Aplikasi Algoritma A*(A-Star) pada Sistem Navigasi GPS

Bram Musuko Panjaitan – 13517089
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
bram.musuko@gmail.com

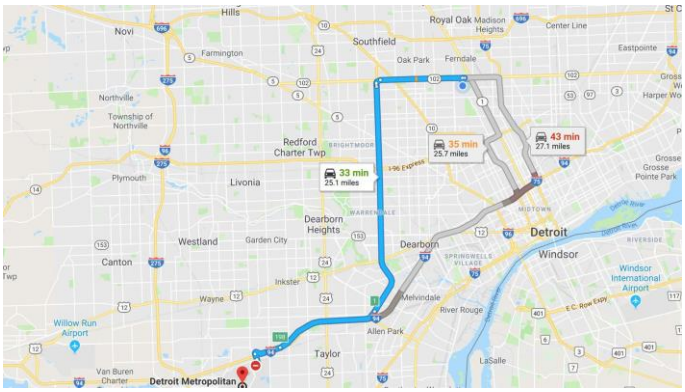
Abstrak* – Era teknologi mempermudah kita untuk berpergian kemana saja. Sekarang kita tidak diharuskan untuk menghafal tiap lokasi yang ingin kita tuju. Cukup dengan menyalakan GPS yang kita miliki maka kita bisa berpergian ke tempat yang kita tuju. Pencarian jalan yang dilakukan oleh GPS menggunakan salah satu prinsip matematika diskrit yaitu pada topik graf. Pada makalah ini kita akan membahas bagaimana mencari jalan terpendek yang dilakukan GPS dengan menggunakan Algoritma A

***Keyword* = Algoritma A*, Graf, *Shortest Path*.**

I. PENDAHULUAN

Pada masa ini teknologi berkembang dengan pesat, dengan perkembangan ini banyak sekali orang yang setiap harinya bergantung dengan teknologi. Tidak lepas juga dengan bidang transportasi. Banyak diantara kita yang menggunakan sistem navigasi GPS (*Global Positioning System*) yang terdapat pada aplikasi seperti *Google Maps* atau *Waze*.

Dengan menggunakan GPS, seseorang tidak perlu lagi menghafal jalur yang akan digunakan untuk mencapai posisi yang ingin dituju, ditambah lagi kita tidak perlu memikirkan segala kemungkinan jalur yang akan digunakan supaya kita mendapatkan jalur paling optimal untuk mencapai tujuan kita. GPS dengan automatis menyediakan jalur terbaik untuk kita sehingga jalur tersebut adalah jalur yang paling optimal pada kondisi saat itu.



Gambar 1 contoh penggunaan GPS

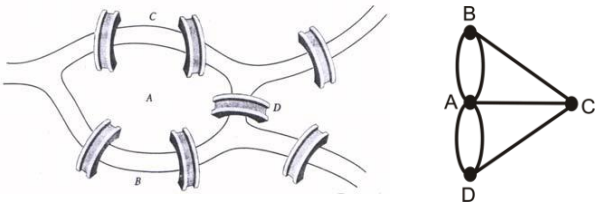
Dalam mencari jarak dari posisi awal ke posisi tujuan banyak sekali algoritma yang dapat kita gunakan seperti algoritma Dijkstra, algoritma *Breadth-first Search*, algoritma *Depth-first Search*, dan lain-lain. Tetapi pada makalah ini kita akan membahas salah satu algoritma yang paling efektif dalam mencari jarak terpendek dari satu titik ke titik lainnya pada sistem navigasi GPS yaitu algoritma A* (A-Star).

Algoritma A*(A-Star) dapat dikatakan salah satu algoritma terbaik dalam mencari rute terpendek dari satu titik ke titik lainnya adalah karena Algoritma A*(A-Star) memiliki informasi tambahan di setiap perhitungannya yaitu fungsi *heuristic* yang memiliki nilai dari jarak antara posisi sekarang dan posisi yang ingin kita capai. Kebanyakan dari algoritma untuk mencari rute terpendek tidak memperhitungkan fungsi *heuristic*, algoritma-algoritma tersebut hanya memperhitungkan jarak dari titik awal ke simpul graf. Hal inilah yang membuat algoritma A*(A-star) spesial.

II. TEORI GRAF

1. Sejarah Graf

Graf pertama kali digunakan pada tahun 1736 untuk menyelesaikan masalah Jembatan Königsberg



Gambar 2 Representasi masalah Jembatan Königsberg dalam Graf

2. Definisi Graf

Graf adalah suatu himpunan beberapa objek dan hubungan antar objek itu sendiri terhadap objek yang lain. Dalam istilah graf ,objek disebut dengan simpul (*vertices*) dan hubungan antar objek disebut dengan sisi (*edges*).
Secara matematis graf dapat dituliskan sebagai berikut, G = (V,E)
V = Himpunan tidak kosong dari simpul (*vertices*).
E = Himpunan sisi yang menghubungkan pasangan simpul (*edges*).

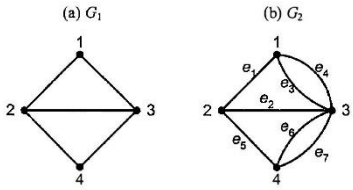
Berdasarkan gelang atau sisi ganda, graf dibagi menjadi dua jenis :

1. Graf Sederhana

Graf Sederhana adalah graf yang tidak mengandung sisi-ganda dan gelang

2. Graf Tak-Sederhana

Graf Tak-sederhana adalah graf yang mengandung sisi-ganda dan gelang



Gambar 3

G1 = Graf Sederhana
G2 = Graf Tak-Sederhana

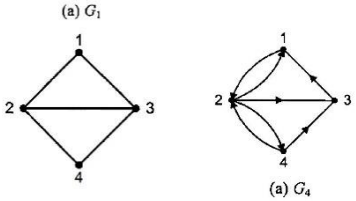
Berdasarkan orientasi arah pada sisi, graf dibagi menjadi dua jenis :

1. Graf Berarah

Graf Berarah adalah graf yang sisinya mempunyai orientasi arah

2. Graf Tak-Berarah

Graf Tak-Berarah adalah graf yang sisinya tidak mempunyai orientasi arah



Gambar 4

G1 = Graf Tak-Berarah

G4 = Graf Berarah

Graf memiliki beberapa terminologi dasar, berikut adalah beberapa terminology dasar graf :

1. Bertetangga (*Adjacent*)

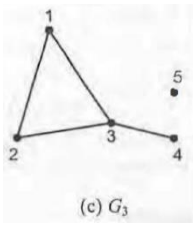
Dua buah simpul dikatakan bertetangga bila keduanya dihubungkan oleh sebuah sisi. Pada gambar 3 kita dapat melihat bahwa simpul 1 bertetangga terhadap simpul 2 dan 3, tetapi simpul 1 tidak bertetangga dengan simpul 4.

2. Bersisian (*Incident*)

Untuk sembarang sisi s = (V_1,V_2), sisi s dapat dikatakan bersisian dengan simpul V_1 dan V_2. Pada gambar 3 kita dapat melihat sisi (1,2) bersisian dengan simpul 1 dan simpul 2, tetapi simpul (1,2) tidak bersisian dengan simpul 3.

3. Simpul Terpencil (*Isolated Vertex*)

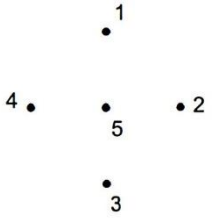
Simpul Terpencil adalah simpul yang tidak bertetangga dengan simpul lainnya dan tidak bersisian dengan simpul lainnya. Pada gambar 5, simpul 5 dapat dikatakan sebagai simpul terpencil



Gambar 5

4. Graf Kosong (*Null Graph atau Empty Graph*)

Graf Kosong adalah graf yang himpunan sisinya adalah himpunan kosong. Pada gambar 6, graf tersebut merupakan graf kosong



Gambar 6

5. Derajat (*Degree*)

Derajat suatu simpul adalah jumlah dari seluruh sisi yang bersisian dengan simpul tersebut. Misalkan v adalah simpul dari graf maka notasi dari derajat v adalah d(v). Pada gambar 3 kita dapat melihat bahwa d(1) = 2, d(2) = 3, d(3) = 3, dan d(4) = 2.

Hasil tersebut sesuai dengan Lemma Jabat Tangan yang mengatakan Jumlah derajat semua simpul pada suatu graf adalah genap, yaitu dua kali jumlah sisi pada graf tersebut.

6. Lintasan (*Path*)

Lintasan yang panjangnya n dari simpul awal V_0 ke simpul tujuan V_n di dalam suatu graf G adalah barisan berselang – seling simpul – simpul dan sisi – sisi yang akan membentuk pola $V_0,E_0,V_1,E_1,...,V_{n-1},E_n,V_n$ sedemikian sehingga $E_1 = (V_0,V_1)$, E2 = (V_1,V_2), ..., E_n = (V_{n-1},V_n) adalah sisi dari graf. Pada gambar 3 salah satu lintasan yang dapat kita ambil adalah $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$.

7. Siklus (*Cycles*) atau Sirkuit (*Circuit*)

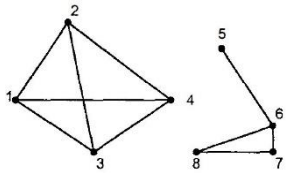
Siklus atau Sirkuit adalah lintasan yang berawal dan berakhir pada simpul yang sama. Salah satu contoh siklus atau sirkuit pada gambar 3 adalah $1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1$.

8. Terhubung (*Connected*)

Misalkan ada dua buah simpul V_0 dan V_1, simpul tersebut dikatakan terhubung jika terdapat lintasan dari V_0 ke V_1.

Graf G disebut graf terhubung (*Connected Graph*) jika pada setiap pasang simpul V_0 danV_1 di dalam himpunan V, maka terdapat lintasan dari V_0 ke V_1. Jika tidak maka graf disebut graf tidak terhubung (*Disconnected Graph*).

Pada gambar 7 kita dapat melihat bahwa graf yang berada di kiri adalah graf terhubung (*Connected Graph*), sedangkan graf yang berada di sisi kanan adalah graf tidak terhubung (*Disconnected Graph*).

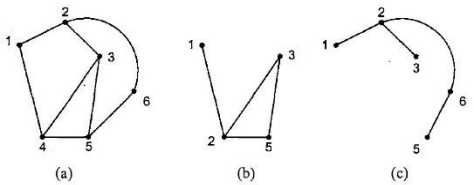


Gambar 7

9. Upagraf (*Subgraph*) dan Komplemen Upagraf

Misalkan G = (*V*,*E*) merupakan suatu graf. G1 = (V_1,E_1) adalah upagraf (*subgraph*) dari G jika $V_1 \subseteq V$ dan $E_1 \subseteq E$.

Komplemen dari upagraf G_1 terhadap graf G adalah G_2 = (V_2,E_2) sedemikian sehingga $E_2 = E - E_1$ dan V_2 adalah himpunan simpul yang anggota – anggota E_2 bersisian dengannya.

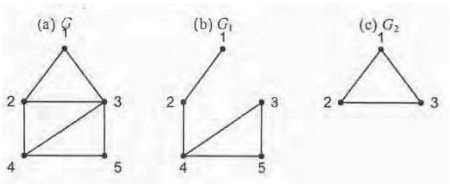


Gambar 8

(a) = Graf G5
(b) = Sebuah upagraf dari graf G5
(c) = Komplemen dari upagraf (b) terhadap Graf G5

10. Upagraf Merentang (*Spanning Subgraph*)

Upagraf G1 = (V_1,E_1) dari G = (V,E) dikatakan upagraf merentang (*spanning subgraph*) jika $V_1 - V$ (yaitu G_1 mengandung semua simpul dari G).

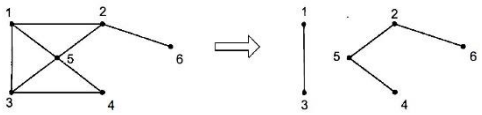


Gambar 9

(a) = Graf G
(b) = Upagraf dari G
(c) = Bukan upagraf dari G

11. *Cut-Set*

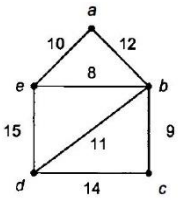
Cut-Set dari graf terhubung G adalah himpunan sisi yang bila dibuang dari G menyebabkan G tidak terhubung. Jadi, *Cut-Set* selalu menghasilkan dua buah komponen terhubung.



Gambar 10 {(1,2),(1,5),(3,5),(3,4)} adalah *Cut-Set*

12. Graf Berbobot (*Weighted Graph*)

Graf Berbobot adalah graf yang setiap sisinya memiliki sebuah bobot (harga). Gambar 11 adalah contoh dari Graf Berbobot



Gambar 11

III. Algoritma A* (A-star)

1. Sejarah Algoritma A*

Algoritma A* pertama kali ditemukan oleh Peter Hart, Nils Nillson, dan Bertram Raphael pada *Shakey Project* yaitu membuat robot yang dapat merencanakan aksi robot itu sendiri. Awalnya mereka ingin menggunakan algoritma *Graph Traverser Algorithm*, tetapi algoritma tersebut hanya menghitung jarak *heurestic function* yaitu jarak dari suatu simpul ke simpul tujuan dan tidak memperhitungkan jarak dari titik awal ke simpul. Oleh sebab itu salah satu diantara mereka untuk menggunakan jumlah keduanya yaitu jarak dari titik awal ke simpul + jarak *heurestic function*.

2. Definisi Algoritma A*

Algoritma A* adalah algoritma yang digunakan pada graf berbobot (*Weighted Graph*) yang secara umum berfungsi untuk mencari jarak terdekat dan menelusuri isi dari graf. Pada setiap pengulangannya Algoritma A* akan menentukan jalur mana yang akan dipilih sebagai jalur untuk mencapai titik akhir. Algoritma A* dapat menentukan jalur yang akan dipilih dengan menggunakan rumus sebagai berikut :

$$f(n) = g(n) + h(n)$$

n adalah simpul yang akan dipilih, g(n) adalah jarak dari simpul n ke simpul awal, dan h(n) adalah jarak absolut dari simpul n ke simpul tujuan.

Cara kerja dari Algoritma A* dapat diibaratkan seperti *Priority Queue* dengan nilai dari tiap antrian adalah {*Vertex*,f(n),*Vertex* sebelumnya}. Algoritma tersebut akan mendahulukan pengerjaan berdasarkan f(n). f(n) yang lebih kecil akan dikerjakan terlebih dahulu daripada f(n) yang memiliki nilai lebih besar. Sesudah memastikan elemen mana yang ingin dikerjakan proses dari pengerjaan pun akan berlangsung. Program akan mengecek semua simpul yang bertetetangga dengan simpul yang sedang di kerjakan. Setelah itu program akan memasukkan semua data dari simpul yang bertetangga ke dalam *priority queue* yang telah ada dengan format yang sama yaitu {*Vertex*,f(n),*Vertex* sebelumnya}.

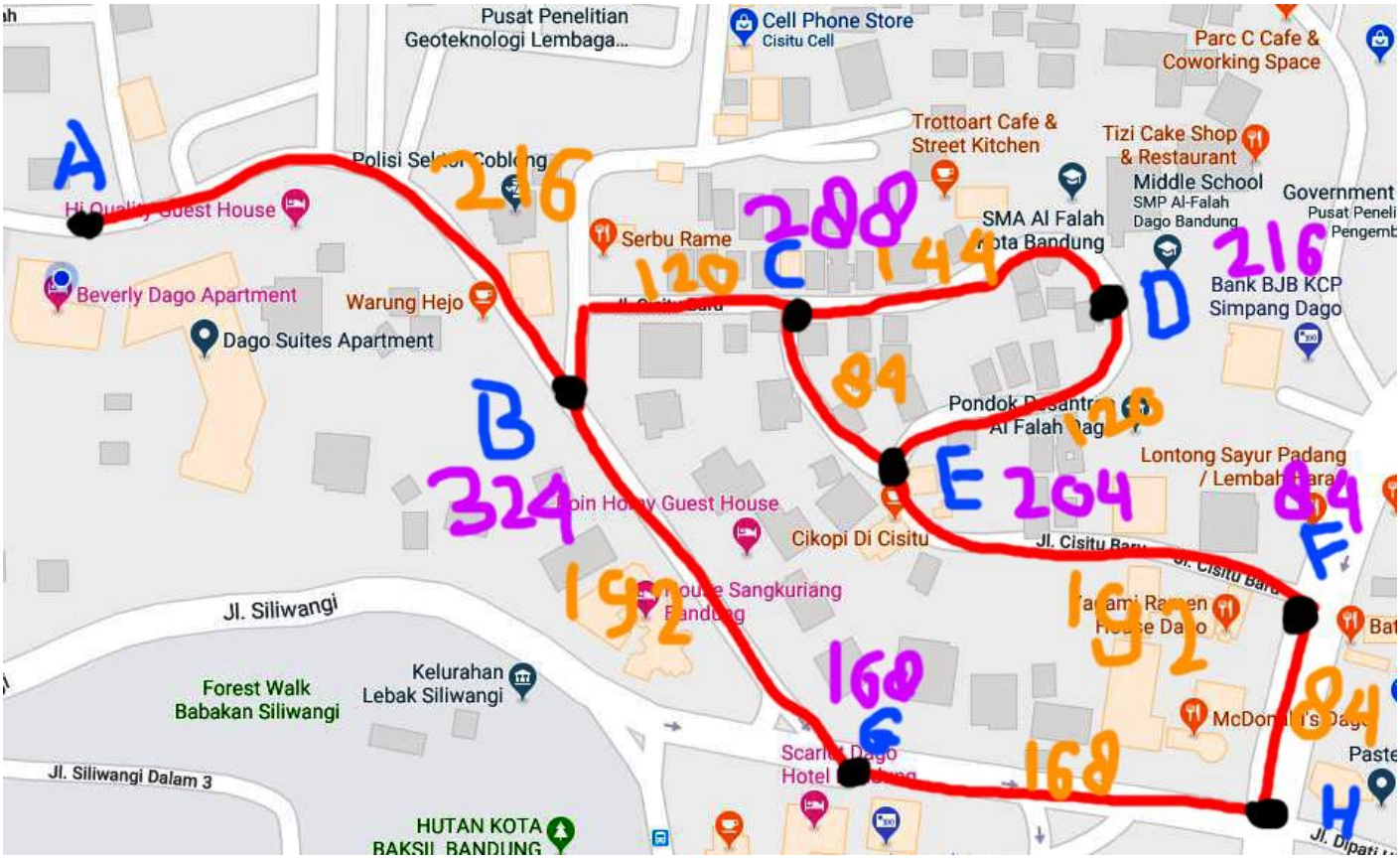
Sesampainya program pada simpul tujuan maka program akan melakukan *backtrack* dengan menggunakan informasi dari *Vertex* sebelumnya yang terdapat pada bagian dari elemen *priority queue*, program akan melakukan backtrack hingga mencapai *Vertex* awal, tujuan dari *backtrack* ini adalah untuk mendapatkan lintasan paling optimal yang dapat dicapai dengan menggunakan Algoritma A*.

Algoritma A* memiliki kompleksitas waktu sebesar $O(b^m)$ dan kompleksitas ruang sebesar $O(b^m)$.

3. Pseudocode Algoritma A*

```
function reconstruct_path(cameFrom, current)
    total_path := {current}
    while current in cameFrom.Keys:
        current := cameFrom[current]
        total_path.append(current)
    return total_path

function A_Star(start, goal)
    // The set of nodes already evaluated
    closedSet := {}
    // The set of currently discovered nodes that are not
     evaluated yet. Initially, only the start node is known.
    openSet := {start}
    // For each node, which node it can most efficiently be
     reached from. If a node can be reached from many nodes,
     cameFrom will eventually contain the most efficient
     previous step.
    cameFrom := an empty map
    // For each node, the cost of getting from the start node to
     that node.
    gScore := map with default value of Infinity
```



Gambar 12

```
// The cost of going from start to start is zero.
   gScore[start] := 0

fScore := map with default value of Infinity
   // For the first node, that value is completely heuristic.
   fScore[start] := heuristic_cost_estimate(start, goal)
   while openSet is not empty

current := the node in openSet having the lowest fScore[] value
      if current = goal
         return reconstruct_path(cameFrom, current)
      openSet.Remove(current)
      closedSet.Add(current)
for each neighbor of current
         if neighbor in closedSet
            continue
            // Ignore the neighbor which is already evaluated.
         // The distance from start to a neighbor
         tentative_gScore := gScore[current] +
            dist_between(current, neighbor)
         if neighbor not in openSet    // Discover a new node
            openSet.Add(neighbor)
         else if tentative_gScore >= gScore[neighbor]
            continue;
 // This path is the best until now. Record it!
   // For each node, the total cost of getting from the start node
    to the goal by passing by that node. That value is partly
    known, partly heuris
      cameFrom[neighbor] := current
      gScore[neighbor] := tentative_gScore
      fScore[neighbor] := gScore[neighbor] +
         heuristic_cost_estimate(neighbor, goal)
```

Pseudocode tersebut diambil dari
https://nb4799.neu.edu/wordpress/?p=2589

IV. Aplikasi Algoritma A*(A-Star) pada Sistem Navigasi GPS

Pada bagian ini saya akan memberikan contoh pengaplikasian algoritma A* pada sistem navigasi GPS. Contoh yang saya berikan adalah bagaimana sistem navigasi GPS dapat menemukan jarak terdekat dari lokasi Beverly Dago Apartment ke McDonald's Simpang dago. Untuk sistem navigasi GPS saya akan menggunakan *Google Maps*.

Pada graf yang terdapat pada gambar 12, warna merah melambangkan sisi dari graf, warna hitam melambangkan simpul dari graf, warna biru melambangkan label dari simpul, warna oren melambangkan jarak antar kedua simpul, warna ungu melambangkan jarak absolut dari simpul ke simpul tujuan. Perlu diingat bahwa panjang yang ditampilkan pada gambar 12

adalah hasil pengukuran dari skala 1 : 2400. Saya menghitung skala antara peta dan jarak asli adalah dengan mencari jarak sebenarnya dari *google maps* dan mencari jarak di peta menggunakan penggaris. Sehingga total jarak yang didapat tidak sama dengan jarak aslinya, tetapi mendekati jarak aslinya.

Untuk memulai proses perhitungan, saya akan menetapkan simpul A sebagai simpul awal dan simpul H akan menjadi simpul tujuan. Untuk mempermudah penjelasan kita akan menggunakan *priority queue* dalam proses penghitungan

dimana elemen dengan f(n) paling rendah akan menjadi elemen pertama.

Formula yang akan digunakan :

$$f(n) = g(n) + h(n)$$

dengan ketentuan :

n = Simpul yang sedang ditinjau

g(n) = Jarak dari Simpul awal ke simpul yang sedang ditinjau

h(n) = Jarak absolut dari simpul ke simpul akhir

Format dari elemen *queue* yang kita gunakan adalah {V,F,Vs}, dimana V adalah simpul yang sedang dikunjungi, F adalah penjumlahan dari jarak simpul awal ke simpul sekarang dan *heurestic function*, dan Vs adalah simpul yang kita gunakan untuk mengunjungi V.

Isi *queue* awal = {}

Kita memulai dengan simpul A. Simpul A bertetangga dengan simpul B.

$$f(B) = g(B) + h(B) = 216 + 324 = 540$$

Dengan memasukkan hasil perhitungan kita, maka isi *queue* saat ini = {{B,540,A}}

Simpul B bertetangga dengan simpul C dan simpul G.

$$f(C) = g(C) + h(C) = (216+120) + 288 = 624$$
$$f(G) = g(G) + h(G) = (216+192) + 168 = 576$$

Dengan menghapus elemen pertama dan memasukkan hasil perhitungan maka isi *queue* saat ini = {{G,576,B},{C,624,B}}

Simpul G bertetangga dengan simpul H.

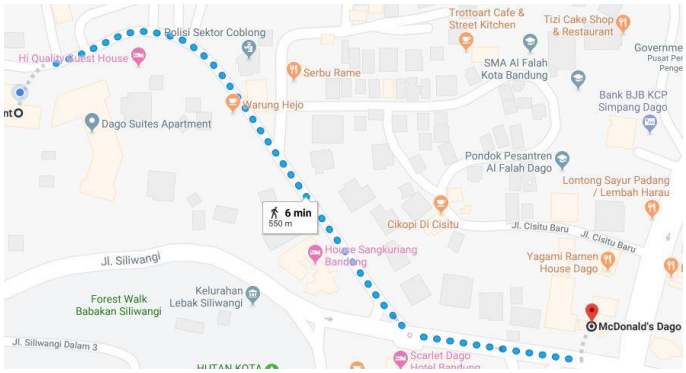
$$f(H) = g(H) + h(H) = (216+192+168) + 0 = 576$$

Dengan menghapus elemen pertama dan memasukkan hasil perhitungan maka isi *queue* saat ini adalah = {{H,576,G},{C,624,B}}

Karena kita telah mencapai simpul tujuan kita, maka proses pencarian telah selesai. Proses pencarian akan berhenti hanya kepada dua kondisi, kondisi pertama adalah ketika program telah menemukan simpul yang dituju seperti contoh kita diatas, yang kedua adalah ketika program telah menelusuri semua kemungkinan dan simpul tidak ada pada graf yang disajikan, maka program juga akan berhenti.

Proses selanjutnya adalah menetapkan lintasan dengan proses *Backtracking*. Pertama – tama kita mencari Vs dari elemen {H,576,G}, yaitu G, maka jalur saat ini H ← G . kemudian kita mencari Vs dari elemen {{G,576,B}, yaitu B, maka jalur saat ini adalah H ← G ← B, dan yang terakhir kita mencari Vs dari elemen {{B,540,A}, yaitu A, maka jalur yang kita dapatkan adalah H ← G ← B ← A. Dengan menyusun ulang posisi jalur maka kita akan mendapatkan A → B → G → H.

Total jarak yang kita tempuh sebanyak 574 m, sedangkan hasil yang di dapatkan di *google maps* adalah 550 m perbedaan ini disebabkan oleh perhitungan jarak yang saya lakukan tidak terlalu akurat karena saya menggunakan penggaris dalam mengukur jarak antar simpul pada peta. Hasil yang kita peroleh ini sesuai dengan hasil dari sistem navigasi *google maps*, seperti gambar berikut



Gambar 13

V. KESIMPULAN

Algoritma A* sangat efektif digunakan untuk mencari jarak terdekat ke simpul yang kita tuju. Dengan algoritma ini kita tidak menelusuri simpul – simpul yang tidak optimal, sehingga proses pengerjaan bisa lebih cepat. Pada kondisi nyata bobot dari jarak antar simpul dapat berubah-ubah, perubahan tersebut dapat disebabkan oleh kondisi kemacetan jalan, cuaca, dan kondisi – kondisi lain yang dapat menyebabkan perubahan kelancaran jalan. Oleh karena itu sistem navigasi GPS yang dimiliki oleh *Google Maps*, *Waze*, dan lain lain memiliki algoritma – algoritma tambahan yang kompleks dalam mengukur jarak terdekat.

Saran saya untuk makalah selanjutnya jika ingin membahas topik yang serupa adalah membahas tentang bagaimana Algoritma A* dapat diberi variasi sehingga mendapatkan solusi yang lebih optimal dalam keadaan nyata mencari jarak terpendek.

VI. UCAPAN TERIMA KASIH

Puji dan syukur saya panjatkan kepada Tuhan Yang Maha Kuasa karena atas kasih dan karunia-Nya saya dapat menyelesaikan makalah yang berjudul “Aplikasi Algoritma A*(A-Star) pada Sistem Navigasi GPS”. Ucapan terma kasih juga saya ucapkan kepada dosen saya, Dra. Harlili S., M.Sc. , Dr. Ir. Rinaldi Munir, MT. , dan Drs. Judhi Santoso, M.Sc. yang telah memberi saya ilmu terkait dengan matematika diskrit sehingga saya dapat mengerti topik yang saya bahas. Saya juga mengucapkan terima kasih kepada kedua orang tua saya yang telah mendukung perkuliahan saya di Bandung, dan yang terakhir saya berterima kasih kepada teman – teman saya yang telah mendukung keberlangsungan dari makalah ini.

REFERENSI

[1] http://informatika.stei.itb.ac.id/~rinaldi.munir/ Matdis/2015-2016/Graf%20(2015).pdf. Tanggal akses 7 Desember 2018

[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/ Stmik/2017-2018/A-Star-Best-FS-dan-UCS-(2018).pdf. Tanggal akses 7 Desember 2018

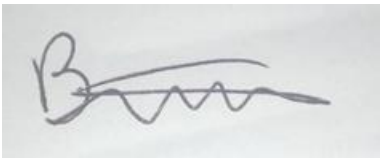
[3] https://www.google.com/maps. Tanggal akses 8 Desember 2018

[4] https://www.geeksforgeeks.org/a-search- algorithm/. Tanggal akses 8 Desember 2018
[5] https://nb4799.neu.edu/wordpress/?p=2589. Tanggal akses 8 Desember 2018
[6] https://www.youtube.com/watch?v=ySN5Wn u88nE. Tanggal akses 7 Desember 2018

PERNYATAAN

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

Bandung, 8 Desember 2018



Bram Musuko Panjaitan / 13517089