

LAPORAN TUGAS KECI 3

IF2211 STRATEGI ALGORITMA

IMPLEMENTASI ALGORITMA A\* UNTUK MENENTUKAN LINTASAN TERPENDEK



OLEH

13519183 Afifah Fathimah Qur'ani

13519208 Awwala Nisa Kamila

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

## 1. Penjelasan Singkat

Algoritma A\* (atau A star) dapat digunakan untuk menentukan lintasan terpendek dari suatu titik ke titik lain. Pada tugas kecil 3 ini, anda diminta menentukan lintasan terpendek berdasarkan peta Google Map jalan-jalan di kota Bandung. Dari ruas-ruas jalan di peta dibentuk graf. Simpul menyatakan persilangan jalan atau ujung jalan. Asumsikan jalan dapat dilalui dari dua arah. Bobot graf menyatakan jarak (m atau km) antar simpul. Jarak antar dua simpul dapat dihitung dari koordinat kedua simpul menggunakan rumus jarak Euclidean (berdasarkan koordinat) atau dapat menggunakan ruler di Google Map, atau cara lainnya yang disediakan oleh Google Map.

Langkah pertama di dalam program ini adalah membuat graf yang merepresentasikan peta (di area tertentu, misalnya di sekitar kampus ITB). Sisi diperoleh dari jalan antara dua simpul dan bobot sisi adalah jarak Euclidean. Berdasarkan graf yang dibentuk, lalu program A\* menerima input simpul asal dan simpul tujuan, lalu menentukan lintasan terpendek antara keduanya. Lintasan terpendek dapat ditampilkan pada peta/graf. Nilai heuristik yang dipakai adalah jarak garis lurus dari suatu titik ke tujuan.

Spesifikasi program:

1. Program menerima input file graf (direpresentasikan sebagai matriks ketetanggaan berbobot), jumlah simpul minimal 8 buah.
2. Program dapat menampilkan peta/graf
3. Program menerima input simpul asal dan simpul tujuan.
4. Program dapat menampilkan lintasan terpendek beserta jaraknya antara simpul asal dan simpul tujuan.
5. Antarmuka program bebas, apakah pakai GUI atau command line saja.

## 2. Kode Program

Nama Program : printGraf.ipynb

```
import math
from math import radians, cos, sin, asin, sqrt
import networkx as nx
import matplotlib.pyplot as plt

def readfile(namafilegraf):
    global graf, grafsimpul, banyaksimpul, bobot
    #file = open("./test/" + namafile + ".txt", "r")
    file = open('C:\\Users\\Asus\\Documents\\Tubes 3 Stima\\test\\' + namafilegraf + '.txt', 'r')
```

```

banyaksimpul = int(file.readline())
print("Banyak Simpul : ", banyaksimpul)
print()
grafsimpul = file.readline()
graf = file.readlines()
print("grafsimpul:", grafsimpul)
print()
print("graf:", graf)
print()

def readposisi(namafileposisi):
    global posisi, hasilposisi
    file = open('C:\\Users\\Asus\\Documents\\Tubes 3 Stima\\test\\' + namafileposisi + '.txt', 'r')
    posisi = file.readlines()

    b = ""
    isi = []
    hasilposisi = []
    posisipersimpul = []
    for i in range(banyaksimpul):
        for j in range(len(posisi[i])):
            if (posisi[i][j] == ','):
                posisipersimpul += isi
                b = ""
                #print("posisipersimpul : ", posisipersimpul)
            elif (posisi[i][j] == '\n'):
                hasilposisi += [posisipersimpul]
                posisipersimpul = []
            elif (posisi[i][j] != ','):
                b += posisi[i][j]
                isi = [b]
                #print("ISI : ", isi)
    print("posisi:", posisi)
    print("hasilposisi:", hasilposisi)

def ceksimpul():
    global simpul
    simpul = []
    huruf = ""
    isiSimpul = ""
    simpul = grafsimpul.split(',')
    lastnode = simpul[-1][-1]
    simpul = simpul[:-1]
    simpul.append(lastnode)
    print("simpul:", simpul)
    #splitsimpul()

def splitsimpul():
    newsimpul = [char for char in simpul[0]]
    simpul.clear()
    simpul.extend(newsimpul)

def isimatriks():
    global hasil
    b = ""
    isi = []
    hasil = []
    bobotpersimpul = []
    for i in range(banyaksimpul):

```

```

    for j in range(len(graf[i])):
        if (graf[i][j] == ','):
            bobotpersimpul += isi
            b = ""
            #print("BOBOTPERSIMPUL : ", bobotpersimpul)
        elif (graf[i][j] == '\n'):
            hasil += [bobotpersimpul]
            bobotpersimpul = []
        elif (graf[i][j] != ','):
            b += graf[i][j]
            isi = [b]
            #print("ISI : ", isi)

def grafindict():
    global grafberbobot
    grafberbobot = []
    akhir = []
    for i in range(banyaksimpul):
        akhir += [(simpul[i], hasil[i])]
        #akhir += [(simpul[i], i)]

    grafberbobot = dict(akhir)

def grafberbobotberpasangan():
    global g
    g = []
    for i in range(banyaksimpul):
        for j in range(0+i, banyaksimpul):
            if (hasil[i][j] != '-1' and hasil[j][i] != '-1'):
                if (hasil[i][j] == hasil[j][i]):
                    g += [(simpul[i], simpul[j], int(hasil[i][j]))]
            else:
                continue
        else:
            continue

def haversine(lat1, lon1, lattujuan, lontujuan):
    R = 6372.8 * 1000
    dLat = radians(lattujuan - lat1)
    dLon = radians(lontujuan - lon1)
    lat1 = radians(lat1)
    lattujuan = radians(lattujuan)

    a = sin(dLat/2)**2 + cos(lat1)*cos(lattujuan)*sin(dLon/2)**2
    c = 2*asin(sqrt(a))

    return R * c

def jarak(simpultujuan):
    global heuristic, posisitujuan
    heuristic = [] #heuristic hitung ke node goal
    akhir = []

    posisitujuan = 0
    for i in range(banyaksimpul):
        if (simpultujuan == simpul[i]):
            posisitujuan = i
            break

```

```

    for i in range(banyaksimpul):
        dist = haversine(int(hasilposisi[i][0]), int(hasilposisi[i][1]), int(hasilposisi[posisitujuan][0]),
int(hasilposisi[posisitujuan][1]))
        akhir += [(simpul[i], dist)]

    heuristic = dict(akhir)

class Graph:
    def __init__(self, graph_dict=None, directed=True):
        self.graph_dict = graph_dict or {}
        self.directed = directed
        if not directed:
            self.make_undirected()
    def make_undirected(self):
        for a in list(self.graph_dict.keys()):
            for (b, dist) in self.graph_dict[a].items():
                self.graph_dict.setdefault(b, {})[a] = dist
    def connect(self, A, B, distance=1):
        self.graph_dict.setdefault(A, {})[B] = distance
        if not self.directed:
            self.graph_dict.setdefault(B, {})[A] = distance
    def get(self, a, b=None):
        links = self.graph_dict.setdefault(a, {})
        if b is None:
            return links
        else:
            return links.get(b)
    def nodes(self):
        s1 = set([k for k in self.graph_dict.keys()])
        s2 = set([k2 for v in self.graph_dict.values() for k2, v2 in v.items()])
        nodes = s1.union(s2)
        return list(nodes)

class Node:
    def __init__(self, name:str, parent:str):
        self.name = name
        self.parent = parent
        self.g = 0 # jarak ke node start
        self.h = 0 # jarak heuristic ke node tujuan
        self.f = 0 # Total jarak
    # Compare nodes
    def __eq__(self, other):
        return self.name == other.name
    # Sort nodes
    def __lt__(self, other):
        return self.f < other.f
    # Print node
    def __repr__(self):
        return '({0},{1})'.format(self.name, self.f)

def add_to_open(open, neighbor):
    for node in open:
        if (neighbor == node and neighbor.f > node.f):
            return False
    return True

def astar_search(graph, heuristics, start, end):
    open = []
    closed = []

```

```

start_node = Node(simpulasal, None)
goal_node = Node(simpultujuan, None)
open.append(start_node)
#print("start:", start_node)

while len(open) > 0:
    # Sort secara bobot ascending
    open.sort()
    current_node = open.pop(0)
    closed.append(current_node)
    #print("current node:", current_node)

    if current_node == goal_node:
        pathweighted = []
        while current_node != start_node:
            pathweighted.append(current_node.name + ':' + str(current_node.g))
            current_node = current_node.parent
        pathweighted.append(start_node.name + ':' + str(start_node.g))
        # Return path
        return pathweighted[::-1]
    neighbors = graf.get(current_node.name)
    #print("neighbors :", neighbors)
    for key, value in neighbors.items():
        neighbor = Node(key, current_node)
        if(neighbor in closed):
            continue
        # hitung total bobot
        neighbor.g = current_node.g + graph.get(current_node.name, neighbor.name)
        neighbor.h = heuristics.get(neighbor.name)
        neighbor.f = neighbor.g + neighbor.h
        if(add_to_open(open, neighbor) == True):
            open.append(neighbor)
    return None

#Algoritma Utama
namafilegraf = input("Masukkan file graf : ")
readfile(namafilegraf)

namafileposisi= input("Masukkan file posisi : ")
readposisi(namafileposisi)
print("hasilposisi:", hasilposisi)
print()

ceksimpul()
print("simpul:", simpul)
print()

isimatriks()
print("hasil:", hasil)
print()

grafindict()
print("grafberbobot:", grafberbobot)
print()

grafberbobotherpasangan()
print("g:", g)
print()

```

```

simpulasal = input("Masukkan simpul asal : ")
simpultujuan = input("Masukkan simpul tujuan : ")
jarak(simpultujuan)
#print("heuristic:", heuristic)

graf = Graph()
for edge in g :
    graf.connect(edge[0], edge[1], edge[2])

pathweighted = astar_search(graf, heuristic, simpulasal, simpultujuan)
print("solusi:", pathweighted)
print()

path = []
for solution in pathweighted:
    currpath = solution.partition(":")
    #print(currpath)
    path.append(currpath[0])
print(path)
print()

'''VISUALISASIGRAF'''

G = nx.Graph()
for vertex in simpul:
    G.add_node(vertex)

for v1,v2,w in g:
    G.add_edge(v1,v2, weight=w)

print("Ini Simpul Awal : ")
pos = nx.spring_layout(G)
bobot = nx.get_edge_attributes(G, 'weight')
nx.draw_networkx_labels(G,pos)
nx.draw_networkx_edge_labels(G,pos,edge_labels=bobot)
nx.draw_networkx_edges(G,pos)

color_map = []
print(vertex)

for vertex in simpul:
    color_map.append('white')

nx.draw(G, pos, node_color=color_map)

nx.draw_networkx_nodes(G, pos, nodelist=path, node_color="yellow")

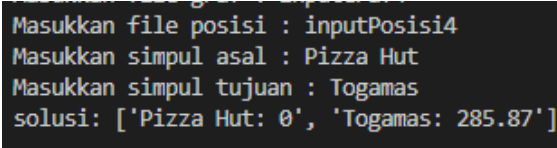

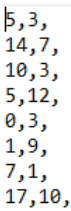
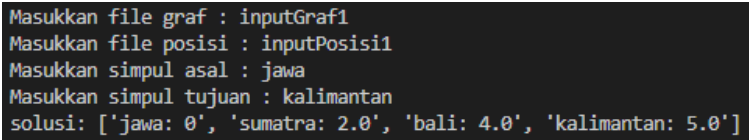
print("Graf adalah rute dari ", simpulasal, " ke ", simpultujuan)
print("Yang berwarna kuning adalah rute terpendek dari ",
simpulasal, " ke ", simpultujuan)

```

### 3. Peta / Graf Input

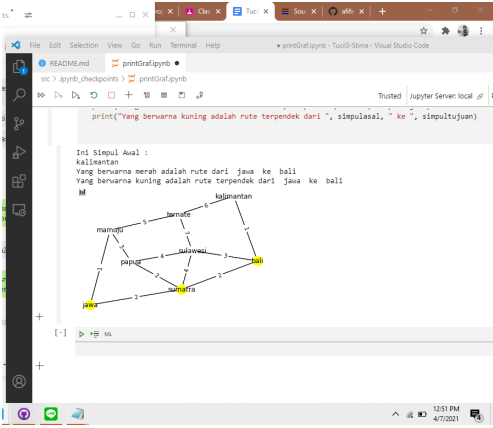
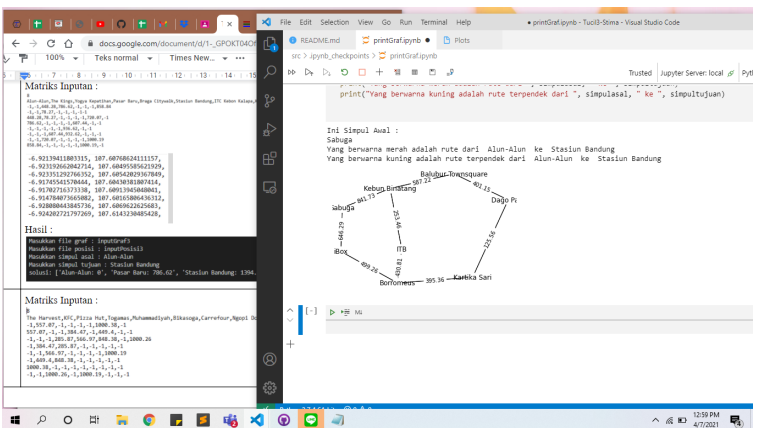
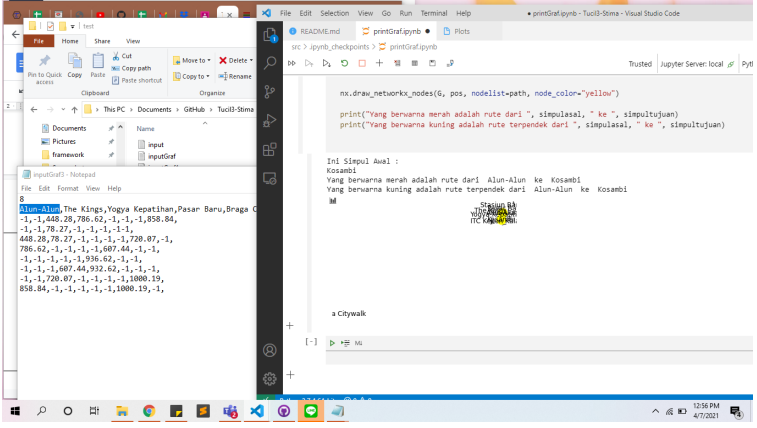
No	Kasus Uji	Screenshoot hasil
1	Peta jalan sekitar kampus ITB/Dago	<p>Matriks Inputan :</p> <pre> 8 ITB,Borromeus,Kartika Sari,iBox,Kebun Binatang,Dago Park,Balubur Townsquare,Sabuga -1,430.81,-1,-1,253.46,-1,-1,-1 430.81,-1,395.36,499.26,-1,-1,-1,-1 -1,395.36,-1,-1,-1,125.56,-1,-1 -1,499.26,-1,-1,-1,-1,-1,646.29 253.46,-1,-1,-1,-1,-1,587.22,841.73 -1,-1,125.56,-1,-1,-1,401.15,-1 -1,-1,-1,-1,587.22,401.15,-1,-1 -1,-1,-1,646.29,841.73,-1,-1,-1] </pre> <pre> -6.893196882055465, 107.61043696393973, -6.893761751356862, 107.61302393750051, -6.897482340211029, 107.61272806663133, -6.889540984237854, 107.61336125199308, -6.893696655106442, 107.60841405139047, -6.898687190494478, 107.61257411240334, -6.898751043225343, 107.60933959291256, -6.88766977245812, 107.6099282200383, </pre> <p>Hasil :</p> <pre> Masukkan file graf : inputGraf2 Masukkan file posisi : inputPosisi2 Masukkan simpul asal : ITB Masukkan simpul tujuan : Sabuga solusi: ['ITB: 0', 'Kebun Binatang: 253.46', 'Sabuga: 1095.19'] </pre>
2	Peta jalan sekitar Alun-alun Bandung	<p>Matriks Inputan :</p> <pre> 8 Alun-Alun,The Kings,Yogya Kepatihan,Pasar Baru,Braga Citywalk,Stasiun Bandung,ITC Kebon Kalapa,Kosambi -1,-1,448.28,786.62,-1,-1,-1,858.84 -1,-1,78.27,-1,-1,-1,-1,-1 448.28,78.27,-1,-1,-1,-1,720.07,-1 786.62,-1,-1,-1,-1,607.44,-1,-1 -1,-1,-1,-1,936.62,-1,-1 -1,-1,-1,607.44,932.62,-1,-1,-1 -1,-1,720.07,-1,-1,-1,-1,1000.19 858.84,-1,-1,-1,-1,-1,-1,1000.19,-1 </pre> <pre> -6.92139411803315, 107.60768624111157, -6.923192662042714, 107.60495585621929, -6.923351292766352, 107.60542029367849, -6.91745541570444, 107.60430381807414, -6.91702716373338, 107.60913945048041, -6.914784073665082, 107.60165806436312, -6.928080443845736, 107.6069622625683, -6.924202721797269, 107.6143230485428, </pre> <p>Hasil :</p> <pre> Masukkan file graf : inputGraf3 Masukkan file posisi : inputPosisi3 Masukkan simpul asal : Alun-Alun Masukkan simpul tujuan : Stasiun Bandung solusi: ['Alun-Alun: 0', 'Pasar Baru: 786.62', 'Stasiun Bandung: 1394.06'] </pre>
3	Peta jalan sekitar Buahbatu	<p>Matriks Inputan :</p> <pre> 8 The Harvest,KFC,Pizza Hut,Togamas,Muhammadiyah,Bikasoga,Carrefour,Ngopi Doeloe -1,557.07,-1,-1,-1,-1,1000.38,-1 557.07,-1,-1,384.47,-1,449.4,-1,-1 -1,-1,-1,285.87,566.97,848.38,-1,1000.26 -1,384.47,285.87,-1,-1,-1,-1,-1 -1,-1,566.97,-1,-1,-1,-1,1000.19 -1,449.4,848.38,-1,-1,-1,-1,-1 1000.38,-1,-1,-1,-1,-1,-1,-1,-1 -1,-1,1000.26,-1,1000.19,-1,-1,-1,-1 </pre>

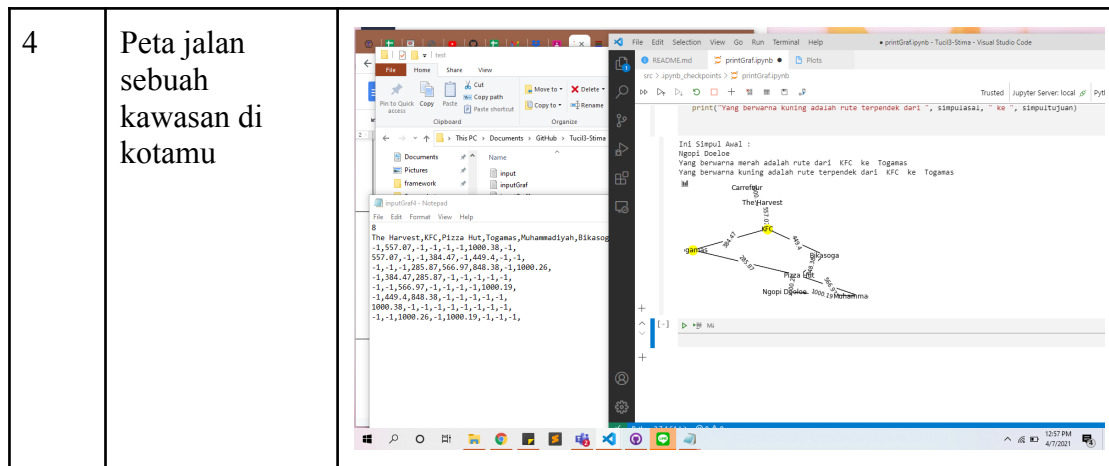


		[-6.9447739121134475, 107.63012338833475, -6.9408513219725325, 107.62718226857638, -6.936844909840429, 107.62316302519258, -6.93895847364346, 107.62433586437992, -6.933173018020938, 107.6230006320837, -6.9432077885478325, 107.62455215858365, -6.9264670325238695, 107.62744249763288, -6.946084274357763, 107.64096791255656,  Hasil :  <pre> Masukkan file posisi : inputPosisi4 Masukkan simpul asal : Pizza Hut Masukkan simpul tujuan : Togamas solusi: ['Pizza Hut: 0', 'Togamas: 285.87'] </pre>
4	Peta jalan sebuah kawasan di kotamu	Inputan Matriks :  <pre> jawa,sumatra,bali,papua,sulawesi,mamuju,ternate,kalimantan -1,2,-1,-1,-1,1,-1,-1, 2,-1,2,2,4,-1,-1,-1, -1,2,-1,-1,3,-1,-1,1, -1,2,-1,-1,4,3,-1,-1, -1,4,3,4,-1,-1,7,-1, 1,-1,-1,3,-1,-1,5,-1, -1,-1,-1,-1,7,5,-1,6, -1,-1,1,-1,-1,-1,6,-1, </pre>  <pre> 5,3, 14,7, 10,3, 5,12, 0,3, 1,9, 7,1, 17,10, </pre> Hasil :  <pre> Masukkan file graf : inputGraf1 Masukkan file posisi : inputPosisi1 Masukkan simpul asal : jawa Masukkan simpul tujuan : kalimantan solusi: ['jawa: 0', 'sumatra: 2.0', 'bali: 4.0', 'kalimantan: 5.0'] </pre>

#### 4. Screenshot Hasil Program

No	Kasus Uji	Screenshoot hasil
----	-----------	-------------------

1	Peta jalan sekitar kampus ITB/Dago	 <pre> print("Yang berwarna kuning adalah rute terpendek dari ", simpulasal, " ke ", simpultujuan)  Ini Simpul Awal : Kajanten Yang berwarna merah adalah rute dari Jawa ke Bali Yang berwarna kuning adalah rute terpendek dari Jawa ke Bali M </pre>
2	Peta jalan sekitar Alun-alun Bandung	 <pre> print("Yang berwarna kuning adalah rute terpendek dari ", simpulasal, " ke ", simpultujuan)  Ini Simpul Awal : Sabuga Yang berwarna merah adalah rute dari Alun-Alun ke Stasiun Bandung Yang berwarna kuning adalah rute terpendek dari Alun-Alun ke Stasiun Bandung </pre>
3	Peta jalan sekitar Buahbatu	 <pre> mx.draw_networkx_nodes(G, pos, nodelist=path, node_color="yellow")  print("Yang berwarna merah adalah rute dari ", simpulasal, " ke ", simpultujuan) print("Yang berwarna kuning adalah rute terpendek dari ", simpulasal, " ke ", simpultujuan)  Ini Simpul Awal : Kosambi Yang berwarna merah adalah rute dari Alun-Alun ke Kosambi Yang berwarna kuning adalah rute terpendek dari Alun-Alun ke Kosambi M </pre>



## 5. Link Menuju Kode Program

private repo on github : <https://github.com/afifahfq/Tucil3-Stima>

## 6. Tabel Penilaian

No	Keterangan	Dikerjakan
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