LAPORAN TUGAS KECIL 3

IF2211 STRATEGI ALGORITMA

IMPLEMENTASI ALGORITMA A* UNTUK MENENTUKAN LINTASAN TERPENDEK

Disusun oleh:

Dzaki Muhammad - 13519049 - K1

Rezda Abdullah Fachrezzi - 13519194 - K4

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

BAB 1

SOURCE CODE PROGRAM

1.1. File util.py

1.1.1. Fungsi tetangga

Tipe luaran : Array of integer (nodeid)
Parameter : Id graf, matriks adjacent graf
Prekondisi : id dan graph valid dan tidak kosong
Kegunaan : Mencari tetangga dari suatu simpul dengan indeks id

```python
def tetangga(id, graph):
    ttg = []
    for i, adj in enumerate(graph[id]):
        if adj != 0:
            ttg.append(i)
    return ttg
```

Gambar 1.1.1. Implementasi fungsi tetangga

1.1.2. Fungsi getdistancefrompath

Tipe luaran : Float
Parameter : Array of simpul (membentuk jalur) dan hasil parsing file
Prekondisi : Path tidak kosong, parsed valid dan terdefinisi
Kegunaan : Menghitung jarak dari suatu jalur pada path

```python
def getdistancefrompath(path, parsed):
    out = []
    node = parsed[1]
    adj = parsed[3]
    distance = 0.0

    for i in range(len(path)-1):
        out.append((node.index(path[i]), node.index(path[i+1])))

    for edge in out:
        distance += adj[edge[0]][edge[1]]

    return distance
```

Gambar 1.1.2. Implementasi fungsi getdistancefrompath

1.1.3. Fungsi haversine

Tipe luaran : Float
Parameter : koordinat 1 dan koordinat 2
Prekondisi : pos1 dan pos2 adalah tipe koordinat yang sesuai (gambar 1.1.3)
Kegunaan : Menghitung jarak antar dua koordinat dengan formula haversine

```python
def haversine(pos1, pos2):
  # pos type
  # {x:int, y:int}
  # x and y must be latitude and longitude

  # reference: https://en.wikipedia.org/wiki/Haversine_formula
  r = 6371 # jari2 bumi dalam km
  deg = pi/180
  dlat = (pos2["x"]-pos1["x"])*deg
  dlon = (pos2["y"]-pos1["y"])*deg
  akar = sin(dlat/2)**2 + cos(pos2["x"]*deg) * cos(pos1["x"]*deg) * sin(dlon/2)**2
  return 2*r*asin(sqrt(akar))
```

Gambar 1.1.3. Implementasi fungsi haversine

1.1.4. Fungsi parse

Tipe luaran : List of node, array of string (simpul), array of coordinate, adjacent matrix with weight

Parameter : nama file

Prekondisi : nama file terdefinisi

Kegunaan : Memparsing file menjadi list of node, array of node (simpul), array of coordinate, dan matriks adjacent dengan bobotnya.

```
def parse(nama_file):
  file = open(nama_file, "r")

  lines = file.readlines()
  lineslen = len(lines)
  node = []
  coor = []
  adj = []

  print("Removing new line..")
  for i in range(lineslen):
    lines[i] = lines[i].replace("\n", "")

  nodetotal = int(lines[0])

  print("Getting the coordinates..")
  for i in range(1, nodetotal+1):
    split = lines[i].split(" ")
    node.append(split[0])
    coor.append({
      "x": float(split[1]),
      "y": float(split[2])
    })

  print("Getting the adj matrix..")
  for i in range(nodetotal+1, lineslen):
    split = lines[i].split(" ")
    row = []
    for jarak in split:
      row.append(int(jarak))
    adj.append(row)

  print("Parsing", nama_file, "is done")

  listnode = {}

  for i in range(len(node)):
    ttg = {}
    for nodeid in tetangga(i, adj):
      distance = haversine(coor[nodeid], coor[i])
      ttg[node[nodeid]] = distance
      adj[i][nodeid] = distance
    listnode[node[i]] = ttg

  return listnode, node, coor, adj
```

Gambar 1.1.4. Implementasi fungsi parse

1.2. File astar.py

1.2.1. Fungsi heuristics_distance

Tipe luaran : dictionary
Parameter : parsed, goal_node
Prekondisi : parsed terdefinisi, goal_node merupakan simpul yang valid berada pada graf
Kegunaan : Membuat dictionary dengan key adalah nama simpul pada graf dan value berupa jarak lurus dari simpul ke simpul tujuan

```python
def heuristics_distance(parsed, goal_node):
    for i in range(len(parsed[1])):
        # cari indeks simpul tujuan
        if parsed[1][i] == goal_node:
            end = i
    # Buat dictionary key = node, value = jarak lurus node ke simpul tujuan
    goal_dist = []
    h={}
    for i in range(len(parsed[1])):
        goal_dist.append(haversine(parsed[2][end],parsed[2][i]))
        h[parsed[1][i]] = goal_dist[i]
    return h
```

Gambar 1.2.1. Implementasi fungsi heuristics_distance

1.2.2. Fungsi sort_f

Tipe luaran : list of string
Parameter : list (list of string), dictf(dictionary)
Prekondisi : list dan dictf terdefinisi
Kegunaan : Mengurutkan list simpul sesuai dengan nilai F value secara menaik

```python
def sort_f (list, dictf):
    # Mengurutkan list simpul sesuai dengan F value terminimum
    dicttemp = {}
    for node in list:
        if dictf.get(node, "Not Available") != "Not Available":
            dicttemp[node] = dictf[node]
    dicttemp = dict(sorted(dicttemp.items(), key=lambda item: item[1]))
    sorted_list = []
    for i in dicttemp.keys():
        sorted_list.append(i)
    return sorted_list
```

Gambar 1.2.2. Implementasi fungsi sort_f

1.2.3. Fungsi astar_search

Tipe luaran : list of string
Parameter : parsed (hasil parsing file), heuristics (list of string), start_node

(string), goal_node (string)

Prekondisi : parsed dan heuristics terdefinisi, start_node dan goal_node merupakan simpul yang valid berada pada graf

Kegunaan : Mencari jalur pencarian dari simpul awal (start_node) ke simpul tujuan (goal_node) menggunakan algoritma A*

```python
# A* search
def astar_search(parsed, heuristics, start_node, goal_node):

    # Inisialisasi open node dan closed node
    open_node = []
    closed_node = []

    # Bikin dictionary prev, key = node dan value = parent node
    prev = {}
    for i in parsed[1]:
        prev[i] = None

    # Bikin dictionary F value, G value
    dict_f= {}
    dict_f[start_node] = heuristics[start_node]

    dict_g = {}
    dict_g[start_node] = 0

    # Append simpul awal ke list open node
    open_node.append(start_node)
```

```python
    # Looping open node hingga open node kosong
    while len(open_node) > 0:
        # Ambil simpul yang memiliki f value terkecil
        open_node = sort_f(open_node, dict_f)
        current_node = open_node.pop(0)
        closed_node.append(current_node)

        # Jika sudah mencapai simpul tujuan
        if current_node == goal_node:
            path = []
            while current_node != start_node:
                path.append(current_node)
                current_node = prev[current_node]
            path.append(start_node)
            return path[::-1]

        # looping simpul tetangga dari current node
        neighbors = parsed[0][current_node]

        for neighbor in neighbors.keys():
            # Kasus simpul sudah diperiksa
            if(neighbor in closed_node):
                continue
            prev[neighbor] = current_node

            # Update nilai g value dan f value jika f value baru lebih minimum
            if(dict_g[current_node] + neighbors[neighbor] + heuristics[neighbor] < dict_f.get(neighbor, 99999999)):
                dict_g[neighbor] = dict_g[current_node] + neighbors[neighbor]
                dict_f[neighbor] = dict_g[neighbor] + heuristics[neighbor]
                open_node.append(neighbor)
    # Return None jika tidak ada jalur
    return None
```

Gambar 1.2.3. Implementasi fungsi astar_search

1.3. File graphdrawer.py

1.3.1. Prosedur drawgraph

Tipe luaran : -

Parameter : tipe graf, hasil parsing file, dan jalur

Prekondisi : Tipe graf harus terdefinisi dan valid, hasil parsing file harus terdefinisi dan valid, path boleh kosong

Kegunaan : Menggambar graf dengan tipe tertentu dari hasil parsing file dan mewarnai jalurnya jika ada

```
def drawgraph(type, parsed, path = None):
    # reference = https://networkx.org/documentation/stable/reference/drawing.ht

    # kamus
    G = nx.Graph()
    colored = False
    node = parsed[1]
    coor = parsed[2]
    adj = parsed[3]

    pos = {} # buat posisi graf (x,y) kalo dipake
    edgelabel = {} # buat label jarak

    # coloring
    # reference: https://stackoverflow.com/questions/25639169/networkx-change-co
    #            https://stackoverflow.com/questions/27030473/how-to-set-colors-

    nodecolor = []
    #edgecolor = []
    pathcolor = []
    if path is not None:
        colored = True
        pathcolor = getedgefrompath(path)

    # iterasi buat assign graf ke visualizer
    for i in range(len(adj)):
        # assign posisi graf dari input ke visualizer
        pos[node[i]] = (coor[i]["x"], coor[i]["y"])
        for j in range(len(adj[i])):
            if (i != j and i < j and adj[i][j] != 0):
                # assign warna edge
                color = None
                if (node[i],node[j]) in pathcolor:
                    #edgecolor.append("red")
                    color = "red"
                else:
                    #edgecolor.append("black")
                    color = "black"

                # assign edge ke visualizer
                G.add_edge(node[i],node[j],color=color)
                edgelabel[(node[i], node[j])] = '%.2f'%adj[i][j]

    for node in G:
        # assign warna node
        if colored and node in path:
            nodecolor.append("red")
        else:
            nodecolor.append("white")
```

```
# type = tipe graf
# -1 = default (pake x y)
# 0 = planar
# 1 = circular
# 2 = spectral
# 3 = spring
# 4 = shell

options = {
    "with_labels": True,
    "node_color": nodecolor,
    "edge_color": [G[i][j]['color'] for i,j in G.edges()],
    "edgecolors": "black"
}

if type == -1:
    nx.draw_networkx(G, pos, **options)
elif type == 0:
    pos = nx.planar_layout(G)
    nx.draw_planar(G, **options)
elif type == 1:
    pos = nx.circular_layout(G)
    nx.draw_circular(G, **options)
elif type == 2:
    pos = nx.spectral_layout(G)
    nx.draw_spectral(G, **options)
elif type == 3:
    pos = nx.spring_layout(G)
    nx.draw_spring(G, **options)
elif type == 4:
    pos = nx.shell_layout(G)
    nx.draw_shell(G, **options)

nx.draw_networkx_edge_labels(G, pos, edge_labels = edgelabel)

# Set margins for the axes so that nodes aren't clipped
ax = plt.gca()
ax.margins(0.20)
plt.axis("off")
plt.show()
```

Gambar 1.3.1. Implementasi prosedur drawgraph

1.3.2. Fungsi getedgefrompath

Tipe luaran : Array of tuple
Parameter : Path (array of string (simpul))
Prekondisi : Path tidak kosong
Kegunaan : Men-generate sisi dari simpul yang dilewati path

```
def getedgefrompath(path):
    out = []
    for i in range(len(path)-1):
        out.append((path[i], path[i+1]))
        out.append((path[i+1], path[i]))
    return out
```

Gambar 1.3.2. Implementasi fungsi getedgefrompath

1.4. File program.py

1.4.1. Prosedur start

Tipe luaran : -
Parameter : Nama file (jika ada)
Prekondisi : -
Kegunaan : Memulai program dengan meminta file dan menggambar graf hasil parsing file tersebut

```
def start(filename = None):
    if filename is None:
        filename = input("Masukkan nama file: ")

    global parsed
    parsed = util.parse(filename)

    print()
    print("Visualisasi graf masukan (bobot dalam km): ")
    graphdrawer.drawgraph(type, parsed)
```

Gambar 1.4.1. Implementasi prosedur start

1.4.2. Prosedur process

Tipe luaran : -
Parameter : Simpul awal dan simpul tujuan
Prekondisi : -
Kegunaan : Meminta masukan nama simpul awal dan simpul tujuan untuk dicari jalurnya dengan algoritma A*.

```
def process(startNode = None, goalNode = None):
    if startNode is None or goalNode is None:
        startNode = input("Simpul awal: ")
        goalNode = input("Simpul tujuan: ")

    heuristic = astar.heuristics_distance(parsed, goalNode)
    searchPath = astar.astar_search(parsed, heuristic, startNode, goalNode)

    if searchPath is not None:
        print()
        print("Hasil: ")
        distance = util.getdistancefrompath(searchPath, parsed)
        print("Jarak terpendek dari", startNode, "dan", goalNode, "adalah", '%.2f'%distance, "km")
        graphdrawer.drawgraph(type, parsed, searchPath)
    else:
        print("No path found")
```

Gambar 1.4.2. Implementasi prosedur process

1.4.3. Prosedur setgraphtype

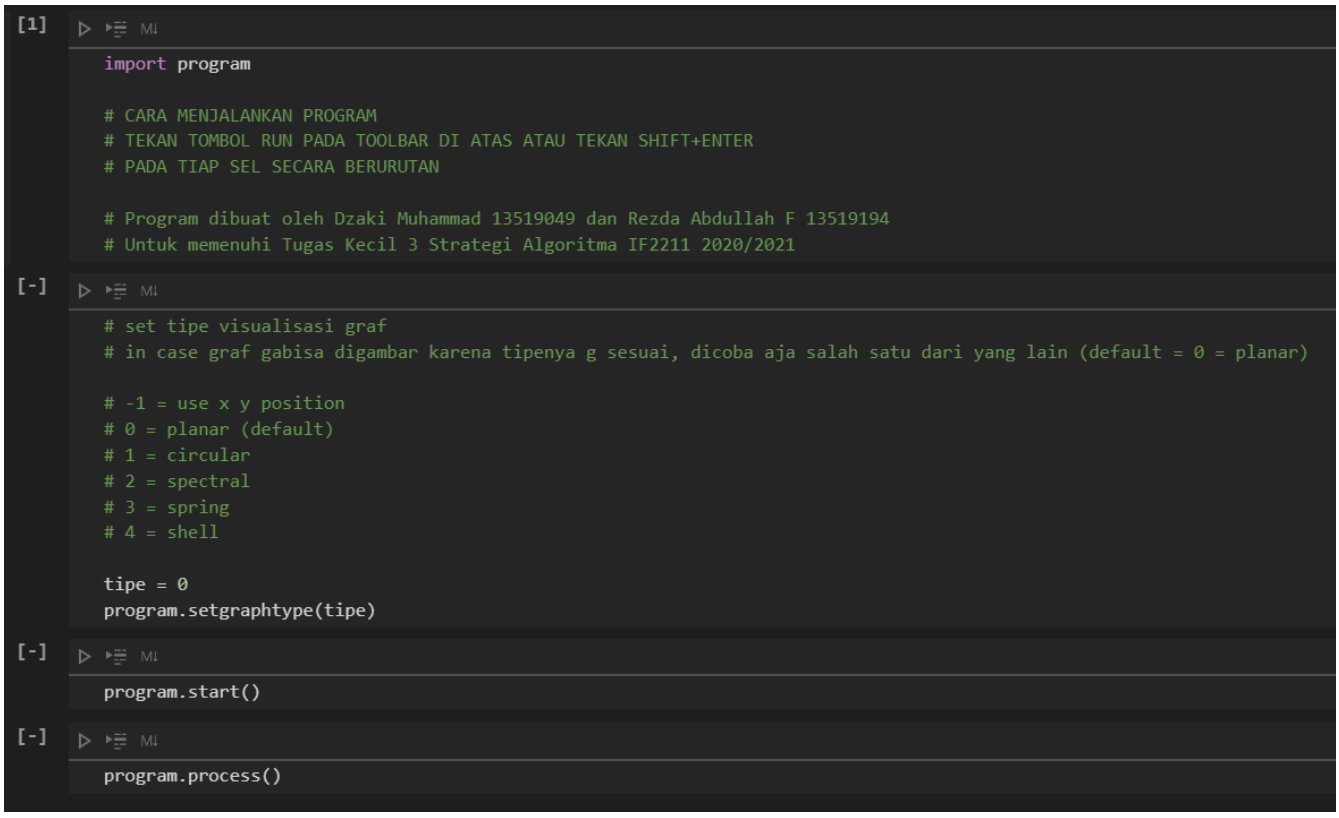
Tipe luaran : -
Parameter : Tipe graf
Prekondisi : Tipe graf harus terdefinisi dan valid
Kegunaan : Mengubah tipe graf untuk digambar

```
def setgraphtype(tipe):
    if tipe == -1 or tipe == 0 or tipe == 1 or tipe == 2 or tipe == 3 or tipe == 4:
        # type = tipe graf
        # -1 = default (pake x y)
        # 0 = planar
        # 1 = circular
        # 2 = spectral
        # 3 = spring
        # 4 = shell
        # 5 = random
        name = ["default", "planar", "circular", "spectral", "spring", "shell"]
        global type
        type = tipe
        print("Graph type:", name[tipe+1])
    else:
        raise Exception("Wrong type")
```

Gambar 1.4.1. Implementasi prosedur start

1.5. File Tucil3.ipynb

File Tucil3.ipynb merupakan program utama yang dapat menampilkan visualisasi graf dari file masukan, meminta masukan pengguna, dan menampilkan visualisasi lintasan terpendek dari simpul masukan pengguna berdasarkan algoritma A* pada graf.



```
[1]
import program

# CARA MENJALANKAN PROGRAM
# TEKAN TOMBOL RUN PADA TOOLBAR DI ATAS ATAU TEKAN SHIFT+ENTER
# PADA TIAP SEL SECARA BERURUTAN

# Program dibuat oleh Dzaki Muhammad 13519049 dan Rezda Abdullah F 13519194
# Untuk memenuhi Tugas Kecil 3 Strategi Algoritma IF2211 2020/2021

[-]
# set tipe visualisasi graf
# in case graf gabisa digambar karena tipenya g sesuai, dicoba aja salah satu dari yang lain (default = 0 = planar)

# -1 = use x y position
# 0 = planar (default)
# 1 = circular
# 2 = spectral
# 3 = spring
# 4 = shell

tipe = 0
program.setgraphtype(tipe)

[-]
program.start()

[-]
program.process()
```

Gambar 1.5. Implementasi program jupyter notebook Tucil3.ipynb

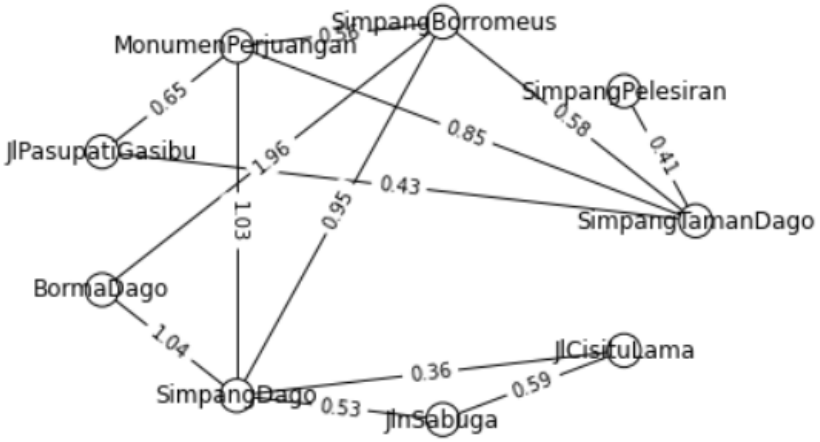
BAB 2

KASUS UJI

2.1. Peta jalan sekitar kampus ITB/Dago

Simpul graf :

1. SimpangTamanDago. Koordinat: -6.898914, 107.612686
2. SimpangPelesiran. Koordinat: -6.896861, 107.609609
3. SimpangBorromeus. Koordinat: -6.893749, 107.612943
4. JlnSabuga. Koordinat: -6.887330, 107.609360
5. JlCisituLama. Koordinat: -6.882602, 107.611693
6. BormaDago. Koordinat: -6.876839, 107.617807
7. MonumenPerjuangan. Koordinat: -6.893374, 107.618035
8. JlPasupatiGasibu. Koordinat: -6.899059, 107.616565
9. SimpangDago. Koordinat : -6.885205 107.613663



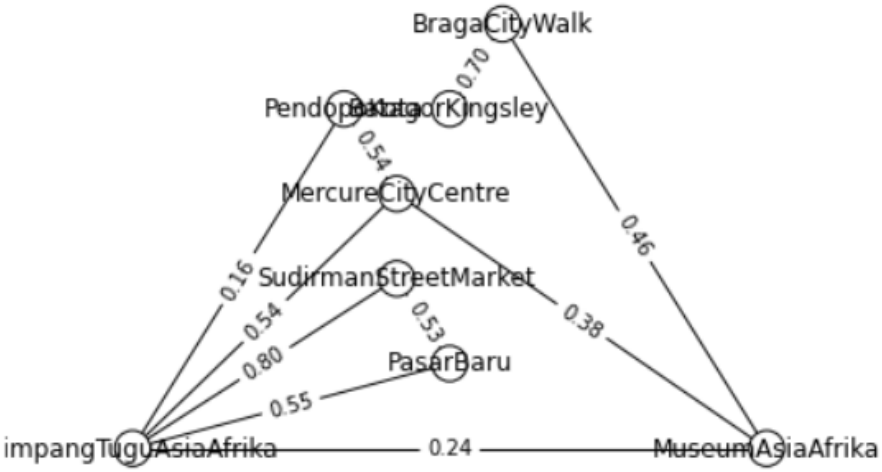
Gambar 2.1. Visualisasi graf peta jalan sekitar kampus ITB/Dago

2.2. Peta jalan sekitar Alun-alun Bandung

Simpul graf :

1. SimpangTuguAsiaAfrika. Koordinat : -6.921210, 107.607689
2. MuseumAsiaAfrika. Koordinat : -6.921023, 107.609848
3. BragaCityWalk. Koordinat : -6.916968, 107.609178
4. MercureCityCentre. Koordinat : -6.923878, 107.611800
5. SudirmanStreetMarket. Koordinat : -6.920395, 107.600530
6. PendopoKota. Koordinat : -6.922503, 107.607066

7. PasarBaru. Koordinat : -6.917585, 107.604353
8. BatagorKingsley. Koordinat : -6.919173, 107.615111

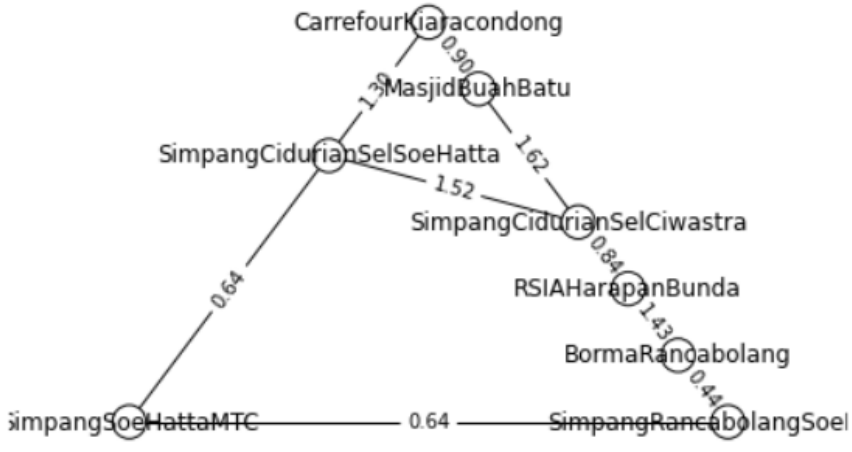


Gambar 2.2. Visualisasi graf peta jalan sekitar Alun-alun Bandung

2.3. Peta jalan sekitar Buahbatu

Simpul graf :

1. SimpangSoeHattaMTC. Koordinat : -6.940351, 107.658245
2. SimpangRancabolangSoeHatta. Koordinat : -6.939252, 107.66391
3. BormaRancabolang. Koordinat : -6.943234, 107.66356
4. SimpangCidurianSelSoeHatta. Koordinat : -6.942138, 107.652719
5. SimpangCidurianSelCiwastra. Koordinat : -6.955690, 107.654484
6. RSIAHarapanBunda. Koordinat : -6.956029, 107.662112
7. MasjidBuahBatu. Koordinat : -6.954222, 107.639885
8. CarrefourKiaracondong. Koordinat : -6.946367, 107.641756

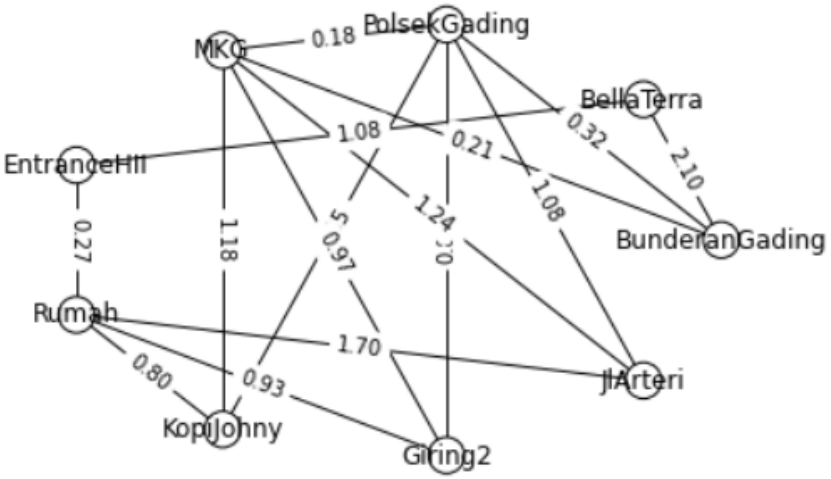


Gambar 2.3. Visualisasi graf peta jalan sekitar Buahbatu

2.4. Peta jalan sebuah kawasan di Kelapa Gading

Simpul graf :

1. BunderanGading. Koordinat : -6.160544, 106.905369
2. BellaTerra. Koordinat : -6.175968, 106.894382
3. EntranceHII. Koordinat : -6.179253, 106.903545
4. PolsekGading. Koordinat : -6.162058, 106.907879
5. Rumah. Koordinat : -6.177274, 106.904999
6. KopiJohny. Koordinat : -6.170336, 106.903228
7. Giring2. Koordinat : -6.169280, 106.907381
8. JlArteri. Koordinat : -6.166997, 106.916339
9. MKG. Koordinat : -6.160542853447745, 106.90724221853478



Gambar 2.4. Visualisasi graf peta jalan kawasan di Kelapa Gading

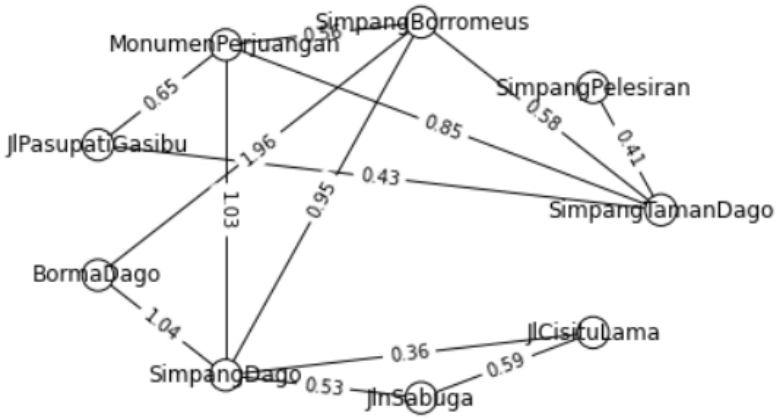
BAB 3

HASIL UJI

3.1. Kasus Uji 1 pada TC 1

Masukan :

```
Visualisasi graf masukan (bobot dalam km):
```

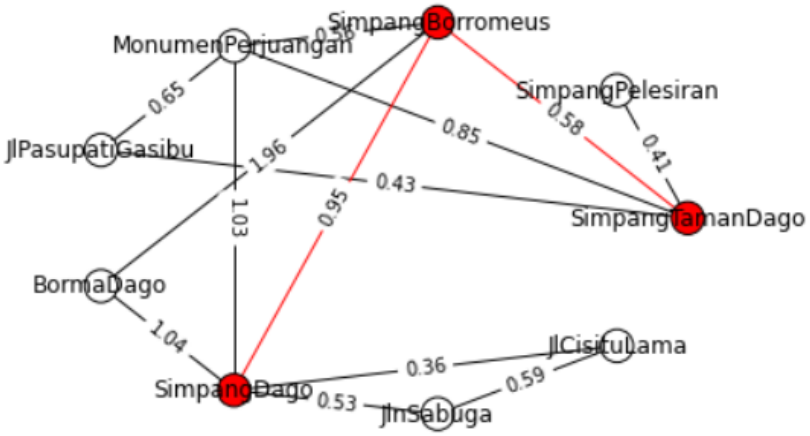


Gambar 3.1.1 Isi file masukan kasus uji 1 TC 1

Luaran :

```
Simpul awal:  SimpangDago
Simpul tujuan:  SimpangTamanDago

Hasil:
Jarak terpendek dari SimpangDago dan SimpangTamanDago adalah 1.53 km
```

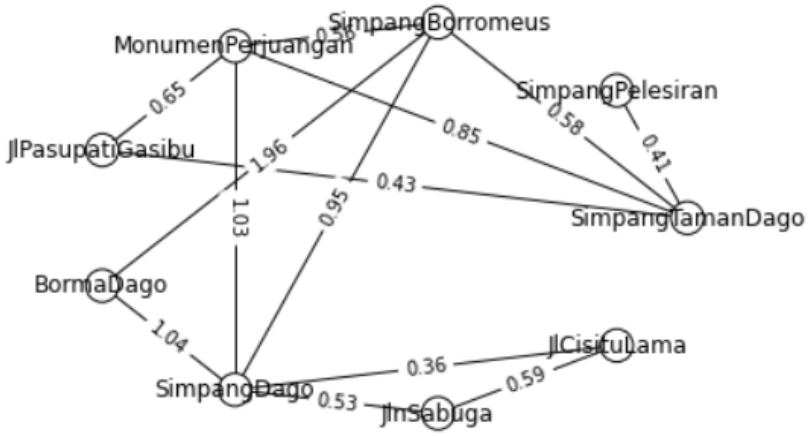


Gambar 3.1.2 Luaran kasus uji 1 TC 1

3.2. Kasus Uji 2 pada TC 1

Masukan :

```
Visualisasi graf masukan (bobot dalam km):
```

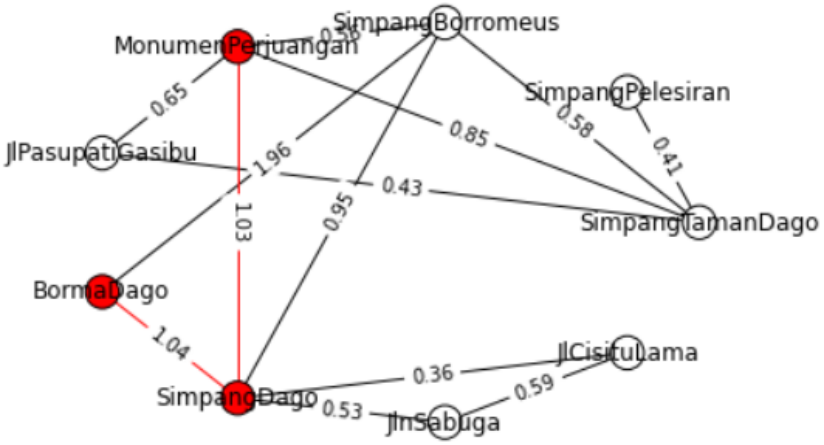


Gambar 3.2.1 Isi file masukan kasus uji 2 TC 1

Luaran :

```
Simpul awal:  MonumenPerjuangan
Simpul tujuan:  BormaDago

Hasil:
Jarak terpendek dari MonumenPerjuangan dan BormaDago adalah 2.07 km
```

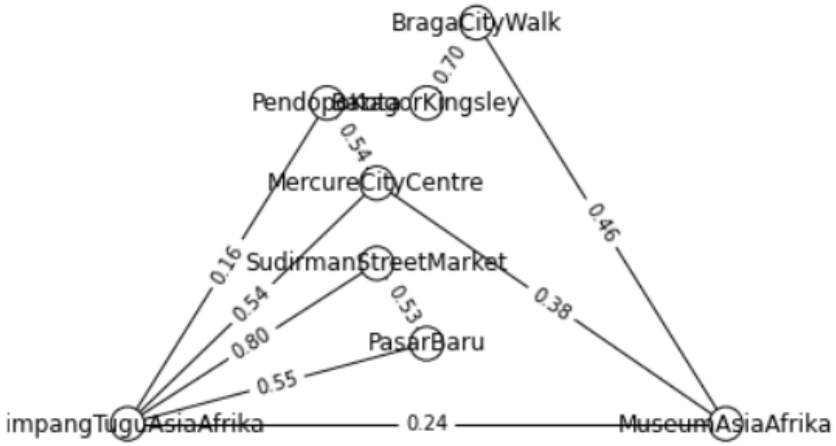


Gambar 3.2.2 Luaran kasus uji 2 TC 1

3.3. Kasus Uji 1 pada TC 2

Masukan :

```
Visualisasi graf masukan (bobot dalam km):
```

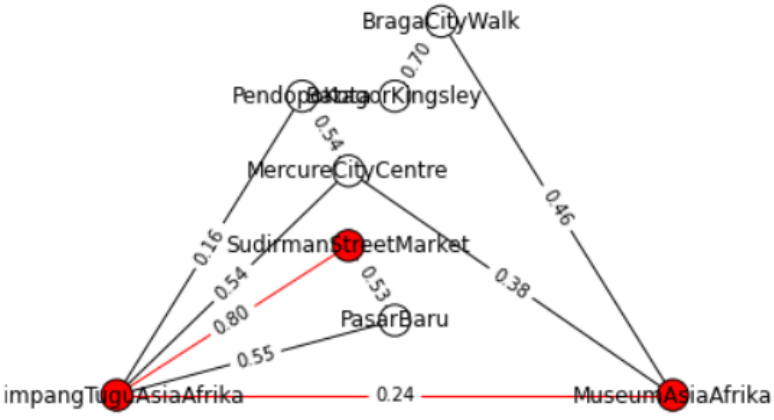


Gambar 3.3.1 Isi file masukan kasus uji 1 TC 2

Luaran :

```
Simpul awal:  SudirmanStreetMarket
Simpul tujuan:  MuseumAsiaAfrika

Hasil:
Jarak terpendek dari SudirmanStreetMarket dan MuseumAsiaAfrika adalah 1.03 km
```

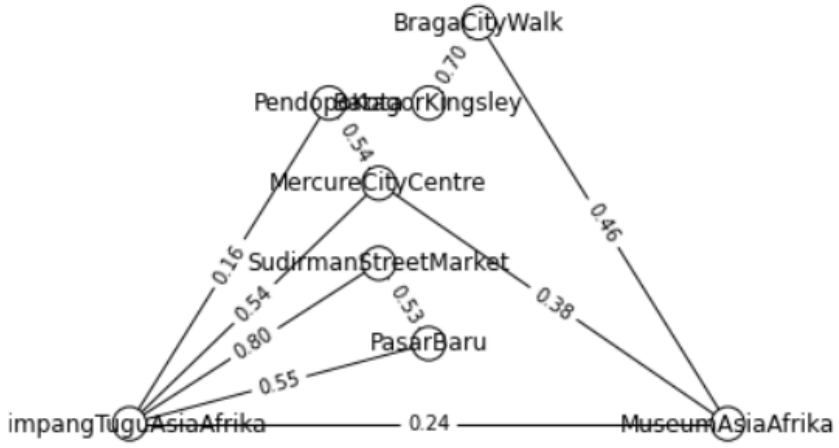


Gambar 3.3.2 Luaran kasus uji 1 TC 2

3.4. Kasus Uji 2 pada TC 2

Masukan :

```
Visualisasi graf masukan (bobot dalam km):
```

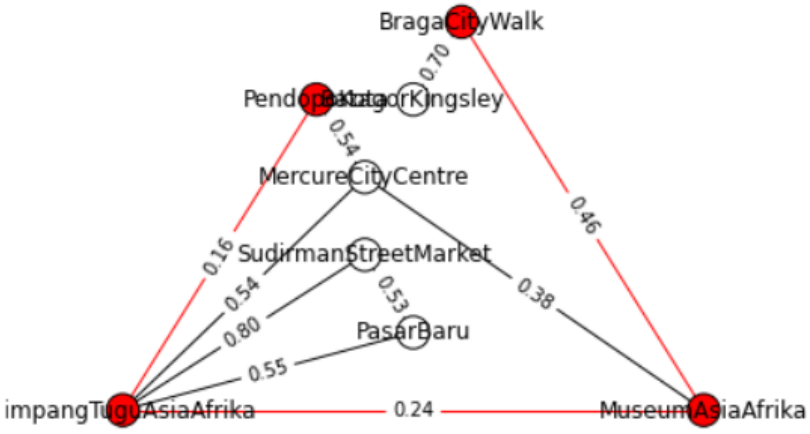


Gambar 3.4.1 Isi file masukan kasus uji 2 TC 2

Luaran :

```
Simpul awal:  PendopoKota
Simpul tujuan:  BragaCityWalk

Hasil:
Jarak terpendek dari PendopoKota dan BragaCityWalk adalah 0.86 km
```

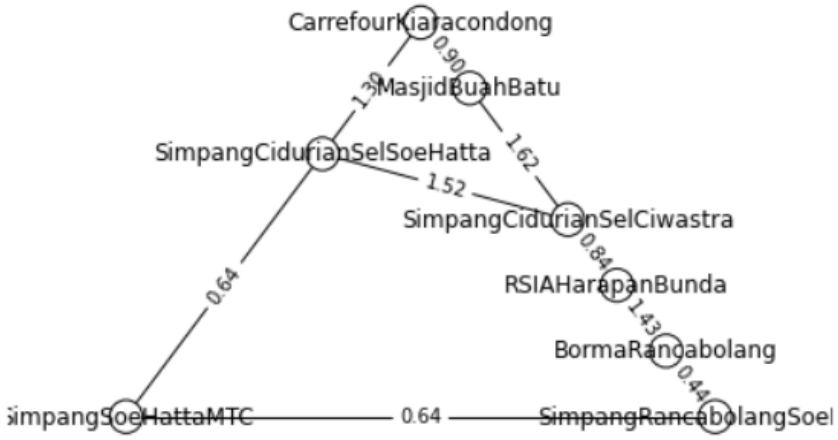


Gambar 3.4.2 Luaran kasus uji 2 TC 2

3.5. Kasus Uji 1 pada TC 3

Masukan :

```
Visualisasi graf masukan (bobot dalam km):
```

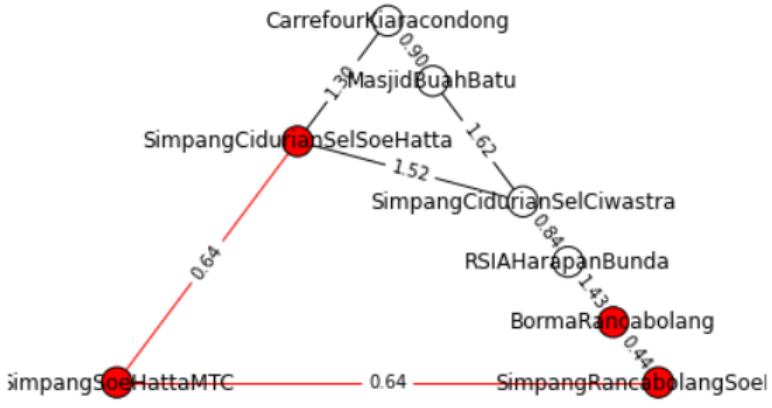


Gambar 3.5.1 Isi file masukan kasus uji 1 TC 3

Luaran :

```
Simpul awal:  SimpangCidurianSelSoeHatta
Simpul tujuan:  BormaRancabolang

Hasil:
Jarak terpendek dari SimpangCidurianSelSoeHatta dan BormaRancabolang adalah 1.72 km
```

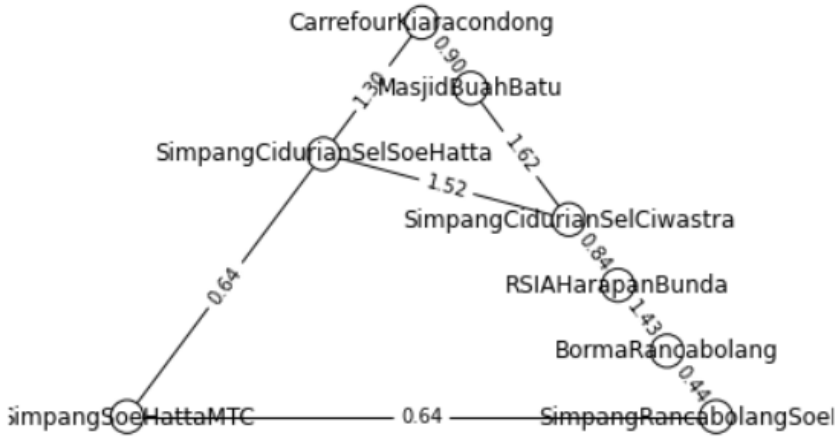


Gambar 3.5.2 Luaran kasus uji 1 TC 3

3.6. Kasus Uji 2 pada TC 3

Masukan :

```
Visualisasi graf masukan (bobot dalam km):
```

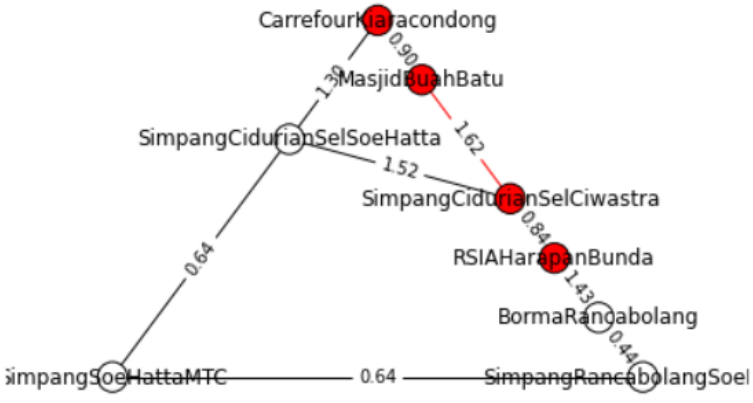


Gambar 3.6.1 Isi file masukan kasus uji 2 TC 3

Luaran :

```
Simpul awal:  RSIAHarapanBunda
Simpul tujuan:  CarrefourKiaracondong

Hasil:
Jarak terpendek dari RSIAHarapanBunda dan CarrefourKiaracondong adalah 3.36 km
```

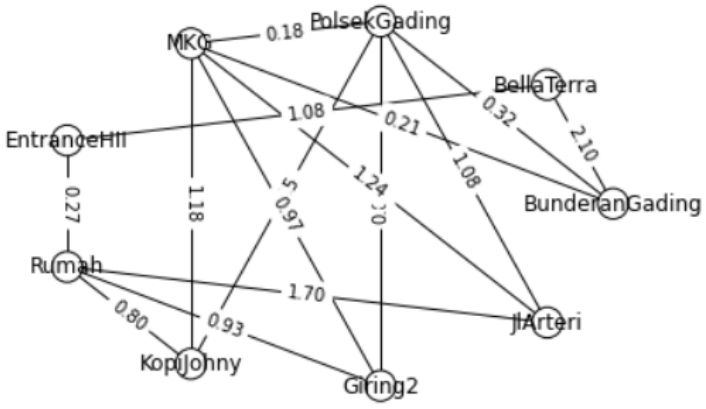


Gambar 3.6.2 Luaran kasus uji 2 TC 3

3.7. Kasus Uji 1 pada TC 4

Masukan :

```
Visualisasi graf masukan (bobot dalam km):
```

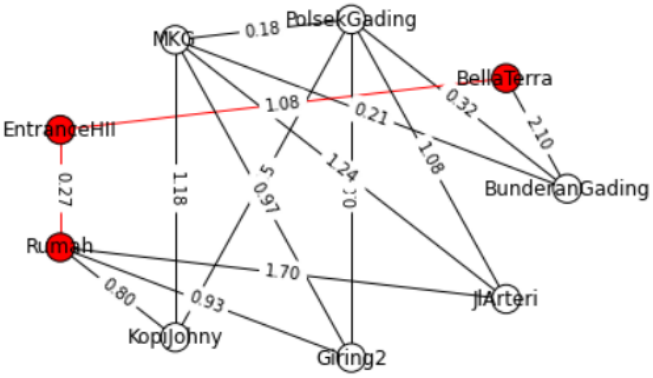


Gambar 3.7.1 Isi file masukan kasus uji 1 TC 4

Luaran :

```
Simpul awal: Rumah
Simpul tujuan: BellaTerra

Hasil:
Jarak terpendek dari Rumah dan BellaTerra adalah 1.35 km
```

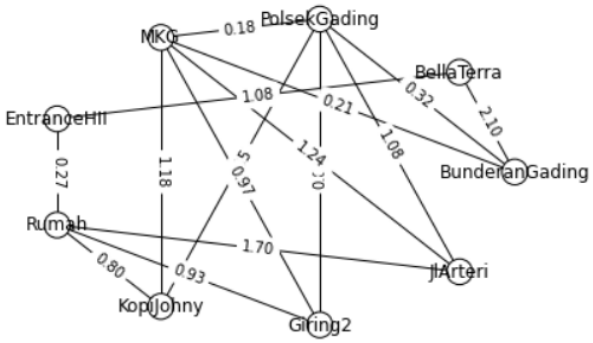


Gambar 3.7.2 Luaran kasus uji 1 TC 4

3.8. Kasus Uji 2 pada TC 4

Masukan :

```
Visualisasi graf masukan (bobot dalam km):
```

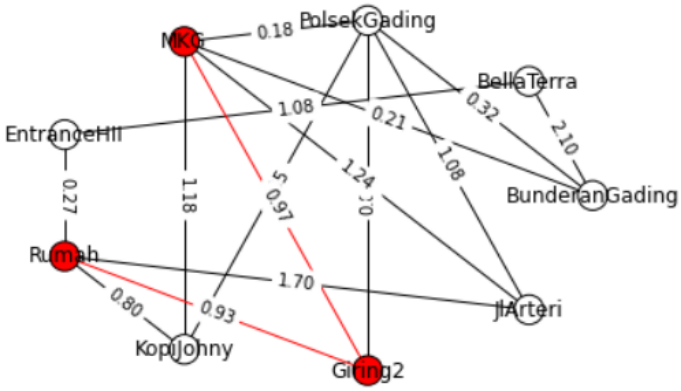


Gambar 3.8.1 Isi file masukan kasus uji 2 TC 4

Luaran :

```
Simpul awal: Rumah
Simpul tujuan: MKG

Hasil:
Jarak terpendek dari Rumah dan MKG adalah 1.90 km
```



Gambar 3.8.2 Luaran kasus uji 2 TC 4

LAMPIRAN

Lampiran 1

Checklist penilaian :

No	Poin Penilaian	Centang (□) jika ya
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	-

Lampiran 2

Alamat link *github* repositori program : https://github.com/dzakimuhammad/Tucil3Stima