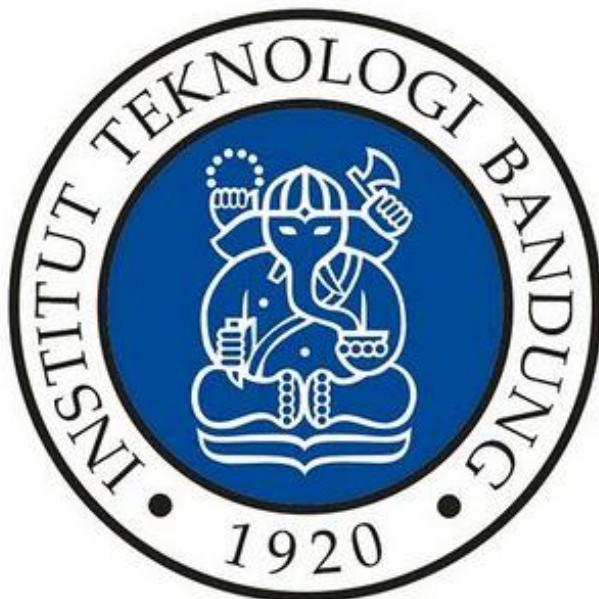


IF2211 Strategi Algoritma

Implementasi Algoritma A untuk Menentukan Lintasan Terpendek*



Disusun oleh:

13519104 Nabelanita Utami

13519207 Rafidika Samekto

**PROGRAM STUDI SARJANA INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2021**

1. Kode Program

Pada tugas kecil ini, kami membuat beberapa modul untuk menunjang pembangunan aplikasi dengan menggunakan algoritma A*.

1.1. Modul Input

Modul ini menangani input file eksternal hingga pembuatan struktur data yang dibutuhkan

```
from Utility import *
import os

def inputToAdj(filename):
    # Format text input
    # 0 1 0 2 -> dalam km, float
    # 1 0 3 4
    # 0 3 0 5
    # 2 4 5 0

    # Hasilnya dictionary dengan array isinya tuple
    # nama simpul tetang dan jaraknya (for now)
    # ie. {A: [(B, 10), (C, 17)]}

    # file = open(filename, "r")
    dir_path = os.path.dirname(os.path.realpath(__file__))
    lines = os.path.join(dir_path, "../test/" + filename)
    file = open(lines)
    N = int(file.readline())
    result = {}
    nodes = []

    for i in range(N):
        line = file.readline().split(' ')
        result.update({line[0]: []})
        nodes.append(line[0])

    for i in range(N):
        curr = nodes[i]
        row = file.readline().split(' ')
        for j in range(N):
            if (float(row[j]) > 0):
                result[curr].append((nodes[j], float(row[j])))

    file.close

    return result


def inputToCoor(filename):
    # Format text input
    # A (x, y)
    # B (x, y)
    # etc

    # Hasil output: dictionary simpul dengan koordinatnya
    # ie. {'A': (latitude, longitude), 'B': (latitude, longitude)}
    # -> ln lt dalam degree
```

```

dir_path = os.path.dirname(os.path.realpath(__file__))
if os.name == "nt" :
    lines = os.path.join(dir_path, "..\\test\\" + filename)
else:
    lines = os.path.join(dir_path, "../test/" + filename)
file = open(lines)
N = int(file.readline())
result = {}

for i in range(N):
    line = file.readline().replace('(', ' ').replace(',', ', ')
    .replace(')', ' ').split()
    result.update({line[0]: (float(line[1]), float(line[2]))})

file.close()

return result

```

1.2. Modul Utility

Modul ini menangani fungsi penunjang yang berguna untuk membantu pembuatan algoritma

```

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

def findDistanceTo(coorNode, target):
    targetCoor = coorNode[target]
    result = {}

    for key in coorNode:
        if (key != target):
            result.update({key: findDistanceTwoNodes(coorNode,
key, target)})

    return result

def findDistanceTwoNodes(coorNode, source, target):
    srcCoor = coorNode[source]
    trgCoor = coorNode[target]
    return haversine(srcCoor[0], srcCoor[1], trgCoor[0],
trgCoor[1])

def haversine(ltA, lnA, ltB, lnB):
    lnA, ltA, lnB, ltB = map(radians, [lnA, ltA, lnB, ltB])

    # rumus haversine
    # 6317 = jari2 bumi dlm km
    return 6371.0088 * 2.0 * asin(sqrt(sin((ltB - ltA)/2.0)**2.0 +
cos(ltA) * cos(ltB) * sin((lnB - lnA)/2.0)**2.0))

```

1.3. Modul Algoritma

Modul ini berisi algoritma utama yang digunakan dalam aplikasi

```

from Input import *

def AStar(simpulAwal, simpulTujuan, adjDict, listOfDistance):
    # adjDict adalah dictionary hasil return fungsi inputToAdj
    # listOfDistance adalah return fungsi inputToCoor
    tampunganSimpulTetangga = [{simpulAwal: []}]
    # print(tampunganSimpulTetangga)
    listSimpulTetangga = []
    # for tup in adjDict[simpulAwal]:
    #     listSimpulTetangga.append(tup[0])
    # print(listSimpulTetangga)
    visited = {}
    for key in listOfDistance:
        visited[key] = False
    # print(visited)
    result =
evaluateSimpul(simpulAwal, simpulAwal, listSimpulTetangga, simpulTuju
an, tampunganSimpulTetangga, visited, adjDict, listOfDistance)
    return result

def evaluateSimpul(simpulAwal, simpulSekarang, listSimpulTetangga,
simpulTujuan, tampunganSimpulTetangga, visited, adjDict,
listOfDistance):
    tempListSimpulTetangga = list(listSimpulTetangga)
    tempVisited = visited
    tempTampunganSimpulTetangga = list(tampunganSimpulTetangga)
    if (simpulSekarang == simpulTujuan):
        listReturn = list(listSimpulTetangga)
        listReturn.append(simpulSekarang)
        cost = getJarakSimpulXKeSimpulAwal(listReturn, adjDict)
        formattedCost = "{:.2f}".format(cost)
        # print(cost)
        listReturn.append(formattedCost)
        return listReturn
    else:
        # print(simpulSekarang)
        listTampungan = list(tempListSimpulTetangga)
        listTampungan.append(simpulSekarang)
        # print(listTampungan)
        # for dictionary in tampunganSimpulTetangga:
        #     if simpulSekarang in dictionary:
        #         listTampungan = list(dictionary[simpulSekarang])
        #         listTampungan.append(simpulSekarang)

        for key in adjDict:
            if (key == simpulSekarang):
                for tup in adjDict[key]:
                    if not (tempVisited[tup[0]]):
                        calonKey = tup[0]
                        calonDict = {calonKey: listTampungan}

tempTampunganSimpulTetangga.append(calonDict)
    # print("Ini tampunganSimpulTetangga yang udah ditambah
sama tetangga baru")
    # print(tempTampunganSimpulTetangga)

    for dictionary in tempTampunganSimpulTetangga:
        if (simpulSekarang in dictionary):

```

```

    if
(isTwoListsEqual(tempListSimpulTetangga,dictionary[simpulSekarang])
):
    tempVisited[simpulSekarang] = True
    tempTampunganSimpulTetangga.remove(dictionary)
    break
# if not(visited[simpulAwal]):
#     visited[simpulAwal] = True
# print("visited yang baru:")
# print(tempVisited)
# print("tampunganSimpulTetangga hasil udah remove simpul
sekarang")
# print(tempTampunganSimpulTetangga)

nilaiTerkecil = 0.0
simpulWithNilaiTerkecil = "X"
pathSimpulWithNilaiTerkecil = []
for dictionary in tempTampunganSimpulTetangga:
    for key in dictionary:
        listSementara = list(dictionary[key])
        listSementara.append(key)
        jarakSementara =
getJarakSimpulXKeSimpulAwal(listSementara,adjDict) +
getJarakLurusSimpulXKeSimpulTujuan(listOfDistance,simpulSekarang,s
impulTujuan)
        # print(jarakSementara)
        if (nilaiTerkecil == 0.0):
            nilaiTerkecil = jarakSementara
            simpulWithNilaiTerkecil = key
            pathSimpulWithNilaiTerkecil =
list(dictionary[key])
        else:
            if (jarakSementara < nilaiTerkecil):
                nilaiTerkecil = jarakSementara
                simpulWithNilaiTerkecil = key
                pathSimpulWithNilaiTerkecil =
list(dictionary[key])
            # print("nilai terkecil: ")
            # print(nilaiTerkecil)
            # print("Simpul yang bakal dikunjungin selanjutnya: ")
            # print(simpulWithNilaiTerkecil)
            # print("Path simpul yang mau dikunjungin: ")
            # print(pathSimpulWithNilaiTerkecil)

            # for dictionary in tampunganSimpulTetangga:
            #     if simpulWithNilaiTerkecil in dictionary:
            #         listTampungan =
list(dictionary[simpulWithNilaiTerkecil])
            #         # listTampungan.append(simpulSekarang)
            #         print("listTampungan baru:")
            #         print(path)
            return
evaluateSimpul(simpulAwal,simpulWithNilaiTerkecil,pathSimpulWithNi
laiTerkecil,simpulTujuan,tempTampunganSimpulTetangga,tempVisited,a
djDict,listOfDistance)

def getJarakSimpulXKeSimpulAwal(listUrutanJalan, adjDict):
    jarak = 0.0
    for i in range(len(listUrutanJalan)-1):

```

```

        for key in adjDict:
            if (key == listUrutanJalan[i]):
                for tup in adjDict[key]:
                    if (tup[0] == listUrutanJalan[i+1]):
                        jarak += tup[1]
                        # print(jarak)
        return jarak

def getJarakLurusSimpulXKeSimpulTujuan(listOfDistance, X,
simpulTujuan):
    # X adalah simpul saat ini
    # listOfDistance adalah return dari fungsi inputToCoor
    dictJarakLurus = findDistanceTo(listOfDistance,simpulTujuan)
    jarakLurus = 0.0
    for key in dictJarakLurus:
        if (key == X):
            jarakLurus = dictJarakLurus[key]
    return jarakLurus

def isTwoListsEqual(list1, list2):
    equal = True
    for element1 in list1:
        if not (element1 in list2):
            equal = False
            break
    if not (equal):
        return equal
    else:
        for element2 in list2:
            if not (element2 in list1):
                equal = False
                break
    return equal

```

1.4. Modul App (app.py)

Modul ini merupakan backend flask dari website yang menjadi platform dijalankannya program. Pada modul ini terdapat server yang akan dikunjungi oleh program selama digunakan.

```

from flask import *
from Algoritma import *

app = Flask(__name__)

file = []
listAdj = []
listCoor = []

@app.route('/search', methods=['POST', 'GET'])
def search():
    file.clear()

```

```

listAdj.clear()
listCoor.clear()

file.append(request.data.decode(encoding="utf-8"))

data = file[0]
listAdj.append(inputToAdjWeb(data))
listCoor.append(inputToCoorWeb(data))
node = findAllNode(listAdj[0])

return render_template('nodes.html', nodeList=node)

@app.route('/result', methods=['POST'])
def result():
    origin = request.form['origin']
    goal = request.form['goal']
    hasil = AStar(origin, goal, listAdj[0], listCoor[0])
    dist = hasil[len(hasil)-1]
    hasilLatLng = convertToLatLng(hasil, listCoor[0])
    temp = findAllNode(listAdj[0])
    temp.append("X")
    temp = convertToLatLng(findAllNode(temp), listCoor[0])

    name = findAllNode(listAdj[0])
    coor = convertAdjToLatLng(listAdj[0], listCoor[0])

    nodeNames = json.dumps(name)
    node = json.dumps([{"lat": n[0], "lng": n[1]} for n in temp])
    res = json.dumps([{"lat": hasil[0], "lng": hasil[1]} for hasil
in hasilLatLng])
    adjCoor = json.dumps(coor)
    return render_template('result.html', res=res, node=node,
dist=dist, nodeNames=nodeNames, adjCoor=adjCoor)

@app.route('/')
def index():
    return render_template('index.html')

if __name__ == "__main__":
    app.run(debug=True)

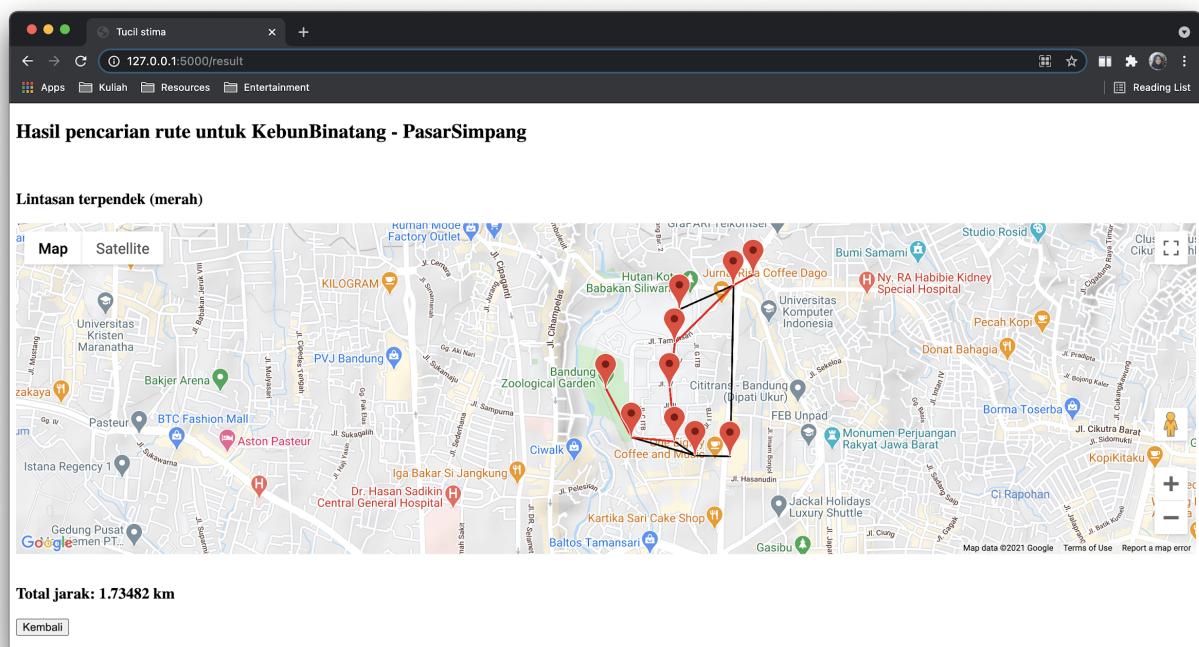
```

2. Hasil Pengujian

2.1. Uji coba 1 - Peta sekitar kampus ITB

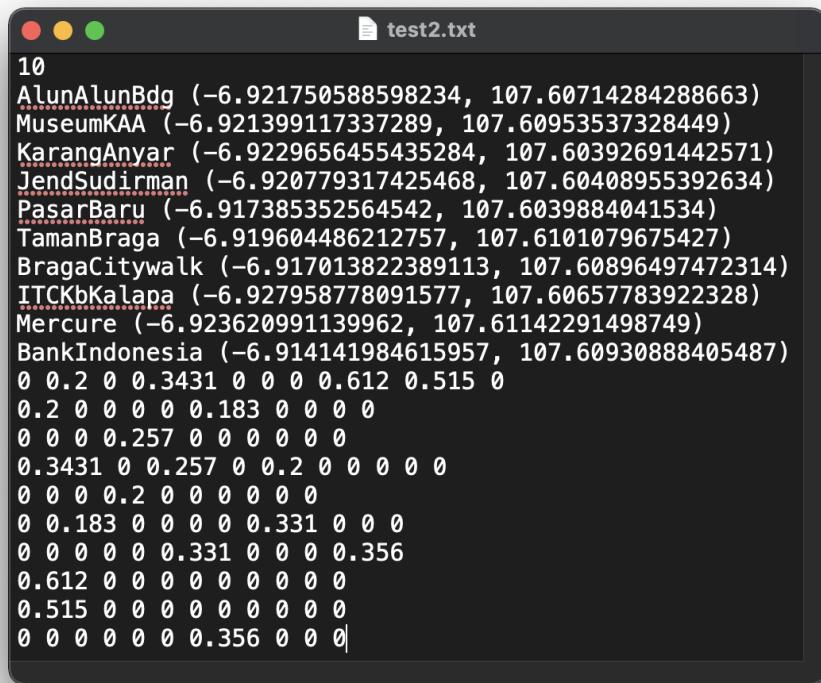
```
test1.txt — Edited
10
GerbangDepan (-6.8929179213398095, 107.61040160487973)
GerbangBelakang (-6.8878487983016745, 107.6104161919234)
LabtekV (-6.890116428449828, 107.61014172977413)
Salman (-6.893602629671023, 107.61145050553453)
Borromeus (-6.893657530272571, 107.61328463494524)
McdDago (-6.884864202804152, 107.61344131943518)
Saraga (-6.886090337319542, 107.61064864996013)
PasarSimpang (-6.884296885713012, 107.61449202679331)
TamanSari (-6.892671403553275, 107.60818253279623)
KebunBinatang (-6.890162179290764, 107.60682370664745)
0 0 0.29700 0.3933 0 0 0 0 0.24662 0
0 0 0.2588 0 0 0.47275 0.19333 0 0 0
0.29700 0.2558 0 0 0 0 0 0 0 0 0
0.19333 0 0 0 0.24837 0 0 0 0.36043 0
0 0 0 0.24837 0 1.01 0 0 0 0
0 0.427275 0 0 1.01 0 0.3473 0.14515 0 0
0 0.1923 0 0 0 0.3493 0 0 0 0
0 0 0 0 0.14514 0 0 0 0
0.24662 0 0 0.36043 0 0 0 0 0 0.3175
0 0 0 0 0 0 0 0 0.3175 0
```

Gambar 2.1.1 File uji 1 - Peta sekitar kampus ITB



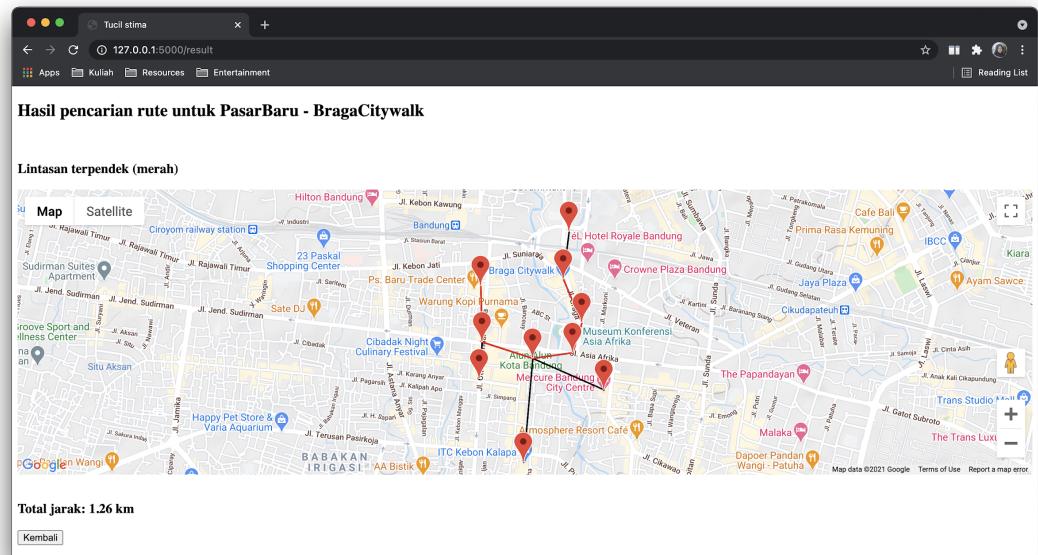
Gambar 2.1.2 Peta hasil pencarian rute KebunBinatang - PasarSimpang

2.2. Uji coba 2 - Peta sekitar Alun-Alun Bandung



```
10
AlunAlunBdg (-6.921750588598234, 107.60714284288663)
MuseumKAA (-6.921399117337289, 107.60953537328449)
KarangAnyar (-6.9229656455435284, 107.60392691442571)
JendSudirman (-6.920779317425468, 107.60408955392634)
PasarBaru (-6.917385352564542, 107.6039884041534)
TamanBraga (-6.919604486212757, 107.6101079675427)
BragaCitywalk (-6.917013822389113, 107.60896497472314)
ITCKbKalapa (-6.927958778091577, 107.60657783922328)
Mercure (-6.923620991139962, 107.61142291498749)
BankIndonesia (-6.914141984615957, 107.60930888405487)
0 0.2 0 0.3431 0 0 0 0.612 0.515 0
0.2 0 0 0 0.183 0 0 0 0
0 0 0 0.257 0 0 0 0 0 0
0.3431 0 0.257 0 0.2 0 0 0 0 0
0 0 0 0.2 0 0 0 0 0 0
0 0.183 0 0 0 0 0.331 0 0 0
0 0 0 0 0.331 0 0 0 0.356
0.612 0 0 0 0 0 0 0 0 0
0.515 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0.356 0 0 0|
```

Gambar 2.2.1 File uji 2 - Peta sekitar Alun-Alun Bandung

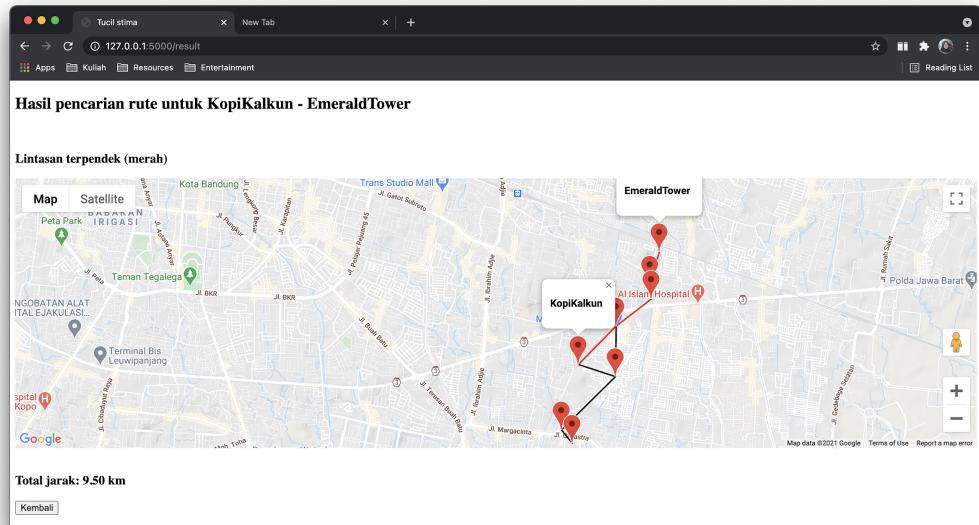


Gambar 2.2.2 Peta hasil pencarian rute PasarBaru - BragaCityWalk

2.3. Uji coba 3 - Peta sekitar Jalan Buah Batu

```
8
MetroIndahMall (-6.941697224134178, 107.65883213726502)
BormaMargacinta (-6.954830299246624, 107.65186570330314)
TotalKedaiKopi (-6.95647313995252, 107.65321849985241)
SushiMargahayu (-6.948029078230468, 107.65868637342342)
EmeraldTower (-6.932263667470207, 107.66433091460414)
KanaKawaluyan (-6.936415966128023, 107.66305541270974)
KopiKalkun (-6.946526567314817, 107.65395808300423)
BaliWorld (-6.938296591592579, 107.66329925867379)
0 0 0 0.95 0 0 1.0 4.6
0 0 0.28 2.80 0 0 0 0
0 0.28 0 0 0 0 0 0
0.95 2.80 0 0 0 0 0.75 0
0 0 0 0 0.5 0 0
0 0 0 0 0.5 0 0 3.40
1.0 0 0 0.75 0 0 0 0
4.6 0 0 0 0 3.40 0 0|
```

Gambar 2.3.1 File uji 3 - peta sekitar Jalan Buah Batu

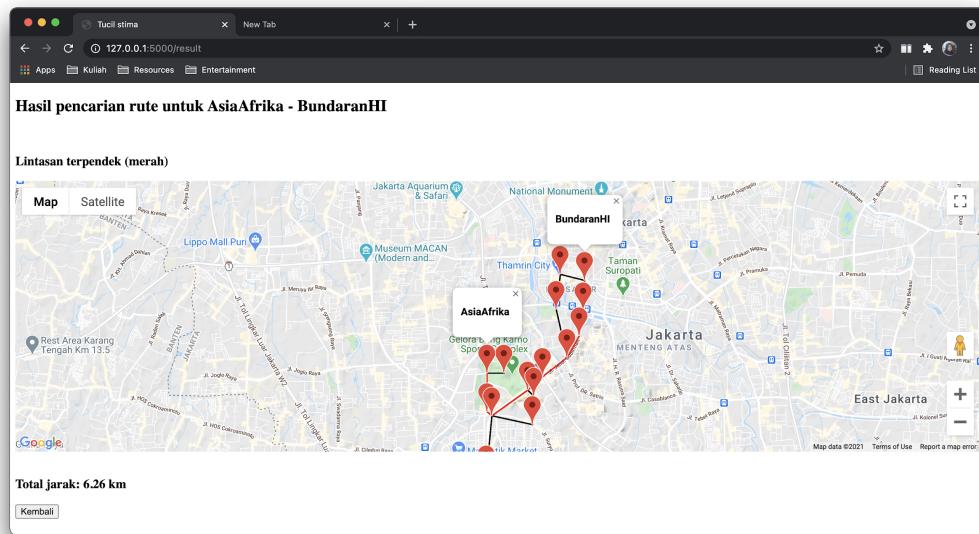


Gambar 2.3.2 Peta hasil pencarian rute KopiKalkun - EmeraldTower

2.4. Uji coba 4 - Peta Kompleks Gelora Bung Karno

```
15
GBK (-6.218574900537932, 106.80251739989217)
AsiaAfrika (-6.218563883337373, 106.79818511177022)
SenayanCity (-6.228303755354513, 106.79822215380817)
PemudaMembangun (-6.229279573780161, 106.79940749816892)
BlokM (-6.243947862013003, 106.79810368664364)
Senopati (-6.2315994367988, 106.80977926145765)
BursaEfek (-6.222743396863673, 106.80861243821562)
PacificPlace (-6.224418875311825, 106.81016820268908)
Semanggi (-6.219447659878315, 106.81242776554073)
StBenhil (-6.214622353043824, 106.81846171196122)
StSetiabudi (-6.20910001612952, 106.82166974766358)
StSudirman (-6.202827236529358, 106.82267595279515)
KaretBivak (-6.202545175887939, 106.81579858882525)
BundaranHI (-6.195221502224207, 106.82296770785364)
ThamrinCity (-6.193706787839241, 106.81680843289432)
0 0.40678 0 0 0 0 0 0 0 0 0 0 0 0
0.40678 0 1.14 0 0 0 0 0 0 0 0 0 0 0
0 1.14 0 0.15014 0 0 0 0 0 0 0 0 0 0
0 0 0.15014 0 0.0875 1.17 1.24 0 0 0 0 0 0 0
0 0 0 0.0875 0 0 0 0 0 0 0 0 0 0
0 0 0 1.17 0 0 0 0.77326 0 0 0 0 0 0
0 0 0 1.24 0 0 0 0.189583333 0.56672 0 0 0 0 0
0 0 0 0 0.77326 0.189583333 0 0 0 0 0 0 0
0 0 0 0 0 0.56672 0 0 0.77721 0 0 0 0 0
0 0 0 0 0 0 0.77721 0 0.8 0 1.43 0 0
0 0 0 0 0 0 0 0.8 0 0.74435 0 0 0
0 0 0 0 0 0 0 0 0.74435 0 0 0.84080 0
0 0 0 0 0 0 0 0 1.43 0 0 0 1.02
0 0 0 0 0 0 0 0 0 0 0 0 0.77320
0 0 0 0 0 0 0 0 0 0 0 1.02 0.77320 0
```

Gambar 2.4.1 File uji 4 - peta sekitar Kompleks Gelora Bung Karno



Gambar 2.4.2 Peta hasil pencarian rute AsiaAfrika - BundaranHI

3. Lampiran

3.1. Alamat kode

Kode program secara lengkap dapat diakses pada repository github dengan alamat <https://github.com/nabelanita/tucil-3-stima>

3.2. 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 Maps API dan menampilkan peta	✓