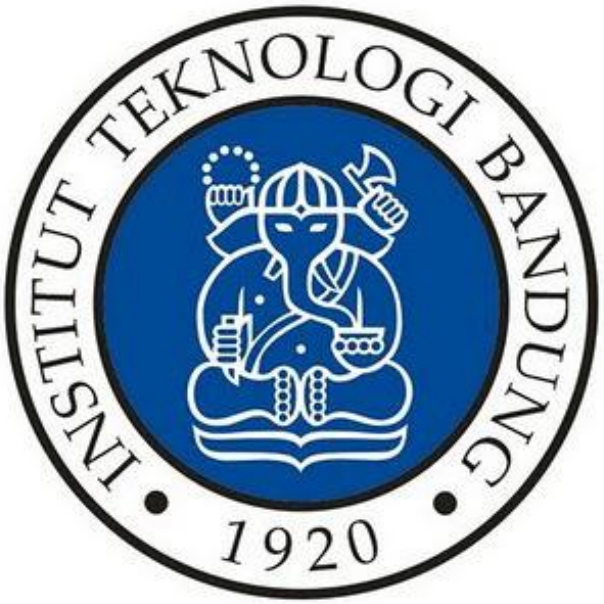
Tugas Kecil 3 IF2211 Strategi Algoritma

Implementasi Algoritma A untuk Menentukan Lintasan Terpendek*

13519040 - Shafira Naya Aprisadianti
13519133 - Delisha Azza Naadira



PROGRAM STUDI SARJANA TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2021

1. Kode Program

File graph.py

```
# File: graph.py
import math

def make_coordinates(lines):
    global list_of_coordinates
    global list_of_names
    n = int(lines[0])
    list_of_coordinates = []
    list_of_names = []
    for i in range(1, n+1):
        coordinates = lines[i].split(" ")
        coords_list = [float(coordinates[1]), float(coordinates[2])]
        list_of_coordinates.append(coords_list)
        list_of_names.append(coordinates[0])
    return list_of_coordinates, list_of_names

def make_list_of_lat(c):
    global list_lat
    list_lat = []
    for i in range(len(c)):
        list_lat.append(c[i][0])
    return list_lat

def make_list_of_lon(c):
    global list_lon
    list_lon = []
    for i in range(len(c)):
        list_lon.append(c[i][1])
    return list_lon

def avg_lat(lat):
    return sum(lat) / len(lat)

def avg_lon(lon):
    return sum(lon) / len(lon)

def make_matrix(lines):
    n = int(lines[0])
    adj_matrix = []
```

```
    for i in range(n+1, n*2+1):
        line = lines[i].split(" ")
        adj_matrix.append(line)
    return adj_matrix

def make_adj_list(m):
    global adj_list
    global list_of_names
    adj_list = []
    for i in range(0, len(m)):
        neighbor = []
        for j in range(0, len(m)):
            if m[i][j] == '1':
                name = convert_to_name(j)
                neighbor.append(name)
        adj_list.append(neighbor)
    return adj_list

def make_adj_matrix(m):
    global adj_matrix
    global list_of_coordinates
    n = len(m)
    adj_matrix = [[ 0 for i in range(n)] for j in range(n)]
    for i in range(0,n):
        for j in range(0,n):
            if m[i][j] == '1':
                distance =
haversineDistance(list_of_coordinates[i],list_of_coordinates[j])
                adj_matrix[i][j] = distance
    return adj_matrix

def make_heuristic_matrix(m):
    global heuristic_matrix
    global list_of_coordinates
    n = len(m)
    heuristic_matrix = [[ 0 for i in range(n)] for j in range(n)]
    for i in range(0,n):
        for j in range(0,n):
            if (i!=j):
                distance =
haversineDistance(list_of_coordinates[i],list_of_coordinates[j])
                heuristic_matrix[i][j] = distance
            else:
```

```
                heuristic_matrix[i][j] = 0
    return heuristic_matrix

# parameternya list coordinate a [lat,lon], list coordinate b [lat,lon]
# return: haversineDistance dalam meters
def haversineDistance(a,b):
    # ambil nilai latitude dan longtitude
    lat1 = a[0]
    lon1 = a[1]
    lat2 = b[0]
    lon2 = b[1]
     # convert ke radian
    lat1_rad = lat1 * math.pi / 180.0
    lat2_rad = lat2 * math.pi / 180.0
    # selisih latitude dan longtitude dalam radian
    delta_lat = (lat2 - lat1) * math.pi / 180.0
    delta_lon = (lon2 - lon1) * math.pi / 180.0
    # bagian dari rumus (yang di dalam akar)
    a = (pow(math.sin(delta_lat / 2), 2) + pow(math.sin(delta_lon / 2), 2) *
math.cos(lat1_rad) * math.cos(lat2_rad))
    # radius bumi
    r = 6371
    # rumus untuk mendapatkan distance dalam meter
    distance = 2 * r * math.asin(math.sqrt(a)) * 1000
    return distance

# convert node name to node index
def convert_to_idx(node_name):
    global list_of_names
    idx = 0
    for i in range(len(list_of_names)):
        if (list_of_names[i] == node_name):
            idx = i
    return idx

# convert node index to node name
def convert_to_name(idx):
    global list_of_names
    name = ''
    for i in range(len(list_of_names)):
        if (i == idx):
            name = list_of_names[i]
    return name
```

```
# initial adalah nama start node (string)
# final adalah nama goal node (string)
def astar(initial, final):
    global adj_matrix
    global heuristic_matrix
    global adj_list
    global list_of_names
    global path

    # inisialisasi
    idx_initial = convert_to_idx(initial)
    idx_final = convert_to_idx(final)
    queue = [[idx_initial, 0, [initial]]]
    current_node = []

    # selama queue belum kosong
    while(len(queue) != 0):
        # dequeue
        current_node = queue.pop(0)
        current_node_idx = convert_to_idx(current_node[0])
        # jika start node sama dengan goal node
        if (current_node_idx == idx_final):
            break
        # mengunjungi node-node yang bertetangga dengan current_node
        for neighbor in adj_list[current_node_idx]:
            # copy visited node
            visited_node = []
            for c in current_node[2]:
                visited_node.append(c)
            # masukkan neighbor ke visited node
            i = convert_to_idx(neighbor)
            visited_node.append(neighbor)
            # masukkan node ke queue
            # masukkan informasi: current_node name, f(current_node),
visited_node ke queue
            queue.append([neighbor, adj_matrix[current_node_idx][i] +
heuristic_matrix[i][idx_final], visited_node])
            # urutkan menaik, agar selalu pop yang costnya terkecil
            queue.sort(key = lambda q : q[1])

    # path adalah shortest path
    path = current_node[2]
```

```
    # hitung cost
    cost = 0
    path_cost = []
    for node in path:
        path_cost.append(convert_to_idx(node))
    for i in range(len(path)-1):
        cost += adj_matrix[path_cost[i]][path_cost[i+1]]

    return path, cost

def path_coords(path):
    global list_of_names
    global list_of_coordinates
    global list_of_path_coords
    list_of_path_coords = []
    for node in path[0]:
        list_of_path_coords.append(list_of_coordinates[convert_to_idx(node)])
    return list_of_path_coords

def print_route(solution):
    print("Lintasan terpendek: ", end=" ")
    for i in range(len(solution[0])):
        if (i == (len(solution[0])-1)):
            print(solution[0][i])
        else:
            print(solution[0][i], end=" -> ")
    print("Panjang lintasan: ", solution[1], "meter. ")
    print("Buka map.html dengan Google Chrome untuk melihat visualisasi peta.")

def string_route(solution):
    solution_list = []
    for i in range(len(solution[0])):
        if (i == (len(solution[0])-1)):
            solution_list.append(solution[0][i])
        else:
            solution_list.append(solution[0][i]+" -> ")
    solution_list = ' '.join([str(elem) for elem in solution_list])
    return "Lintasan terpendek: " + (solution_list) + ", dengan panjang lintasan:
" + str(solution[1]) + " meter. "

def initialize(file_name):
    data_folder = "../test/"
```

```
    file_to_open = data_folder + file_name
    f = open(file_to_open, "r")
    lines = f.read().splitlines()
    coordinates = make_coordinates(lines)[0]
    list_lat = make_list_of_lat(coordinates)
    list_lon = make_list_of_lon(coordinates)
    node_names = make_coordinates(lines)[1]
    matrix = make_matrix(lines)
    adj_list = make_adj_list(matrix)
    adj_matrix = make_adj_matrix(matrix)
    heur_matrix = make_heuristic_matrix(matrix)
```

File main.py

```
# File: main.py
# Program utama dan visualisasi map

import folium
import graph as g

# PROGRAM UTAMA
file_name = input("Masukkan nama file dalam format .txt: ")
g.initialize(file_name)
start_node = input("Masukkan start node: ")
goal_node = input("Masukkan goal node: ")
print("Hasil: ")
path_solution = g.astar(start_node, goal_node)
list_path = g.path_coords(path_solution)
g.print_route(path_solution)

def color(name, solution):
    # kalau starting point
    if name == solution[0][0]:
        color = 'green'
    else:
        # kalau di path
        if name in solution[0]:
            color = 'red'
        else:
            color = 'blue'
    return color

map =
```

```
folium.Map(location=[g.avg_lat(g.list_lat),g.avg_lon(g.list_lon)],zoom_start=15)

# make markers
for point in range(0, len(g.list_of_coordinates)):
   folium.Marker(g.list_of_coordinates[point], popup=g.list_of_names[point],
icon=folium.Icon(color=color(g.list_of_names[point],
path_solution))).add_to(map)

# make path
fg = folium.FeatureGroup("Path")
line = folium.vector_layers.PolyLine(list_path, color='red',
weight=10).add_to(fg)
fg.add_to(map)

# add title
title_html = '''
             <h3 align="center" style="font-size:20px"><b>Nama file: {}</b>
             </h3>
             '''.format(file_name)
map.get_root().html.add_child(folium.Element(title_html))

# add lintasan terpendek
solution = g.string_route(path_solution)
solution_html = '''
             <h3 align="center" style="font-size:20px"><b>{}</b>
             </h3>
             '''.format(solution)
map.get_root().html.add_child(folium.Element(solution_html))

map.add_child(folium.LayerControl())
map.save(outfile='map.html')
```

2. Peta/Graf Input

a. Peta jalan sekitar kampus ITB/Dago: itb.txt

```
12
A -6.884893 107.611445
B -6.885191 107.613017
C -6.885257 107.613733
D -6.887256 107.611540
E -6.887386 107.613611
F -6.887910 107.608289
G -6.893882 107.608450
H -6.893230 107.610447
I -6.893605 107.611944
J -6.893780 107.613036
K -6.894759 107.611723
L -6.894883 107.608839
0 1 0 1 0 0 0 0 0 0 0 0
1 0 1 1 0 0 0 0 0 0 0 0
0 1 0 0 1 0 0 0 0 0 0 0
1 1 0 0 1 1 0 0 0 0 0 0
0 0 1 1 0 0 0 0 0 1 0 0
0 0 0 1 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 1 0 0 0 1
0 0 0 0 0 0 1 0 1 0 0 0
0 0 0 0 0 0 0 1 0 1 1 0
0 0 0 0 1 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 1 0 0 1
0 0 0 0 0 0 1 0 0 0 1 0
```

b. Peta jalan sekitar Alun-alun Bandung: alun2.txt

```
16
A -6.921141 107.607668
B -6.920768 107.604056
C -6.918263 107.604216
D -6.919038 107.606670
E -6.920995 107.606441
F -6.922551 107.607619
G -6.922771 107.609810
H -6.925376 107.610599
I -6.926075 107.610524
J -6.925835 107.607071
K -6.924356 107.607253
L -6.924317 107.606160
M -6.923950 107.603842
N -6.922388 107.606404
O -6.923443 107.606298
P -6.923100 107.603892
0 0 0 0 1 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 1
0 1 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
1 1 0 1 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 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 1 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 1 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 1 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
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 1 0 1 0 1
0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0
```

c. Peta jalan sekitar Buahbatu: buahbatu.txt

```
8
SimpangSoeHattaMTC -6.940351 107.658245
SimpangRancabolangSoeHatta -6.939252 107.663915
BormaRancabolang -6.943234 107.663564
SimpangCidurianSelSoeHatta -6.942138 107.652719
SimpangCidurianSelCiwastra -6.955690 107.654484
RSIAHarapanBunda -6.956029 107.662112
MasjidBuahBatu -6.954222 107.639885
CarrefourKiaracondong -6.946367 107.641756
0 1 0 1 0 0 0 0
1 0 1 0 0 0 0 0
0 1 0 0 0 1 0 0
1 0 0 0 1 0 0 1
0 0 0 1 0 1 1 0
0 0 1 0 1 0 0 0
0 0 0 0 1 0 0 1
0 0 0 1 0 0 1 0
```

d. Peta jalan sebuah kawasan di kotamu (Jalan Ahmad Dahlan): ahmaddahlan.txt

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

e. Peta jalan sekitar Jakarta: jakarta.txt

```
15
A -6.245131 106.788753
B -6.246568 106.791361
C -6.246584 106.791921
D -6.243914 106.791703
E -6.244357 106.790184
F -6.244625 106.789645
G -6.244878 106.789209
H -6.246105 106.791314
I -6.245636 106.791288
J -6.245589 106.791838
K -6.245394 106.790106
L -6.245254 106.791829
M -6.244802 106.790382
N -6.245968 106.790115
O -6.246423 106.789829
0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
0 0 1 0 0 0 0 1 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 1 0 0 0 0 0 0 1 0 0 0
0 0 0 1 0 1 0 0 0 0 0 0 1 0 0
0 0 0 0 1 0 1 0 0 0 1 0 0 0 0
1 0 0 0 0 1 0 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 1 0 0 0 0 1 0
0 0 0 0 0 0 0 1 0 1 1 0 0 0 0
0 0 1 0 0 0 0 0 1 0 0 1 0 0 0
0 0 0 0 0 1 0 0 1 0 0 0 1 0 0
0 0 0 1 0 0 0 0 0 1 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 1 1 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
```

f. Peta jalan sekitar ITB Dago (2): itbdago.txt

```
14
A -6.893634 107.611970
B -6.893252 107.610425
C -6.893890 107.608426
D -6.898057 107.609559
E -6.898827 107.612643
F -6.893761 107.613038
G -6.891472 107.613236
H -6.892391 107.617819
I -6.885191 107.613701
J -6.884902 107.611468
K -6.887359 107.611214
L -6.887895 107.608260
M -6.897417 107.611396
N -6.895880 107.609392
0 1 0 0 0 1 0 0 0 0 0 0 1 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 1 0 1
0 0 0 0 1 0 0 0 0 0 0 0 0 1
0 0 0 1 0 1 0 0 0 0 0 0 1 0
1 0 0 0 1 0 1 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1 1 0 0 0 0 0
0 0 0 0 0 0 1 0 1 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 1 0 1 0 0 0
0 0 0 0 0 0 0 0 1 1 0 1 0 0
0 0 1 0 0 0 0 0 0 1 0 0 0
1 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 1 1 0 0 0 0 0 0 0 0 0 0
```

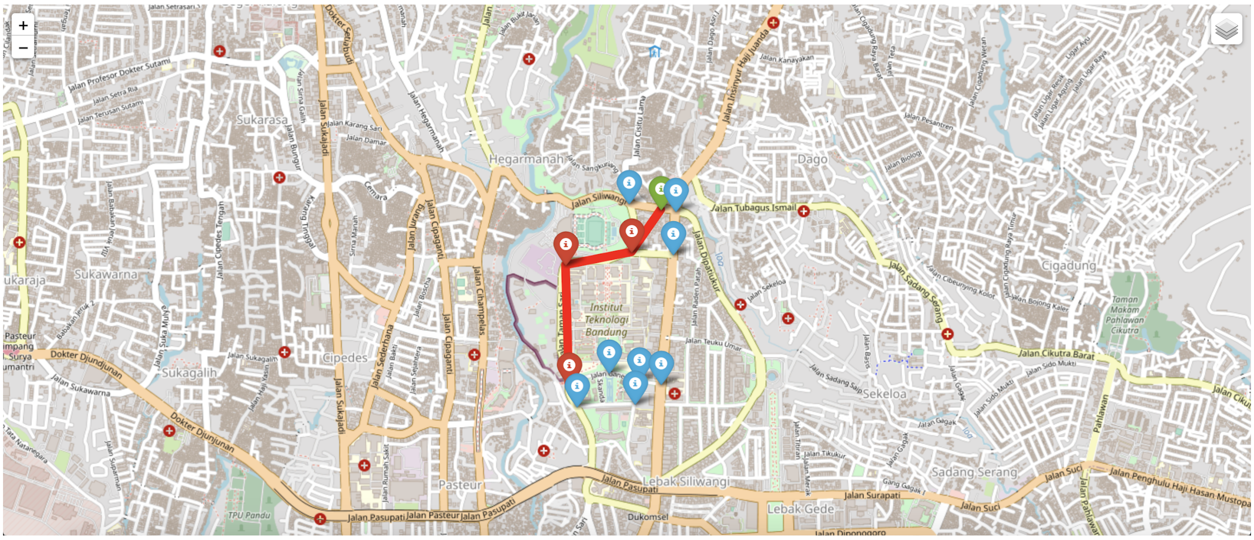
3. Screenshot peta yang memperlihatkan lintasan terpendek untuk sepasang simpul

a. Peta jalan sekitar kampus ITB/Dago: itb.txt

```
(base) Shafiras-MacBook-Pro-2:src shafiranaya$ python3 main.py
Masukkan nama file dalam format .txt: itb.txt
Masukkan start node: B
Masukkan goal node: G
Hasil:
Lintasan terpendek:  B -> D -> F -> G
Panjang lintasan:  1312.0930611549384 meter.
Buka map.html dengan Google Chrome untuk melihat visualisasi peta.
```

Nama file: itb.txt

Lintasan terpendek: B -> D -> F -> G, dengan panjang lintasan: 1312.0930611549384 meter.

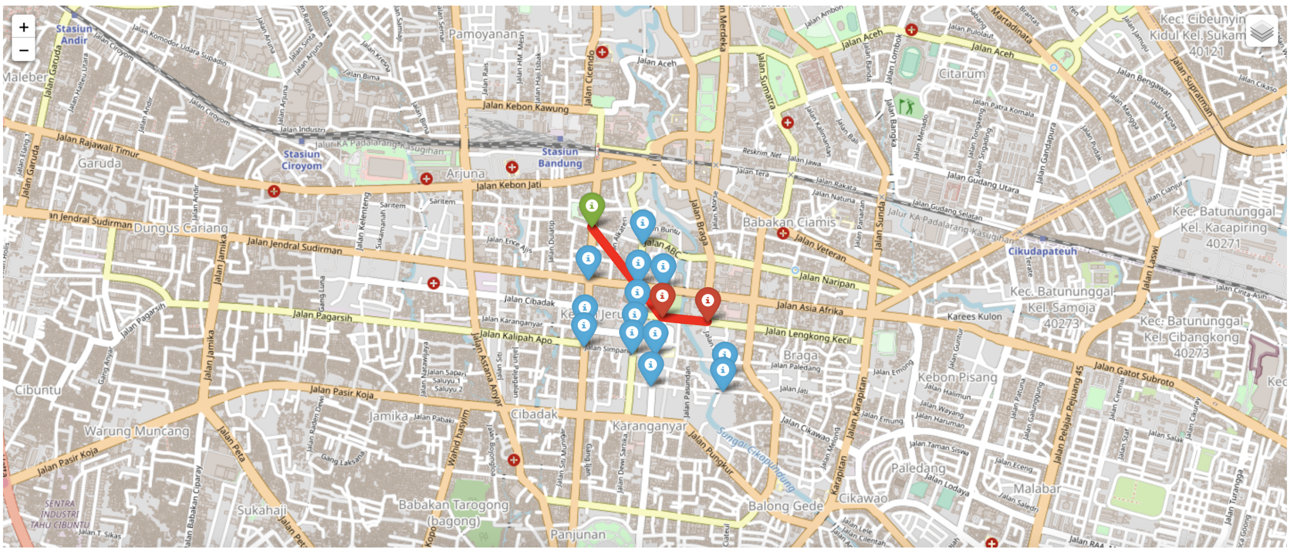


b. Peta jalan sekitar Alun-alun Bandung: alun2.txt

```
(base) Shafiras-MacBook-Pro-2:src shafiranaya$ python3 main.py
Masukkan nama file dalam format .txt: alun2.txt
Masukkan start node: C
Masukkan goal node: G
Hasil:
Lintasan terpendek:  C -> F -> G
Panjang lintasan:  243.08601402216502 meter.
Buka map.html dengan Google Chrome untuk melihat visualisasi peta.
```

Nama file: alun2.txt

Lintasan terpendek: C -> F -> G, dengan panjang lintasan: 243.08601402216502 meter.

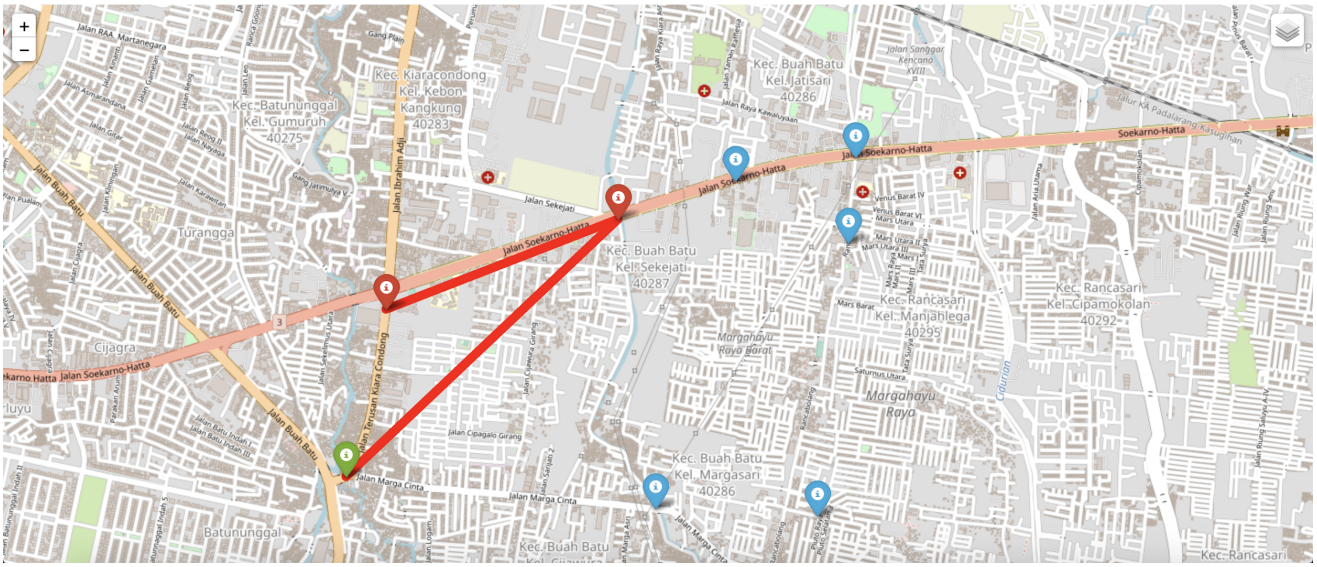


c. Peta jalan sekitar Buahbatu: buahbatu.txt

```
(base) Shafiras-MacBook-Pro-2:src shafiranaya$ python3 main.py
Masukkan nama file dalam format .txt: buahbatu.txt
Masukkan start node: MasjidBuahBatu
Masukkan goal node: CarrefourKiaracondong
Hasil:
Lintasan terpendek:  MasjidBuahBatu -> SimpangCidurianSelSoeHatta -> CarrefourKiaracondong
Panjang lintasan:  1298.245176814189 meter.
Buka map.html dengan Google Chrome untuk melihat visualisasi peta.
```

Nama file: buahbatu.txt

Lintasan terpendek: MasjidBuahBatu -> SimpangCidurianSelSoeHatta -> CarrefourKiaracondong, dengan panjang lintasan: 1298.245176814189 meter.

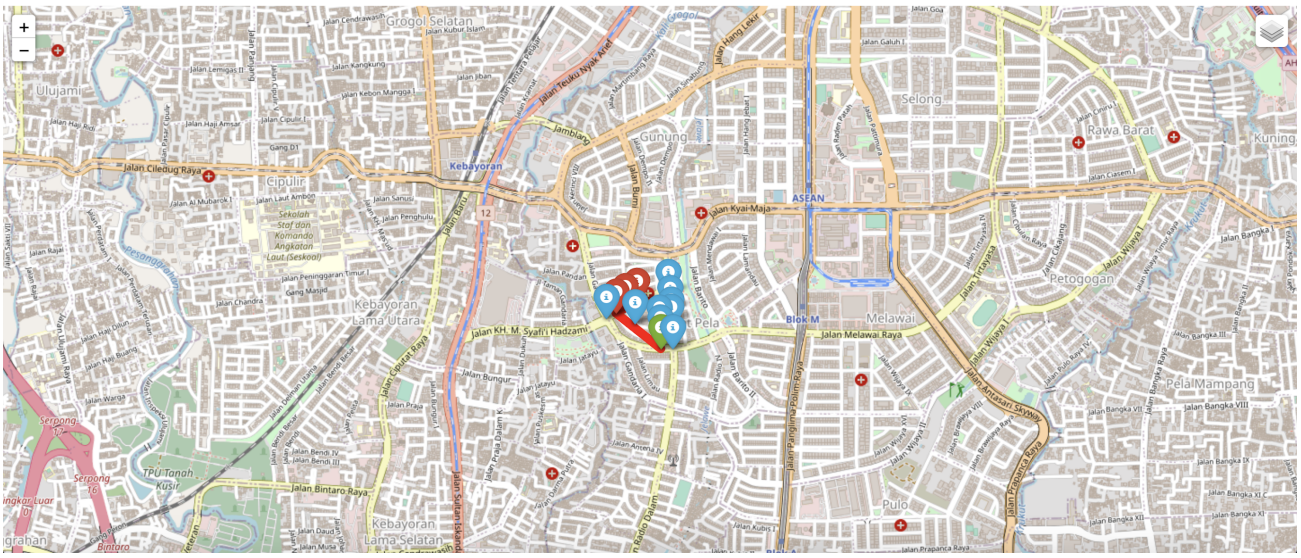


d. Peta jalan sebuah kawasan di kotamu (Jalan Ahmad Dahlan): ahmaddahlan.txt

```
(base) Shafiras-MacBook-Pro-2:src shafiranaya$ python3 main.py
Masukkan nama file dalam format .txt: ahmaddahlan.txt
Masukkan start node: 2
Masukkan goal node: 5
Hasil:
Lintasan terpendek:  2 -> 7 -> 6 -> 5
Panjang lintasan:  122.36444541240593 meter.
Buka map.html dengan Google Chrome untuk melihat visualisasi peta.
```

Nama file: ahmaddahlan.txt

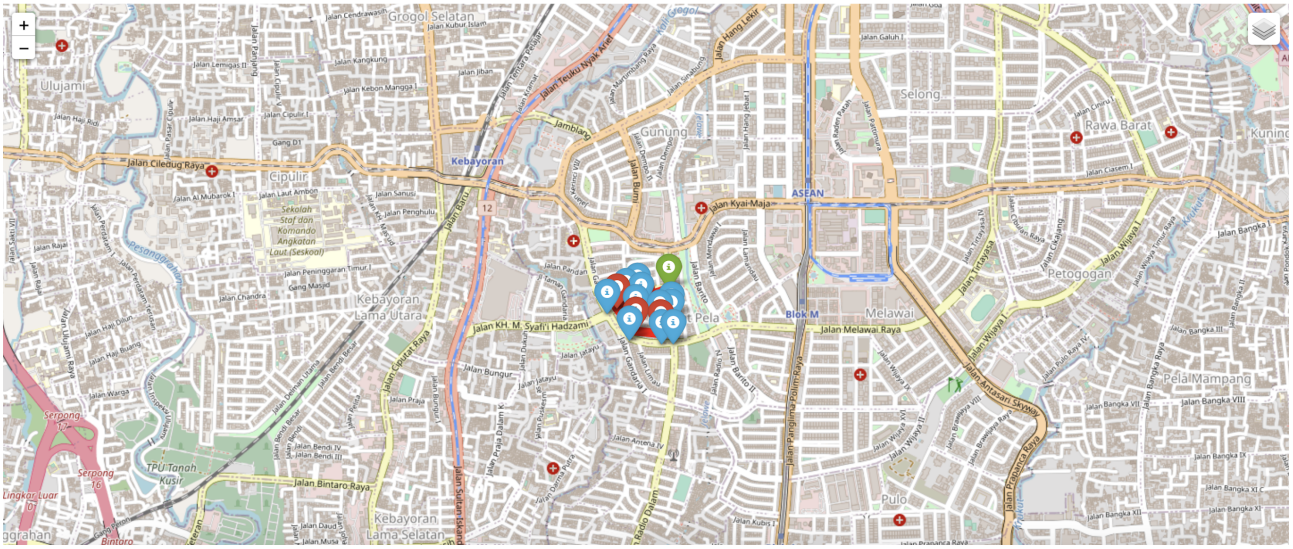
Lintasan terpendek: 2 -> 7 -> 6 -> 5, dengan panjang lintasan: 122.36444541240593 meter.



e. Peta jalan sekitar Jakarta: jakarta.txt

Nama file: jakarta.txt

Lintasan terpendek: D -> G -> N -> H, dengan panjang lintasan: 290.6267063031675 meter.

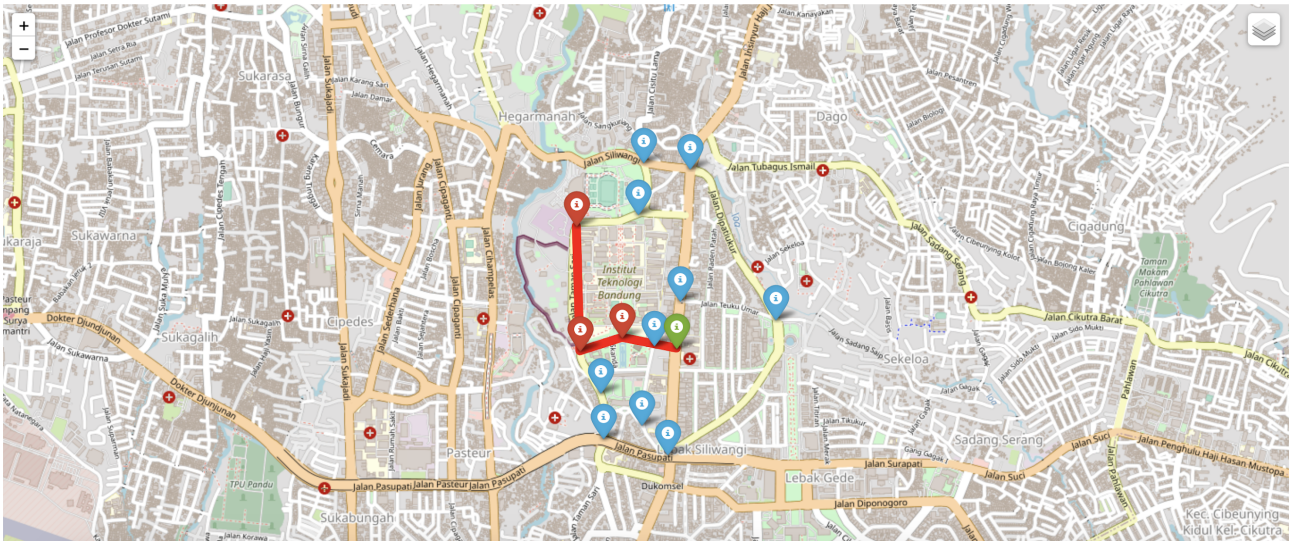


f. Peta jalan sekitar ITB Dago (2): itbdago.txt

```
(base) Shafiras-MacBook-Pro-2:src shafiranaya$ python3 main.py
Masukkan nama file dalam format .txt: itbdago.txt
Masukkan start node: F
Masukkan goal node: L
Hasil:
Lintasan terpendek:  F -> B -> C -> L
Panjang lintasan:  898.6602502353802 meter.
Buka map.html dengan Google Chrome untuk melihat visualisasi peta.
```

Nama file: itbdago.txt

Lintasan terpendek: F -> B -> C -> L, dengan panjang lintasan: 898.6602502353802 meter.



4. Alamat source code program

https://github.com/shafiranaya/Tucil3_13519040

5. Tabel

No.	Keterangan	Checklist
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	X