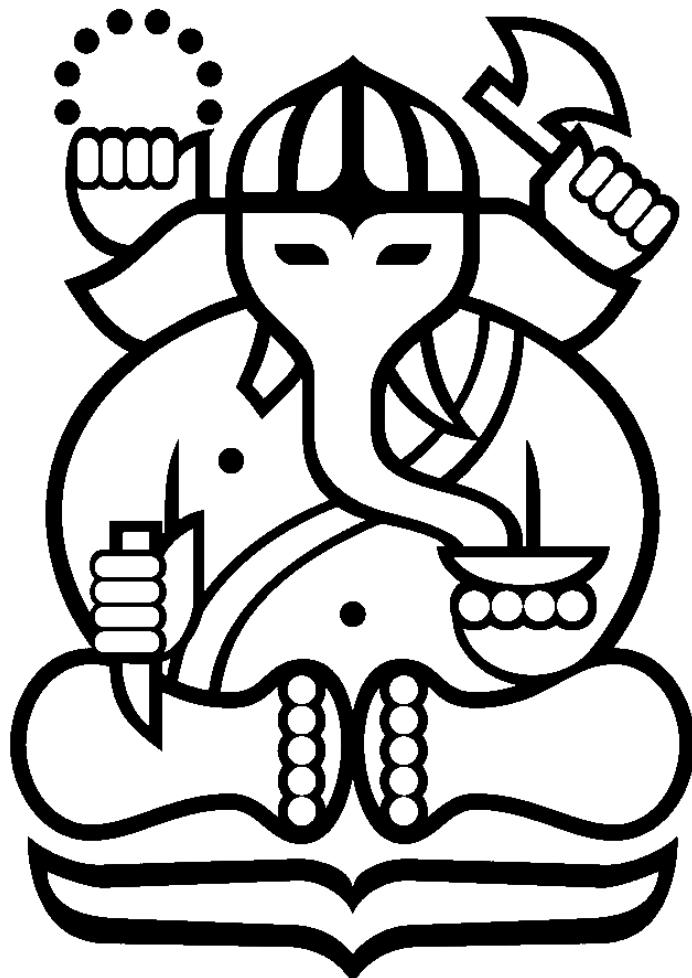


Tugas Kecil 3 IF2211 Strategi Algoritma

Implementasi Algoritma UCS dan A untuk Menentukan
Lintasan Terpendek*



Disusun oleh :
13521072 - Irsyad Nurwidianto Basuki
13521081 - Bagas Aryo Seto

INSTITUT TEKNOLOGI BANDUNG
2023

DAFTAR ISI

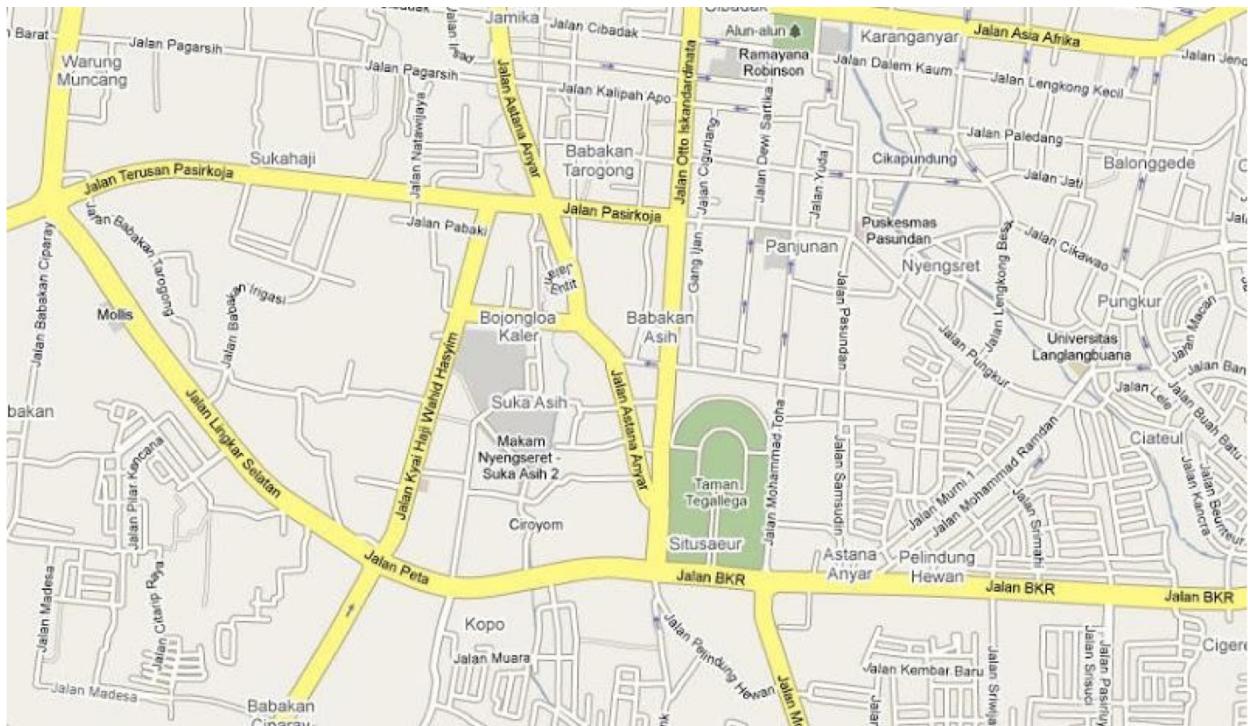
BAB I DESKRIPSI MASALAH DAN ALGORITMA	1
1.1 Deskripsi Masalah	1
1.2 Algoritma	2
1.2.1 Algoritma UCS dan A*	2
1.2.2 Algoritma Penyelesaian Implementasi Algoritma UCS dan A* untuk Menentukan Lintasan Terpendek	2
BAB II SOURCE CODE PROGRAM	4
2.1 Pranala Program	4
2.2 Source Code Program	4
BAB III MASUKAN DAN LUARAN PROGRAM	10
BAB IV KESIMPULAN	24
4.1. Kesimpulan	24
4.2 Komentar	24
LAMPIRAN	25
5.1. Link Repository Github	25
5.2. Tabel Ketercapaian Program	25

BAB I

DESKRIPSI MASALAH DAN ALGORITMA

1.1 Deskripsi Masalah

Algoritma UCS (*Uniform cost search*) dan A* (atau *A star*) dapat digunakan untuk menentukan lintasan terpendek dari suatu titik ke titik lain. Pada tugas kecil 3 ini, anda diminta menentukan lintasan terpendek berdasarkan peta Google Map jalan-jalan di kota Bandung. Dari ruas-ruas jalan di peta dibentuk graf. Simpul menyatakan persilangan jalan (simpang 3, 4 atau 5) atau ujung jalan. Asumsikan jalan dapat dilalui dari dua arah. Bobot graf menyatakan jarak (m atau km) antar simpul. Jarak antar dua simpul dapat dihitung dari koordinat kedua simpul menggunakan rumus jarak Euclidean (berdasarkan koordinat) atau dapat menggunakan *ruler* di Google Map, atau cara lainnya yang disediakan oleh Google Map.



Langkah pertama di dalam program ini adalah membuat graf yang merepresentasikan peta (di area tertentu, misalnya di sekitar Bandung Utara/Dago). 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.

1.2 Algoritma

1.2.1 Algoritma UCS dan A*

Uniform-Cost Search adalah sebuah algoritma pencarian tanpa informasi yang menggunakan biaya kumulatif terendah untuk mencari jalur dari suatu node sumber ke node tujuan dalam graf berbobot.

UCS merupakan salah satu jenis algoritma pencarian uninformed search atau blind search karena tidak mempertimbangkan keadaan node atau ruang pencarian. Algoritma ini digunakan untuk menemukan jalur dengan biaya kumulatif terendah dalam graph berbobot di mana node diperluas sesuai dengan biaya traversalnya dari node root.

Untuk implementasi algoritma UCS, biasanya digunakan priority queue yang menurunkan biaya operasi sebagai prioritas.

Algoritma A Star adalah salah satu algoritma pencarian graf dengan menggunakan fungsi jarak-plus-biaya untuk menentukan urutan titik yang akan dikunjungi. Dalam permasalahan ini yang harus dipecahkan adalah bagaimana cara agar bisa mengunjungi tempat yang dituju dengan jarak, waktu dan biaya yang minimum

1.2.2 Algoritma Penyelesaian Implementasi Algoritma UCS dan A* untuk Menentukan Lintasan Terpendek

Algoritma UCS akan memeriksa simpul yang memiliki rute dari simpul awal terlebih dahulu serta skor dari rute simpul tersebut. Untuk menentukan simpul yang akan dicek selanjutnya, dibuat priority queue yang diurutkan berdasarkan skor atau *weight* terkecil. Simpul selanjutnya yang diperiksa akan diambil dari depan priority queue. Simpul tersebut akan diperiksa seperti simpul awal dengan skor yang didapatkan oleh simpul merupakan skor total dari simpul awal sampai simpul tersebut. Simpul-simpul yang diberikan akan terus ditelusuri berdasarkan priority queue yang dibuat sampai ditemukan simpul yang dicari. Ketika simpul yang dicari ditemukan, proses pencarian berhenti.

Algoritma A* mencari lintasan terpendek dengan menggabungkan dua faktor, yaitu biaya aktual dari titik awal ke suatu simpul tertentu dan perkiraan biaya yang tersisa dari simpul tersebut ke titik tujuan. Algoritma ini memanfaatkan fungsi heuristik untuk memperkirakan biaya yang tersisa dari simpul tertentu ke titik tujuan.

Proses pencarian dimulai dengan memasukkan simpul awal ke dalam antrian prioritas, di mana simpul awal memiliki biaya aktual 0 dan biaya perkiraan ke titik tujuan. Selanjutnya, simpul dengan biaya perkiraan terkecil akan dipilih untuk dieksplorasi. Kemudian, simpul-simpul yang berdekatan dengan simpul yang sedang dieksplorasi akan dianalisis untuk menentukan apakah akan memperpanjang jalur atau tidak. Setelah itu, simpul-simpul yang baru ditemukan akan dimasukkan ke dalam antrian prioritas.

Selama pencarian berlangsung, A* akan selalu mempertahankan antrian prioritas yang mengandung semua simpul yang harus dieksplorasi berdasarkan prioritas mereka. Ketika simpul tujuan akhir terdeteksi, maka lintasan yang ditempuh dari titik awal ke simpul tujuan akhir akan diberikan sebagai output.

Proses pencarian dilakukan secara heuristik, yang berarti bahwa A* hanya akan mengeksplorasi simpul-simpul yang memiliki biaya aktual yang lebih rendah dan perkiraan biaya yang lebih kecil dari simpul yang telah dikunjungi. Algoritma ini menggabungkan biaya aktual dan perkiraan biaya dengan heuristik untuk menemukan jalur terpendek dari titik awal ke titik tujuan.

BAB II

SOURCE CODE PROGRAM

2.1 Pranala Program

Pranala *repository* dan Google Drive :
github.com/bagas003/Tucil3_13521072_13521081

2.2 Source Code Program

	ucs.py
<pre>import queue def ucs(graph, start, goal): prioQueue = queue.PriorityQueue() prioQueue.put((0, start, [])) # cost = 0 , starting node, path = " " visited = set() while not prioQueue.empty(): (totalCost, currentNode, path) = prioQueue.get() # Get the front of queue based on its weight if currentNode in visited: #currentNode already visited, continue next iteration continue if currentNode == goal: return [start] + path visited.add(currentNode) for nextNode in graph.neighbors(currentNode): tempCost = graph.get_edge_data(currentNode, nextNode)['weight'] if nextNode not in visited: nextCost = totalCost + tempCost newPath = path.copy() # make a copy of the current path before updating it newPath += [nextNode] # add the next node to the new path prioQueue.put((nextCost, nextNode, newPath)) return None</pre>	
	main.py

```

import astar
import ucs
from utility import *

while True:
    inp = 0

    filename = input("\nEnter file name:\n> ")
    try:
        graph = readFile(filename)
    except ValueError:
        print("There is something wrong with your input file!")
        continue
    except FileNotFoundError:
        print("No file .txt found, Please check your file name or its directory.")
        continue
    except IsADirectoryError:
        print(("Please enter your file name!"))
        continue

    while True:
        start, goal = input_destination(graph)

        inputAlg = int(input("\n Choose your algorithm :\n1. UCS \n2. A* (A Star)\n>"))

        if(inputAlg == 1):
            path = ucs.ucs(graph, start, goal)
        elif (inputAlg == 2):
            path = astar.a_star(graph, start, goal)
        else:
            print("Invalid input. Try again.")
            continue

        print_path(path)
        print(f"Shortest path distance: {get_total_dist(graph, path):.2f} m")
        if input("\nShow path? (Y/N):\n> ").lower().find("y") != -1:
            show_path(graph, path)

        temp = input("\nContinue?\n1. Continue with the same graph\n2. Continue with different graph\n3. Quit\n> ")
        while not temp.isnumeric() or int(temp) < 1 or int(temp) > 3:
            print("Invalid input!")
            temp = input("\nContinue?\n1. Continue with the same graph\n2. Continue with different graph\n3. Quit\n> ")

        inp = int(temp)
        if inp == 2 or inp == 3:
            break

        if inp == 3:
            break

```

utility.py

```

import networkx as nx
import matplotlib.pyplot as plt
import math

def dist(graph, node1, node2):
    return math.dist(graph.nodes[node1]['pos'], graph.nodes[node2]['pos'])

def readFile(filename):
    filename = "test/" + filename
    f = open(filename.encode('unicode_escape').decode().replace("\u20ac", ""), 'r')
    raw = f.readlines()

    graph = nx.Graph()

    n = int(raw[0])
    for i in range(1,n+1):
        name = raw[2*i-1].replace("\n","")
        [y, x] = [float(i)*111139 for i in raw[2*i].replace(",","",).split(" ")]
        graph.add_node(name, pos=(x, y))

    nodes = list(graph.nodes())

    for i in range(n*2+1, n*3):
        temp = [int(j) for j in raw[i].split(" ")]
        for j in range(n):
            if temp[j] == 1:
                n1, n2 = nodes[i-n*2-1], nodes[j]
                w = dist(graph, n1, n2)
                graph.add_edge(n1, n2, weight=w)

    return graph

def show(graph):
    coor = nx.get_node_attributes(graph, 'pos')
    nx.draw(graph, coor, with_labels=True)
    edge_labels = {(u, v): f"{w:.2f}" for u, v, w in graph.edges(data='weight')}
    nx.draw_networkx_edge_labels(graph, pos=coor, edge_labels=edge_labels)
    plt.grid()
    plt.show()

```

```
def show_path(graph, path):
    coor = nx.get_node_attributes(graph, 'pos')

    path_graph = nx.Graph()
    path_graph.add_node(path[0], pos=coor[path[0]])
    for i in range(1, len(path)):
        path_graph.add_node(path[i], pos=coor[path[i]])
        path_graph.add_edge(path[i-1], path[i], weight=dist(graph, path[i-1], path[i]))

    path_coors = nx.get_n (variable) path_coors: Any 'pos'
    nx.draw(graph, coor, path_coors
    nx.draw(path_graph, path_coors, edge_color='red', node_color='red')
    edge_labels = {(u, v): f"{w:.2f}" for u, v, w in graph.edges(data='weight')}
    nx.draw_networkx_edge_labels(graph, pos=coor, edge_labels=edge_labels)
    plt.show()
```

```

def input_destination(graph):
    nodes = list(graph.nodes())

    print("\nNodes list:")

    i = 1
    for node in nodes:
        print(f"{i}. {node}")
        i += 1

    n1 = 0
    n2 = 0

    while n1 == 0:
        temp = input("\nEnter starting point number.\nEnter 0 to show graph\n>> ")
        if not temp.isnumeric() or int(temp) < 0 or int(temp) > len(nodes):
            print("Invalid input!")
            continue

        n1 = int(temp)
        if n1 == 0:
            show(graph)

    while n2 == 0:
        temp = input("\nEnter goal point number.\nEnter 0 to show graph\n>> ")
        if not temp.isnumeric() or int(temp) < 0 or int(temp) > len(nodes):
            print("Invalid input!")
            continue

        n2 = int(temp)
        if n2 == 0:
            show(graph)

    return nodes[n1-1], nodes[n2-1]

def print_path(path):
    print('\nShortest path: ', end=' ')
    if path != None:
        print(path[0], end=' ')
        for i in range(1, len(path)):
            print(' - ', path[i], end=' ')
    else:
        print('None')
    print()

```

```
def get_total_dist(graph, path):
    d = 0
    for i in range(1, len(path)):
        d += graph.get_edge_data(path[i-1], path[i])['weight']
    return d

if __name__ == "__main__":
    # create a graph
    G = nx.Graph()
    G.add_edges_from([(1, 2), (2, 3), (3, 1)])

    # draw the graph
    pos = nx.spring_layout(G)
    nx.draw(G, pos)

    # draw grid lines
    plt.grid(True, linestyle='-', color='0.75', linewidth=1)

    # set the drawing order of the grid lines
    plt.gca().set_axisbelow(True)

    # show the plot
    plt.show()
```

BAB III

MASUKAN DAN LUARAN PROGRAM

Map Alun Alun Bandung
22
museum konferensi asia afrika
-6.921459543322524, 107.60980041967157
asia afrika & sukarno
-6.921351574787512, 107.60873396370731
simpang monumen asia afrika
-6.92126422152903, 107.60802347546638
masjid raya bandung
-6.921020673989582, 107.60646661706087
banceuy & belakang factory
-6.920001958105287, 107.60656998996423
banceuy & abc st
-6.9188845256273614, 107.60668089282302
abc st & naripan
-6.919590417249778, 107.60834995581101
cikapundung barat & belakang factory
-6.920209459417882, 107.6081690841581
naripan & sukarno
-6.919729638682148, 107.60897141549356
naripan & braga
-6.91971687793879, 107.60994094349797
braga & kejaksaan
-6.91856896166219, 107.60962062289434
simpang keluar braga
-6.91615261820008, 107.60897866500093
el hotel royale bandung
-6.91652671983771, 107.61002786463581
lembong & telepon
-6.916993788817839, 107.61087955289882
telepon & kejaksaan
-6.918424318438478, 107.6108295623009
lembong & markoni
-6.917350767243049, 107.61158157568948
markoni & kejaksaan
-6.918485313019517, 107.61144665306023
football monument
-6.917936163465482, 107.61250319551031
kejaksaan & tamblong
-6.918663429278685, 107.61236354274216
naripan & tamblong
-6.920108881036783, 107.61218074929876
tamblong & asia afrika
-6.921733451561692, 107.61195596030652
markoni & naripan

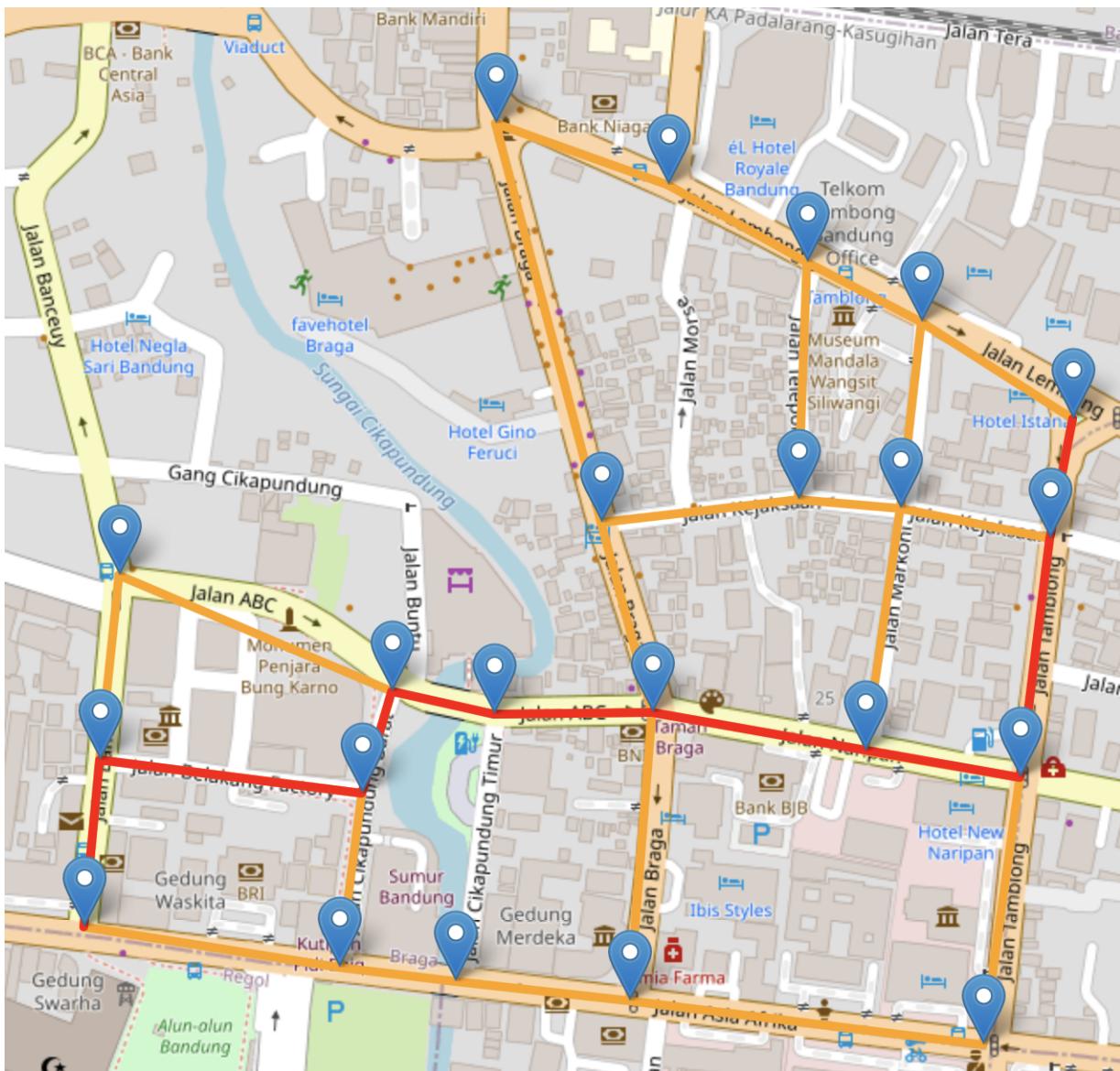
```
-6.919954165085783, 107.61123947012125
0100000001000000000010
1010000000000000000000
0101000000000000000000
0010100000000000000000
0001010100000000000000
0000101010000000000000
0000010110000000000000
0010101000000000000000
0000001001000000000000
10000000010100000000100
0000000001010000000000
0000000000010100000000
0000000000001010000000
0000000000000100000000
0000000000000010011000
00000000000000011001001
00000000000000001001000
00000000000000001101000
00000000000000000110100
0000000000000000001011
10000000000000000000100
00000000001000000000100
```

```
Choose your algorithm :
```

- 1. UCS
 - 2. A* (A Star)
- >>1

```
Shortest path: masjid raya bandung - banceuy & belakang factory - cikapundung barat  
& belakang factory - abc st & naripan - naripan & sukarno - naripan & braga - nari  
pan & tamblong - kejaksaan & tamblong - football monument
```

```
Shortest path distance: 1040.17 m
```



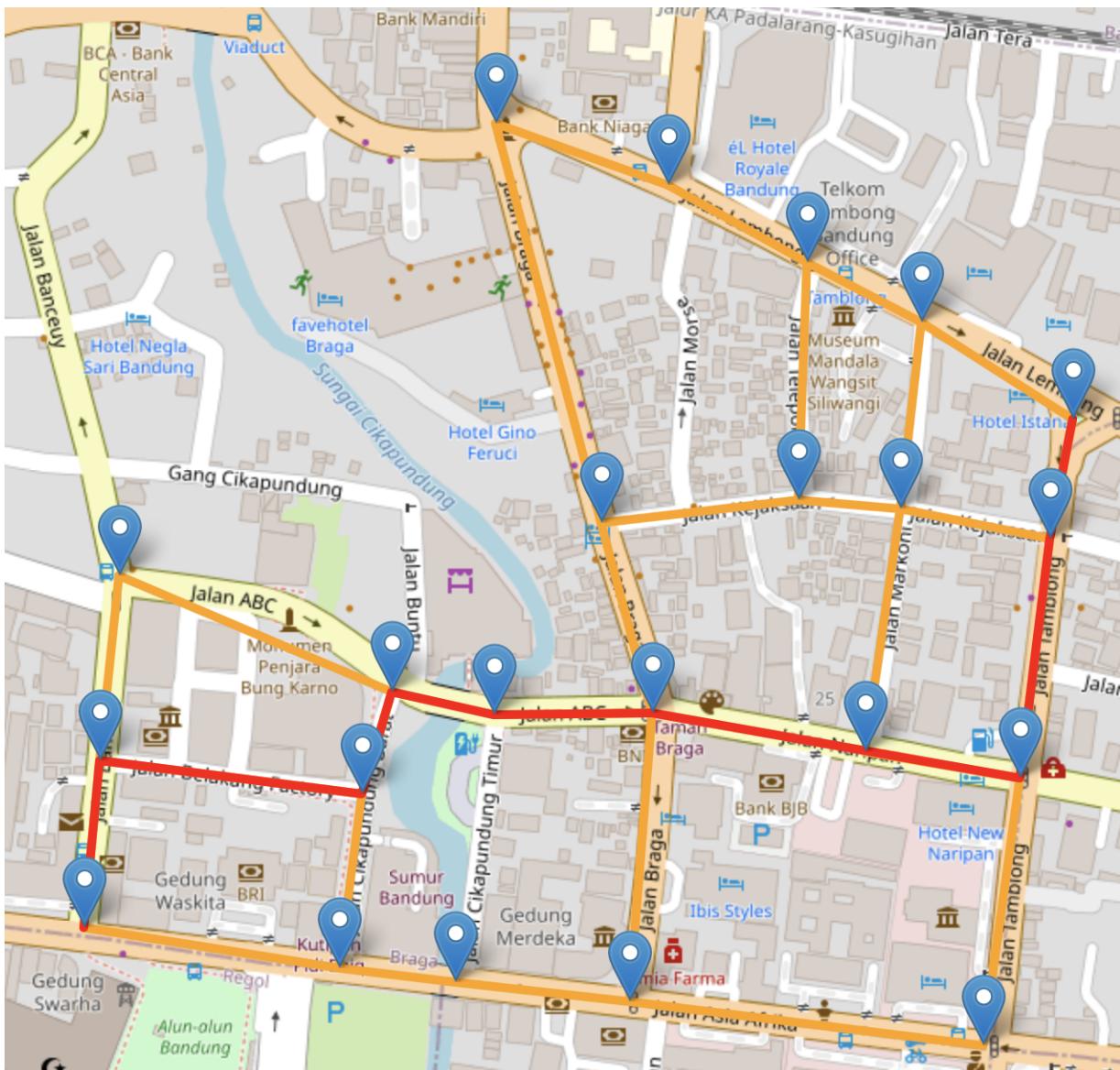
Choose your algorithm :

1. UCS
 2. A* (A Star)
- >>2

Shortest path: masjid raya bandung - banceuy & belakang factory - cikapundung barat & belakang factory - abc st & naripan - naripan & sukarno - naripan & braga - nari pan & tamblong - kejaksaan & tamblong - football monument

Shortest path distance: 1040.17 m

Visualize path? : (You'll need to re-run the program if you visualize its path)



Map Cisitu

25

simpang sadikin

-6.879290656604597 107.61258245513986

simpang alpina

-6.879319790375955 107.61283800273434

simpang indomaret cisitu

-6.879616251410727 107.61202593243291

simpang asrama kidang pananjung itb

-6.879807224382633 107.61430888221496
nasgor portugal
-6.876734 107.612927
simpang dermaga kopi
-6.882836344622385 107.61167076619141
simpang nasgor tombo lapar
-6.882910049424211 107.61125540465267
simpang sangkuring cisitu
-6.883734705833403 107.61123685708792
simpang cisitu lama baru
-6.883378937465031 107.61125103709509
simpang cisitu baru
-6.883382862085628 107.61200628053746
simpang cisitu dago
-6.884427161864515 107.61379914556336
simpang dago kidang pananjung
-6.883686378119053 107.61405563792717
simpang kozi
-6.882448573870303 107.61391896977317
simpang cisgar
-6.879935086874405 107.61475841837175
simpang cisitu indah 2 3
-6.8764852225931214 107.61207900302091
simpang cisitsu indah 3 4
-6.87594358956996 107.61223036489227
simpang cili korner
-6.87548812614364 107.61236063121113
gor cisitu
-6.8756134543451575 107.6131335337327
simpang cisitu indah dagas
-6.877107550223327 107.61472524305452
simpang alpina dagas 4
-6.8783213811870265 107.61321340059672

simpang dagas 2 4 5

-6.878674921513658 107.61435168305536

simpang dagas 2 raya

-6.878318056126893 107.61448180155028

simpang dagas gym

-6.8783914316229415 107.61527413949476

urban gym

-6.8775963166017 107.61536715478714

pintu dagas

-6.878724679388998 107.61652133410841

0 1 1 0 1 0

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

1 0 0 0 0 1 0

0 1 0

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

0 0 1 0 0 0 1 0

0 0 0 0 0 1 0 0 1 0

0 0 0 0 0 0 0 1 0

0 0 0 0 0 0 0 0 1 0

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

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

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

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

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

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

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

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

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

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

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

0 1 0 0 0 0 0 0 0 0

0 1 0 1 0 0 0 0 0

0 1 0 1 0 0 0 0

0 1 0 1 1 0 0

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
```

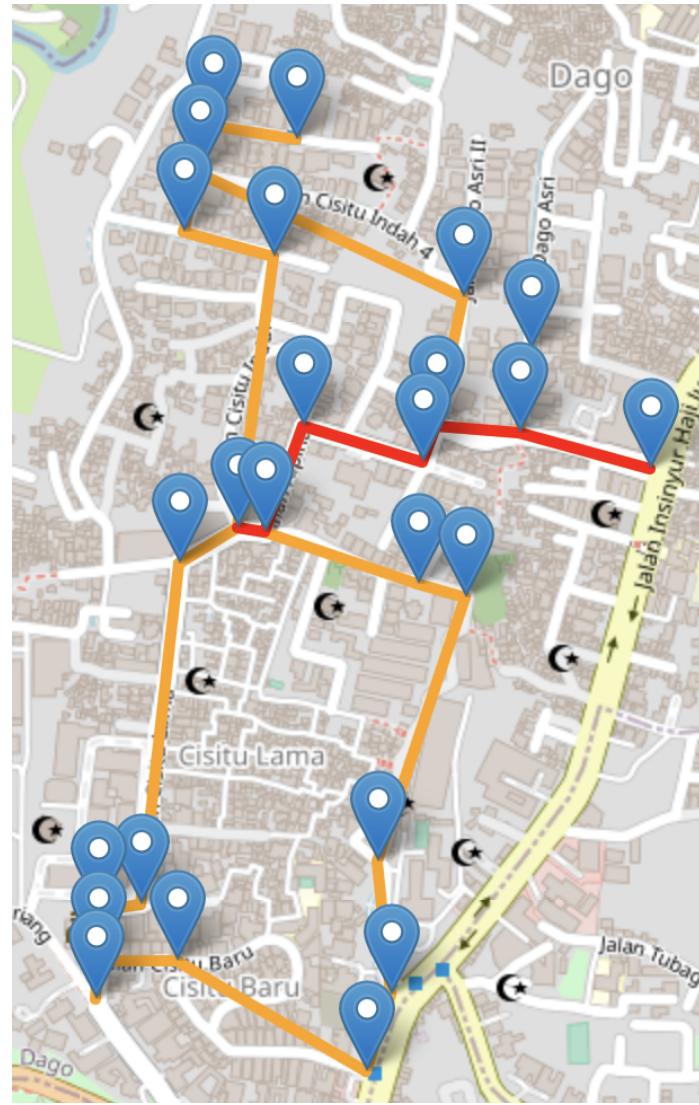
```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
```

Choose your algorithm :

- 1. UCS
 - 2. A* (A Star)
- >>1

Shortest path: pintu dagas - simpang dagas gym - simpang dagas 2 raya - simpang dag as 2 4 5 - simpang alpina dagas 4 - simpang alpina - simpang sadikin

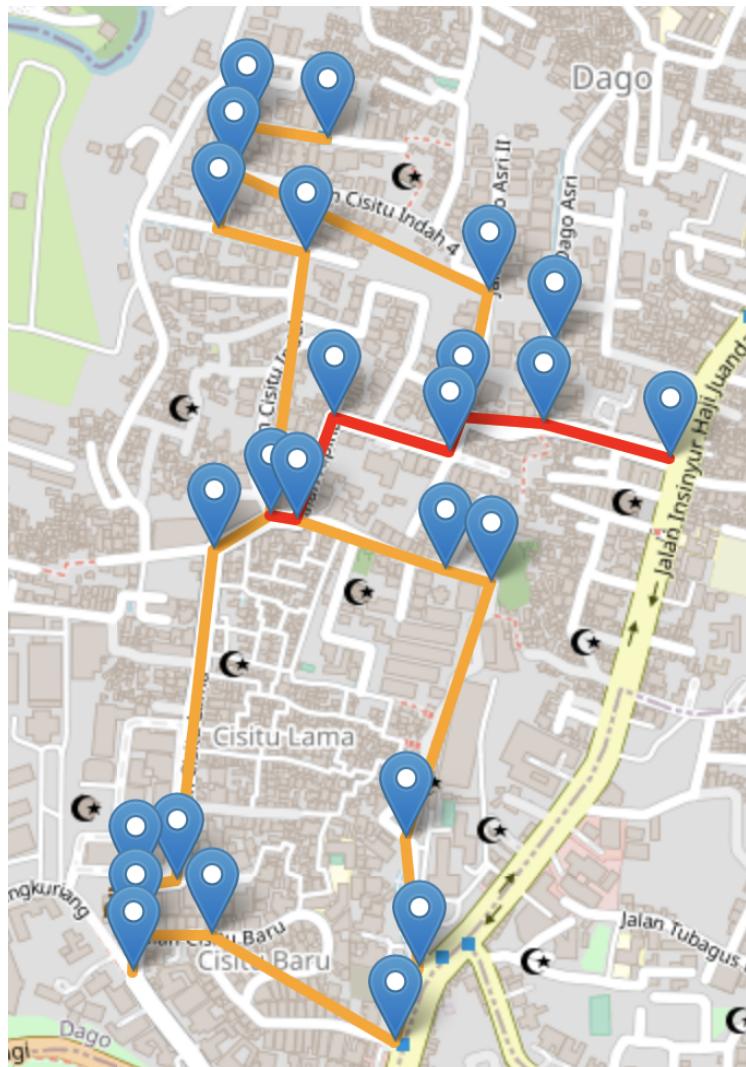
Shortest path distance: 553.73 m



Choose your algorithm :

1. UCS
 2. A* (A Star)
- >>2

Shortest path: pintu dagas - simpang dagas gym - simpang dagas 2 raya - simpang dag
as 2 4 5 - simpang alpina dagas 4 - simpang alpina - simpang sadikin
Shortest path distance: 553.73 m



Map Buah batu

10

metro indah mall

-6.940314601333511, 107.6581918440202

sop djanda metro bandung
-6.939077498135321, 107.66385859740213

borma rancabolang
-6.943180129634853, 107.66364615216605

simpag rancabolang & ciwastra
-6.958935999365598, 107.65958807396471

pertamina ciwastra
-6.955794652155699, 107.65522958657377

margacinta park
-6.955046165729226, 107.64761120546093

pln ulp cijawura
-6.954019500876578, 107.64023966933507

simpang ibrahim adjie & soetta
-6.945553582435032, 107.64179702718326

cijawura girang
-6.944348043940916, 107.64898613148235

cijwarua girang & soetta
-6.943925847355845, 107.6468618117993

0 1 0 0 0 0 0 0 1

1 0 1 0 0 0 0 0 0

0 1 0 1 0 0 0 0 0

0 0 1 0 1 0 0 0 0

0 0 0 1 0 1 0 0 0

0 0 0 0 1 0 1 0 1 0

0 0 0 0 0 1 0 1 0 0

0 0 0 0 0 0 1 0 0 1

0 0 0 0 0 0 1 0 0 1

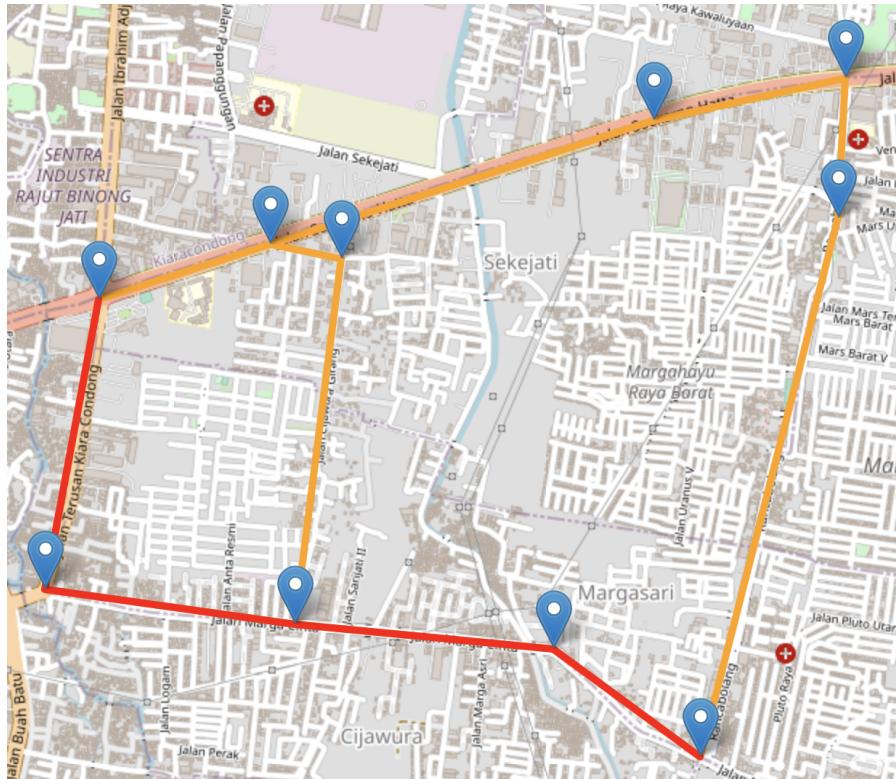
1 0 0 0 0 0 0 1 0 0

Choose your algorithm :

1. UCS
 2. A* (A Star)
- >>1

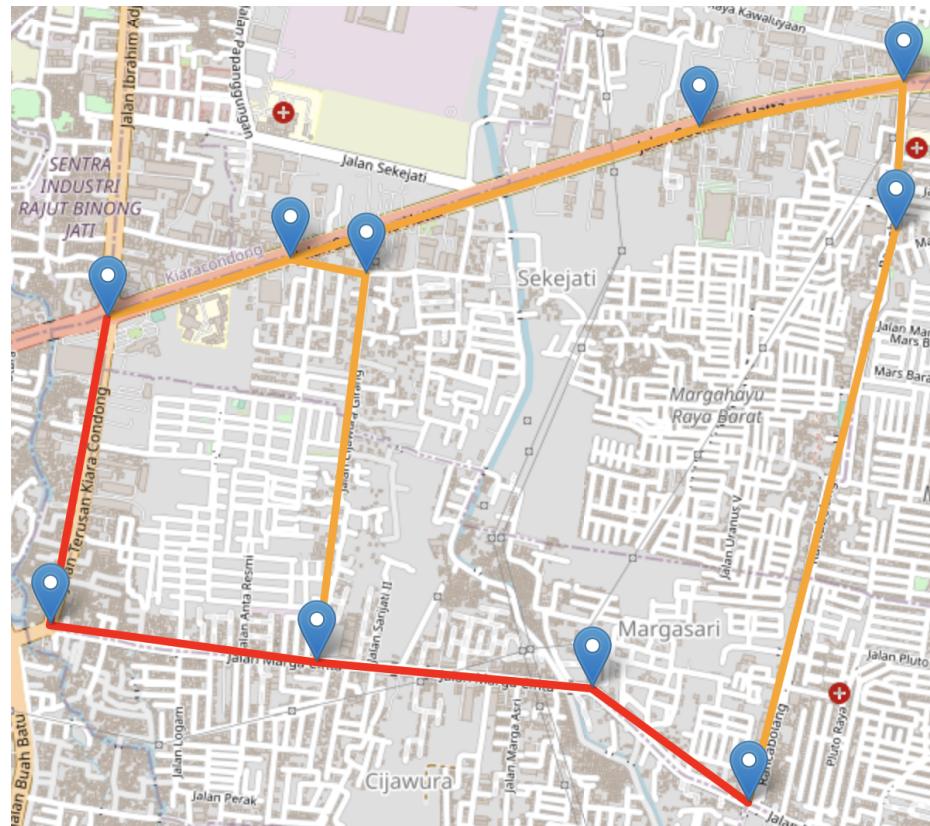
Shortest path: simpag rancabolang – ciwastra – pertamina ciwastra – margacinta park – pln ulp cijawura – simpang ibrahim adjie – soetta

Shortest path distance: 3231.73 m



Shortest path: simpag rancabolang & ciwastra – pertamina ciwastra – margacinta park – pln ulp cijawura – simpang ibrahim adjie & soetta

Shortest path distance: 3231.73 m



Map Sukoharjo

9

proliman

-7.682596871255136, 110.84275321385148

carikan

-7.681308608050543, 110.83221996052072

kejakaan

-7.671499248640465, 110.83869189211791

rsud

-7.684545506028019, 110.85019320411206

agus salim

-7.6905518588473765, 110.84654002179997

perum korpri

-7.69083828410905, 110.84910511922179

terminal

-7.694597597785216, 110.84884017958404

sritex

-7.68796774999181, 110.83012669144897

prapatan ringin

-7.689489696784349, 110.84032598292255

0 1 1 1 1 0 0 0 1

1 0 1 0 0 0 0 1 0

1 1 0 0 0 0 0 0 0

1 0 0 0 0 1 0 0 0

1 0 0 0 0 1 1 0 1

0 0 0 1 1 0 1 0 0

0 0 0 0 1 1 0 0 0

0 1 0 0 0 0 0 0 1

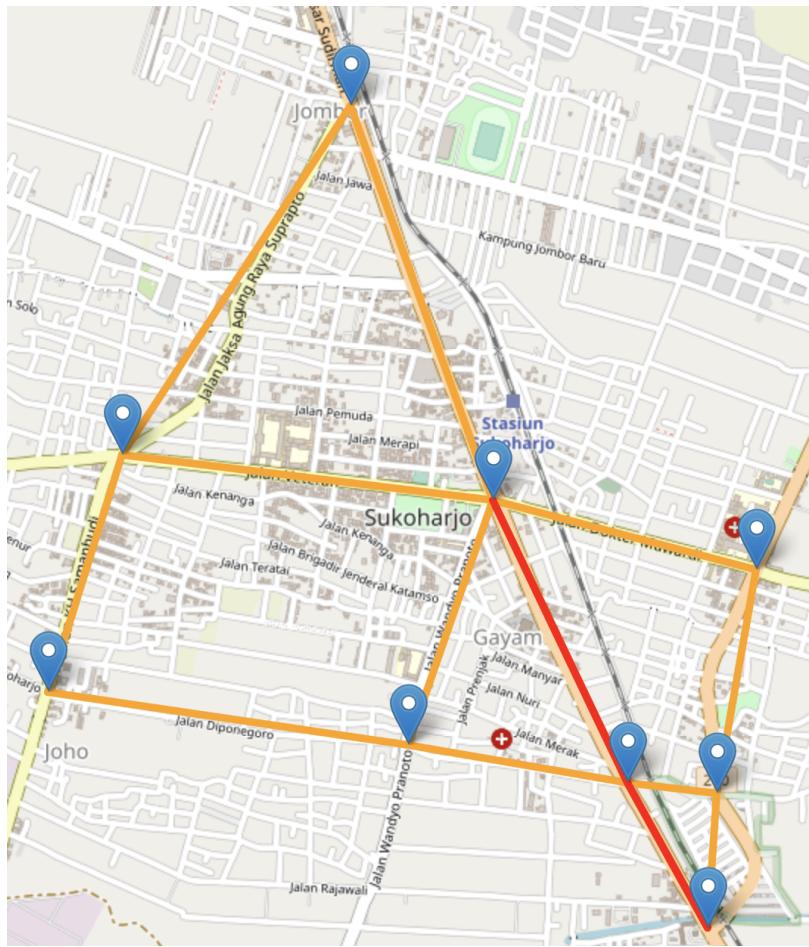
1 0 0 0 1 0 0 1 0

Choose your algorithm :

- 1. UCS
 - 2. A* (A Star)
- >>1

Shortest path: proliman – agus salim – terminal

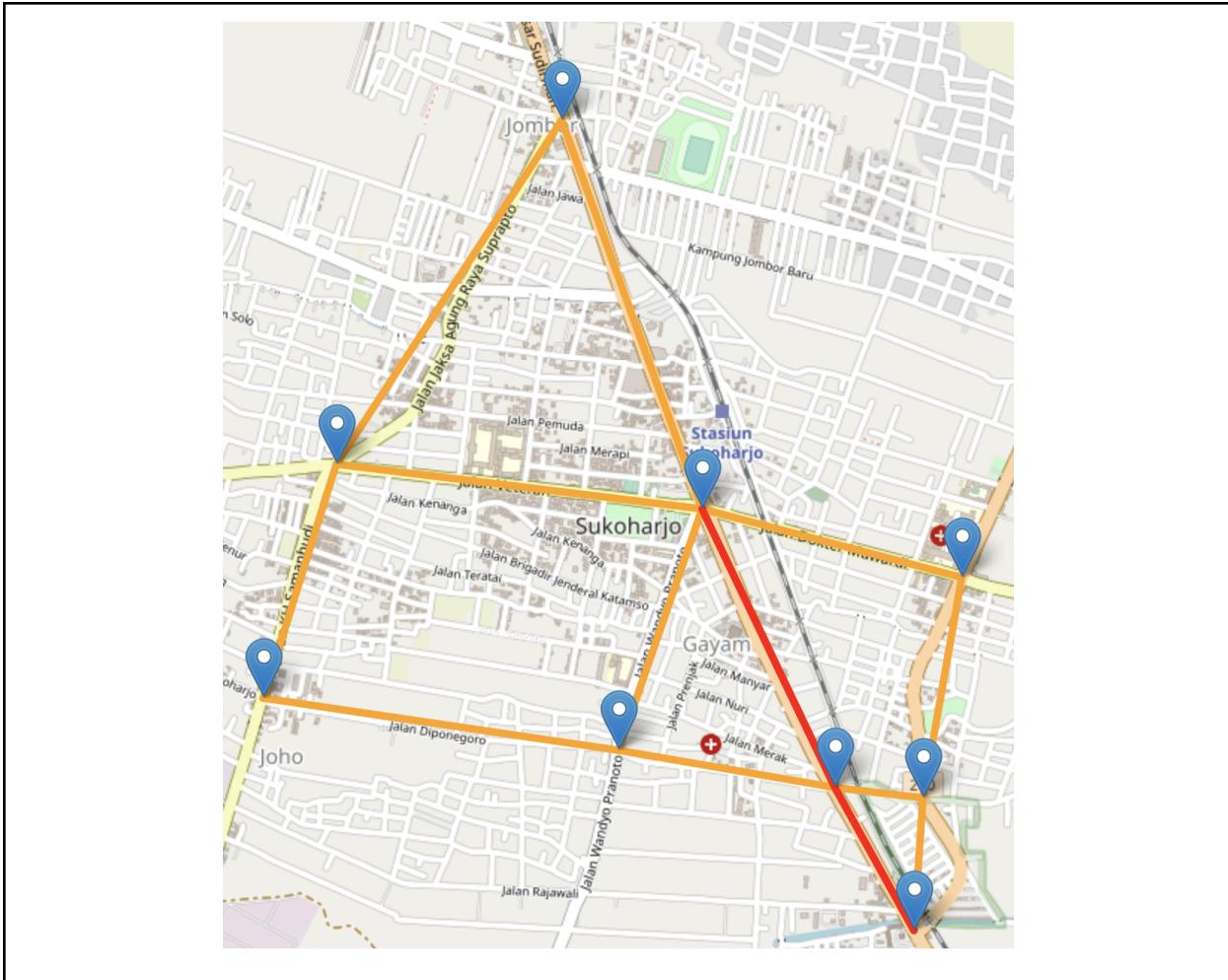
Shortest path distance: 1496.40 m



Choose your algorithm :

1. UCS
 2. A* (A Star)
- >>2

Shortest path: proliman - agus salim - terminal
Shortest path distance: 1496.40 m



BAB IV

KESIMPULAN

4.1. Kesimpulan

Algoritma UCS dan A* dapat digunakan untuk menemukan rute terpendek dengan solusi yang optimal serta *complete*. Algoritma UCS dan A* juga dapat mendeteksi apabila tidak ada jalur dari titik *start* menuju titik *finish*.

4.2 Komentar

Bonus tugas berupa penggunaan Google Map API cenderung menyulitkan mahasiswa yang tidak memiliki kartu kredit.

LAMPIRAN

Link Repository Github serta Google Drive :

github.com/bagas003/Tucil3_13521072_13521081

drive.google.com/drive/folders/1SDN178e_ofq8VCK0-rGX9WXkkXivbvc

Tabel Ketercapaian Program

Poin	Ya	Tidak
1. Program dapat menerima input graf	✓	
2. Program dapat menghitung lintasan terpendek dengan UCS	✓	
3. Program dapat menghitung lintasan terpendek dengan A*	✓	
4. Program dapat menampilkan lintasan terpendek serta jaraknya	✓	
5. Bonus: Program dapat menerima input peta dengan Google Map API		✓
6. Bonus: Program dapat menampilkan peta serta lintasan terpendek pada peta	✓	