

**Tugas Kecil III IF2211 Strategi Algoritma**

**IMPLEMENTASI ALGORITMA UCS DAN A\* UNTUK**

**MENENTUKAN LINTASAN TERPENDEK**

*Diajukan sebagai pemenuhan tugas kecil III.*



Oleh:

1. 13521052 - Melvin Kent Jonathan
2. 13521100 - Alexander Jason

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

**2023**

## DAFTAR ISI

<b>DAFTAR ISI</b>	<b>2</b>
<b>BAB 1</b>	
<b>DESKRIPSI MASALAH</b>	<b>3</b>
<b>BAB 2</b>	
<b>LANDASAN TEORI</b>	<b>4</b>
2.1. Uniform Cost Search (UCS)	4
2.2. A Star (A*)	4
<b>BAB 3</b>	
<b>IMPLEMENTASI ALGORITMA UCS DAN A*</b>	<b>5</b>
3.1. Alur Program	5
3.2. Struktur File	5
3.3. Algoritma UCS dan A*	5
<b>BAB 4</b>	
<b>PENGUJIAN</b>	<b>8</b>
4.1. Peta/Graf Input	8
4.2. Hasil Pengujian	8
4.2.1. Pengujian 1	8
4.2.2. Pengujian 2	11
4.2.3. Pengujian 3	14
4.2.4. Pengujian 4	17
4.2.5. Pengujian 5 (Bonus)	20
4.2.6. Pengujian 6 (Bonus)	24
4.2.7. Pengujian 7 (Bonus)	25
4.2.8. Pengujian 8 (Bonus)	29
<b>BAB 5</b>	
<b>PENUTUP</b>	<b>33</b>
5.1. Kesimpulan	33
5.2. Saran	33
5.3. Program Checkpoint	33
<b>LAMPIRAN</b>	<b>35</b>

## BAB 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.

2 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.

Spesifikasi program:

1. Program menerima input file graf (direpresentasikan sebagai matriks ketetanggan berbobot), jumlah simpul minimal 8 buah.
2. Program dapat menampilkan peta/graf
3. Program menerima input simpul asal dan simpul tujuan.
4. Program dapat menampilkan lintasan terpendek beserta jaraknya antara simpul asal dan simpul tujuan.
5. Antarmuka program bebas, apakah pakai GUI atau command line saja.

Bonus: Bonus nilai diberikan jika dapat menggunakan Google Map API untuk menampilkan peta, membentuk graf dari peta, dan menampilkan lintasan terpendek di peta (berupa jalan yang diberi warna). Simpul graf diperoleh dari peta (menggunakan API Google Map) dengan mengklik ujung jalan atau persimpangan jalan, lalu jarak antara kedua simpul dihitung langsung dengan rumus Euclidean.

## BAB 2

### LANDASAN TEORI

#### 2.1. Uniform Cost Search (UCS)

UCS menggunakan struktur data graf. Graf adalah representasi visual dari objek dan relasi di antara mereka. Graf terdiri dari simpul (*node*) yang mewakili objek, dan sisi (*edge*) yang mewakili relasi antara objek tersebut. Setiap sisi pada graf memiliki bobot (*weight*) yang menunjukkan biaya atau jarak antara dua simpul yang terhubung. Bobot bisa berupa angka real atau integer, dan bisa mewakili berbagai jenis informasi seperti jarak fisik, waktu tempuh, atau biaya. UCS memerlukan node awal (*start node*) dan node tujuan (*goal node*) untuk memulai pencarian jalur terpendek. Node awal adalah simpul dari mana pencarian dimulai, sedangkan node tujuan adalah simpul yang ingin dicapai.

UCS menggunakan *explored set* (set yang telah dieksplorasi) untuk melacak simpul-simpul yang akan dieksplorasi dan yang telah dieksplorasi. Dalam prosesnya, UCS juga menghitung biaya akumulatif (*cost-to-come*) dari jalur yang telah dieksplorasi. Biaya akumulatif ini digunakan untuk membandingkan jalur-jalur yang berbeda dan memilih jalur dengan biaya terendah. Biaya ini lalu dimasukkan ke dalam *priority queue* untuk mengurutkan simpul-simpul berdasarkan biaya akumulatif jalur yang telah dieksplorasi. Simpul dengan biaya akumulatif yang lebih rendah diberikan prioritas lebih tinggi untuk dieksplorasi lebih dahulu.

#### 2.2. A Star (A\*)

Algoritma A\* (A-star) adalah algoritma pencarian informasi yang digunakan untuk mencari jalur terpendek dalam graf berbobot. A\* memproses struktur data graf, yang terdiri dari simpul (*node*) yang mewakili objek, dan sisi (*edge*) yang mewakili relasi antara objek tersebut. Setiap sisi pada graf memiliki bobot (*weight*) yang menunjukkan biaya atau jarak antara dua simpul yang terhubung. A\* memerlukan node awal (*start node*) dan node tujuan (*goal node*) untuk memulai pencarian jalur terpendek.

A\* menggunakan fungsi heuristik untuk memperkirakan biaya tersisa atau jarak yang tersisa (*heuristic cost*) dari simpul saat ini ke node tujuan. Fungsi heuristik ini membantu A\* dalam memilih simpul yang paling berpotensi mengarah ke jalur terpendek, sehingga bisa mempercepat pencarian. A\* menghitung biaya akumulatif (*cost-to-come*) dari jalur yang telah dieksplorasi, ditambah dengan nilai heuristik, untuk membandingkan jalur-jalur yang berbeda dan memilih jalur dengan biaya terendah dan heuristik terbaik. Sama seperti UCS, A\* juga menggunakan *explored set* (set yang telah dieksplorasi) dan *priority queue* untuk melacak simpul-simpul yang akan dieksplorasi dan yang telah dieksplorasi, serta untuk mengurutkan simpul-simpul pada graf berdasarkan biaya akumulatif dan nilai heuristik. Simpul dengan nilai heuristik dan biaya akumulatif yang lebih rendah diberikan prioritas lebih tinggi untuk dieksplorasi lebih dahulu.

## BAB 3

### IMPLEMENTASI ALGORITMA UCS DAN A\*

#### 3.1. Alur Program

Alur kerja pada program ini adalah sebagai berikut. Pengguna dapat memilih ingin menjalankan program melalui *Command Line Interface* (CLI) atau *Graphic User Interface* (GUI). Jika pengguna ingin menjalankan melalui CLI, maka pengguna menjalankan file `main.py`, jika GUI maka pengguna menjalankan file `gui.py`.

Cara kerja kedua file serupa. Pengguna akan diminta memasukkan file .txt, kemudian list tempat akan ditampilkan. Pengguna memilih titik awal dan akhir rute. Rute akan ditampilkan melalui plot diagram. Jika ingin ditampilkan melalui OpenStreetMap API, pengguna tinggal memilih pilihan yang disediakan. Jika tidak ada rute yang ditemukan, maka program akan menampilkan pesan error.

#### 3.2. Struktur File

1. Algorithm.py  
File ini berisi algoritma inti dari program yaitu UCS dan A\*
2. Bonus.py  
File ini berisi fungsi untuk menghubungkan kode program dengan OpenStreetMap API
3. Gui.py  
File ini berisi kode utama dari GUI
4. Main.py  
File ini berisi kode utama dari CLI
5. Reader.py  
File ini berisi string parser untuk membaca dari input file .txt
6. utils.py  
File ini berisi fungsi tambahan untuk program seperti visualizer, jarak heuristik, dan lain-lain

#### 3.3. Algoritma UCS dan A\*

```
def visitedNode(node, path):  
    for i in range(len(path)):  
        if (node == path[i]):  
            return False  
    return True  
  
def UCS (matrix, start, goal):
```

```
q= PQ()
path = []
path.append(start)
q.put((0, start, path))
visited = []

while not q.empty():
    cost, currNode, currPath = q.get()
    if (currNode not in visited):
        visited.append(currNode)

        if(currNode==goal):
            return cost, currPath

        # Generate neighbors of the current node from adjacency
matrix
        neighbors=[]
        for i in range(len(matrix[currNode])):
            if(matrix[currNode][i]!=0):
                neighbors.append(i)

        for neighbor in neighbors:
            if((neighbor not in currPath) and (neighbor not in
visited)):
                # If neighbor is not visited before, push it to
the priority queue with its cost and path
                new_cost = cost + matrix[currNode][neighbor]
                new_path = currPath + [neighbor]
                # Calculate cost to reach neighbor from current
node
                q.put((new_cost, neighbor,new_path))

return -999,None

def astar(start_index, goal_index, places, matrix):
    heuristic_distance_list =
make_heuristic_distance_list(goal_index, places)

    # Initialize the first element of the queue
    prediction_distance = heuristic_distance_list[start_index]
    passed_distance = 0
    current_index = start_index
    path_index = [start_index]
    first_element = [prediction_distance, passed_distance,
current_index, path_index]

    # Initialize queue
    queue = PQ()
    queue.put(first_element)

    # Initialize the state of solution
```

```
path_found = False
visited_node_index = set()

while not path_found and not queue.empty():
    current_element = queue.get()
    if (current_element[2] not in visited_node_index):
        visited_node_index.add(current_element[2])
        # Check if current_element's last node is destination
        if (current_element[2] == goal_index):
            solution_element = current_element
            path_found = True
        else:
            for i in range(len(matrix)):
                # if i hasn't been visited and has connection
                to the last node in the current_element
                if ((i not in visited_node_index) and
                    (matrix[current_element[2]][i] != 0)):
                    new_passed_distance = current_element[1] +
matrix[current_element[2]][i]
                    new_prediction_distance =
new_passed_distance + heuristic_distance_list[i]
                    new_curent_index = i
                    new_path_index = current_element[3][:]
                    new_path_index.append(i)

                    new_element = [new_prediction_distance,
new_passed_distance, new_curent_index, new_path_index]
                    queue.put(new_element)

    if path_found:
        return solution_element[1], solution_element[3]
    else:
        return -999, None
```

## BAB 4

### PENGUJIAN

#### 4.1. Peta/Graf Input

Format Input yang kami gunakan berupa file .txt yang diasumsikan selalu valid mengikuti format sebagai berikut:

##### contoh.txt

```
3          // ukuran matrix dan jumlah simpul yang akan diuji
Asia_Afrika // Nama tempat
-6.921141 107.607668 // Latitude dan longitude tempat tersebut pada peta
Otto_Iskandar
-6.920768 107.604056
Pasar_Baru
-6.918263 107.604216
0.4 0 0      // matrix ketetanggaan berbobot dengan jarak asli antar simpul
0 0.4 0      // tersebut dalam satuan kilometer (km)
0 0 0
```

#### 4.2. Hasil Pengujian

##### 4.2.1. Pengujian 1

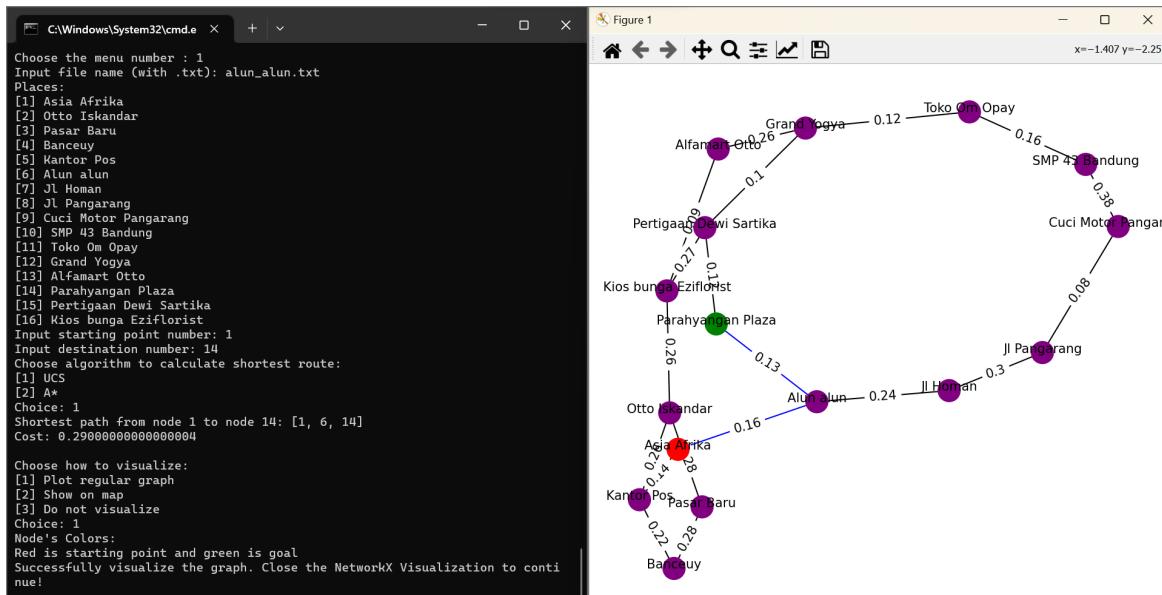
###### Input: alun\_alun.txt

```
16
Asia_Afrika
-6.921141 107.607668
Otto_Iskandar
-6.920768 107.604056
Pasar_Baru
-6.918263 107.604216
Banceuy
-6.919038 107.606670
Kantor_Pos
-6.920995 107.606441
Alun_alun
-6.922551 107.607619
Jl_Homan
-6.922771 107.609810
Jl_Pangarang
-6.925376 107.610599
Cuci_Motor_Pangarang
-6.926075 107.610524
SMP_43_Bandung
-6.925835 107.607071
```

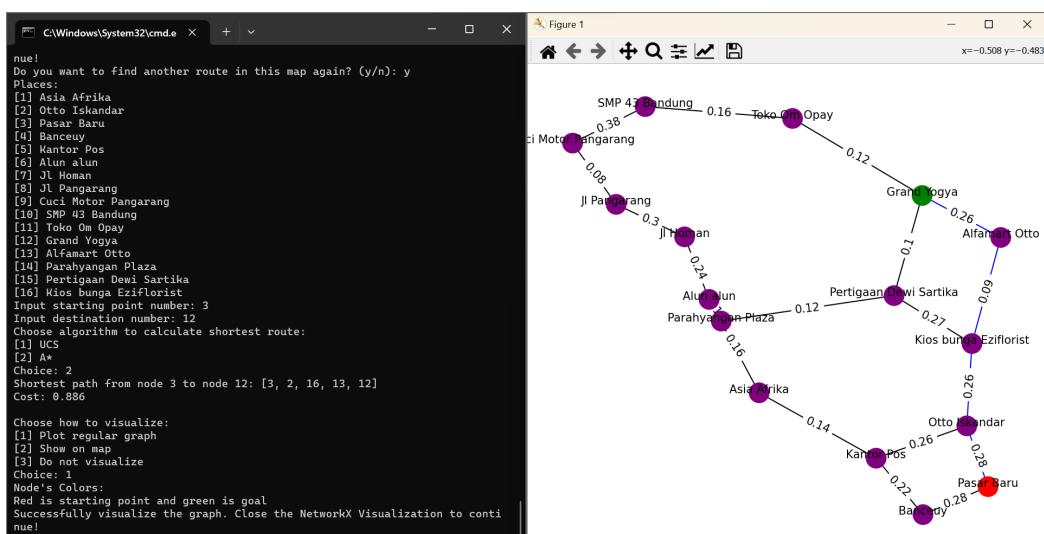
## Hasil Pengujian

UCS:

Tugas Kecil III IF2211 Strategi Algoritma  
Implementasi Algoritma UCS dan Pencarian Rute Terdekat



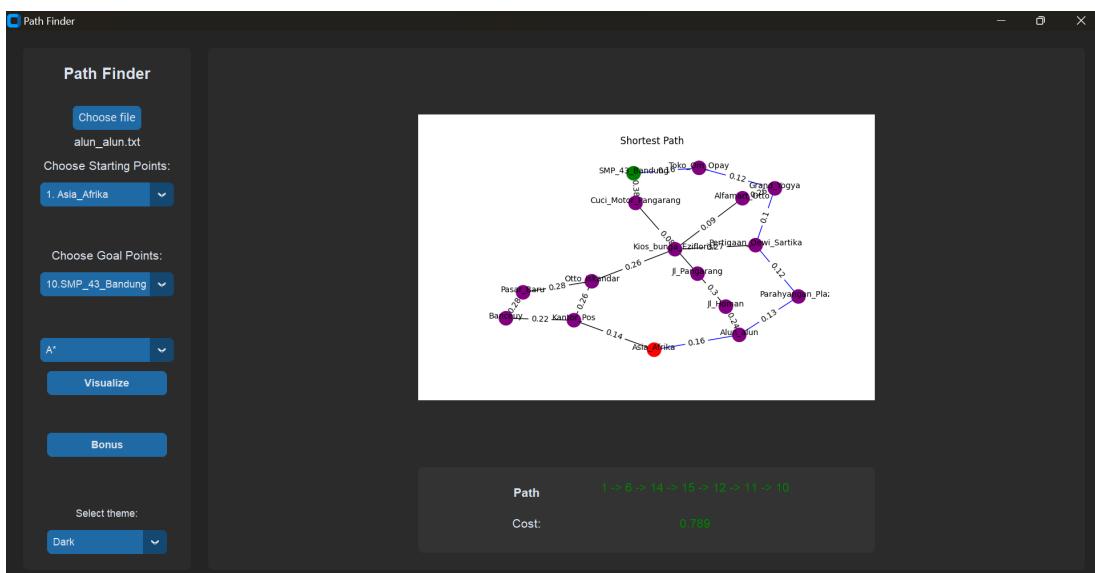
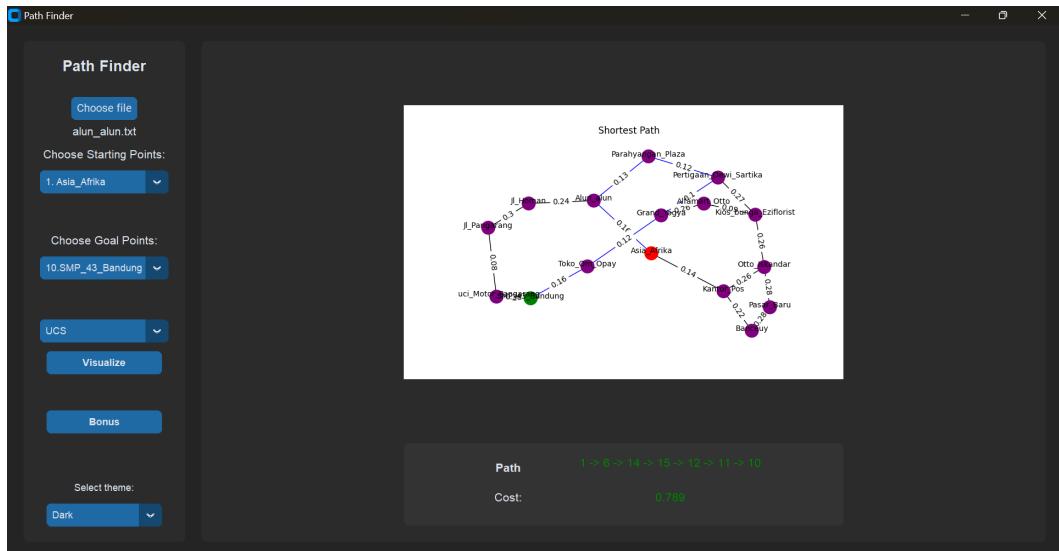
A\*:

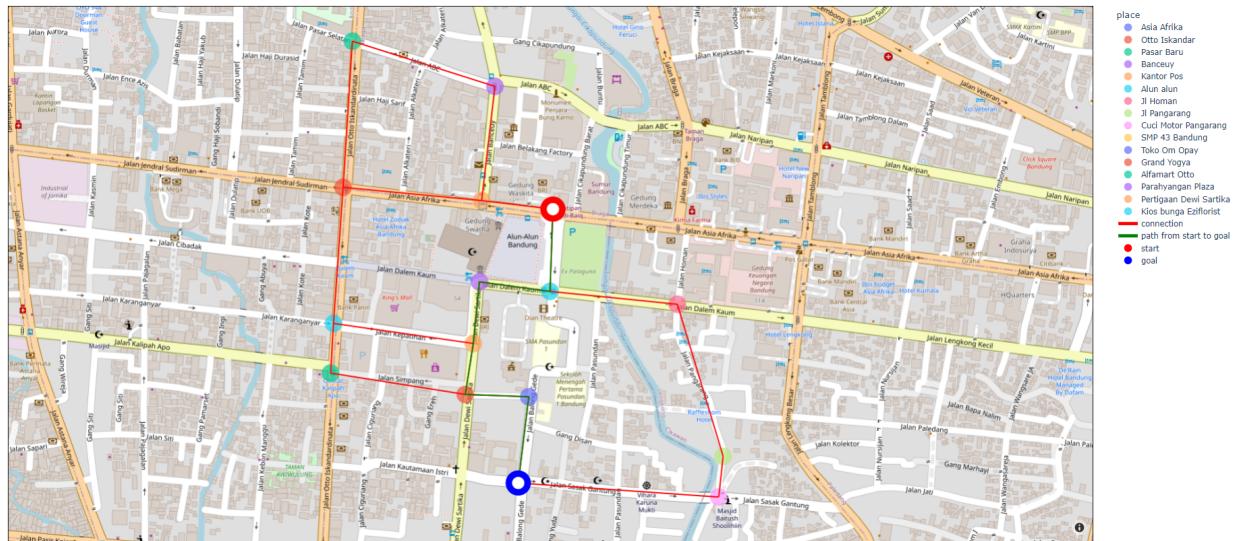


GUI:

# Tugas Kecil III IF2211 Strategi Algoritma

## Implementasi Algoritma UCS dan A\* dalam Pencarian Rute Terdekat





#### 4.2.2. Pengujian 2

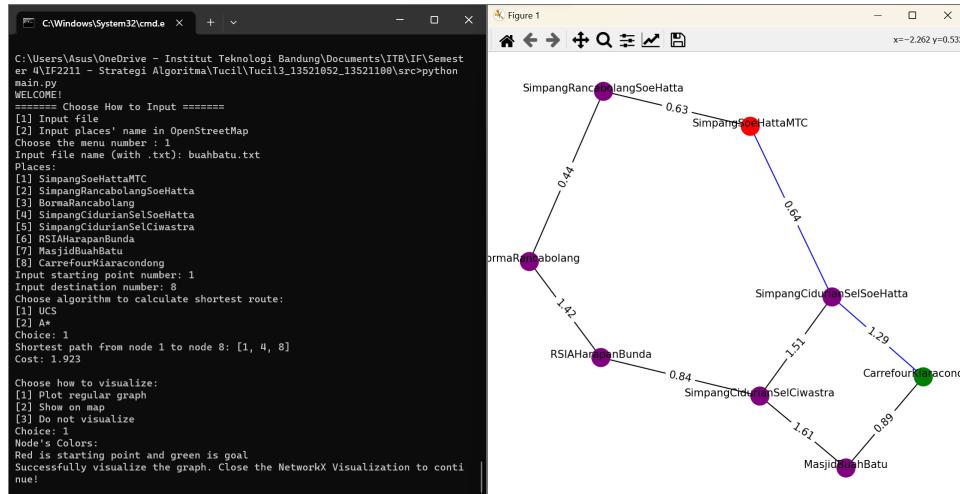
Input: buahbatu.txt

```

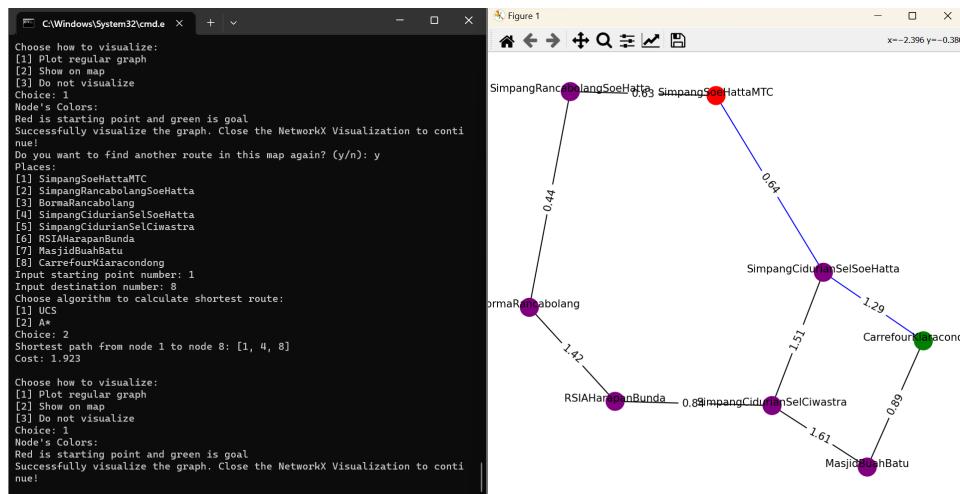
8
SimpangSoeHattaMTC
-6.940351 107.658245
SimpangRancabolangSoeHatta
-6.939252 107.663915
BormaRancabolang
-6.943234 107.663564
SimpangCidurianSelSoeHatta
-6.942138 107.652719
SimpangCidurianSelCiwastra
-6.955690 107.654484
RSIAHarapanBunda
-6.956029 107.662112
MasjidBuahBatu
-6.954222 107.639885
CarrefourKiaracondong
-6.946367 107.641756
0.0 0.632 0.0 0.636 0.0 0.0 0.0 0.0
0.632 0.0 0.441 0.0 0.0 0.0 0.0 0.0
0.0 0.441 0.0 0.0 0.0 1.42 0.0 0.0
0.636 0.0 0.0 0.0 1.507 0.0 0.0 1.287
0.0 0.0 0.0 1.507 0.0 0.836 1.606 0.0
0.0 0.0 1.42 0.0 0.836 0.0 0.0 0.0
0.0 0.0 0.0 0.0 1.606 0.0 0.0 0.89
0.0 0.0 0.0 1.287 0.0 0.0 0.89 0.0

```

## Hasil pengujian UCS:

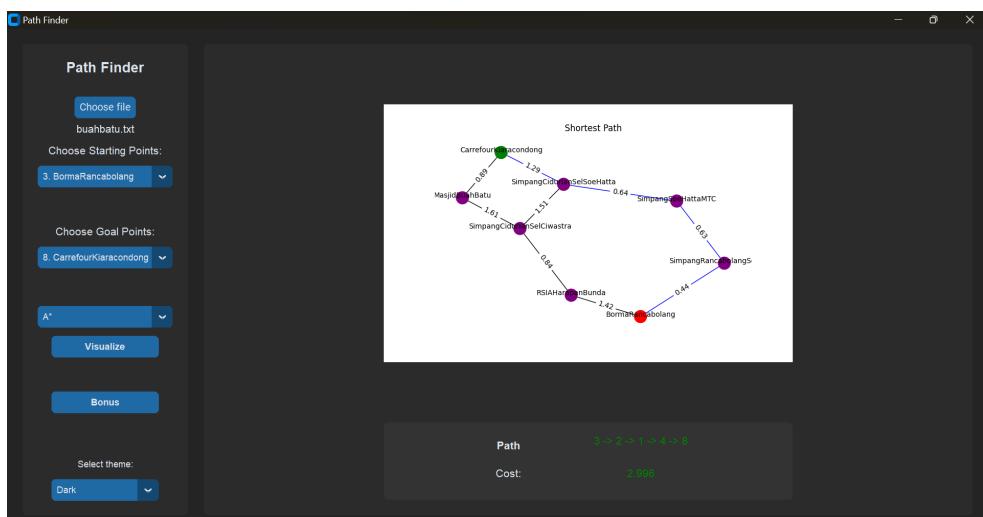
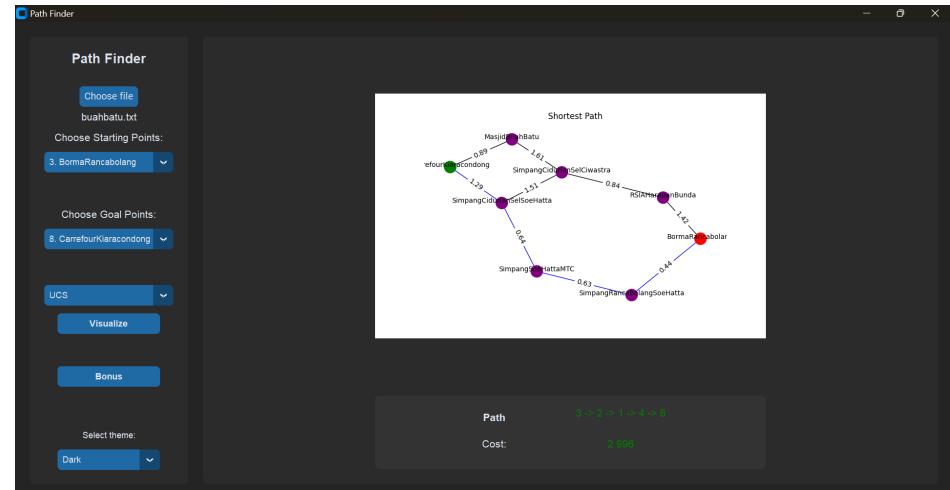


## A\*:



GUI:

Tugas Kecil III IF2211 Strategi Algoritma  
Implementasi Algoritma UCS dan A\* dalam Pencarian Rute Terdekat



#### 4.2.3. Pengujian 3

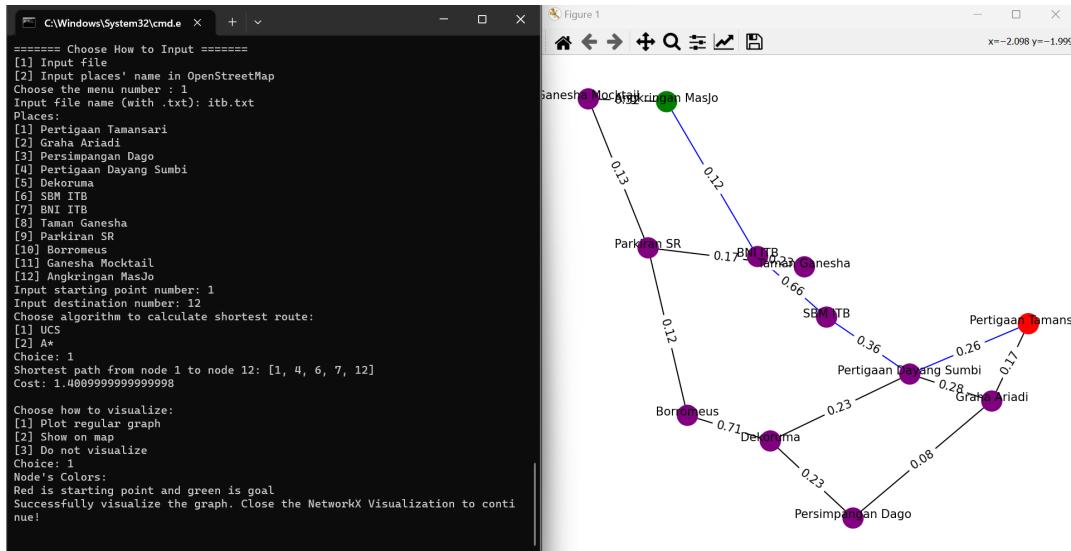
Input: **itb.txt**

```
12
Pertigaan_Tamansari
-6.884893 107.611445
Graha_Ariadi
-6.885191 107.613017
Persimpangan_Dago
-6.885257 107.613733
Pertigaan_Dayang_Sumbi
-6.887256 107.611540
Dekoruma
-6.887386 107.613611
SBM_ITB
-6.887910 107.608289
BNI_ITB
-6.893882 107.608450
Taman_Ganesha
-6.893230 107.610447
Parkiran_SR
-6.893605 107.611944
Borromeus
-6.893780 107.613036
Ganesha_Mocktail
-6.894759 107.611723
Angkringan_MasJo
-6.894883 107.608839
0.0 0.175 0.0 0.261 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.175 0.0 0.079 0.279 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 0.079 0.0 0.0 0.0 0.235 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.261 0.279 0.0 0.0 0.227 0.363 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 0.235 0.227 0.0 0.0 0.0 0.0 0.0 0.0 0.708 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.363 0.0 0.0 0.659 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 0.659 0.0 0.23 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.118
0.0 0.0 0.0 0.0 0.0 0.0 0.23 0.0 0.0 0.169 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 0.169 0.0 0.0 0.121 0.0 0.13 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.708 0.0 0.0 0.0 0.0 0.121 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 0.13 0.0 0.0 0.0 0.316 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.118 0.0 0.0 0.0 0.0 0.0 0.316 0.0
```

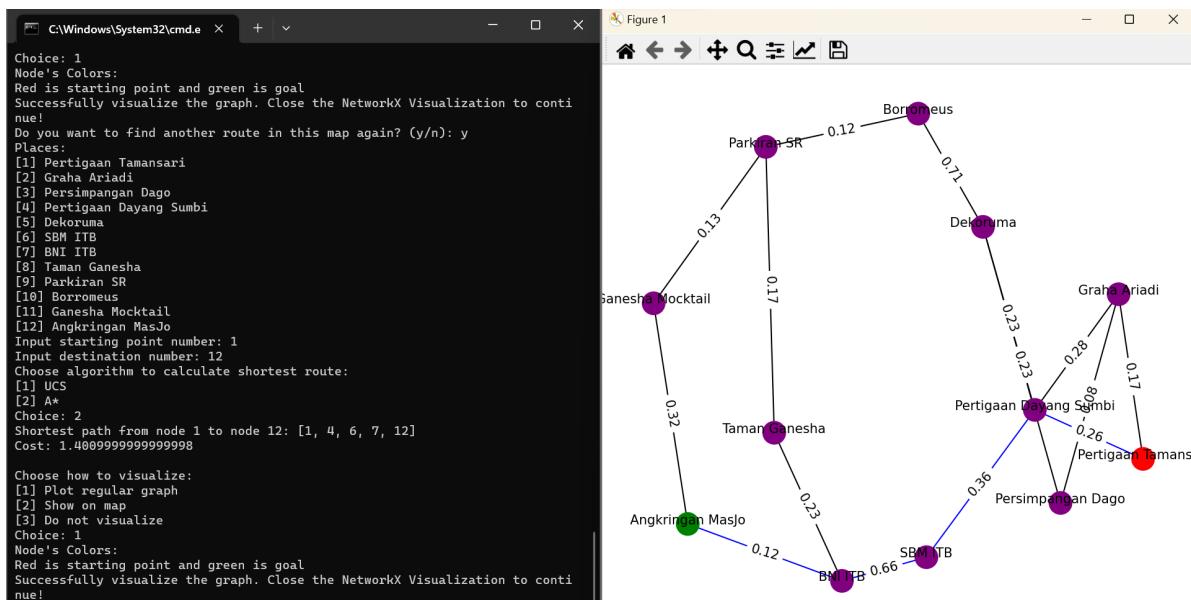
Hasil Pengujian:

UCS:

Tugas Kecil III IF2211 Strategi Algoritma  
Implementasi Algoritma UCS dan A\* dalam Pencarian Rute Terdekat

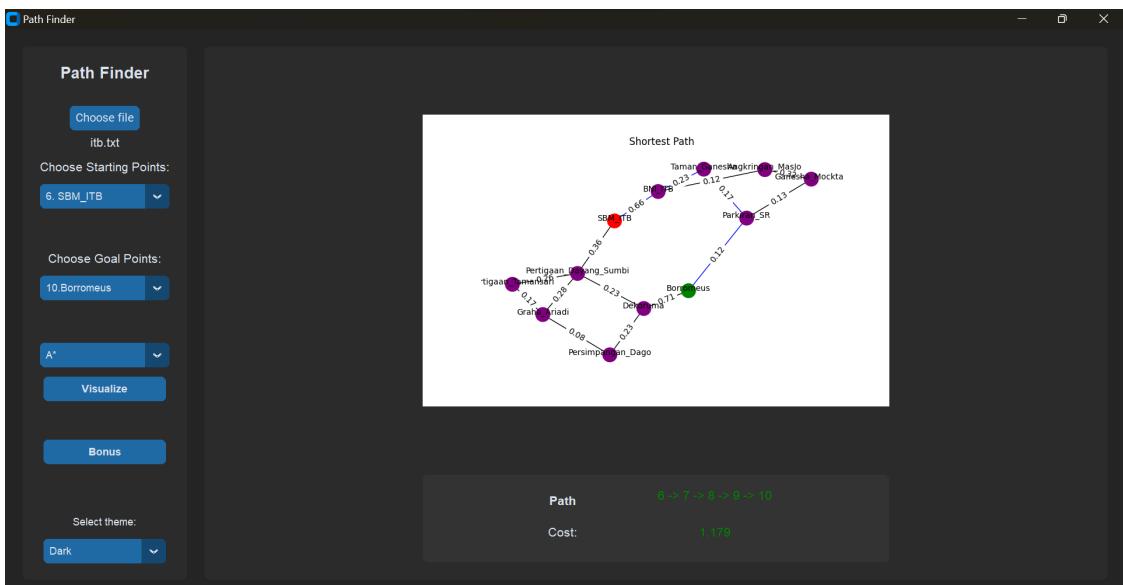
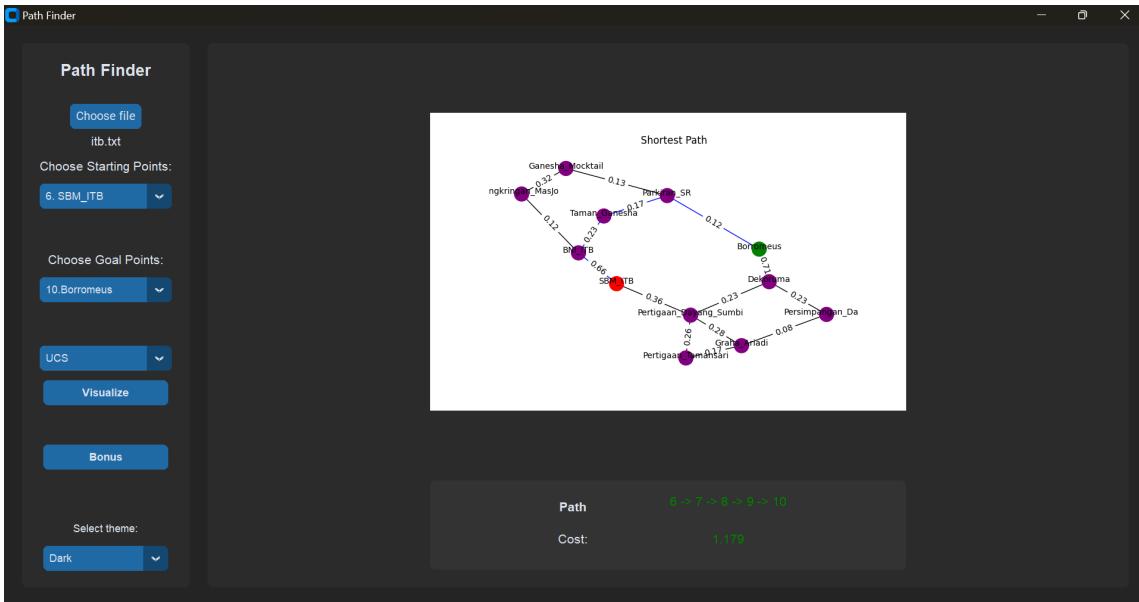


A\*:



GUI:

Tugas Kecil III IF2211 Strategi Algoritma  
Implementasi Algoritma UCS dan A\* dalam Pencarian Rute Terdekat





#### 4.2.4. Pengujian 4

Input: **jakarta.txt**

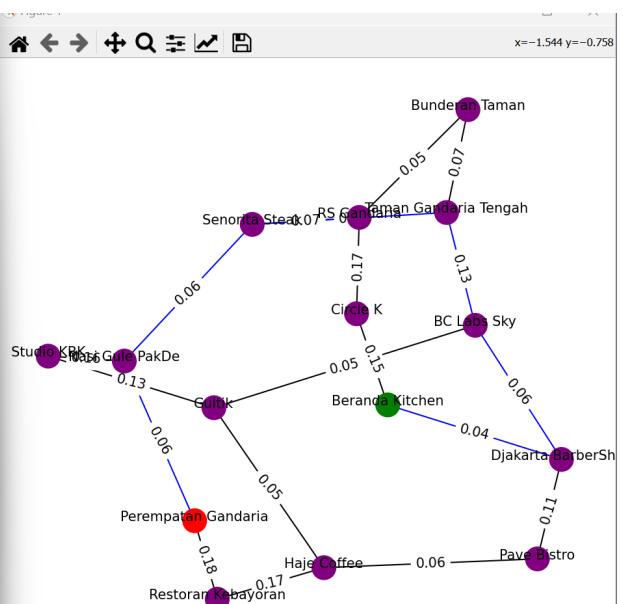
```
15
Perempatan_Gandaria
-6.245131 106.788753
Haje_Coffee
-6.246568 106.791361
Pave_Bistro
-6.246584 106.791921
Circle_K
-6.243914 106.791703
RS_Gandaria
-6.244357 106.790184
Senorita_Steak
-6.244625 106.789645
Nasi_Gule_PakDe
-6.244878 106.789209
Gultik
-6.246105 106.791314
BC_Labs_Sky
-6.245636 106.791288
Djakarta_BarberShop
-6.245589 106.791838
Taman_Gandaria_Tengah
-6.245394 106.790106
Beranda_Kitchen
-6.245254 106.791829
Bunderan_Taman
-6.244802 106.790382
Studio_KBK
```

## Hasil Pengujian

UCS:

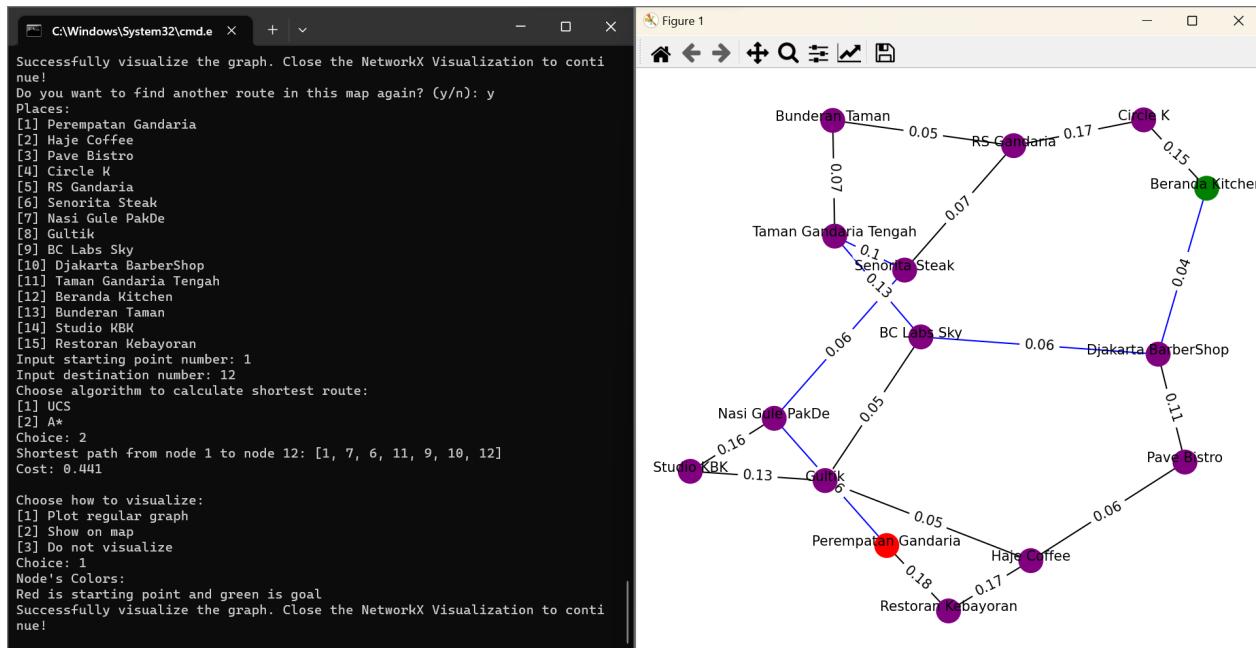
```
[2] Input places' name in OpenStreetMap
Choose the menu number : 1
Input file name (with .txt): jakarta.txt
Places:
[1] Perempatan Gandaria
[2] Haje Coffee
[3] Pave Bistro
[4] Circle K
[5] RS Gandaria
[6] Senorita Steak
[7] Nasi Gule PakDe
[8] Gultik
[9] BC Labs Sky
[10] Djakarta BarberShop
[11] Taman Gandaria Tengah
[12] Beranda Kitchen
[13] Bunderan Taman
[14] Studio KBK
[15] Restoran Kebayoran
Input starting point number: 1
Input destination number: 12
Choose algorithm to calculate shortest route:
[1] UCS
[2] A*
Choice: 1
Shortest path from node 1 to node 12: [1, 7, 6, 11, 9, 10, 12]
Cost: 0.441

Choose how to visualize:
[1] Plot regular graph
[2] Show on map
[3] Do not visualize
Choice: 1
Node's Colors:
Red is starting point and green is goal
Successfully visualize the graph. Close the NetworkX Visualization to continue!
```

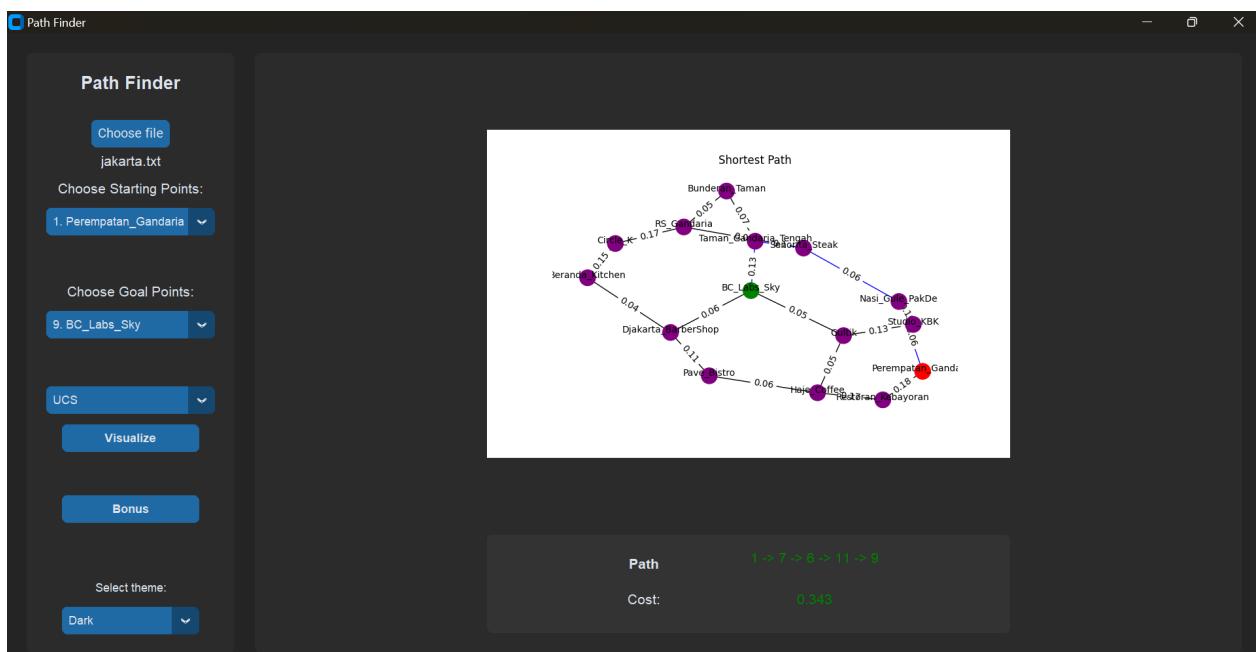


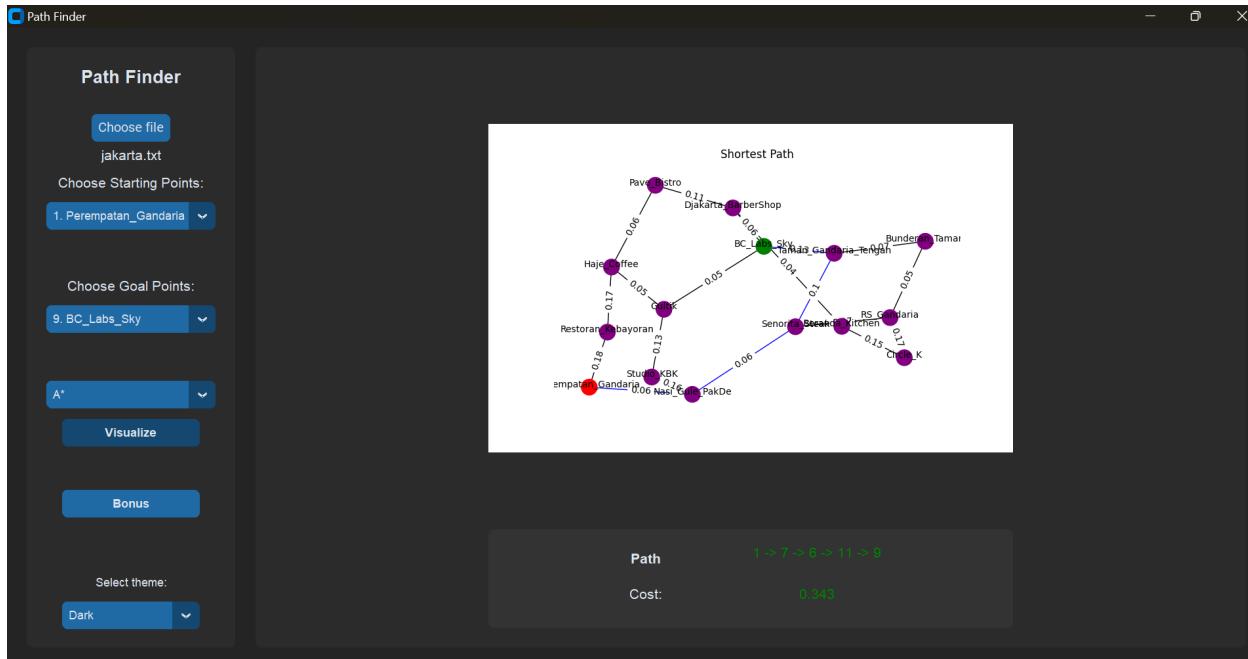
A\*:

Tugas Kecil III IF2211 Strategi Algoritma  
Implementasi Algoritma UCS dan Pencarian Rute Terdekat



GUI:





#### 4.2.5. Pengujian 5 (Bonus)

Penjelasan:

Pengguna menjalankan main CLI, lalu memasukkan nama tempat yang ingin dijadikan node. Nama tempat tersebut akan di-retrieve dari OpenStreetMap. Apabila tidak terdapat nama tempat yang **menyerupai** input dari user pada OpenStreetMap, maka program akan mencetak pesan validasi.

input:
===== Choose How to Input =====

[1] Input file

[2] Input places' name in OpenStreetMap

Choose the menu number : **2**

===== ADD LOCATION =====

Enter a location name: **itb**

===== LOCATIONS =====

1. Institut Teknologi Bandung, 10-12, Ganesha, Lebak Siliwangi, Coblong, Bandung, Jawa Barat, 40132, Indonesia

Type "done" to stop adding locations

Enter a Location name: **ithb**

===== LOCATIONS =====

{ – *Tidak dimasukkan dalam laporan untuk mempersingkat – }*

Type "done" to stop adding locations

Enter a Location name: **dago asri**

===== LOCATIONS =====

{ – *Tidak dimasukkan dalam laporan untuk mempersingkat – }*

Type "done" to stop adding locations

Enter a Location name: **unpad**

===== LOCATIONS =====

{ – *Tidak dimasukkan dalam laporan untuk mempersingkat – }*

Type "done" to stop adding locations

Enter a Location name: **tubagus ismail**

===== LOCATIONS =====

{ – *Tidak dimasukkan dalam laporan untuk mempersingkat – }*

Type "done" to stop adding locations

Enter a Location name: **galeri ciumbuleuit**

===== LOCATIONS =====

{ – *Tidak dimasukkan dalam laporan untuk mempersingkat – }*

Type "done" to stop adding locations

Enter a Location name: **unpar**

===== LOCATIONS =====

{ – Tidak dimasukkan dalam laporan untuk mempersingkat – }

Type "done" to stop adding locations

Enter a Location name: **punclut**

===== LOCATIONS =====

1. Institut Teknologi Bandung, 10-12, Ganesha, Lebak Siliwangi, Coblong, Bandung, Jawa Barat, 40132, Indonesia
2. Pos Satpam ITHB, Jalan Dipatiukur, Lebak Gede, Coblong, Bandung, Jawa Barat, 40132, Indonesia
3. Jalan Dago Vila Asri, Cisitu Lama, Dago, Coblong, Bandung, Jawa Barat, 40135, Indonesia
4. UNPAD, Jalan Dago Pojok, Dago, Coblong, Bandung, Jawa Barat, 41035, Indonesia
5. Jalan Tubagus Ismail, Sekeloa, Coblong, Bandung, Jawa Barat, 40134, Indonesia
6. Galeri Ciembuleuit, 42A, Jalan Ciumbuleuit, Hegarmanah, Cidadap, Bandung, Jawa Barat, Jawa, 40132, Indonesia
7. UNPAR, Jalan Ciumbuleuit, Hegarmanah, Cidadap, Bandung, Jawa Barat, Jawa, 40141, Indonesia
8. Kebun Hidroponik Punclut, Ciumbuleuit, Cidadap, Bandung, Jawa Barat, Indonesia

Type "done" to stop adding locations

Enter a Location name: **done**

===== ADD CONNECTION =====

- Do you want to add connection? (y/n): **y**  
Input the number of the first place: **1**  
Input the number of the second place: **2**  
Do you want to add connection? (y/n): **y**  
Input the number of the first place: **2**  
Input the number of the second place: **3**  
Do you want to add connection? (y/n): **y**  
Input the number of the first place: **3**  
Input the number of the second place: **4**  
Do you want to add connection? (y/n): **y**  
Input the number of the first place: **4**  
Input the number of the second place: **5**  
Do you want to add connection? (y/n): **y**  
Input the number of the first place: **5**  
Input the number of the second place: **6**  
Do you want to add connection? (y/n): **y**  
Input the number of the first place: **6**  
Input the number of the second place: **7**  
Do you want to add connection? (y/n): **y**  
Input the number of the first place: **7**  
Input the number of the second place: **8**  
Do you want to add connection? (y/n): **y**  
Input the number of the first place: **6**  
Input the number of the second place: **8**  
Do you want to add connection? (y/n): **y**  
Input the number of the first place: **5**

Input the number of the second place: **7**

Do you want to add connection? (y/n): **n**

Places:

[1] Institut Teknologi Bandung, 10-12, Ganesha, Lebak Siliwangi, Coblong, Bandung, Jawa Barat, 40132, Indonesia

[2] Pos Satpam ITHB, Jalan Dipatiukur, Lebak Gede, Coblong, Bandung, Jawa Barat, 40132, Indonesia

[3] Jalan Dago Vila Asri, Cisitu Lama, Dago, Coblong, Bandung, Jawa Barat, 40135, Indonesia

[4] UNPAD, Jalan Dago Pojok, Dago, Coblong, Bandung, Jawa Barat, 41035, Indonesia

[5] Jalan Tubagus Ismail, Sekeloa, Coblong, Bandung, Jawa Barat, 40134, Indonesia

[6] Galeri Ciembuleuit, 42A, Jalan Ciumbuleuit, Hegarmanah, Cidadap, Bandung, Jawa Barat, 40132, Indonesia

[7] UNPAR, Jalan Ciumbuleuit, Hegarmanah, Cidadap, Bandung, Jawa Barat, 40141, Indonesia

[8] Kebun Hidroponik Punclut, Ciumbuleuit, Cidadap, Bandung, Jawa Barat, Indonesia

Input starting point number: **1**

Input destination number: **8**

Choose algorithm to calculate shortest route:

[1] UCS

[2] A\*

Choice: **2**

Shortest path from node 1 to node 8: [1, 2, 3, 4, 5, 7, 8]

Cost: 8.050028218077106

Choose how to visualize:

[1] Plot regular graph

[2] Show on map

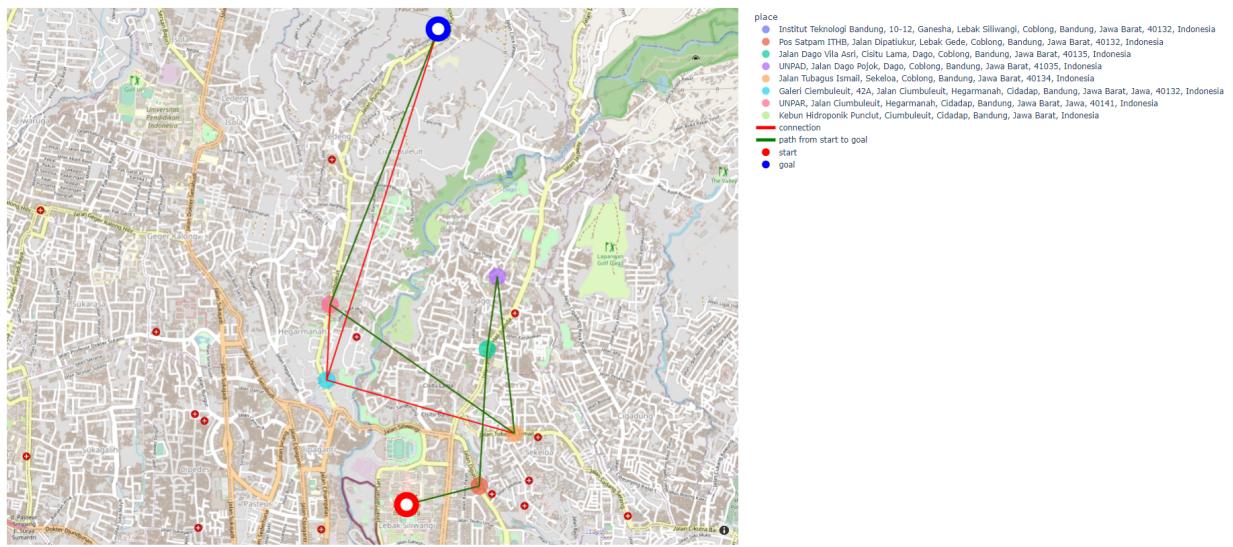
[3] Do not visualize

Choice: **2**

Result:

Shortest path from node 1 to node 8: [1, 2, 3, 4, 5, 7, 8]

Cost: 8.050028218077106



#### 4.2.6. Pengujian 6 (Bonus)

input:

Do you want to find another route in this map again? (y/n): **y**

Places:

- [1] Institut Teknologi Bandung, 10-12, Ganesha, Lebak Siliwangi, Coblong, Bandung, Jawa Barat, 40132, Indonesia
- [2] Pos Satpam ITHB, Jalan Dipatiukur, Lebak Gede, Coblong, Bandung, Jawa Barat, 40132, Indonesia
- [3] Jalan Dago Vila Asri, Cisitu Lama, Dago, Coblong, Bandung, Jawa Barat, 40135, Indonesia
- [4] UNPAD, Jalan Dago Pojok, Dago, Coblong, Bandung, Jawa Barat, 41035, Indonesia
- [5] Jalan Tubagus Ismail, Sekelo, Coblong, Bandung, Jawa Barat, 40134, Indonesia
- [6] Galeri Ciembuleuit, 42A, Jalan Ciumbuleuit, Hegarmanah, Cidadap, Bandung, Jawa Barat, 40132, Indonesia
- [7] UNPAR, Jalan Ciumbuleuit, Hegarmanah, Cidadap, Bandung, Jawa Barat, Jawa, 40141, Indonesia
- [8] Kebun Hidroponik Punclut, Ciumbuleuit, Cidadap, Bandung, Jawa Barat, Ip, Bandung, Jawa Barat, Indonesia

Input starting point number: **7**

Input destination number: **3**

Choose algorithm to calculate shortest route:

- [1] UCS

- [2] A\*

Choice: **1**

Shortest path from node 7 to node 3: [7, 5, 4, 3]

Cost: 3.8083583313577885

Choose how to visualize:

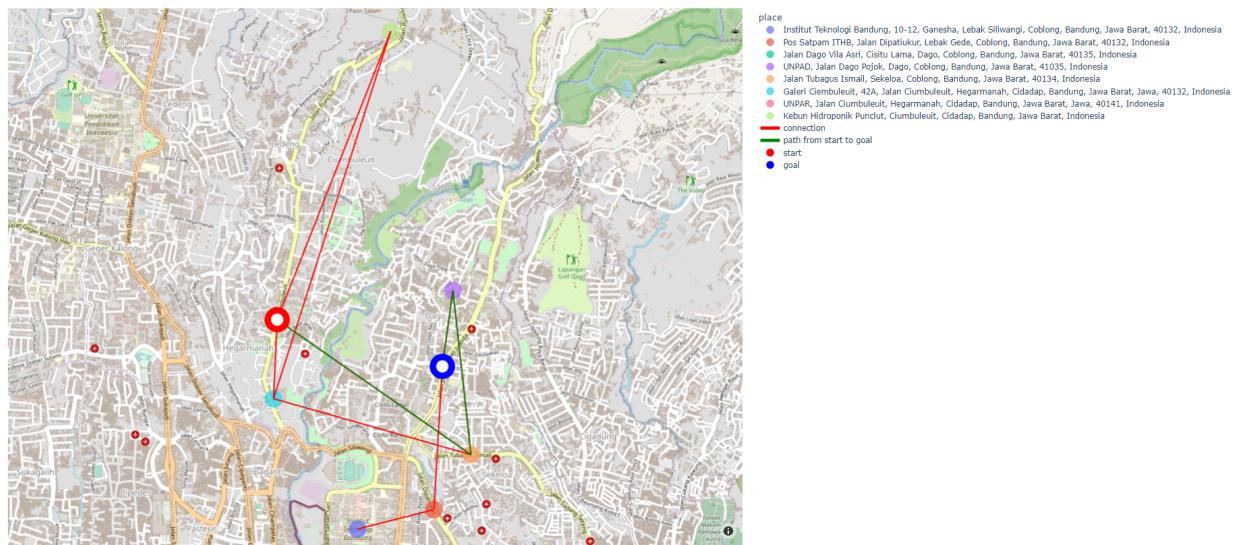
- [1] Plot regular graph
- [2] Show on map
- [3] Do not visualize

Choice: 2

Result:

Shortest path from node 7 to node 3: [7, 5, 4, 3]

Cost: 3.8083583313577885



#### 4.2.7. Pengujian 7 (Bonus)

input:

WELCOME!

===== Choose How to Input =====

- [1] Input file
- [2] Input places' name in OpenStreetMap

Choose the menu number : 2

===== ADD LOCATION =====

Enter a location name: **mall of indonesia**

===== LOCATIONS =====

1. Mall of Indonesia, Jalan Tol Sunter–Pulo Gebang, RW 19, Kelapa Gading Barat, Kelapa Gading, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14240, Indonesia

Type "done" to stop adding locations

Enter a Location name: **paladian park**

===== LOCATIONS =====

{ – Tidak dimasukkan dalam laporan untuk mempersingkat – }

Type "done" to stop adding locations

Enter a Location name: **mall artha gading**

===== LOCATIONS =====

{ – Tidak dimasukkan dalam laporan untuk mempersingkat – }

Type "done" to stop adding locations

Enter a Location name: **janur asri, gading**

===== LOCATIONS =====

{ – Tidak dimasukkan dalam laporan untuk mempersingkat – }

Type "done" to stop adding locations

Enter a Location name: **kelapa hibrida, gading**

===== LOCATIONS =====

{ – Tidak dimasukkan dalam laporan untuk mempersingkat – }

Type "done" to stop adding locations

Enter a Location name: **mall kelapa gading**

===== LOCATIONS =====

{ – Tidak dimasukkan dalam laporan untuk mempersingkat – }

Type "done" to stop adding locations

Enter a Location name: **wisma gading permai, gading**

===== LOCATIONS =====

{ – Tidak dimasukkan dalam laporan untuk mempersingkat – }

Type "done" to stop adding locations

Enter a Location name: **pulo mas**

===== LOCATIONS =====

1. Mall of Indonesia, Jalan Tol Sunter–Pulo Gebang, RW 19, Kelapa Gading Barat, Kelapa Gading, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14240, Indonesia

2. Paladian Park, Kelapa Gading, Jakarta Utara, Daerah Khusus Ibukota Jakarta, Jawa, Indonesia

3. Mall Artha Gading, No.1, Jalan Artha Gading Selatan, RW 21, Kelapa Gading Barat, Kelapa Gading, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14240, Indonesia

4. Jalan Janur Asri V, RW 12, Kelapa Gading Barat, Kelapa Gading, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14240, Indonesia
5. Kelapa Hibrida Raya, RW 18, Pegangsaan Dua, Kelapa Gading, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14250, Indonesia
6. Mall Kelapa Gading, Kelapa Gading, Jakarta Utara, Daerah Khusus Ibukota Jakarta, Indonesia
7. Apartemen Wisma Gading Permai, Jalan Bulevar Utara Kelapa Gading, RW 20, Kelapa Gading Timur, Kelapa Gading, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14250, Indonesia
8. Pulo Mas, Trans Jakarta Busway Koridor 2, Kawasan Pergudangan Bulog, Kelapa Gading Barat, Kelapa Gading, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14240, Indonesia

Type "done" to stop adding locations

Enter a Location name: done

===== ADD CONNECTION =====

Do you want to add connection? (y/n): **y**

Input the number of the first place: **1**

Input the number of the second place: **2**

Do you want to add connection? (y/n): **2**

Invalid input. Please enter a valid input!

Do you want to add another connection? (y/n): **y**

Input the number of the first place: **2**

Input the number of the second place: **3**

Do you want to add connection? (y/n): **y**

Input the number of the first place: **3**

Input the number of the second place: **4**

Do you want to add connection? (y/n): **y**

Input the number of the first place: **4**

Input the number of the second place: **5**

Do you want to add connection? (y/n): **y**

Input the number of the first place: **5**

Input the number of the second place: **6**

Do you want to add connection? (y/n): **y**

Input the number of the first place: **6**

Input the number of the second place: **7**

Do you want to add connection? (y/n): **y**

Input the number of the first place: **7**

Input the number of the second place: **8**

Do you want to add connection? (y/n): **y**

Input the number of the first place: **8**

Input the number of the second place: **9**

Invalid input. Please enter a valid input!

Input the number of the second place: **3**

Do you want to add connection? (y/n): **y**

Input the number of the first place: **4**

Input the number of the second place: **6**

Do you want to add connection? (y/n): **y**

Input the number of the first place: **1**

Input the number of the second place: **3**

Do you want to add connection? (y/n): **n**

Places:

- [1] Mall of Indonesia, Jalan Tol Sunter–Pulo Gebang, RW 19, Kelapa Gading Barat, Kelapa Gading, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14240, Indonesia
- [2] Paladian Park, Kelapa Gading, Jakarta Utara, Daerah Khusus Ibukota Jakarta, Jawa, Indonesia
- [3] Mall Artha Gading, No.1, Jalan Artha Gading Selatan, RW 21, Kelapa Gading Barat, Kelapa Gading, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14240, Indonesia
- [4] Jalan Janur Asri V, RW 12, Kelapa Gading Barat, Kelapa Gading, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14240, Indonesia
- [5] Kelapa Hibrida Raya, RW 18, Pegangsaan Dua, Kelapa Gading, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14250, Indonesia
- [6] Mall Kelapa Gading, Kelapa Gading, Jakarta Utara, Daerah Khusus Ibukota Jakarta, Indonesia
- [7] Apartemen Wisma Gading Permai, Jalan Bulevar Utara Kelapa Gading, RW 20, Kelapa Gading Timur, Kelapa Gading, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14250, Indonesia
- [8] Pulo Mas, Trans Jakarta Busway Koridor 2, Kawasan Pergudangan Buleleng, Kelapa Gading Barat, Kelapa Gading, Jakarta Utara, Daerah Khusus Ibukota Jakarta, 14240, Indonesia

Input starting point number: **1**

Input destination number: **6**

Choose algorithm to calculate shortest route:

[1] UCS

[2] A\*

Choice: **2**

Shortest path from node 1 to node 6: [1, 3, 4, 6]

Cost: 3.9702719846178063

Choose how to visualize:

[1] Plot regular graph

[2] Show on map

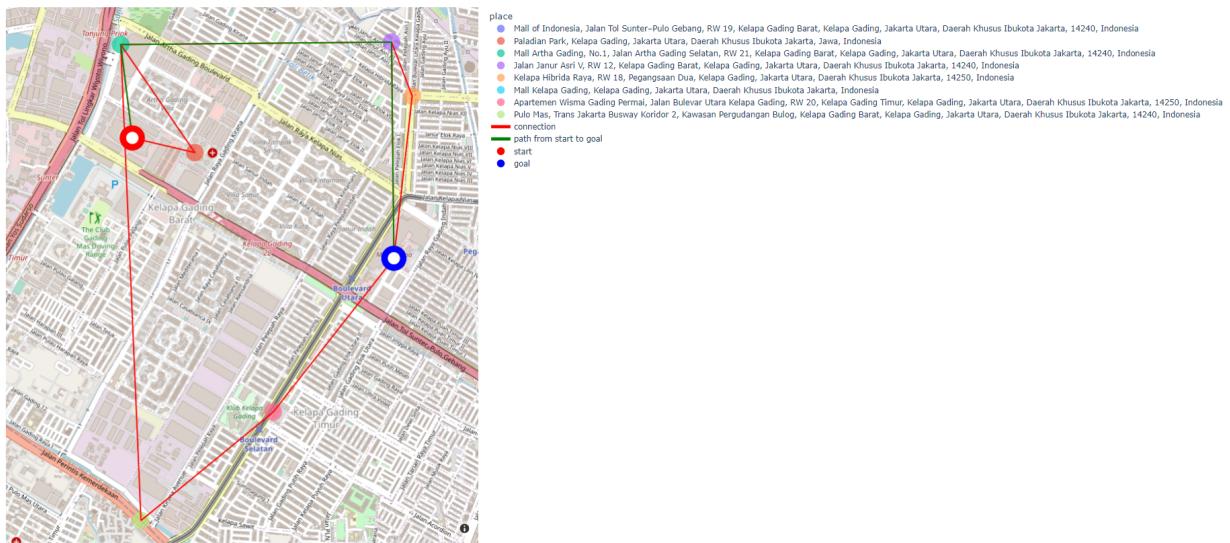
[3] Do not visualize

Choice: **2**

Result:

Shortest path from node 1 to node 6: [1, 3, 4, 6]

Cost: 3.9702719846178063



#### 4.2.8. Pengujian 8 (Bonus)

input:

WELCOME!

===== Choose How to Input =====

[1] Input file

[2] Input places' name in OpenStreetMap

Choose the menu number : 2

===== ADD LOCATION =====

Enter a location name: kuta

===== LOCATIONS =====

1. Kuta, Badung, Bali, 80631, Indonesia

Type "done" to stop adding locations

Enter a Location name: surabaya

===== LOCATIONS =====

{ – Tidak dimasukkan dalam laporan untuk mempersingkat – }

Type "done" to stop adding locations

Enter a Location name: jogja

===== LOCATIONS =====

{ – Tidak dimasukkan dalam laporan untuk mempersingkat – }

Type "done" to stop adding locations

Enter a Location name: **semarang**

===== LOCATIONS =====

{ – *Tidak dimasukkan dalam laporan untuk mempersingkat – }* }

Type "done" to stop adding locations

Enter a Location name: **solo**

===== LOCATIONS =====

{ – *Tidak dimasukkan dalam laporan untuk mempersingkat – }* }

Type "done" to stop adding locations

Enter a Location name: **bandung**

===== LOCATIONS =====

{ – *Tidak dimasukkan dalam laporan untuk mempersingkat – }* }

Type "done" to stop adding locations

Enter a Location name: **jakarta**

===== LOCATIONS =====

{ – *Tidak dimasukkan dalam laporan untuk mempersingkat – }* }

Type "done" to stop adding locations

Enter a Location name: **palembang**

===== LOCATIONS =====

{ – *Tidak dimasukkan dalam laporan untuk mempersingkat – }* }

Type "done" to stop adding locations

Enter a Location name: **riau**

===== LOCATIONS =====

{ – *Tidak dimasukkan dalam laporan untuk mempersingkat – }* }

Type "done" to stop adding locations

Enter a Location name: **bangka belitung**

===== LOCATIONS =====

{ – *Tidak dimasukkan dalam laporan untuk mempersingkat – }* }

Type "done" to stop adding locations

Enter a Location name: **medan**

===== LOCATIONS =====

1. Kuta, Badung, Bali, 80631, Indonesia
2. Surabaya, Jawa Timur, Indonesia
3. Kota Yogyakarta, Daerah Istimewa Yogyakarta, Jawa, Indonesia
4. Semarang, Jawa Tengah, Jawa, 50241, Indonesia
5. Surakarta, Jawa Tengah, Indonesia
6. Bandung, Jawa Barat, Jawa, Indonesia
7. Daerah Khusus Ibukota Jakarta, Jawa, Indonesia
8. Palembang, Sumatera Selatan, Indonesia
9. Riau, Indonesia
10. Kepulauan Bangka Belitung, Indonesia
11. Kota Medan, Sumatera Utara, Indonesia

Type "done" to stop adding locations

Enter a Location name: done

===== ADD CONNECTION =====

Do you want to add connection? (y/n): **y**

Input the number of the first place: **2**

Input the number of the second place: **3**

Do you want to add connection? (y/n): **y**

Input the number of the first place: **3**

Input the number of the second place: **4**

Do you want to add connection? (y/n): **y**

Input the number of the first place: **4**

Input the number of the second place: **5**

Do you want to add connection? (y/n): **y**

Input the number of the first place: **5**

Input the number of the second place: **6**

Do you want to add connection? (y/n): **y**

Input the number of the first place: **6**

Input the number of the second place: **7**

Do you want to add connection? (y/n): **y**

Input the number of the first place: **8**

Input the number of the second place: **9**

Do you want to add connection? (y/n): **y**

Input the number of the first place: **9**

Input the number of the second place: **11**

Do you want to add connection? (y/n): **n**

Places:

- [1] Kuta, Badung, Bali, 80631, Indonesia
- [2] Surabaya, Jawa Timur, Indonesia
- [3] Kota Yogyakarta, Daerah Istimewa Yogyakarta, Jawa, Indonesia
- [4] Semarang, Jawa Tengah, Jawa, 50241, Indonesia
- [5] Surakarta, Jawa Tengah, Indonesia
- [6] Bandung, Jawa Barat, Jawa, Indonesia
- [7] Daerah Khusus Ibukota Jakarta, Jawa, Indonesia
- [8] Palembang, Sumatera Selatan, Indonesia
- [9] Riau, Indonesia

[10] Kepulauan Bangka Belitung, Indonesia  
[11] Kota Medan, Sumatera Utara, Indonesia

Input starting point number: 2

Input destination number: 6

Choose algorithm to calculate shortest route:

[1] UCS

[2] A\*

Choice: 1

Shortest path from node 2 to node 6: [2, 3, 4, 5, 6]

Cost: 793.2573405454141

Choose how to visualize:

[1] Plot regular graph

[2] Show on map

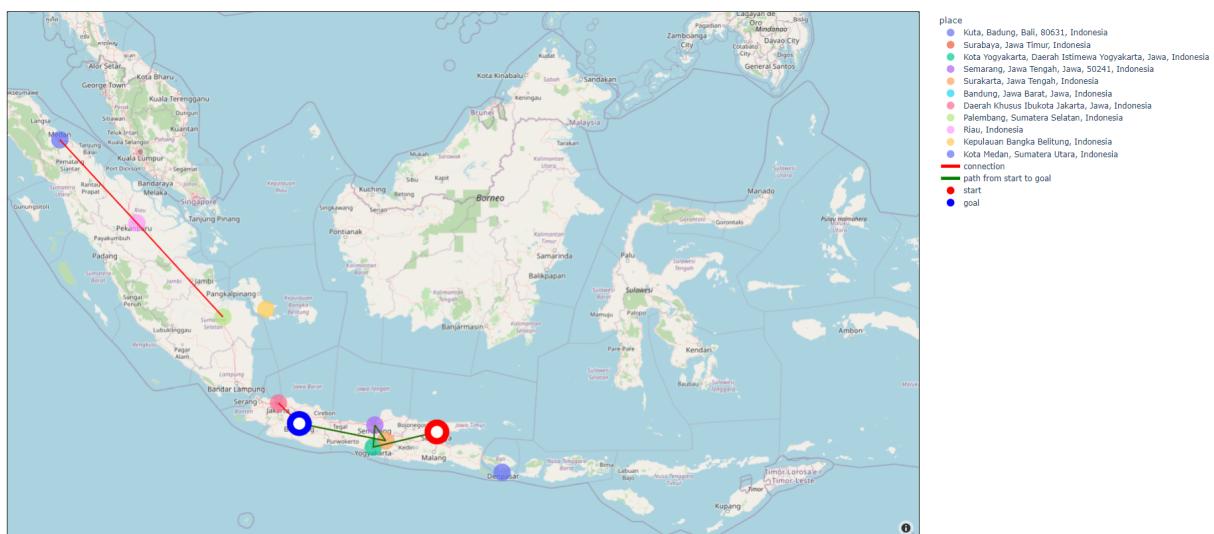
[3] Do not visualize

Choice: 2

Result:

Shortest path from node 2 to node 6: [2, 3, 4, 5, 6]

Cost: 793.2573405454141



## BAB 5

### PENUTUP

#### 5.1. Kesimpulan

Dalam tugas besar IF2211 Strategi Algoritma ini, kami berhasil membuat aplikasi desktop yang mengaplikasikan algoritma UCS dan A\* dalam menyelesaikan pencarian rute terpendek dari Program desktop yang kami buat memiliki tampilan yang interaktif dan mudah digunakan, serta berhasil menampilkan rute yang dihasilkan, serta langkah-langkah yang dilakukan untuk menemukan rute tersebut.

Dari penggerjaan tugas ini, kami mendapatkan beberapa kesimpulan, yaitu

- Algoritma UCS dapat menemukan rute optimal/terpendek dari suatu persoalan yang dapat direpresentasikan sebagai persoalan graph berbobot. UCS memprioritaskan rute dengan total bobot terendah.
- Algoritma A\* dapat menemukan rute optimal/terpendek dari suatu persoalan yang dapat direpresentasikan sebagai persoalan graph berbobot. A\* memprioritaskan rute dengan prediksi cost terendah.

#### 5.2. Saran

Beberapa saran yang kami dapatkan dari penggerjaan pengembangan aplikasi ini adalah sebagai berikut,

1. Perlu dilakukan pendalaman dan pemahaman terlebih dahulu terhadap penggunaan google map API atau sejenisnya, serta tools-tools lain yang digunakan dalam pengembangan program ini.
2. Perlunya perencanaan terlebih dahulu mengenai struktur dari program dan fungsi apa saja yang akan digunakan untuk menyelesaikan persoalan.
3. Perlu perencanaan terlebih dahulu mengenai algoritma UCS dan A\* yang ingin dikembangkan dapat menyelesaikan persoalan dengan optimal.

#### 5.3. *Program Checkpoint*

Poin	Ya	Tidak
1. Program dapat menerima input graf	✓	
2. Program dapat menghitung lintasan terpendek UCS	✓	
3. Program dapat menghitung lintasan terpendek A*	✓	
4. Program dapat menampilkan lintasan terpendek serta	✓	

jaraknya		
5. Bonus: Program dapat menerima input peta dengan OpenStreetMap API dan menampilkan peta serta lintasan terpendek pada peta	✓	

## LAMPIRAN

### LINK REPOSITORY

Link repository GitHub : [https://github.com/melvinkj/Tucil3\\_13521052\\_13521100](https://github.com/melvinkj/Tucil3_13521052_13521100)