Laporan Tugas Kecil 3 IF2211 Strategi Algoritma

Oleh Muhammad Rifky Muthahhari – 13519123

Deskripsi Program

Program bernama AStar Path Finder. Program ini dibangun menggunakan bahasa pemrograman Python 3 dan Jupyter Notebook. Program menggunakan NetworkX dan Mathplotlib untuk menampilkan graf yang merepresentasikan map input yang diberikan. Baca README untuk melihat prerequisites dan cara menjalankan program.

Algoritma

1. Input map dibaca dengan menggunakan NetworkX. Bagian pertama input dibaca untuk membuat node pada graf dan begian kedua input dibaca untuk membuat edge antara node yang ada.
2. Pengguna memberikan input posisi awal dan tujuan pada map.
3. Fungsi AStar dipanggil untuk mencari jalur terdekat antara posisi awal dan tujuan.
4. Fungsi AStar akan memanggil fungsi g dan h dan melakukan penelusuran pada nodes menggunakan fungsi BFS
5. Result path kemudian diproses untuk ditampilkan pada jupyter output cell

Input File

Contoh input file:

```
./itb.txt
Pintu 3 KRB, -6.59577, 106.80434
Tugu Kujang, -6.60142, 106.80503
Pintu Masuk KRB, -6.60308, 106.79886
Bogor Trade Mall (BTM), -6.60430, 106.79663
McD Djuanda, -6.60104, 106.79500
SMAN 1 Bogor, -6.59669, 106.79385
Regina Pacis, -6.59300, 106.79719
Taman Sempur, -6.59222, 106.80084
Taman Air Mancur, -6.58141, 106.79674
Plaza Jambu Dua, -6.56898, 106.80909
Baso Seuseupan, -6.58086, 106.80678
Stasiun Bogor, -6.59602, 106.79033
---
Tugu Kujang, Pintu Masuk KRB
Pintu Masuk KRB, Bogor Trade Mall (BTM)
Bogor Trade Mall (BTM), SMAN 1 Bogor
SMAN 1 Bogor, Stasiun Bogor
SMAN 1 Bogor, Regina Pacis
Regina Pacis, Taman Sempur
Regina Pacis, Taman Air Mancur
Stasiun Bogor, Taman Air Mancur
Taman Sempur, Pintu 3 KRB
Pintu 3 KRB, Tugu Kujang
Pintu 3 KRB, Baso Seuseupan
Plaza Jambu Dua, Taman Air Mancur
Plaza Jambu Dua, Baso Seuseupan
```

Bagian pertama sebelum “---” dibaca untuk membuat node pada graf. Bagian kedua setelah “---” dibaca untuk membuat edge pada graf.

Source Code

```
./FileReader.txt
import matplotlib.pyplot as plt
```

```
import networkx as nx
import csv
import json

def read_config():
    f = open('config.json', 'r')
    data = json.load(f)
    f.close()
    return data['path']

def create_graph(G):
    path = read_config()
    f = open(path, 'r')
    input_text = f.read().split("---")
    input_line_nodes = input_text[0].strip().split('\n')
    input_nodes = []
    x_dplc = 0
    y_dplc = 0
    for l in input_line_nodes:
        input_nodes.append(l.split(','))
        firstNode = False
    for l in input_nodes:
        if (not firstNode):
            x_dplc = float(l[2])
            y_dplc = float(l[1])
            firstNode = True
        name = l[0].strip()
        posX = (float(l[2]) - x_dplc) * 100000
        posY = (float(l[1]) - y_dplc) * 100000
        pos = (posX, posY)
        G.add_node(name, pos=pos)
    input_line_edges = input_text[1].strip().split('\n')
    input_edges = []
    for edge in input_line_edges:
        e = edge.split(',')
        input_edges.append((e[0].strip(), e[1].strip()))
    G.add_edges_from(input_edges)
    nx.set_edge_attributes(G, 'k', 'color')
    f.close()

./AStar.py
import matplotlib.pyplot as plt
import networkx as nx
import math


def h(N, a, b):
    return (math.sqrt(math.pow((N[a][0] - N[b][0]), 2) + math.pow((N[a][1] - N[b][1]), 2)))


def g(N, route, a):
    cost = 0.0
    for i in range(len(route)-1):
        cost += h(N, route[i], route[i+1])
    cost += h(N, route[-1], a)
    return cost


def AStar(G, start, goal):
    result = []
    nodes = nx.get_node_attributes(G, 'pos')
    f = h(nodes, start, goal)
    stack = [[start, f]]
    BFS(G, goal, stack, result, nodes)
    return result


def BFS(G, goal, stack, result, nodes):
    if (stack and checkSmallerThanResult(stack, result)):
        currRoute = stack.pop(0)
        route_list = currRoute[0].split("-")
        route_cost = currRoute[1]
        if (route_list[-1] == goal):
            if (not result):
```

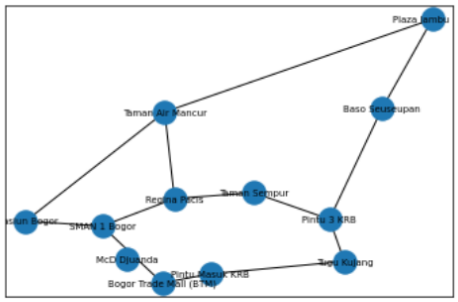
```
                result.append(currRoute)
            elif (route_cost < result[0][1]):
                result.clear()
                result.append(currRoute)
        else:
            for node in G[route_list[-1]]:
                if (node in [route_list]):
                    continue
                else:
                    f = g(nodes, route_list, node) + h(nodes, node, goal)
                    route_list.append(node)
                    stack.append(["-".join(route_list), f])
                    route_list.pop()
            stack.sort(key=lambda item: item[1])
            BFS(G, goal, stack, result, nodes)


def checkSmallerThanResult(stack, result):
    if (not result):
        return True
    else:
        for item in stack:
            if item[1] < result[0][1]:
                return True
        return False

./tucil3.ipynb
import matplotlib.pyplot as plt
import networkx as nx
from FileReader import create_graph
from AStar import AStar

# Create Graph
G = nx.Graph()
create_graph(G)
nodes = nx.get_node_attributes(G, 'pos')
colors = list(nx.get_edge_attributes(G,'color').values())
n = len(G)
nodes_list = list(nodes)

# Show map
print('Welcome to AStar Path Finder')
print('Maps:')
nx.draw_networkx(G, pos=nodes, with_labels=True, font_size=7, edge_color=colors)
plt.show()

# Gettong input
print('Select start and goal position [1-%d]:' %n)
i = 1
for item in G:
    print('%d. %s' % (i, item))
    i+=1

while True:
    try:
        a = int(input('choose start: '))
        b = int(input('choose goal: '))
        if (a < 1 or b < 1 or a > n or b > n):
            raise 'invalid input'
        break
    except:
        print('Masukkan input angka yang valid')

s = nodes_list[a-1]
g = nodes_list[b-1]

print('\nStart from %s to %s' % (s, g))

# AStar and create result
result = AStar(G, s, g)
result_path = result[0][0].split('-')
attrs = {}
for i in range(len(result_path)-1):
    attrs[(result_path[i],result_path[i+1])] = {'color':'r'}
```

```
nx.set_edge_attributes(G, attrs)
colors = list(nx.get_edge_attributes(G,'color').values())

# Show result in map
nx.draw_networkx(G, pos=nodes, with_labels=True, font_size=7, edge_color=colors)
plt.show()
print('Path     :', result[0][0])
print('Distance : %.2f m' % (result[0][1]))
```

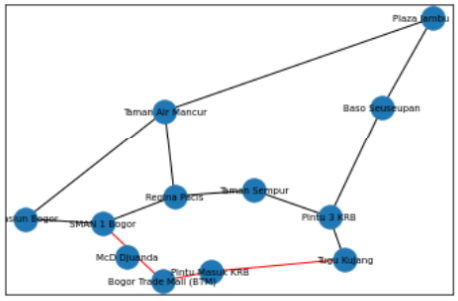
Screenshot

- Test case 1

```
Welcome to AStar Path Finder
Maps:
```



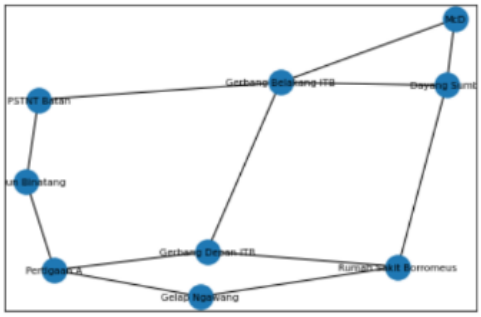

```
Select start and goal position [1-12]:
1. Pintu 3 KRB
2. Tugu Kujang
3. Pintu Masuk KRB
4. Bogor Trade Mall (BTM)
5. McD Djuanda
6. SMAN 1 Bogor
7. Regina Pacis
8. Taman Sempur
9. Taman Air Mancur
10. Plaza Jambu Dua
11. Baso Seuseupan
12. Stasiun Bogor
choose start: 2
choose goal: 6

Start from Tugu Kujang to SMAN 1 Bogor
```



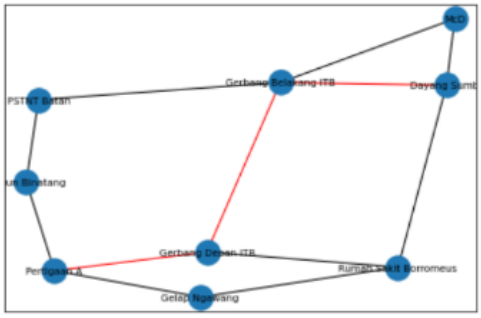

```
Path     : Tugu Kujang-Pintu Masuk KRB-Bogor Trade Mall (BTM)-SMAN 1 Bogor
Distance : 1703.32 m
```

- Test case 2

```
Welcome to AStar Path Finder
Maps:
```




```
Select start and goal position [1-9]:
1. Rumah Sakit Borromeus
2. Gerbang Depan ITB
3. Pertigaan A
4. Kebun Binatang
5. Gerbang Belakang ITB
6. PSTNT Batan
7. Gelap Ngawang
8. Dayang Sumbi
9. McD
choose start: 3
choose goal: 8

Start from Pertigaan A to Dayang Sumbi
```




```
Path     : Pertigaan A-Gerbang Depan ITB-Gerbang Belakang ITB-Dayang Sumbi
Distance : 1025.88 m
```

Penilaian

No	Penilaian	Ketercapaian
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

Source code juga dapat dilihat pada tautan berikut:
https://github.com/rifkymuth/Tucil3_13519123