# Penerapan Konsep Graf dalam Penentuan Rute pada Google Maps

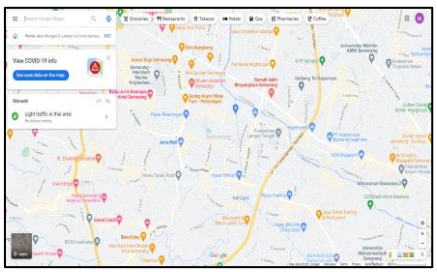
Mochammad Fatchur Rochman - 13519009
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*13519009@std.itb.ac.id*

***Abstract*—Kemajuan teknologi membawa banyak kemudahan salah satunya adalah kemudahan untuk mengetahui informasi suatu lokasi. Google Maps merupakan aplikasi pemetaan web yang memberikan akses peta seluruh dunia dan salah satu kemampuannya adalah dapat memberikan rute perjalanan suatu tempat. Dalam menentukan rute perjalanan suatu tempat Google Maps menerapkan konsep graf untuk memetakan koordinat lokasi suatu tempat yaitu garis lintang/*latitude* dan garis bujur/*longitude* kemudian dengan algoritma penentuan rute didapatkan rute perjalanan yang diinginkan. Pada makalah ini penulis akan mencoba membahas bagaimana konsep graf tersebut diterapkan.**

***Keywords*—Google Maps, graf, garis lintang, garis bujur**

## I. PENDAHULUAN

Google Maps adalah aplikasi pemetaan web yang dikembangkan oleh perusahaan Google yang memberikan citra satelit, peta jalan, panorama 360◦, kondisi lalu lintas, dan perencaan rute perjalanan dengan berjalan kaki, mobil, atau angkutan umum.



Gambar 1. Tampilan Google Maps pada Web
(Sumber : Dokumentasi Penulis)

Dengan menggunakan aplikasi Google Maps ketika kita hendak berpergian ke suatu tempat kita dapat dengan mudah menentukan rute perjalanan yang harus ditempuh dengan memasukkan lokasi tujuan ke aplikasi Google Maps, kemudian aplikasi tersebut akan memberikan berbagai rute perjalanan yang dapat dicapai dan memberikan rute yang paling sesuai dengan kondisi kita saat ini dan tentunya rute yang paling efisien.

Gambar 2. Tampilan Google Maps ketika menentukan Rute Perjalanan
(Sumber : Dokumentasi Penulis)

Rute perjalanan yang diberikan Google Maps tersebut didapatkan dari hasil algoritma pencarian rute dari graf peta dunia yang ukurannya sangat besar yang menghubungkan satu tempat ke tempat lainnya.

Dalam makalah ini penulis akan menggunakan penerapan algoritma yaitu A* untuk menentukan rute perjalanan dari satu tempat ke tempat lainnya sebagai pendekatan algoritma pencarian rute yang bekerja di Google Maps.

## II. DASAR TEORI

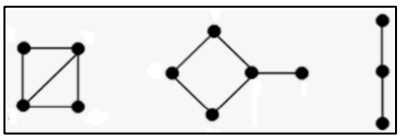
### A. Definisi Graf

Graf dalam matematika dan informatika merupakan struktur yang merepresentasikan objek-objek diskrit yang ditandai dengan simpul (*vertex*) dan sisi (*edge*) atau dapat digambarkan juga sebagai himpunan dari objek-objek yang dinamakan titik, simpul, atau sudut (*vertex*) yang dihubungkan oleh penghubung yang dinamakan garis atau sisi (*edge*). Secara formal, graf didefinisikan sebagai G = (V, E) dengan G adalah graf, V adalah sebuah sebuah himpunan yang elemennya dinamakan sudut atau simpul, V = {$v_1$, $v_2$, $v_3$, …, $v_n$}. E adalah himpunan dari pasangan - pasangan simpul yang terpisah, yang dinamakan sisi atau garis, E = {$e_1$, $e_2$, $e_3$, …, $e_n$}.

### B. Jenis-Jenis Graf

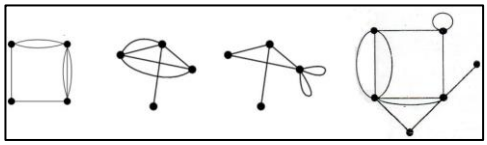
Berdasarkan ada tidaknya gelang atau sisi-ganda pada graf, graf dibedakan menjadi dua jenis:

1. Graf sederhana (*simple graph*)
   Graf sederhana merupakan graf yang didalamnya tidak mengandung gelang maupun sisi ganda.



Gambar 3. Graf sederhana
(Sumber : PPT Rinaldi Munir/IF2120/ Graf bag.1)

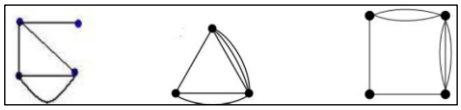
2. Graf tak-sederhana (*unsimple-graph*)
   Graf tak-sederhana merupakan graf yang didalamnya mengandung sisi ganda atau gelang.



Gambar 4. Graf sederhana
(Sumber : PPT Rinaldi Munir/IF2120/ Graf bag.1)

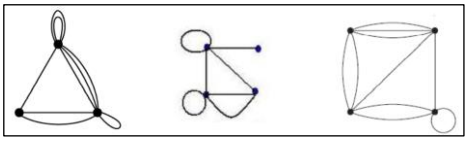
Graf tak-sederhana dibedakan lagi menjadi dua jenis:

a. Graf ganda (*multi graph*)
   Graf ganda merupakan graf yang mengandung sisi ganda.



Gambar 5. Graf ganda
(Sumber : PPT Rinaldi Munir/IF2120/ Graf bag.1)

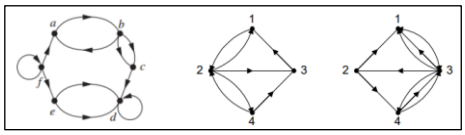
b. Graf semu (*pseudo graph*)
   Graf semu merupakan graf yang mengandung sisi gelang.



Gambar 6. Graf semu
(Sumber : PPT Rinaldi Munir/IF2120/ Graf bag.1)

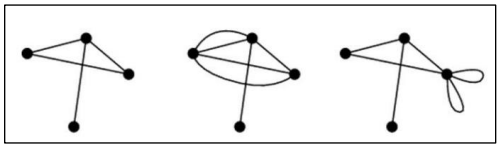
Berdasarkan ada atau tidaknya orientasi arah pada sisi, graf dapat dibagi menjadi dua jenis:

1. Graf berarah (*directed graph*)
   Graf berarah adalah graf yang memiliki orientasi arah pada setiap sisinya



Gambar 7. Graf berarah
(Sumber : PPT Rinaldi Munir/IF2120/ Graf bag.1)

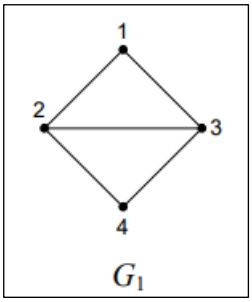
2. Graf tak-berarah (*indirected graph*)
   Graf tak-berarah adalah graf yang sisinya tidak mempunyai orientasi arah



Gambar 8. Graf tak-berarah
(Sumber : PPT Rinaldi Munir/IF2120/ Graf bag.1)

C. Terminologi Graf

Dalam graf terdapat istilah-istilah/terminology yang digunakan , diantaranya :




Gambar 9. Graf G1
(Sumber : PPT Rinaldi Munir/IF2120/ Graf bag.1)

1. Ketetanggaan (*adjacency*)
   Dua buah simpul dikatakan bertetangga apabila kedua simpul terhubung langsung oleh sebuah sisi.
   Sebagai contoh pada Gambar 9, simpul 1 bertetangga dengan simpul 2 dan simpul 3.

2. Bersisian (*Incidency*)
   Sebuah sisi e dikatakan bersisian dengan simpul $v_i$ dan simpul $v_j$ apabila terdapat sembarang sisi $e = (v_i , v_j)$.
   Sebagai contoh pada Gambar 9, sisi 1 bersisian dengan simpul 2 dan simpul 3.

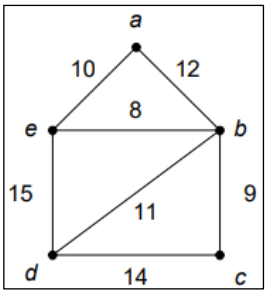
3. Derajat (*Degree*)
   Derajat suatu simpul menyatakan jumlah sisi yang bersisian dengan simpul tersebut atau jumlah simpul lain yang terhubung dengan simpul tersebut. Dengan notasi *d(v)* untuk graf tidak berarah. Pada graf berarah terdapat *din(v)* yaitu jumlah sisi yang bersisian dengan simpul tersebut dengan arah masuk dan *dout(v)* yaitu jumlah sisi yang bersisian dengan simpul tersebut dengan arah keluar.
   Sebagai contoh pada Gambar 9, simpul 1 memiliki derajat yaitu *d(1)* = 2.

4. Lintasan (*Path*)
   Lintasan dengan panjang n dari simpul awal $v_0$ ke simpul tujuan $v_n$ di dalam graf G merupakan barisan berselang seling simpul-simpul dan sisi-sisi yang berbentuk $v_0, e_1, v_1, e_2, v_2, \dots , v_{n-1}, e_n, v_n$ sedemikian sehingga sisi-sisi dari graf G adalah $e_1 = (v_0, v_1), e_2 = (v_1, v_2), \dots , e_n = (v_{n-1}, v_n)$.

5. Sirkuit (*Circuit*)
   Sirkuit merupakan lintasan yang berawal dan berakhir pada simpul yang sama.

6. Keterhubungan (*Connected*)
   Simpul $v_1$ dan $v_2$ disebut terhubung jika terdapat lintasan dari $v_1$ ke $v_2$ dan graf G disebut graf terhubung (connected graph) jika terdapat lintasan dari $v_i$ ke $v_j$ untuk setiap pasang simpul $v_i$ dan $v_j$ dalam himpunan V.

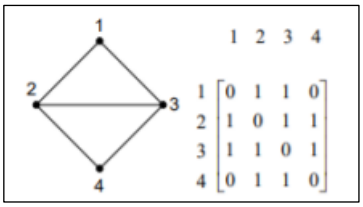
7. Graf berbobot (*Weighted graph*)
   Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga (bobot).




Gambar 10. Graf berbobot
(Sumber : PPT Rinaldi Munir/IF2120/ Graf bag.1)

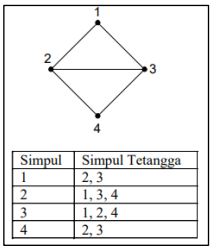
D. Representasi Graf
Dalam praktiknya di dalam pemrograman, graf dapat direpresentasikan menjadi 3 bentuk :

1. Matriks Ketetanggan (*Adjacency Matrix*)
   Pada matriks ketetanggan, baris dan kolom matriks akan berisi 1 dan 0 untuk graf tak berbobot. Untuk setiap A = [$a_{ij}$], jika simpul *i* dan *j* betetangga maka $a_{ij}$ akan bernilai 1, sedangkan jika simpul *i* dan *j* tidak bertetangga maka $a_{ij}$ akan bernilai 0.



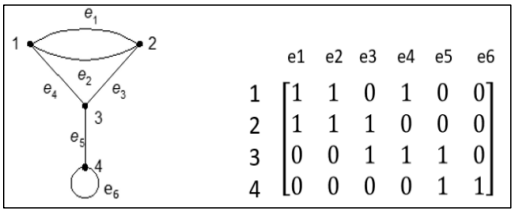

Gambar 11. Representasi dengan Matrix Ketetanggan
(Sumber : PPT Rinaldi Munir/IF2120/ Graf bag.2)

2. Adjacency List
   Pada adjacency list, keterhubungan antar simpul didefinisikan dengan menggunakan linked list. Setiap simpul memiliki list dari simpul-simpul yang terhubung dengan simpul tersebut




Gambar 12. Representasi dengan Adjacency List
(Sumber : PPT Rinaldi Munir/IF2120/ Graf bag.2)

3. Matriks Bersisian (*Incidency Matrix*)
   Pada matriks bersisian, baris dan kolom matriks berisi 1 dan 0 untuk graf tak berbobot, Untuk setiap A = [aij], jika simpul i dan j bertetangga maka aij akan bernilai 1, jika simpul i bersisian dengan sisi j, dan aij bernilai 0 jika simpul i tidak bersisian dengan sisi j




Gambar 13. Representasi dengan Matriks Bersisian
(Sumber : PPT Rinaldi Munir/IF2120/ Graf bag.2)

E. Algoritma A*
Algoritma A* merupakan algoritma *pathfinding* yang sering digunakan dalam penentuan rute dan graph traversal. Algoritma A* memiliki beberapa optimisasi heuristic yang digunakan untuk mencapai hasil yang lebih baik. Algoritma A* menggunakan Best First Search untuk menemukan jalur dengan jumlah weight terkecil dari simpul awal (simpul *start*) ke simpul akhir (simpul lokasi). Algoritma ini menggunakan fungsi estimasi yaitu $f(n) = g(n) + h(n)$, dengan g(n) merupakan penjumlahan *weight* dari simpul *start* sampai ke simpul *n*, dan h(n) merupakan perkiraan/estimasi *weight* dari simpul n ke simpul lokasi untuk menentukan urutan pencarian. Algoritma ini menggunakan dua list yang biasanya dinamai open dan closed. List open menyimpan simpul-simpul yang pernah dicek tetapi belum terpilih sebagai simpul terbaik dan closed menyimpan simpul yang telah terpilih menjadi simpul terbaik. Beberapa terminologi yang digunakan pada algoritma A* adalah :

1. f(n) = Estimasi weight terendah.
2. g(n) = Weight dari simpul *start* ke simpul *n*.
3. h(n) = Perkiraan weight dari simpul *n* ke simpul *lokasi/akhir* atau fungsi heuristic.

Algoritma A*
Algoritma ini dalam implementasinya memanfaatkan dua list yaitu *open-list* dan *closed-list*

1. Inisialisasi *open-list*, letakkan *starting-node* pada *open-list*
2. Inisialisasi *closed-list*
3. Selama *open-list* masih kosong
   - Cari *node* dengan **f** paling kecil pada *open-list*, missal kita panggil "q".
   - Pop/keluarkan "q" dari *open-list*
   - *Generate* simpul-simpul yang merupakan *child* dari simpul q
   - Untuk setiap suksesor,
     - Apabila suksesor tersebut adalah ***Goal/simpul tujuan***, hentikan pencarian.

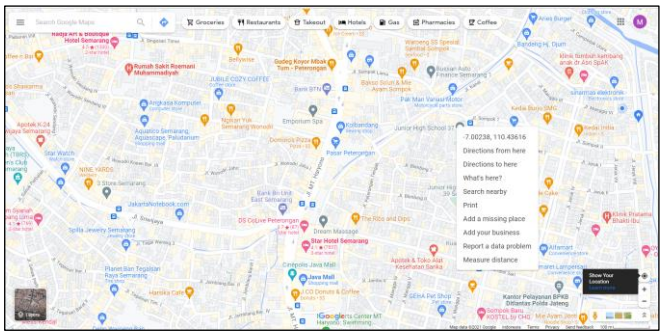
- *Suksesor.**g*** = ***q.g* + *jarak antara suksor dan q***
- *Suksesor.**h*** = ***jarak dari Goal ke suksesor***
- *Suksesor.**f*** = *Suksesor.**g*** + *Suksesor.**h***
- Apabila *node* dengan posisi yang sama dengan suksesor sudah ada di *open-list* dengan nilai **f** yang lebih kecil dari suksesor, maka abaikan suksesor tersebut.
- Apabila *node* dengan posisi yang sama dengan suksesor sudah ada di *closed-list* dengan nilai **f** yang lebih kecil dari suksesor, maka abaikan atau masukan ke *open-list*
- Akhir perulangan (*for loop*)

o *Push*/masukkan "q" ke *closed-list*

4. Akhir (*while loop*)

## III. APLIKASI KONSEP GRAF PADA GOOGLE MAPS

### *A. Pembentukan Graf*



Gambar 14. Koordinat suatu tempat pada Google Maps
(Sumber : Dokumentasi Penulis)

*Google Maps* memetakan tempat-tempat di seluruh dunia berdasarkan koordinat dari garis lintang/ *latitude* dan garis bujur/*longitude* dari tempat/lokasi itu, koordinat-koordinat tersebut menjadi identitas dari suatu lokasi/tempat di bumi, yang mana dalam graf kita anggap sebagai *simpul/node*. Kemudian terbentuklah kumpulan simpul lokasi, dari kumpulan simpul lokasi tersebut tempat-tempat yang terhubung/ bisa diakses seperti jalan raya, dihubungkan antara satu simpul dengan simpul lainnya.

Dalam makalah ini, akan diambil sebuah peta kecil yaitu peta ITB untuk menggambarkan bagaimana tempat-tempat di peta ITB saling terhubung. Dalam peta kecil ini memetakan 12 titik lokasi di ITB yaitu Jalan A-I, Jalan A-B, Jalan B-VI, Jalan B-II Timur, Jalan B-II Barat, Jalan C-D-I, Jalan C-II, Jalan D-II, Jalan G-II, Jalan I-VI, Jalan H-I-II, Jalan G-IV. Koordinat garis lintang dan garis bujur dari masing-masing lokasi adalah sebagai berikut :

```
listKoordinat = [[-6.892600475482855, 107.61042823898438],
                 [-6.891931820281012, 107.61038897538347],
                 [-6.891318635643979, 107.61104437607533],
                 [-6.891008294957601, 107.61105041663436],
                 [-6.891032297939212, 107.6097228808863],
                 [-6.892657579894951, 107.60876370074594],
                 [-6.891056442869117, 107.6087061492608],
                 [-6.8910484769940705, 107.60821107844863],
                 [-6.8909909675987135, 107.61157998012055],
                 [-6.891270520196028, 107.61218224512506],
                 [-6.890981841922636, 107.61210783772913],
                 [-6.889890806018179, 107.61156852669643]
                 ]
listNode = ["Jalan A-I",
            "Jalan A-B",
            "Jalan B-VI",
            "Jalan B-II Timur",
            "Jalan B-II Barat",
            "Jalan C-D-I",
            "Jalan C-II",
            "Jalan D-II",
            "Jalan G-II",
            "Jalan I-VI",
            "Jalan H-I-II",
            "Jalan G-IV"
            ]
```

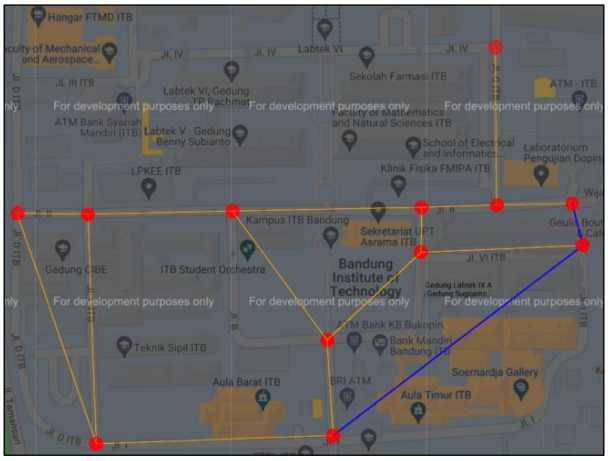
Gambar 15. *Latitude* dan *Longitude* dari masing-masing lokasi
(Sumber : Dokumentasi Penulis)

Dari kumpulan koordinat-koordinat pada Gambar 15, terbentuklah kumpulan dari simpul lokasi, dari kumpulan simpul lokasi tersebut dibentuklah graf, dalam makalah ini penulis merepresentasikan peta ITB dengan representasi Matrix Ketetanggaan/*adjacency matrix*. Dengan adjacency matrix sebagai berikut:

```
adjacencyMatrix = [[0,1,0,0,0,1,0,0,0,1,0,0],
                   [1,0,1,0,1,0,0,0,0,0,0,0],
                   [0,1,0,1,0,0,0,0,0,1,0,0],
                   [0,0,1,0,1,0,0,0,1,0,0,0],
                   [0,1,0,1,0,0,1,0,0,0,0,0],
                   [1,0,0,0,0,0,1,1,0,0,0,0],
                   [0,0,0,0,1,1,0,1,0,0,0,0],
                   [0,0,0,0,0,1,1,0,0,0,0,0],
                   [0,0,0,1,0,0,0,0,0,0,1,1],
                   [1,0,1,0,0,0,0,0,0,0,1,0],
                   [0,0,0,0,0,0,0,0,1,1,0,0],
                   [0,0,0,0,0,0,0,0,1,0,0,0]
                   ]
```

Gambar 16. Matrix Ketetanggaan/ *Adjacency Matrix* untuk peta ITB
(Sumber : Dokumentasi Penulis)

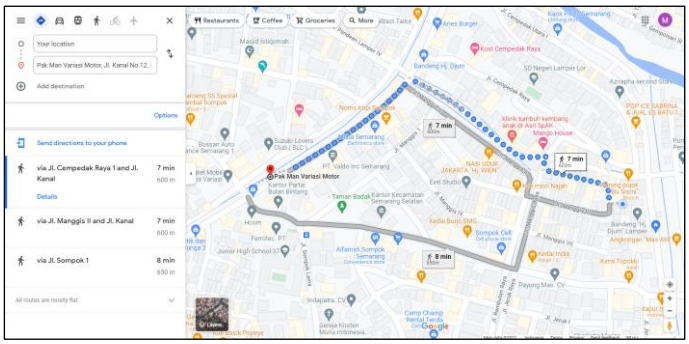
Dengan hubungan simpul-simpul yang ditujukkan pada matrix ketetanggaan pada Gambar 16. maka graf yang terbentuk ditunjukkan pada Gambar 17. Graf Peta ITB.




Gambar 17. Graf Peta ITB
(Sumber : Dokumentasi Penulis)

Graf yang pada Gambar 17, memperlihatkan bagaimana lokasi-lokasi di ITB dihubungkan dari satu lokasi ke lokasi lainnya. Pada Google Maps, graf yang terbentuk sama hal-nya dengan graf yang terbentuk pada Gambar 17, Namun lebih besar atau bisa dikatakan *Big Graph* dan tentunya representasi graf-nya lebih optimum.

### *B. Penentuan Rute*



Gambar 18. Rute Perjalanan yang ditampilkan oleh Google Maps
(Sumber : Dokumentasi Penulis)

Setelah Graf dari peta dunia/ suatu peta terbentuk, penentuan rute perjalanan dapat dilakukan menggunakan algoritma *pathfinding*. Penentuan rute perjalanan yang ditampilkan pada Gambar 18, merupakan hasil dari algoritma *pathfinding*, dalam makalah ini algoritma *pathfinding* yang akan digunakan adalah algoritma A*.

### *C. Fungsi Estimasi f(n)*

Dalam mengimplementasikan algoritma A* untuk mencari rute suatu perjalanan, harus ditentukkan terlebih dahulu fungsi estimasi yang akan digunakan pada algoritma A*, pada makalah ini fungsi estimasi yang digunakan adalah sebagai berikut

f(n) = g(n) + h(n)
g(n) = Weight dari simpul *start* ke simpul *n*.
h(n) = Euclidean distance koordinat simpul *n* dan simpul tujuan

$$EuclideanDistance(simpul_1, simpul_2) = \sqrt{(simpul_1.x - simpul_2.x)^2 + (simpul_1.y - simpul_2.y)^2}$$

### *D. Implementasi A* dalam Penentuan Rute dalam Bahasa Python*

Dalam mengimplementasikannya ada beberapa fungsi-fungsi tambahan, sebagai berikut:

1. Fungsi *Euclidean Distance*
   Fungsi ini digunakan untuk mencari jarak antara dua simpul.

```
def h(p1,p2):
    return f.euclideanDistance(p1,p2)
```

Gambar 19. Implementasi *Euclidean Distance*
(Sumber : Dokumentasi Penulis)

2. Fungsi *reconstructPath*
   Fungsi ini digunakan untuk menggenerate *closed-list*, yaitu rute perjalanan yang merupakan hasil dari algoritma A*.

```
def reconstructPath(cameFrom, current):
    # current sebelum while loop adalah end Of path/goal
    # akan dilakukan backtracking
    totalPath = [current]
    while current in cameFrom:
        current = cameFrom[current]
        totalPath = [current] + totalPath
    return totalPath
```

Gambar 20. Implementasi *reconstructPath*
(Sumber : Dokumentasi Penulis)

Implementasi algoritma A* adalah sebagai berikut :

- Mula - mula akan dibuat openSet yang berupa Priority Queue, yang beranggota awal yaitu simpul awal
- Selama openSet masih belum kosong, node dengan nilai f paling kecil akan dikeluarkan dari queue/antrian
- Selama belum ditemukan jalan menuju ke simpul goal dan simpul saat ini masih memiliki neighbors dilakukan update f dan g dan neighbors tersebut dimasukkan ke antrian openSet
- Algoritma ini akan berakhir hingga node yang dikeluarkan dari antrian (node dengan nilai f paling kecil) sama dengan simpul goal

```
def aStar(g, start, goal):
    numOfNode = g.getNumOfNode()

    count = 0

    openSet = PriorityQueue()
    openSet.put((0, count, start))
    cameFrom = {}

    gScore = [float("inf") for j in range(numOfNode)]
    gScore[start] = 0

    fScore = [float("inf") for j in range(numOfNode)]
    fScore[start] = h(g.getNodeCoordinate(start),g.getNodeCoordinate(goal))

    openSetHash = {start} # nantinya untuk mengetahui apakah suatu node/neighbor sudah ada di openList

    while (not openSet.empty()):
        current = openSet.get()[2]
        openSetHash.remove(current)

        if(current == goal):
            return reconstructPath(cameFrom, current)

        for neighbor in g.getNeighbors(current):
            temp_gScore = gScore[current] + f.euclideanDistance(g.getNodeCoordinate(current), g.getNodeCoordinate(neighbor))
            if(temp_gScore < gScore[neighbor]):
                cameFrom[neighbor] = current
                gScore[neighbor] = temp_gScore
                fScore[neighbor] = temp_gScore + h(g.getNodeCoordinate(neighbor), g.getNodeCoordinate(goal))
                if (neighbor not in openSetHash):
                    count += 1
                    openSet.put((fScore[neighbor], count, neighbor))
                    openSetHash.add(neighbor)

    return -1
```

Gambar 21. Implementasi algoritma A*
(Sumber : Dokumentasi Penulis)

### *E. Pengujian/Eksperimen*

Dalam pengujian ini akan ditunjukkan bagaimana algoritma A* dapat digunakan untuk menentukan rute perjalanan dari graf peta ITB sebagai pendekatan untuk penentuan rute pada Google Maps. Misalnya dari Jalan A-I ingin menuju ke Jalan H-I-II.

```
List of node dari peta yang dipilih:
-  Jalan A-I
-  Jalan A-B
-  Jalan B-VI
-  Jalan B-II Timur
-  Jalan B-II Barat
-  Jalan C-D-I
-  Jalan C-II
-  Jalan D-II
-  Jalan G-II
-  Jalan I-VI
-  Jalan H-I-II
-  Jalan G-IV

Masukkan node start: Jalan A-I
Masukkan node tujuan: Jalan H-I-II

Rute Terpendeknya adalah:
Jalan A-I --> Jalan I-VI --> Jalan H-I-II

Jarak terpendek dari Jalan A-I menuju Jalan H-I-II adalah 33.1865811915439 meter.
```

Gambar 22. Interface Program
(Sumber : Dokumentasi Penulis)


Gambar 23. Rute Perjalanan yang ditawarkan A*
(Sumber : Dokumentasi Penulis)

Dari Gambar 21 dan Gambar 22, ketika ingin berpergian dari Jalan A-I ingin menuju ke Jalan H-I-II, rute yang harus dilalui menurut A* adalah Jalan A-I ke Jalan I-VI ke Jalan H-I-II, pada Google Maps hal yang sama juga terjadi seperti ditunjukkan pada Gambar 18.

## IV. KESIMPULAN

Google Maps dalam melakukan penentuan rute perjalanan dari suatu tempat ke tempat lainnya memanfaatkan konsep graf yang mana memafaatkan algortma *pathfinding* dalam proses penentuannya.

## V. UCAPAN TERIMA KASIH

Puji syukur penulis panjatkan pada hadirat Allah SWT. Atas rahmat dan kemudahan yang diberikan kepada penulis, sehingga makalah yang berjudul "Penerapan Konsep Graf dalam Penentuan Rute pada Google Maps" dapat diselesaikan. Banyak pihak yang telah membantu dalam penyusunan makalah ini. Penulis menyampaikan terima kasih kepada semua pihak yang telah mendukung studi dan proses pembelajaran dalam mata kuliah IF2120 Matematika Diskrit. Penulis ingin menyampaikan rasa syukur dan terima kasih kepada Bapak Dr. Rinaldi Munir, selaku Dosen Mata Kuliah IF2120 Matematika Diskrit dan teman-teman seperjuangan yang telah memberikan banyak masukan, referensi, dan dukungan.

## VII. LINK GITHUB PERCOBAAN PETA ITB

https://github.com/mochfatchur/Tucil3_13519009.git

## REFERENCES

[1] R.Munir, Slide Graf bag.1(2021), Bandung
[2] R.Munir, Slide Graf bag.2(2021), Bandung
[3] A* Search Algorithm - GeeksforGeeks.
[4] https://github.com/mochfatchur/Tucil3_13519009.git

## PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Desember 2021

Mochammad Fatchur Rochman
13519009