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

Graphical user interface, application

Description automatically generated

* Test case 2

Graphical user interface, application

Description automatically generated

# 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>