IMPLEMENTASI ALGORITMA A* UNTUK MENENTUKAN LINTASAN TERPENDEK

Laporan Tugas Kecil III IF2211 Strategi Algoritma

Semester 2 Tahun Akademik 2020/2021

Disusun Oleh:

Rizky Anggita S. Siregar **13519132**

Wilson Tandya **13519209**

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

A. Kode Program

main.py

```
import os
import folium
from folium import plugins
from Graph import Graph
from Utility import add_graph_from_txt, SearchIdxNode

# A* Algorithm for Shortest Path Finding
# Author    :
#               Rizky Anggita S Siregar - 13519132
#               Wilson Tandya           - 13519228
# Date      :   04-03-2021


# Install this dependency if it's not installed
# Install numpy -> pip3 install numpy
# Install matplotlib -> pip3 install matplotlib
# Install networkx -> pip3 install networkx
# Install decorator v4.4.2 -> pip install decorator==4.4.2
# Install folium -> pip3 install folium

# ------Main Program-------

filename = input(str("Masukkan nama file: "))
filename = filename+".txt"
a = os.path.abspath(os.curdir)

if os.name=='nt':
    file_path = os.path.join("..\\test", filename)
else:
    file_path = os.path.join(a, "test", filename)

f = open(file_path, "r")

nNodes = int(f.readline())
nodeCoordinate = []
g2 = Graph(nNodes)
nodeCoordinate = add_graph_from_txt(g2, nodeCoordinate, f, nNodes)
```

```
daftarNode = g2.nodes

print("Daftar Titik yang dapat dikunjungi: ")
print(daftarNode)

From = input(str("Asal   : "))
To = input(str("Tujuan   : "))

while (From not in daftarNode or To not in daftarNode):
    print("Node tidak terdapat pada graph, silahkan input ulang!")
    From = input(str("Asal:"))
    To = input(str("Tujuan:"))

found, solusi, gn = g2.AStar(From, To, nodeCoordinate)

if not found:
    print("Lintasan tidak dapat ditemukan!")
else:
    print("Solusi A*: ", end = "")
    for i in range (len(solusi)):
        if (i != len(solusi) - 1):
            print(solusi[i], end = " → ")
        else:
            print(solusi[i])
    print("Jarak tempuh: ", gn[g2.nodes.index(To)], " meter")
    g2.visualize_graph(solusi)

    # Visualizing route with Folium
    # Use the Jupyter Notebook version of this program to see this

    #Koordinat lokasi dari file eksternal
    coordinates = []
    for i in (nodeCoordinate):
        coordinates.append(i)

    name = []
    for i in (g2.nodes):
        name.append(i)

    latitude = []
    longitude = []
```

```
    for i in range(len(coordinates)):
        latitude.append(coordinates[i][0])
        longitude.append(coordinates[i][1])

    #Koordinat jalur shortest path
    mapHasil = []
    for i in range (len(solusi)):
        mapHasil.append(nodeCoordinate[SearchIdxNode(solusi[i], g2)])


    visualisasiMap = folium.Map(location=coordinates[0], zoom_start=16)
    for index,lat in enumerate(latitude):
        folium.Marker([lat,
        longitude[index]],
        popup=('{} \n'.format(name[index])),
        icon = folium.Icon(color='blue'), tooltip=name[index]).add_to(visualisasiMap)
        #Buat Ant Path dari jalur yang ditempuh
        plugins.AntPath(locations=mapHasil,weight=5, color =
"green").add_to(visualisasiMap)
    #Menampilkan visualisasi peta pada jupyter notebook, ada animasi jalannya :)
    visualisasiMap
```

<End Of main.py>

main.ipynb (sama dengan main.py, hanya untuk visualisasi rute terpendek)

Graph.py (included A* and visualize_graph methods)

```
import queue
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
from Utility import haversin, rad

# Self-made Graph Class
# Graph implemented with list of nodes and adjacency weighted matrix
class Graph():
    def __init__(self, nNodes):
        self.nodes = []
        self.adj_matrix = []
```

```
        # Initialize Weighted Graph Adjacency Matrix
        for i in range(nNodes):
            col = []
            for j in range(nNodes):
                col.append(0)
            self.adj_matrix.append(col)

    def add_node(self, node):
        self.nodes.append(node)

    def add_edge(self, u, v, weight):
        index_u = self.nodes.index(u)
        index_v = self.nodes.index(v)
        self.adj_matrix[index_u][index_v] = weight

    def print_node(self):
        for node in self.nodes:
            print(node, end=" ")
        print()

    def print_graph(self):
        nNodes = len(self.nodes)
        print("  " *len(self.nodes[0]), end="" )
        self.print_node()
        for i in range (nNodes):
            for j in range (nNodes):
                #if(j==0):
                #    print(self.nodes[i], end=" ")
                print(self.adj_matrix[i][j], end=" \t")
            print()

    # A* method
    def AStar(self, From, To, nodeCoordinate):
        def construct_path(cameFrom, From, To, self):
            path = []
            idxFrom = self.nodes.index(From)
            idxTo = self.nodes.index(To)

            i = idxTo
            path.append(self.nodes[idxTo])
```

```
            while(i != idxFrom):
                path.append(cameFrom[i])
                i = self.nodes.index(cameFrom[i])

            path.reverse()
            return path

        def isInPrioQueue(prioQueue, node):
            for item in prioQueue.queue:
                if(item[1] == node):
                    return True
            return False

        found = False
        openSet = queue.PriorityQueue()
        idxFrom = self.nodes.index(From)
        idxTo = self.nodes.index(To)

        f2 = 0 + haversin(nodeCoordinate[idxFrom], nodeCoordinate[idxTo])
        openSet.put((f2, From))

        # // For node n, gn[n] is the cost of the cheapest path from start to n currently
known.
        gn = []

        # // For node n, fn[n] := gn[n] + h(n). fn[n] represents our current best guess
as to
        # // how short a path from start to finish can be if it goes through n.
        fn = []
        cameFrom = []

        for node in self.nodes:
            if (node==From):
                gn.append(0)
                fn.append(haversin(nodeCoordinate[idxFrom], nodeCoordinate[idxTo]))
            else:
                gn.append(99999)
                fn.append(99999)
            cameFrom.append("")

        while (not openSet.empty()):
```

```
            current = openSet.get()
            currentIdx = self.nodes.index(current[1])

            if(current[1] == To):
                print("FOUND\n")
                found = True
                break

            # Foreach neighbour of current
            # neighbour is every node that connected with current node
            # with weight > 0
            for i in range(len(self.nodes)):
                if (self.adj_matrix[currentIdx][i] > 0):
                    #  tentative_gn is the distance from start to the neighbor through
current
                    # d jarak dari node current ke tetangganya (weigthnya)
                    d = self.adj_matrix[currentIdx][i]
                    tentative_gn = gn[currentIdx] + d

                    # kalau kita mengunjungi node yang sama dua kali, kita cek apakah gn
node ini
                    # lebih pendek jika dibandingkan dengan tentative_gn, yaitu
dikunjungi melalui node lain

                    if (tentative_gn < gn[i]):
                        cameFrom[i] = current[1]
                        gn[i] = tentative_gn
                        titikNeighbour = nodeCoordinate[i]
                        hn = haversin(titikNeighbour, nodeCoordinate[idxTo])
                        fn[i] = gn[i] + hn

                        if not (isInPrioQueue(openSet, self.nodes[i])):
                            openSet.put((fn[i], self.nodes[i]))
        if not openSet.empty():
            return True, construct_path(cameFrom, From, To, self), gn
        else:
            return False, False, False

    def visualize_graph(self, solusi):
        g2 = self
        adj_np = np.array(g2.adj_matrix)
```

```
        namaNode = g2.nodes
        mylabels = {}
        for i in range (len(namaNode)):
            mylabels[i] = namaNode[i]

        G = nx.from_numpy_matrix(adj_np, create_using=nx.DiGraph)
        color_map = []
        edge_colors = []
        solusi_idx = []

        for i in range (len(solusi)):
            a = g2.nodes.index(solusi[i])
            solusi_idx.append(a)

        for node in G.nodes:
            if node in solusi_idx:
                color_map.append("green")
            else:
                color_map.append("red")

        for edges in G.edges:
            if(edges[0] in solusi_idx and (edges[1] in solusi_idx) ):
                edge_colors.append("green")
            else:
                edge_colors.append("red")

        layout = nx.spring_layout(G)
        nx.draw(G, layout, node_size=1000, width=5, node_color = color_map, edge_color =
edge_colors, labels=mylabels, with_labels=True, arrows=False)
        jarak = nx.get_edge_attributes(G, "weight")
        nx.draw_networkx_edge_labels(G, pos=layout, edge_labels=jarak)
        plt.show()
```

<End Of Graph.py>

Utility.py

```
import math

def add_graph_from_txt(g, nodeCoordinate, file, nNodes):
   nodeCoordinate = []

   for i in range (nNodes):
       temp = file.readline()
       temp = temp.rsplit(" ")
       nodeCoordinate.append((float(temp[0]), float(temp[1])))
       g.add_node(temp[2].rstrip("\n"))

   for i in range (nNodes):
       j = 0
       temp = file.readline().rsplit(" ")

       for weight in temp:
           g.add_edge(g.nodes[i], g.nodes[j], float(temp[j].rstrip("\n")))
           j+= 1

   return nodeCoordinate

def rad (x):
   return x*math.pi / 180

def haversin (p1, p2):
   R = 6378137 #Radius Bumi dalam meter
   dLat = rad(p2[0] - p1[0])
   dLong = rad(p2[1] - p1[1])
   a = (math.sin (dLat / 2))**2 + math.cos (rad(p1[0])) * math.cos (rad(p2[0])) *
(math.sin(dLong / 2))**2
   c = 2 * math.atan2(math.sqrt(a), math.sqrt(1-a))
   res = R * c
   return res

def SearchIdxNode(N, g):
   for i in range(len(g.nodes)):
       if (N == g.nodes[i]):
           return i
```

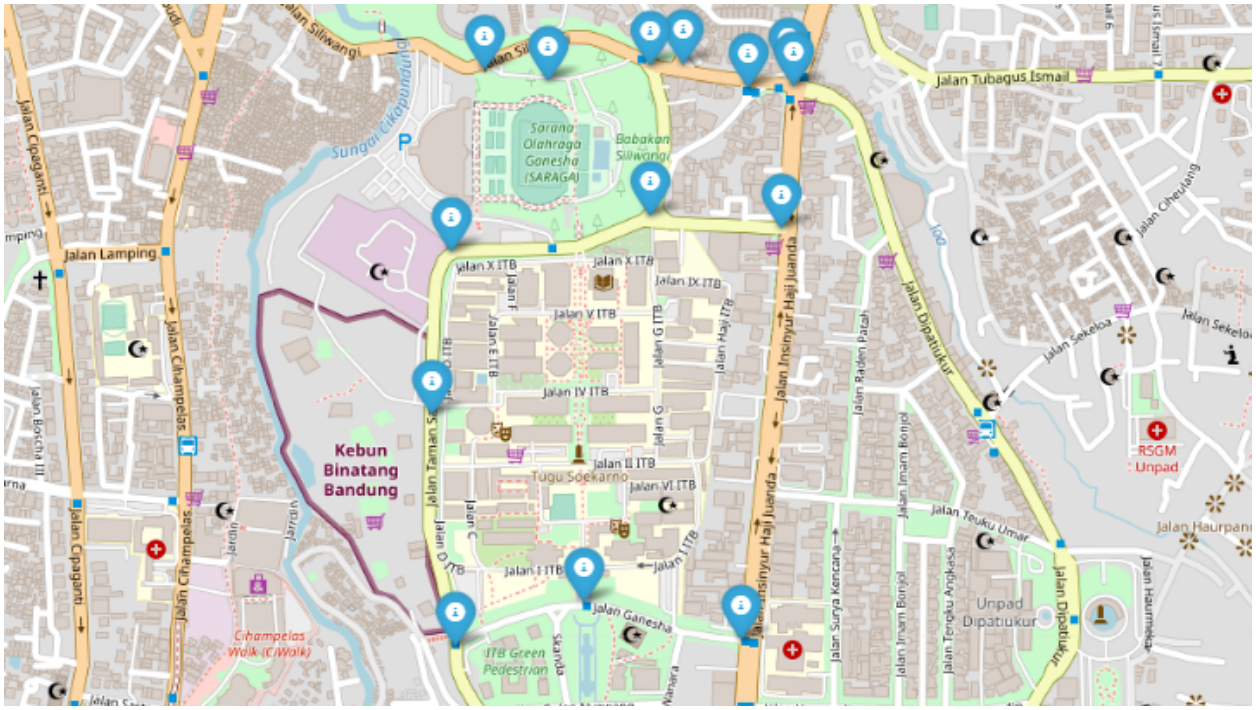
<End Of Utility.py>

B. Peta / Graf Input

1. Peta Jalan Sekitar Kampus ITB

```
15
-6.893185 107.610447 GerbangITBGanesa
-6.893863 107.608447 SimpanganTamansariGanesa
-6.893753 107.612889 SimpanganGanesaJuanda
-6.890277 107.608071 BonBin
-6.887770 107.608370 TikunganTamfest
-6.885217 107.613703 SimpanganEmpatMcD
-6.885004 107.613614 McDDago
-6.887219 107.611475 PertigaanTamansariDayangSumbi
-6.887393 107.613511 SimpanganDayangSumbiJuanda
-6.884954 107.608891 SimpanganSiliwangiSaraga
-6.885132194268106 107.6098881898018 SaragaITB
-6.884898 107.611465 PertigaanTamansariSiliwangi
-6.885239 107.612999 PertigaanSiliwangiSumur
-6.884859 107.611981 PertigaanSiliwangiSangkuriang
-6.882108 107.611705 LIPICisitu
0 240 280 0 0 0 0 0 0 0 0 0 0 0 0
240 0 0 400 0 0 0 0 0 0 0 0 0 0 0
280 0 0 0 0 0 0 0 700 0 0 0 0 0 0
0 400 0 0 290 0 0 0 0 0 0 0 0 0 0
0 0 0 290 0 0 0 350 0 0 0 0 0 0 0
0 0 0 0 0 0 27 0 250 0 0 0 76 0 0
0 0 0 0 0 27 0 0 0 0 0 0 0 0 0
0 0 0 0 350 0 0 0 270 0 0 270 290 0 0
0 0 700 0 0 250 0 270 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 110 300 0 0 0
0 0 0 0 0 0 0 0 0 110 0 0 0 0 0
0 0 0 0 0 0 0 270 0 300 0 0 0 77 0
0 0 0 0 0 76 0 290 0 0 0 0 0 120 0
0 0 0 0 0 0 0 0 0 0 0 77 120 0 350
0 0 0 0 0 0 0 0 0 0 0 0 0 350 0
```

File ITB.txt

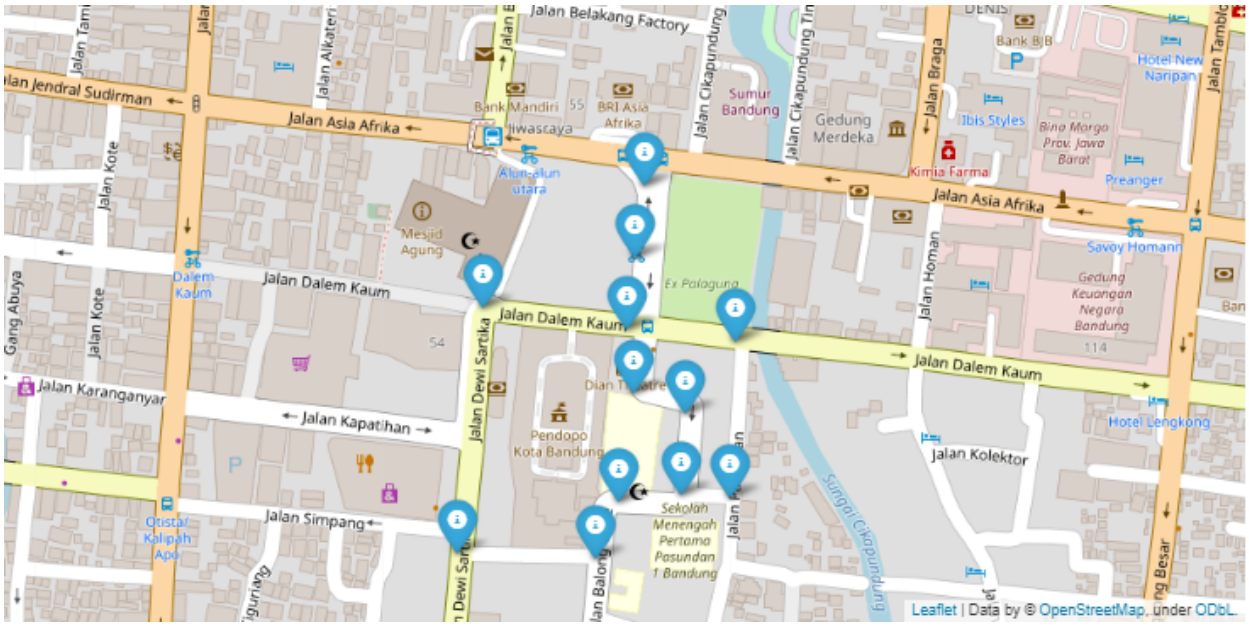


Gambar 1. Peta Jalan Sekitar Kampus ITB

2. Peta Jalan Sekitar Alun-Alun Bandung

```
12
-6.921436394616056 107.607661484276 MasjidAgung
-6.922000878600341 107.60759174684199 AlunAlun
-6.922565361909443 107.60752200940799 PersimpanganAlunAlun
-6.922389626613821 107.60639011720698 PlazaParahyangan
-6.922655892190464 107.60838031628494 PesimpanganPasundan
-6.923058327177523 107.60757534366579 TokoJajang
-6.923225014863285 107.60798560651199 JalanBalonggede
-6.923856416489483 107.60794726369312 Kelurahan
-6.923913663427083 107.60745882981979 SMAPasundan1
-6.923876386354761 107.60833859437177 BandarGantungan
-6.924356412057512 107.607277410052 AhliGiziZai
-6.924315287379696 107.60618449351989 UbiCilembuLegit
0 62 0 0 0 0 0 0 0 0 0 0
62 0 65 0 0 0 0 0 0 0 0 0
0 65 0 130 98 65 0 0 0 0 0 0
0 0 130 0 0 0 0 0 0 0 0 210
0 0 98 0 0 0 0 0 0 130 0 0
0 0 65 0 0 0 50 0 0 0 0 0
0 0 0 0 0 50 0 72 0 0 0 0
0 0 0 0 0 0 72 0 58 43 0 0
0 0 0 0 0 0 0 58 0 0 61 0
0 0 0 0 130 0 0 43 0 0 0 0
0 0 0 0 0 0 0 0 61 0 0 120
0 0 0 210 0 0 0 0 0 0 120 0
```

File AlunAlun.txt

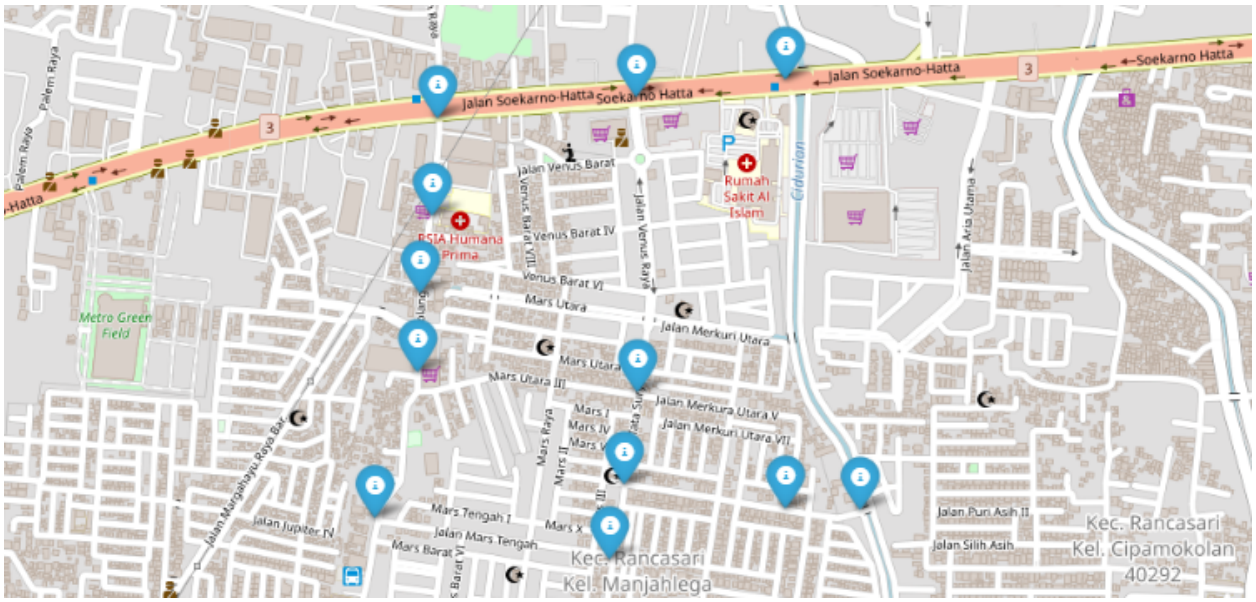


Gambar 2. Peta Jalan Sekitar Alun-Alun Bandung

3. Peta Jalan Sekitar Buahbatu

```
12
-6.939256723596976 107.66387290534682 SimpaganRancabolang
-6.940805192424694 107.66378302440877 RSIAHumanaPrima
-6.9420323752968605 107.66358094904432 Amanda
-6.943288973680201 107.66354267262001 Borma
-6.945621118273786 107.66286575457511 WarungBuAde
-6.946283824560161 107.66664379805458 RMAndalus
-6.943623448495856 107.66708899211018 MartabakJuragan
-6.945470503078475 107.66946595101516 GreenHousePC
-6.9454880331149775 107.67067335008035 SimpanganMerkuri
-6.9389147591021665 107.66707536618159 KFC
-6.938636121159263 107.66946126330632 RSAIBandung
-6.945075746080642 107.66687089909222 KopiSoe
0 170 0 0 0 0 0 0 0 350 0 0
170 0 150 0 0 0 0 0 0 0 0 0
0 150 0 150 0 0 0 0 0 0 0 0
0 0 150 0 270 0 450 0 0 0 0 0
0 0 0 270 0 450 0 0 0 0 0 0
0 0 0 0 450 0 0 0 0 0 0 130
0 0 0 450 0 0 0 0 0 550 0 160
0 0 0 0 0 0 0 0 130 0 0 290
0 0 0 0 0 0 0 0 130 0 0 750 0
350 0 0 0 0 0 550 0 0 0 300 0
0 0 0 0 0 0 0 0 750 300 0 0
0 0 0 0 0 130 160 290 0 0 0 0
```

File BuahBatu.txt

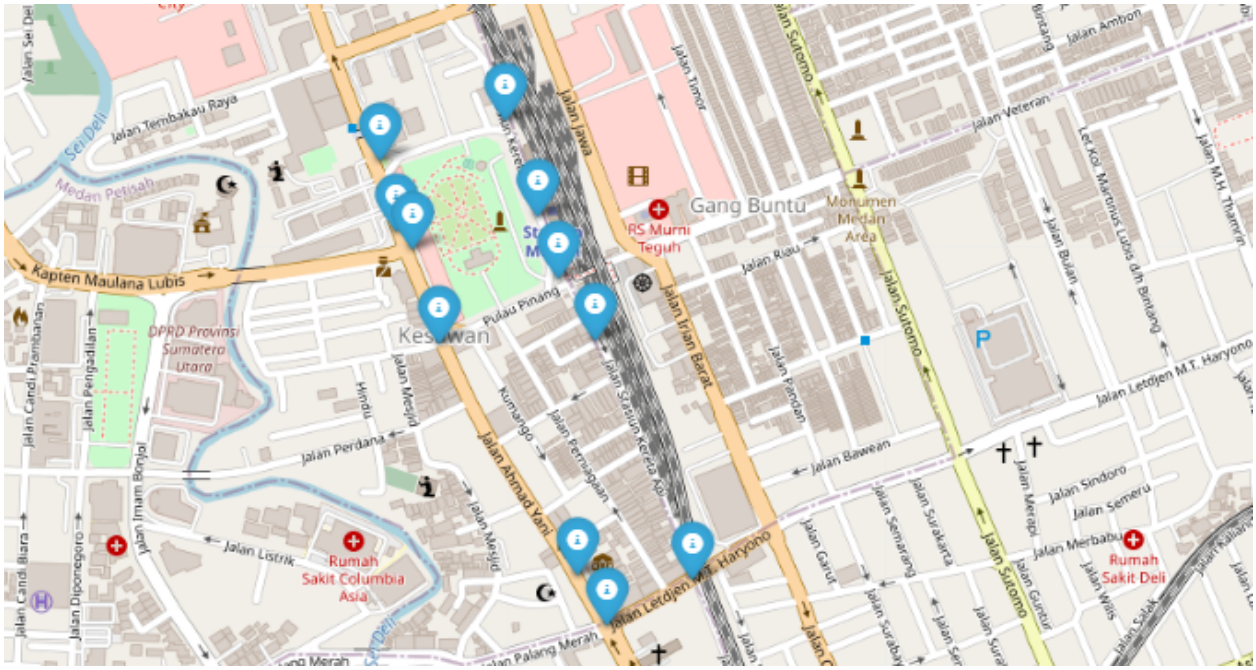


Gambar 3. Peta Jalan Sekitar Buahbatu

4. Peta Jalan Sebuah Kawasan di Kota Medan

```
11
3.590135 98.677815 LapanganMerdeka
3.590390 98.677569 BalaiKota
3.591422 98.677341 TuguKM0
3.592004 98.679166 SimpanganKeretaApi
3.590604 98.679632 StasiunMedan
3.589701 98.679933 SimpanganTitiGantung
3.588809 98.680455 PajakIkanLama
3.585338 98.681876 SimpanganPalangMerah
3.584673 98.680642 SimpanganPemuda
3.585412 98.680209 TjongAFie
3.588771 98.678195 SimpanganPulauPinang
0 37 0 0 0 0 0 0 0 160
37 0 170 0 0 0 0 0 0 0
0 170 210 0 0 0 0 0 0 0
0 0 210 0 170 0 0 0 0 0
0 0 0 170 0 110 0 0 0 0
0 0 0 0 110 0 130 0 0 0 220
0 0 0 0 0 130 0 450 0 0 0
0 0 0 0 0 0 450 0 160 0 0
0 0 0 0 0 0 0 160 0 97 0
0 0 0 0 0 0 0 0 97 0 450
160 0 0 0 0 220 0 0 0 450 0
```

File Medan.txt

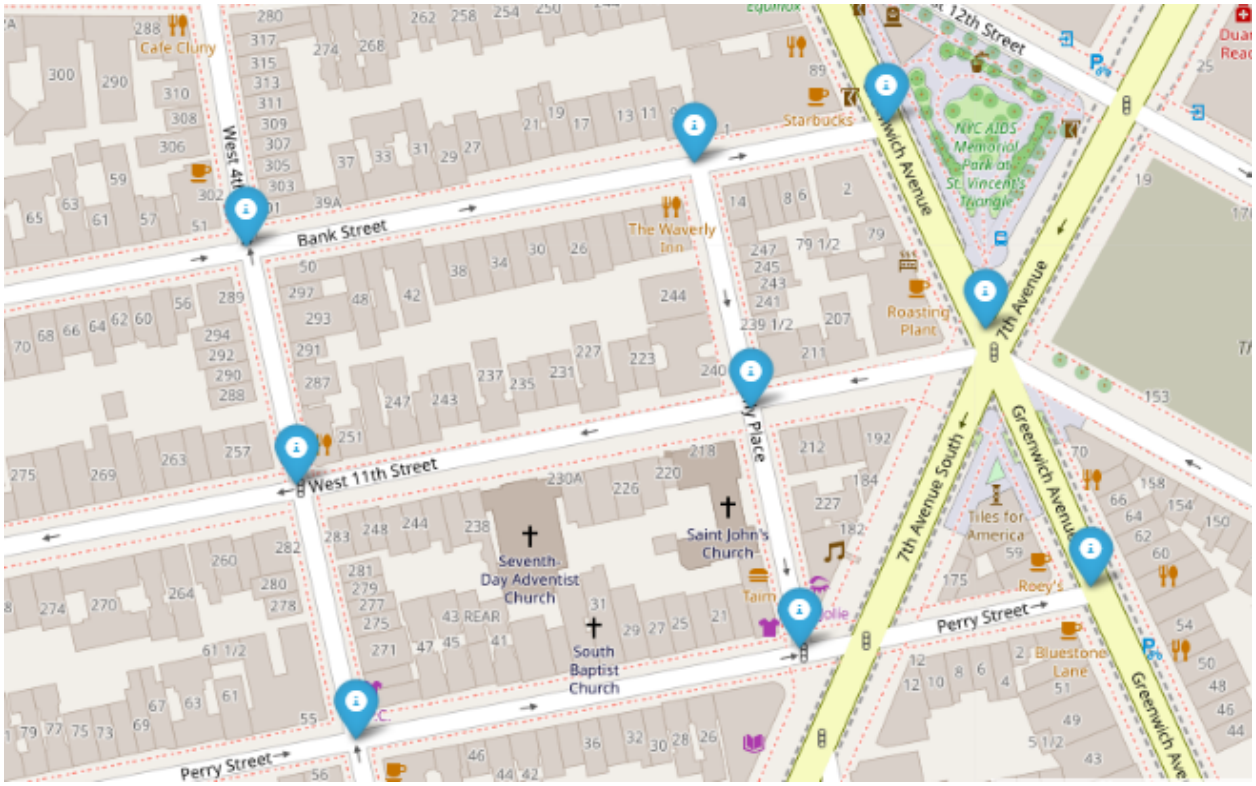


Gambar 4. Peta Jalan Sebuah Kawasan di Kota Medan

5. Peta Jalan Sebuah Kawasan di Kota Manhattan

```
9
40.73684998848145 -74.00372144640423 LeFanion
40.73707234146876 -74.00214573357314 WaverlyInn
40.73717626706666 -74.00147270643275 NYCMemorialPark
40.736630490832404 -74.00112138815496 ZazzyPizza
40.73641555807072 -74.00194900408943 PerempatanWaverly
40.736212370416474 -74.00354672205468 Tartine
40.73553692914271 -74.00333809516732 APCStore
40.73577836425789 -74.00177908333627 LampuMerah
40.735942356562184 -74.00075415308669 Roey
0 140 0 0 0 72 0 0 0
140 0 58 0 75 0 0 0 0
0 58 0 65 0 0 0 0 0
0 0 65 0 67 0 0 92 80
0 75 0 67 0 130 0 90 0
72 0 0 0 130 0 76 0 0
0 0 0 0 0 76 0 150 0
0 0 0 92 90 0 150 0 63
0 0 0 80 0 0 0 63 0
```

File Manhattan.txt

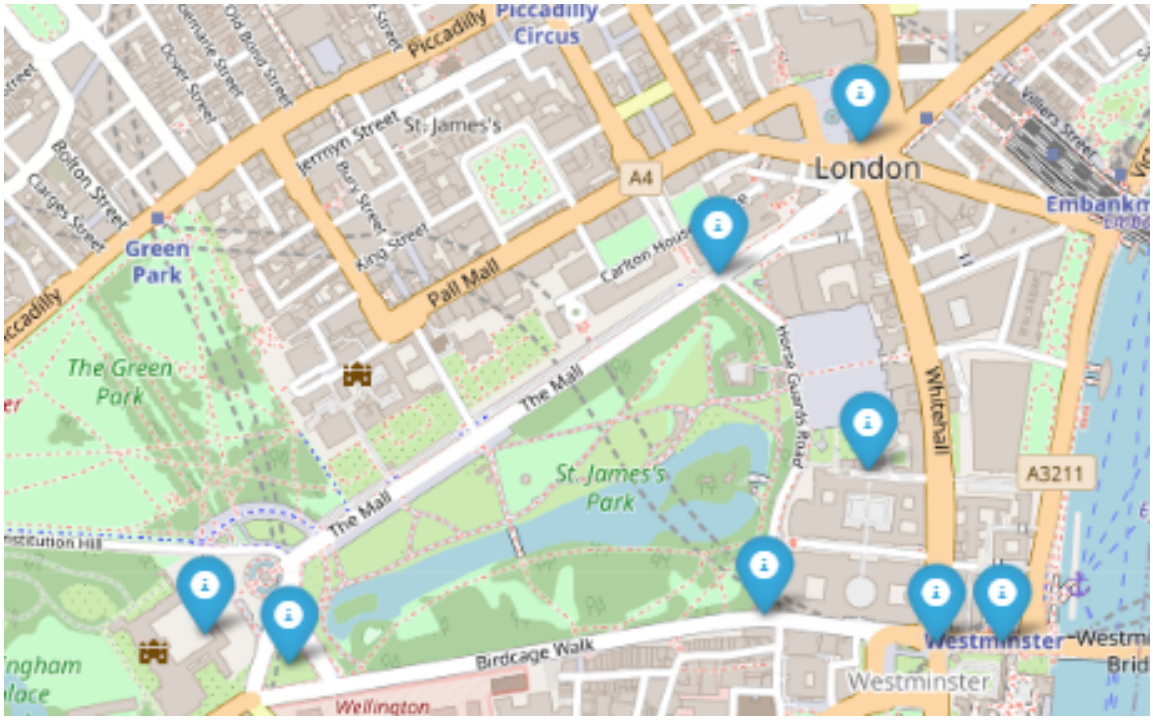


Gambar 5. Peta Jalan Sebuah Kawasan di Kota Manhattan

6. Peta Jalan Sebuah Kawasan di Kota London

```
8
51.500970519986815 -0.12474448210377943 BigBen
51.500982 -0.126183 PerempatanParliamentSquare
51.503256 -0.127642 10DowningStreet
51.50763968631513 -0.12779566240939946 TragalgarSquare
51.505887 -0.130833 PertigaanTheMall
51.50134197705323 -0.1298556589393768 PertigaanBirdcage
51.500674 -0.140084 PertigaanBuckingham
51.50107775031579 -0.14188409136738678 BuckinghamPalace
0 96 0 0 0 0 0 0
96 0 0 850 0 300 0 0
0 0 0 0 0 0 0 0
0 850 0 0 350 0 0 0
0 0 0 350 0 550 0 1100
0 300 0 0 550 0 700 0
0 0 0 0 0 700 0 100
0 0 0 0 1100 0 100 0
```

File London.txt



Gambar 6. Peta Jalan Sebuah Kawasan di Kota London

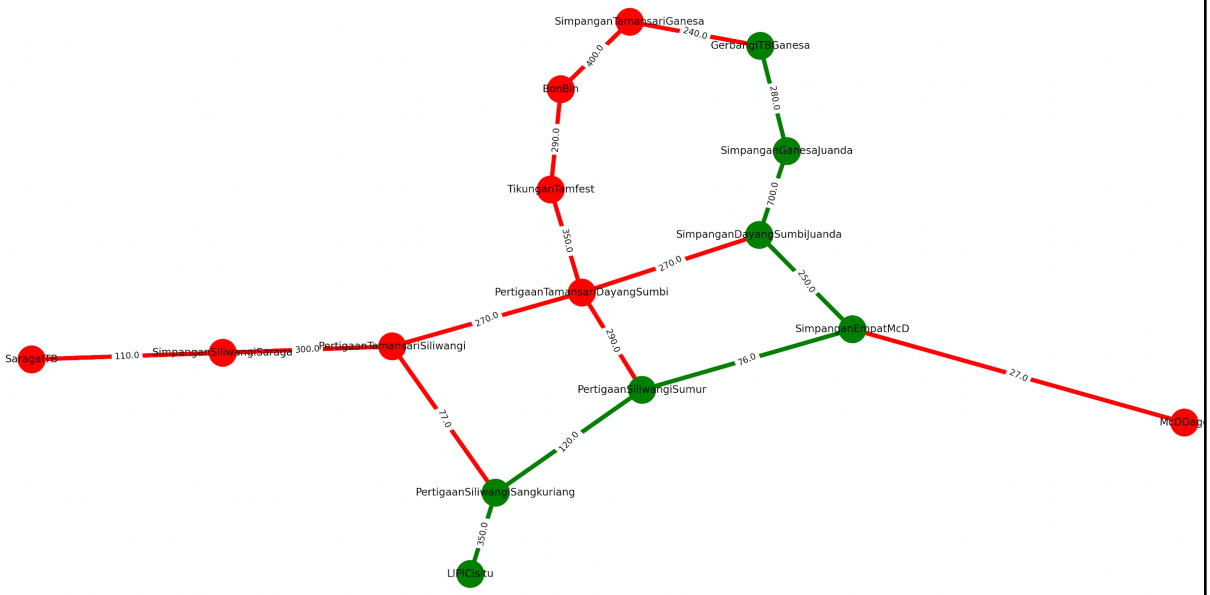
C. Screenshot Peta

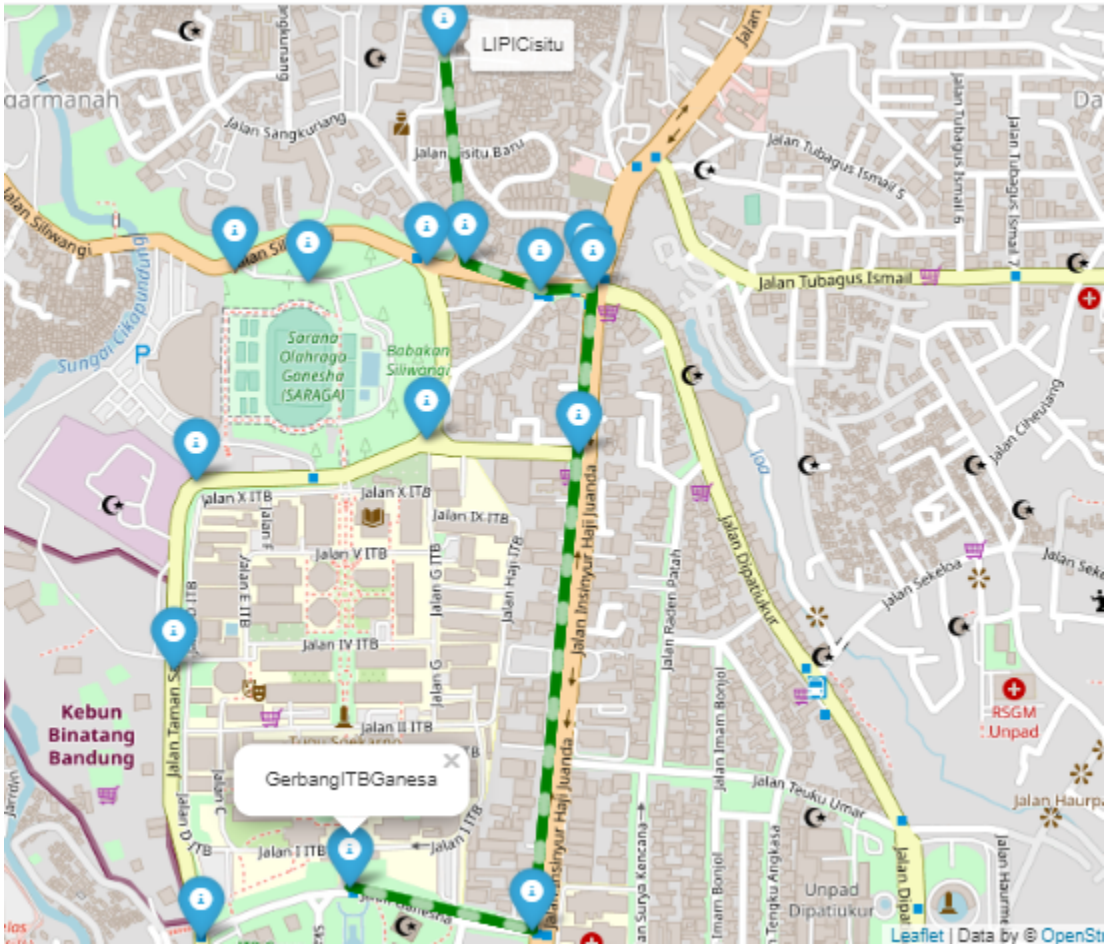
Note: Setiap titik dapat di klik dan menampilkan nama simpul, atau dapat di *hover* dengan kursor untuk menampilkan nama simpul pada main.ipynb. Jalur terpendek juga akan bergerak(animasi) sesuai arah simpul awal ke simpul tujuan bila dijalankan pada *Jupyter Notebook.*

1. **Peta Jalan Sekitar Kampus ITB**
    a. GerbangITBGanesa ke LIPICisitu

Solusi A*: **GerbangITBGanesa** → SimpanganGanesaJuanda → SimpanganDayangSumbiJuanda → SimpanganEmpatMcD → PertigaanSiliwangiSumur → PertigaanSiliwangiSangkuriang → **LIPICisitu**

Jarak tempuh: 1776.0 meter

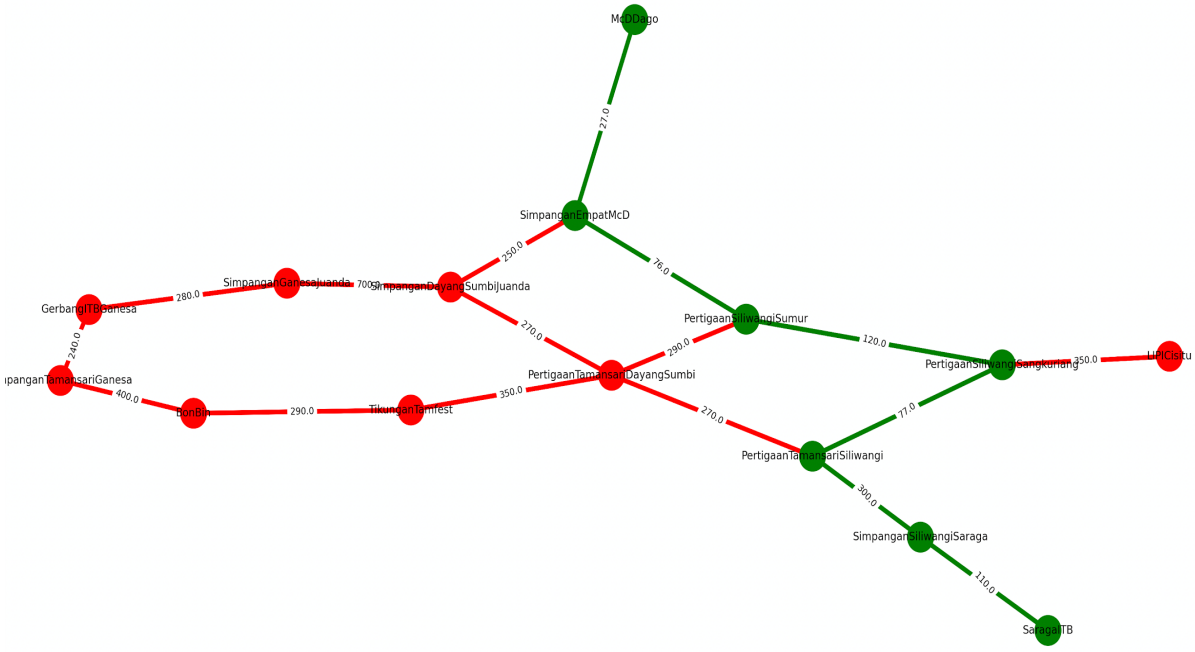

SimpanganTamansariGanesa
240.0
GerbangITBGanesa
400.0
BonBin
280.0
290.0
SimpanganGaneshaJuanda
TikunganTamfest
700.0
350.0
SimpanganDayangSumbiJuanda
270.0
250.0
PertigaanTamansariDayangSumbi
270.0
SimpanganEmpatMcD
SaragaITB
110.0
SimpanganSiliwangiSaraga
300.0
PertigaanTamansariSiliwangi
290.0
76.0
PertigaanSiliwangiSumur
27.0
77.0
120.0
PertigaanSiliwangiSangkuriang
350.0
LIPICisitu


LIPICisitu
Jalan Sangkuriang
Jalan Cisitu Baru
Jalan Tubagus Ismail
Jalan Tubagus Ismail 5
Jalan Tubagus Ismail 6
Jalan Tubagus Ismail 7
Jalan Siliwangi
Sungai Cikapundung
Sarana Olahraga Ganesha (SARAGA)
Babakan Siliwangi
Jalan X ITB
Jalan IX ITB
Jalan V ITB
Jalan G ITB
Jalan Haji ITB
Jalan IV ITB
Jalan III ITB
Jalan I ITB
Jalan Taman Sari
Jalan Insinyur Haji Juanda
Jalan Dipatiukur
Jalan Ciheulang
Jalan Sekeloa
Jalan Imam Bonjol
Jalan Teuku Umar
Jalan Haurpa
Kebun Binatang Bandung
GerbangITBGanesa
RSGM Unpad
Unpad Dipatiukur
Leaflet | Data by © OpenSt

b. SaragaITB ke McDDago

Solusi A*: **SaragaITB** → SimpanganSiliwangiSaraga → PertigaanTamansariSiliwangi → PertigaanSiliwangiSangkuriang → PertigaanSiliwangiSumur → SimpanganEmpatMcD → **McDDago**

Jarak tempuh: 710.0 meter



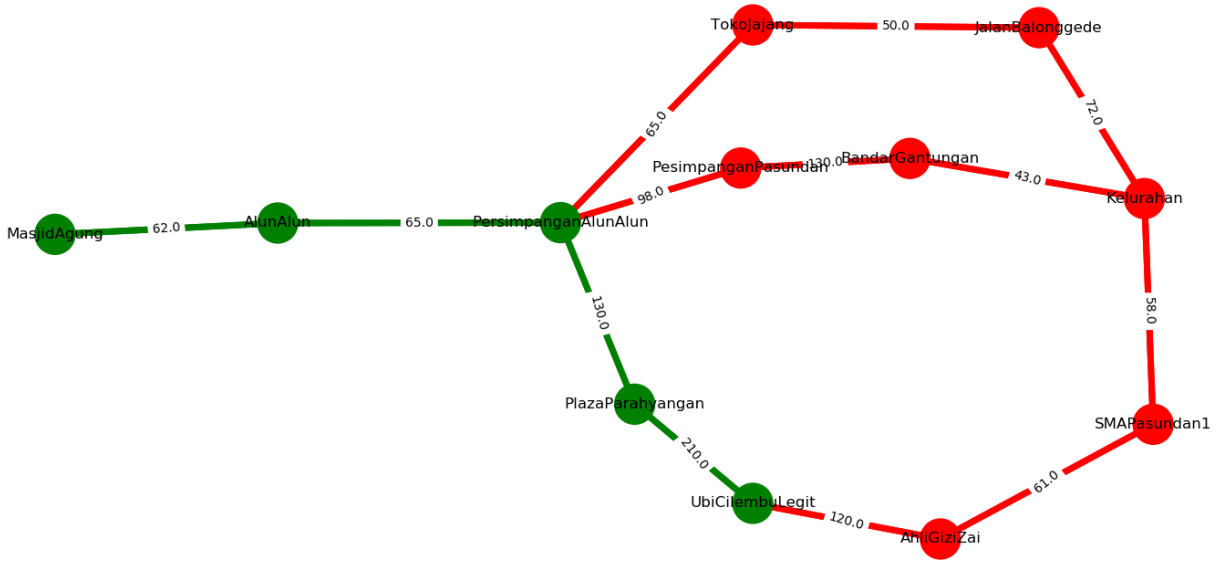

SaragaITB
McDDago
Jalan Taman Sari
Hotel Scarlet Dago
Jalan Sumur Bandung
Jalan Insinyur Haji Juanda
Sasana Budaya Ganesha
Lap Tennis Sabuga
Lap Basket Sabuga
Lap Tennis Sabuga
Sarana Olahraga Ganesha (SARAGA)
Kolam Renang Saraga (3M)
Lapangan Futsal A Saraga
Babakan Siliwangi
Taman Segi Tiga
Taman Utara
Jalan Dayang Sumbi
Dayang Sumbi
Jalan Taman Sari
Jalan X ITB
Jalan X ITB
Jalan X ITB
Gedung Riset
CAS
CRCS
CRCS
ESIA
Jalan IX ITB
Wisma

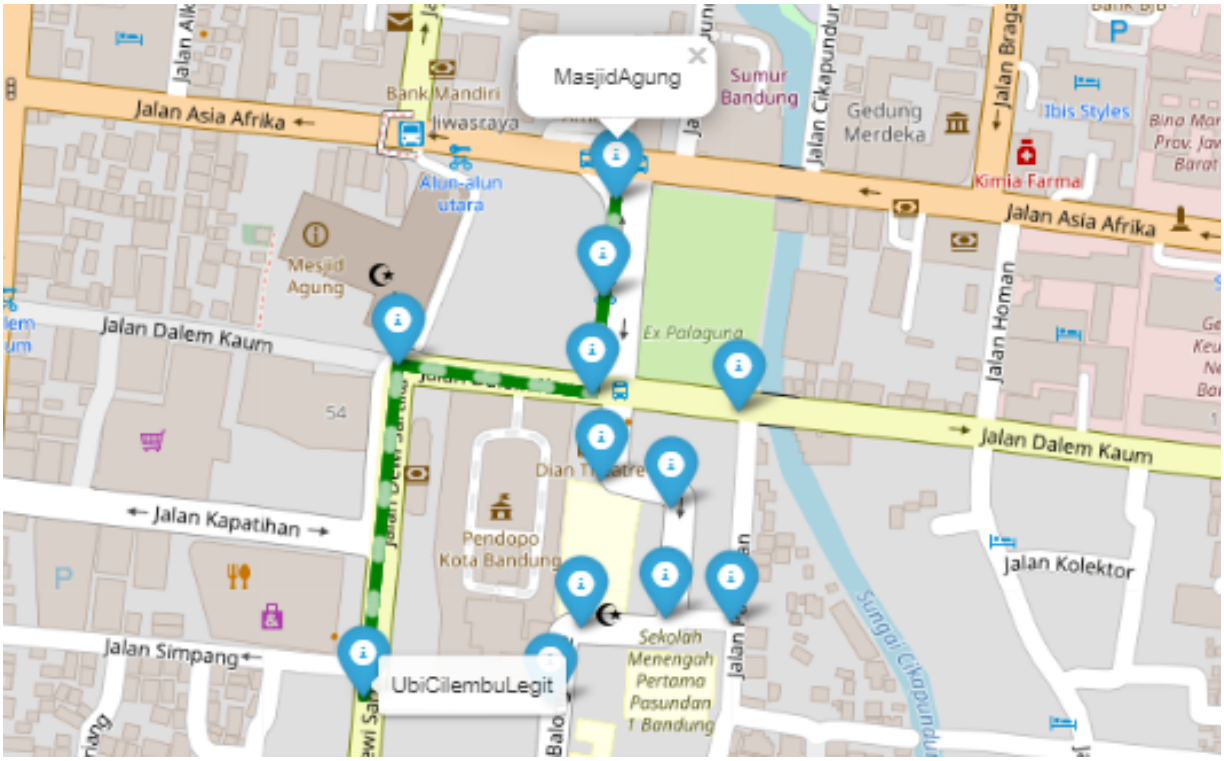
2. **Peta Jalan Sekitar Alun-Alun Bandung**

    a. MasjidAgung ke UbiCilembuLegit

Solusi A*: **MasjidAgung** → AlunAlun → PersimpanganAlunAlun → PlazaParahyangan → **UbiCilembuLegit**

Jarak tempuh: 467.0

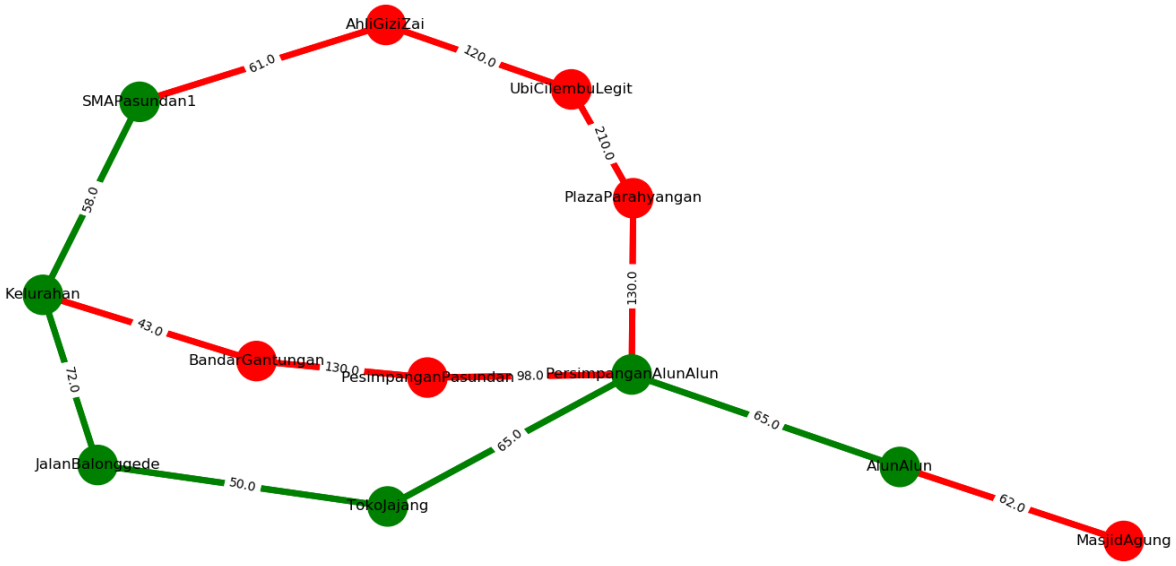




b. AlunAlun ke SMAPasundan1

Solusi A*: **AlunAlun** → PersimpanganAlunAlun → TokoJajang → JalanBalonggede → Kelurahan → **SMAPasundan1**

Jarak tempuh: 310.0 meter

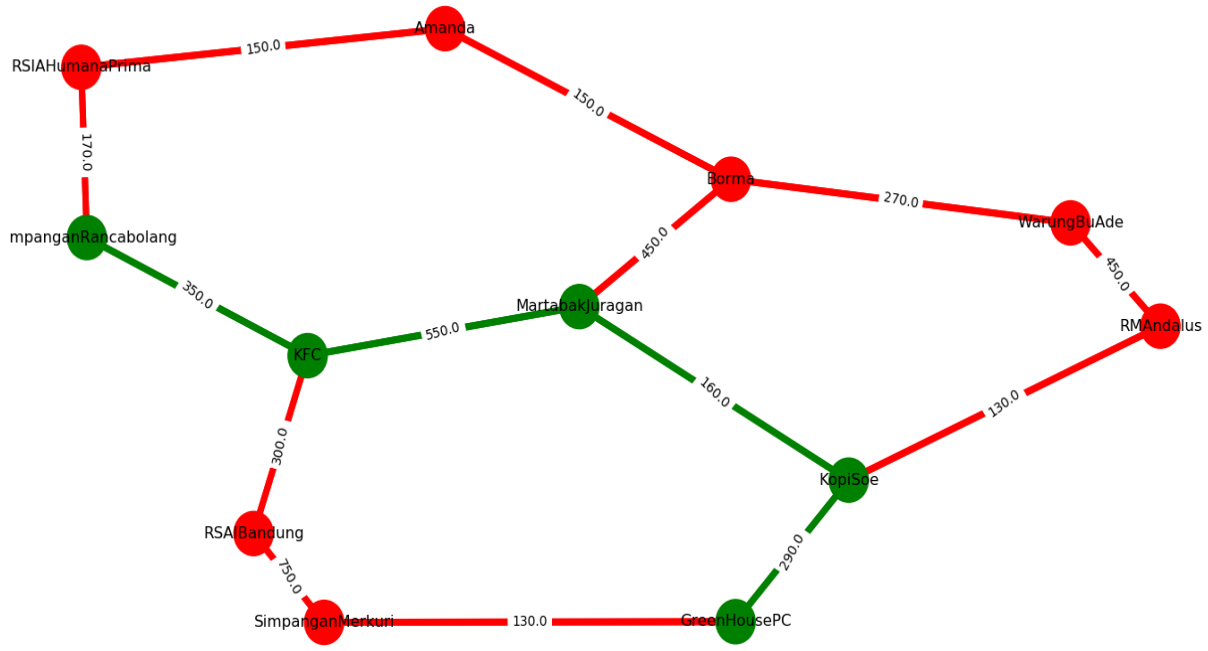


3. **Peta Jalan Sekitar Buahbatu**

    a. PersimpanganRancabolang ke GreenHousePC

Solusi A*: **SimpanganRancabolang** → KFC → MartabakJuragan → KopiSoe → **GreenHousePC**

Jarak tempuh: 1350.0 meter

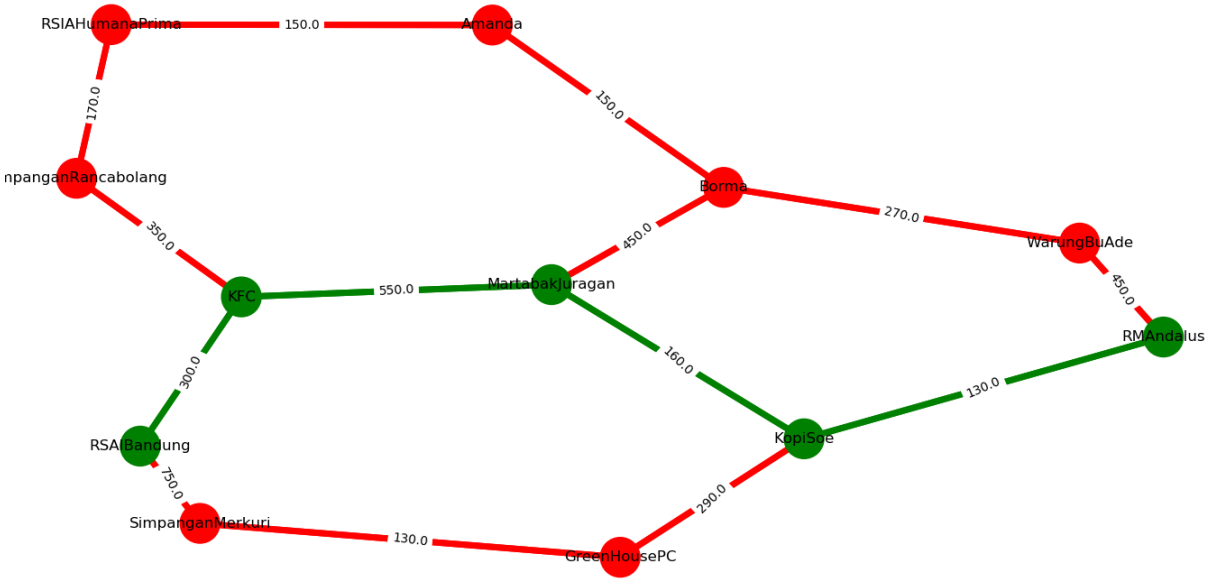


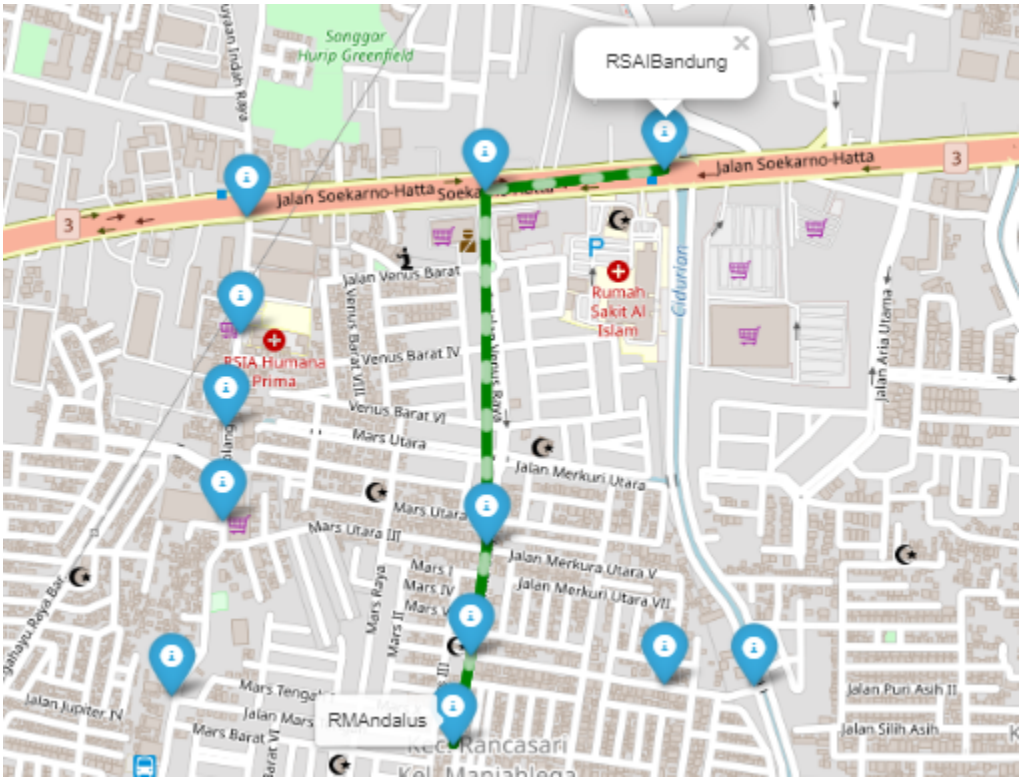

SimpanganRancabolang
Jalan Soekarno-Hatta
Jalan Soekarno
3
Jalan Venus Barat
Rumah Sakit Al Islam
Cidurian
RSIA Humana Prima
Venus Barat IV
Venus Barat VI
Mars Utara
Jalan Merkuri Utara
Mars Utara III
GreenHousePC
Mars I
Mars IV
Mars Raya
Mars II
Jalan Margahayu Raya Bar
Jalan Jupiter N
Mars Tengah I
Jalan Mars Tengah
Mars Barat VI
Mars X

b. RSAIBandung ke RMAndalus

Solusi A*: **RSAIBandung** → KFC → MartabakJuragan → KopiSoe → **RMAndalus**

Jarak tempuh: 1140.0 meter



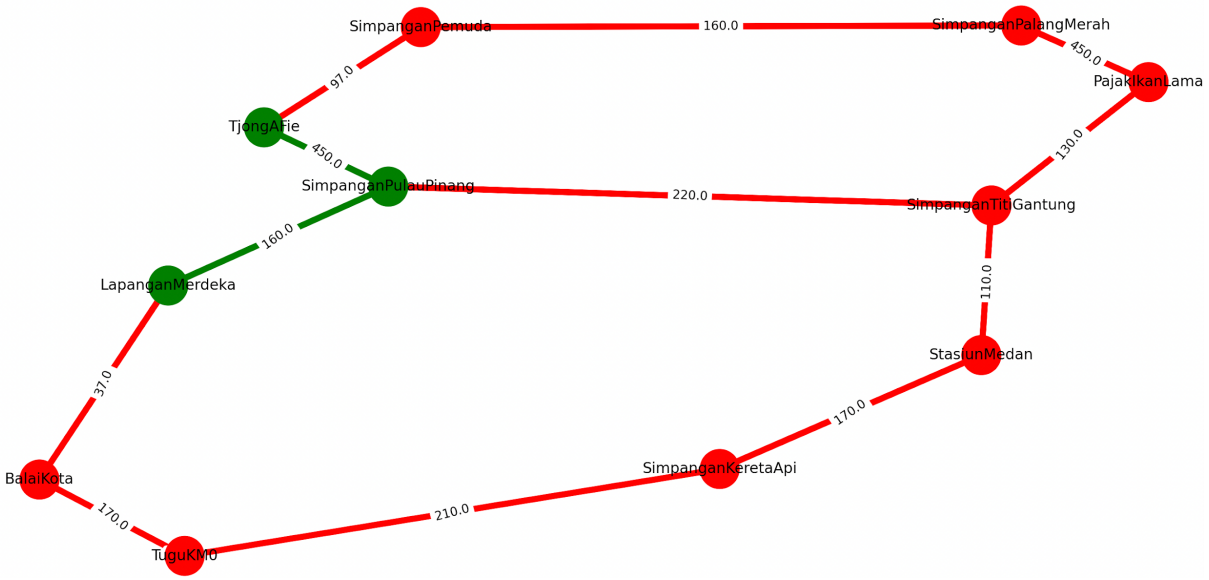


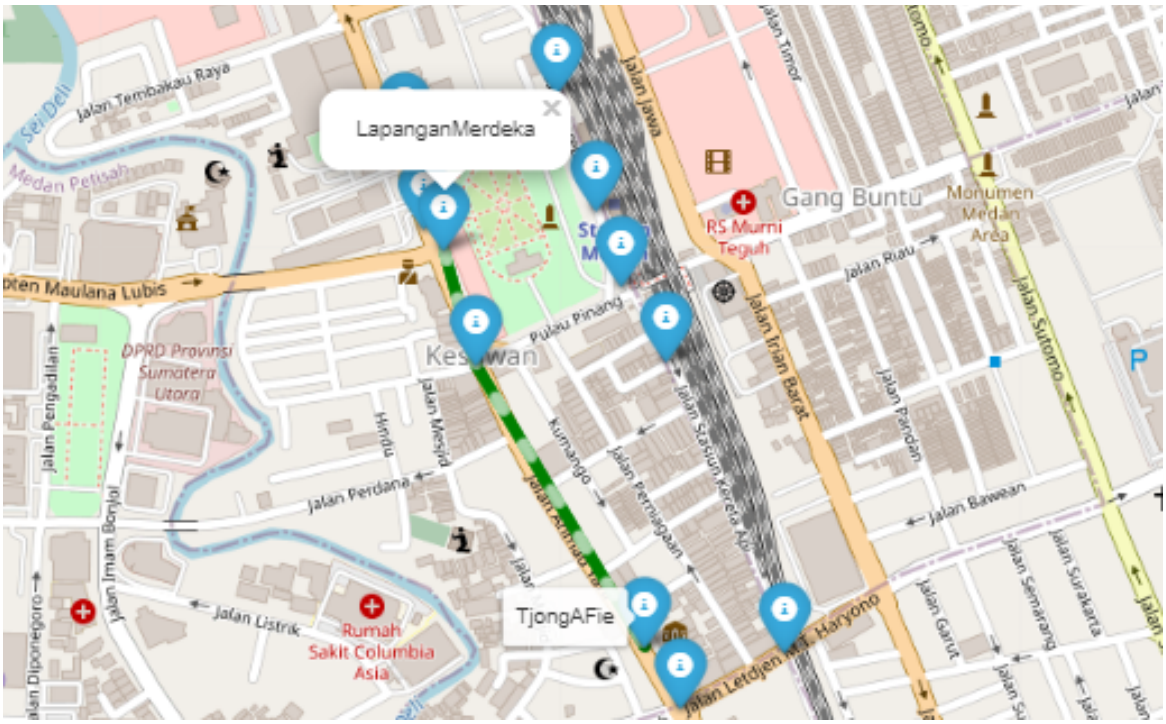
4. Peta Jalan Sebuah Kawasan di Kota Medan

a. LapanganMerdeka ke TjongAFie

Solusi A*: **LapanganMerdeka** → SimpanganPulauPinang → **TjongAFie**

Jarak tempuh: 610.0 meter

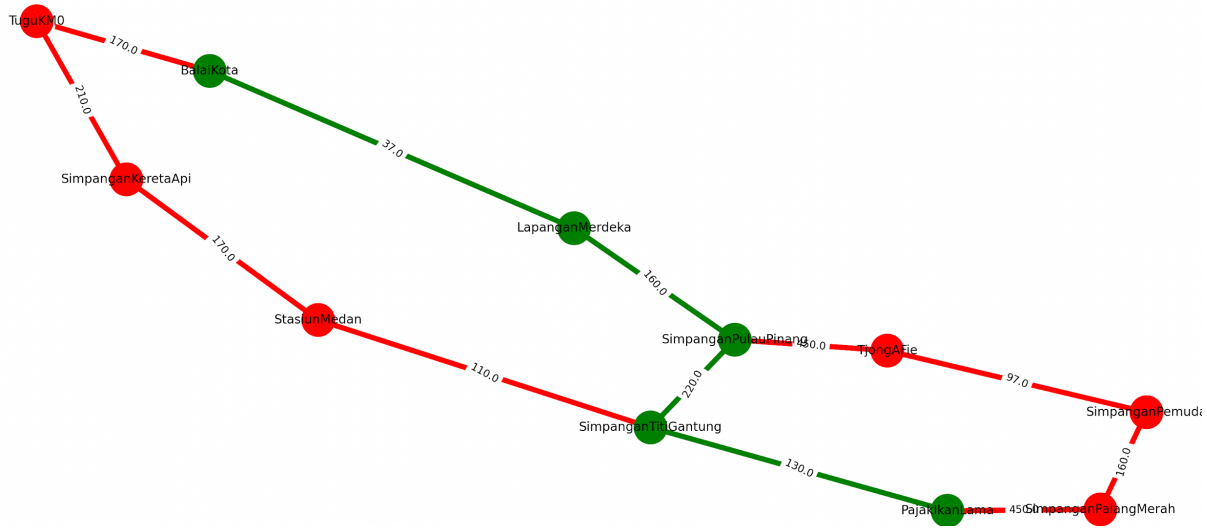




b. BalaiKota ke PajakIkanLama

Solusi A*: **BalaiKota** → LapanganMerdeka → SimpanganPulauPinang → SimpanganTitiGantung → **PajakIkanLama**

Jarak tempuh: 547.0 meter

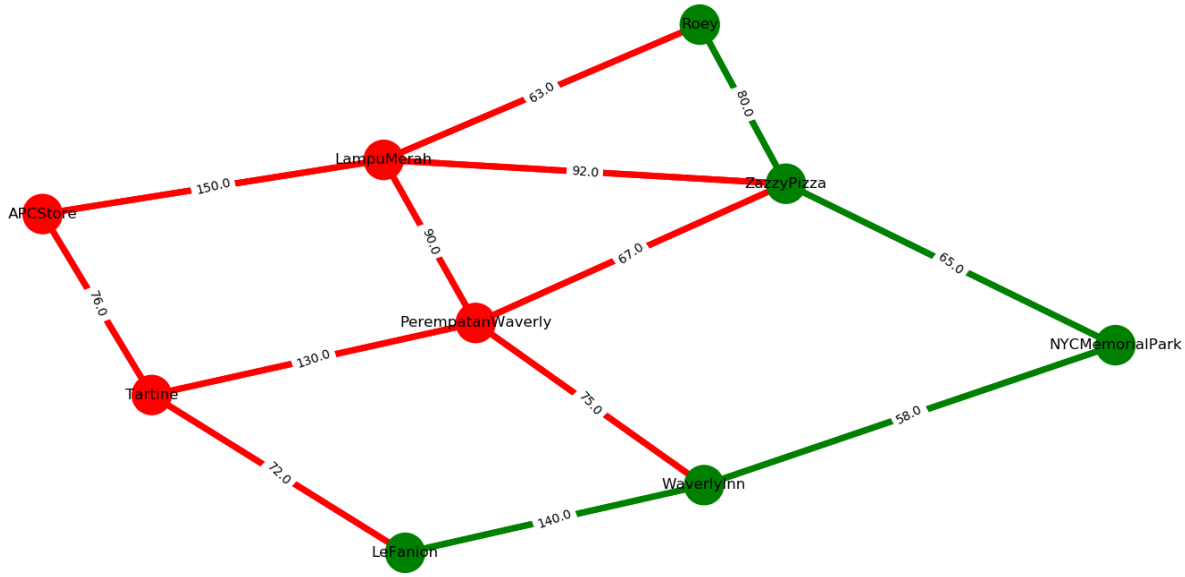


5. Peta Jalan Sebuah Kawasan di Kota Manhattan

a. LeFanion ke Roey

Solusi A*: **LeFanion** → WaverlyInn → NYCMemorialPark → ZazzyPizza → **Roey**

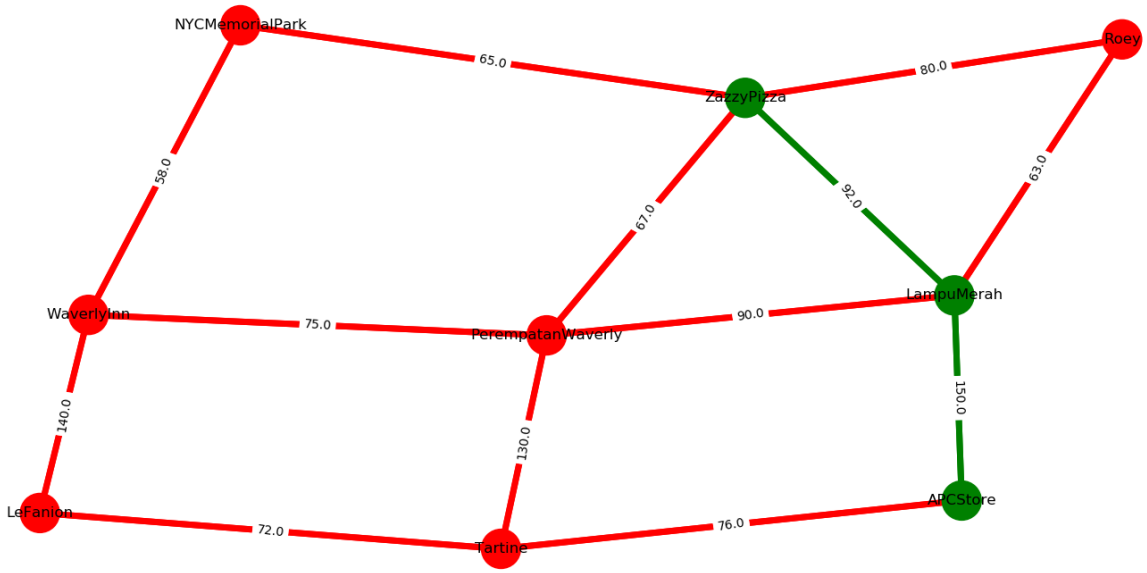
Jarak tempuh: 343.0 meter



b. ZazzyPizza ke APCStore

Solusi A*: **ZazzyPizza** → LampuMerah → **APCStore**

Jarak tempuh: 242.0 meter



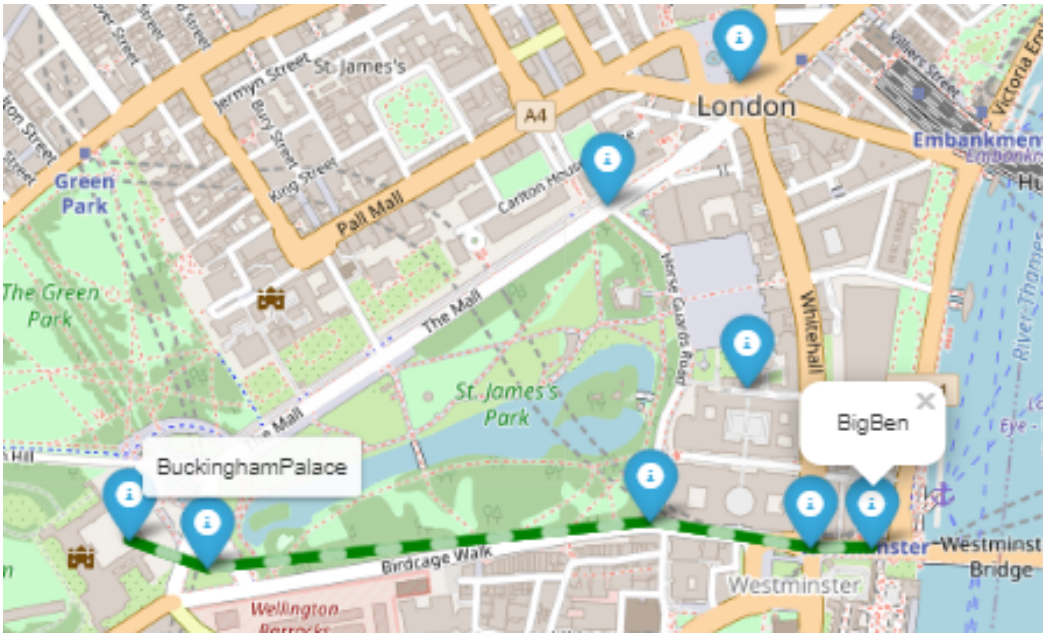
6. **Peta Jalan Sebuah Kawasan di Kota London**

   a. BigBen ke BuckinghamPalace

Solusi A*: **BigBen** → PerempatanParliamentSquare → PertigaanBirdcage → PertigaanBuckingham → **BuckinghamPalace**

Jarak tempuh: 1196.0 meter

***visualisasi dalam *graph* seperti ini karena memang dibuat tidak ada jalan ke 10DOwningStreet**



b. BigBen ke PertigaanTheMall

Solusi A*: **BigBen** → PerempatanParliamentSquare → PertigaanBirdcage → **PertigaanTheMall**

Jarak tempuh: 946.0 meter

***visualisasi dalam *graph* seperti ini karena memang dibuat tidak ada jalan ke 10DOwningStreet**

D. *Link* Kode Program

https://github.com/RizkyAnggita/tucil3_stima

1.	Program dapat menerima input graf	✔
2.	Program dapat menghitung lintasan terpendek	✔
3.	Program dapat menampilkan lintasan terpendek serta jaraknya	✔
4.	Bonus: Program dapat menerima input peta dengan Google Map API dan menampilkan peta	