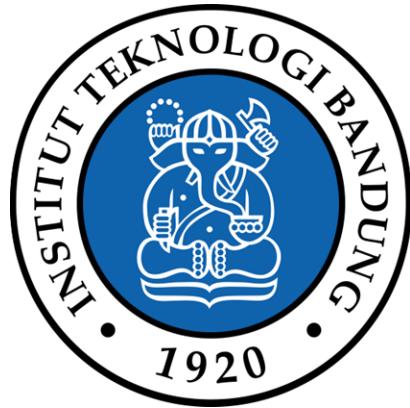


LAPORAN TUGAS KECIL 3

IF2211 Strategi Algoritma

Find Shortest Path

Menentukan Jalur Terdekat dengan Algoritma UCS dan A*



Oleh:

Afnan Edsa Ramadhan 13521011

Agsha Athalla Nurkareem 13521027

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

DAFTAR ISI

DAFTAR ISI	1
BAB I - Pendahuluan	2
1.1. Deskripsi Persoalan	2
BAB II - Source Code dalam Bahasa Python	3
2.1 File SplashScreen.py	3
2.2 File main.py	4
2.3 File FungsiUtama.py	9
2.4 File FungsiTambahan.py	13
2.5 File maps.py	15
BAB III - Uji Coba Program	17
3.1 Hasil Pengujian dengan peta Bandung Selatan	17
3.2 Hasil Pengujian dengan peta Alun-Alun Bandung	20
3.3 Hasil Pengujian dengan peta Jatinangor	24
3.4 Hasil Pengujian dengan peta Bandung Utara	28
BAB 4 - Penutup	31
4.1 Kesimpulan	31
4.2 Komentar	31
BAB 5 - Lampiran	32
5.1 Tautan repository GitHub	32
5.2 Cek list	32

BAB I

Pendahuluan

1.1. Deskripsi Persoalan

Uniform Cost Search (UCS) merupakan sebuah algoritma yang digunakan untuk mencari jalur terpendek pada graf berbobot (weighted graph). Algoritma UCS memperhitungkan biaya total yang dibutuhkan untuk mencapai suatu simpul, dan mengembangkan pencarian dengan mengeksplorasi simpul dengan biaya yang lebih rendah terlebih dahulu. Algoritma UCS sangat berguna dalam menyelesaikan masalah optimasi yang melibatkan pencarian jalur terpendek pada graf dengan biaya edge yang bervariasi. Algoritma ini merupakan modifikasi dari algoritma Breadth First Search (BFS).

A* (A-Star) merupakan algoritma pencarian graf yang menggabungkan konsep dari algoritma Uniform Cost Search (UCS) dengan heuristik yang diterapkan pada fungsi biaya. Heuristik diartikan sebagai suatu fungsi yang memberikan suatu nilai berupa nilai estimasi dari suatu solusi. Algoritma A* menemukan jalur terpendek dengan mempertimbangkan biaya total dari node awal hingga node akhir dan heuristik untuk memperkirakan biaya yang tersisa. Heuristik yang digunakan dalam A* harus konsisten dan dapat diterima untuk menjamin pencarian jalur terpendek.

Langkah pertama di dalam program ini adalah membuat graf yang merepresentasikan peta. Berdasarkan graf yang dibentuk, lalu program menerima input simpul asal dan simpul tujuan, lalu menentukan lintasan terpendek antara keduanya menggunakan algoritma UCS dan A*. Lintasan terpendek dapat ditampilkan pada peta/graf (misalnya jalan-jalan yang menyatakan lintasan terpendek diberi warna merah). Nilai heuristik yang dipakai adalah jarak garis lurus dari suatu titik ke tujuan.

BAB II

Source Code dalam Bahasa Python

2.1 File SplashScreen.py

```
splashScreen()
```

2.2 File main.py

```
import networkx as nx

import matplotlib.pyplot as plt

import webbrowser

from SplashScreen import *

from FungsiUtama import *

from FungsiTambahan import *

from maps import *

def mainScreen():

    splashScreen()

    print("=====")

    isiFolder = os.listdir(os.getcwd() + '/test')

    print("Daftar peta yang tersedia dalam folder test: ")

    for i in range(len(isiFolder)):

        print(i+1, isiFolder[i].replace(".txt", ""))
```



```

a = mintaInputInt("Masukkan Node Awal: ", len(matrix))

b = mintaInputInt("Masukkan Node Tujuan: ", len(matrix))

#convert ke dictionary

dictMat = convertMatrixToDict(matrix)

if(algo==1):

    hasil, visited, cost, path = UCS(a, b, dictMat)

    print("Hasil UCS: ")

    print("Jarak:", cost,"Meter")

    printPath(path, nama)

elif(algo==2):

    hasil, visited, cost, path = AStar(a, b, dictMat, koordinat)

    print("Hasil A*: ")

    print("Jarak:", cost,"Meter")

    printPath(path, nama)

# Membuat objek graf kosong

G = nx.Graph()

# Menambahkan simpul ke dalam graf

for i in range(len(matrix)):

    G.add_node(i+1)

```

```

# Menambahkan sambungan antar simpul beserta bobotnya ke dalam graf

for i in range(1, len(dictMat)+1):

    for j in dictMat[i-1]:

        if(i in path and j in path):

            # warnain edge yang dikunjungin jadi merah

            G.add_edge(i, j, weight=dictMat[i-1][j], color='r')

        else:

            # warnain edge yang ga dikunjungin jadi hitam

            G.add_edge(i, j, weight=dictMat[i-1][j], color='black')

# warnain semua jadi biru

node_colors = []

for i in range(len(dictMat)):

    node_colors.append('b')

# warnain yang dikunjungi jadi merah

for i in (path):

    node_colors[i-1] = 'r'

# setting buat warnain edge

edges = G.edges()

colors = [G[u][v]['color'] for u,v in edges]

```

```

# Menggambar graf menggunakan matplotlib

nx.draw_networkx_labels(G, koordinat)

nx.draw_networkx_edges(G, koordinat, edge_color=colors, width=1.0)

nx.draw_networkx_nodes(G, koordinat, node_color=node_colors,
node_size=500)

labels = nx.get_edge_attributes(G, 'weight')

nx.draw_networkx_edge_labels(G, koordinat, edge_labels=labels)

print("Visualisasi Graf akan ditampilkan")

print("Tutup visualisasi untuk melanjutkan")

plt.show()

showMap(koordinat, path)

tanyaMaps = input("Apakah ingin menampilkan maps? (Y/N) : ")

if(tanyaMaps=="Y" or tanyaMaps=="y"):

    print("Maps akan ditampilkan")

    webbrowser.open_new_tab('file://'+os.getcwd()+'/bin/maps.html')

pilihan = input("Apakah ingin mengulang pencarian? (Y/N) : ")

if(pilihan=="Y" or pilihan=="y"):

    main()

else:

    splashScreen()

print("=====")

```

```
print("Terima kasih telah menggunakan program ini")

if __name__ == "__main__":
    main()
```

2.3 File FungsiUtama.py

```
from queue import PriorityQueue

from FungsiTambahan import *

def UCS(start, end, dict):
    visited = []
    hasil = []
    avail = PriorityQueue()

    # cost, path
    avail.put((0, [start]))

    target = end

    if start == end:
        return hasil, visited, 0, [start]

    if(len(dict[start-1]) == 0):
        print("Tidak ada jalur")
        return hasil, visited, 0, [start]

    elif(len(dict[end-1]) == 0):
```

```

print("Tidak ada jalur")

return hasil, visited, 0, [end]

while not avail.empty():

    cost, path = avail.get()

    tempPath = path.copy()

    curr = path[-1]

    visited.append(path)

    for i in dict[curr-1]:

        path = tempPath.copy()

        if i not in path:

            if(curr != target):

                path.append(i)

                avail.put((cost + dict[curr-1][i], path))

                hasil.append((cost + dict[curr-1][i], (path)))

#cari nilai minimum

min = 9999999

path = list

for i in range(len(hasil)):

    if hasil[i][1][0] == (start) and hasil[i][1][-1] == (end):

        if(hasil[i][0] < min):

            min = hasil[i][0]

            path = (hasil[i][1])

```

```
return hasil, visited, min, path

def AStar(start, end, dict, coordinate):
    visited = []
    hasil = []
    avail = PriorityQueue()

    # cost, path
    avail.put((0, [start]))
    target = end
    count = 0

    if start == end:
        return hasil, visited, 0, [start]

    if(len(dict[start-1]) == 0):
        print("Tidak ada jalur")
        return hasil, visited, 0, [start]
    elif(len(dict[end-1]) == 0):
        print("Tidak ada jalur")
        return hasil, visited, 0, [end]
```

```

while not avail.empty():

    cost, path = avail.get(0)

    tempPath = path.copy()

    curr = path[len(path)-1]

    if curr == target:

        break

    if cost>0:

        cost -= getHeuristic(path[-1], end, coordinate)

        visited.append(path)

        for i in dict[curr-1]:

            path = tempPath.copy()

            if i not in path:

                if(curr != target):

                    path.append(i)

                    # avail.put((cost + dict[curr-1][i] + heuristic[i-1] ,
path))

                    avail.put((cost + dict[curr-1][i] + getHeuristic(i,
end, coordinate) , path))

                    hasil.append((cost + (dict[curr-1][i]) , (path)))

                    # print(curr-1," = ", hasil[len(hasil)-1])

                    count += 1

return hasil, visited, cost, path

```

2.4 File FungsiTambahan.py

```
import numpy as np

def mintaInputInt(string, max=10):
    try:
        x = int(input(string))
        if(x > max):
            print("Input harus kurang dari sama dengan", max)
            return mintaInputInt(string, max)
        elif(x < 0):
            print("Input harus lebih dari 0")
            return mintaInputInt(string, max)
        return x
    except:
        print("Input harus berupa angka")
        return mintaInputInt(string)

def bacaFile(filename):
    try:
        file = open(filename, "r")
        file = file.readlines()

        matrix = []
        namaTempat = []
        coordinates = {}
        for i in range(len(file)):
            if(i==0):
                file[i] = file[i].split(",")
            else:
                file[i] = file[i].split(" ")
            if(i==0):
                for j in range(len(file[0])):
                    namaTempat.append(file[i][j].strip())
            elif(len(file[i])==3):
                coordinates[i-len(file[1])] = np.array([float(file[i][1]),
                float(file[i][2])])
            else:
                temp = []
                for j in range(len(file[i])):
```

```

        file[i][j] = int(file[i][j])
        temp.append(file[i][j])
    matrix.append(temp)
    # Validasi jumlah node perbaris
    if(i>0 and len(file[i]) != len(file[i-1])):
        print("Jumlah node perbaris tidak sama")
        filename = input("Masukkan nama file lain: ")
        return bacaFile(filename)

    # Validasi apakah jumlah node kurang dari 8
    if(len(matrix[0])<8):
        print("Jumlah node kurang dari 8")
        filename = input("Masukkan nama file lain: ")
        return bacaFile(filename)

    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            #validasi apakah matrix simetris
            if(matrix[i][j] != matrix[j][i]):
                print("Matrix tidak simetris")
                filename = input("Masukkan nama file lain: ")
                return bacaFile(filename)
            #validasi apakah matrix diagonal nol
            if(matrix[i][i]!=0):
                print("Diagonal matrix tidak nol")
                filename = input("Masukkan nama file lain: ")
                return bacaFile(filename)

    print("File berhasil dibaca")
    return matrix, coordinates, namaTempat
except:
    print("File tidak ditemukan atau tidak dapat dibuka")
    filename = input("Masukkan nama file: ")
    return bacaFile(filename)

def getHeuristic(start, end, coordinate):
    x = coordinate[start][0] - coordinate[end][0]
    y = coordinate[start][1] - coordinate[end][1]
    return np.linalg.norm(np.array([x,y]))

```

```

def printMatrix(matrix):
    for i in range(len(matrix)):
        for j in range(len(matrix[i])):
            print(matrix[i][j], end=" ")
    print()

def printPath(path, nama):
    for i in range(len(path)):
        if(i == len(path)-1):
            print(nama[path[i]-1], end="")
        else:
            print(nama[path[i]-1], "-> ", end="")
    print()

def convertMatrixToDict(matrix):
    temp = []
    for i in range(len(matrix)):
        tempDict = {}
        for j in range(len(matrix[i])):
            if matrix[i][j] >= 1:
                tempDict[j+1] = matrix[i][j]
        temp.append(tempDict)
    return temp

```

2.5 File maps.py

```

import gmplot

def findTengah(koordinat):
    minLat = 999999999
    minLong = 999999999
    maxLat = -999999999
    maxLong = -999999999

    for i in koordinat:
        if(koordinat[i][0] < minLong):
            minLong = koordinat[i][0]
        if(koordinat[i][0] > maxLong):
            maxLong = koordinat[i][0]

```

```

        if(koordinat[i][1] < minLat):
            minLat = koordinat[i][1]
        if(koordinat[i][1] > maxLat):
            maxLat = koordinat[i][1]
    return (minLat+maxLat)/2, (minLong+maxLong)/2

def showMap(koordinat, path):

    latitude_list = []
    longitude_list = []
    for i in path:
        latitude_list.append(koordinat[i][1])
        longitude_list.append(koordinat[i][0])

    lat_Tengah, long_Tengah = findTengah(koordinat)

    gmap = gmapplot.GoogleMapPlotter(lat_Tengah, long_Tengah, 16,
map_type='roadmap', apikey='')

    gmap.scatter( latitude_list, longitude_list, '#FF0000', size = 40,
marker = False )
    gmap.plot(latitude_list, longitude_list, 'cornflowerblue', edge_width
= 2.5)

    gmap.draw('bin/maps.html')

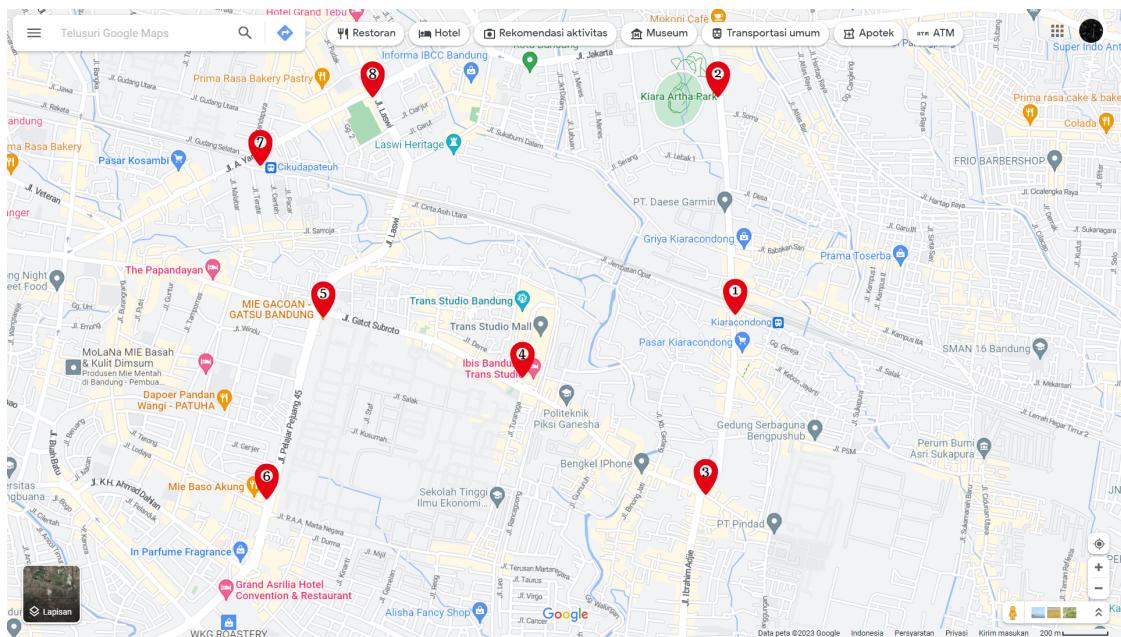
```

BAB III

Uji Coba Program

3.1 Hasil Pengujian dengan peta Bandung Selatan

Bentuk Peta:



Graf input:

St. Kiaracondong,Kiara Artha Park,Jabrix Hotel,TSM,Mie Gacoan,Plaza Toyota,st.
Cikudapeteuh,Stadion Persib
0 949 837 0 0 0 0 0
949 0 0 0 0 0 0 0
837 0 0 938 0 0 0 0
0 0 938 0 1020 0 0 0
0 0 0 1020 0 890 0 1140
0 0 0 0 890 0 0 0
0 0 0 0 0 0 638
0 0 0 0 1140 0 638 0
1 107.64445579682963 -6.924553425378429
2 107.64386913703503 -6.9160919669451015
3 107.64306750082548 -6.931818653404809
4 107.63574637977118 -6.927510412327061
5 107.62767286864126 -6.924540250235155

```
6 107.62570745767948 -6.932191032147432
7 107.62473245601261 -6.91850239447493
8 107.62981717366264 -6.915756663252279
```

Keluaran dalam terminal:

a. Algoritma UCS

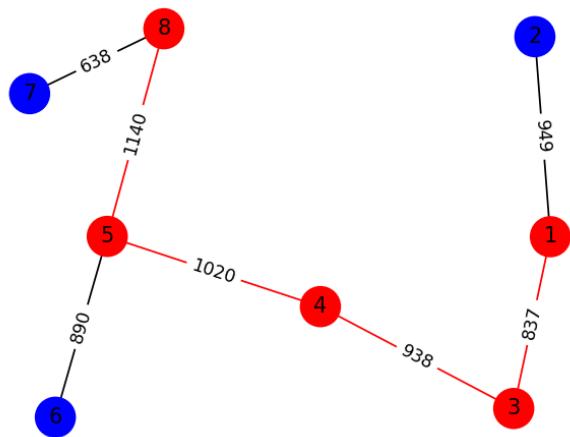
```
=====
Daftar peta yang tersedia dalam folder test:
1 adakosong
2 alun-alun
3 bandungSelatan
4 bandungUtara
5 nangor
6 turki
Pilih nomor file yang ingin dibuka: 3
File berhasil dibaca

Pilih Algorithm yang ingin digunakan:
1. UCS
2. A*
Masukkan pilihan: 1
Kamu memilih UCS
DAFTAR NODE:
1 St. Kiaracondong
2 Kiara Artha Park
3 Jabrix Hotel
4 TSM
5 Mie Gacoan
6 Plaza Toyota
7 st. Cikudapeteuh
8 Stadion Persib
Masukkan Node Awal: 1
```

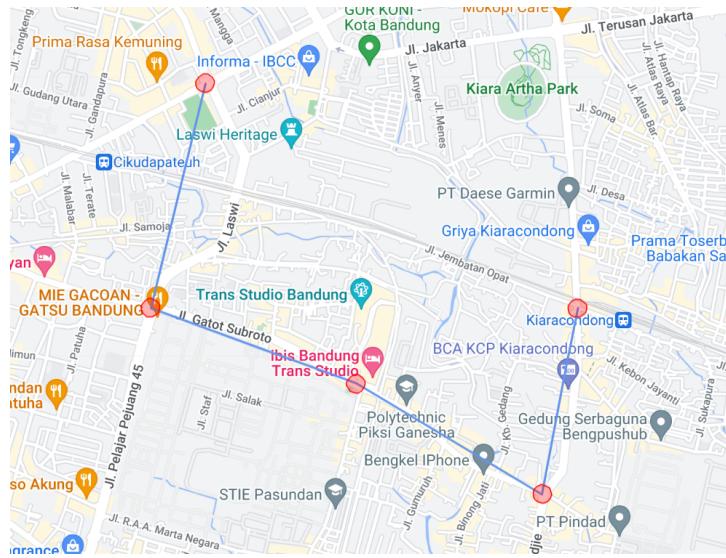
```
Masukkan Node Tujuan: 8
Hasil UCS:
Jarak: 3935 Meter
St. Kiaracondong -> Jabrix Hotel -> TSM -> Mie Gacoan -> Stadion Persib
Visualisasi Graf akan ditampilkan
Tutup visualisasi untuk melanjutkan
```

b. Algoritma A*

Keluaran dalam visualisasi:

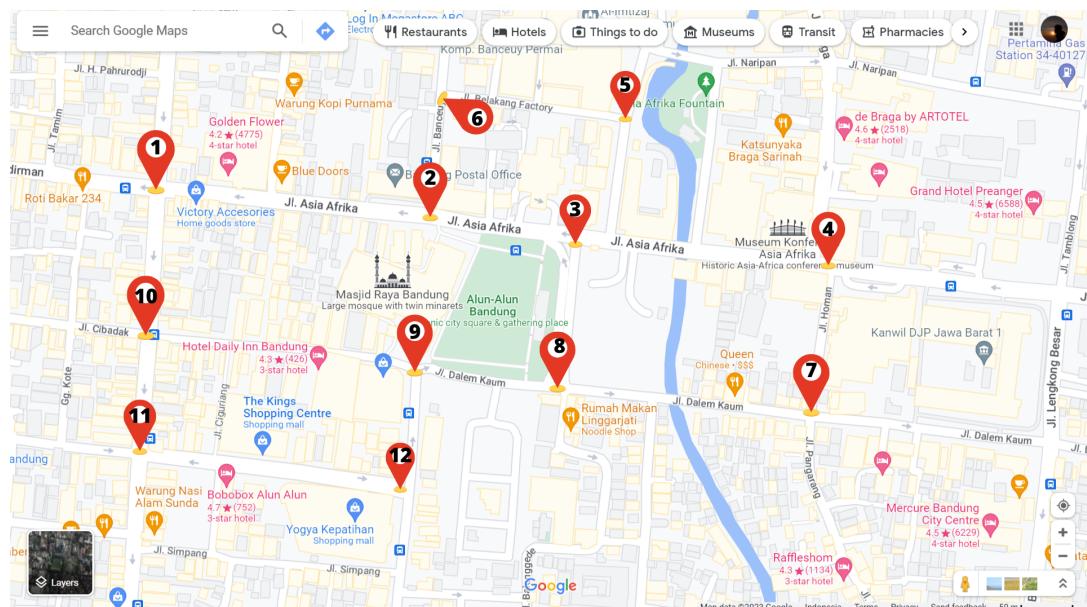


Keluaran dalam peta:



3.2 Hasil Pengujian dengan peta Alun-Alun Bandung

Betuk Peta:



Graf input:

Victory Accesorice,Kantor Pos,Kanan Atas Alun-Alun,Museum KAA,Kopi Senja,Qss

Premium Care,Golden Princes Karaoke,Kanan Bawah Alun-Alun,Kiri Bawah
Alun-Alun,Toko Emas Eiffel,Roti Boy,Bebek Salero

0 264 0 0 0 0 0 0 0 141 0 0
264 0 148 0 0 120 0 0 0 0 0 0
0 148 0 231 148 0 0 140 0 0 0 0
0 0 231 0 0 0 144 0 0 0 0 0
0 0 148 0 0 182 0 0 0 0 0 0
0 120 0 0 182 0 0 0 0 0 0 0
0 0 0 144 0 0 0 245 0 0 0 0
0 0 140 0 0 0 245 0 139 0 0 0
0 0 0 0 0 0 0 139 0 265 0 115
141 0 0 0 0 0 0 0 265 0 117 0
0 0 0 0 0 0 0 0 117 0 255
0 0 0 0 0 0 0 115 0 255 0
1 107.60406512310648 -6.920814530300131
2 107.6064791111338 -6.921027543484346
3 107.60771292723665 -6.9212405565724335
4 107.61195081732905 -6.921730486310206
5 107.60816353833508 -6.920239394222596
6 107.6065649418192 -6.920015730003211
7 107.61188644431498 -6.923051163939521
8 107.6075949100442 -6.922507982458791
9 107.60633963627 -6.922412126838457
10 107.6039685635854 -6.922050005430406
11 107.60390419057131 -6.923115067602032
12 107.60624307674891 -6.923391983372905

Keluaran dalam terminal:

a. Algoritma UCS

```
=====
Daftar peta yang tersedia dalam folder test:
1 adakosong
2 alun-alun
3 bandungSelatan
4 bandungUtara
5 nangor
6 turki
Pilih nomor file yang ingin dibuka: 2
File berhasil dibaca

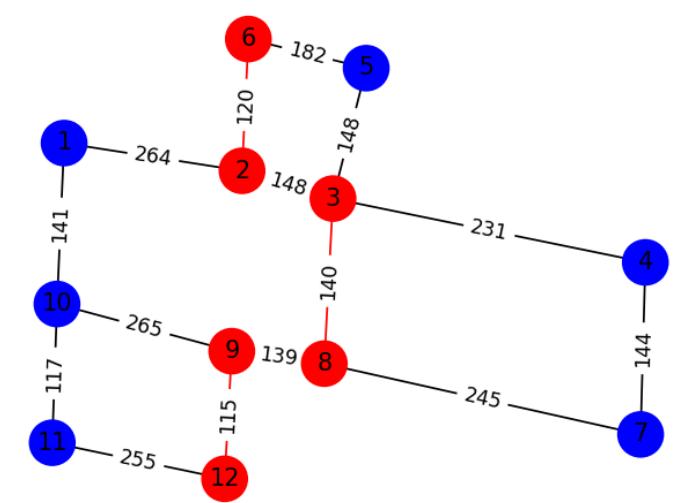
Pilih Algorithm yang ingin digunakan:
1. UCS
2. A*
Masukkan pilihan: 1
Kamu memilih UCS
DAFTAR NODE:
1 Victory Accesorce
2 Kantor Pos
3 Kanan Atas Alun-Alun
4 Museum KAA
5 Kopi Senja
6 Qss Premium Care
7 Golden Princes Karaoke
8 Kanan Bawah Alun-Alun
9 Kiri Bawah Alun-Alun
10 Toko Emas Eiffel
11 Roti Boy
12 Bebek Salero
Masukkan Node Awal: 12
Masukkan Node Tujuan: 6
Hasil UCS:
Jarak: 662 Meter
Bebek Salero -> Kiri Bawah Alun-Alun -> Kanan Bawah Alun-Alun -> Kanan Atas Alun-Alun -> Kantor Pos -> Qss Premium Care
Visualisasi Graf akan ditampilkan
Tutup visualisasi untuk melanjutkan
```

b. Algoritma A*

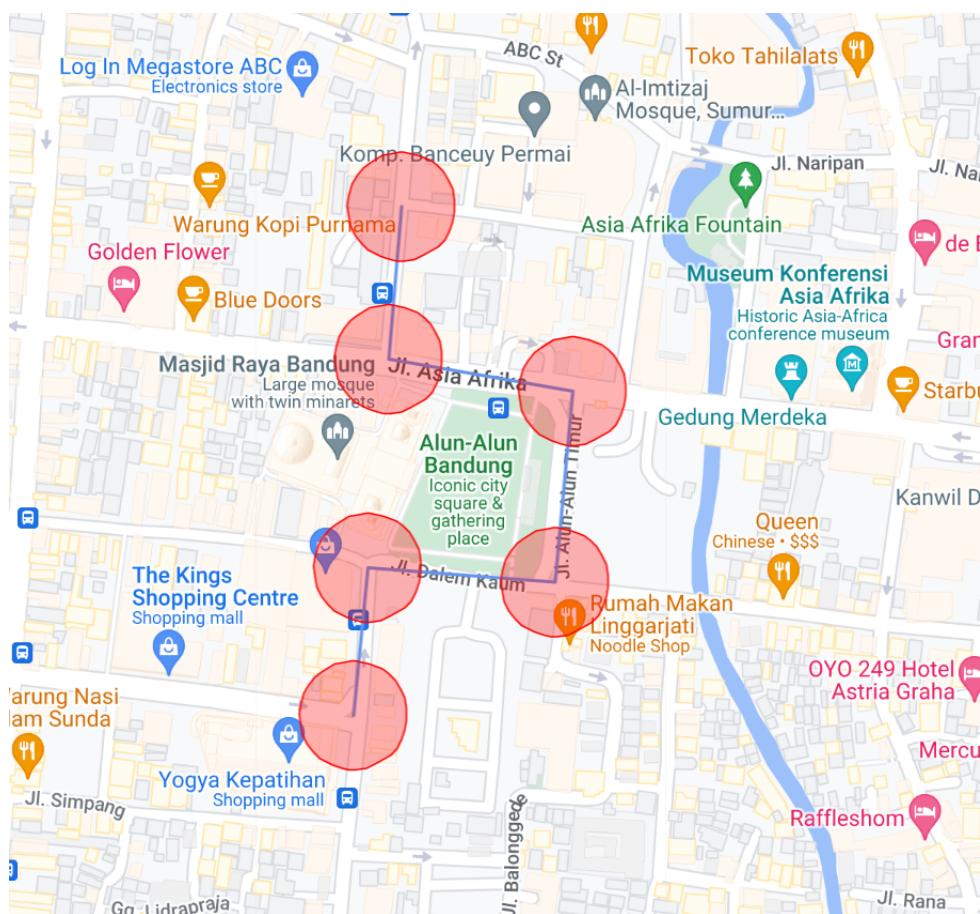
```
=====
Daftar peta yang tersedia dalam folder test:
1 adakosong
2 alun-alun
3 bandungSelatan
4 bandungUtara
5 nangor
6 turki
Pilih nomor file yang ingin dibuka: 2
File berhasil dibaca

Pilih Algorithm yang ingin digunakan:
1. UCS
2. A*
Masukkan pilihan: 2
Kamu memilih A*
DAFTAR NODE:
1 Victory Accesorce
2 Kantor Pos
3 Kanan Atas Alun-Alun
4 Museum KAA
5 Kopi Senja
6 Qss Premium Care
7 Golden Princes Karaoke
8 Kanan Bawah Alun-Alun
9 Kiri Bawah Alun-Alun
10 Toko Emas Eiffel
11 Roti Boy
12 Bebek Salero
Masukkan Node Awal: 12
Masukkan Node Tujuan: 6
Hasil A*:
Jarak: 662.0 Meter
Bebek Salero -> Kiri Bawah Alun-Alun -> Kanan Bawah Alun-Alun -> Kanan Atas Alun-Alun -> Kantor Pos -> Qss Premium Care
Visualisasi Graf akan ditampilkan
Tutup visualisasi untuk melanjutkan
```

Keluaran dalam visualisasi:

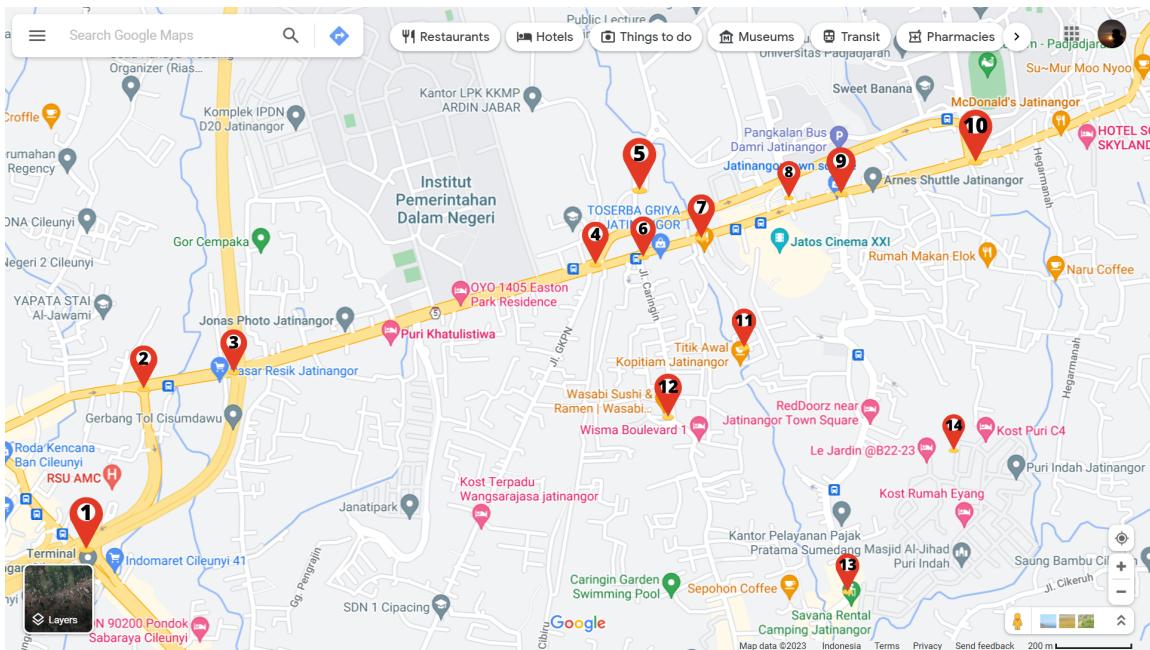


Keluaran dalam peta:



3.3 Hasil Pengujian dengan peta Jatinangor

Bentuk Peta:



Graf input:

```
Terminal,Pintu Tol,Pasar Resik,Puteran ITB,ITB,Caringin,Kp Geulis,Puteran Mixue,Jalan Sayang,Gerlam,Kos Yunis,Wasabi,Puri Indah,Kos Salman  
0 516 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
516 0 260 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 260 0 977 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 977 0 286 155 0 0 0 0 0 0 0 0 0 0  
0 0 0 286 0 0 0 521 0 1007 0 0 0 0 0  
0 0 0 155 0 0 158 0 0 0 426 0 0  
0 0 0 0 0 158 0 314 0 0 360 0 0 0  
0 0 0 0 521 0 314 0 170 0 0 0 0 0  
0 0 0 0 0 0 170 0 346 0 0 1200 0  
0 0 0 0 1007 0 0 0 346 0 0 0 0 0  
0 0 0 0 0 360 0 0 0 0 0 0 0 0  
0 0 0 0 0 426 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 1200 0 0 0 0 695  
0 0 0 0 0 0 0 0 0 0 0 695 0  
1 107.7552542545158 -6.941684198862088  
2 107.75654171475259 -6.93799921868306  
3 107.75890205847419 -6.937317600692943  
4 107.76746366976894 -6.9350597339851845  
5 107.76875113034612 -6.932844458040509
```

```
6 107.76845918670969 -6.934807259429284
7 107.76989457485271 -6.934398831304287
8 107.77209935061252 -6.933642655495336
9 107.77332243796953 -6.933376396111233
10 107.77636406292797 -6.9325509910896495
11 107.77102634738388 -6.93694871279362
12 107.7691657974542 -6.938476139066489
13 107.77346532655446 -6.942770878300951
14 107.77602197752643 -6.939400308730547
```

Keluaran dalam terminal:

a. Algoritma UCS

```
=====
Daftar peta yang tersedia dalam folder test:
1 adaKosong
2 alun-alun
3 bandungSelatan
4 bandungUtara
5 nangor
6 turki
Pilih nomor file yang ingin dibuka: 5
File berhasil dibaca

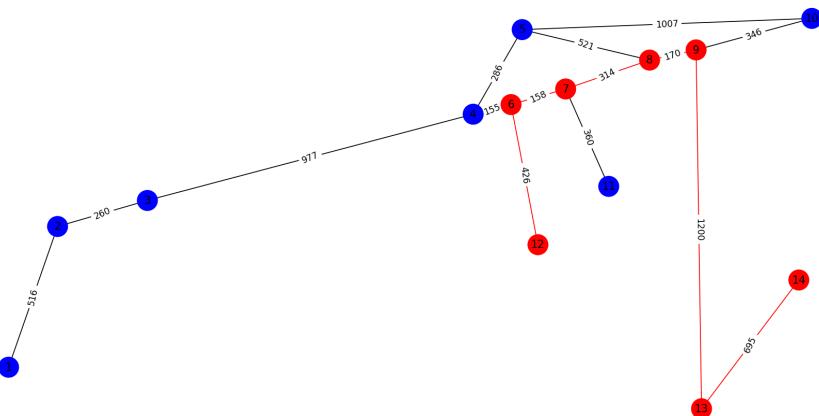
Pilih Algorithm yang ingin digunakan:
1. UCS
2. A*
Masukkan pilihan: 1
Kamu memilih UCS
DAFTAR NODE:
1 Terminal
2 Pintu Tol
3 Pasar Resik
4 Puteran ITB
5 ITB
6 Caringin
7 Kp Geulis
8 Puteran Mixue
9 Jalan Sayang
10 Gerlam
11 Kos Yunis
12 Wasabi
13 Puri Indah
14 Kos Salman
Masukkan Node Awal: 14
Masukkan Node Tujuan: 12
Hasil UCS:
Jarak: 2963 Meter
Kos Salman -> Puri Indah -> Jalan Sayang -> Puteran Mixue -> Kp Geulis -> Caringin -> Wasabi
Visualisasi Graf akan ditampilkan
Tutup visualisasi untuk melanjutkan
```

b. Algoritma A*

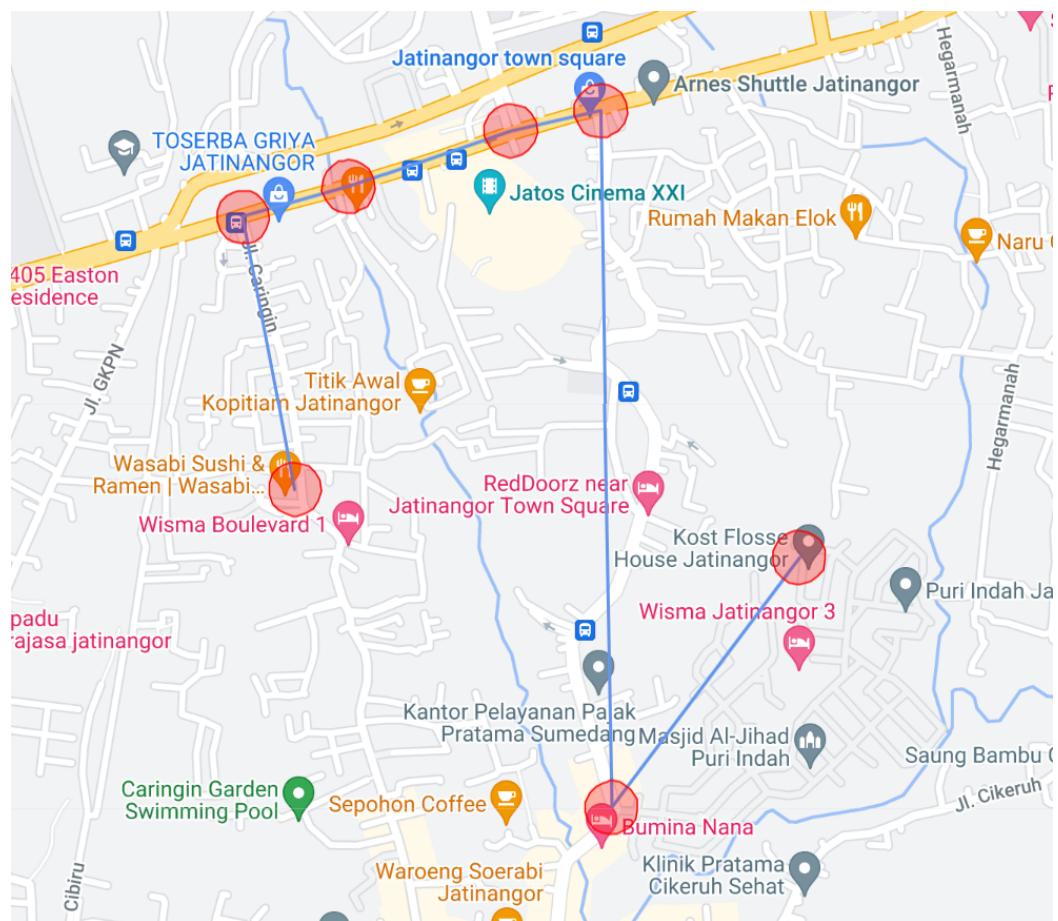
```
=====
Daftar peta yang tersedia dalam folder test:
1 adakosong
2 alun-alun
3 bandungSelatan
4 bandungUtara
5 nangor
6 turki
Pilih nomor file yang ingin dibuka: 5
File berhasil dibaca

Pilih Algorithm yang ingin digunakan:
1. UCS
2. A*
Masukkan pilihan: 2
Kamu memilih A*
DAFTAR NODE:
1 Terminal
2 Pintu Tol
3 Pasar Resik
4 Puteran ITB
5 ITB
6 Caringin
7 Kp Geulis
8 Puteran Mixue
9 Jalan Sayang
10 Gerlam
11 Kos Yunis
12 Wasabi
13 Puri Indah
14 Kos Salman
Masukkan Node Awal: 14
Masukkan Node Tujuan: 12
Hasil A*:
Jarak: 2963.0 Meter
Kos Salman -> Puri Indah -> Jalan Sayang -> Puteran Mixue -> Kp Geulis -> Caringin -> Wasabi
Visualisasi Graf akan ditampilkan
Tutup visualisasi untuk melanjutkan
```

Keluaran dalam visualisasi:

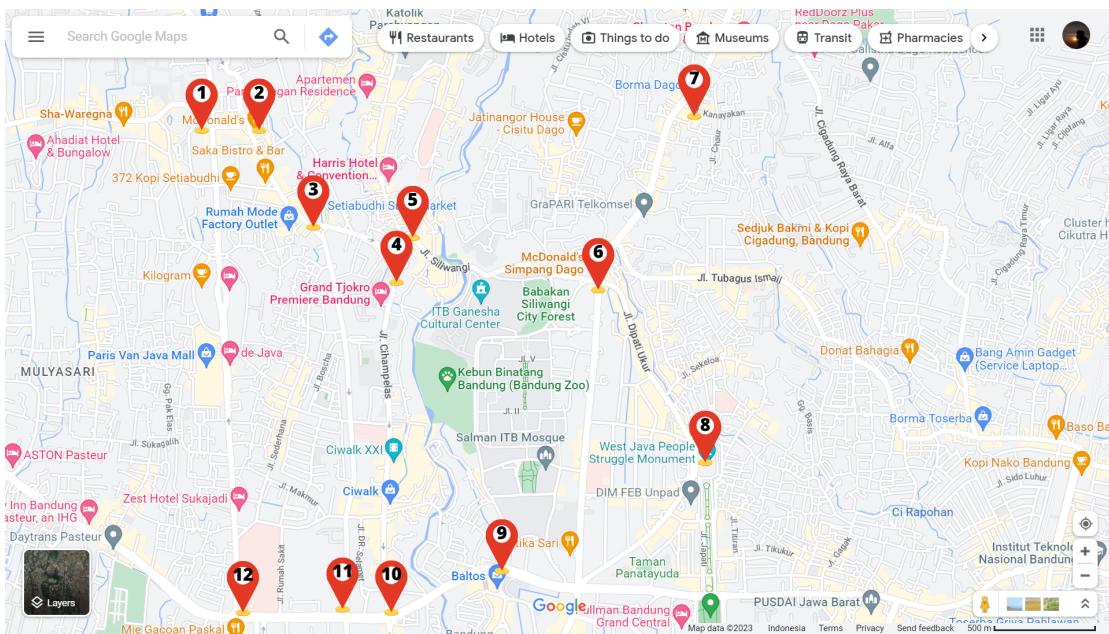


Keluaran dalam peta:



3.4 Hasil Pengujian dengan peta Bandung Utara

Bentuk Peta:



Graf input:

UNPAR,MCD UNPAR,Setia Budi Market,Grand Tjokro,Cihampelas,MCD Dago,Borma Dago,UNPAD DU,Baltos,Ciwalk,Boscha,Pasteru

0 157 0 0 0 0 0 0 0 0 2410
157 0 598 0 0 0 0 0 0 0 0
0 598 0 438 0 0 0 0 0 0 1950 0
0 0 438 0 192 0 0 0 0 1700 0 0
0 0 0 192 0 500 0 0 0 0 0 0
0 0 0 0 500 0 1000 1030 1501 0 0 0
0 0 0 0 0 1000 0 0 0 0 0 0
0 0 0 0 0 1030 0 0 860 0 0 0
0 0 0 0 0 1501 0 860 0 960 0 0
0 0 0 1700 0 0 0 0 960 0 227 0
0 0 1950 0 0 0 0 0 0 227 0 502
2410 0 0 0 0 0 0 0 0 0 502 0
1 107.59617516817069 -6.878250868322604
2 107.59772012054502 -6.878250868322604
3 107.60068127926249 -6.882639315099353
4 107.60462949088577 -6.88470570560123
5 107.60497281363561 -6.8832997089183445
6 107.61362025551486 -6.88519567318329

```
7 107.61761138253057 -6.8771857051969665  
8 107.61801907833657 -6.8932481112060815  
9 107.6127404910264 -6.898850652667312  
10 107.60220477404731 -6.900533532435832  
11 107.60226914700256 -6.899958371706619  
12 107.59748408617651 -6.9002139987935625
```

Keluaran dalam terminal:

a. Algoritma UCS

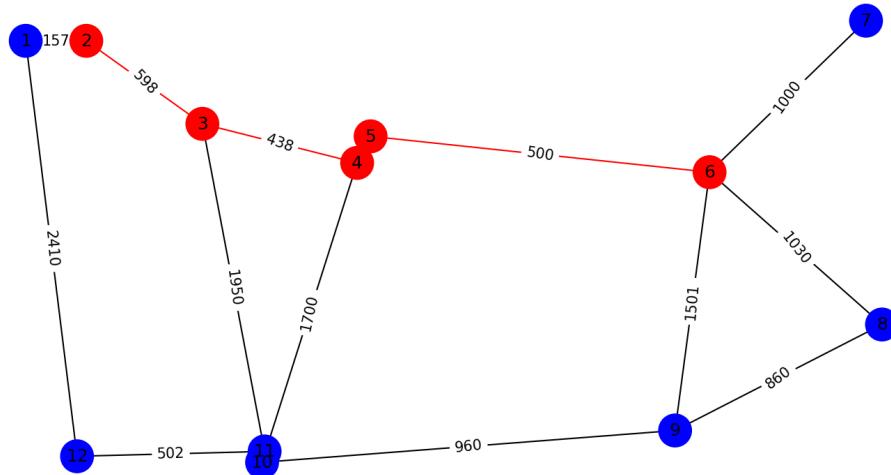
```
Daftar peta yang tersedia dalam folder test:  
1 adakosong  
3 bandungSelatan  
4 bandungUtara  
5 nangor  
6 turki  
Pilih nomor file yang ingin dibuka: 4  
File berhasil dibaca  
  
Pilih Algorithm yang ingin digunakan:  
1. UCS  
2. A*  
Masukkan pilihan: 1  
Kamu memilih UCS  
DAFTAR NODE:  
1 UNPAR  
2 MCD UNPAR  
3 Setia Budi Market  
4 Grand Tjokro  
5 Cihampelas  
6 MCD Dago  
7 Borma Dago  
8 UNPAD DU  
9 Baltos  
10 Ciwalk  
11 Boscha  
12 Pasteru  
Masukkan Node Awal: 2  
Masukkan Node Tujuan: 6  
Hasil UCS:  
Jarak: 1728 Meter  
MCD UNPAR -> Setia Budi Market -> Grand Tjokro -> Cihampelas -> MCD Dago
```

b. Algortima A*

```
=====
Daftar peta yang tersedia dalam folder test:
1 adaKosong
2 alun-alun
3 bandungSelatan
4 bandungUtara
5 nangor
6 turki
Pilih nomor file yang ingin dibuka: 4
File berhasil dibaca

Pilih Algorithm yang ingin digunakan:
1. UCS
2. A*
Masukkan pilihan: 2
Kamu memilih A*
DAFTAR NODE:
1 UNPAR
2 MCD UNPAR
3 Setia Budi Market
4 Grand Tjokro
5 Cihampelas
6 MCD Dago
7 Borma Dago
8 UNPAD DU
9 Baltos
10 Ciwalk
11 Boscha
12 Pasteru
Masukkan Node Awal: 2
Masukkan Node Tujuan: 6
Hasil A*:
Jarak: 1728.0 Meter
MCD UNPAR -> Setia Budi Market -> Grand Tjokro -> Cihampelas -> MCD Dago
```

Keluaran dalam visualisasi:



BAB 4

Penutup

4.1 Kesimpulan

Berdasarkan hasil penggerjaan tugas besar pencarian lintasan terpendek, dapat disimpulkan bahwa penentuan lintasan terpendek dapat dilakukan dengan algoritma Uniform Cost Search dan algoritma A*. Algoritma UCS akan mengevaluasi semua kemungkinan dan mencari jalur dengan jarak yang lebih sedikit. Algoritma A* akan mengevaluasi setiap kemungkinan dan mengeliminasi jarak yang lebih besar di setiap tetangganya. Hal ini menyebabkan algoritma A* memiliki efisiensi yang lebih baik dibandingkan dengan algoritma UCS, jika jarak euclidean yang diberikan valid.

4.2 Komentar

Tugas kecil yang menurut kami tidak kecil ini memberikan banyak manfaat bagi penulis mengenai penggunaan algoritma A* dan UCS serta penggunaan API Google Maps. Dalam penggerjaan tugas kecil ini sangat disayangkan kami tidak memaksimalkan penggunaan API karena untuk untuk penggunaan API maksimal diperlukan biaya yang besar.

BAB 5

Lampiran

5.1 Tautan *repository* GitHub

https://github.com/agshaathalla/Tucil3_13521011_13521027.git

5.2 Cek list

Poin	Ya	Tidak
Program dapat menerima input graf	✓	
Program dapat menghitung lintasan terpendek dengan UCS	✓	
Program dapat menghitung lintasan terpendek dengan A*	✓	
Program dapat menampilkan lintasan terpendek serta jaraknya	✓	
Bonus: Program dapat menerima input peta dengan Google Map API dan menampilkan peta serta lintasan terpendek pada peta	✓	✓