Tugas Kecil 3 IF2122 Strategi Algoritma

Implementasi Algoritma A* untuk Menentukan Lintasan Terpendek

Dibuat dalam rangka:
Tugas Kecil 3 IF-2211 Strategi Algoritma

Oleh:

Ryandito Diandaru 13519157
M. Jafar Gundari 13519197

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2021

KODE PROGRAM

1. File Graph.py

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

class Simpul:
    def __init__(self, name, x, y, index):
        self.name = name
        self.x = x
        self.y = y
        self.index = index

    def getName(self):
        return self.name

    def getX(self):
        return self.x

    def getY(self):
        return self.y

    def getIndex(self):
        return self.index

class Graph:
    def __init__(self, path):
        #inisialisasi atribut
        self.simps = []
        self.adjMat = []

        #baca file
        file = open(path, "r")
        lines = file.readlines()
        ens = int(lines[0])
        for i in range(1,ens+1):
            info = lines[i].split()
            self.simps.append(Simpul(info[0], float(info[1]),
float(info[2]), i-1))

        for i in range(ens+1,len(lines)):
            a_list = lines[i].split()
            map_object = map(int, a_list)
            self.adjMat.append(list(map_object))

        for i in range(ens):
            for j in range(ens):
                if(i<j and self.adjMat[i][j] != 0):
```

```
                    hv = self.getDistHaversine(self.simps[i],
self.simps[j])
                    self.adjMat[i][j] = hv
                    self.adjMat[j][i] = hv


    def getDistHaversine(self, pointA, pointB):
        #jadikan dalam satuan radian
        loA = radians(pointA.getY())
        loB = radians(pointB.getY())
        laA = radians(pointA.getX())
        laB = radians(pointB.getX())

        #rumus haversine
        slon = loB - loA
        slat = laB - laA
        a = sin(slat / 2)**2 + cos(laA) * cos(laB) * sin(slon /
2)**2

        #return
        return(2 * asin(sqrt(a)) * 6371)


    def getSimps(self):
        return self.simps

    def getAdjMat(self):
        return self.adjMat

    def printSimps(self):
        for i in self.simps:
            print(i.getName() + " " + str(i.getX()) + " " +
str(i.getY()) + " Index : " + str(i.getIndex()))

    def printAdjMat(self):
        for i in self.adjMat:
            print(i)
```

2. File PrioQueue.py

```
import Graph
```

```
class Path:
    # CostsoFar = jarak sebenarnya
    # bobot = jarak sebenarnya + sld
    def __init__(self, arraySimpul, costSoFar, Bobot):
        self.arraySimps = []
        for item in arraySimpul:
            self.arraySimps.append(item)
        self.Bobot = Bobot
        self.costSoFar = costSoFar

    def insertNewSimp(self, newSimpul):
        self.arraySimps.append(newSimpul)

    def getCostSoFar(self):
        return self.costSoFar

    def getArraySimps(self):
        return self.arraySimps

    def getBobot(self):
        return self.Bobot

    def printPath(self):
        print("Jalur : ")
        for i in range (len(self.arraySimps)):
            if(i == len(self.arraySimps)-1):
                print(self.arraySimps[i].getName())
            else:
                print(self.arraySimps[i].getName(), end=" → ")
        print("Jarak tempuh : ", round(self.Bobot*1000,2), "m")

class PrioQueue:
    def __init__(self):
        self.Queue = [] # Berisi tipe Path

    def enqueue(self, Path):
        i = 0
        while(i < len(self.Queue) and Path.Bobot >
self.Queue[i].Bobot):
            i += 1
        if(i == len(self.Queue)):
            self.Queue.append(Path)
        else:
            self.Queue.insert(i, Path)

    def Top(self):
        return self.Queue[0]
```

```
    def isEmpty(self):
        return (len(self.Queue) == 0)

    def dequeue(self):
        if(self.isEmpty()):
            return 404 #error code
        else:
            return self.Queue.pop(0)

    def printQueue(self):
        if(self.isEmpty()):
            print("Priority queue is empty")
        else:
            for jalur in self.Queue:
                jalur.printPath()
```

3. File Function.py

```
from Graph import Graph, Simpul
from Prioqueue import PrioQueue, Path

# Mengecek apakah pada suatu path sudah terdapatt simpul yang
dicari
def dikunjungi(path, indexDicari):
    for item in path.getArraySimps():
        if(item.getIndex() == indexDicari):
            return True
    return False

# Mengambil simpul berdasarkan index
def getSimpulFromIndex(index, ArraySimps):
    for item in ArraySimps:
        if(item.getIndex() == index):
            return item
    return -1 # kalau ga ditemukan

def getSimpulbyName(Name, ArraySimps):
    for item in ArraySimps:
        if(item.getName() == Name):
            return item
    return -1 # error code

# Find path using A* algorithm
# Source dan destination bertipe Simpul
```

```
# Mengembalikan array 1 elemeen bertipe Path
def findPath(source, destination, Graf):
    FinalPath = []
    # inisialisasi sebuah priority queue
    Queue = PrioQueue()
    jalur = Path([source], 0, Graf.getDistHaversine(source,
destination))
    Queue.enqueue(jalur)
    found = False # Untuk mendeteksi apakah sudah menemukan
Simpul tujuan
    i = 1
    while(not Queue.isEmpty()):
        jalur = Queue.dequeue()
        IndexLast = len(jalur.getArraySimps()) -1
        lastVertex = jalur.getArraySimps()[IndexLast]
        index = lastVertex.getIndex()
        arraySimpulBaru = []

        for i in range(len(Graf.simps)):
            # print(str(i) + "-------")
            if(Graf.adjMat[index][i] > 0 and not
dikunjungi(jalur, i)):
                # Kumpulkan jalur sebelumnya + simpul baru yang
dibangkitkan
                arraySimpulBaru = jalur.getArraySimps()
                newSimpul = getSimpulFromIndex(i,
Graf.getSimps())
                arraySimpulBaru.append(newSimpul)

                # Siapkan atribut new Path
                newCost =
jalur.getCostSoFar()+Graf.adjMat[index][i] # g(n)
                sld = Graf.getDistHaversine(newSimpul,
destination) #h(n)
                newPath = Path(arraySimpulBaru,newCost,
newCost+sld)
                Queue.enqueue(newPath)
                # Cek apakah sudah sampai ke destination
                if(newSimpul.getIndex() ==
destination.getIndex()):
                    found = True
                    if(len(FinalPath) == 0):
                        FinalPath.append(newPath)
                    else:
                        if(newPath.getBobot() >
FinalPath[0].getBobot()):
                            FinalPath[0] = newPath
```

```
                # Hapus simpul baru untuk disiapkan menjadi
simpul yang lain
                arraySimpulBaru.pop()
        # Sudah tidak ada jalur yang lebih pendek dari final path
sekarang
        if(found and (Queue.isEmpty() or FinalPath[0].getBobot()
<= Queue.Top().getBobot())):
            FinalPath[0].printPath()
            return FinalPath
    # Jika tidak ditemukan, FinalPath kosong
    return FinalPath
```

4. File run.py

```
#untuk map
import folium
from folium import plugins

#untuk rute
from Graph import Graph, Simpul
from Prioqueue import PrioQueue, Path
from function import *

#untuk widget tool
import ipywidgets as widgets

### MARKER AND PATH ROUTE ###
def createAllMarker(arraySimpul, m, warna, labelGroup): # f=
figure group, m = map
    f=folium.FeatureGroup(labelGroup)
    for simpul in arraySimpul:
        folium.Marker(location=[simpul.getX(),
simpul.getY()],popup=simpul.getName(),
                      icon = folium.Icon(color=warna),
                      tooltip=simpul.getName()).add_to(f)
    f.add_to(m)

# Draw All line
def drawPathfromGraph(listSimpul, adjMat, Map):
    f1=folium.FeatureGroup("All path")
    for i in range(len(adjMat)):
        for j in range(len(adjMat)):
            if(i<j and adjMat[i][j] > 0):
                garis = [[listSimpul[i].getX(),
listSimpul[i].getY()], [listSimpul[j].getX(),
```

```
listSimpul[j].getY()]]

line_1=folium.vector_layers.PolyLine(garis,popup="Jarak : " +
str(adjMat[i][j]) + " km",

tooltip=listSimpul[i].getName() + " - " +
listSimpul[j].getName(),

color='#4878b8',weight=7.5).add_to(f1)
    f1.add_to(Map)


# Draw final path
def drawFinalPath(Path, Map, arraySimpFromGraf, adjMat):
    f=folium.FeatureGroup("Final Path")
    for i in range (len(Path.getArraySimps())-1):
        indexA = Path.getArraySimps()[i].getIndex()
        simpA = getSimpulFromIndex(indexA, arraySimpFromGraf)
        indexB = Path.getArraySimps()[i+1].getIndex()
        simpB = getSimpulFromIndex(indexB, arraySimpFromGraf)
        garis = [[simpA.getX(), simpA.getY()], [simpB.getX(),
simpB.getY()]]
        line_1=folium.vector_layers.PolyLine(garis,
                                             popup = "Jarak : "
+ str(adjMat[indexA][indexB]) + " km",

tooltip=simpA.getName() + " - " + simpB.getName(),

color='#cf1b1b',weight=7.5).add_to(f)
    f.add_to(Map)


### WIDGETS ###
#input box text file
namaFile = widgets.Text(
    value='',
    placeholder='Input file name',
    description='File:',
    disabled=False
)

#dropdown asal
asal = widgets.Dropdown(
    options=['placeholder'],
    value='placeholder',
    description='From :',
    disabled=False
```

```
)

#dropdown tujuan
tujuan = widgets.Dropdown(
    options=['placeholder'],
    value='placeholder',
    description='To :',
    disabled=False
)

#go button
go = widgets.Button(
    description='Go',
    disabled=False,
    button_style='', # 'success', 'info', 'warning', 'danger' or
''
    tooltip='Click me',
    icon='' # (FontAwesome names without the `fa-` prefix)
)

#cancel button
cancel = widgets.Button(
    description='Reset',
    disabled=False,
    button_style='danger', # 'success', 'info', 'warning',
'danger' or ''
    tooltip='Click me',
    icon='' # (FontAwesome names without the `fa-` prefix)
)

#findroutebutton
find = widgets.Button(
    description='Find Route',
    disabled=False,
    button_style='success', # 'success', 'info', 'warning',
'danger' or ''
    tooltip='Click me',
    icon='' # (FontAwesome names without the `fa-` prefix)
)

#labels
title = widgets.VBox([widgets.Label('GOOGOLMEPZ')]
,layout=widgets.Layout(width='100%', display='flex' ,
align_items='center'))
initial_map = widgets.Label(value="Initial Map")

### GROUPING ###
```

```
upfile = [namaFile,go]
whereto = [asal,tujuan,find,cancel]

### OUTPUTS ###
outputFileSelect = widgets.Output()
outputSearch = widgets.Output()
outputCancel = widgets.Output()

### HANDLERS ###
def cuancel(event):
    with outputCancel:
        outputFileSelect.clear_output()
        outputSearch.clear_output()

def searchGo(event):
    with outputSearch:
        a = Graph(namaFile.value)
        listSimpul = a.getSimps()
        adjmat = a.getAdjMat()

        # A* algorithm
        try:
            outputSearch.clear_output()
            #mencari dari asal ke tujuan dengan A*
            source = getSimpulbyName(asal.value,listSimpul)
            destination =
getSimpulbyName(tujuan.value,listSimpul)
            if(source.getName() == destination.getName()):
                # jika asal dan tujuan di tempat yang sama
                print("0 m")
            else:
                p =
findPath(getSimpulbyName(asal.value,listSimpul),
getSimpulbyName(tujuan.value,listSimpul), a)
                #path hasil adalah p[0]
                finalP = p[0]

                #menggambar alur ke layar
                m=folium.Map(location=[listSimpul[0].getX(),
listSimpul[0].getY()], width='100%', height='100%',
zoom_start=15.5)
                createAllMarker(listSimpul, m, "cadetblue", "All
Vertex")
                drawPathfromGraph(listSimpul, adjmat, m)
                drawFinalPath(finalP, m, listSimpul, adjmat)
                createAllMarker(finalP.getArraySimps(), m,
"orange", "Visited")
```

```
                folium.LayerControl().add_to(m)
                #Add scroll Zoom toggler left bottom
                plugins.ScrollZoomToggler().add_to(m)
                # Add fullscreen button

plugins.Fullscreen(position='topright').add_to(m)

                #mencetak map ke layar
                display(m)
        except IndexError:
            # Jalur tidak ditemukan
            print("Jalur tidak ditemukan")

def fileSelect(event):
    with outputFileSelect:
        try:
            #coba loadfile jika file ada
            outputFileSelect.clear_output()
            a = Graph(namaFile.value)
            listSimpul = a.getSimps()
            adjmat = a.getAdjMat()

            #menggambar map awal
            m = folium.Map(location=[listSimpul[0].getX(),
listSimpul[0].getY()], width='100%', height='100%',
zoom_start=15.5)
            createAllMarker(listSimpul, m, "cadetblue", "All
Vertex")
            drawPathfromGraph(listSimpul, adjmat, m)
            #Add scroll Zoom toggler left bottom
            plugins.ScrollZoomToggler().add_to(m)
            # Add fullscreen button
            plugins.Fullscreen(position='topright').add_to(m)

            #set options untuk dropdown
            names = []
            for i in listSimpul:
                names.append(i.getName())
            asal.options = names
            asal.value = names[0]
            tujuan.options = names
            tujuan.value = names[0]

        except FileNotFoundError:
            print("File tidak ditemukan, pastikan nama file
benar")
        else:
```

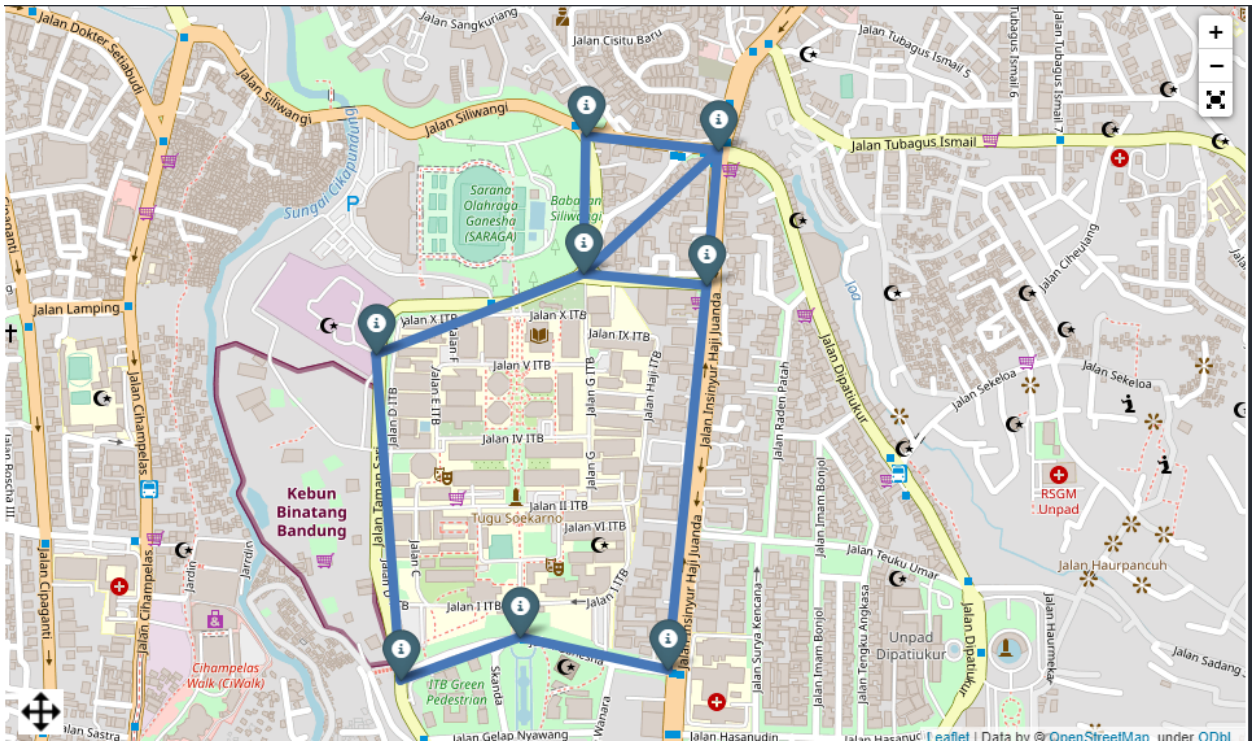
```
            #tidak ada error, file fitemukan
            display(widgets.HBox(whereto))
            display(outputSearch)
            display(initial_map)
            display(m)

### ONCLICKS ###
go.on_click(fileSelect)
cancel.on_click(cuancel)
find.on_click(searchGo)

### START ###
start = widgets.HBox(upfile, layout =
widgets.Layout(width='100%', justify_content='flex-start'))
```

5. File Stima.ipynb

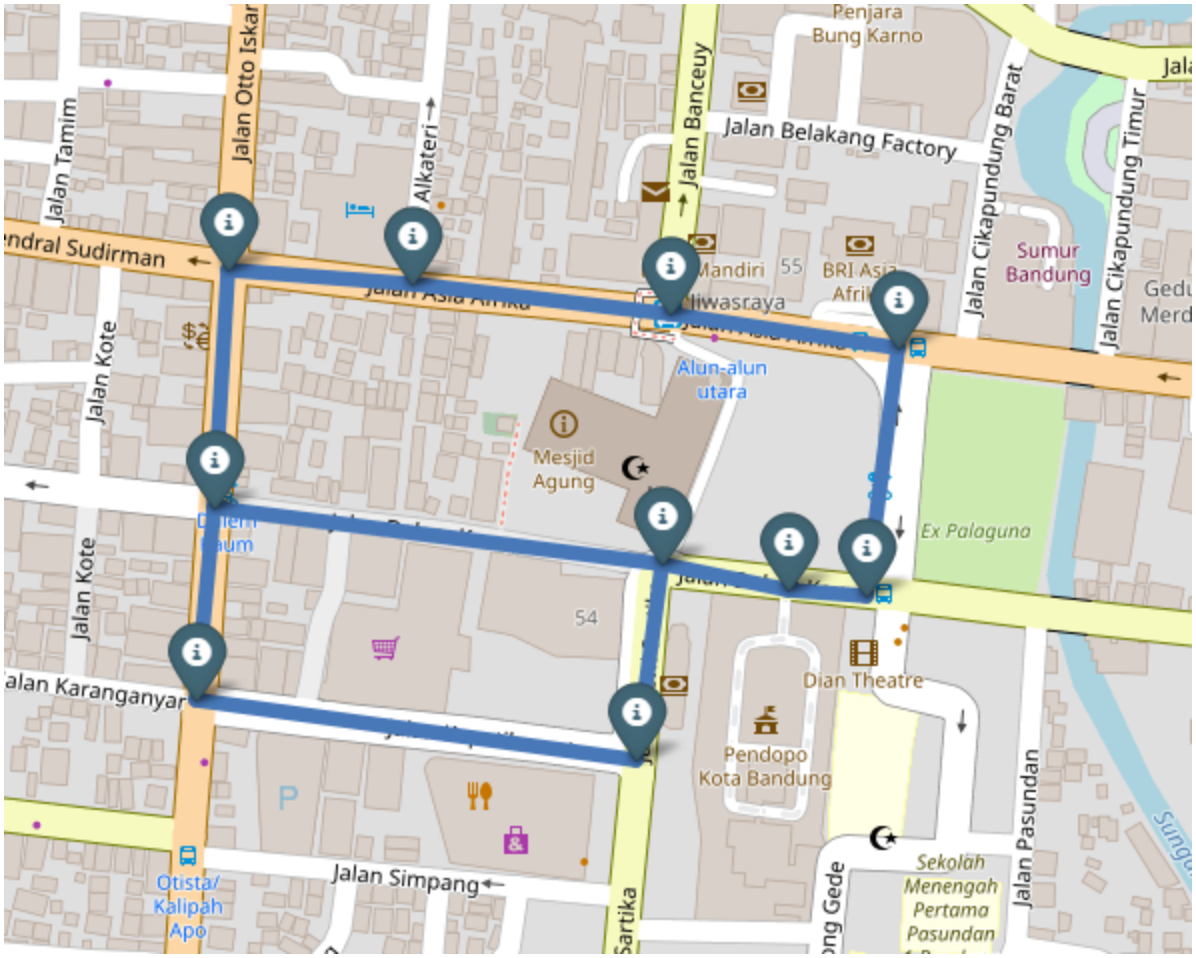
```
### RUN ###
from run import *
display(title)
display(start)
outputFileSelect
```

PETA

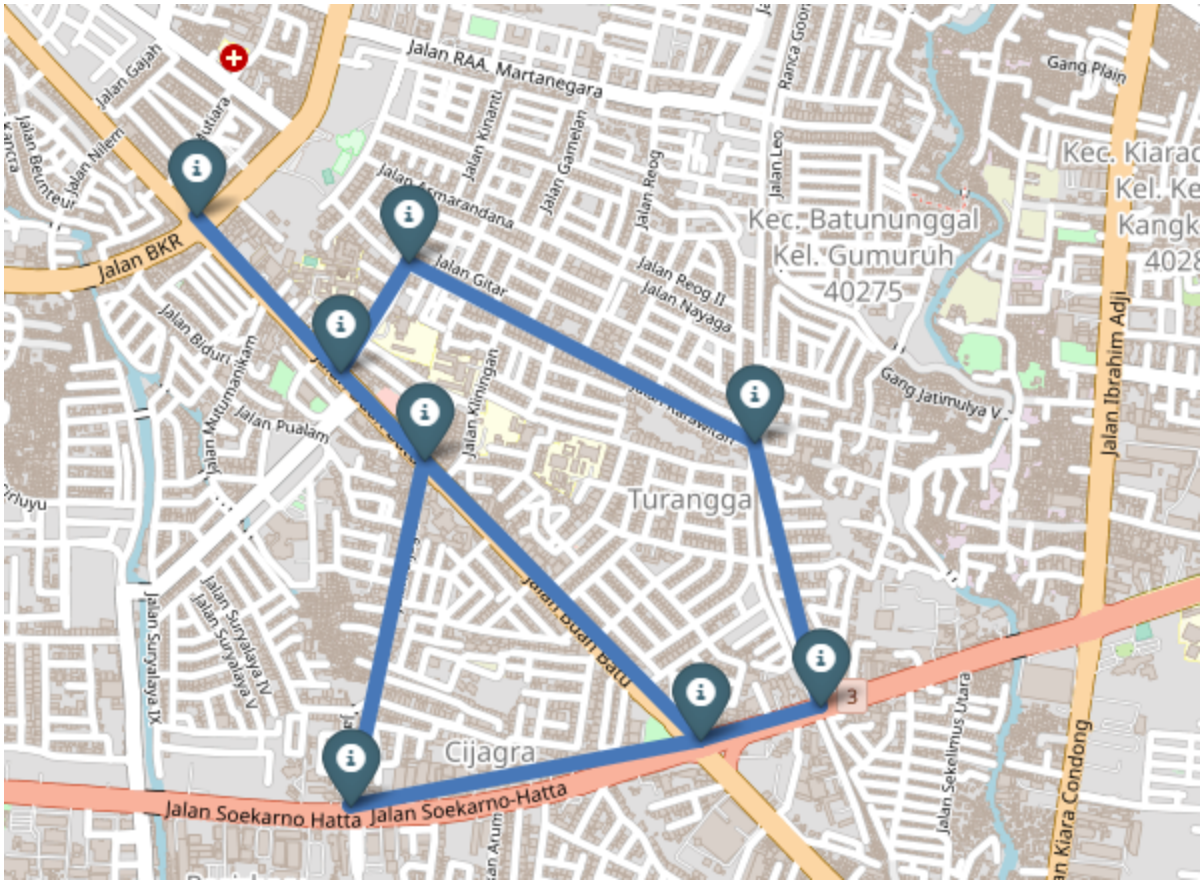
1. Sekitar ITB (SekitarITB.txt)



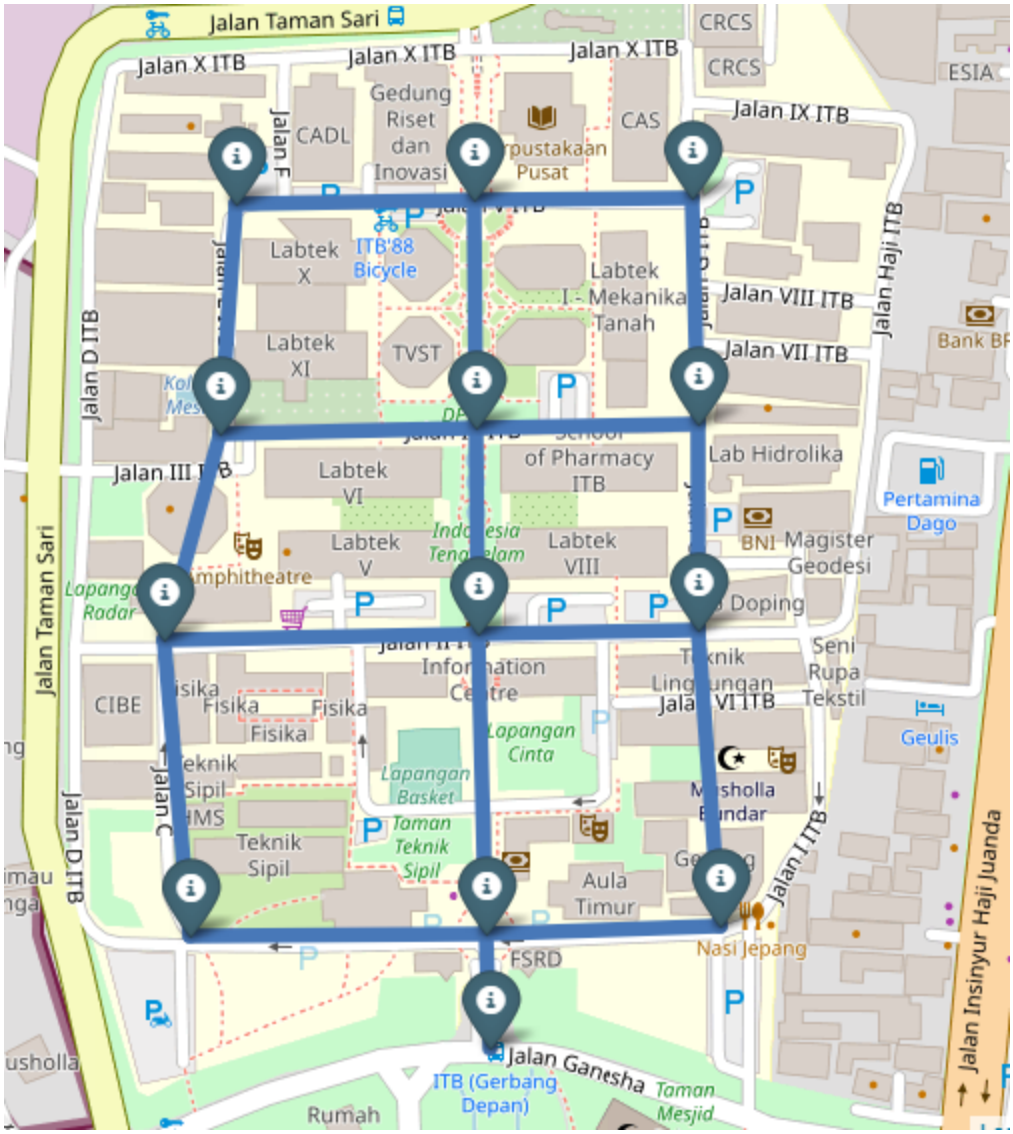
2. Alun-alun (Alunalun.txt)



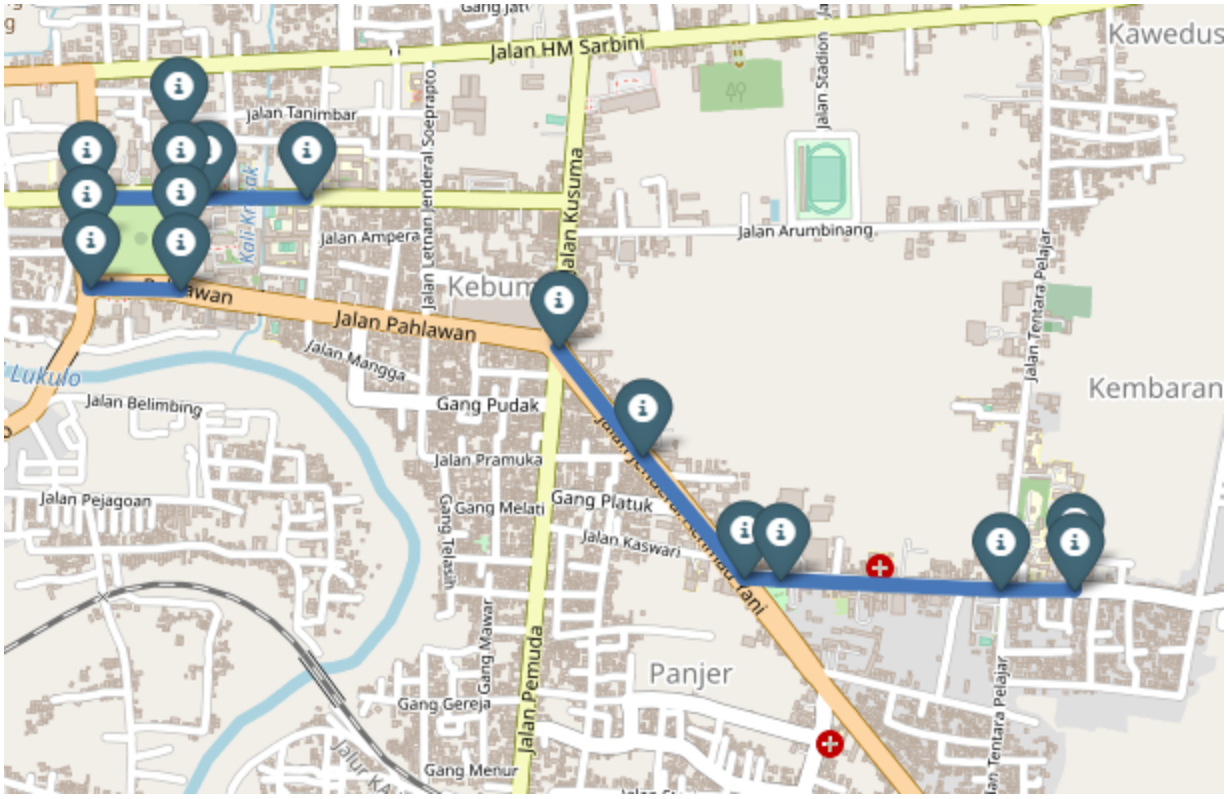
3. Buahbatu (Buahbatu.txt)



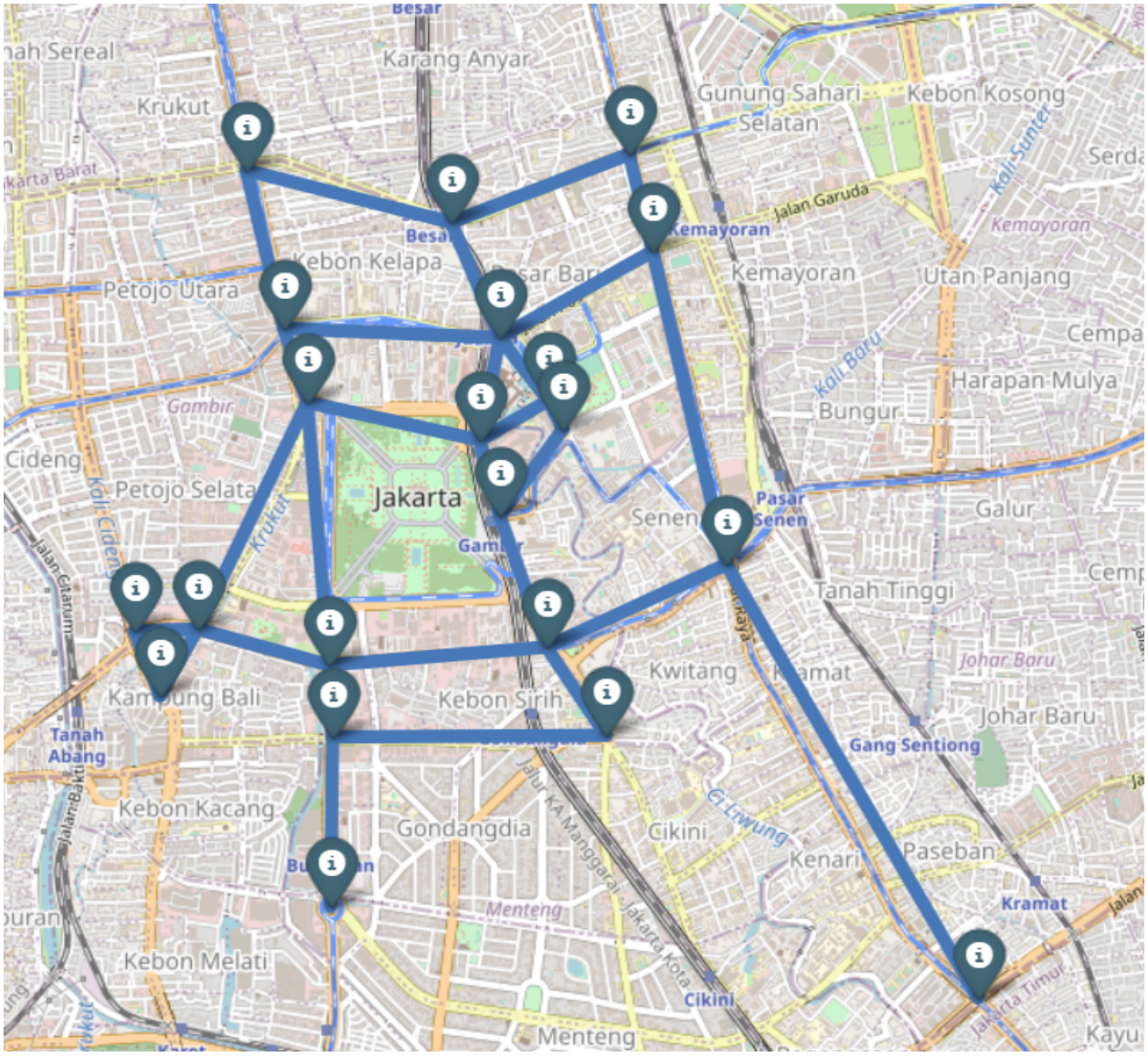
4. ITB (itb.txt)


Jalan Taman Sari
Jalan X ITB
Jalan X ITB
Jalan X ITB
CRCS
CRCS
ESIA
Gedung Riset dan Inovasi
CADL
CAS
Jalan IX ITB
Labtek X
ITB'88 Bicycle
Labtek I - Mekanika Tanah
Jalan VIII ITB
Jalan Haji ITB
Jalan D ITB
Labtek XI
TVST
Jalan VII ITB
Bank BR
Jalan III ITB
Labtek VI
School of Pharmacy ITB
Lab Hidrolika
Pertamina Dago
Labtek V
Labtek VIII
BNI
Magister Geodesi
Lapangan Radar
Amphitheatre
Doping
Seni Rupa Tekstil
CIBE
Fisika
Fisika
Fisika
Geulis
Lapangan Cinta
Teknik Sipil
Lapangan Basket
Musholla Bundar
Taman Teknik Sipil
Teknik Sipil
Aula Timur
Jalan I ITB
Nasi Jepang
FSRD
Jalan Ganesha
ITB (Gerbang Depan)
Taman Mesjid
Rumah
Jalan Insinyur Haji Juanda

5. Kebumen (Kebumen.txt)



6. Monas (Monas.txt)



Lintasan Terpendek

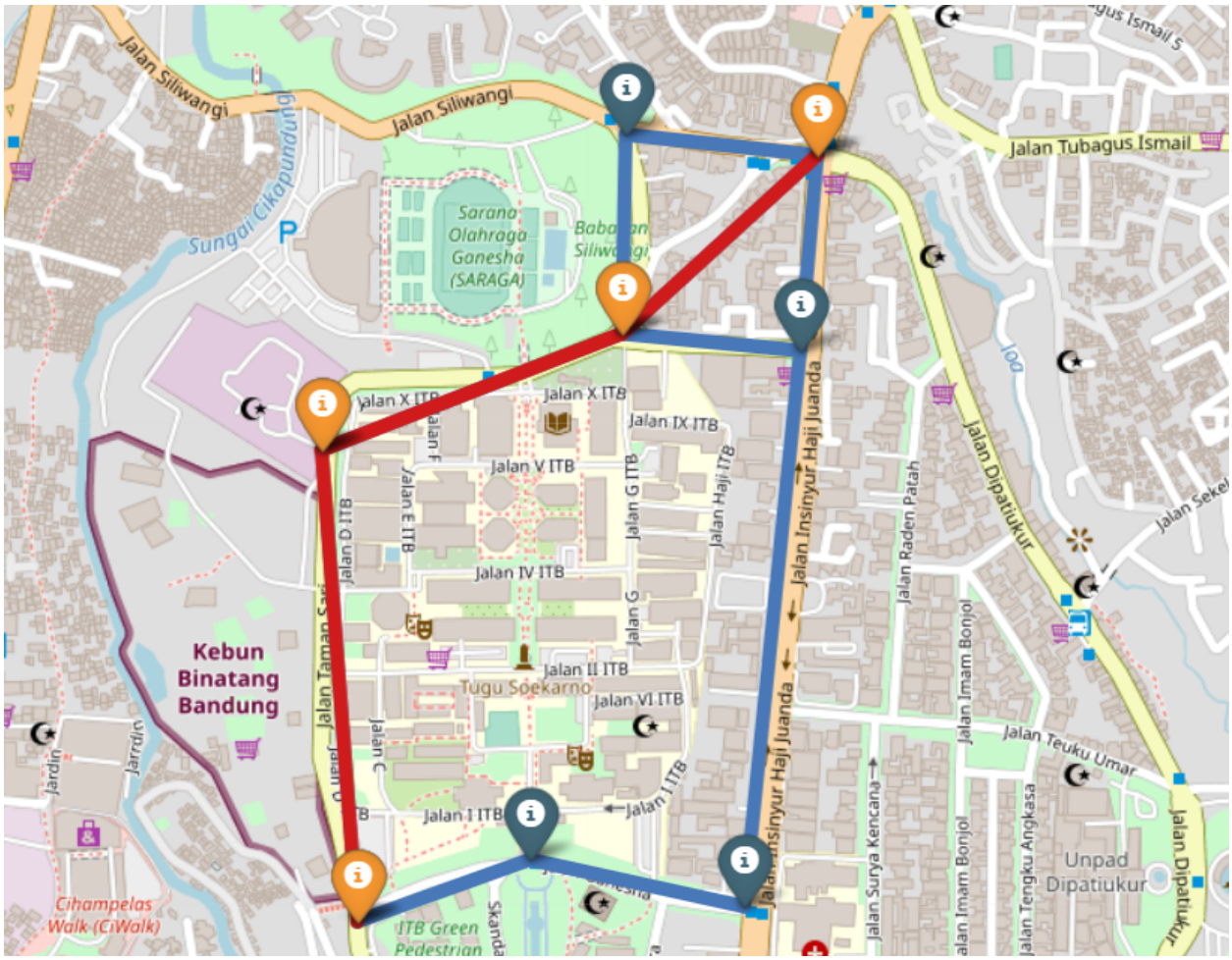
1. Sekitar ITB (SekitarITB.txt)

GOOGOLMEPZ

File: ../test/SekitarITB.txt Go

From : Pos_Polisi To : Bonbin Find Route Reset

```
Jalur :
Pos_Polisi → Segitiga_Dayang_Sumbi → Batan → Bonbin
Jarak tempuh :  1333.25 m
```



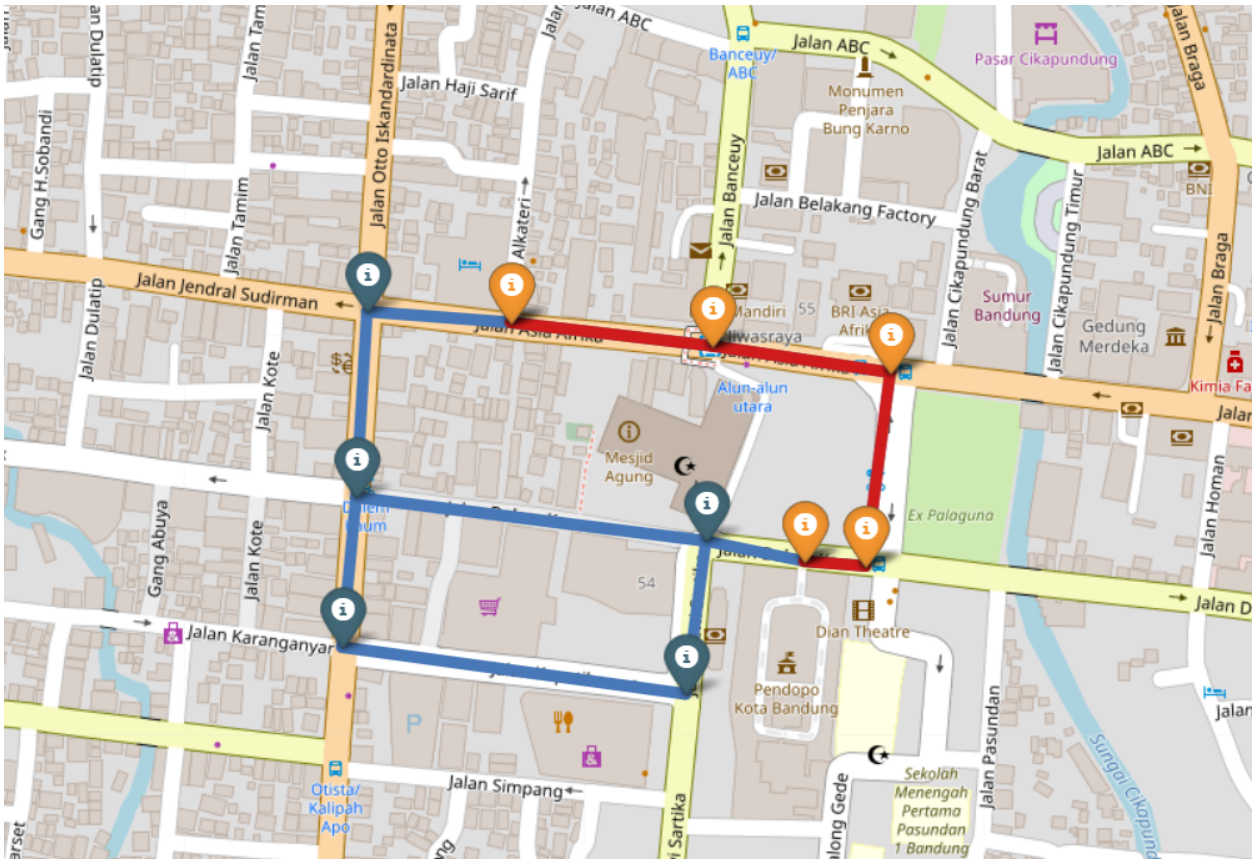
2. Alun-alun (Alunalun.txt)

GOOGOLMEPZ

File: ../test/Alunalun.txt Go

From : PendopoBandung To : KontikiLintasMedia Find Route Reset

```
Jalur :
PendopoBandung → DalemKaum63 → MasjidAgung → TokoKomputerNewPelangi → KontikiLintasMedia
Jarak tempuh :  484.44 m
```



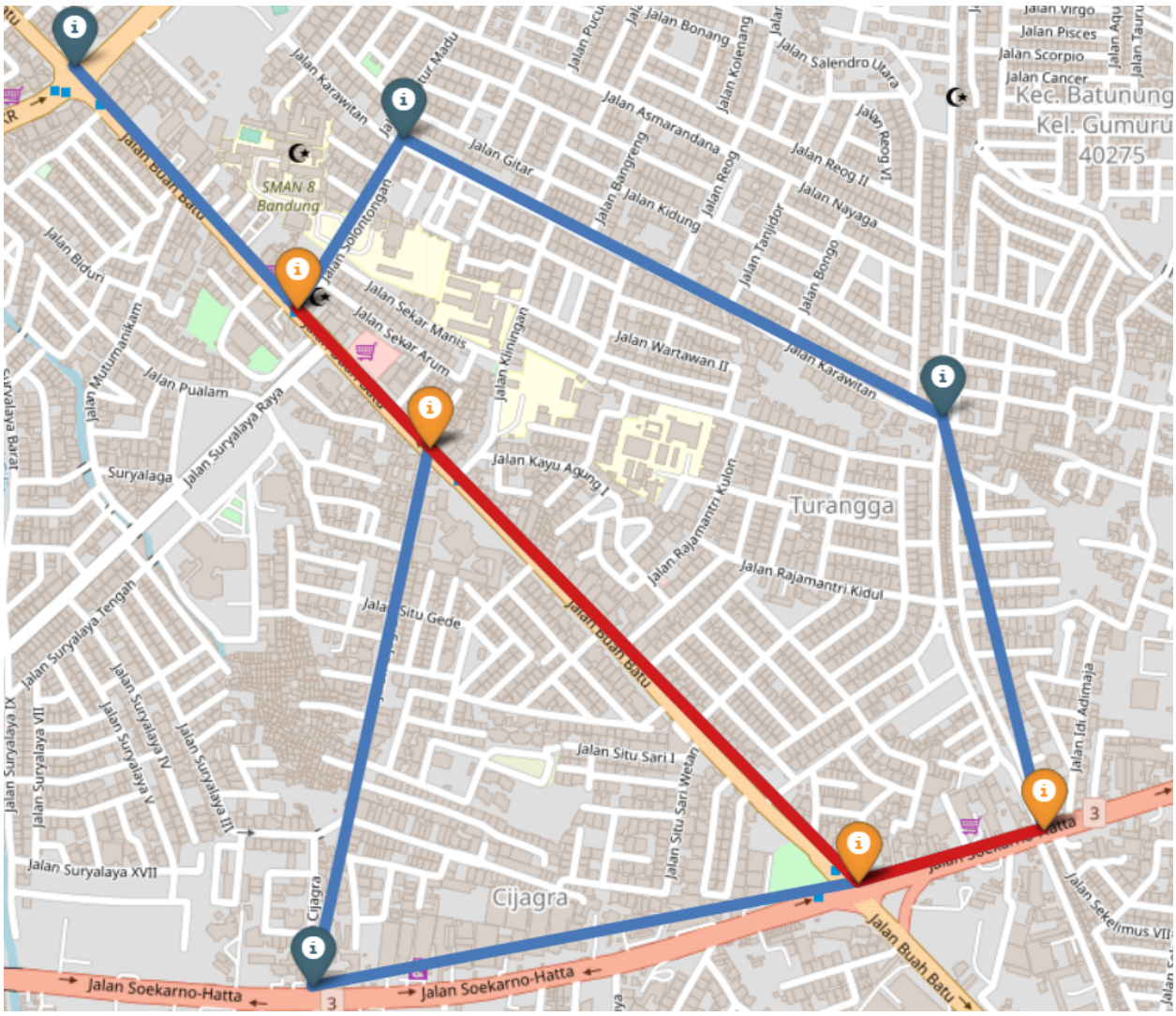
3. Buahbatu (Buahbatu.txt)

GOOGOLMEPZ

File: ../test/Buahbatu.txt Go

From : F To : C Find Route Reset

```
Jalur :
F → G → B → C
Jarak tempuh :  1513.64 m
```



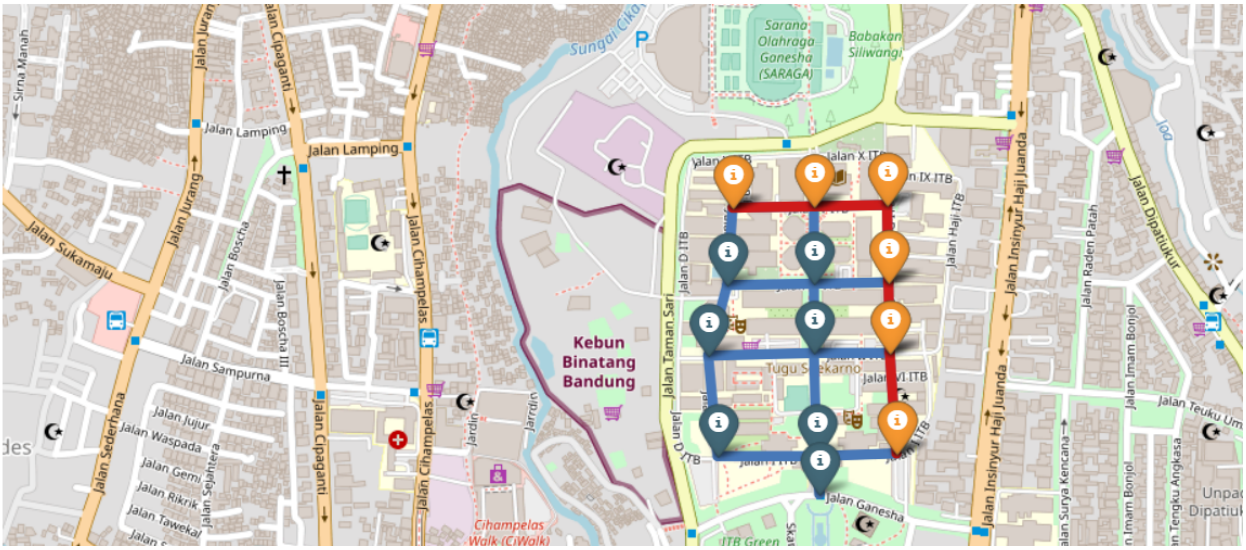
4. ITB (itb.txt)

GOOGOLMEPZ

File: ../test/itb.txt Go

From : K To : G Find Route Reset

```
Jalur :
K → L → M → J → F → G
Jarak tempuh :  702.27 m
```



5. Kebumen (Kebumen.txt)

GOOGOLMEPZ

File: ../test/Kebumen.txt Go

From : Pertigaan_Alun-Alun To : SMA_Negeri_1_Kebumen Find Route Reset

```
Jalur :
Pertigaan_Alun-Alun → Kantor_Pos → Kantor_Bupati → Bundaran_Alun-Alun → SMA_Negeri_1_Kebumen
Jarak tempuh :  494.35 m
```

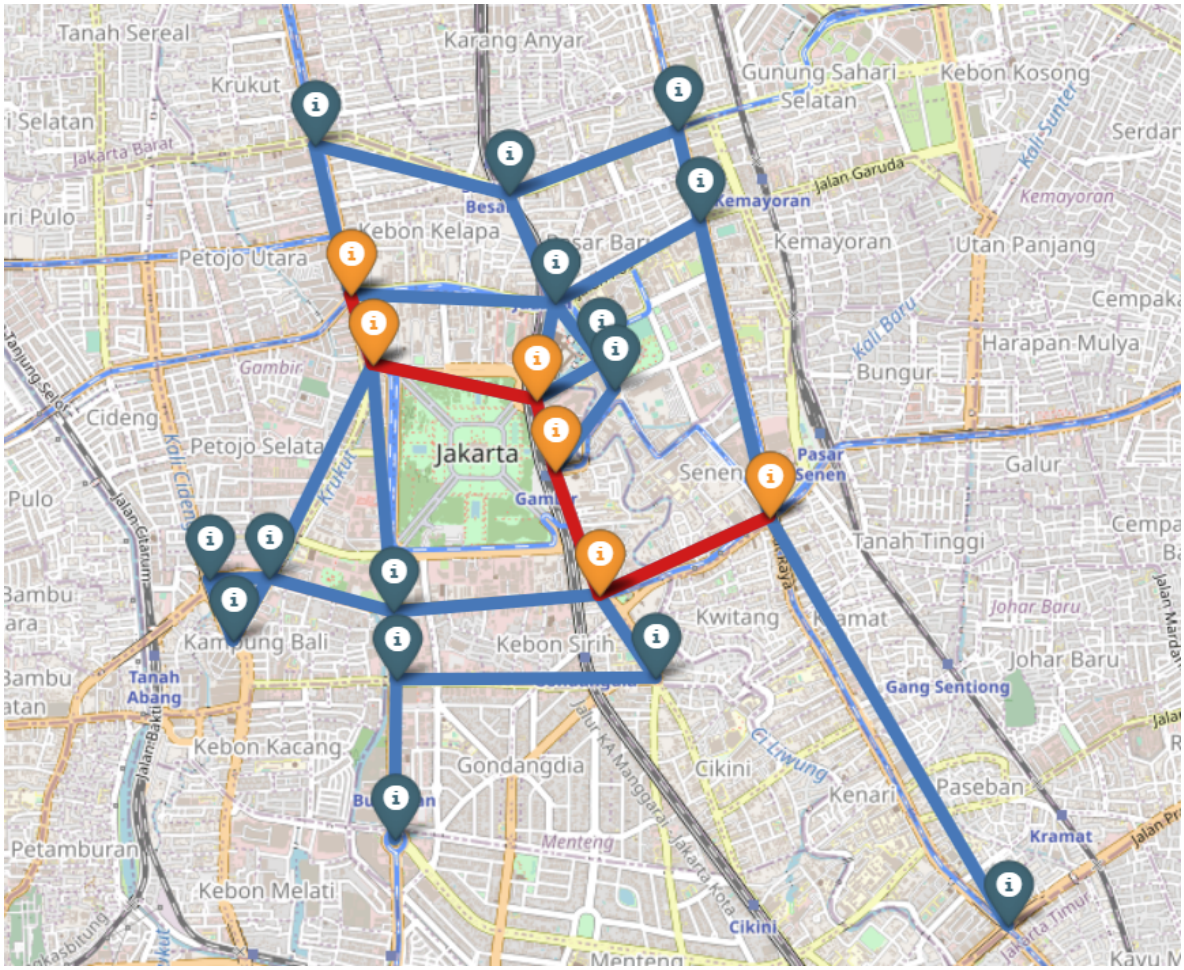
6. Monas (Monas.txt)
   Lintasan dari F - K

GOOGOLMEPZ

File: ../test/Monas.txt Go

From : F To : K Find Route Reset

```
Jalur :
F → E → N → P → M → K
Jarak tempuh :  3521.94 m
```


Tanah Sereal
Karang Anyar
Krukut
Gunung Sahari Selatan
Kebon Kosong
Jakarta Barat
Kemayoran
Jalan Garuda
Kali Sunter
Kebon Kelapa
Petojo Utara
Kemayoran
Utan Panjang
Kali Baru
Harapan Mulya
Bungur
Gambir
Cideng
Petojo Selata
Jakarta
Senen
Pasar Senen
Galur
Tanah Tinggi
Kwitang
Johar Baru
Kampung Bali
Kebon Sirih
Tanah Abang
Gang Sentiong
Kebon Kacang
Gondangdia
Cikini
Kenari
Paseban
Kramat
Petamburan
Menteng
Kebon Melati
Jakarta Kota
Cikini

Alamat kode program :

github.com/rayendito/threecilStima

Tabel ceklis mandiri

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. **Implementasi :** Digunakan OpenStreetView untuk menampilkan peta, membentuk graf dari peta, dan menampilkan lintasan terpendek di peta (berupa jalan yang diberi warna).	✓ (openstreetview)