Laporan Tugas Kecil 3 IF2211 Strategi Algoritma
Semester II Tahun 2020/2021

Implementasi Algoritma A* untuk Menentukan Lintasan Terpendek

Disusun Oleh:
Syarifah Aisha Geubrina Yasmin - 13519089
Aulia Adila - 13519100

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
SEMESTER II TAHUN 2020/2021

I. Kode Program

```python
from math import radians, cos, sin, asin, sqrt

# Membaca masukan dari file txt dan mengembalikannya
# dalam bentuk array
def readinput(filename):
    file = open(filename)
    lines = file.read().splitlines()
    return lines

# Mengubah array of strings menjadi array of words
def makeArrayofWords(lines):
    arrayOfWords = []
    for line in lines:
        line = line.split(' ')
        arrayOfWords.append(line)
    return arrayOfWords

# Memisahkan matriks ketetanggaan dari input file
def makeAdjancentMatrix(arrayOfWords):
    adjacentMatrix = []
    N = int(arrayOfWords[0][0])
    for i in range (N+1,(len(arrayOfWords))):
        adjacentMatrix.append(arrayOfWords[i])
    return adjacentMatrix
```

```python
# Memisahkan matriks node dari input file
def makeNodeMatrix(arrayOfWords):
    nodeMatrix = []
    N = int(arrayOfWords[0][0])
    for i in range (1,N+1):
        nodeMatrix.append(arrayOfWords[i])
    return nodeMatrix

def dist(nodeMatrix, source, destination):
    #Penyederhanaan : lat (x), long(y)
    for i in range (len(nodeMatrix)):
        if (source == nodeMatrix[i][0]):
            node1 = nodeMatrix[i]
        if (destination == nodeMatrix[i][0]):
            node2 = nodeMatrix[i]

    x1 = radians(float(node1[2]))
    x2 = radians(float(node2[2]))
    y1 = radians(float(node1[1]))
    y2 = radians(float(node2[1]))
    # Haversine formula
    dx = x2 - x1 #longitude
    dy = y2 - y1 #latitude
    a = sin(dy / 2)**2 + cos(y1) * cos(y2) * sin(dx / 2)**2
    c = 2 * asin(sqrt(a))

    r = 6371
    result = (c * r)*1000
    return round(result,3)
```

```python
def changeMatrix(adjacentMatrix, nodeMatrix):
    newadjacentMatrix = [ [ 0 for i in range(len(adjacentMatrix)) ] for j in range(len(adjacentMatrix)) ]

    for i in range(len(adjacentMatrix)):
        for j in range(i):
            if(adjacentMatrix[i][j] == "1"):
                newadjacentMatrix[i][j] = dist(nodeMatrix, nodeMatrix[i][0],nodeMatrix[j][0])
                newadjacentMatrix[j][i] = newadjacentMatrix[i][j]
            #print(adjacentMatrix[i][j], end=' ')
        #print('\n')
    return newadjacentMatrix

#Membuat dictionary dari matrix dan tetangganya
def makeDictionary(adjacentMatrix,nodeMatrix):
    dictionary_adj = dict()
    weighted_adj_matrix = changeMatrix(adjacentMatrix, nodeMatrix)
    for i in range(len(adjacentMatrix)):
        tempAdj = []
        for j in range (len(adjacentMatrix)):
            if (adjacentMatrix[i][j] != '0'):
                tempAdj.append((nodeMatrix[j][0], weighted_adj_matrix[i][j]))
        dictionary_adj.setdefault(nodeMatrix[i][0], tempAdj)
    return dictionary_adj
```

```python
from read import dist

def functionHN(nodeMatrix, current, destination):
    return dist(nodeMatrix, current, destination)

def functionGN(adjacentMatrix, parent, current, curr_cost):
    adj_parent = adjacentMatrix.get(parent)
    for i in range (len(adj_parent)):
        if (adj_parent[i][0] == current):
            return round(adj_parent[i][1]+curr_cost,3)
    return -1

def searchDist (dictionary_adj,node1, node2):
    for key, value in dictionary_adj.items():
        if key == node1:
            for node in value:
                if node[0] == node2:
                    distance = node[1]
                    break
    return distance

def getIdx(nodeMatrix, node):
    for i in range (len(nodeMatrix)):
        if (node == nodeMatrix[i][0]):
            return i
    return -1
```

```python
def updateGN(start_node, end_node, visited_Nodes, dictionary_adj, nodeMatrix):
    #path = []
    total_gn = 0
    from_node = start_node
    while (from_node != end_node):
        # Node tujuan sementara adalah previous node dari from_node
        to_node = visited_Nodes.get(from_node)[3]
        #cari jarak dari curr_node ke start_node
        total_gn += searchDist(dictionary_adj, from_node, to_node)
        from_node = to_node
        continue

    return total_gn


def functionFN(adjacentMatrix, parent, current, destination, curr_cost, nodeMatrix):
    return round(functionGN(adjacentMatrix, parent, current, curr_cost)+functionHN(nodeMatrix,current, destination),3)

def sortVisitedNodes(visited_Nodes):
    #new_visited_Nodes = dict()
    new_visited_Nodes = dict(sorted(visited_Nodes.items(), key=lambda item: item[1][2]))
    return new_visited_Nodes
```

```python
def aStar(adjacentMatrix, nodeMatrix, source, destination):
    opened = dict() #berisi node yang diekspan dan akan dibandingkan nilainya
    visited_Nodes = dict()
    closed = []
    # inisialisasi dict visited_Nodes
    # visited_Nodes dictionary
    # KEY    VALUE
    #          0      1      2        3
    # node - g(n) - h(n) - f(n) - prev_node

    current_gn = 0
    current_node = source
    visited_Nodes.setdefault(source,(0,functionHN(nodeMatrix, source, destination),functionHN(nodeMatrix, source, destination),""))
    #jangan lupa tangani kalau ga nemu path
    while (current_node != destination):
        temp_adj = adjacentMatrix.get(current_node) #tetangga yg lg ditinjau
        for i in range(len(temp_adj)):
            temp_gn = functionGN(adjacentMatrix, current_node, temp_adj[i][0], current_gn)
            temp_hn = functionHN(nodeMatrix, temp_adj[i][0], destination)
            temp_fn = temp_gn + temp_hn
            if (temp_adj[i][0] not in visited_Nodes.keys()):
                visited_Nodes.setdefault(temp_adj[i][0], (temp_gn, temp_hn, temp_fn, current_node))
                opened.setdefault(temp_adj[i][0], (temp_gn, temp_hn, temp_fn, current_node))
            else:
                if (temp_fn < visited_Nodes.get(temp_adj[i][0])[2]):
                    update_value = {temp_adj[i][0] : (temp_gn, temp_hn, temp_fn, current_node)}
                    visited_Nodes.update(update_value)
                    opened.update(update_value)

        #sort visited nodes
        visited_Nodes = sortVisitedNodes(visited_Nodes)
        for key, value in visited_Nodes.items():
            if (key in opened.keys()):
                current_node = key
                current_gn = value[0]
                del opened[current_node]
                break

    if (current_node == destination):
        current_backtrack = current_node

        while (current_backtrack != ""):
            closed.insert(0,current_backtrack)
            current_backtrack = visited_Nodes.get(current_backtrack)[3]

    return closed, visited_Nodes
```

```python
def getTotalCost(closed,visited_Nodes):
    return visited_Nodes.get(closed[len(closed)-1])[2]

def getCost(node1, node2, weighted_adj_matrix, nodeMatrix):
    return weighted_adj_matrix[getIdx(nodeMatrix,node1)][getIdx(nodeMatrix,node2)]
```

```
# import folium package
import folium
from AStar_2 import aStar, getIdx, getTotalCost
from read import readinput, makeArrayofWords, makeAdjancentMatrix, makeDictionary, makeNodeMatrix

#Masukan argumen pada filename dengan format "namafile.txt"
#File testcase yang tersedia : [itb | alun2 | ahmaddahlan | buahbaru]
filename = "itb.txt"
lines = readinput("../test/" + filename)
arrayOfWords = makeArrayofWords(lines)
adjacentMatrix = makeAdjancentMatrix(arrayOfWords)
nodeMatrix = makeNodeMatrix(arrayOfWords)
dictionary_adj = makeDictionary(adjacentMatrix,nodeMatrix)

listofNodes = []
for nodes in nodeMatrix:
    listofNodes.append(nodes[0])

print("============================")
print("|List of Available Nodes : |")
for node in listofNodes:
    print("|- ",node, "                    |")
print("============================")


def Pusat(nodeMatrix):
    # Ambil rerataan koordinat

    sum_lat = 0
    sum_long = 0
    for i in range(len(nodeMatrix)):
        sum_lat += float(nodeMatrix[i][1])
        sum_long += float(nodeMatrix[i][2])
    sum_lat = sum_lat/len(nodeMatrix)
    sum_long = sum_long/len(nodeMatrix)

    pusat = [sum_lat,sum_long]
    return pusat


pusat = Pusat(nodeMatrix)
my_map1 = folium.Map(location = pusat, zoom_start = 12 )

def initializedMap(dictionary_adj, nodeMatrix, my_map1):
    for key,value in dictionary_adj.items():
        #node1
        m_long1 = float(nodeMatrix[getIdx(nodeMatrix,key)][1])
        m_lat1 = float(nodeMatrix[getIdx(nodeMatrix,key)][2])
        name1 = key
```

```
        name1 = key

        #menandai letak node
        folium.Marker([m_long1,m_lat1], popup = name1).add_to(my_map1)

        for v in value:
            #node2
            m_long2 = float(nodeMatrix[getIdx(nodeMatrix,v[0])][1])
            m_lat2 = float(nodeMatrix[getIdx(nodeMatrix,v[0])][2])

            #warnain jalur antara node 1 dan node 2
            folium.PolyLine(locations = [(m_long1,m_lat1), (m_long2,m_lat2)],
            line_opacity = 0.2).add_to(my_map1)

    return my_map1

my_map1 = initializedMap(dictionary_adj, nodeMatrix, my_map1)
my_map1
```

```
# import folium package
import folium
from AStar_2 import aStar, getIdx, getTotalCost
from read import readinput, makeArrayofWords, makeAdjancentMatrix, makeDictionary, makeNodeMatrix

#Masukan argumen pada filename dengan format "namafile.txt"
#File testcase yang tersedia : [itb | alun2 | ahmaddahlan | buahbaru]
filename = "itb.txt"
lines = readinput("../test/" + filename)
arrayOfWords = makeArrayofWords(lines)
adjacentMatrix = makeAdjancentMatrix(arrayOfWords)
nodeMatrix = makeNodeMatrix(arrayOfWords)
dictionary_adj = makeDictionary(adjacentMatrix,nodeMatrix)

#Ganti argumen 3 dan 4 pada fungsi aStar menjadi list of nodes yang tersedia
#============================
#|List of Available Nodes : |
#|-  A                     |
#|-  B                     |
#|-  C                     |
#|-  D                     |
#|-  E                     |
#|-  F                     |
#|-  G                     |
#|-  H                     |
#|-  I                     |
#|-  J                     |
#|-  K                     |
#|-  L                     |
#/-  M                     |
#============================
closed,visited_Nodes = aStar(dictionary_adj, nodeMatrix, 'C', 'J')

#INFORMASI TAMBAHAN
print("Shortest Path  : ", end="")
for i in range(len(closed)):
    if i < len(closed)-1:
        print(closed[i], end= " -> ")
    else:
        print(closed[i])

print("Total Distance : ", getTotalCost(closed,visited_Nodes), " m")
print("")
```

```
def Pusat(nodeMatrix):
    sum_lat = 0
    sum_long = 0
    for i in range(len(nodeMatrix)):
        sum_lat += float(nodeMatrix[i][1])
        sum_long += float(nodeMatrix[i][2])
    sum_lat = sum_lat/len(nodeMatrix)
    sum_long = sum_long/len(nodeMatrix)

    pusat = [sum_lat,sum_long]
    return pusat


pusat = Pusat(nodeMatrix)
my_map1 = folium.Map(location = pusat, zoom_start = 12 )

def initializedMap(dictionary_adj, nodeMatrix, my_map1):
    for key,value in dictionary_adj.items():
        #node1
        m_long1 = float(nodeMatrix[getIdx(nodeMatrix,key)][1])
        m_lat1 = float(nodeMatrix[getIdx(nodeMatrix,key)][2])
        name1 = key

        #menandai letak node
        folium.Marker([m_long1,m_lat1], popup = name1).add_to(my_map1)

        for v in value:
            #node2
            m_long2 = float(nodeMatrix[getIdx(nodeMatrix,v[0])][1])
            m_lat2 = float(nodeMatrix[getIdx(nodeMatrix,v[0])][2])

            #warnain jalur antara node 1 dan node 2
            folium.PolyLine(locations = [(m_long1,m_lat1), (m_long2,m_lat2)],
            line_opacity = 0.2).add_to(my_map1)

    return my_map1


def updateMap(closed, dictionary_adj, my_map1, nodeMatrix):
    for i in range(len(closed)-1):
        #node1
        m_long1 = float(nodeMatrix[getIdx(nodeMatrix,closed[i])][1])
        m_lat1 = float(nodeMatrix[getIdx(nodeMatrix,closed[i])][2])

        #node2
        m_long2 = float(nodeMatrix[getIdx(nodeMatrix,closed[i+1])][1])
        m_lat2 = float(nodeMatrix[getIdx(nodeMatrix,closed[i+1])][2])

        #warnain jalur antara node 1 dan node 2
        folium.PolyLine(locations = [(m_long1,m_lat1), (m_long2,m_lat2)], color = 'red',
        line_opacity = 0.5).add_to(my_map1)

    return my_map1
```

```
my_map1 = initializedMap(dictionary_adj, nodeMatrix, my_map1)
my_map1 = updateMap(closed, dictionary_adj, my_map1, nodeMatrix)
my_map1
```

II. File Input

ahmaddahlan.txt

```
13
A -6.2451305520528315 106.78875286822782
B -6.2465684548354234 106.79136069567708
C -6.246583916134292 106.79192062681287
D -6.243914258430079 106.79170287581563
E -6.244357484396679 106.79018380338242
F -6.244625481310769 106.78964461043682
G -6.244878016738897 106.78920910844232
H -6.246104615657203 106.7913140347491
I -6.245635622292481 106.79128811201133
J -6.245589238310509 106.791837674052
K -6.245393750344293 106.79010575095998
L -6.245254172270691 106.79182922350667
M -6.244703936570421 106.79178387323083
0 1 0 0 0 0 1 0 0 0 0 0 0
1 0 1 0 0 0 0 1 0 0 0 0 0
0 1 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 1
0 0 0 1 0 1 0 0 0 0 1 0 0
0 0 0 0 1 0 1 0 0 0 1 0 0
1 0 0 0 0 1 0 1 0 0 0 0 0
0 1 0 0 0 0 1 0 1 0 0 0 0
0 0 0 0 0 0 0 1 0 1 1 0 0
0 0 1 0 0 0 0 0 1 0 0 1 0
0 0 0 0 1 1 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 1
0 0 0 1 0 0 0 0 0 0 0 1 0
```

alun2.txt

```
16
A -6.921141 107.607668
B -6.920768 107.604056
C -6.918263 107.604216
D -6.919038 107.606670
E -6.920995 107.606441
F -6.922551 107.607619
G -6.922771 107.609810
H -6.925376 107.610599
I -6.926075 107.610524
J -6.925835 107.607071
K -6.924356 107.607253
L -6.924317 107.606160
M -6.923950 107.603842
N -6.922388 107.606404
O -6.923443 107.606298
P -6.923100 107.603892
0 0 0 0 1 1 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 1
0 1 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
1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0
0 0 0 0 0 1 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 1 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 1 0 1 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 1 0 0 0 1
0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1
0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0
```

buahbatu.txt

```
10
A -6.936757 107.622691
B -6.947901 107.633486
C -6.947165 107.636042
D -6.941532 107.634648
E -6.93784 107.6272
F -6.94008 107.62576
G -6.94199 107.62754
H -6.94937 107.62593
I -6.9396 107.63096
J -6.93691 107.63211
0 0 0 0 0 1 0 0 0 0
0 0 1 0 0 0 1 1 0 0
0 1 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 1 0
0 0 0 0 0 1 0 0 1 0
1 0 0 0 1 0 1 0 0 0
0 1 0 0 0 1 0 1 0 0
0 1 0 0 0 0 1 0 0 0
0 0 0 1 1 0 0 0 0 1
0 0 0 0 0 0 0 0 1 0
```

itb.txt

```
13
A -6.893220 107.610454
B -6.892613 107.610428
C -6.892625 107.608842
D -6.891049 107.608705
E -6.891012 107.610386
F -6.890995 107.611570
G -6.892568 107.611680
H -6.889913 107.610374
I -6.889943 107.609006
J -6.889898 107.611563
K -6.888721 107.609086
L -6.888697 107.610370
M -6.888690 107.611537
0 1 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 1 0 1 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0 0 0 0
0 0 1 0 1 0 0 0 1 0 0 0 0
0 1 0 1 0 1 0 1 0 0 0 0 0
0 0 0 0 1 0 1 0 0 1 0 0 0
0 1 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 1 1 0 1 0
0 0 0 1 0 0 0 1 0 0 1 0 0
0 0 0 0 0 1 0 1 0 0 0 0 1
0 0 0 0 0 0 0 0 1 0 0 1 0
0 0 0 0 0 0 0 1 0 0 1 0 1
0 0 0 0 0 0 0 0 0 1 0 1 0
```

jakarta.txt

```
15
A -6.245131 106.788753
B -6.246568 106.791361
C -6.246584 106.791921
D -6.243914 106.791703
E -6.244357 106.790184
F -6.244625 106.789645
G -6.244878 106.789209
H -6.246105 106.791314
I -6.245636 106.791288
J -6.245589 106.791838
K -6.245394 106.790106
L -6.245254 106.791829
M -6.244802 106.790382
N -6.245968 106.790115
O -6.246423 106.789829
0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
0 0 1 0 0 0 0 1 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0
0 0 0 1 0 1 0 0 0 0 0 0 1 0 0
0 0 0 0 1 0 1 0 0 0 1 0 0 0 0
1 0 0 0 0 1 0 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 1 0 0 0 0 1 0
0 0 0 0 0 0 0 1 0 1 1 0 0 0 0
0 0 1 0 0 0 0 0 1 0 0 1 0 0 0
0 0 0 0 0 1 0 0 1 0 0 0 1 0 0
0 0 0 1 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 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
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
```

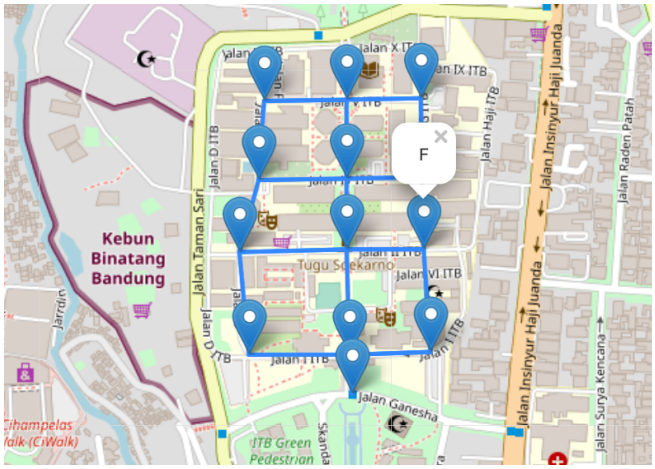
bandung.txt

```
8
A -6.940351 107.658245
B -6.939252 107.663915
C -6.943234 107.663564
D -6.942138 107.652719
E -6.955690 107.654484
F -6.956029 107.662112
G -6.954222 107.639885
H -6.946367 107.641756
0 1 0 1 0 0 0 0
1 0 1 0 0 0 0 0
0 1 0 0 0 1 0 0
1 0 0 0 1 0 0 1
0 0 0 1 0 1 1 0
0 0 1 0 1 0 0 0
0 0 0 0 1 0 0 1
0 0 0 1 0 0 1 0
```

III. Visualisasi Output

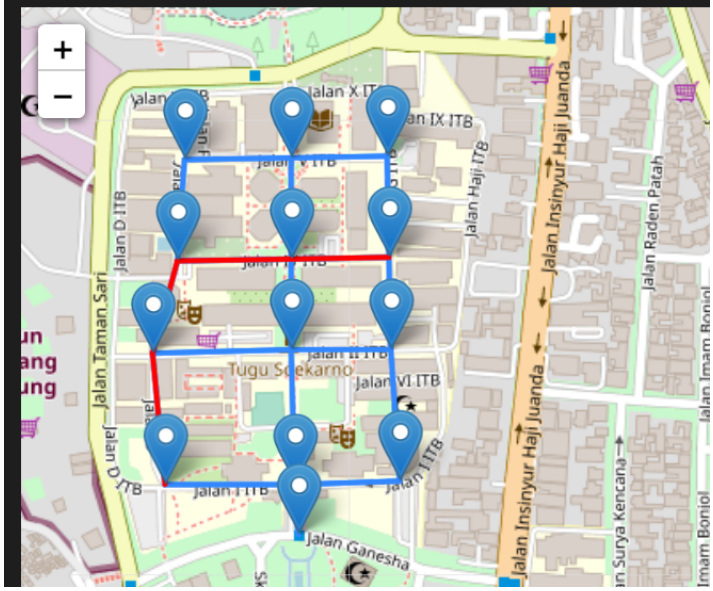
Inisialisasi Awal

```
============================
|List of Available Nodes : |
|-  A                      |
|-  B                      |
|-  C                      |
|-  D                      |
|-  E                      |
|-  F                      |
|-  G                      |
|-  H                      |
|-  I                      |
|-  J                      |
|-  K                      |
|-  L                      |
|-  M                      |
============================
```



Visualisasi Shortest Path

```
Shortest Path  : C -> D -> I -> H -> J
Total Distance :  585.606  m
```



IV. Tabel Checklist

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	✔

Visualisasi yang kami gunakan, meskipun tidak menggunakan Google Map API, tetapi dapat memvisualisasikan peta sesuai input koordinat pada file masukan dengan library Folium.

V. Link

https://github.com/auliaadila/TucilStima3.git