


**Tugas Kecil 3 IF 2211 Strategi Algoritma
Implementasi Algoritma A* untuk Menentukan Lintasan
Terpendek
Semester II Tahun 2020/2021**



**Zaidan Naufal Sudrajat / 13518021
Stefanus / 13519101**

Sekolah Teknik Elektro dan Informatika - Institut Teknologi
Bandung

Jl. Ganesha 10, Bandung 40132

	Sekolah Teknik Elektro dan Informatika ITB	Nomor Dokumen		Halaman
		<i>IF2211-TK-63-1</i>		<i>21</i>
		<i>Revisi</i>	<i>1</i>	<i>7/4/2021</i>

BAB I

Deskripsi Masalah

Algoritma 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 atau ujung jalan. Asumsikan jalan dapat dilalui dari dua arah. Bobot graf menyatakan jarak (m atau km) antar simpul. Jarak antara 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.



Gambar 1.1. Ilustrasi Peta Google Map

(Sumber:

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Tugas-Kecil-3-\(2021\).pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Tugas-Kecil-3-(2021).pdf))

Langkah pertama di dalam program ini adalah membuat graf yang merepresentasikan peta (di area tertentu, misalnya di sekitar kampus ITB). Sisi diperoleh dari jalan antar dua simpul dan bobot sisi adalah jarak Euclidean. Berdasarkan graf yang dibentuk, lalu program A* menerima input simpul asal dan simpul tujuan, lalu menentukan lintasan terpendek antara keduanya. Lintasan terpendek dapat ditampilkan pada peta/graf. Nilai heuristik yang dipakai adalah jarak garis lurus dari suatu titik ke tujuan.

Spesifikasi program:

1. Program menerima input filegraf (direpresentasikan sebagai matriks ketetanggaan 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.

Berkas yang dikumpulkan: Laporan berisi kode program, peta/graf input, screenshot peta yang memperlihatkan lintasan terpendek untuk sepasang simpul, alamat tempat kode sumber program diletakkan jika perlu dieksekusi oleh asisten. Tampilkan hasil untuk beberapa lintasan terpendek.

BAB II

Landasan Teori

2.1. A^*

Algoritma A^* memiliki tujuan mencari nilai minimum dari jumlah biaya suatu *path*. Ide dari A^* sendiri adalah dengan menghindari *path* yang terlalu mahal. A^* memiliki fungsi $f(n)=g(n)+h(n)$ dengan $g(n)$ adalah jarak dari awal sampai titik n , sementara $h(n)$ adalah estimasi jarak dari n ke titik akhir.

2.2. Skema Umum Algoritma A^*

Algoritma A^* secara garis besar adalah membandingkan jarak dari nodal start hingga titik tertentu dan mengeliminasi nodal yang terlalu mahal.

- 2.2.1. Traversal dimulai dari simpul v
- 2.2.2. kunjungi semua simpul yang bertetangga dengan simpul v terlebih dahulu
- 2.2.3. bandingkan simpul tetangga mana yang paling kecil, kemudian simpan kedua nilai total harga *path*
- 2.2.4. ulangi langkah dengan simpul yang paling murah.
- 2.2.5. lakukan perbandingan antara tiap nilai total harga *path*, bila di suatu titik harga lebih mahal, pindah ke simpul yang lebih murah.

2.3. Penjelasan *Jupyter Gmaps*

Pada Tugas besar kali ini, digunakan *library* *gmaps* dalam memvisualisasikan *path* yang dimiliki. *gmaps* sendiri dibuat dengan landasan ide menambahkan lapisan layer dari map google. Hal pertama yang harus dilakukan adalah dengan menggunakan API Google Maps. Dikarenakan harganya yang relatif mahal dan kami mahasiswa memiliki banyak kebutuhan lainnya, kami sekelompok memutuskan “patungan” dengan 2 kelompok lainnya demi tujuan menghemat uang kami yang tidak seberapa ini dalam membeli API. Map sendiri dapat divisualisasikan dengan *library* *gmaps* dengan cara menggunakan `gmaps.figure()` untuk menginisiasikan maps.

BAB III

Analisis Pemecahan Masalah

3.1. Langkah-Langkah Pemecahan Masalah

1. Euclidean Distance

Seperti konsepsi hakikatnya, jarak antara 2 node dihitung menggunakan Euclidean Distance. Euclidean Distance sendiri merupakan hasil buah pikir dari pitagoras, yang mencari jarak terpendek dari 2 titik dengan cara mengambil garis lurus yang menghubungkan 2 titik. Rumus dari Euclidean Distance sendiri adalah $\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$.

2. A Star

Bila kita ingin mencari biaya termurah dari suatu graf, yang pertama dilakukan adalah dengan membuat path minimum. Hal ini dapat dilakukan dengan cara:

1. Masukkan koordinat awal ke dalam path
2. Lakukan looping hingga node awal = node akhir
3. isi loopingnya adalah dengan mencari node ketetanggaan yang belum ada di dalam path dan juga jaraknya bersifat minimum dari antara arah lainnya.
4. Ulangi algoritma ini hingga selesai, bilamana terjadi kondisi dimana node awal tidak memiliki tetangga lain, disimpulkan node tersebut tidak benar.

BAB IV

Implementasi dan Pengujian

Pada bab ini akan dijelaskan mengenai bagaimana solusi penyelesaian masalah A^* .

4.1. Penjelasan Struktur Data

a. Euclidean Distance

```
def euclidean_distance(koordinat1, koordinat2):  
    return (  
        (float(koordinat1[0]) - float(koordinat2[0]))**2 +  
        (float(koordinat1[1]) - float(koordinat2[1]))**2  
    )**0.5
```

b. Do A Star

```
def do_a_star(node, nodeakhir, graf):  
    if node == nodeakhir:  
        return nodeakhir  
    else:  
        return "{} ->  
{ }".format(node, do_a_star(next_node(node, nodeakhir,  
graf), nodeakhir, graf))
```

c. Next Node

```
def next_node(node, nodeakhir, graf):  
    listtetangga = graf[node]["tetangga"]  
    minjarak =  
        (euclidean_distance(graf[node]['koordinat'],  
graf[listtetangga[0]]['koordinat'])  
        +  
        euclidean_distance(graf[listtetangga[0]]['koordinat'],  
graf[nodeakhir]['koordinat']))  
    mintetangga = listtetangga[0]  
    for tetangga in listtetangga:  
        newjarak =  
            (euclidean_distance(graf[node]['koordinat'],  
graf[tetangga]['koordinat'])
```

```
        +
euclidean_distance(graf[tetangga]['koordinat'], graf
[nodeakhir]['koordinat']))
        if (newjarak < minjarak):
            minjarak = newjarak
            mintetangga = tetangga
        print(minjarak)
    return mintetangga
```

d. Measure

```
def measure(lat1, lon1, lat2, lon2):
    R = 6378.137
    dLat = lat2 * math.pi / 180 - lat1 * math.pi /
180
    dLon = lon2 * math.pi / 180 - lon1 * math.pi /
180
    a = (math.sin(dLat/2) * math.sin(dLat/2) +
        math.cos(lat1 * math.pi / 180) * math.cos(lat2
* math.pi / 180) *
        math.sin(dLon/2) * math.sin(dLon/2))
    c = 2 * math.atan2((a)**0.5, (1-a)**0.5)
    d = R * c
    return d * 1000
```

4.2. Graf Input dan Peta Lintasan Terpendek

oh galih oh ratna

1. Buah Batu

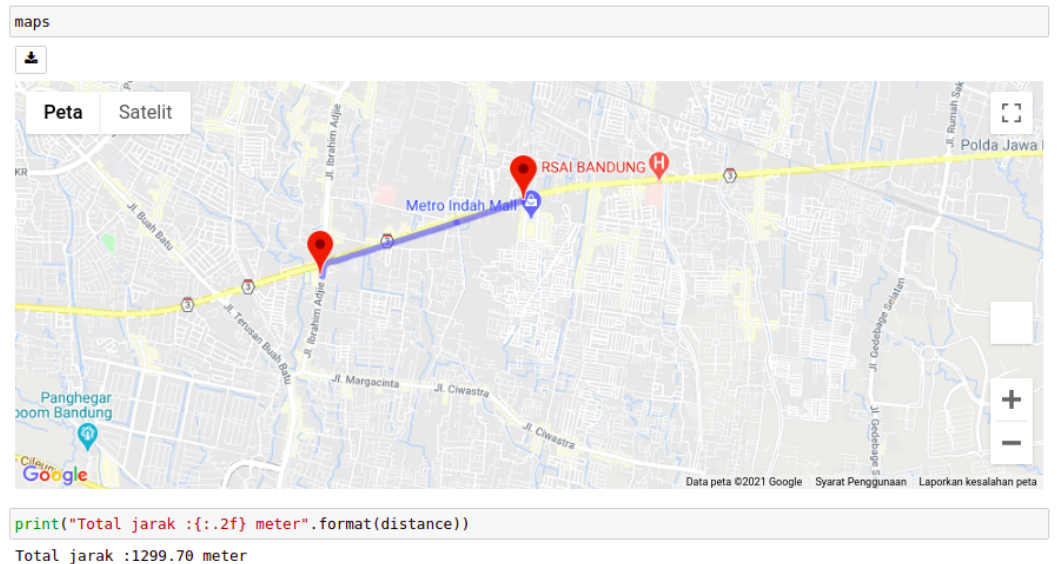
input :
buahbatu.txt

```
8
SoeHattaMTC:-6.940351,107.658245
Rancabolang:-6.939252,107.663915
Borma:-6.943234,107.663564
SoeHatta:-6.942138,107.652719
Ciwastra:-6.955690,107.654484
HarapanBunda:-6.956029,107.662112
BuahBatu:-6.954222,107.639885
Carrefour:-6.946367,107.641756
[0,1,0,1,0,0,0,0]
[1,0,1,0,0,0,0,0]
```

```
[0,1,0,0,0,1,0,0]
[1,0,0,0,1,0,0,1]
[0,0,0,1,0,1,1,0]
[0,0,1,0,1,0,0,0]
[0,0,0,0,1,0,0,1]
[0,0,0,1,0,0,1,0]
```

Output :

Berikut merupakan list node:
1. SoeHattaMTC
2. Rancabolang
3. Borma
4. SoeHatta
5. Ciwastra
6. HarapanBunda
7. BuahBatu
8. Carrefour
Masukkan nomor node awal: 1
Masukkan nomor node tujuan: 8



2. Alun - Alun Bandung

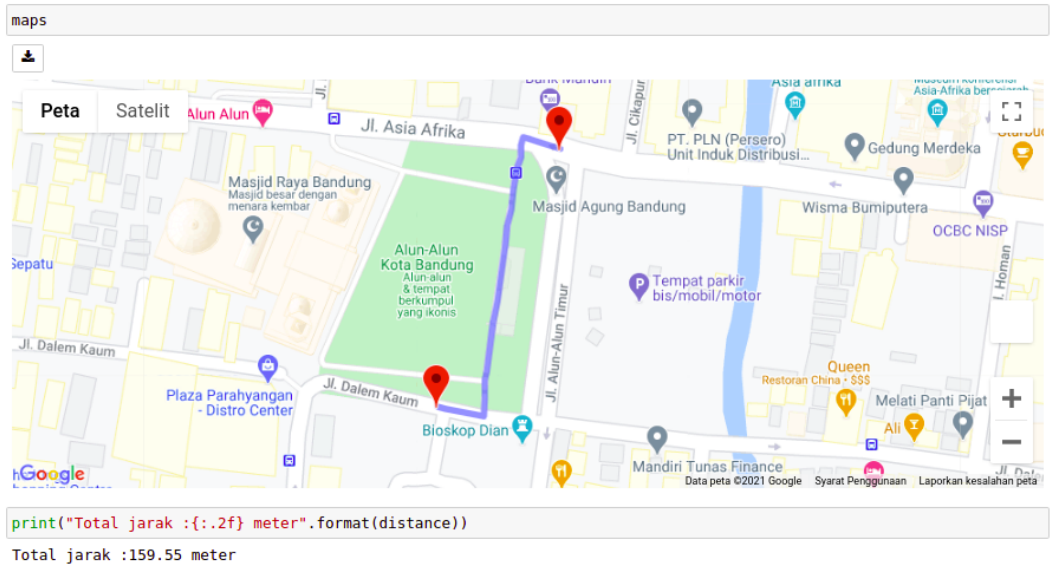
input :

alunalun.txt


```
8
SimpangTugu:-6.921210,107.607689
Museum:-6.921023,107.609848
Braga:-6.916968,107.609178
Mercure:-6.923878,107.611800
Sudirman:-6.920395,107.600530
PendopoKota:-6.922503,107.607066
PasarBaru:-6.917585,107.604353
Kingsley:-6.919173,107.615111
[0,1,0,1,1,1,0]
[1,0,1,1,0,0,0]
[0,1,0,0,0,0,1]
[1,1,0,0,0,1,0]
[1,0,0,0,0,0,1]
[1,0,0,1,0,0,0]
[1,0,0,0,1,0,0]
[0,0,1,0,0,0,0]
```

Output :

Berikut merupakan list node:
 1.SimpangTugu
 2.Museum
 3.Braga
 4.Mercure
 5.Sudirman
 6.PendopoKota
 7.PasarBaru
 8.Kingsley
 Masukkan nomor node awal: 1
 Masukkan nomor node tujuan: 6



3. ITB

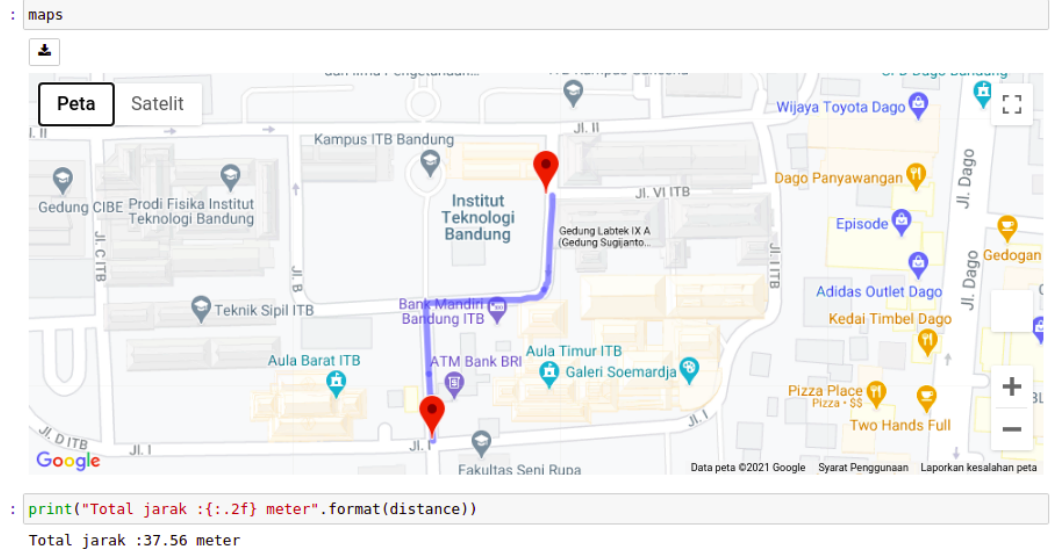
input :

itb.txt

```
8
Gerbang:-6.892615347458373, 107.61043109261696
JL 1 Kanan:-6.8924875318819625,107.61186339214424
JL 1 IMG:-6.891342515805926,107.61219062162424
JL VI:-6.891363347430485,107.6110159993639
ATM Bukopin:-6.891847982458015,107.61099454169307
JL B:-6.8919345243750785,107.61039104470129
JL A:-6.892278550874567,107.61041050192205
JL 1 Kiri:-6.892667988160328,107.60878036423408
[0,1,0,0,0,0,1,1]
[1,0,1,0,0,0,0,0]
[0,1,0,1,0,0,0,0]
[0,0,1,0,1,0,0,0]
[0,0,0,1,0,1,1,0]
[0,0,0,0,1,0,1,0]
[1,0,0,0,1,1,0,0]
[1,0,0,0,0,0,0,0]
```

Output :

Berikut merupakan list node:
 1. Gerbang
 2. Jl. 1 Kanan
 3. Jl. 1 IMG
 4. Jl. VI
 5. ATM Bukopin
 6. Jl. B
 7. Jl. A
 8. Jl. 1 Kiri
 Masukkan nomor node awal: 4
 Masukkan nomor node tujuan: 1



4. Dadap

input :

dadap.txt

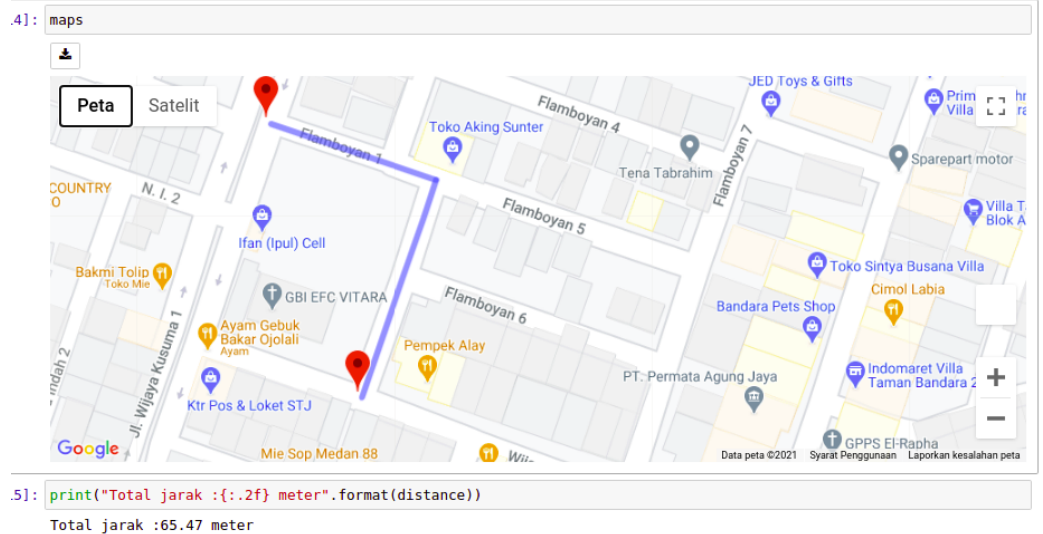
```

9
AyamGebuk:-6.0938061466451865, 106.6966327623154
PempekAlay:-6.093915565939859, 106.6970624466169
Wijaya4:-6.094181298419215, 106.69778306559265
AyamMami:-6.0942935323900524, 106.69817184641683
Flamboyan1:-6.093186156184438, 106.69681738419062
TokoAking:-6.09335824858256, 106.69725131375567
Flamboyan7:-6.093575745954564, 106.6979686763123
Flamboyan6:-6.093856657920988, 106.69789213958865
Wijaya4Flamboyan6:-6.093618861437266,
106.69718662242383
[0,1,1,1,1,0,0,0]
[1,0,1,1,0,1,0,0,1]
[1,1,0,1,0,0,1,1,0]
[1,1,1,0,0,0,0,0,0]
[1,0,0,0,0,1,1,0,0]
[0,1,0,0,1,0,1,0,1]
    
```

```
[0,0,1,0,1,1,0,1,0]
[0,0,1,0,0,0,1,0,1]
[0,1,0,0,0,1,0,1,0]
```

Output :

Berikut merupakan list node:
 1. AyamGebuk
 2. PempekAlay
 3. Wijaya4
 4. AyamMami
 5. Flamboyon1
 6. TokoAking
 7. Flamboyon7
 8. Flamboyon6
 9. Wijaya4Flamboyon6
 Masukkan nomor node awal: 5
 Masukkan nomor node tujuan: 2



5. Bandung

input :

bdg.txt

```
9
Pasir,Koja-Soekarno,Hatta:-6.930397349712235,107.576062784
12402
SoekarnoHatta-Kopo:-6.945605940280078,107.5897565463632
9
SoekarnoHatta-LeuwiPanjang:-6.94730995342355,107.5951653
6781202
Peta-Kopo:-6.936915377247668,107.59503658634895
Peta-LeuwiPanjang:-6.937810001438914,107.59696830829492
Peta-Otista:-6.937085781986556,107.60306396421336
PasirKoja-AstanaAnyar:-6.926861388562902,107.59988735479
```

108
PasirKoja-Kopo:-6.926690980126949,107.59829905007993
Peta-PasirKoja-Jamika:-6.926776184352622,107.585506758082
18
[0,1,0,0,0,0,0,1]
[1,0,1,1,0,0,0,0]
[0,1,0,0,1,0,0,0]
[0,1,0,0,1,0,0,1]
[0,0,1,1,0,1,0,0]
[0,0,0,0,1,0,1,0]
[0,0,0,0,0,1,0,1]
[0,0,0,1,0,0,1,0]
[1,0,0,1,0,0,0,1]

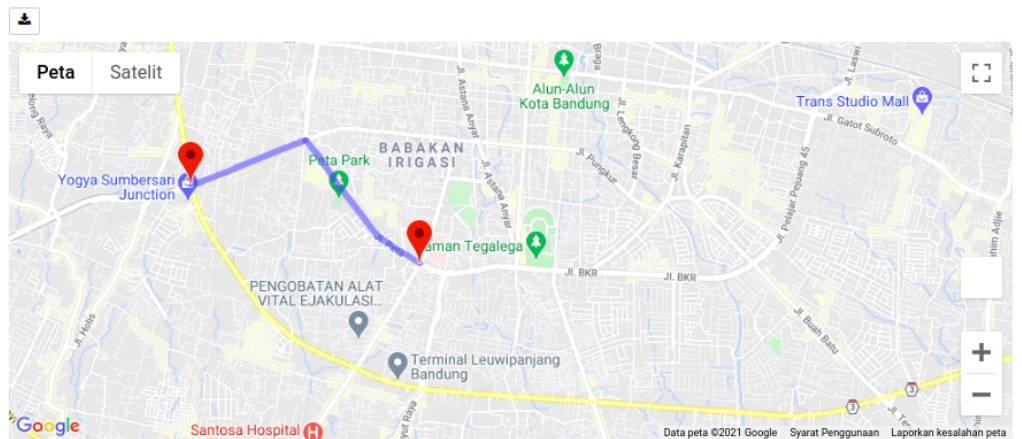
Output :

```

Berikut merupakan list node:
1.Pasir,Koja-Soekarno,Hatta
2.SoekarnoHatta-Kopo
3.SoekarnoHatta-LeuwiPanjang
4.Peta-Kopo
5.Peta-LeuwiPanjang
6.Peta-Otista
7.PasirKoja-AstanaAnyar
8.PasirKoja-Kopo
9.Peta-PasirKoja-Jamika
Masukkan nomor node awal: 4
Masukkan nomor node tujuan: 1

```

```
]: maps
```



```
]: print("Total jarak :{:0.2f} meter".format(distance))
```

Total jarak :1118.77 meter

6. City in Java

input :

cityinjava.txt

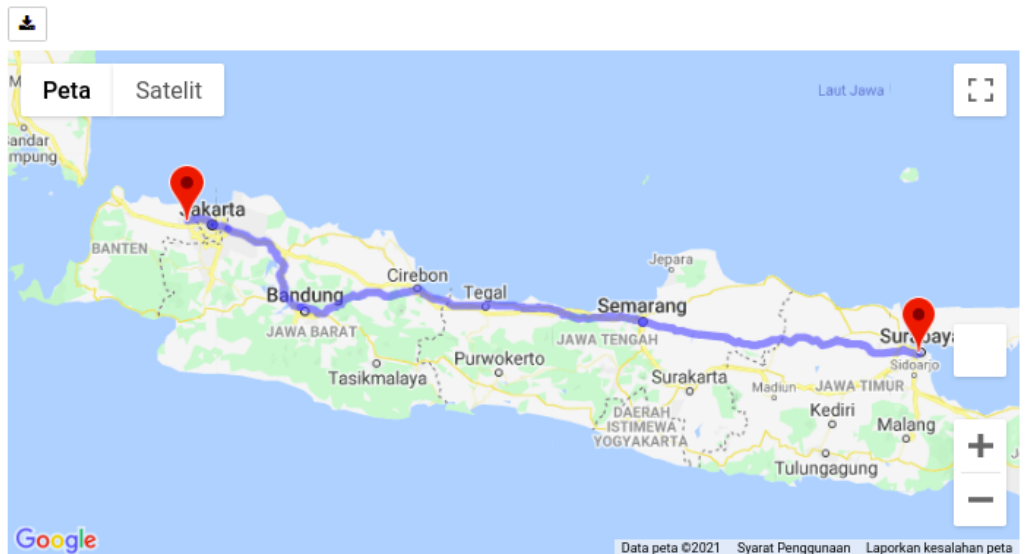
8
Tangerang:-6.171735198813916,106.63677675214166

```
Jakarta:-6.209621461822658,106.84764764863442
Depok:-6.400428103357031,106.80122130287162
Bogor:-6.5990406055062705,106.8120701983816
Bekasi:-6.239336066722454,106.97804215105538
Bandung:-6.878534382551147,107.57976471826818
Semarang:-7.012419413547074,110.43110305103966
Surabaya:-7.266792435809086,112.73972600316542
[0,1,0,1,0,0,0,0]
[1,0,1,0,1,0,0,0]
[0,1,0,1,0,0,0,0]
[1,0,1,0,0,0,0,0]
[0,1,0,0,0,1,0,0]
[0,0,0,0,1,0,1,0]
[0,0,0,0,0,1,0,1]
[0,0,0,0,0,0,1,0]
```

Output :

Berikut merupakan list node:
1.Tangerang
2.Jakarta
3.Depok
4.Bogor
5.Bekasi
6.Bandung
7.Semarang
8.Surabaya
Masukkan nomor node awal: 1
Masukkan nomor node tujuan: 8

: maps



4.3. Tabel Mempermudah Asisten

Link Github : https://github.com/zaidannaufal/tucil_stima_3

No	Jenis	Ketercapaian
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	✓

BAB V

Kesimpulan dan Saran

5.1. Kesimpulan

Dengan mengimplementasikan konsep dan teori dari Strategi Algoritma, terutama materi mengenai algoritma A*, ke dalam program pencarian jalur terdekat., kita dapat mengetahui jalur dan keterhubungan antar nodal sedemikian rupa sehingga dapat dilacak keterhubungan terpendeknya.

5.2. Saran

Kami sebagai penulis sadar bahwasanya program ini jauh dari kata sempurna, program ini akan lebih baik bilamana :

- a. Memiliki GUI yang lebih interaktif serta *good-looking*
- b. Mengurangi rekursif dan looping yang banyak sehingga algoritma dapat bekerja lebih efektif dan efisien
- c. Menambahkan fungsi-fungsi objektif yang mampu mengefisienkan algoritma

5.3. Refleksi

Tugas ini mengajarkan betapa pentingnya manajemen waktu serta pentingnya pemahaman atas diktat dasar bahasa pemrograman terutama untuk pemrograman berorientasi objek dan prosedural pada bahasa python dan juga betapa pentingnya memiliki uang agar dapat membeli API Google Maps. Oleh sebab itu kami akan berusaha keras agar menjadi orang yang mampu membeli API Google Maps dan menjadi programmer handal.

DAFTAR PUSTAKA

Anany Levitin, Introduction to the Design & Analysis of Algorithms, 3rd Edition, Addison-Wesley, 2012

Bhardwaj, Anuj; Verma, Parag, Parag, Design and Analysis of Algorithm, , Alpha Science International Ltd, 2017