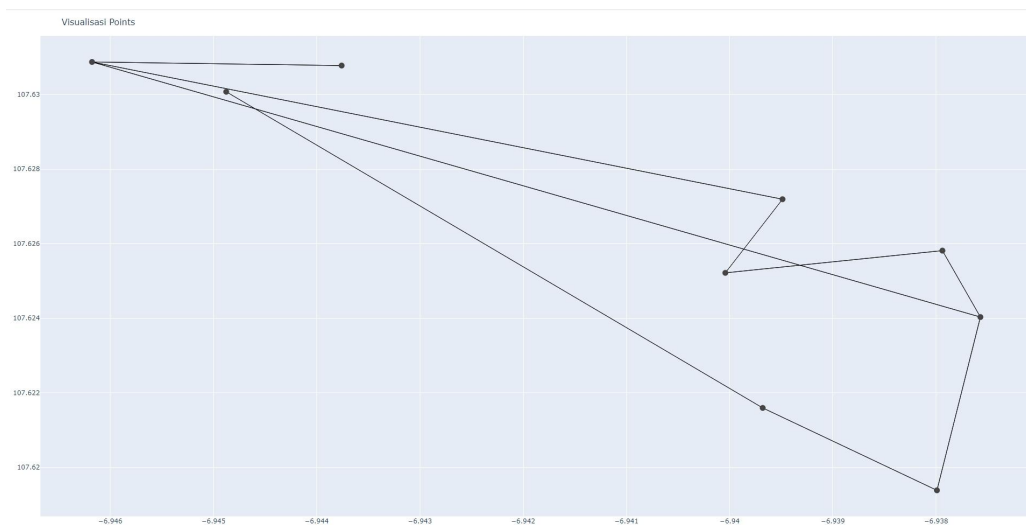
Peta / Graf

Visualisasi graf pada tugas kecil kali ini memanfaatkan library plotly. Fitur yang digunakan adalah fitur scatter dengan mode lines sehingga program dapat menggambar titik-titik koordinat pada bidang kartesian dan menghubungkan titik-titik tersebut dengan garis sesuai route yang telah ditentukan. Berikut beberapa contoh graf dari text file yang diambil dengan cara menggunakan OpenLayers API yang membuka kemungkinan bagi user untuk meng-click daerah di peta untuk mendapatkan koordinat :

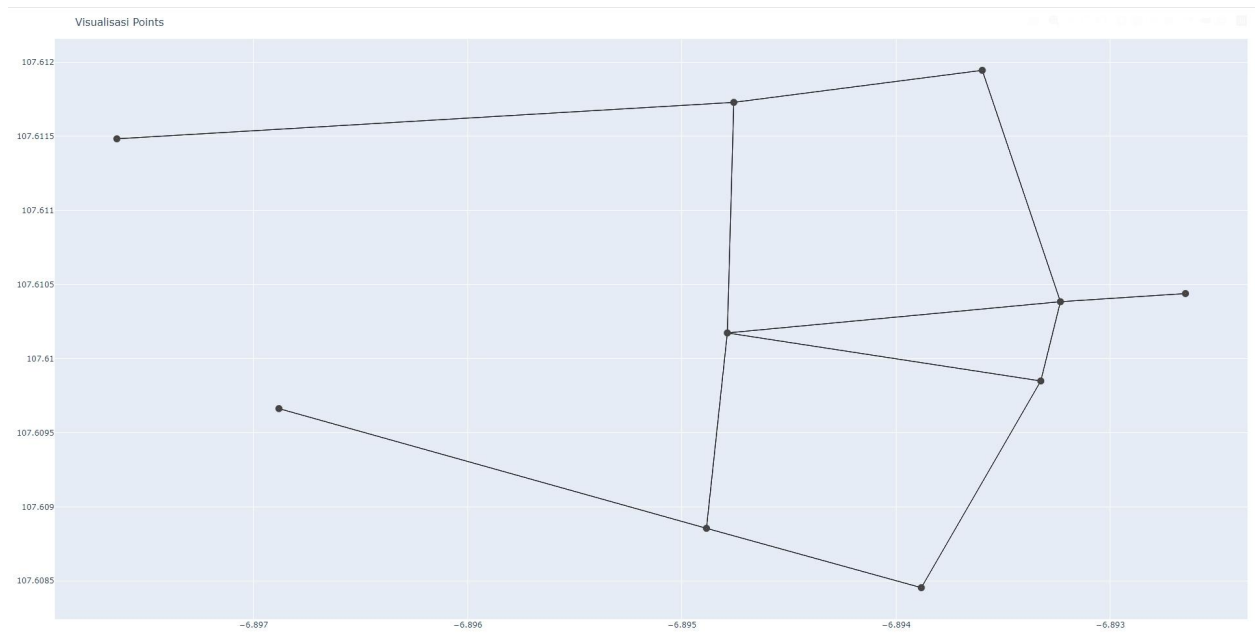
1. AlunAlun.txt



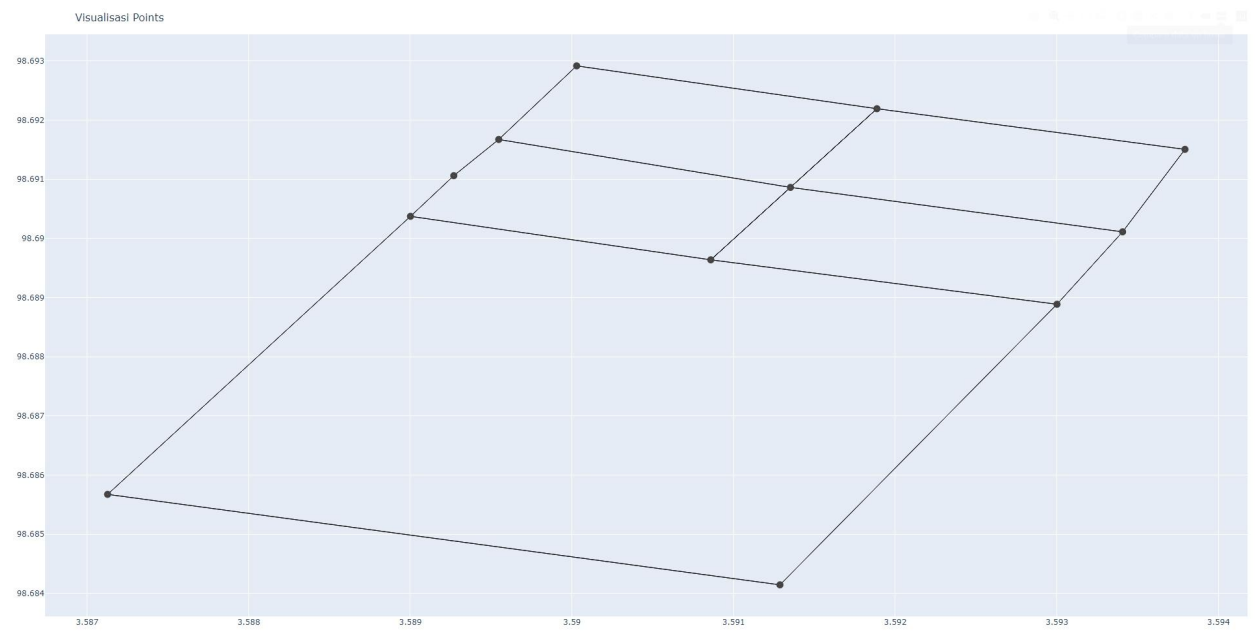
2. BuahBatu.txt



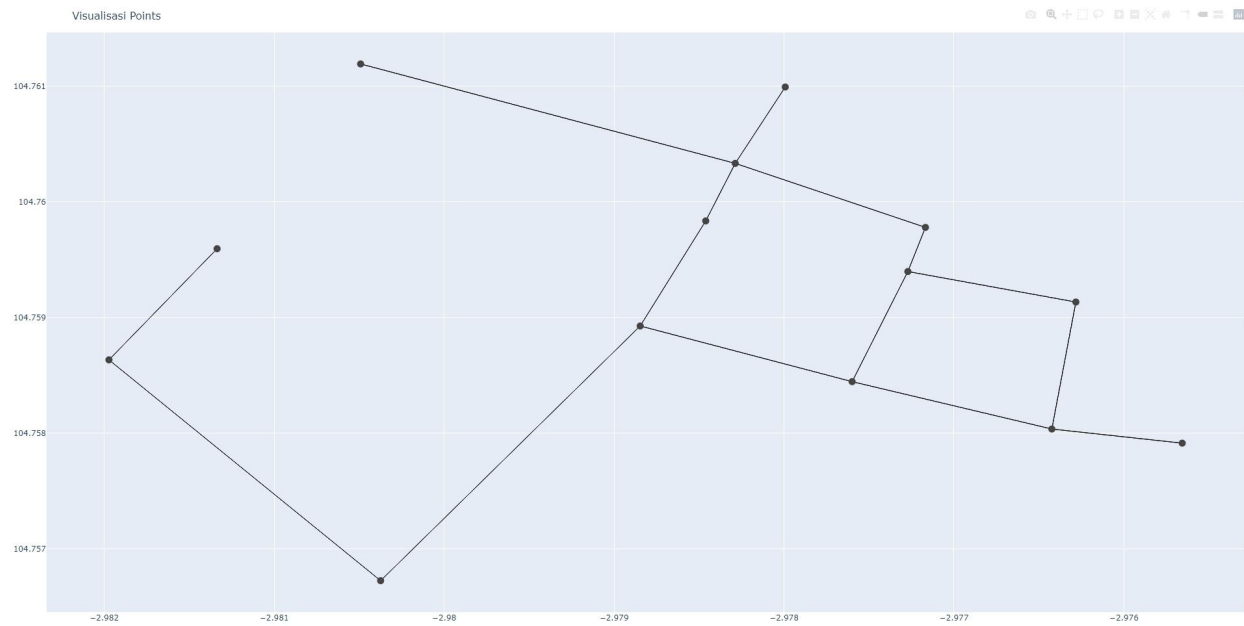
3. ITB.txt



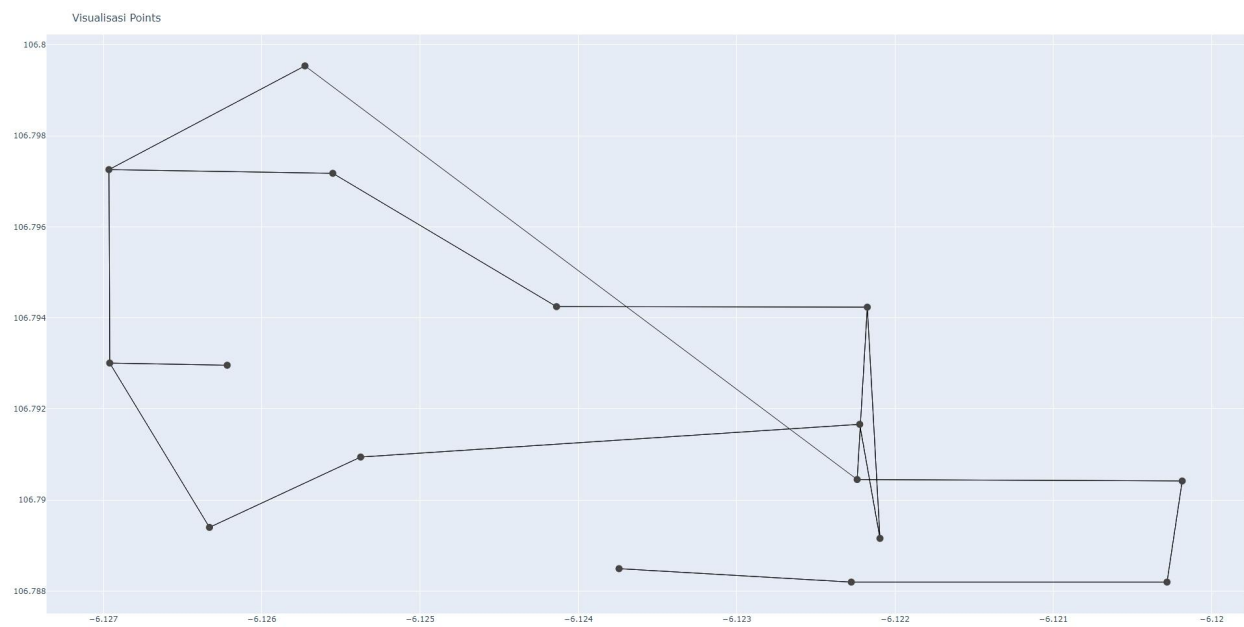
4. Medan.txt



5. Palembang.txt



6. Pluit.txt



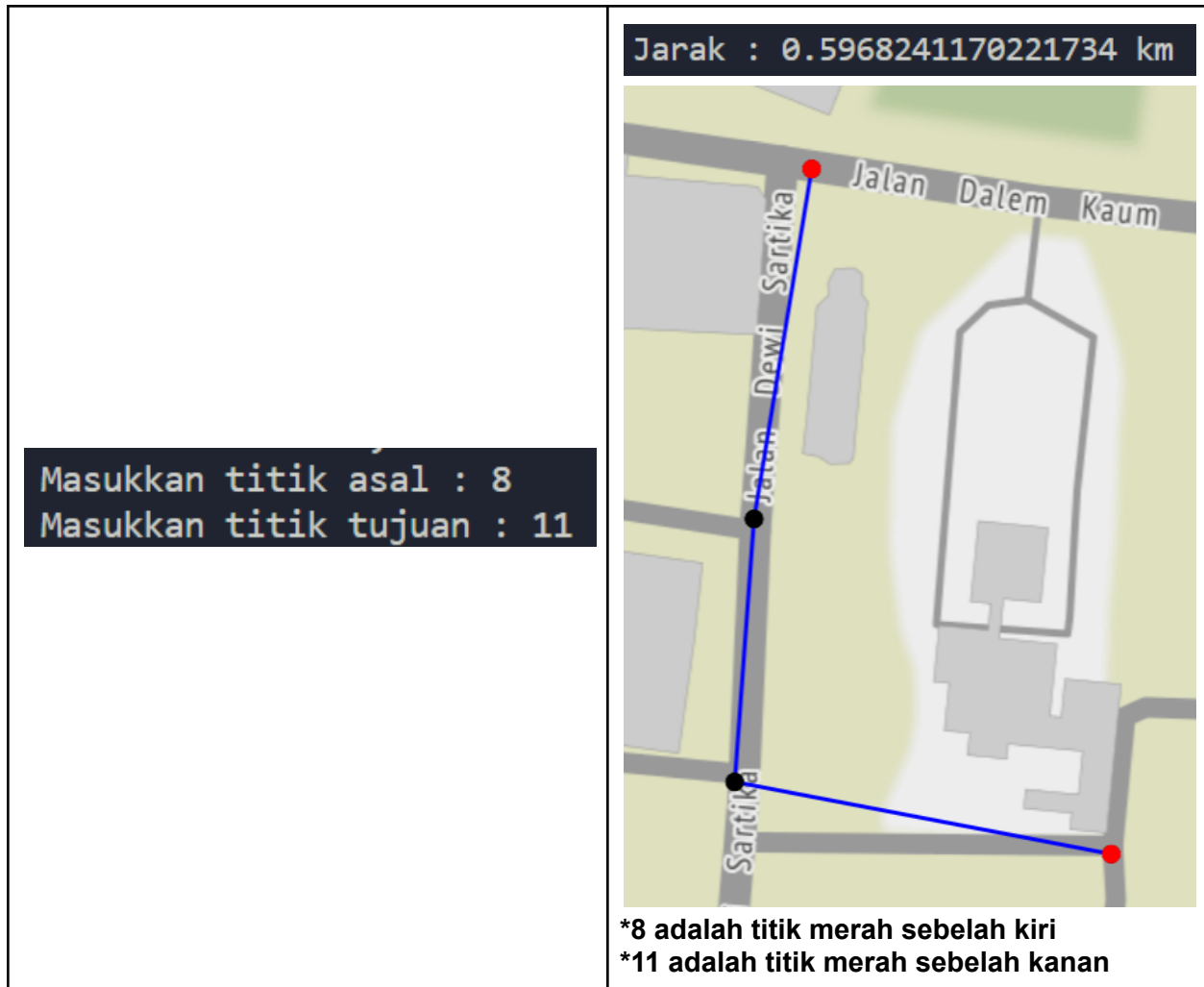
Bab III

Screenshot

1. AlunAlun.txt

Nama Jalan
<ol style="list-style-type: none"> 1. Dalem Kaum, Jalan Otto Iskandardinata, Kebon Jeruk, Jawa Barat, 40251, Indonesia 2. Hotel Golden Flower, 15-17, Jalan Asia Afrika, Kebon Jeruk, Jawa Barat, 40111, Indonesia 3. Kebon Jeruk, Jawa Barat, 40251, Indonesia 4. MEGARIA, 17, Jalan Kalipah Apo, Kebon Jeruk, Jawa Barat, 40241, Indonesia 5. Jalan Simpang, Kebon Jeruk, Jawa Barat, 40251, Indonesia 6. Jalan Kapatihan, Kebon Jeruk, Jawa Barat, 40251, Indonesia 7. Jalan Dalem Kaum, Kebon Jeruk, Jawa Barat, 40251, Indonesia 8. Jalan Dalem Kaum, Kebon Jeruk, Jawa Barat, 40251, Indonesia 9. Jalan Kapatihan, Kebon Jeruk, Jawa Barat, 40251, Indonesia 10. Jalan Simpang, Kebon Jeruk, Jawa Barat, 40251, Indonesia 11. Pendopo Kota Bandung, Jalan Balong Gede, Karanganyar, Jawa Barat, 40251, Indonesia 12. Dian Theatre, Jalan Balong Gede, Karanganyar, Jawa Barat, 40251, Indonesia

Input	Output
<p>Masukkan titik asal : 8 Masukkan titik tujuan : 3</p>	<p>Jarak : 0.5922217084603991 km</p>  <p>*8 adalah titik merah sebelah kanan **3 adalah titik merah sebelah kiri</p>
<p>Masukkan titik asal : 7 Masukkan titik tujuan : 12</p>	<p>Jarak : 0.4581631073356866 km</p>  <p>*7 adalah titik merah sebelah kiri *12 adalah titik merah sebelah kanan</p>

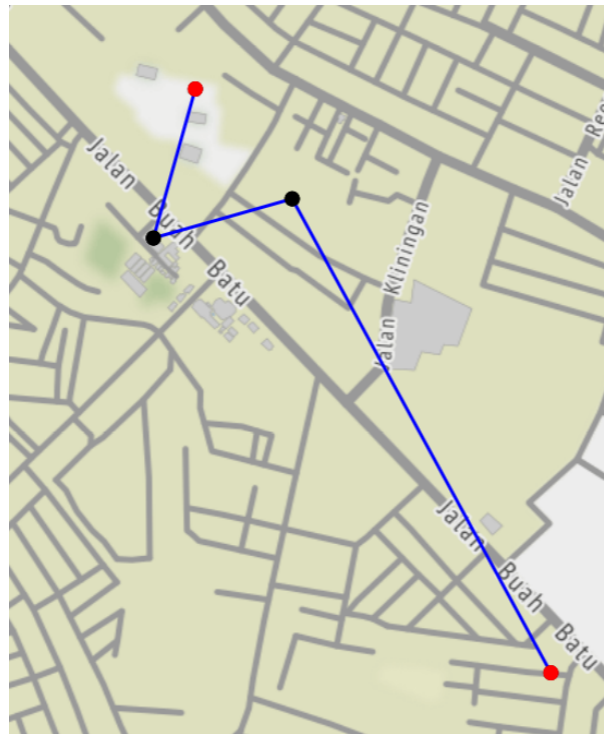


2. BuahBatu.txt

Nama Jalan	
<div>1. Bandung Eye Center, 147, Jalan Buah Batu, Bandung, Jawa Barat, 40265, Indonesia 2. Mesjid, Jalan Matraman, Malabar, Jawa Barat, 40265, Indonesia 3. BCA, Jalan Buah Batu Dalam V, Bandung, Jawa Barat, 40265, Indonesia 4. SMK Negeri 3 Bandung, Jalan Solontongan, Bandung, Jawa Barat, 40264, Indonesia 5. The Dreams Cake, 22, Jalan BKR, Bandung, Jawa Barat, 40254, Indonesia 6. Jalan Radio, Bandung, Jawa Barat, 40265, Indonesia 7. The Harvest, Jalan Buah Batu, Cijagra, Jawa Barat, 40264, Indonesia 8. Jalan Rajamantri Kulon, Turangga, Jawa Barat, 40264, Indonesia 9. Jalan Situ Sari I, Cijagra, Jawa Barat, 40266, Indonesia</div>	
Input	Output

Masukkan titik asal : 2
Masukkan titik tujuan : 9

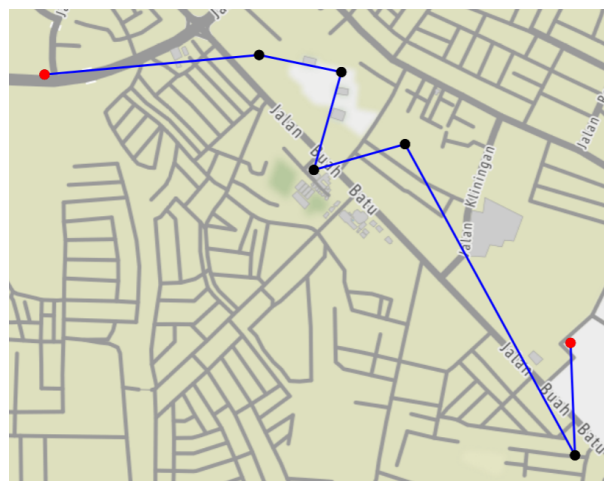
Jarak : 3.0898485676425875 km



*2 adalah titik merah sebelah kiri
**9 adalah titik merah sebelah kanan

Masukkan titik asal : 5
Masukkan titik tujuan : 8


Jarak : 5.791816306208413 km



*5 adalah titik merah sebelah kiri
**8 adalah titik merah sebelah kanan


Masukkan titik asal : 4
Masukkan titik tujuan : 6

Jarak : 3.06428513258416 km

	 <p>*4 adalah titik merah sebelah kanan **6 adalah titik merah sebelah kiri</p>
--	---

3. ITB.txt

Nama Jalan
<ol style="list-style-type: none"> 1. Rumah C, Jalan Gelap Nyawang, Lebak Siliwangi, Jawa Barat, 40132, Indonesia 2. Gelap Nyawang, Jalan Gelap Nyawang, Lebak Siliwangi, Jawa Barat, 40132, Indonesia 3. SMA Negeri 1 Bandung, Jalan Ciung Wanara, Lebak Siliwangi, Jawa Barat, 40132, Indonesia 4. Institut Teknologi Bandung, 10-12, Ganesha, Lebak Siliwangi, Jawa Barat, 40132, Indonesia 5. ITB (Gerbang Depan), Jalan Ganesha, Lebak Siliwangi, Jawa Barat, 40132, Indonesia 6. Rumah A, Jalan Ganesha, Lebak Siliwangi, Jawa Barat, 40132, Indonesia 7. BNI, Jalan Taman Sari, Lebak Siliwangi, Jawa Barat, 40131, Indonesia 8. Institut Teknologi Bandung, 10-12, Ganesha, Lebak Siliwangi, Jawa Barat, 40132, Indonesia 9. PDAM Tirtawening Kota Bandung, Jalan Taman Sari, Lebak Siliwangi, Jawa Barat, 40131, Indonesia 10. Lumpia Semarang, Jalan Badak Singa, Lebak Siliwangi, Jawa Barat, 40132, Indonesia

Input	Output
<p>Masukkan titik asal : 3 Masukkan titik tujuan : 5</p>	<p>Jarak : 0.48551275295281565 km</p> 

	<p>*3 adalah titik merah sebelah kanan **5 adalah titik merah sebelah kiri</p>
<p>Masukkan titik asal : 1 Masukkan titik tujuan : 4</p>	<p>Jarak : 0.4342868272042737 km</p>  <p>*1 adalah titik merah sebelah kiri **4 adalah titik merah sebelah kanan</p>
<p>Masukkan titik asal : 10 Masukkan titik tujuan : 6</p>	<p>Jarak : 1.0884650597434573 km</p>

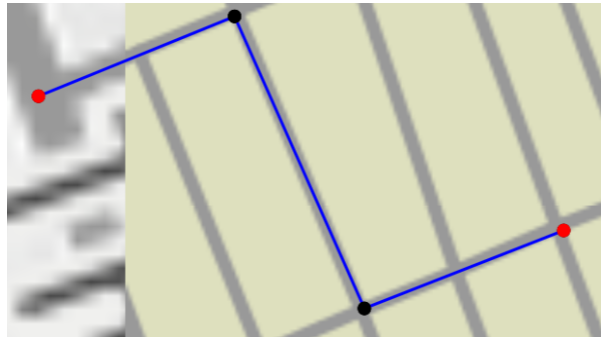



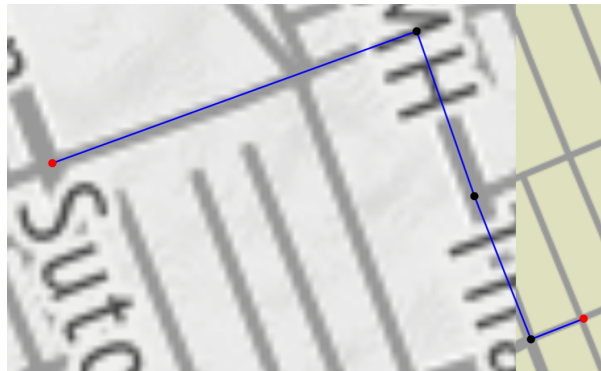
*10 adalah titik merah sebelah kanan
**6 adalah titik merah sebelah kiri

4. Medan.txt

Nama Jalan

1. Rumah C, Jalan Gelap Nyawang, Lebak Siliwangi, Jawa Barat, 40132, Indonesia
2. Gelap Nyawang, Jalan Gelap Nyawang, Lebak Siliwangi, Jawa Barat, 40132, Indonesia
3. SMA Negeri 1 Bandung, Jalan Ciung Wanara, Lebak Siliwangi, Jawa Barat, 40132, Indonesia
4. Institut Teknologi Bandung, 10-12, Ganesha, Lebak Siliwangi, Jawa Barat, 40132, Indonesia
5. ITB (Gerbang Depan), Jalan Ganesha, Lebak Siliwangi, Jawa Barat, 40132, Indonesia
6. Rumah A, Jalan Ganesha, Lebak Siliwangi, Jawa Barat, 40132, Indonesia
7. BNI, Jalan Taman Sari, Lebak Siliwangi, Jawa Barat, 40131, Indonesia
8. Institut Teknologi Bandung, 10-12, Ganesha, Lebak Siliwangi, Jawa Barat, 40132, Indonesia
9. PDAM Tirtawening Kota Bandung, Jalan Taman Sari, Lebak Siliwangi, Jawa Barat, 40131, Indonesia
10. Lumpia Semarang, Jalan Badak Singa, Lebak Siliwangi, Jawa Barat, 40132, Indonesia

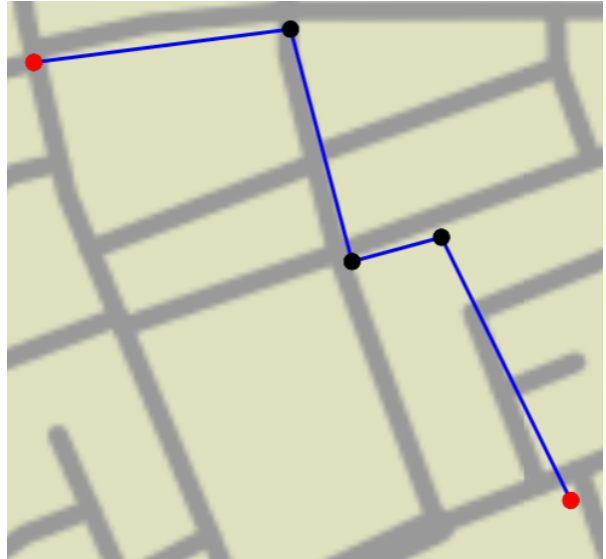
Input	Output
<p>Masukkan titik asal : 5 Masukkan titik tujuan : 12</p>	<p>Jarak : 0.9341728390563981 km</p>  <p>*5 adalah titik merah sebelah kiri **12 adalah titik merah sebelah kanan</p>
<p>Masukkan titik asal : 3 Masukkan titik tujuan : 12</p>	<p>Jarak : 1.0309534082241527 km</p>

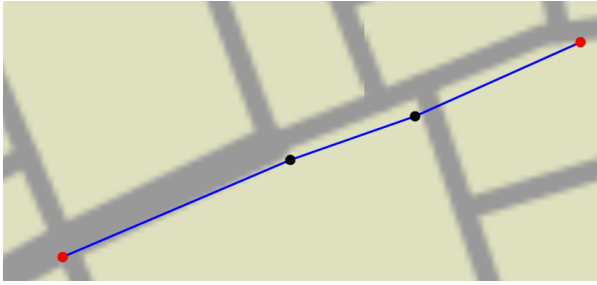

	 <p>*3 adalah titik merah sebelah kiri **12 adalah titik merah sebelah kanan</p>
<p>Masukkan titik asal : 10 Masukkan titik tujuan : 1</p>	<p>Jarak : 3.0245382787070705 km</p>  <p>*10 adalah titik sebelah kanan **1 adalah titik sebelah kiri</p>

5. Palembang.txt

Nama Jalan

1. Bank Sumsel Cabang Palembang, Jalan Kolonel Atmo, RW 06 Kelurahan 17 Ilir, Kec Ilir Timur 1, Ilir Timur I, Palembang, Sumatera Selatan, 30124, Indonesia
2. SMPN 06, Jalan Semeru, RW 05 Kelurahan 17 Ilir, Kec Ilir Timur 1, Ilir Timur I, Palembang, Sumatera Selatan, 30124, Indonesia
3. RW 05 Kelurahan 17 Ilir, Kec Ilir Timur 1, Ilir Timur I, Kepandean Baru, Sumatera Selatan, 30124, Indonesia
4. RW 05 Kelurahan 17 Ilir, Kec Ilir Timur 1, Ilir Timur I, Kepandean Baru, Sumatera Selatan, 30124, Indonesia
5. Jalan Air Mancur, RW 05 Kelurahan 17 Ilir, Kec Ilir Timur 1, Ilir Timur I, Kepandean Baru, Sumatera Selatan, 30124, Indonesia
6. Dempo Car Area, Jalan Dempo Dalam, RT 13 15 ILIR, RW 03 Kelurahan 15 Ilir, Kec Ilir Timur 1, Ilir Timur I, Palembang, Sumatera Selatan, 30124, Indonesia
7. Myrepublic Palembang 2, Jalan Kolonel Atmo, RW 06 Kelurahan 17 Ilir, Kec Ilir Timur 1, Ilir Timur I, Palembang, Sumatera Selatan, 30124, Indonesia
8. Gereja Betle, Jalan Dempo, RT 13 15 ILIR, RW 03 Kelurahan 15 Ilir, Kec Ilir Timur 1, Ilir Timur I, Palembang, Sumatera Selatan, 30124, Indonesia
9. RT 13 15 ILIR, RW 03 Kelurahan 15 Ilir, Kec Ilir Timur 1, Ilir Timur I, Palembang, Sumatera Selatan, 30124, Indonesia
10. Dempo Car Area, Jalan Dempo Dalam, RT 13 15 ILIR, RW 03 Kelurahan 15 Ilir, Kec Ilir Timur 1, Ilir Timur I, Palembang, Sumatera Selatan, 30124, Indonesia
11. Ilir Timur I, Kepandean Baru, Sumatera Selatan, 30124, Indonesia
12. Jalan Menumbing, Ilir Timur I, Kepandean Baru, Sumatera Selatan, 30124, Indonesia
13. Ilir Timur I, Kepandean Baru, Sumatera Selatan, 30124, Indonesia
14. Ilir Timur I, Kepandean Baru, Sumatera Selatan, 30124, Indonesia

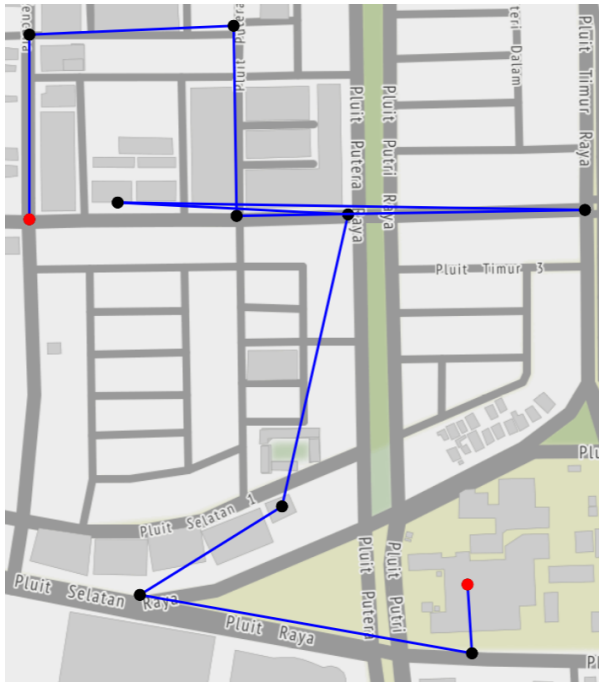
Input	Output
<p>Masukkan titik asal : 6 Masukkan titik tujuan : 13</p>	<p>Jarak : 0.9316623649702761 km</p>  <p>*6 adalah titik sebelah kanan **13 adalah titik sebelah kiri</p>
<p>Masukkan titik asal : 2 Masukkan titik tujuan : 10</p>	<p>Jarak : 0.46777666991079236 km</p>

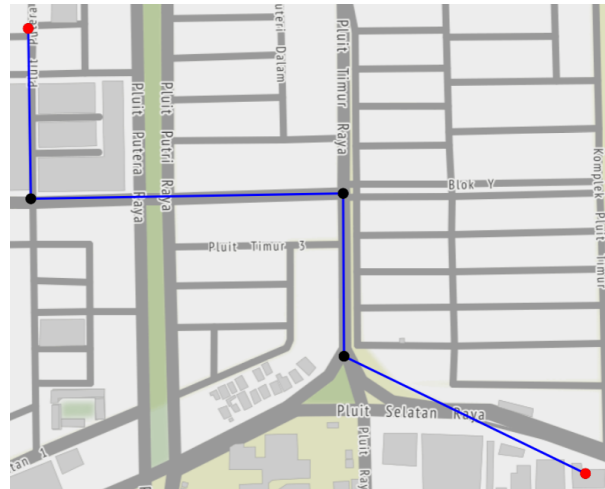
	 <p>*2 adalah titik merah sebelah kiri **10 adalah titik merah sebelah kanan</p>
<p>Masukkan titik asal : 14 Masukkan titik tujuan : 8</p>	<p>Jarak : 2.8356256008778247 km</p>  <p>*14 adalah titik merah sebelah atas **8 adalah titik merah sebelah bawah</p>

6. Pluit.txt

Nama Jalan

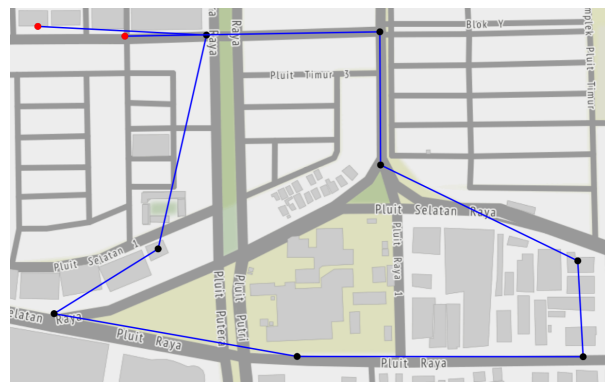
1. Rumah Sakit Pluit, Jalan Pluit Selatan Raya, RW 07, Penjaringan, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14430, Indonesia
2. SPBU Shell, Jalan Pluit Selatan Raya, RW 08, Penjaringan, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14430, Indonesia
3. Jalan Pluit Raya, RW 07, Penjaringan, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14430, Indonesia
4. Rumah Sakit Atma Jaya, Jalan Pluit Selatan Raya, RW 08, Penjaringan, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14430, Indonesia
5. Gardu Bou 14, Jalan Pluit Raya, RW 08, Penjaringan, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14430, Indonesia
6. Jalan Pluit Selatan Raya, RW 06, Pluit, Penjaringan, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14440, Indonesia
7. Gereja Bethel, Jalan Pluit Selatan I, RW 06, Pluit, Penjaringan, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14440, Indonesia
8. Laboratorium Klinik Bio Test, Jalan Pluit Sakti, RW 06, Pluit, Penjaringan, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14440, Indonesia
9. Bank Dinar, Jalan Pluit Sakti, RW 06, Pluit, Penjaringan, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14440, Indonesia
10. Jalan Pluit Sakti, RW 06, Pluit, Penjaringan, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14440, Indonesia
11. Jalan Pluit Timur Raya, RW 09, Pluit, Penjaringan, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14440, Indonesia
12. Laboratorium Klinik Prodia, Jalan Pluit Kencana, RW 07, Pluit, Penjaringan, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14440, Indonesia
13. Permata, Jalan Pluit Sakti, RW 06, Pluit, Penjaringan, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14440, Indonesia
14. Apotik Kencana Raya, Jalan Pluit Kencana, RW 06, Pluit, Penjaringan, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14440, Indonesia
15. Jalan Pluit Putra Kencana, RW 06, Pluit, Penjaringan, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14440, Indonesia
16. Rezeki, Jalan Pluit Sakti, RW 06, Pluit, Penjaringan, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14440, Indonesia

Input	Output
<p>Masukkan titik asal : 4</p> <p>Masukkan titik tujuan : 13</p>	<p>Jarak : 6.646149172155379 km</p>  <p>*4 adalah titik sebelah kanan</p> <p>**13 adalah titik sebelah kiri</p>
<p>Masukkan titik asal : 2</p> <p>Masukkan titik tujuan : 15</p>	<p>Jarak : 2.541295741827663 km</p>



***2 adalah titik sebelah kanan**
****15 adalah titik sebelah kiri**

Jarak : 7.059216273689549 km



***9 adalah titik merah sebelah kiri**
****16 adalah titik merah sebelah kanan**

Masukkan titik asal : 9
Masukkan titik tujuan : 16

Bab IV

Tabel Keberhasilan

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 OpenLayers API dan menampilkan peta	√

Bab V

Link Github

<https://github.com/kevinryann/map-a-star>