LAPORAN TUGAS KECIL

Implementasi Algoritma A* untuk Menentukan Lintasan Terpendek

Dibuat dalam rangka:
Tugas Kecil 2 IF-2211 Strategi Algoritma

Oleh:
Fabian Savero Diaz P. - 13519140
Fadel Ananda Dotty - 13519146

PROGRAM STUDI TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2020

I. Kode Program

Kode Program Algoritma A*

```
from math import radians, cos, sin, asin, sqrt
from prioqueue import PriorityQueue

def haversine(node1, node2):
    # convert decimal degrees to radians
    R = 6372.8 # Dalam km

    dLat = radians(node2.latitude - node1.latitude)
    dLon = radians(node2.longitude - node1.longitude)
    lat1 = radians(node1.latitude)
    lat2 = radians(node2.latitude)

    a = sin(dLat/2)**2 + cos(lat1)*cos(lat2)*sin(dLon/2)**2
    c = 2*asin(sqrt(a))

    return R * c
class Astar:
    def __init__(self, graph, start, goal):
        self.graph = graph
        self.start = start
        self.goal = goal
        self.frontier = PriorityQueue()
        self.came_from = {self.start: None}
        self.total_cost = {self.start: 0}

    def solve(self):
        self.start.f = 0 + haversine(self.start, self.goal)
        self.frontier.insert(self.start)
        while not self.frontier.empty():
            current = self.frontier.pop()
            if current == self.goal:
                break

            for neighbour in current.neighbour:
                new_cost = self.total_cost[current] +
haversine(current, neighbour)
                if neighbour not in self.total_cost or new_cost <
```

```
self.total_cost[neighbour]:
                    self.total_cost[neighbour] = new_cost
                    neighbour.f = new_cost + haversine(neighbour,
self.goal)
                    self.frontier.insert(neighbour)
                    self.came_from[neighbour] = current

        if (self.goal not in self.came_from):
            self.came_from = {}

        return self.came_from, self.total_cost

    def get_path(self):
        current = self.goal
        path = [current]
        while current != self.start:
            current = self.came_from[current]
            path.append(current)

        path.reverse()
        return path
```

Kode Program Kelas Graph

```
class Graph(object):
    def __init__(self, numOfNodes):
        self.numOfNodes = numOfNodes
        self.nodes = []

    def addNode(self, node):
        self.nodes.append(node)

    def printGraph(self):
        print("Daftar Node: ")
        for node in self.nodes:
            node.print()
            print()

    def addNeighbours(self, matrix):
```

```
        for i in range(self.numOfNodes):
            for j in range(self.numOfNodes):
                if matrix[i][j] != 0 :
                    self.nodes[i].addNeighbour(self.nodes[j])

    def findNode(self, name):
        for node in self.nodes:
            if (node.name == name):
                return node
class Node(object):
    def __init__(self, latitude, longitude, name):
        self.latitude = latitude
        self.longitude = longitude
        self.name = name
        self.neighbour = []
        self.f = 0

    def getLat(self):
        return self.latitude

    def getLong(self):
        return self.longitude

    def print(self):
        print("Name:", self.name)
        print("Latitude:", self.latitude)
        print("Longitude:", self.longitude)
        print("Neighbours: ", end="")
        for i in range(len(self.neighbour)):
            if i == len(self.neighbour) - 1:
                print(self.neighbour[i].name)
            else:
                print(self.neighbour[i].name, end=", ")

    def addNeighbour(self, node):
        self.neighbour.append(node)
```

Kode Main Program

```
def initializeGraph(f):
    fileArr = f.read().splitlines()
    numOfNodes = int(fileArr[0])
    nodeList = fileArr[1:numOfNodes+1]
    fileMatrix = fileArr[numOfNodes+1:]
    adjacencyMatrix = [ [ 0 for i in range(numOfNodes) ] for j in
range(numOfNodes) ]
    # Init graph
    graph = Graph(numOfNodes)

    # Tambahkan node dari file txt
    for i in range(numOfNodes):
        splitString = nodeList[i].split(" ", 2)
        newNode = Node(float(splitString[0]),
float(splitString[1]), splitString[2])
        graph.addNode(newNode)
        adjacencyMatrix[i] = [int(x) for x in
fileMatrix[i].split(" ")]

    graph.addNeighbours(adjacencyMatrix)

    return graph

def main(graph):
    # Input Lokasi
    print("Masukkan rute lokasi yang ingin dicari:")
    start = None
    goal = None

    while start == None or goal == None:
        start_node = input("Masukkan lokasi awal: ")
        start = graph.findNode(start_node)
        goal_node = input("Masukkan lokasi tujuan: ")
        goal = graph.findNode(goal_node)
        if (start == None or start == None):
            print("Masukkan tujuan lagi! Node tidak ditemukan")
        print()
```

```
    # Penghitungan path yang benar
    astar = Astar(graph, start, goal)
    came_from, total_cost = astar.solve()
    path = astar.get_path()
    distance = total_cost[goal]

    return start, goal, path, distance

def processPath(path):
    name = []
    latitude = []
    longitude = []
    for i in range(len(path)):
        name.append(path[i].name)
        latitude.append(path[i].latitude)
        longitude.append(path[i].longitude)

    dictname = dict(enumerate(name))
    dicts = []
    dicts.append(dictname)

    dictlat = dict(enumerate(latitude))
    latss = []
    latss.append(dictlat)

    dictlong = dict(enumerate(longitude))
    longss = []
    longss.append(dictlong)

    return dicts, latss, longss

def processGraph(graph):
    latitude = []
    longitude = []
    for i in range(len(graph.nodes)):
        latitude.append(graph.nodes[i].latitude)
        longitude.append(graph.nodes[i].longitude)

    return latitude, longitude
```

```
def processNeighbour(graph):
    listOfDict = []
    nodes = graph.nodes
    for i in range(len(nodes)):
        listOfDict.append({
            'lat': nodes[i].latitude,
            'lng': nodes[i].longitude,
            'neighbour': []
        })
        for j in range(len(nodes[i].neighbour)):
            listOfDict[i]['neighbour'].append({
                'lat': nodes[i].neighbour[j].latitude,
                'lng': nodes[i].neighbour[j].longitude
            })
    return listOfDict

# Main program
while True:
    filename = input("Masukkan filename: ")
    # Baca File
    try:
        f = open("test/" + filename)
    except FileNotFoundError:
        print("File not found")
    else:
        break
# Inisialiasi Graph
graph = initializeGraph(f)
graph.printGraph()

# Flask
@app.route("/get-data")
def getData():
    listOfDict = processNeighbour(graph)
    return jsonify(listOfDict = listOfDict)

@app.route("/")
def init():
```

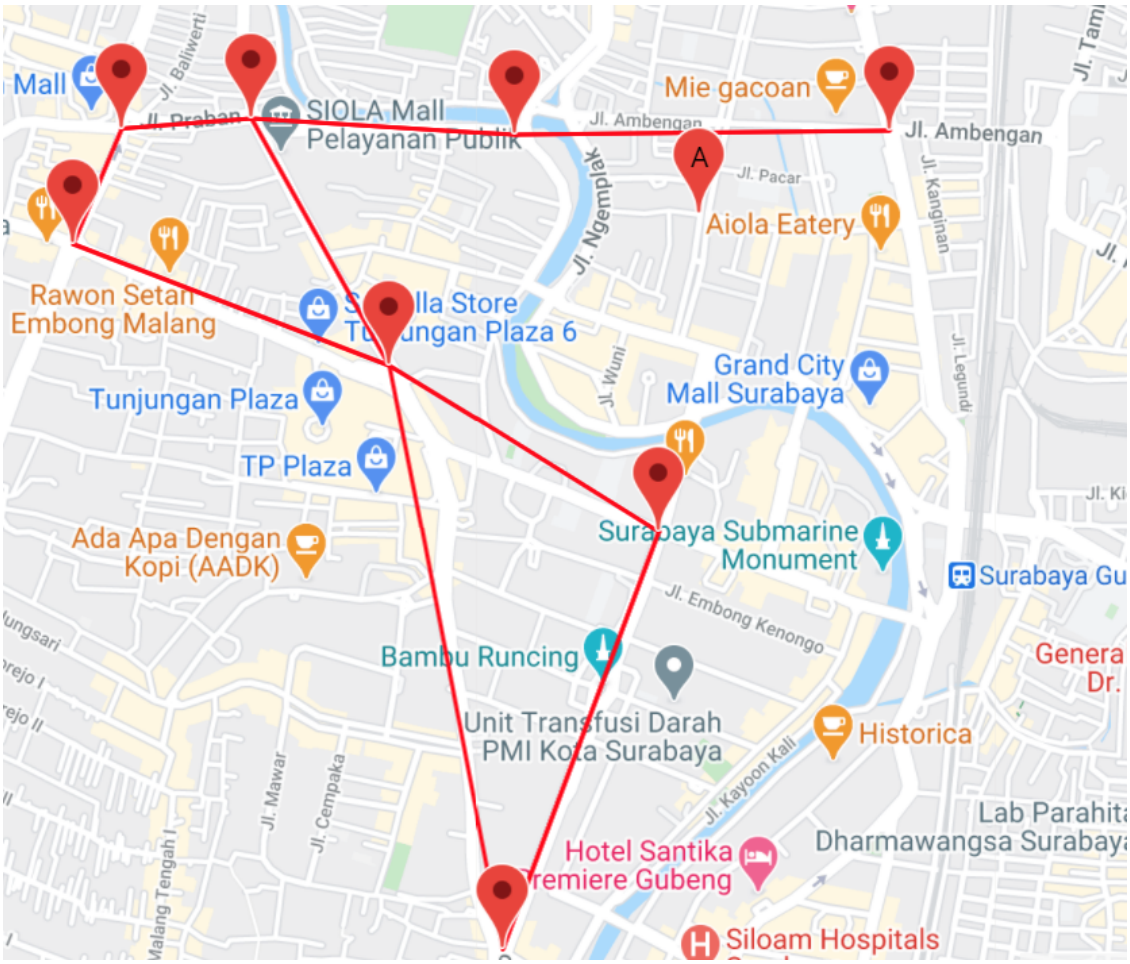
```
    # Solve Astar
    start, goal, path, distance = main(graph)
    dicts, latss, longss = processPath(path)
    marklat, marklong = processGraph(graph)
    solutionpath = json.dumps(dicts)
    arrayLat = json.dumps(latss)
    arrayLong = json.dumps(longss)
    startlat = start.latitude
    startlong = start.longitude
    destlat = goal.latitude
    destlong = goal.longitude
    sspath = path

    return render_template('map.html',
        slat=startlat,
        slong=startlong,
        dlat=destlat,
        dlong=destlong,
        spath=solutionpath,
        latss=arrayLat,
        longss=arrayLong,
        sspath=sspath,
        dist = distance,
        marklat = marklat,
        marklong = marklong)
```

II. Peta/Graf Input

Input File surabaya.txt

```
9
-7.258430454122149 112.73315378153994 Perempatan JW Marriot
-7.255995705794762 112.73419498545097 Perempatan Blauran
-7.255809862621428 112.73694437703911 Perempatan Jl. Tunjungan
-7.260969770069051 112.73985681377984 Belokan Embong Malang
-7.264469987209561 112.74554565386252 Bundaran Balai Pemuda
-7.273228857222445 112.74225190126364 Belokan Basra
-7.256125686005877 112.74252339902509 Belokan Genteng Kali
-7.256046352140946 112.75044600070753 SMAN 5 Surabaya
-7.257808530770473 112.74643307758659 Jimerto
0 1 0 1 0 1 0 0 0
1 0 1 0 0 0 0 0 0
```

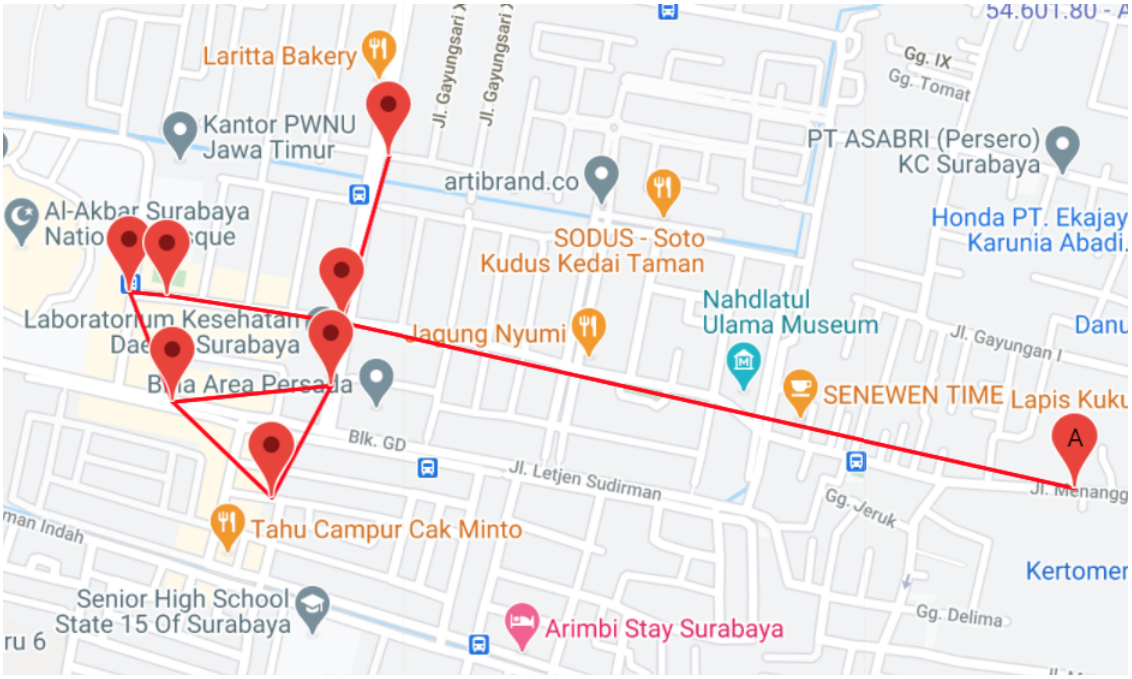
0 1 0 1 0 0 1 0 0
1 0 1 0 1 0 0 0 0
0 0 0 1 0 1 0 0 0
1 0 0 1 1 0 0 0 0
0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0



Input File tes1.txt (daerah Surabaya)

8
-7.338379 112.716986 Jl. Wisma Pagesangan
-7.338177 112.718859 TK Dunia Anak
-7.337086 112.716468 Masjid Al-Akbar
-7.337118 112.716919 SMP 22
-7.337437 112.718996 Jl. Gayungsari Barat
-7.335474 112.719547 Jl. Gayungsari 12
-7.339512 112.718156 Masjid Al-Wahyu
-7.339405 112.727731 Trans Icon

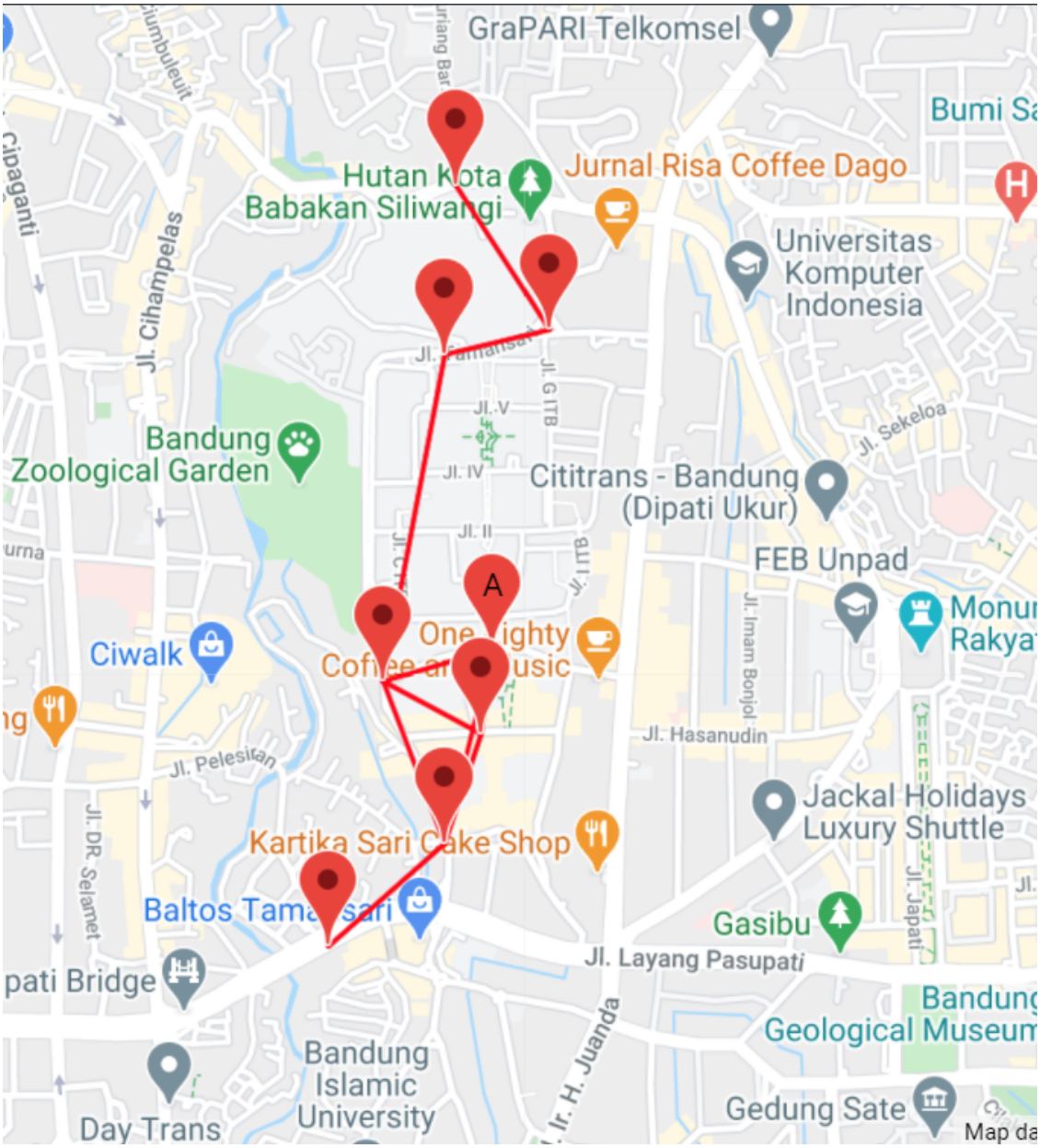
0 1 1 0 0 0 1 0
1 0 0 0 1 0 1 0
1 0 0 1 0 0 0 0
0 0 1 0 1 0 0 0
0 1 0 1 0 1 0 1
0 0 0 0 1 0 0 0
1 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0



Input File tes2.txt (daerah sekitar ITB)

8
-6.884588 107.609802 Jalan Siliwangi
-6.887740 107.609577 Jalan Tamansari
-6.887293 107.611563 Jalan Dayang Sumbi
-6.893854 107.608407 Kebun Binatang
-6.893258 107.610468 Jalan Ganesha
-6.896879 107.609588 Pelesiran
-6.898786 107.607387 Jl. Pasopati
-6.894802 107.610287 Gelap Nyawang
0 0 1 0 0 0 0 0
0 0 1 1 0 0 0 0
1 1 0 0 0 0 0 0
0 1 0 0 1 1 0 1
0 0 0 1 0 0 0 1
0 0 0 0 1 0 1 1
0 0 0 0 0 1 0 0

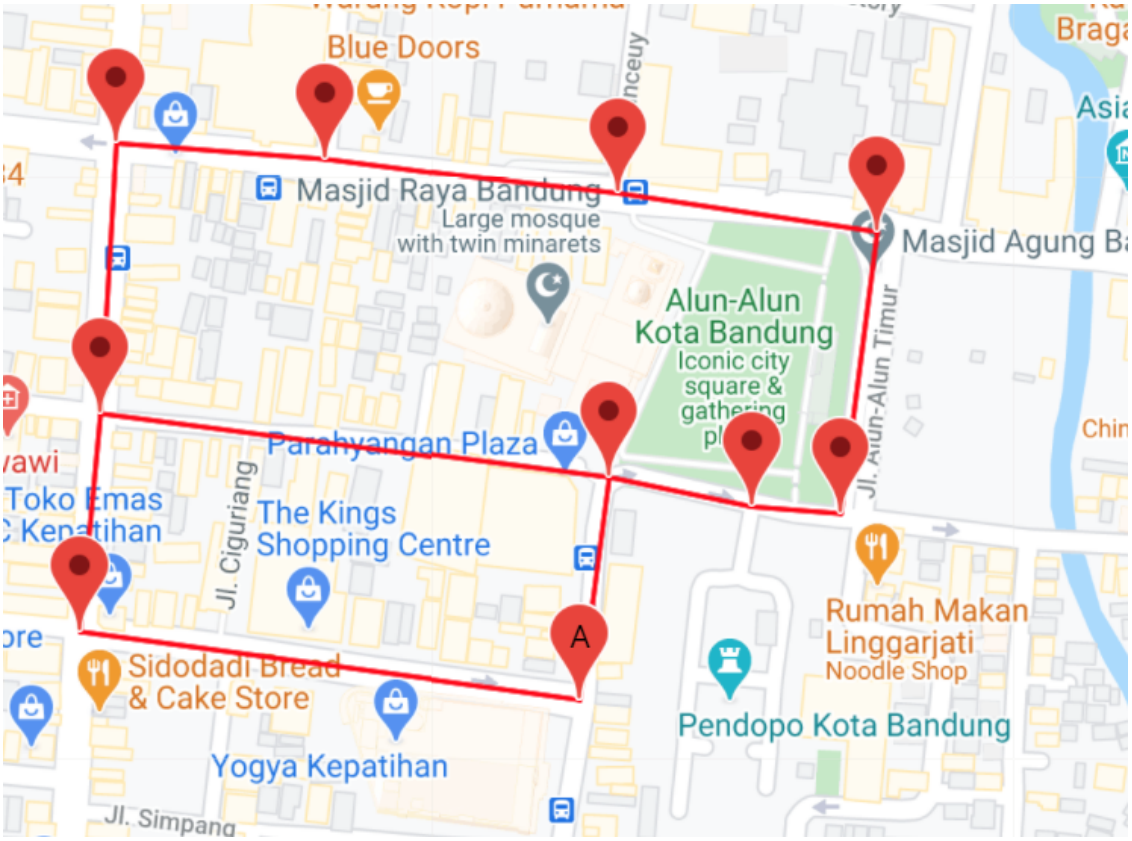
0 0 0 1 1 1 0 0



Input File tes3.txt (daerah sekitar Alun-Alun Bandung)

```
10
-6.920817 107.604100 JendSudirmanNavaro
-6.920893 107.605089 KontikiLintasMedia
-6.921052 107.606472 TokoKomputerNewPelangi
-6.921228 107.607690 MasjidAgung
-6.922083 107.604024 GarudaKencanaToko
-6.922383 107.606429 DalemKaum
```

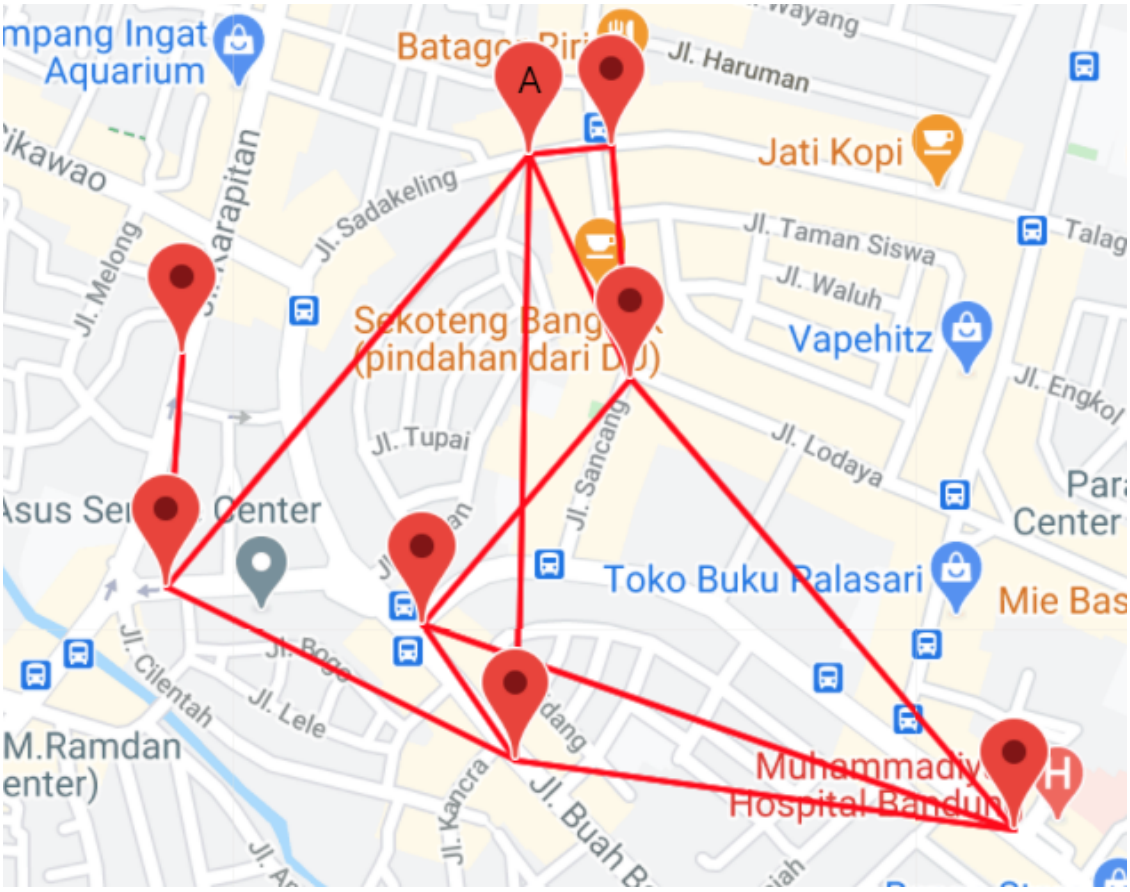
-6.922521 107.607099 PendopoBandung
-6.922551 107.607520 DalemKaum63
-6.923107 107.603929 OtistaKepatihan
-6.923423 107.606288 Balonggede
0 1 0 0 1 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 1 0 0
1 0 0 0 0 1 0 0 1 0
0 0 0 0 1 0 1 0 0 1
0 0 0 0 0 1 0 1 0 0
0 0 0 1 0 0 1 0 0 0
0 0 0 0 1 0 0 0 0 1
0 0 0 0 0 1 0 0 1 0



Input File tes4.txt (daerah sekitar Buahbatu)

8
-6.931831 107.618106 Jalan Banteng
-6.932928 107.618847 Jalan Buah Batu
-6.929882 107.619781 Jalan Lodaya
-6.928029 107.619619 Jalan Talaga Bodas
-6.931533 107.616035 Jalan Gurame

-6.929680 107.616164 Jalan Karapitan
-6.928093 107.618955 Jalan Sadakeling
-6.933482 107.622871 Jalan KH. Ahmad Dahlan
0 1 1 0 0 0 0 1
1 0 0 0 1 0 1 1
1 0 0 1 0 0 1 1
0 0 1 0 0 0 1 0
0 1 0 0 0 1 1 0
0 0 0 0 1 0 0 0
0 1 1 1 1 0 0 0
1 1 1 0 0 0 0 0



III. *Screenshot* Hasil Program

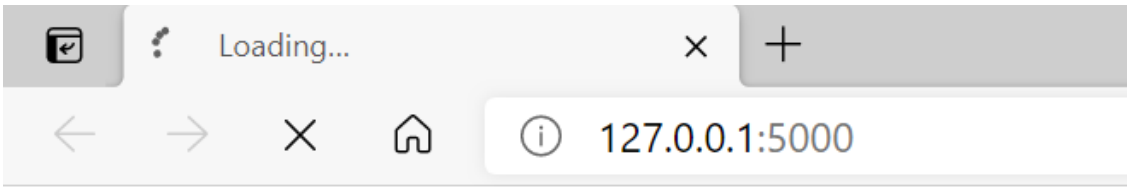
Menjalankan Program

```
TES@DESKTOP-M0MIBGF MINGW64 /h/ITB Semester 4/Strategi Algoritma/Tucil 3/TucilStima3 (main)
$ export FLASK_APP=src/main.py

TES@DESKTOP-M0MIBGF MINGW64 /h/ITB Semester 4/Strategi Algoritma/Tucil 3/TucilStima3 (main)
$ flask run
 * Serving Flask app "src/main.py"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
Masukkan filename: surabaya.txt
Daftar Node:
Name: Perempatan JW Marriot
Latitude: -7.258430454122149
Longitude: 112.73315378153994
Neighbours: Perempatan Blauran, Belokan Embong Malang, Belokan Basra

Name: Perempatan Blauran
Latitude: -7.255995705794762
Longitude: 112.73419498545097
Neighbours: Perempatan JW Marriot, Perempatan Jl. Tunjungan

Name: Perempatan Jl. Tunjungan
Latitude: -7.255809862621428
Longitude: 112.73694437703911
Neighbours: Perempatan Blauran, Belokan Embong Malang, Belokan Genteng Kali

Name: Belokan Embong Malang
Latitude: -7.260969770069051
Longitude: 112.73985681377984
Neighbours: Perempatan JW Marriot, Perempatan Jl. Tunjungan, Bundaran Balai Pemuda

Name: Bundaran Balai Pemuda
Latitude: -7.264469987209561
Longitude: 112.74554565386252
Neighbours: Belokan Embong Malang, Belokan Basra
```

```
Name: Belokan Embong Malang
Latitude: -7.260969770069051
Longitude: 112.73985681377984
Neighbours: Perempatan JW Marriot, Perempatan Jl. Tunjungan, Bundaran Balai Pemuda

Name: Bundaran Balai Pemuda
Latitude: -7.264469987209561
Longitude: 112.74554565386252
Neighbours: Belokan Embong Malang, Belokan Basra

Name: Belokan Basra
Latitude: -7.273228857222445
Longitude: 112.74225190126364
Neighbours: Perempatan JW Marriot, Belokan Embong Malang, Bundaran Balai Pemuda

Name: Belokan Genteng Kali
Latitude: -7.256125686005877
Longitude: 112.74252339902509
Neighbours: Perempatan Jl. Tunjungan

Name: SMAN 5 Surabaya
Latitude: -7.256046352140946
Longitude: 112.75044600070753
Neighbours: Belokan Genteng Kali

Name: Jimerto
Latitude: -7.257808530770473
Longitude: 112.74643307758659
Neighbours:
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Tampilan CLI Setelah Membuka http://127.0.0.1:5000/

```
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
Masukkan rute lokasi yang ingin dicari:
Masukkan lokasi awal:
```

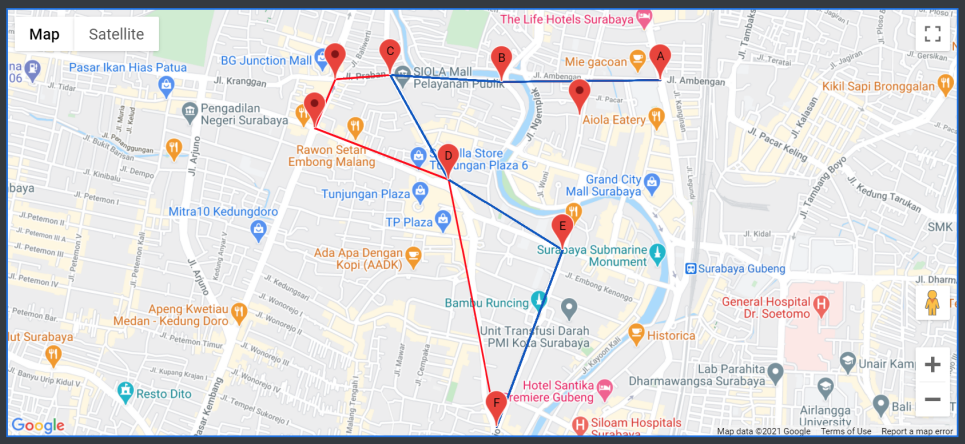


Tampilan Program Pada Web Setelah Memasukkan Input Node Asal dan Node Tujuan (untuk test case surabaya.txt)

```
Masukkan rute lokasi yang ingin dicari:
Masukkan lokasi awal: Bundaran Balai Pemuda
Masukkan lokasi tujuan: Perempatan JW Marriot

127.0.0.1 - - [06/Apr/2021 23:22:45] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [06/Apr/2021 23:22:45] "GET /get-data HTTP/1.1" 200 -
```

Jarak antara SMAN 5 Surabaya dengan Belokan Basra

Visualization



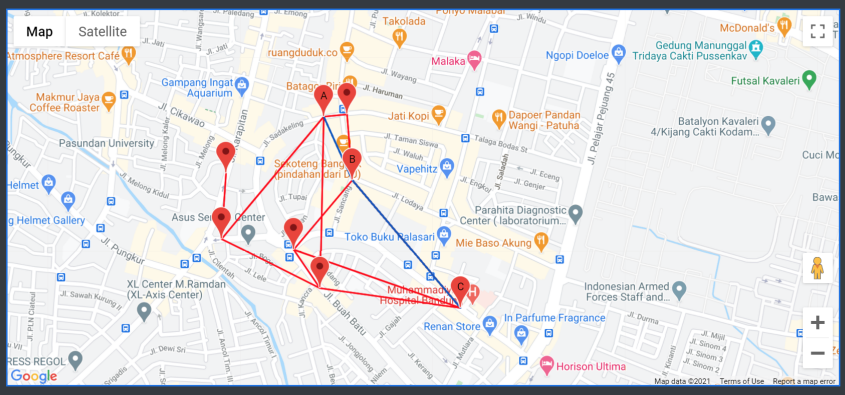
Solution:
SMAN 5 Surabaya
Belokan Genteng Kali
Perempatan Jl. Tunjungan
Belokan Embong Malang
Bundaran Balai Pemuda
Belokan Basra

Jarak antara simpul asal dan tujuan:
3.9269090912503146 km

Tampilan Program Pada Web Setelah Memasukkan Input Node Asal dan Node Tujuan (untuk test case test4.txt)

Jarak antara Jalan Sadakeling dengan Jalan KH. Ahmad Dahlan

Visualization

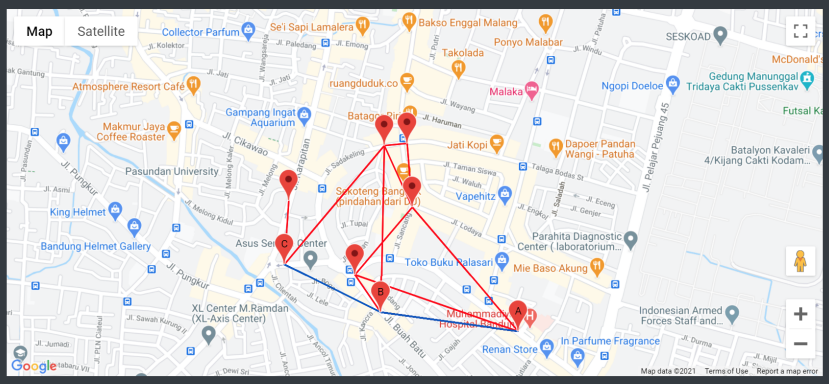


Solution:
Jalan Sadakeling
Jalan Lodaya
Jalan KH. Ahmad Dahlan

Jarak antara simpul asal dan tujuan:
0.7449441270427658 km

Jarak antara Jalan KH. Ahmad Dahlan dengan Jalan Gurame

Visualization



Solution:
Jalan KH. Ahmad Dahlan
Jalan Buah Batu
Jalan Gurame

Jarak antara simpul asal dan tujuan:
0.7956480807854303 km

IV. Link

Repository tugas kecil 3 dapat diakses pada pranala di bawah ini
https://github.com/fabiansdp/TucilStima3

V. Checklist

Poin	Ya	Tidak
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	V	