Tugas Kecil 3 IF2122 Strategi Algoritma
Implementasi Algoritma A* untuk Menentukan Lintasan Terpendek

Dibuat dalam rangka:
Tugas Kecil 3 IF-2211 Strategi Algoritma

Oleh:
Haning Nanda Hapsari 13519042
Farhan Fadillah Rafi 13519204

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

KODE PROGRAM

1. gambarnode.py

```
import networkx as nx
import matplotlib.pyplot as plt

def gambarjadi(edge,hasill,bobot):
    G = nx.DiGraph()
    for i in range(len(edge)):
        G.add_edges_from([edge[i]], weight =
'{:.2f}'.format(bobot[i]*1000))

    edge_labels = dict([((u, v,), d['weight'])
                        for u, v, d in G.edges(data=True)])
    # Specify the edges you want here
    red_edges = hasill
    edge_colours = ['black' if not edge in red_edges else 'red'
                    for edge in G.edges()]
    black_edges = [edge for edge in G.edges() if edge not in
red_edges]

    pos = nx.spring_layout(G)
    nx.draw_networkx_nodes(G, pos, cmap=plt.get_cmap('jet'),
                           node_size = 600)

    nx.draw_networkx_labels(G, pos,
font_size=5,horizontalalignment='center')
    nx.draw_networkx_edges(G, pos, edgelist=red_edges, edge_color='r',
arrows=False)
    nx.draw_networkx_edges(G, pos, edgelist=black_edges, arrows=False)
    nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels)
    fig = plt.gcf()
    fig.set_size_inches(8,8)
    plt.show()

def gambar(edge,bobot):
    G = nx.DiGraph()
    for i in range(len(edge)):
        G.add_edges_from([edge[i]], weight =
'{:.2f}'.format(bobot[i]*1000))
    edge_labels = dict([((u, v,), d['weight'])
                        for u, v, d in G.edges(data=True)])
    black_edges = edge
    edge_colours = ['' if not edge in black_edges else 'black'
                    for edge in G.edges()]
    pos = nx.spring_layout(G)
    nx.draw_networkx_nodes(G, pos, cmap=plt.get_cmap('jet'),
                           node_size = 1500)
    nx.draw_networkx_labels(G, pos,
```

```
font_size=6,horizontalalignment='center')
    nx.draw_networkx_edges(G, pos, edgelist=black_edges, arrows=False)
    nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels)
    fig = plt.gcf()
    fig.set_size_inches(8,8)
    plt.show()
```

2. main.py

```
import gambarnode as gmbr

def openfile(namafile):
    file = open(namafile)
    r = file.read()
    seplit = r.split()
    return seplit

def atr_nodes(file_read_split):
    jumlah_node = int(file_read_split[0])
    g = {}
    for i in range(jumlah_node):
        posisi_node = i * 3 + 1
        node = file_read_split[posisi_node]
        g[node] = {}
        g[node]["lat"] = file_read_split[int(posisi_node+1)]
        g[node]["long"] = file_read_split[int(posisi_node + 2)]
    return g

def list_nodes(file_read_split):
    jumlah_node = int(file_read_split[0])
    node_list = []

    for i in range(jumlah_node):
        posisi_node = i*3+1
        node_list.append(file_read_split[posisi_node])
    return node_list


def matriks_ketetangaan_to_edge(file_read_split, list_of_nodes):
    list_edge = []
    jumlah_node = int(file_read_split[0])
    initial_index_matriks = jumlah_node*3+1
    for j in range(initial_index_matriks,len(file_read_split)):
        x = (j-1) // jumlah_node - 3
        y = (j-1) % jumlah_node
        edge = ()
        temp = list(edge)
        if int(file_read_split[j]):
            temp.append(list_of_nodes[x])
            temp.append(list_of_nodes[y])
```

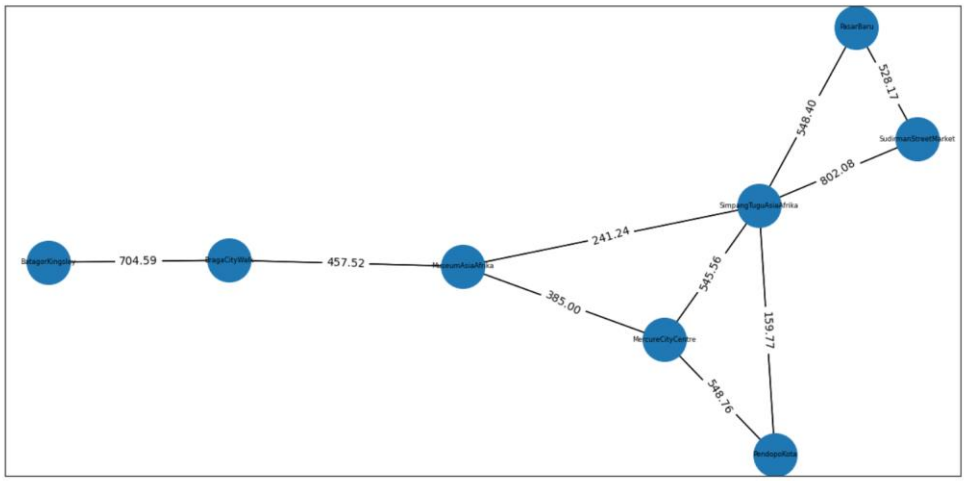
```
            edge = tuple(temp)
            list_edge.append(edge)

    return list_edge

def jarakeuclidian(init,dest,dict_atr_node):
    y1 = float(dict_atr_node[init]["long"])
    x2 = float(dict_atr_node[dest]["lat"])
    y2 = float(dict_atr_node[dest]["long"])
    x1 = float(dict_atr_node[init]["lat"])

    latlangToKM = 111.319

    count = (((x2-x1)**2 + (y2-y1)**2)**0.5) * latlangToKM

    return count

def bobot_edge(list_edge,dict_atr_node):
    bobot=[]
    for i in range(len(list_edge)):
        temp =
jarakeuclidian(list_edge[i][0],list_edge[i][1],dict_atr_node)
        bobot.append(temp)
    return bobot

def getTetangga(init,list_edges):
    temp = []
    for i in range(len(list_edges)):
        if(list_edges[i][0] == init):
            temp.append(list_edges[i][1])
    return temp

def getbobotadj (namasimpul, list_edges, atr):
    matbobot = []
    adj = getTetangga(namasimpul,list_edges)
    for i in range (len(adj)):
        matbobot.append(jarakeuclidian(namasimpul, adj[i],atr))
    return matbobot

def adjplusbobot(adj,bobotadj):
    dict = {}
    for i in range(len(adj)):
        dict[adj[i]] = bobotadj[i]
    new = {k: v for k, v in sorted(dict.items(), key=lambda item:
item[1])}
    return new

def arah(hasil):
    tempppp = []
    for i in range(len(hasil)):
        edge = ()
```

```
        temp = list(edge)
        if i > 0:
            temp.append(hasil[i - 1])
            temp.append(hasil[i])

            edge = tuple(temp)
            tempppp.append(edge)
    for j in range(len(hasil)-1,0,-1):
        edge = ()
        temp = list(edge)

        temp.append(hasil[j])
        temp.append(hasil[j-1])

        edge = tuple(temp)
        tempppp.append(edge)
    return tempppp
if __name__ == '__main__':
    file = openfile("./test/buahbatu.txt")
    listnodes = list_nodes(file)
    a = matriks_ketetangaan_to_edge(file,listnodes)
    atr = atr_nodes(file)
    jarak = jarakeuclidian(listnodes[0],listnodes[1],atr)
    bobot = bobot_edge(a,atr)

    print("Daftar Simpul :")
    for i in range(len(listnodes)):
        print(listnodes[i])

    init = str(input("Masukkan Simpul Awal : "))
    dest = str(input("Masukkan Simpul Akhir : "))

    # ALGORITMA A*
    openlist = []
    closedlist = []
    ketemu = False
    cur = init
    g = 0
    h = jarakeuclidian(init,dest,atr)
    f = g + h
    closedlist.append(init)
    while(not(ketemu)):
        ttg = getTetangga(cur, a)
        bobot_ttg = getbobotadj(cur,a,atr)
        dict = adjplusbobot(ttg,bobot_ttg)
        for i in range(len(ttg)):
            g = 0
            temp = 0
            openlist.append(ttg[i])

            tes = dict[ttg[i]] + g
```

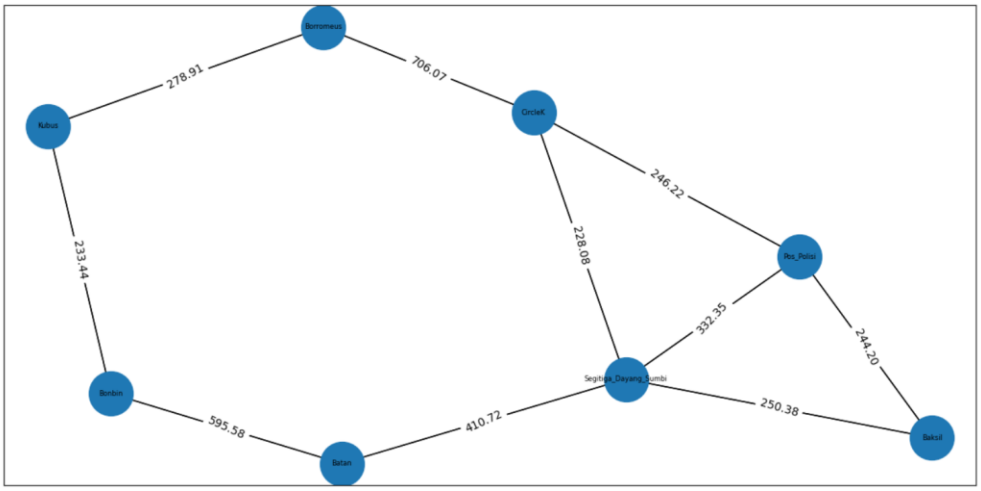
```
            g = tes
            h = jarakeuclidian(ttg[i],dest,atr)
            f = g + h
            openlist.append(f)
            openlist.append(g)
            openlist.append(h)

        for i in range(len(openlist)):
            y = i % 4
            if y == 0:
                if openlist[i] == dest:
                    ketemu = True
                    closedlist.append(dest)
                    break
        idx =0
        kecil = 999999
        pjg = len(openlist)
        for i in range(pjg):
            x = i%4
            if(x == 1):
                if(kecil > openlist[i]):
                    kecil = openlist[i]

        idx = openlist.index(kecil)
        cur = openlist[idx-1]
        if (not(ketemu)):
            closedlist.append(cur)

    hasil = arah(closedlist)
    jumlah = 0
    for i in range(len(hasil)//2):
        jumlah = jumlah+jarakeuclidian(hasil[i][0],hasil[i][1],atr)
    print("Jarak Simpul : ",jumlah*1000)
    gmbr.gambarjadi(a,hasil,bobot)
```

PETA

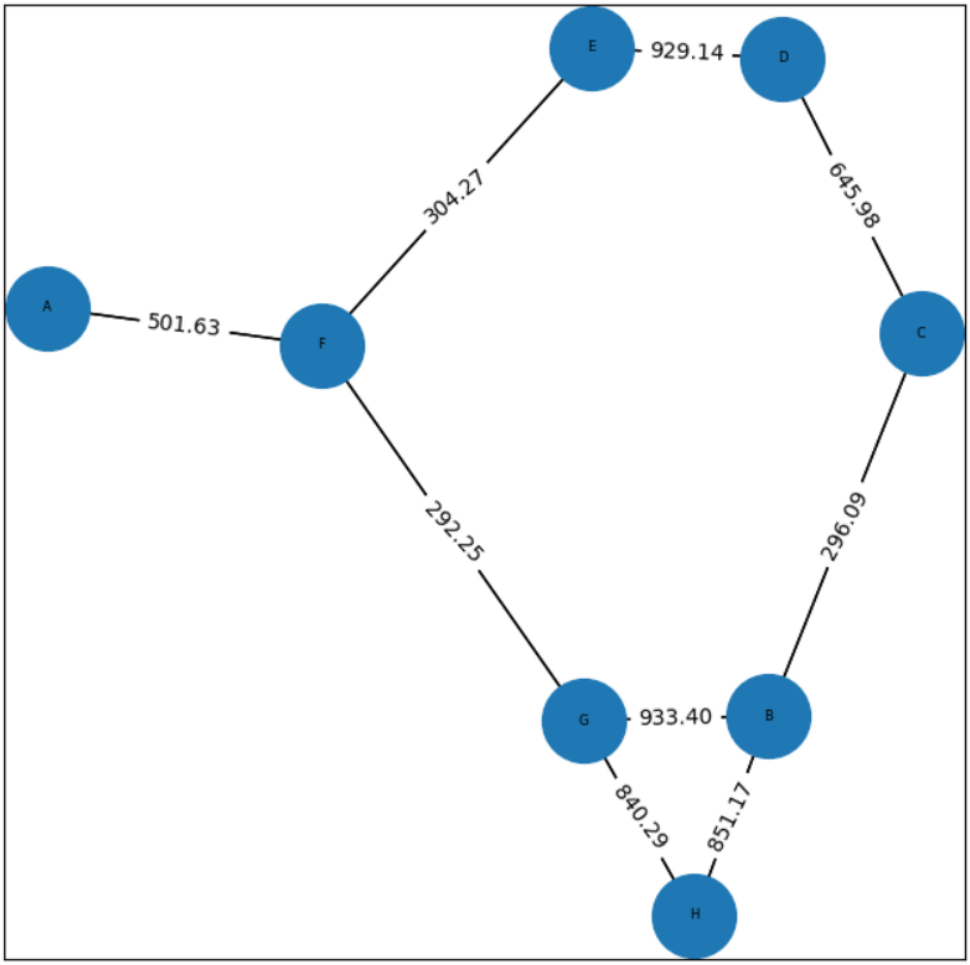
1. alun-alun.txt



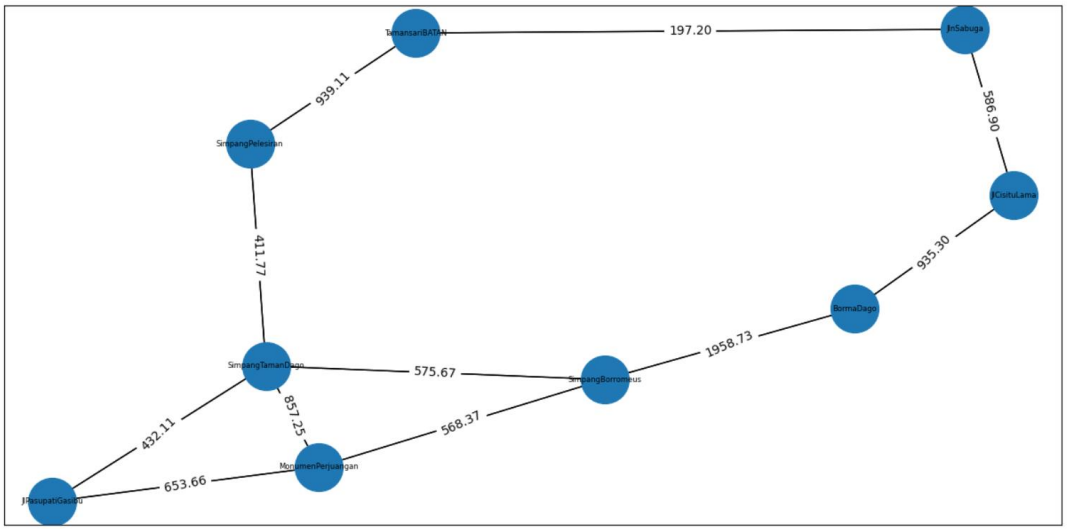
2. sekitarITB.txt



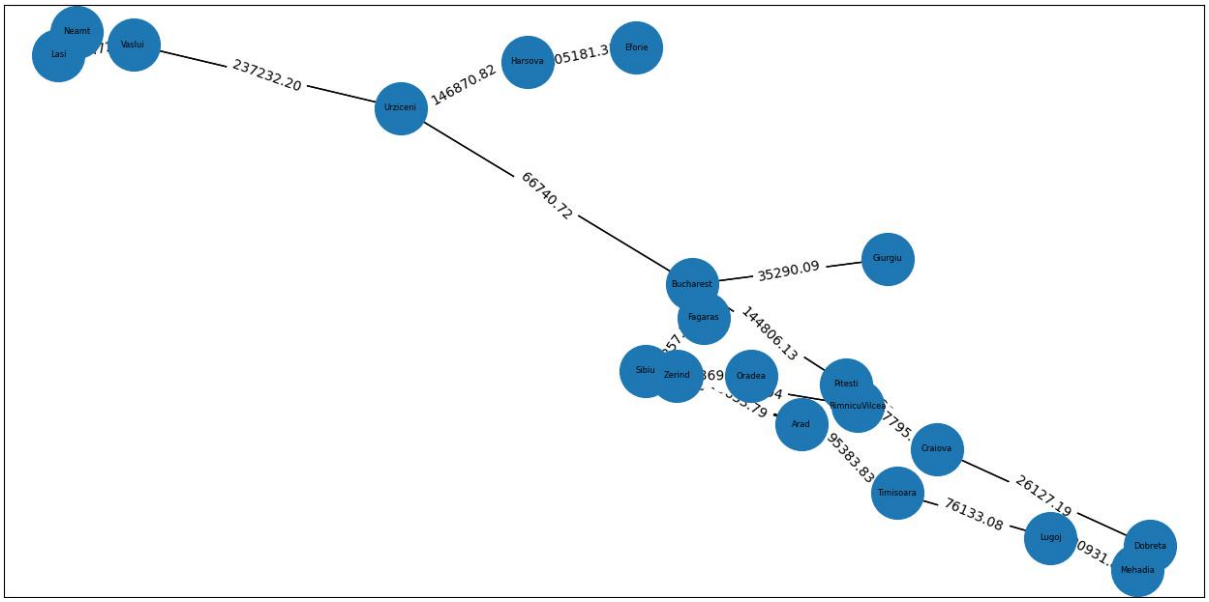
3. buahbatu.txt



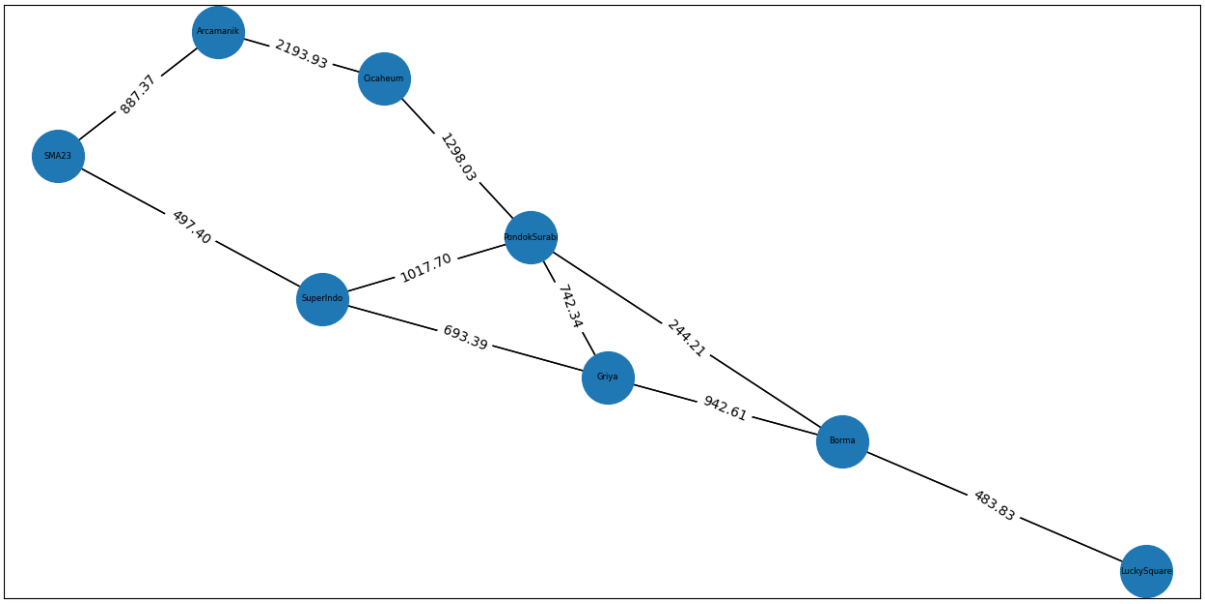
4. dago.txt



5. arad.txt

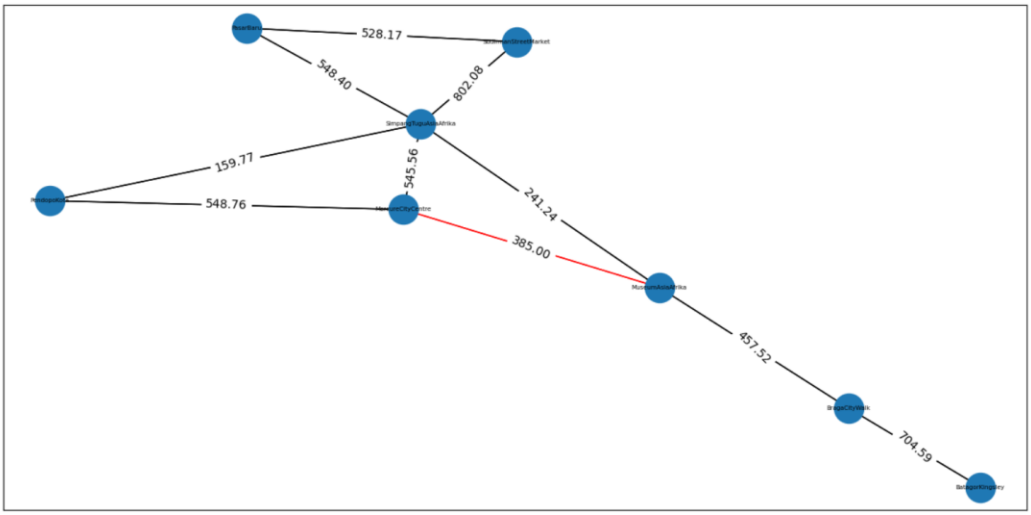


6. antapani.txt



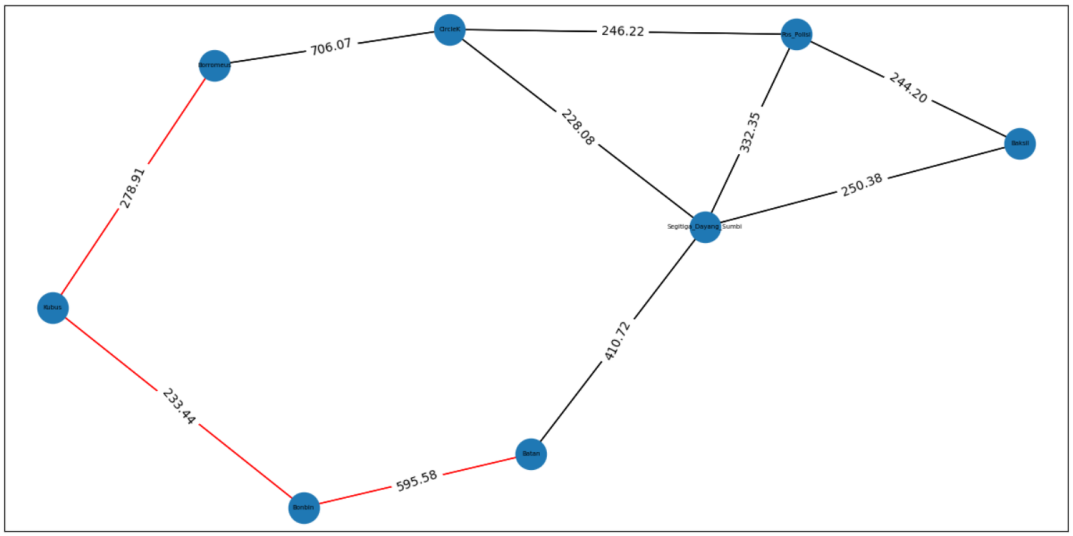
HASIL

1. alun-alun.txt



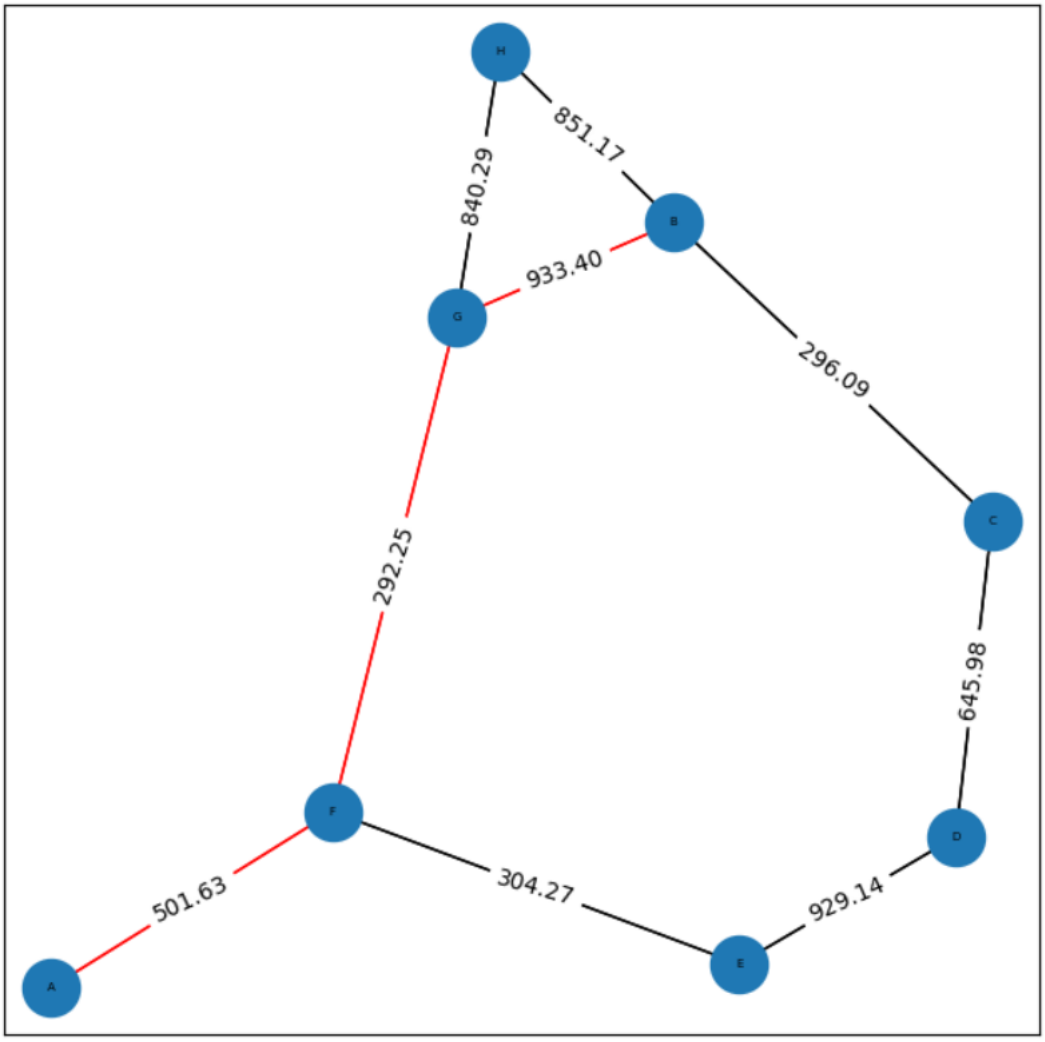

```
Daftar Simpul :
SimpangTuguAsiaAfrika
MuseumAsiaAfrika
BragaCityWalk
MercureCityCentre
SudirmanStreetMarket
PendopoKota
PasarBaru
BatagorKingsley
Masukkan Simpul Awal : MuseumAsiaAfrika
Masukkan Simpul Akhir : MercureCityCentre
Jarak Simpul :  384.9984794815158
```

2. sekitarITB.txt



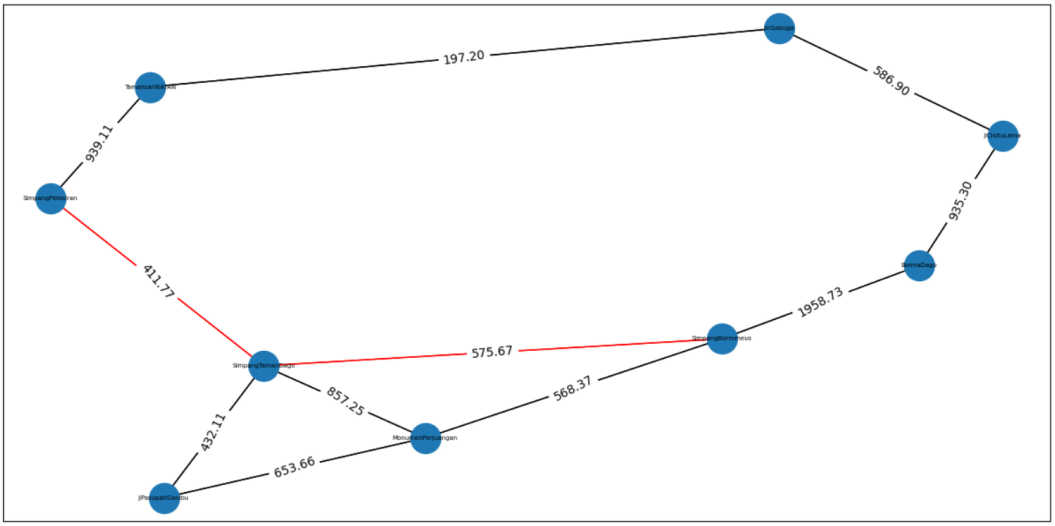
```
Daftar Simpul :
Segitiga_Dayang_Sumbi
Batan
Bonbin
Kubus
Borromeus
CircleK
Pos_Polisi
Baksil
Masukkan Simpul Awal : Borromeus
Masukkan Simpul Akhir : Batan
Jarak Simpul :  1107.928993638089
```

3. buahbatu.txt



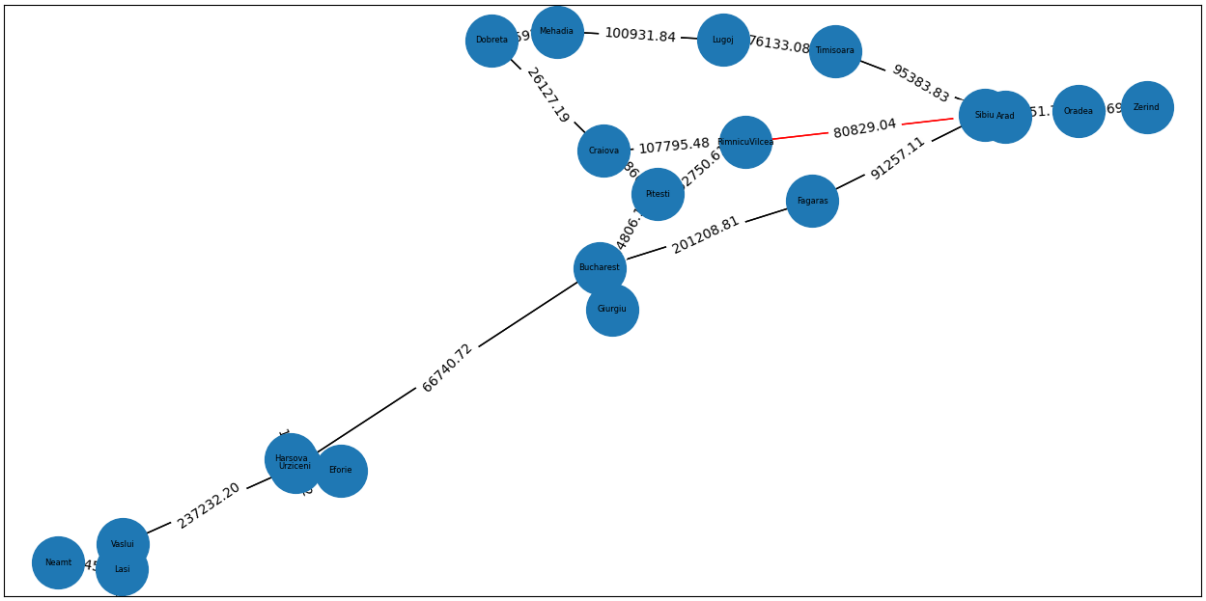
```
Daftar Simpul :
A
B
C
D
E
F
G
H
Masukkan Simpul Awal : A
Masukkan Simpul Akhir : B
Jarak Simpul :  1727.2901746680302
```

4. dago.txt



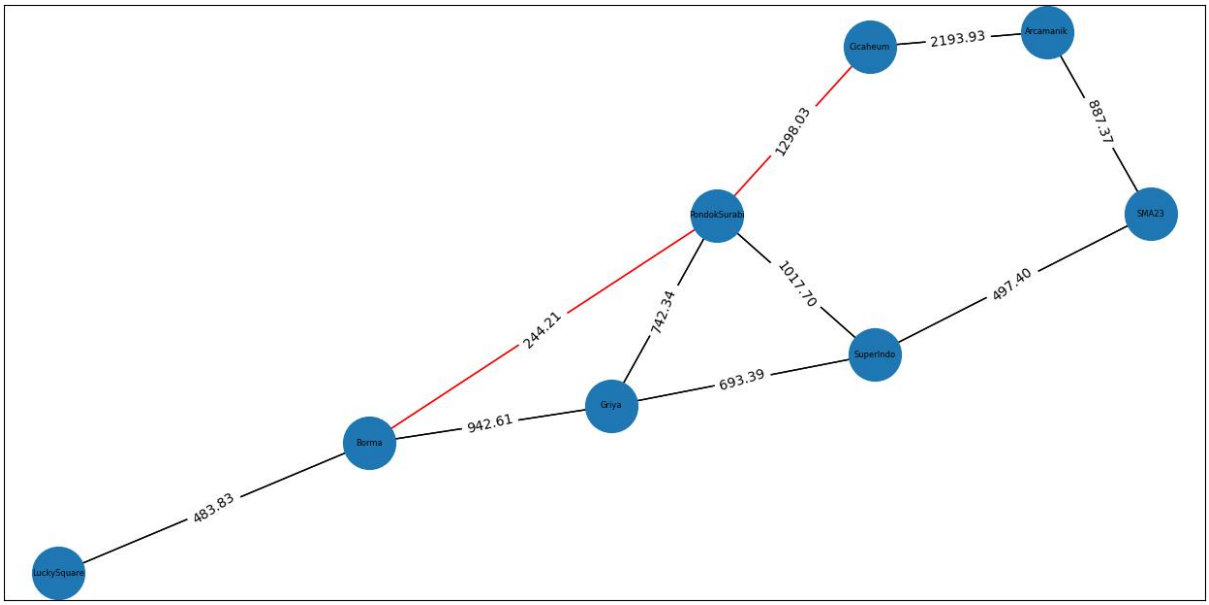

```
Daftar Simpul :
SimpangTamanDago
SimpangPelesiran
SimpangBorromeus
TamansariBATAN
JlnSabuga
JlCisituLama
BormaDago
MonumenPerjuangan
JlPasupatiGasibu
Masukkan Simpul Awal : SimpangBorromeus
Masukkan Simpul Akhir : SimpangPelesiran
Jarak Simpul :  987.4450001400589
```

5. arad.txt




```
Arad
Bucharest
Craiova
Dobreta
Eforie
Fagaras
Giurgiu
Harsova
Lasi
Lugoj
Mehadia
Neamt
Oradea
Pitesti
RimnicuVilcea
Sibiu
Timisoara
Urziceni
Vaslui
Zerind
Masukkan Simpul Awal : Arad
Masukkan Simpul Akhir : Bucharest
Jarak Simpul :  545800.6033765689
```

6. antapani.txt




```
LuckySquare
Borma
Griya
PondokSurabi
SuperIndo
SMA23
Arcamanik
Cicaheum
Masukkan Simpul Awal : Cicaheum
Masukkan Simpul Akhir : Borma
Jarak Simpul :  1542.2379973955042
```

LAMPIRAN

Program dapat diunduh di :
https://github.com/farhanfadillahr/tucil3

NO	Spesifikasi	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 maps API	-