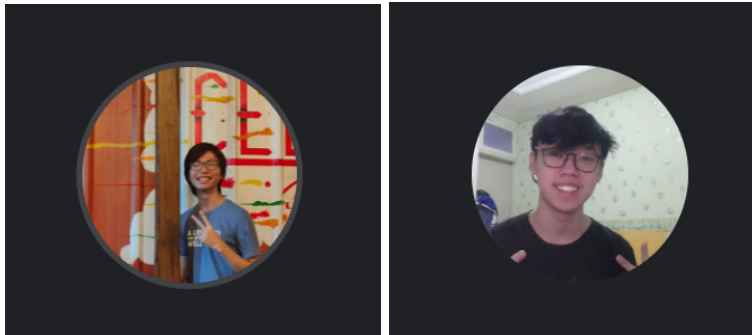


# **IMPLEMENTASI ALGORITMA A\* UNTUK MENENTUKAN LINTASAN TERPENDEK**

Laporan Tugas Kecil 3 IF2211 Strategi Algoritma  
Semester II Tahun Akademik 2020/2021



Oleh:

Dionisius Darryl Hermansyah

13519058

James Chandra

13519078

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
2021**

## I. Kode Program

---

Program dibuat dalam bahasa pemrograman Python dan terdiri dari 4 modul. Modul-modul tersebut adalah main.py (program utama), graph.py, node.py (kelas Graph dan Node), dan visualizer.py (fungsi visualisasi graf). Berikut ini merupakan kode program tersebut:

### main.py

```
import queue
from graph import *
from node import *
from visualizer import *

"""
FUNCTIONS
"""

def welcomeMessage():
    print("=====")
    print("=   A* Pathfinder   =")
    print("=====")

def getGraphInput():
    # Baca nama graf yang mau dibaca
    graphName = input("Masukkan nama graf input (tanpa ekstensi file): ")
    print()

    g = showGraphNode(graphName)

    # Baca input simpul awal dan akhir
    start = input("Masukkan nama simpul awal: ")

    while (start not in g.getAllNodeName()):
        print("Nama simpul tidak ditemukan!")
        start = input("Masukkan nama simpul awal: ")

    end = input("Masukkan nama simpul tujuan: ")

    while (end not in g.getAllNodeName()):
        print("Nama simpul tidak ditemukan!")
        end = input("Masukkan nama simpul tujuan: ")

    print()
    return graphName, start, end

"""
MAIN PROGRAM
"""
welcomeMessage()
```

```

exit = False

while (not exit):
    graphName, start, end = getGraphInput()

    # Buat graf dari nama graf input
    g = makeGraphFromTxt(graphName, end)

    # Cari node start dan end
    startNode = g.findNode(start)
    endNode = g.findNode(end)

    # Make priority queue to store nodes
    q = queue.PriorityQueue()

    # Deklarasikan dictionary kosong untuk rekam jejak parent dan rekam jejak total
    # jarak kumulatif yang paling singkat untuk ke node tertentu
    jalur = {}
    kumulatifMeter = {}

    # Inisiasi dengan none, karena parent dari starting node tidak ada, dan 0.0
    # untuk jarak kumulatif, karena jarak dari start ke start adalah 0
    jalur[start] = None
    kumulatifMeter[start] = 0.0

    # Set starting node
    q.put((0, start))

    # Inisiasikan currNodeName dengan nilai acak agar bisa masuk ke loop
    currNodeName = -999

    # jalankan loop asalkan belum sampai tujuan atau masih ada elemen prioqueue
    while (q.qsize() > 0 and currNodeName != end):
        (currPrio, currNodeName) = q.get()

        for nextNodeName in g.findNode(currNodeName).neighbors:
            addedMeter = kumulatifMeter[currNodeName] + g.findNode(
                currNodeName).neighbors[nextNodeName]

            # menambahkan kumulatifMeter baru apabila belum ada dictionary key : nextNodeName
            if (not (nextNodeName in kumulatifMeter)):
                # menambahkan kumulatifMeter baru
                kumulatifMeter[nextNodeName] = addedMeter

            # prioritas untuk nextNodeName dikalkulasi dengan menambahkan addedMeter dan nilai
            # heuristik nextNodeName
            nextPrio = addedMeter + g.findNode(nextNodeName).heuristik

            # memasukan prio dan nama node ke dalam prioqueue
            q.put((nextPrio, nextNodeName))

            # menambahkan riwayat jalur
            jalur[nextNodeName] = currNodeName

        # menambahkan kumulatifMeter baru apabila kumulatifMeter yang sebelumnya

```

```

# ternyata tidak optimal
if (addedMeter < kumulatifMeter[nextNodeName]):
    # menambahkan kumulatifMeter baru
    kumulatifMeter[nextNodeName] = addedMeter

    # prioritas untuk nextNodeName dikalkulasi dengan menambahkan addedMeter dan nilai
    # heuristik nextNodeName
    nextPrio = addedMeter + g.findNode(nextNodeName).heuristik

    # memasukan prio dan nama node ke dalam prioqueue
    q.put((nextPrio, nextNodeName))

    # menambahkan riwayat jalur
    jalur[nextNodeName] = currNodeName

# skema pencetakan hasil
if (currNodeName == end):
    # mencetak jarak terpendek antar node awal dan tujuan
    print(f"Jarak terpendek dari {start} ke {end}: {kumulatifMeter[end]} meter \n")
    print("Lintasan:")

    # list untuk nanti dibalikan
    reverseDirection = []

    # iterasi mulai dari element parent ending node
    iterPrint = jalur[end]

    # selagi belum none (iterprint merupakan elemen parent dari starting node)
    while (iterPrint != None):
        # append ke list kemudian iterasikan berikutnya
        reverseDirection.append(iterPrint)
        iterPrint = jalur[iterPrint]

    # print array dengan orientasi reverse agar dari node awal-tujuan
    for nodeName in reversed(reverseDirection):
        print(nodeName, end=" → ")

    # node tujuan
    print(end, end="\n\n")

    # Visualisasi graph
    full_path = list(reversed(reverseDirection))
    full_path.append(end)

    visualize(g, full_path, kumulatifMeter[end])

# apabila tidak ditemukan jalur
else:
    # cetak tidak ada jalur
    print(f"Tidak ada jalur yang menghubungkan {start} dan {end}\n")

# exit
exitChoice = input("Apakah anda ingin membaca file lain? (Y/N) : ")

if (exitChoice == "N" or exitChoice == "n"):

```

```
exit = True  
  
print("Terima kasih!")
```

### graph.py

```
"""  
Definisi kelas Graph dan method yang berhubungan dengan pemrosesan graph  
"""  
from node import *  
import os  
  
class Graph:  
    def __init__(self, size):  
        self.nodes = []  
        self.adj = []  
  
    def addNode(self, N):  
        # Menambahkan node N ke Graph  
        self.nodes.append(N)  
  
    def addAdj(self, value):  
        self.adj.append(value)  
  
    def printGraph(self):  
        # Mencetak graph ke layar  
        for n in self.nodes:  
            n.printNode()  
  
        print("Adjacency Matrix: ")  
        self.printAdj()  
  
        print()  
  
    def printAdj(self):  
        # mencetak adjacency matrix  
        for row in self.adj:  
            print(row)  
  
    def findNode(self, name):  
        # mengembalikan node yang memiliki nama sesuai dengan parameter  
        for node in self.nodes:  
            if (node.name == name):  
                return node  
  
    def printAllNode(self):  
        # mengoutput semua pilihan node yang tersedia  
        i = 1  
  
        for node in self.nodes:  
            print(f"{i}. {node.name}")  
            i += 1
```

```

def getAllNodeName(self):
    # mengekstrak nama setiap node yang ada
    res = []
    for node in self.nodes:
        res.append(node.name)
    return res

def makeGraphFromTxt(file_name, end):
    # Get current path
    currpath = str(os.getcwd()).split("\\")

    # Membuat graph dari file eksternal .txt
    # Variabel
    lines = []
    all_nodes = []

    # Open dan read file berdasarkan current path
    if (currpath[len(currpath)-1] in ["src", "bin"]):
        f = open(f"../test/{file_name}.txt", "r")
    else:
        f = open(f"./test/{file_name}.txt", "r")

    # Iterate line file
    lines = f.readlines()

    size = int(lines[0])

    graph = Graph(size)

    # Add adjacency matrix ke graph
    for i in range(size + 1, len(lines)):
        line = lines[i].split(" ")

        for i in range(len(line)):
            line[i] = int(line[i].replace("\n", ""))

        graph.addAdjm(line)

    # Add node ke graph
    for i in range(1, size + 1):
        line = lines[i].split(",")

        for i in range(len(line)):
            line[i] = line[i].replace("\n", "")

        curr_node = Node(line[0], float(line[1]), float(line[2]))

        graph.addNode(curr_node)

    # Cari node akhir
    for node in graph.nodes:
        if (node.name == end):
            endNode = node

    # Add nilai heuristik masing-masing node

```

```

for node in graph.nodes:
    node.calcHeuristik(endNode)

# Add edge ke graph
for i in range(len(graph.adjm)):
    for j in range(len(graph.adjm)):
        if (graph.adjm[i][j] == 1):
            (graph.nodes[i]).addNeighbors(
                graph.nodes[j],
                haversineDist(graph.nodes[i], graph.nodes[j]))

return graph

def showGraphNode(file_name):
    # Membuat graph dari file eksternal .txt
    # dan mengoutput semua node yang tersedia
    # Get current path
    currpath = str(os.getcwd()).split("\\")

    # Variabel
    lines = []
    all_nodes = []

    # Open dan read file berdasarkan current path
    if currpath[len(currpath)-1] in ["src", "bin"]:
        f = open(f'../test/{file_name}.txt', "r")
    else:
        f = open(f'./test/{file_name}.txt', "r")

    # Iterate line file
    lines = f.readlines()

    size = int(lines[0])

    graph = Graph(size)

    # Add adjacency matrix ke graph
    for i in range(size + 1, len(lines)):
        line = lines[i].split(" ")

        for i in range(len(line)):
            line[i] = int(line[i].replace("\n", ""))

        graph.addAdjm(line)

    # Add node ke graph
    for i in range(1, size + 1):
        line = lines[i].split(",")

        for i in range(len(line)):
            line[i] = line[i].replace("\n", "")

        curr_node = Node(line[0], float(line[1]), float(line[2]))

        graph.addNode(curr_node)

```

```
graph.printAllNode()
```

```
return graph
```

### node.py

```
"""
```

Definisi kelas Node dan method yang berhubungan dengan pemrosesan Node

```
"""
```

```
from math import sin, cos, sqrt, atan2, radians
```

```
class Node:
```

```
    def __init__(self, name, x, y):
```

```
        self.name = name          # Nama node
```

```
        self.x = x                # Titik latitude
```

```
        self.y = y                # Titik longitude
```

```
        self.neighbors = dict()   # Dictionary of node, menunjukkan hubungan neighbors antar Node  
        # dengan sebuah weight
```

```
        self.heuristik = 0.0      # Nilai heuristik h(n), jarak antara node dengan node yang dicari
```

```
    def addNeighbors(self, N, weight):
```

```
        # Menambahkan neighbor self dengan node N dan bobot weight
```

```
        self.neighbors[N.name] = weight
```

```
    def printNode(self):
```

```
        # mencetak node beserta simpul tetangga serta nilai heuristik ke ending node
```

```
        print(f"{self.name} at ({self.x}, {self.y})")
```

```
        for neighbor, weight in self.neighbors.items():
```

```
            print(f"Neighbor: {neighbor} ({weight})")
```

```
        print(f"Nilai heuristik h(n) ke simpul tujuan: {self.heuristik}")
```

```
        print()
```

```
    def calcHaversineDist(self, n):
```

```
        # Menghitung jarak dari node dan node lain (n) berdasarkan longitude dan latitude
```

```
        # Menggunakan Haversine formula dalam meter
```

```
        haversineDist(self, n)
```

```
    def calcHeuristik(self, find):
```

```
        # Menghitung nilai heuristik dari node ini ke node akhir yang ingin dicari
```

```
        # Menggunakan Haversine formula dalam meter
```

```
        self.heuristik = haversineDist(self, find)
```

```
def haversineDist(n1, n2):
```

```
    # Menghitung jarak dari node n1 dan n2 berdasarkan longitude dan latitude
```

```
    # Menggunakan Haversine formula dalam meter
```

```
    # Aproksimasi radius bumi dalam meter
```

```
    R = 6373.0 * 1000
```



```

lat1 = radians(n1.x)
lon1 = radians(n1.y)

lat2 = radians(n2.x)
lon2 = radians(n2.y)

dlon = lon2 - lon1
dlat = lat2 - lat1

a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
c = 2 * atan2(sqrt(a), sqrt(1 - a))

distance = R * c

return distance

```

### visualizer.py

```

from graph import *
from node import *
import networkx as nx
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

def visualize(graph, path, dist):
    # melakukan visualisasi dari sebuah graph
    # sekaligus memvisualisasikan path dari algoritma A*

    # Matplotlib figure
    plt.figure(figsize=(8,7))
    plt.margins(.2, .2)
    plt.tight_layout()

    # Legend
    start_patch = mpatches.Patch(color='#BFFF80', label='Starting point')
    end_patch = mpatches.Patch(color='#FF9F80', label='Destination point')
    path_patch = mpatches.Patch(color='#FFFF99', label='Path')
    plt.legend(handles=[start_patch, end_patch, path_patch])

    # Buat graph
    G = nx.DiGraph()

    # List dari seluruh node
    all_nodes = []

    # Path element
    edgelist_path = [] # Menyimpan edge yang termasuk ke dalam path
    start = [path[0]] # Starting node
    end = [path[len(path)-1]] # Node tujuan
    other_path = [node for node in path if node not in start and node not in end]

    # Non-path element
    nodelist = []

```

```

edgelist = []

# Ambil list of nodes dan edges
for i in range(len(path)-1):
    edgelist_path.append((path[i], path[i+1]))

for node in graph.nodes:
    if (node.name not in path):
        nodelist.append(node.name)
        G.add_node(node.name)
        all_nodes.append(node.name)

for i in range(len(graph.adjm)):
    for j in range(i+1, len(graph.adjm)):
        if (graph.adjm[i][j] == 1):
            if ((graph.nodes[i].name, graph.nodes[j].name) not in edgelist_path and
                (graph.nodes[j].name, graph.nodes[i].name) not in edgelist_path):
                edgelist.append((graph.nodes[i].name, graph.nodes[j].name))

# Ambil posisi dari node yang ada
pos=nx.spring_layout(G)

# Draw nodes
nx.draw_networkx_nodes(G,pos,
                        nodelist=start,
                        node_color='#BFFF80',
                        node_size=300,
                        alpha=1)

nx.draw_networkx_nodes(G,pos,
                        nodelist=end,
                        node_color='#FF9F80',
                        node_size=300,
                        alpha=1)

nx.draw_networkx_nodes(G,pos,
                        nodelist=other_path,
                        node_color='#FFFF99',
                        node_size=300,
                        alpha=1)

nx.draw_networkx_nodes(G,pos,
                        nodelist=nodelist,
                        node_color='#E0DEDD',
                        node_size=300,
                        alpha=1)

# Draw edges
nx.draw_networkx_edges(G,pos,
                        edgelist=edgelist_path,
                        width=1,
                        alpha=1,
                        edge_color='red',
                        arrowsize= 10,
                        arrows=True)

```

```

nx.draw_networkx_edges(G,pos,
    edgelist=edgelist,
    width=1,
    alpha=.5,
    edge_color='black',
    arrows=False)

# Draw labels
labels={}

for i in range(0, len(all_nodes)):
    labels[all_nodes[i]] = all_nodes[i]

nx.draw_networkx_labels(G,pos,labels,font_size=7)

# Set title
plt.title(f'{start[0]} - {end[0]} : " + "%.2f"%dist + " m", x=0.4, y=0.95)
# Show
plt.show()

```

## II. Peta dan Graf Input

Pada program yang telah dibuat, dapat diberikan input dari file eksternal yang berekstensi .txt.

Data test yang disediakan berjumlah 6 buah peta, yaitu:

1. Peta jalan sekitar kampus ITB/Dago
2. Peta jalan sekitar Alun-alun Bandung
3. Peta jalan sekitar Buahbatu
4. Peta jalan sebuah kawasan Kemayoran
5. Peta jalan antar *landmark* di dunia yang bersifat *totally disconnected*
6. Peta jalan sekitar Menara Eiffel, Paris, Prancis

Input peta dan graf dalam bentuk file eksternal .txt secara lebih rinci ditunjukkan oleh tabel 1.

Tabel 1. Input peta dan graf

MapITB.txt
11
Jalan Ganesha Tengah,-6.8932080262242375,107.61044386464839
Jalan Ganesha Barat,-6.893862447419398,107.60844786645244
PTSNT Batan,-6.8885944268242465,107.60809581514792
Jalan Dayang Sumbi,-6.88737709762237,107.6114562833061
Jalan Juanda,-6.887404556208068,107.61355369500218

<p>Jalan Ganesha Timur,-6.893755746377439,107.6129508222805  Angkringan Narji,-6.894803037596001,107.61022928428314  Scoop and Skoops,-6.886819345538713,107.6133885945972  Gedung CRCS,-6.887926701957486,107.61176338697865  Jl IV Timur,-6.889880635467217,107.61157162729022  Plaza Widya Nusantara,-6.889963010791021,107.61037771612914  0 1 0 0 0 1 1 0 0 0 1  1 0 1 0 0 0 0 0 0 0 0  0 1 0 1 0 0 0 0 0 0 0  0 0 1 0 1 0 0 0 1 1 0  0 0 0 1 0 1 0 1 0 0 0  1 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 1 0 0 0 0 0 0  0 0 0 1 0 0 0 0 0 0 0  0 0 0 1 0 0 0 1 0 1 0  1 0 0 0 0 0 0 0 0 0 0</p>
<p><b>MapAlunBandung.txt</b></p> <p>10  Kantor Pos Bandung,-6.921059074042508,107.60648416401784  Monumen Asia Afrika,-6.921175462231705,107.60756695706431  Jl Alun-Alun Timur,-6.922524224973004,107.60754971801818  Gerbang Alun-Alun,-6.922555034702758,107.60731177866587  Jl Alun Dalam Tenggara,-6.9223462162952885,107.60728763938474  Jl Alun Dalam Timur Laut,-6.921411667504991,107.60745316046116  Jl Alun Dalam Barat Laut,-6.921302124814193,107.60688072810532  Jl Alun Dalam Barat Daya,-6.922281175417579,107.60646347403119  Masjid Raya Bandung,-6.921949120113879,107.60625657080152  Parahyangan Plaza,-6.922404412455186,107.60615656749039  0 1 0 0 0 0 1 0 0 0  1 0 1 0 0 0 0 0 0 0  0 1 0 1 0 0 0 0 0 0  0 0 1 0 1 0 0 0 0 1  0 0 0 1 0 1 0 1 0 0  0 0 0 0 1 0 1 0 0 0  1 0 0 0 0 1 0 1 1 0  0 0 0 0 1 0 1 0 0 0  0 0 0 0 0 0 1 0 0 1  0 0 0 1 0 0 0 0 1 0</p>
<p><b>MapBuahbatu.txt</b></p> <p>9  Metro Indah Mall,-6.942120717034387,107.65874719557186  Jl Soekarno Hatta,-6.940290411439278,107.65824797384171  Jembatan Sungai Cicadas,-6.942040661767213,107.65277526794446  Yamaha Service Center,-6.943154221380146,107.65990485388869  Jl Jupiter Barat 26,-6.943648609245794,107.65761911899006  Taman S2,-6.945294809444772,107.65724156585514  Taman Jupiter,-6.9440854448675,107.66042481163986  Jl Jupiter Barat Utama,-6.944109165356709,107.66130417588958  Jl Raya Cirebon Bandung,-6.939199773655335,107.66391628016386  0 1 0 1 1 0 0 0 1</p>

1 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 1 0
<b>MapKemayoran.txt</b>  8 Masjid Akbar,-6.155596454799754,106.85367280900618 Halte Puma Raya,-6.1557251235936015,106.85096449092164 Monumen Ondel Ondel,-6.159923146837551,106.85188711889913 Grand Palace,-6.159665823160958,106.85337530710886 Bakso Pak Pur,-6.158048377736383,106.85268204886337 Apartemen Puri,-6.157055851616249,106.85251566823723 Warung Wonokromo,-6.15677095133923,106.85388369426994 Mediterrania Residence,-6.15469399578285,106.8529316210351 0 1 0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0
<b>MapDisconnected.txt</b>  8 Monas,-6.175222277860541,106.82631846878715 Patung Merlion,1.2868454654432049,103.85448100537842 Stasiun Zuccheli Antartika,-74.68041006646995,164.08050952950413 Gedung Empire States,40.748172093714196,-73.9873494219891 Katedral St Basils,55.752607391442346,37.623065346596945 Masjid Raya Bandung,-6.921949120113879,107.60625657080152 Christ the Redeemer Brazil,-22.95194560868889,-43.21050865717779 Piramida Giza,30.004479815237236,31.112150075375943 0
<b>MapEiffel.txt</b>  8 Eiffel Tower,48.858360039873716,2.294475955810676 Pont Dlena,48.8593457626351,2.292784182393638 Place de Varsovie,48.86058610287078,2.2908863995621305

```

Avenue Dlena,48.86277099832047,2.291935174210374
Avenue de New York,48.862167286755515,2.2931837155056614
Shangri-La Hotel,48.86391826912639,2.2934137597653845
Eiffel Carousel,48.858904730067984,2.2926451748507795
Bateau le Maxim,48.85783350743309,2.290079362305002
0 1 0 0 0 0 1 0
1 0 1 0 0 0 1 0
0 1 0 1 1 0 0 0
0 0 1 0 0 1 0 0
0 0 1 0 0 0 0 0
0 0 0 1 0 0 0 0
1 1 0 0 0 0 0 1
0 0 0 0 0 0 1 0

```

### III. *Screenshot* Hasil dan Peta

Tampilan utama program ditunjukkan oleh gambar 1 dibawah ini:

```

=====
=      A* Pathfinder      =
=====
Masukkan nama graf input (tanpa ekstensi file): MapBuahbatu

1. Metro Indah Mall
2. Jl Soekarno Hatta
3. Jembatan Sungai Cicadas
4. Yamaha Service Center
5. Jl Jupiter Barat 26
6. Taman S2
7. Taman Jupiter
8. Jl Jupiter Barat Utama
9. Jl Raya Cirebon Bandung
Masukkan nama simpul awal: Metro Indah Mall
Masukkan nama simpul tujuan: Jl Jupiter Barat Utama

Jarak terpendek dari Metro Indah Mall ke Jl Jupiter Barat Utama: 387.46728065957495 meter

Lintasan:
Metro Indah Mall → Yamaha Service Center → Taman Jupiter → Jl Jupiter Barat Utama

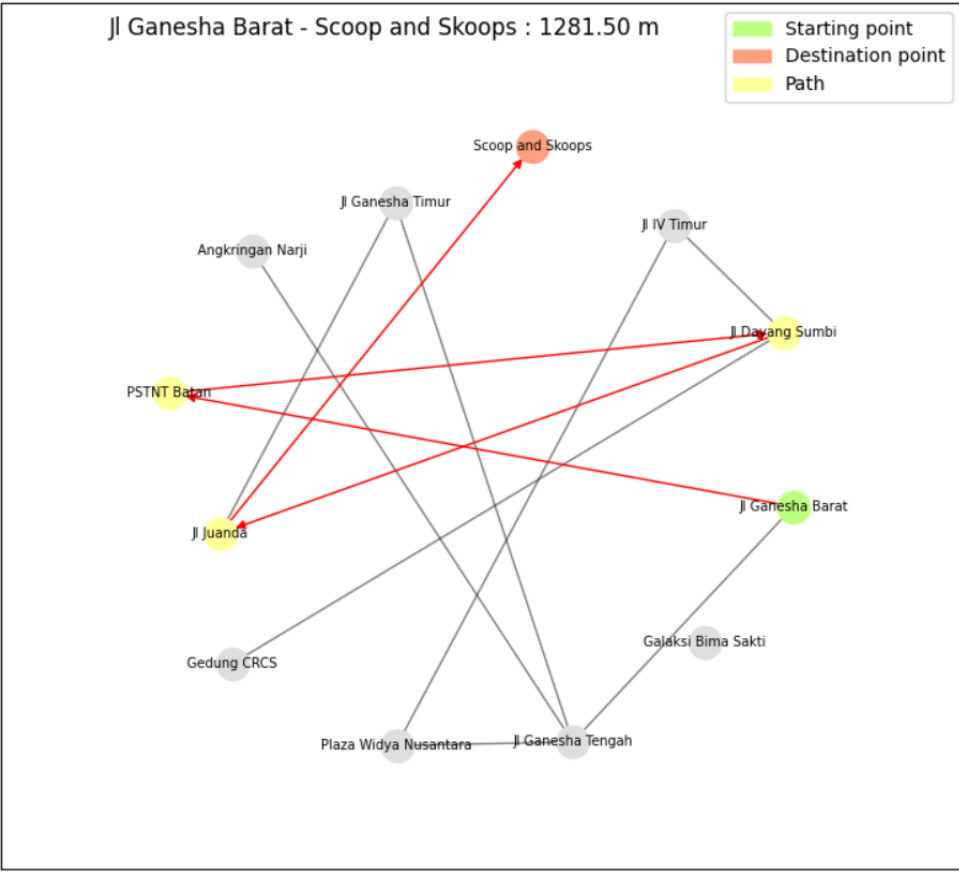
Apakah anda ingin membaca file lain? (Y/N) : y

```

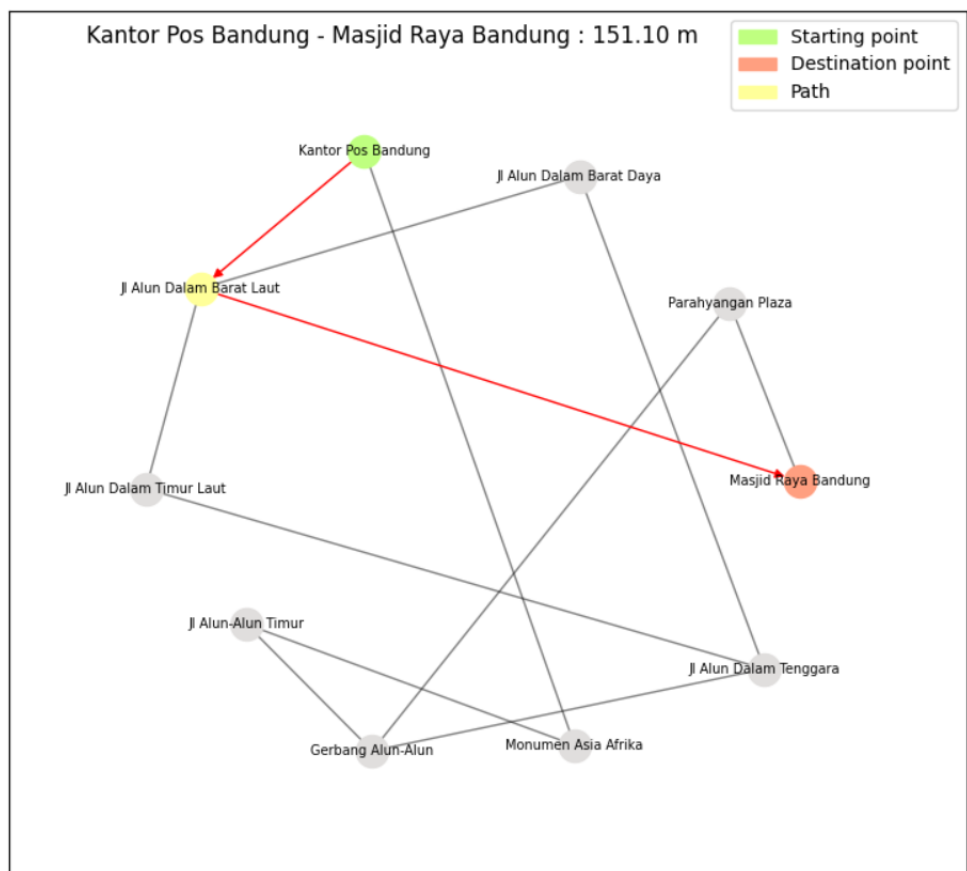
Gambar 1. Tampilan utama program

Berikut ini merupakan tangkapan layar hasil uji jarak terdekat dari beberapa peta program dan outputnya:

Tabel 2. *Screenshot* hasil peta pada berbagai variasi test case

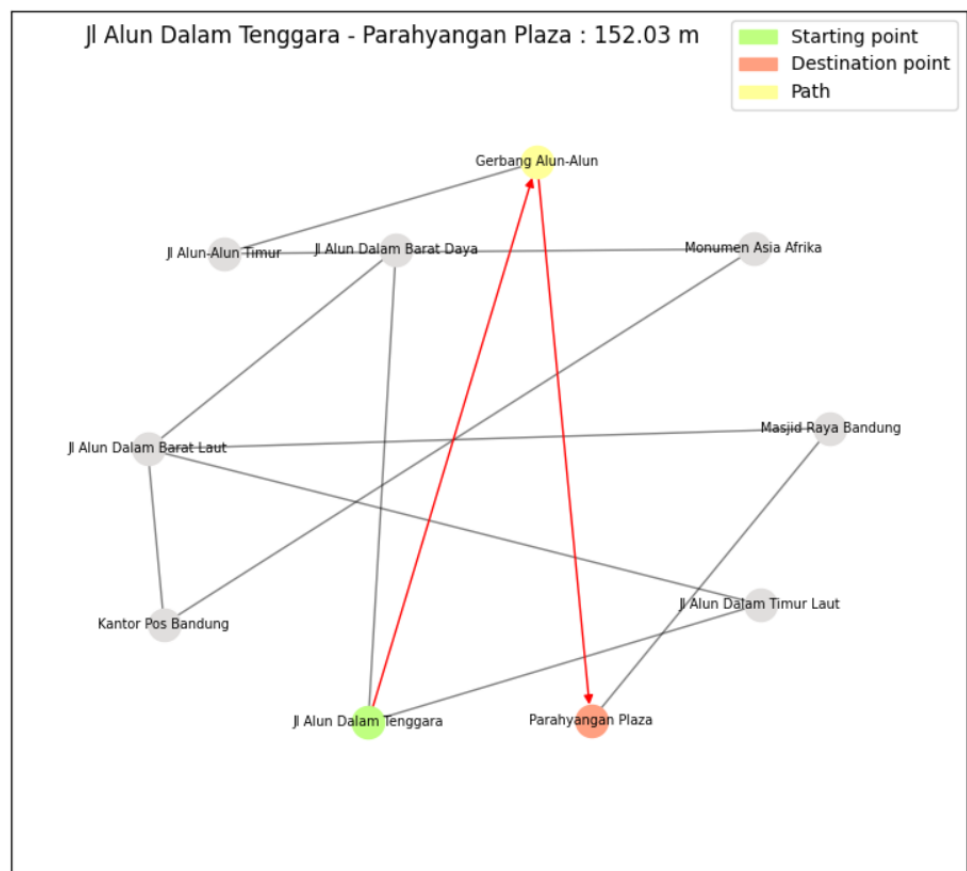
Test Case	Hasil
<p>MapITB.txt  Start : Jl Ganesha Barat  End : Scoop and Scoops</p>	 <p>The screenshot displays a map with a network of roads. A path is highlighted in red, starting from a green circle at 'Jl Ganesha Barat' and ending at an orange circle at 'Scoop and Scoops'. The path consists of several segments connecting various nodes. A legend in the top right corner identifies the green circle as the 'Starting point', the orange circle as the 'Destination point', and the red line as the 'Path'. The map title is 'Jl Ganesha Barat - Scoop and Scoops : 1281.50 m'. Other labeled nodes include 'Angkringan Narji', 'Jl Ganesha Timur', 'Jl IV Timur', 'Jl Dayang Sumbi', 'PSTNT Batan', 'Jl Juanda', 'Gedung CRCS', 'Plaza Widya Nusantara', 'Jl Ganesha Tengah', and 'Galaksi Bima Sakti'.</p>
<p>MapITB.txt  Start : PSTNT Batan  End : Galaksi Bima Sakti</p>	<p>Tidak ada jalur yang menghubungkan PSTNT Batan dan Galaksi Bima Sakti</p>

MapAlunBandung.txt  
Start : Kantor Pos Bandung  
End : Masjid Raya Bandung

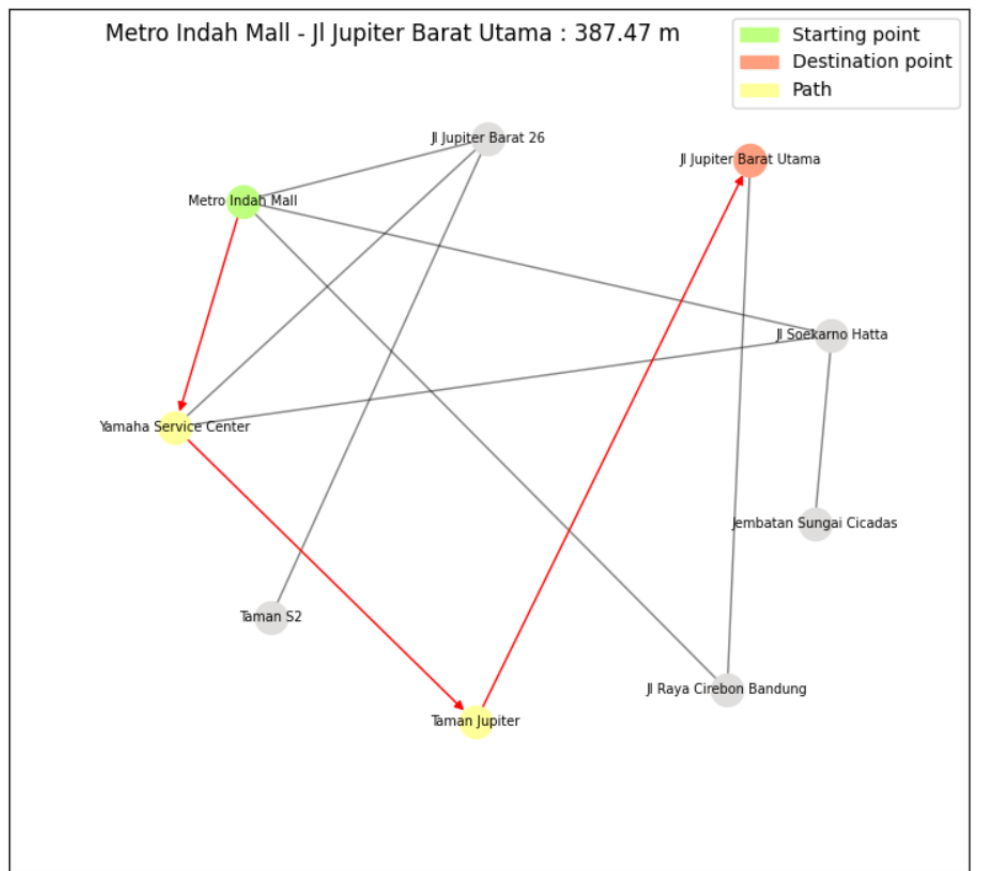




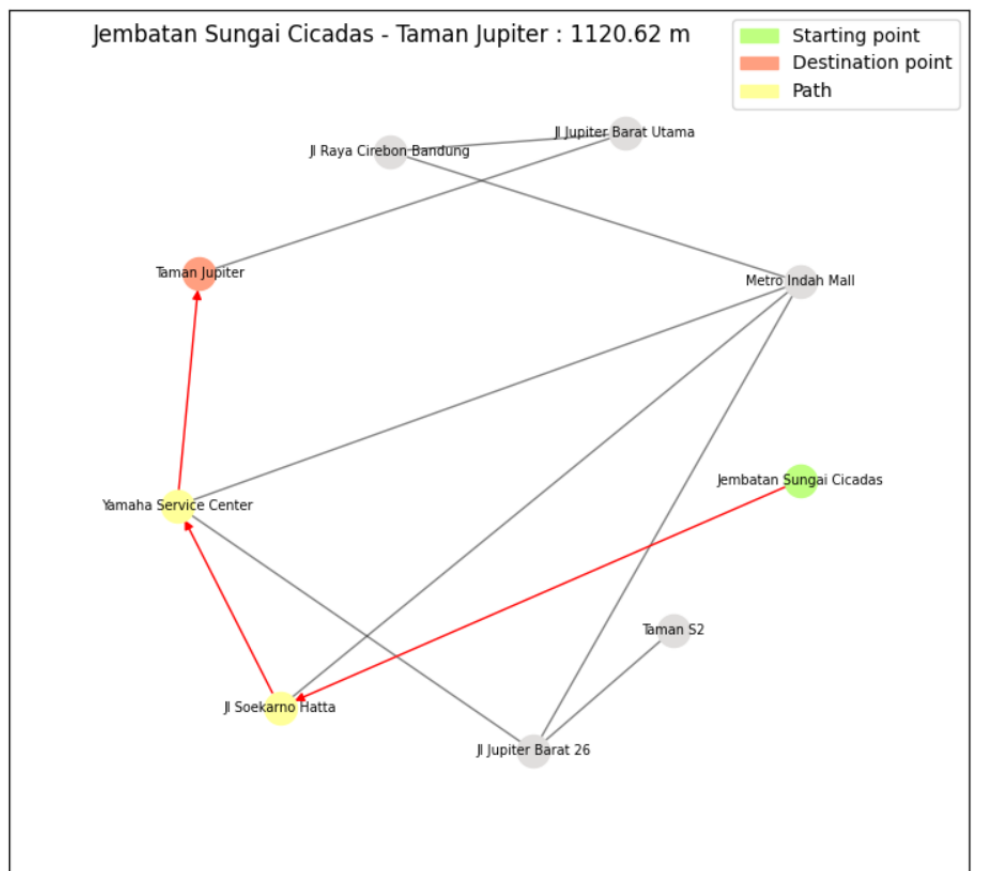
MapAlunBandung.txt  
Start : Jl Alun Dalam  
Tenggara  
End : Parahyangan Plaza



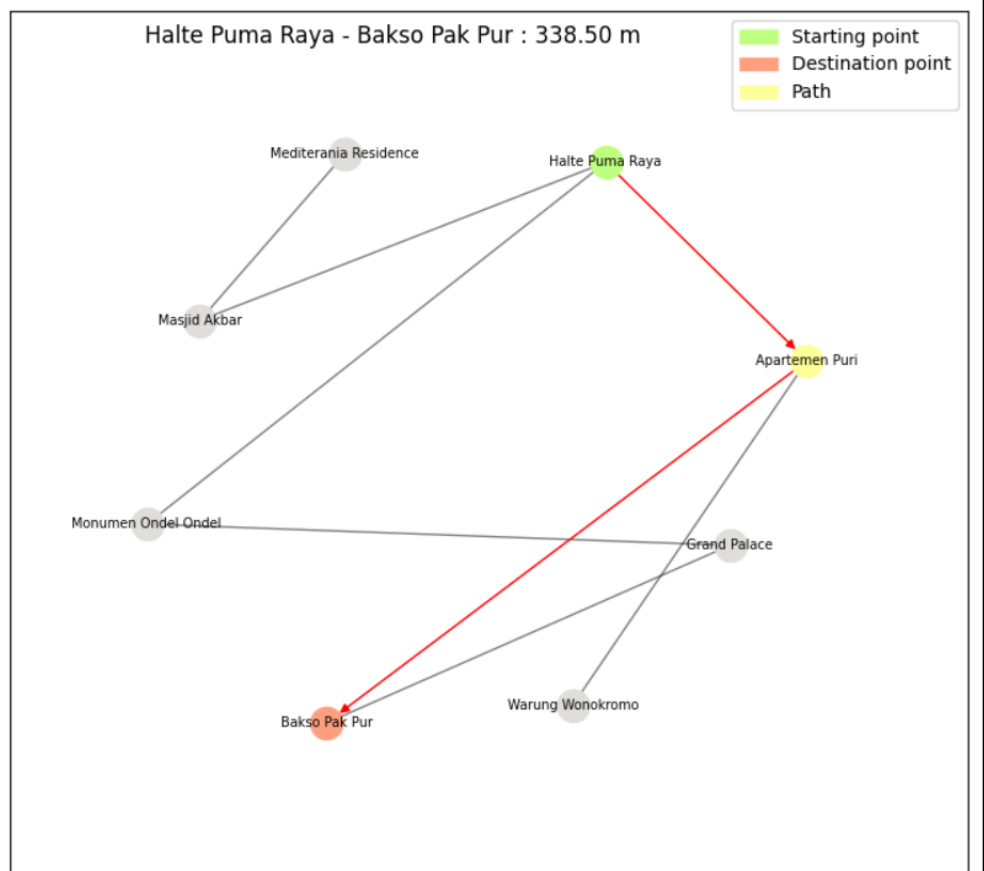
MapBuahbatu.txt  
Start : Metro Indah Mall  
End : Jl Jupiter Barat Utama

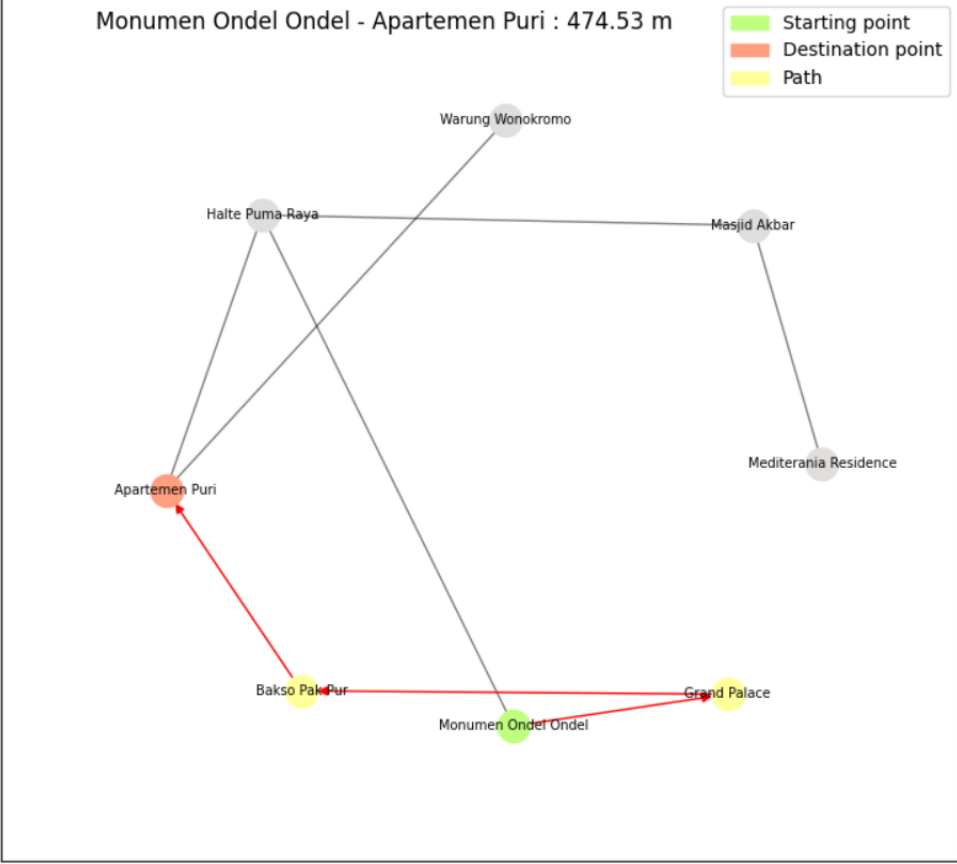


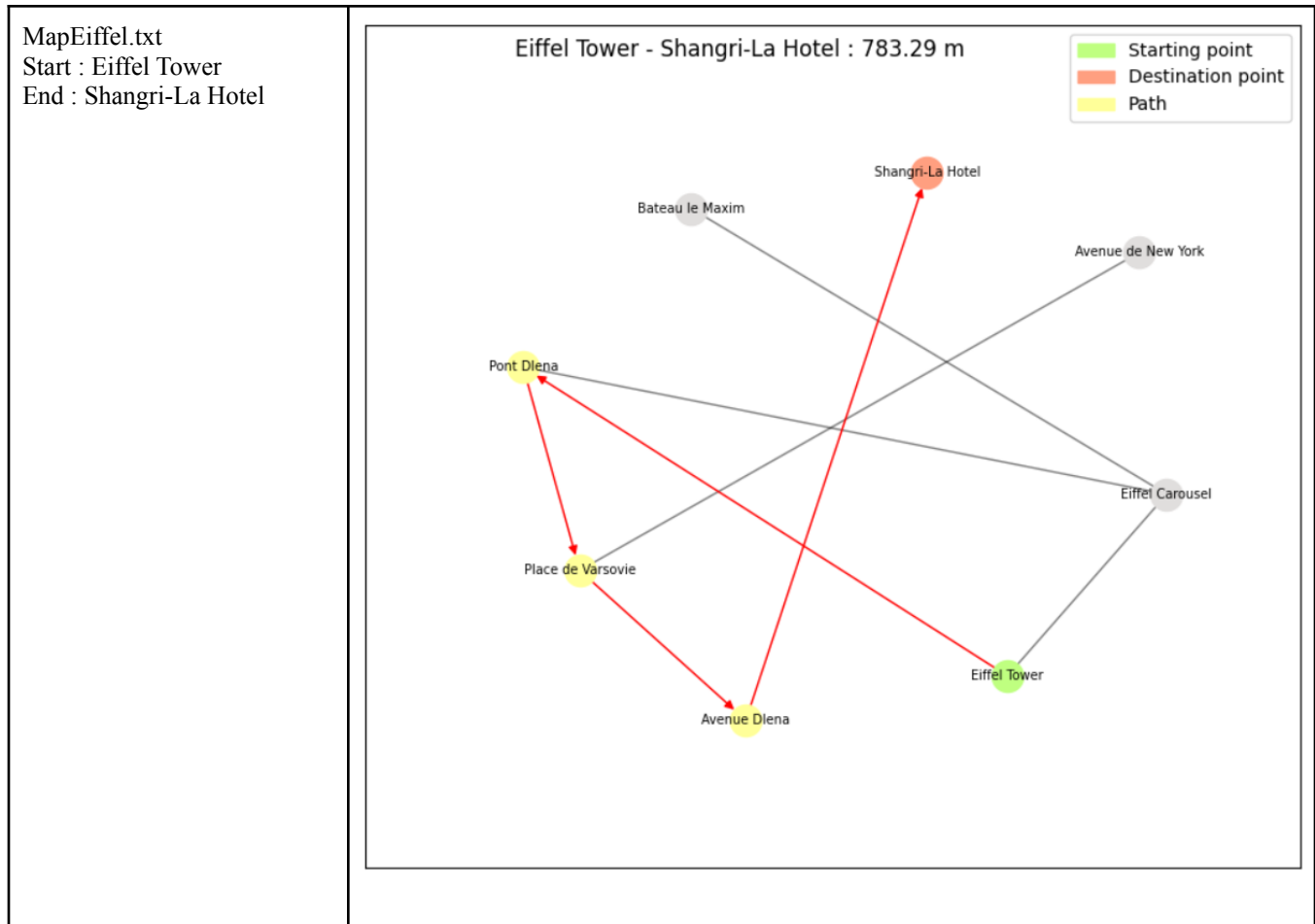
MapBuahbatu.txt  
Start : Jembatan Sungai  
Cicadas  
End : Taman Jupiter



MapKemayoran.txt  
Start : Halte Puma Raya  
End : Bakso Pak Pur



<p>MapKemayoran.txt  Start : Monumen Ondel Ondel  End : Apartemen Puri</p>	<div data-bbox="508 216 1459 1075"> <p>Monumen Ondel Ondel - Apartemen Puri : 474.53 m</p>  </div>
<p>MapDisconnected.txt  Start : Patung Merlion  End : Gedung Empire States</p>	<p>Tidak ada jalur yang menghubungkan Patung Merlion dan Gedung Empire States</p>



#### IV. Alamat Kode

Berikut ini merupakan alamat kode program yang disimpan pada repository GitHub:

<https://github.com/dionisiusdh/tucil-3-stima>

#### V. Evaluasi Program

Berikut ini merupakan tabel hasil evaluasi program:

Tabel 3. Hasil evaluasi program

<b>Fitur Program</b>	<b>Status</b>
1. Program dapat menerima input graf	V
2. Program dapat menghitung lintasan terpendek	V
3. Program dapat menampilkan lintasan terpendek serta jaraknya	V
4. Bonus: Program dapat menerima input peta dengan Google Map API dan menampilkan peta	