LAPORAN TUGAS KECIL 3
IF2211
STRATEGI ALGORITMA

Implementasi Algoritma A* untuk Menentukan Lintasan Terpendek

Disusun oleh:
Nabil Nabighah - 13519168
Muhammad Furqon - 13519184

TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2020/2021

A. Source Code

In [1]:
```
# Install a conda package in the current Jupyter kernel
import sys
!conda install --yes --prefix {sys.prefix} ipyleaflet
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.
```

In [21]:
```
import math
class Graf(object):
    def __init__(self, size, dictgraf = None, dictTetangga = None):
        #atribut
        if dictgraf is None:
            dictgraf = {}
        self.dictgraf = dictgraf
        if dictTetangga is None:
            dictTetangga = {}
        self.dictTetangga = dictTetangga
        self.arraySimpul = []
        #adj matrix size
        self.size = size
        #array adj matrix
        self.adj_matrix = []
        for i in range(size):
            self.adj_matrix.append([0 for i in range(size)])
        self.distance_matrix  = []
        for i in range(size):
            self.distance_matrix.append([0 for i in range(size)])
        self.real_distance_matrix  = []
        for i in range(size):
            self.real_distance_matrix.append([0 for i in range(size)])
    #method (simpul)
    def addSimpul(self,simpul,target):
            if(target == "Simpul"):
                if simpul not in self.dictgraf:
                    self.dictgraf[simpul] = []
            elif(target == "Tetangga"):
                if simpul not in self.dictTetangga:
                    self.dictTetangga[simpul] = []
    # get simpul
    def getSimpul(self,target):
        if(target == "Simpul"):
            return list(self.dictgraf.keys())
        elif(target == "Tetangga"):
            return list(self.dictTetangga.keys())
    # printSimpul
    def printSimpul(self,target):
        if(target == "Simpul"):
            print(list(self.dictgraf.keys()))
        elif(target == "Tetangga"):
            print(list(self.dictTetangga.keys()))
```

```
#method (sisi)
    #add sisi
def addSisi(self, sisi,simpul,target):
    if(target == "Sisi"):
        if simpul in self.dictgraf:
            if len(self.dictgraf[simpul]) ==0:
                self.dictgraf[simpul] = [sisi]
            else:
                self.dictgraf[simpul].append(sisi)
    elif(target == "Tetangga"):
        simpul = self.arraySimpul[simpul]
        sisi = self.arraySimpul[sisi]
        if simpul in self.dictTetangga:
            if len(self.dictTetangga[simpul]) ==0:
                self.dictTetangga[simpul] = [sisi]
            else:
                self.dictTetangga[simpul].append(sisi)


    #print sisi
def printSisi(self,target):
    if(target == "Sisi"):
        print(list(self.dictgraf.values()))
    elif(target == "Tetangga"):
        print(list(self.dictTetangga.values()))
    #get sisi
def getSisi(self):
    if(target == "Simpul"):
        return list(self.dictgraf.values())
    elif(target == "Tetangga"):
        return list(self.dictTetangga.values())
def getX(self,simpul):
    return float(self.dictgraf[simpul][0])
def getY(self,simpul):
    return float(self.dictgraf[simpul][1])
#method size
def getSize(self):
    return self.size
def getArraySimpul(self):
    return self.arraySimpul
def getArrayJarak(self):
    return self.real_distance_matrix
```

```
    #method Matrix
        #print matrix
    def printMatrix(self,target):
        if(target == "Adj"):
            for i in range(self.size):
                for j in range(self.size):
                    print(self.adj_matrix[i][j], end=" ")
                print("")
        elif(target == "Euc"):
            for i in range(self.size):
                for j in range(self.size):
                    print(self.distance_matrix[i][j], end=" ")
                print("")
        elif(target == "Real"):
            for i in range(self.size):
                for j in range(self.size):
                    print(self.real_distance_matrix[i][j], end=" ")
                print("")

        #set  matrix
    def setMatrix(self,matrix,target):
        #set adj matrix sekalian dengan addSisi di dictTetangga
        if(target == "Adj"):
            for baris in range(self.size):
                counter = 0
                for kolom in matrix[baris]:
                    if(kolom == "1"):
                        self.adj_matrix[baris][counter] = 1
                        self.addSisi(counter,baris,"Tetangga")
                    counter += 1

        elif(target == "Euc"):
            for baris in range(self.size):
                counter = 0
                for kolom in matrix[baris]:
                    self.distance_matrix[baris][counter] = kolom
                    counter += 1

        elif(target == "Real"):
            for baris in range(self.size):
                counter = 0
                for kolom in matrix[baris]:
                    self.real_distance_matrix[baris][counter] = kolom
                    counter += 1

    #method array Tetangga
    def addArraySimpul(self,simpul):
        self.arraySimpul.append(simpul)
```

```
def haversineEuclidean(self,simpul1,simpul2):
    x1 = self.getX(simpul1)
    y1 = self.getY(simpul1)
    x2 = self.getX(simpul2)
    y2 = self.getY(simpul2)
    selisihY = (y2 - y1) * math.pi / 180.0
    selisihX = (x2 - x1 ) * math.pi / 180.0
    # convert to radians
    y1 = (y1) * math.pi / 180.0
    y2 = (y2) * math.pi / 180.0

    # apply formulae
    a = (pow(math.sin(selisihY / 2), 2) +
        pow(math.sin(selisihX / 2), 2) *
            math.cos(y1) * math.cos(y2));
    rad = 6371000
    c = 2 * math.asin(math.sqrt(a))
    return rad * c
```

In [22]:

```python
#Kelas untuk menyimpan nama dan nilai g,h,f
class Simpul():
    def __init__(self, parent=None, name=None):
        #nilai parent dan position
        self.parent = parent
        self.name = name

        self.g = 0
        self.h = 0
        self.f = 0

    def __eq__(self,other):
        return self.name == other.name

    def __lt__(self,other):
        return float(self.f) < float(other.f)

# take f for sort
def takeF(elem):
    return float(elem.f)

#Algoritma A* untuk mencari rute
def astar(graf,start,end):

    #Buat simpul awal dan akhir
    start_node = Simpul(None,start)
    start_node.g = start_node.h = start_node.f =0
    end_node = Simpul(None,end)
    end_node.g = end_node.h = end_node.f =0

    #inisialisasi open dan closed list
    #open: live node, closed: expanded node
    open_list = []
    closed_list = []

    #tambahkan start node
    open_list.append(start_node)

    #buat list of simpul
    list_simpul = graf.getSimpul("Simpul")

    #indeks untuk start dan elnd
    start_index = list_simpul.index(start)
    end_index = list_simpul.index(end)

    #Loop sampai mencapai end
    while len(open_list) > 0:
```

```
    while len(open_list) > 0:
        #Ambil curr node dengan f terkecil
        current_node = open_list[0]
        current_index = 0
        for item in open_list:
            if (item < current_node):
                current_node = item
                current_index = index

        #Pop dan masukkan ke closed list
        open_list.pop(current_index)
        closed_list.append(current_node)

        #Menemukan goal, catat pathnya
        if current_node == end_node:
            path = []
            current = current_node
            while current is not None:
                #Debug
                #print("Muncul")
                path.append(current.name)
                current = current.parent
            return path[::-1] #path adalah array yang direversed

        #Generate children
        list_children = graf.dictTetangga[current_node.name]

        children = []
        for child_element in list_children:
            #Buat simpul baru
            new_node = Simpul(current_node,child_element)
            #Append
            children.append(new_node)

        #Loop untuk setiap children
        for child in children:
            skip = False
            #Cek child apakah ada di closed list (sudah expanded)
            for closed_child in closed_list:
                if child == closed_child:
                    skip = True
                    continue
            if(skip):
                continue

            #Mengkalkulasikan nilai f dari nilai g dan h
            child_index = list_simpul.index(child.name)

            #dari matrix real
            child.g = float(graf.real_distance_matrix[child_index][start_index])
            #dari matrix euclidean distance
            child.h = float(graf.distance_matrix[child_index][end_index])
            #f didapat dari g dan h
            child.f = child.g + child.h

            #Cek child apakah ada di open list (di live node) dan g lebih besar
            for open_node in open_list:
                if child == open_node and child.g >= open_node.g:
                    skip = True
                    continue
            if(skip):
                continue

            #Tambahkan child ke open list (live node)
            open_list.append(child)
            open_list.sort(key=takeF)

    return False
```

In [24]:

```
def getJarakTotal(list_hasil,graf):
    arrayNode = graf.getArraySimpul()
    matrixJarak = graf.getArrayJarak()
    totalJarak = 0
    #masuk sebuah list hasil isinya adalah nama simpul
    for i in range(len(list_hasil)):
        if(i != len(list_hasil) -1):
            index1 = arrayNode.index(list_hasil[i])
            index2 = arrayNode.index(list_hasil[i+1])
            totalJarak += float(matrixJarak[index1][index2])
    return (totalJarak)


def handle_click(**kwargs):
    global clicked_once
    koordinat = kwargs["coordinates"]
    if(not clicked_once):
        first_click(koordinat)
        clicked_once = True
    else:
        second_click(koordinat)
        clicked_once = False
# INISIALISASI
filename = input("Masukkan nama file kordinat (filename.txt): ")
filematrix = input("Masukkan nama file matrix (filename.txt): ")
filejarak =  input("Masukkan nama file jarak (filename.txt): ")
f = open(filename, "r")
fm = open(filematrix, "r")
fg = open(filejarak,"r")

tempArr = []
tempMatrix = []
tempEucledean = []
tempReal = []
for item in f:
    tempArr.append(item.rstrip("\n").rsplit(','))
for item in fm:
    tempMatrix.append(item.rstrip("\n"))
for item in fg:
    tempReal.append(item.rstrip("\n").rsplit(" "))
```

```
# MAIN DRIVER
graf = Graf(int(tempArr[0][0]))
del tempArr[0]
for item in tempArr:
    for i in range(3):
        if(i == 2):
            graf.addArraySimpul(item[i])
            graf.addSimpul(item[i],"Simpul")
            graf.addSimpul(item[i],"Tetangga")
            graf.addSisi(item[0],item[i],"Sisi")
            graf.addSisi(item[1],item[i],"Sisi")

tempListEuc = []
for item in tempArr:
    for item2 in tempArr:
        if(item[2] != item2[2]):
            hasil = graf.haversineEuclidean(item[2],item2[2])
            tempListEuc.append(hasil)
        else:
            tempListEuc.append(0)
    tempEucledean.append(tempListEuc)
    tempListEuc = []
graf.setMatrix(tempMatrix,"Adj")
graf.setMatrix(tempEucledean,"Euc")
graf.setMatrix(tempReal,"Real")

#Print
# print("======= MATRIX REAL=======")
# graf.printMatrix("Real")
# print("====== INI SIMPUL========")
# graf.printSimpul("Simpul")
# print("====== KOORDINATNYA======")
# graf.printSisi("Sisi")

#menerima masukan astar(graf,start,end)
# list_hasil = astar(graf,"Gelap Nyawang", "MCD Dago")

#VISUALISASI MAP
#With search ITB
from ipyleaflet import *
from ipywidgets import *
center = [tempArr[0][0], tempArr[0][1]]
zoom = 15

m = Map(basemap=basemaps.OpenStreetMap.Mapnik, center=center, zoom=zoom)
```

```
marker = Marker(icon=AwesomeIcon(name="check", marker_color='green', icon_color='darkgreen'))

#search
m.add_control(SearchControl(
  position="topleft",
  url='https://nominatim.openstreetmap.org/search?format=json&q={s}',
  zoom=5,
  marker=marker
))

#measure
measure = MeasureControl(
    position='bottomleft',
    active_color = 'orange',
    primary_length_unit = 'kilometers'
)
m.add_control(measure)

measure.completed_color = 'red'

measure.add_length_unit('yards', 1.09361, 4)
measure.secondary_length_unit = 'yards'

measure.add_area_unit('sqyards', 1.19599, 4)
measure.secondary_area_unit = 'sqyards'

#marker untuk setiap simpul
list_simpul = graf.getSimpul("Simpul")
for simpul_peta in list_simpul:
    lokasi = graf.dictgraf[simpul_peta]
    marker = Marker(location=lokasi, draggable=False)
    m.add_layer(marker)
    marker.on_click(handle_click)

#Popup
def createPopup(koordinat,isi_message):
    message1 = HTML()
    # cari key
    message1.value = isi_message
    # Popup with a given location on the map:
    popup = Popup(
        location=koordinat,
        child=message1,
        close_button=True,
        auto_close=False,
        close_on_escape_key=False
    )
    m.add_layer(popup)
```

```
def first_click(koordinat):
    global graf,key_list,val_list,titik_awal
    #print("klik pertama")
    koordinat = [str(i) for i in koordinat]
    position = val_list.index(koordinat)
    titik_awal=key_list[position]
    createPopup(koordinat,"Titik mulai:"+titik_awal)

def second_click(koordinat):
    global graf,key_list,val_list, list_hasil,titik_awal
    #print("klik kedua")
    koordinat = [str(i) for i in koordinat]
    position = val_list.index(koordinat)
    titik_selesai = key_list[position]
    list_hasil = astar(graf,titik_awal, titik_selesai)
    if(list_hasil == False):
        createPopup(koordinat,"Tidak ada jalur")
    else:
        createPopup(koordinat,"Titik selesai:"+titik_selesai+" Jarak :"+"{:.2f}".format(getJarakTotal(list_hasil,graf)))
        create_path()

created = False
def create_path():
    #List koordinat untuk path
    global ant_path,created
    if not created:
        created=True
        list_koord = []
        for simpul_peta in list_hasil:
            list_koord.append(graf.dictgraf[simpul_peta])

        #antpath
        ant_path = AntPath(
            locations=list_koord,
            dash_array=[1, 10],
            delay=1000,
            color='#7590ba',
            pulse_color='#3f6fba'
        )
        m.add_layer(ant_path)
    else:
        list_koord = []
        for simpul_peta in list_hasil:
            list_koord.append(graf.dictgraf[simpul_peta])
        ant_path.locations = list_koord
m
```

B. Peta/Graf Input

Pada program kami, terdapat tiga buah input file. File tersebut adalah file map, jarak, dan adjacency matrix. Dengan masing-masing isi file sebagai berikut.

1. File Map

map.txt

```
8
-6.893219312910053,107.61046032792717,Depan ITB
-6.884899385475644,107.6088547011027,Babakan Siliwangi
-6.88503464050968,107.61365684524623,MCD Dago
-6.896847079799714,107.6095702214672,Pelesiran
-6.8987059488531965,107.61265782586379,Dago Park
-6.887686407803358,107.61011081549059,Sabuga
-6.894775281631545,107.61068950499423,Gelap Nyawang
-6.898399101748751,107.61068034023243,Tirta Anugrah
```

Penjelasan mengenai isi dari file ini, pada baris pertama menyatakan banyak simpul yaitu 8 kemudian dari kiri ke kanan, file berisi koordinat x dan koordinat y pada google map, dan nama dari lokasi pada koordinat tersebut. Koordinat x dan koordinat y dan nama lokasi dipisahkan menggunakan koma.

2. File Jarak

```
0 1820 1770 589.36 846.92 1100 197.97 849.44
1820 0 552.38 2140 2630 920.55 2060 2400
1770 552.38 0 1720 2450 508.60 1650 1990
589.36 2140 1720 0 494.75 1220 384.65 257.22
846.92 2630 2450 494.75 0 1650 1180 583.10
1100 920.55 508.60 1220 1650 0 1150 1500
197.97 2060 1650 384.65 1180 1150 0 648.38
849.44 2400 1990 257.22 583.10 1500 648.38 0
```

File jarak memiliki isi sebuah matrix NxN dimana N adalah jumlah simpul/jumlah lokasi. Isi dari matrix tersebut adalah jarak dari setiap simpul ke simpul lainnya dan jarak simpul ke-i dengan simpul ke-j dimana i = j yaitu sebesar 0. Jarak diukur menggunakan rules pada google maps secara manual dari setiap titik simpul ke titik simpul lain.

3. **File Matrix**

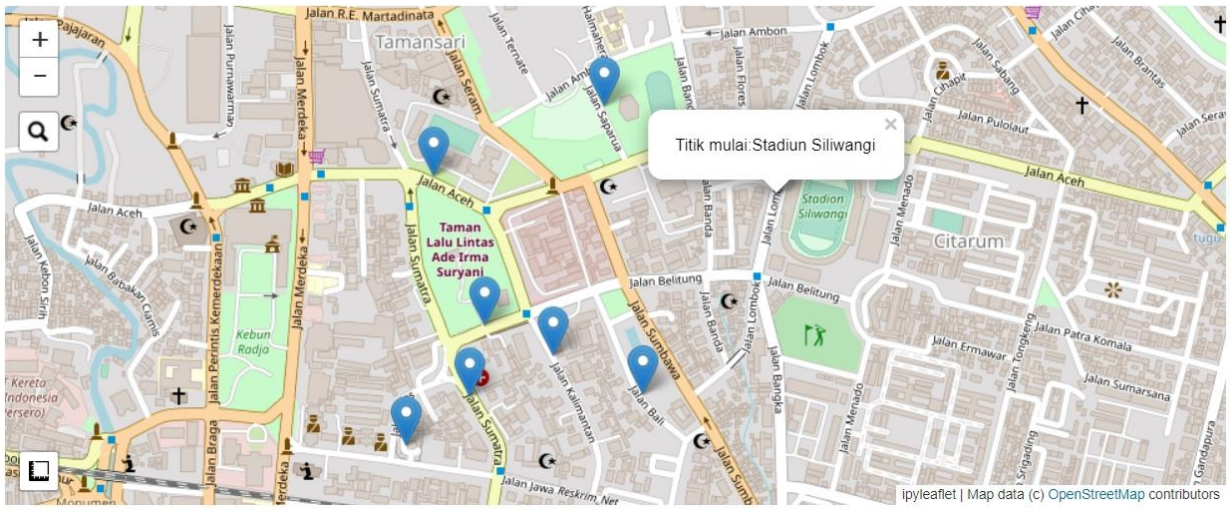
```
00010110
00100100
01001100
10000011
00100011
11100000
10011000
00011000
```

File matrix memiliki isi sebuah matrix NxN dimana N adalah jumlah simpul/jumlah lokasi. Isi dari matrix ini adalah digit 0 dan 1 dimana setiap digit merepresentasikan tetetanggaan. Digit 0 menunjukkan bahwa baris ke-x kolom ke-y tidak bertetangga. Digit 1 menunjukkan bahwa baris ke-x kolom ke-y bertetangga.

C. *Screenshot*

Contoh simpul yang memiliki jalur :

```
Masukkan nama file kordinat (filename.txt): map6.txt
Masukkan nama file matrix (filename.txt): matrix6.txt
Masukkan nama file jarak (filename.txt): jarakreal6.txt
```

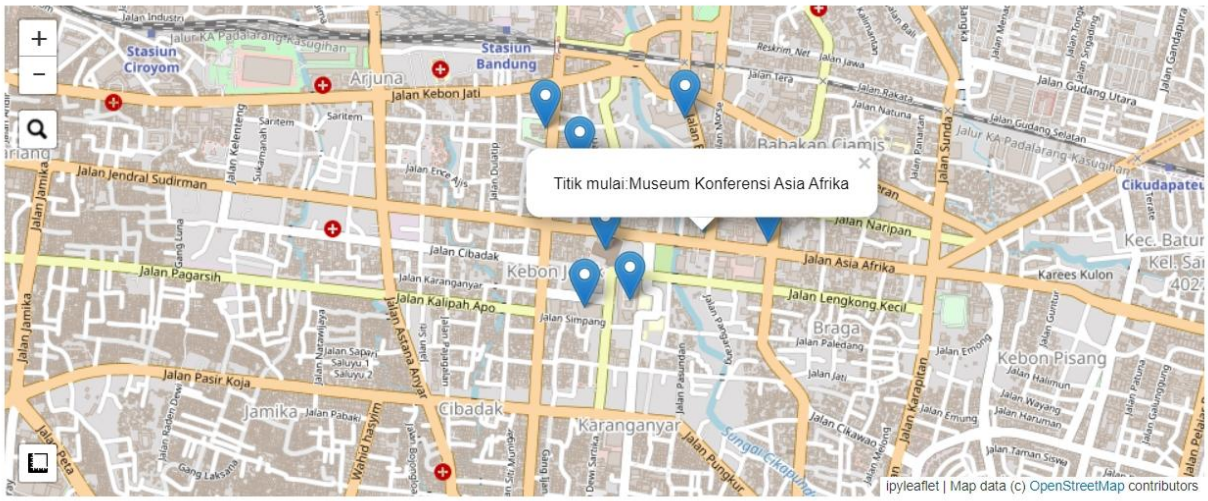


```
Masukkan nama file kordinat (filename.txt): map6.txt
Masukkan nama file matrix (filename.txt): matrix6.txt
Masukkan nama file jarak (filename.txt): jarakreal6.txt
```

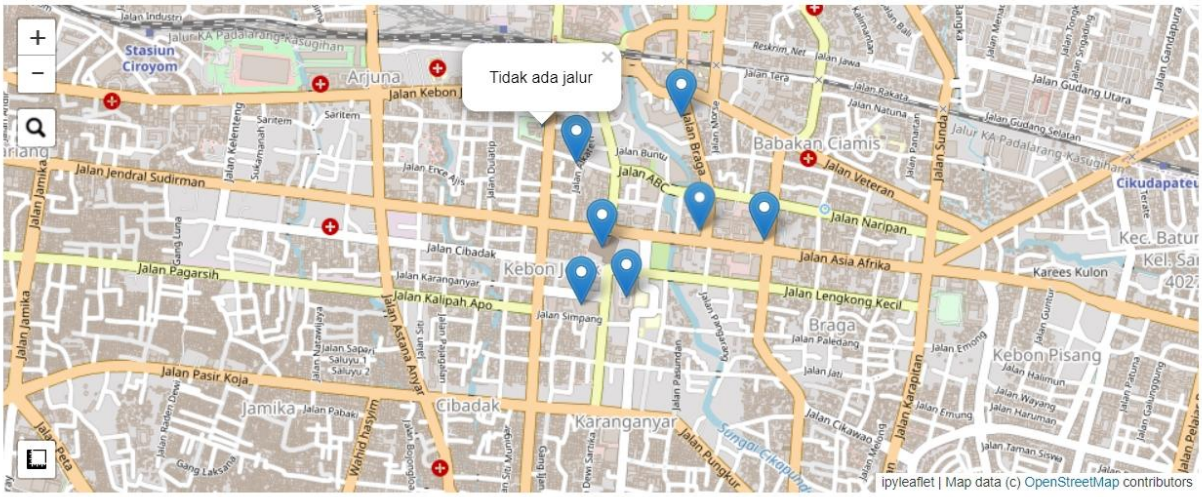
Contoh simpul yang tidak memiliki jalur :

Dalam hal ini, simpul Museum Konferensi Asia Afrika hanya bertetangga dengan PT. Garuda Indonesia (Persero) Tbk sehingga tidak memiliki jalur ke simpul lain selain ke simpul PT. Garuda Indonesia (Persero) Tbk.

```
Masukkan nama file kordinat (filename.txt): map3.txt
Masukkan nama file matrix (filename.txt): matrix3.txt
Masukkan nama file jarak (filename.txt): jarakreal3.txt
```



```
Masukkan nama file kordinat (filename.txt): map3.txt
Masukkan nama file matrix (filename.txt): matrix3.txt
Masukkan nama file jarak (filename.txt): jarakreal3.txt
```



D. Alamat *Source Code*

https://github.com/Uyamikun/Tucil3Stima (private repository)
https://drive.google.com/drive/folders/1gsIiZDOW3TG1IFMOPYTgigdBzI2bnxCK?usp=sharing (public drive)