LAPORAN TUGAS KECIL 3 IF2211 STRATEGI ALGORITMA

Pemanfaatan Algoritma *A* Untuk Menentukan Lintasan Terpendek*

Semester 2 Tahun 2020/2021

Disusun oleh :

Farrell Abieza Zidan (13519182)

Allief Nuriman (13519221)

PROGRAM STUDI SARJANA TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

DAFTAR ISI

BAB 1
LANDASAN TEORI

1.1 Algoritma *A**

1.1.1. Pengertian Metode atau Algoritma *A**

Metode/Algoritma A* merupakan algoritma populer yang biasa dipresentasikan berbentuk Pohon Ruang Status. Algoritma A* merupakan algoritma yang bersifat heuristik dikarenakan pencarian solusinya menggunakan informasi tambahan yang secara intuitif ditentukan oleh manusia / heuristik pula. Solusi Algoritma A* adalah optimal, namun pendekatan Algoritma A* harus memiliki fungsi heuristik yang dapat diterima dan masih ada kemungkinan tidak memberikan solusi pada beberapa kasus, seperti pencarian jalur terpendek yang memiliki node tidak terbatas. Algoritma A* pada dasarnya memiliki goal mencari suatu solusi dengan sum-cost yang minimum.

1.1.2. Prinsip dan Konsep Algoritma *A**

Prinsip utama algoritma A* adalah *"avoid expanding paths that are already expensive !"*. Maksud dari prinsip tersebut adalah sebagai berikut: Pada setiap langkah dalam algoritma A*, Kita perlu bangkitkan suatu node yang memiliki sum-cost yang paling minimum terdahulu, karena Algoritma A* menganggap solusi akan mengarah kepada node node yang memiliki sum-cost terkecil sehingga perlu dibangkitkan. Jika sudah dibangkitkan, maka penentuan node selanjutnya yang akan dibangkitkan akan terus dilakukan dengan mengecek semua node yang belum dibangkitkan dengan cara yang sama sampai solusi ditemukan.

1.1.3. Skema Umum Algoritma *A**

Algoritma A* disusun oleh elemen, dan elemen-elemen yang digunakan dalam penerapan algoritma A* antara lain :

1. f(n) / Estimasi biaya atau cost dari path awal menuju goal melalui n merupakan elemen yang menjadi penentu sum-cost. Dengan begitu, node yang akan dibangkitkan adalah node dengan f(n) terkecil dari node lainnya.
2. g(n) / Cost untuk menuju node n merupakan elemen yang menentukan cost terhadap node n yang terhubung dengan node saat ini.

3. h(n) / Cost untuk menuju node goal dari node n merupakan elemen yang menentukan cost terhadap node goal yang tidak mesti terhubung dengan node goal. Sifat heuristik dalam algoritma ini paling terlihat ketika menentukan h(n).

BAB 2
SOURCE CODE FILE

main.py

```
import visgraf as gv
import graf as g
import simpul as n
import math

# A* search
def astar(graph, heuristics, start, end, data_node, startNode, endNode, jmlNode):

    # Bikin variabel buat visualisasi
    G = gv.GraphVisual()

    # Bikin list untuk node terbuka dan tertutup
    open = []
    closed = []

    # Membuat node awal & node akhir
    x = float(data_node[startNode][1])
    y = float(data_node[startNode][2])
    start_node = n.Node(start, None, x, y)
    goal_node = n.Node(end, None, x, y) # Parent dari goal ialah None karena belum
diketahui

    # Tambahkan node pertama ke node terbuka
    open.append(start_node)

    # Looping sampai list terbuka kosong
    while len(open) > 0:
        # current_node ialah node yang mempunyai f terendah,
        # sehingga urutkan terlebih dahulu baru ambil kemudian
        open.sort()
        current_node = open.pop(0)
        closed.append(current_node)

        # Periksa apakah goal telah dicapai, jika ya kembalikan pathnya secara
reversal & proses selesai
        if current_node == goal_node:
            path = []
            while current_node != start_node:
                path.append(current_node.name + ' ' + str(current_node.g))
                # Bentuk graf dari current_node dan "orang tua" dari current_node
                G.addEdge(current_node.name,current_node.parent.name)
                current_node = current_node.parent
            path.append(start_node.name + ' ' + str(start_node.g))
```

```
            print("\nIni adalah jalan dari ",end='')
            print(node[startNode],end=' ')
            print("ke ",end='')
            print(node[endNode])
            # Kembalikan path secara terbalik (Karena kita menelusuri dari
goal_node)
            path = path[::-1]
            for eachline in path:
                print(eachline)
            G.visualize()
            return path

        # Dapatkan tetangga
        neighbors = graph.get(current_node.name)

        # Untuk setiap tetangganya current...
        for key, value in neighbors.items():
            i = 0
            ketemuIndex = False
            while (not ketemuIndex and i < jmlNode):
                if (key == data_node[i][0]):
                    ketemuIndex = True
                    print("OK")
                else:
                    i += 1
            x = float(data_node[i][1])
            y = float(data_node[i][2])
            neighbor = n.Node(key, current_node, x, y)

            # Cek jika tetangga ada pada closed list
            if(neighbor in closed):
                continue

            # Hitung harga pathnya
            neighbor.g = current_node.g + graph.get(current_node.name,
neighbor.name)
            neighbor.h = heuristics.get(neighbor.name)
            neighbor.f = neighbor.g + neighbor.h

            # Cek jika tetangga ada di list terbuka dan punya nilai f (g + h) yang
lebih tinggi
            if(add_to_open(open, neighbor)):

                # Tambahkan tetangga ke list terbuka
                open.append(neighbor)

    # Kembalikan none, tidak ada path yang ditemukan
    return None
```

```
# Fungsi untuk menambah tetangga ke list terbuka
# Pada dasarnya ini untuk mengecek apakah nilai f lebih rendah
def add_to_open(open, neighbor):
    # cek bila neighbor sama dengan open & nilai f neighbor lebih besar
    # daripada f nya node
    for node in open:
        if (neighbor == node and neighbor.f > node.f):
            return False
    return True

# Rumus menentukan jarak dua titik
def Jarak2Node(x1,x2,y1,y2):
    return (math.sqrt(math.pow(x2-x1, 2) + math.pow(y2-y1, 2)))

# PROGRAM UTAMA

print("Halo, silahkan pilih nomor file yang ingin Anda periksa!")
print("Pilihan:")
print("1. tc1.txt (Kampus ITB)")
print("2. tc2.txt (Alun-alun Bandung)")
print("3. tc3.txt (Kecamatan Buahbatu, Bandung)")
print("4. tc4.txt (Kecamatan Tanjungpandan, Belitung)")
print("5. tc5.txt (Blitar)")
print("6. tc6.txt (Monas)")
print("")
print("> ",end='')

namafile = input()

namafile = "../test/tc" + namafile + ".txt"
f = open(namafile,"r")

# Baca file dan masukkan ke variabel
txtlines = f.readlines()
f.close()

# Cari dan masukkan seluruh node pada peta ke variabel node
# data_node berisi nama, koordinat x, dan koordinat y dari nama
# node berisi nama node

jmlNode = int(txtlines[0])

data_node_temp = []
data_node = []
node = []
i = 1
```

```
for i in range(1,jmlNode+1):
    data_node_temp = txtlines[i].split()
    data_node.append(data_node_temp)
    node.append(data_node[i-1][0])

# Buat weighted graph dalam representasi matrix

graf = []
graf_elemen = []
i = jmlNode + 1
for i in range(jmlNode+1,jmlNode+1+jmlNode):
    j = 0
    for eachchar in txtlines[i]:
        if eachchar != ' ':
            graf_elemen.append(0)
            if eachchar == '1':
                graf_elemen[j] = 1
            j += 1
    graf.append(graf_elemen)
    graf_elemen = []

# Telah dibentuk suatu matriks ketetanggaan
# Matriks ketetanggaan akan dikonversi ke graf

graph = g.Graph()
jarak = 0

print("\n\nList lokasi:")
for i in range(jmlNode):
    print(str(i+1) + ". " + node[i])

print("\n\nSilahkan pilih nomor lokasi start dari lokasi yang tersedia dalam
list!")
print("> ",end='')
startNode = int(input()) - 1
print("\n\nSilahkan pilih nomor lokasi tujuan dari lokasi yang tersedia dalam
list!")
print("> ",end='')
endNode = int(input()) - 1

for i in range(jmlNode):
    for j in range(jmlNode):
        if (graf[i][j] == 1):
            jarak =
Jarak2Node(float(data_node[i][1]),float(data_node[j][1]),float(data_node[i][2]),flo
at(data_node[j][2])) * 100000
            graph.connect(node[i],node[j],jarak)
```

```
heuristics = {}
for i in range(jmlNode):
    heuristics[node[i]] =
Jarak2Node(float(data_node[i][1]),float(data_node[endNode][1]),float(data_node[i][2
]),float(data_node[endNode][2]))

astar(graph, heuristics, node[startNode], node[endNode], data_node, startNode,
endNode, jmlNode)
```

graf.py

```
# Kelas graf
class Graph:
    # Konstruktor kelas
    def __init__(self, graph_dict=None):
        self.graph_dict = graph_dict or {}

    # Menghubungkan A dan B dengan suatu jarak
    def connect(self, A, B, distance=1):
        self.graph_dict.setdefault(A, {})[B] = distance

    # Mendapatkan tetangga dari suatu node
    def get(self, a, b=None):
        tetangga = self.graph_dict.setdefault(a, {})
        if b is None:
            return tetangga
        else:
            return tetangga.get(b)
```

simpul.py

```
# Kelas Node
class Node:
    # Konstruktor kelas
    def __init__(self, name:str, parent:str, x:float, y:float):
        self.x = x
        self.y = y
        self.name = name
        self.parent = parent
        self.g = 0 # Distance to start node
        self.h = 0 # Distance to goal node
        self.f = 0 # Total cost

    # Pengecekan node yang sama
    def __eq__(self, other):
        return self.name == other.name

    # Sorting node
    def __lt__(self, other):
         return self.f < other.f
```

```
    # Cetak node ke layar
    def __repr__(self):
        return ('({0},{1})'.format(self.name, self.f))
```

visgraf.py

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

# Kelas GraphVisual
class GraphVisual:

    def __init__(self):

        # self adalah list yang menyimpan semua
        # himpunan edge yang membentuk graf
        self.visual = []

    # meminta node asal dan node tujuan
    # serta memasukkan ke self
    def addEdge(self, a, b):
        edge = [a, b]
        self.visual.append(edge)

    # Objek dari kelas Graph dari modul networkx
    # nx.draw_networkx(G) - membentuk graf
    # plt.show() menampilkan graf
    def visualize(self):
        G = nx.Graph()
        G.add_edges_from(self.visual)
        nx.draw_networkx(G)
        plt.show()
```

BAB 3
DATA PENGUJIAN DAN HASIL

3.1 Data Pengujian

tc1.txt

```
8
Segitiga_Dayang_Sumbi -6.887221 107.611479
Batan -6.887373 107.611504
Bonbin -6.893890 107.608442
Kubus -6.893188 107.610418
Borromeus -6.893731 107.612864
CircleK -6.886535 107.611607
Pos_Polisi -6.885217 107.613692
Baksil -6.884972 107.611512
0 1 0 0 0 1 1 1
1 0 1 0 0 0 0 0
0 1 0 1 0 0 0 0
0 0 1 0 1 0 0 0
0 0 0 1 0 1 0 0
1 0 0 0 1 0 1 0
1 0 0 0 0 1 0 1
1 0 0 0 0 0 1 0
```

tc2.txt

```
10
Jend_Sudirman_Navaro -6.920817 107.604100
Kontiki_Lintas_Media -6.920893 107.605089
Toko_Komputer_New_Pelangi -6.921052 107.606472
Masjid_Agung -6.921228 107.607690
Garuda_Kencana_Toko -6.922083 107.604024
Dalem_Kaum -6.922383 107.606429
Pendopo_Bandung -6.922521 107.607099
Dalem_Kaum 63 -6.922551 107.607520
Otista_Kepatihan -6.923107 107.603929
Balonggede -6.923423 107.606288
0 1 0 0 1 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 1 0 0
1 0 0 0 0 1 0 0 1 0
0 0 0 0 1 0 1 0 0 1
0 0 0 0 0 1 0 1 0 0
0 0 0 1 0 0 1 0 0 0
0 0 0 0 1 0 0 0 0 1
0 0 0 0 0 1 0 0 1 0
```

tc3.txt

```
8
Scoop_&_Skoops -6.959889011816301 107.66178035853191
```

```
Es_Kelapa_Abah_Anom -6.95957215002226 107.6618439377869
Bearista_Coffee -6.959428534831149 107.66130217452516
TV_Buy_Sell_Used -6.959658170829064 107.66112648983253
Dbestro_Ciwastra -6.959950241012554 107.66205356602796
Batagor_&_Mie_Baso -6.959786668967319 107.66149496878126
Semesta_Clothing -6.9600610416201265 107.66137597430478
Jasti_Hurip -6.960674126738876 107.66186701220137
0 1 0 0 1 1 0 0
1 0 1 0 0 0 0 0
0 1 0 1 0 0 0 0
0 0 1 0 0 0 0 0
1 0 0 0 0 0 0 1
1 0 0 0 0 0 1 0
0 0 0 0 0 1 0 0
0 0 0 0 1 0 0 0
```

tc4.txt

```
8
Rumahku -2.73075273287073 107.62838046756838
Kantor_Lurah_Tanjungpendam -2.7307461739895023 107.62927513061996
Masjid_Ash_Syafa'ah -2.72340678973154 107.63000611034087
Puskesmas_Air_Saga -2.722845992798171 107.6303624294781
SMAN_1_Tanjungpandan -2.7240523093301263 107.63504099932938
Langgar_Istiqomah -2.731264607245666 107.62925382727556
Raja_Ponsel -2.72386461414985 107.6355680946197
MAN_1_Belitung -2.724349955866123 107.63033465051568
0 1 0 0 0 0 0 0
1 0 0 0 0 1 0 1
0 0 0 1 1 0 0 1
0 0 1 0 0 0 0 0
0 0 1 0 0 0 1 0
0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 1 1 0 0 0 0 0
```

tc5.txt

```
11
A -8.099045120315456 112.16493522999168
B -8.100840204992812 112.16458117838853
C -8.100340981063543 112.16244613993318
D -8.098853927392977 112.16268217433529
E -8.099013254834688 112.16429149980415
F -8.099066363967909 112.16586863876363
G -8.099140716742642 112.16737067586789
H -8.097600549319546 112.16749942190542
I -8.097419977580683 112.16605102898342
J -8.097165052635042 112.16450607653333
K -8.096888883761814 112.1628645645551
0 1 0 0 1 1 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0 0
0 0 1 0 1 0 0 0 0 0 0
1 0 0 1 0 0 0 0 0 1 0
```

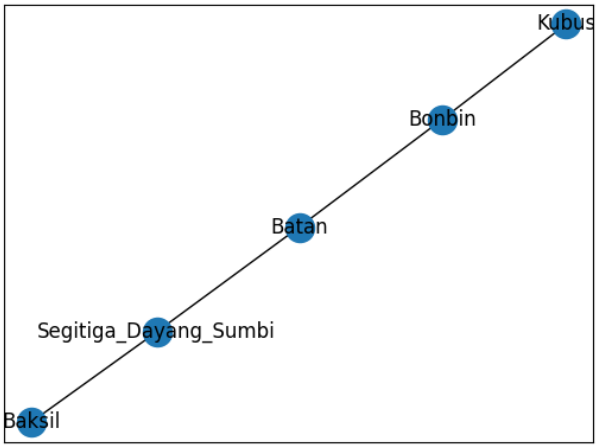
```
1 0 0 0 0 0 1 0 1 0 0
0 0 0 0 0 1 0 1 0 0 0
0 0 0 0 0 0 1 0 1 0 0
0 0 0 0 0 1 0 1 0 1 0
0 0 0 0 0 0 0 0 1 0 1
0 0 0 1 0 0 0 0 0 1 0
```

tc6.txt

```
21
A -6.18186 106.81367
B -6.18175 106.81664
C -6.18493 106.81487
D -6.18345 106.82294
E -6.1826 106.83329
F -6.17873 106.84188
G -6.16388 106.83833
H -6.15928 106.83724
I -6.16252 106.82877
J -6.16002 106.81899
K -6.16755 106.82075
L -6.16791 106.83105
M -6.17104 106.82189
N -6.1764 106.83108
O -6.17228 106.83408
P -6.17282 106.83008
Q -6.17096 106.8334
R -6.19925 106.85389
S -6.18684 106.83612
T -6.18691 106.82306
U -6.19491 106.82302
0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0
0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 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 0 0
0 0 0 0 0 1 0 1 0 0 0 1 0 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 0 0 0 0 0 1 0 1 0 1 0 0 0 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 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 1 1 0 0 0 0
0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 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 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 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 0 0 0 0 0 1 0
```

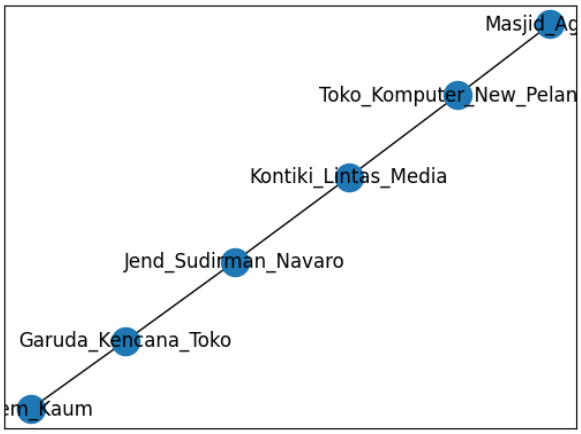
3.2 Data Hasil

tc1.txt



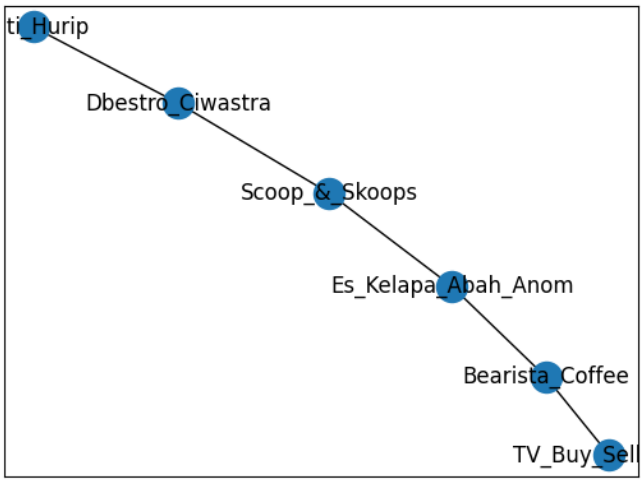

```
> tc1.txt
Ini adalah jalan dari Baksil ke Kubus
['Baksil:    0',    'Segitiga_Dayang_Sumbi:    224.9242094573166',    'Batan:
240.32842965817875',        'Bonbin:        960.3779626765328',        'Kubus:
1170.0772712112953']
```

tc2.txt



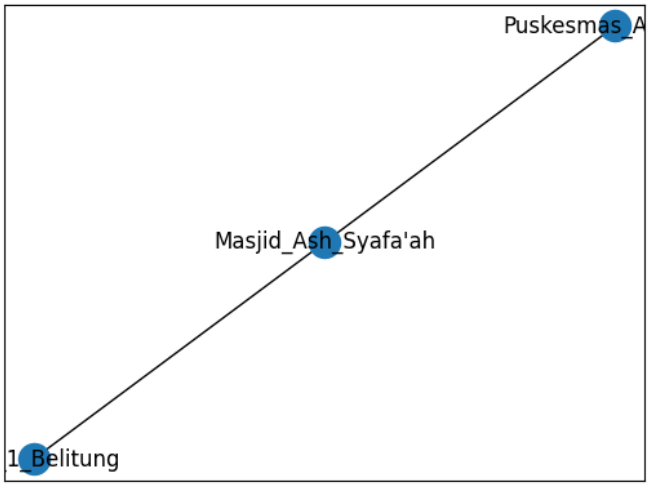

```
> tc2.txt
Ini adalah jalan dari Dalem_Kaum ke Masjid_Agung
['Dalem_Kaum        0',        'Garuda_Kencana_Toko        242.36387932299644',
'Jend_Sudirman_Navaro        369.19179423132255',        'Kontiki_Lintas_Media
468.38337653587877',        'Toko_Komputer_New_Pelangi        607.5943674802768',
'Masjid_Agung 730.6593909443325']
```

tc3.txt



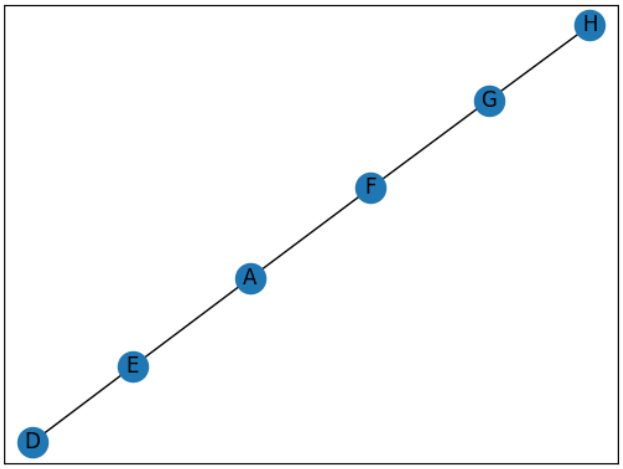

```
Ini adalah jalan dari Jasti_Hurip ke TV_Buy_Sell_Used
['Jasti_Hurip   0',   'Dbestro_Ciwastra   74.75378752881193',   'Scoop_&_Skoops
102.75224290977121',         'Es_Kelapa_Abah_Anom         135.06999626414088',
'Bearista_Coffee 191.1175434794579', 'TV_Buy_Sell_Used 220.0308281017153']
```

tc4.txt



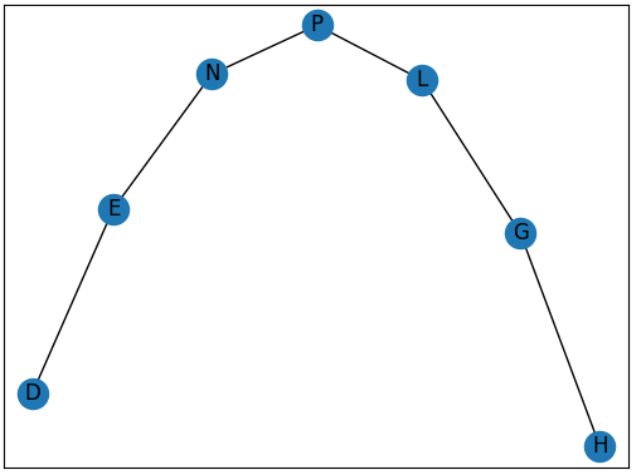

```
> tc4.txt
Ini adalah jalan dari MAN_1_Belitung ke Puskesmas_Air_Saga
['MAN_1_Belitung      0',       "Masjid_Ash_Syafa'ah      99.87497203431495",
'Puskesmas_Air_Saga 166.3171670634257']
```

tc5.txt




```
> tc5.txt
Ini adalah jalan dari H ke D
['H 0', 'G 154.5539140024254', 'F 304.9415397461352', 'A 398.30658825512006',
'E 462.75842788804005', 'D 624.4777433210588']
```

tc6.txt




```
Ini adalah jalan dari H ke D
['H    0',    'G    472.73777086255916',    'L    1304.8393271639527',    'P
1805.3290872991215',  'N  2177.0332707570446',  'E  2835.2437234241215',  'D
3873.7281961055955']
```

BAB 4

CHECKLIST

Poin	Ya	Tidak
Program dapat menerima input graf.	✓	
Program dapat menghitung lintasan terpendek.	✓	
Program dapat menampilkan lintasan terpendek serta jaraknya.	✓	
Bonus: Program dapat menerima input peta dengan Google Map API dan menampilkan peta		✓

BAB 5
REPOSIT

Reposit dari program dapat diakses di sini.

DAFTAR PUSTAKA

[1] Slide Kuliah Minggu 11, Algoritma A*. Rinaldi Munir. 2021. http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian2-2021.pdf

[2] Visualize Graphs in Python. 2020. kshitijjainm. https://www.geeksforgeeks.org/visualize-graphs-in-python/