Penggunaan Algoritma A* (*A star)* pada Aplikasi *GO-JEK*

Joshua Salimin 13514001
Program Studi Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514001@std.stei.itb.ac.id

***Abstrak*—Makalah ini akan membahas salah satu penggunaan algoritma A* (*A star)* pada salah satu aplikasi yang cukup terkenal diindonesia yaitu *GO-JEK. GO-JEK* adalah sebuah aplikasi yang menyediakan jasa pengantaran baik manusia maupun barang. *GO-JEK* sendiri terbagi menjadi beberapa jasa yaitu *GO-RIDE, GO-CAR, GO-FOOD,* dan lain-lain. Ruang lingkup yang akan dibahas pada makalah ini akan terbatas pada bagaimana dan dimana penggunaan algoritma A* (*A star)* pada aplikasi *GO-JEK*. Dengan memanfaatkan algoritma A* (*A star)* ini aplikasi *GO-JEK* dapat mencari supir terdekat dengan posisi pengguna (*user)* sekarang serta memilih jalan terdekat (*shortest path)* ke tujuan pengguna lalu menghitung jarak dan biaya yang harus dibayar oleh penggguna.**

Kata kunci— Strategi Algoritma, A*, GO-JEK, Shortest Path

I. LATAR BELAKANG

Perkembangan teknologi pada abad ke-21 ini mengalami kemajuan yang sangat pesat. Dengan kemajuan teknologi tersebut, perkembangan internet turut ikut mengalami kemajuan. Dengan mudahnya mengakses internet serta kebutuhan orang akan aplikasi penyedia jasa pengantar menjadi dasar munculnya aplikasi *GO-JEK*. Dengan adanya aplikasi *GO-JEK* ini akan memudahkan bagi pengguna yang tidak memiliki kendaraan pribadi dan sulit untuk mencari kendaraan umum serta penjual yang ingin mengantarkan barang kepada pembeli. Aplikasi *GO-JEK* dapat di unduh dari *playstore (android)* maupun *appstore (ios).*

Setiap *smartphone* pada umumnya sudah dibekali dengan sistem navigasi (*GPS*). Dengan memanfaatkan sistem navigasi tersebut aplikasi dapat menerima semua informasi peta yang ada di Indonesia. Lalu dengan memanfaatkan algoritma A* maka aplikasi dapat melakukan kalkulasi dan perhitungan yang sedemikian rupa hingga mendapatkan jalur terpendek untuk mencapat tempat tujuan pengguna.

Judul makalah ini dipilih penulis karena ingin memberitahukan kepada publik bagaimana cara kerja aplikasi *GO-JEK* dalam mencari supir terdekat, pencarian jalan terpendek (*shortest path)* menuju tempat tujuan pengguna, dan menghitung biaya berdasarkan perkiraan jarak ke tempat tujuan.

II. LANDASAN TEORI

2.1 Teori Graf[1]

2.1.1 Definisi Graf

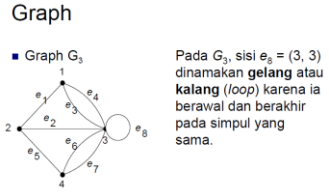
Misalkan ada graf G= (V, E)
V merupakan himpunan simpul-simpul.
E merupakan himpunan sisi-sisi.
V tidak boleh kosong sedangkan E boleh kosong.

Sisi ganda dalam graf adalah dua sisi atau lebih yang menghubungkan dua simpul yang sama.
Gelang atau kalang adalah sisi yang mulai dan berakhir pada simpul yang sama.

2.1.2 Jenis-Jenis Graf

Berdasarkan ada tidaknya gelang atau sisi ganda:

1. Graf sederhana
   Graf yang tidak mengandung gelang dan sisi-ganda (tidak memiliki keduanya).
2. Graf tak-sederhana
   Graf yang mengandung sisi ganda atau gelang (salah satu atau keduanya).

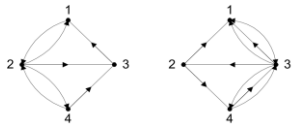


Gambar Graf tak-sederhana diambil dari darkrabbitblog.blogspot.com

Berdasarkan ada tidaknya arah pada sisi:

1. Graf tak-berarah
   Graf yang sisi-sisinya tidak mempunyai arah.
2. Graf berarah

Graf yang sisi-sisinya mempunyai arah.



Gbr (a) graf berarah, (b) graf-ganda berarah

Gambar Graf berarah diambil dari soalampuhh.blogspot.com

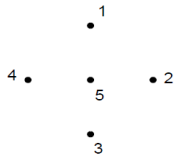
2.1.3 Terminologi Graf

1. Ketetanggan (*Adjacent*)
Jika ada sisi yang menghubungkan dua buah simpul maka dua bua simpul tersebut merupakan tetangga.

2. Bersisian (*Incidency*)
Jika ada sisi yang menghubungkan dua buah simpul maka sisi tersebut bersisian dengan dua buah simpul tersebut.

3. Simpul Terpencil (*Isolated Vertex*)
Simpul yang tidak memiliki satu pun sisi yang bersisian dengannya.

4. Graf Kosong (*Null Graph* atau *Empty Graph*)
Graf yang himpunan sisi-sisinya (E) kosong.



Gambar graf kosong diambil dari rabbitjeyek.blogspot.com

5. Derajat (*Degree*)
Jumlah sisi yang bersisian dengan simpul tersebut. Notasinya ialah d(v). Terdapat suatu lemma jabat tangan dimana jumlah seluruh derajat semua simpul akan genap dimana jumlahnya ialah dua kali jumlah sisi graf tersebut. Sehingga akan mengakibatkan untuk sembarang graf maka jumlah simpul berderajat ganjil akan selalu genap (teorema).

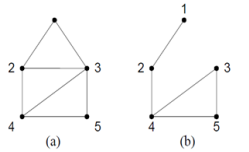
6. Lintasan (*Path*)
Suatu lintasan pada graf ialah barisan yang berisi nama dari simpul asal kemudian simpul berikutnya dan berikutnya hingga sampai di simpul tujuan. Panjang lintasan ialah jumlah sisi dalam lintasan tersebut.

7. Siklus (*Cycle*) atau Sirkuit (*Circuit*)
Lintasan yang dimulai dan diakhiri dengan simpul yang sama.

8. Terhubung (*Connected*)
Jika dua buah simpul terdapat lintasan maka mereka terhubung. Jika graf terhubung maka untuk setiap simpul dalam himpunan V terdapat lintasannya. Jika tidak seluruhnya maka graf tersebut merupakan graf tak-terhubung.

9. Upagraf (*Subgraph*) dan Komplemen Upagraf
Jika ada graf G = (V, E) dan ada graf $G_1 = (V_1, E_1)$ G_1 dikatakan upagraf jika V_1 merupakan himpunan bagian dari V dan E_1 merupakan himpunan bagian dari E. Komplemennya ialah suatu G_2 dimana $E_2 = E - E_1$ dan V_2 adalah himpunan simpul yang anggota anggota E_2 –bersisian dengannya.

10. Upagraf Merentang (*Spanning Subgraph*)
Jika ada graf G = (V, E) dan ada graf $G_1 = (V_1, E_1)$ G_1 dika takan upagraf merentang jika $V_1 = V$.

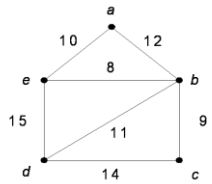


a) Graf *G*
b) Upagraf merentang dari *G*

Gambar diambil dari adytia18.blogspot.com

11. *Cut-Set*
Terdapat suatu graf terhubung G, *cut-set* merupakan himpunan sisi yang jika dibuang dari G menyebabkan G menjadi tidak terhubung dan akan menyebabkan dua buah komponen.

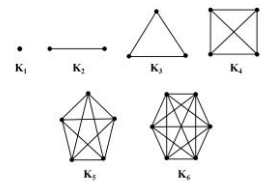
12. Graf Berbobot (*Weighted Graph*)
Graf yang sisinya ada nilai atau bobotnya.



Gambar berbobot diambil dari sha-essa.blogspot.com

2.1.4 Graf Khusus

a.Graf Lengkap (Complete Graph)
Graf sederhana yang setiap simpulnya mempunyai sisi ke sisa simpul lain. Jumlah sisi pada graf ini yang terdiri dari n simpul adalah n(n-1)/2.

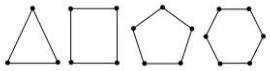


Gambar 12.20 Graf lengkap K_n $1 \le n \le 6$

Graf Lengkap diambil dari slideplayer.info

b Graf Lingkaran

Graf sederhana yang tiap simpulnya mempunyai derajat sebesar dua.



Gambar 12.21 Graf lingkaran C_n $3 \le n \le 6$

Graf lingkaran diambil dari slideplayer.info

c. Graf Teratur (Regular Graph)

Graf yang setiap simpulnya mempunyai derajat yang sama disebut graf teratur. Apabila derajat setiap simpul adalah r, maka graf tersebut disebut sebagai graf teratur derajat r. Jumlah sisi pada graf teratur adalah nr/2.

d. Graf Bipartite (Bipartite Graph)

Graf G yang himpunan simpulnya dapat dipisah menjadi dua himpunan bagian V_1 dan V_2, sedemikian sehingga setiap sisi pada G menghubungkan sebuah simpul di V_1 ke sebuah simpul di V_2 disebut graf bipartit dan dinyatakan sebagai G (V_1, V_2).

2.1.5 Representasi Graf

1. Matriks Ketetangaan
2. Matriks Bersisian
3. Senarai Ketetanggaan (*adjency list)*

2.1.6 Graf Isomorfik

Graf dikatakan isomorfik jika merupakan dua buah graf sama namun berbeda penggambarannya

2.1.7 Graf Planar dan Graf Bidang

Graf Planar adalah graf yang dapat digambarkan tanpa adanya sisi-sisi yang saling memotong sedangkan graf bidang merupakan graf planar yang telah dimodifikasi tanpa adanya sisi yang saling berpotongan.

2.1.8 Teorema Kuratowski

Bertujuan untuk menentukan apakah suatu graf planar atau tidak. Graf G bersifat planar jika dan hanya jika ia tidak mengandung upagraf yang isomorfik dengan salah satu graf Kuratowski atau homeomorfik (homeomorphic) dengan salah satu dari keduanya.

2.1.9 Lintasan dan Sirkuit Euler

Lintasan yang melalui masing-masing sisi dalam graf hanya satu kali secara tepat. Sedangkan sirkuit Euler adalah sirkuit yang melewati masing-masing sisi hanya satu kali saja secara tepat.

2.1.10 Lintasan dan Sirkuit Hamilton

Lintasan yang melalui masing-masing simpul dalam graf hanya satu kali secara tepat. Sedangkan sirkuit Euler adalaha sirkuit yang melewati masing-masing simpul hanya satu kali saja secara tepat kecuali simpul asal.

2.2 Algoritma A* (*A star*)[2][3]

2.2.1 Sejarah

Algoritma A* pertama kali ditemukan pada tahun 1968 oleh Peter Hart, Nils Nilsson, dan Bertram Raphael. Pada mulanya *AI researcher* Nils Nilsson mencoba melakukan perbaikan kinerja dari perencanaan jalan yang dilakukan *Shakey the Robot* sebuah *prototype robot* yang melakukan navigasi sebuah ruangan yang berisi banyak rintangan. Pada mulanya penamaan algoritma ini ialah A1 lalu menjadi A2, dan menjadi A*.

2.2.2 Definisi

Di dalam ilmu komputer (*computer science)* atau informatika (*informatics*) adalah sebuah algoritma yang terkenal untuk menemukan jalan (*pathfinding*) secara efisien dari simpul awal hingga ke simpul tujuan pada graf. Algoritma A* menggunakan pendekatan heuristik yang memperkirakan jarak langsung sebuah simpul ke simpul tujuan.

2.2.3 Heuristik

Kegunaan heuristik ialah untuk menghindari ekspansi simpul atau *path* yang mahal. Fungsi evaluasi (*f(n)*) adalah sebagai berikut :

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

f(n) adalah total *cost* atau biaya dari suatu simpul *x* ke simpul tujuan yang akan menjadi *cost* suatu simpul.
g(n) adalah *cost* atau biaya simpul sejauh ini untuk mencapai suatu simpul *x* (jarak suatu simpul ke suatu simpul lainnya).
h(n) adalah perkiraan jarak dari suatu simpul ke simpul tujuan (heuristik).

2.2.4 Syarat Menggunakan Heuristik

Fungsi heuristik tidak dapat selalu digunakan. Hal ini terjadi jika terdapat fungsi heuristik yang lebih besar dari jarak terpendek sesungguhnya. Syarat menggunakan fungsi heuristik dalam notasi matematika :

$h(n) \leq h^*(n)$

$h(n)$ adalah fungsi heuristik
$h^*(n)$ adalah jarak terpendek sesungguhnya.

2.2.5 Properti A*

Kompleksitas waktu A* adalah $O(b^m)$.
Kompleksitas ruang A* adalah $O(b^m)$.
A* pasti akan memberikan solusi optimal jika $h(n) \leq h^*(n)$ selalu terpenuhi.

2.2.6 Algoritma A* [4]

Pseudocode Algoritma A*

Simpul tujuan direpresentasikan sebagai simpul_tujuan dan simpul awal direpresentasikan sebagai simpul_awal.
Terdapat dua buah list yaitu OPEN dan CLOSE.
OPEN adalah list yang mengandung simpul yang sudah dikunjungi namun belum di ekspansi, list ini merupakan antrian yang akan dikerjakan.
CLOSE adalah list yang mengandung simpul yang sudah dikunjungi dan sudah diekspansi.

Taruh simpul_awal ke OPEN list dengan
f (simpul_awal) = *h* (simpul awal) (inisialisasi)
while Open ≠ ***Empty***
Ambil simpul dari OPEN sebagai simpul_sekarang dengan *f* (simpul_sekarang) = *g* (simpul_sekarang) + *h* (simpul_sekarang)
if simpul_sekarang = simpul_tujuan maka kita telah menemukan solusinya **then break**
Ekspansi node_sekarang dan tambahkan simpul_penerus ke dalam OPEN list
for setiap node_penerus dari node_sekarang
simpul_penerus_biaya = *g* (simpul_sekarang) + *w* (simpul_sekarang, simpul_penerus)
if node_penerus ada di OPEN list **then**
if *g* (simpul_penerus) ≤ simpul_penerus_biaya **then continue**
else if simpul_penerus ada di CLOSE list **then**
if *g* (simpul_penerus) ≤ simpul_penerus_biaya **then continue**
pindahkan simpul_penerus dari CLOSE list ke OPEN list
else
tambahkan simpul_penerus ke OPEN list
h (simpul_penerus) menjadi jarak heuristik ke simpul_tujuan
g (simpul_penerus) = simpul_penerus_biaya
buat leluhur dari simpul_penerus menjadi simpul_sekarang
tambahkan simpul_sekarang ke CLOSE list
if simpul_sekarang ≠ simpul_ tujuan **then** tulis pesan kesalahan (OPEN list sudah kosong)

atau dengan kalimat seperti berikut :

1. Masukan simpul_awal ke OPEN list

2. Loop Langkah – langkah di bawah ini :

a. Cari simpul dengan nilai *f*(n) yang paling kecil dalam OPEN list. Simpul ini sekarang menjadi simpul_sekarang.

b. Keluarkan simpul_sekarang dari OPEN list dan masukan ke dalam CLOSE list

c. Untuk setiap tetangga dari simpul_sekarang lakukan hal berikut :

- Jika tidak dapat dilalui atau sudah ada dalam CLOSE list maka diabaikan saja.

- Jika belum ada di OPEN list . Buat simpul_sekarang menjadi *parent* dari simpul_tetangga ini. Simpan nilai *f*, *g* dan *h* dari node ini.

- Jika sudah ada di OPEN list, cek bila simpul_tetangga ini lebih baik, menggunakan nilai *g* sebagai ukuran. Jika lebih baik ganti *parent* dari simpul ini di OPEN list menjadi simpul_sekarang, lalu kalkulasi ulang nilai *f*, *g* dan *h* dari simpul ini.

d. Hentikan loop jika :

- simpul_tujuan telah ditambahkan ke OPEN list, solusi sudah ditemukan.

- Belum menemukan simpul_tujuan dan OPEN list sudah kosong, solusi tidak ditemukan.

3. Simpan rute. Secara *backward*, urut mulai dari simpul_tujuan ke *parent*-nya sampai mencapai simpul_awal sambil menyimpan simpul ke dalam sebuah *array*.

2.3 *GO-JEK* [5]

2.3.1 Sejarah

GO-JEK berdiri pada tahun 2011 oleh Michaelanglo Maron, Brian Cu, dan Nadiem Makarin. Didirikan dengan nama PT. GO-JEK Indonesia.

2.3.2 Penjelasan Aplikasi

GO-JEK adalah sebuah aplikasi berbasis *mobile* yang dapat diunduh dari *playstore (android)* dan *appstore (ios)*. Aplikasi ini awalnya berfokus pada transportasi khususnya melayani angkutan melalui jasa ojek. Namun aplikasi ini telah berkembang dan melayani

jasa ojek, mobil, pesan makanan, antar barang, dan lain-lain. Layanan *GO-JEK* tersedia untuk wilayah Jabodetabek, Bali, Bandung, Surabaya, Makassar, Medan, Palembang, Semarang, Yogyakarta, dan Balikpapan.

2.3.3 Jenis- jenis jasa

2.3.3.1 *GO-RIDE*
Aplikasi menyediakan jasa transportasi angkutan melalui jasa ojek

2.3.3.2 *GO-CAR*
Aplikasi menyediakan jasa transportasi angkutan dengan menggunakan mobil.

2.3.3.3 *GO-FOOD*
Aplikasi menyediakan jasa pemesanan makanan dengan menggunakan jasa ojek.

2.3.3.4 *GO-SEND*
Aplikasi menyediakan jasa pengiriman barang dengan menggunakan jasa ojek.

2.3.3.5 *GO-MART*
Aplikasi menyediakan jasa pemesanan barang belanjaan di *supermarket* dan semacamnya dengan menggunakan jasa ojek.

2.3.3.6 *GO-BOX*
Aplikasi menyediakan jasa angkut barang dengan menggunakan mobil *box.*

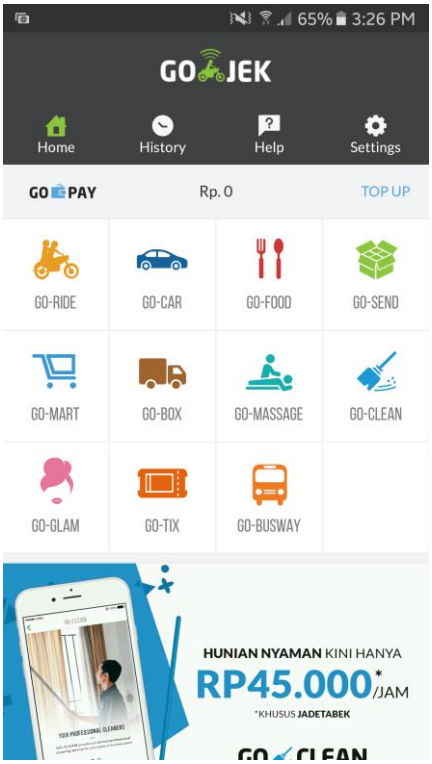
2.3.3.7 *GO-MASSAGE*
Aplikasi menyediakan jasa pijat.

2.3.3.8 *GO-CLEAN*
Aplikasi menyediakan jasa bersih- bersih untuk rumah atau apartemen pengguna.

2.3.3.9 *GO-GLAM*
Aplikasi menyediakan jasa kecantikan bagi para wanita.

2.3.3.10 *GO-TIX*
Aplikasi menyediakan jasa pembelian tiket bioskop dengan menggunakan jasa ojek.

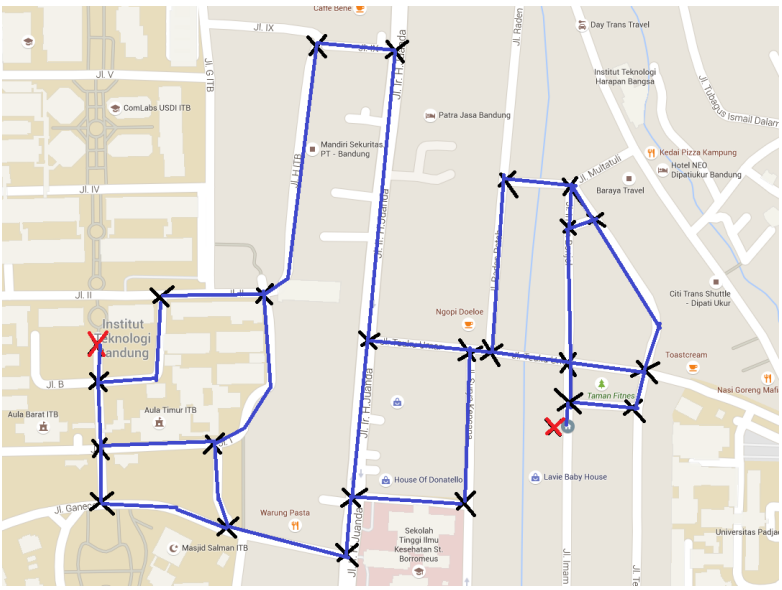
2.3.3.11 *GO-BUSWAY*
Aplikasi menyediakan jasa transportasi hingga menuju halte *busway transjakarta* (hanya khusus DKI Jakarta).



Gambar merupakan koleksi penulis

III. Analisis Algoritma A* pada Aplikasi Go-Jek

Aplikasi *GO-JEK* menggunakan *Google Maps API* untuk menampilkan peta di dalam aplikasi *GO-JEK*.



Gambar merupakan koleksi penulis

Langkah- langkah penerapan algoritma A* :

- Anggap tempat awal dan tujuan sebagai simpul awal dan tujuan.

- Anggap setiap setiap persimpangan sebagai sebuah simpul lainnya.
- Anggap panjang jalan sebagai bobot atau jarak tiap simpul.

Simpul awal pada gambar diatas dimulai dari kos penulis yang terletak di jalan imam bonjol (simbol rumah) dengan tanda X yang berawarna merah. Simpul tujuan adalah Institut Teknologi Bandung.

Fungsi Heuristik setiap simpul adalah jarak antara suatu simpul X dengan simpul tujuan berdasarkan *latitude* dan *longitude* yang didapat dari http://www.gps-coordinates.net/

Pada gambar diatas :
Institut Teknologi Bandung memiliki *latitude* -6.89148 (*decimal degrees*) atau 6º53'29.328" (*degrees, minutes, secondes)* dan *longitude* 107.61065910000002 (*decimal degrees*) atau 107º36'38.372" (*degrees, minutes, secondes).*
Jalan Imam Bonjol memiliki *latitude* -6.892402 (*decimal degrees*) atau 6º53'32.647" (*degrees, minutes, secondes)* dan *longitude* 107.6153501 (*decimal degrees*) atau 107º36'55.26" (*degrees, minutes, secondes).*

Dengan menggunakan rumus jarak antar koordinat (*degrees, minutes, secondes)* [6]
1' = 1.825 m dan 1'' = 30.416 m
Maka didapatkan
Selisih *latitude* antara Institut Teknologi Bandung dengan Jalan Imam Bonjol adalah 3.319"
Lalu dikalikan dengan 30.416m menjadi 100.950704 m
Selisih *longitude* antara Institut Teknologi Bandung dengan Jalan Imam Bonjol adalah 16.888"
Lalu dikalikan dengan 30.146m menjadi 509.105648m
Gunakan rumus jarak
$\sqrt{longitude^2 + latitude^2}$
$\sqrt{(100.950704\ m)^2 + (509.105648\ m)^2} = 519.017924$ m
Menjadi heuristik simpul awal.

Atau dengan menggunakan fungsi berikut untuk *latitude* dan *longitude* dengan *decimal degrees.*

```
function double getDistanceBetween(double latitude1, double
longitude1, double latitude2, double longitude2, string s) {
        double theta = longitude1 - longitude2;
        double distance = (sin(deg2rad(latitude1)) *
sin(deg2rad(latitude2)))  + (cos(deg2rad(latitude1)) *
cos(deg2rad(latitude2)) * cos(deg2rad(theta)));
        double distance = acos(distance);
        distance = rad2deg(distance);
        distance = distance * 60 * 1.1515;
        if s == "km"
                distance = distance * 1.609344;
        return distance;
}
```

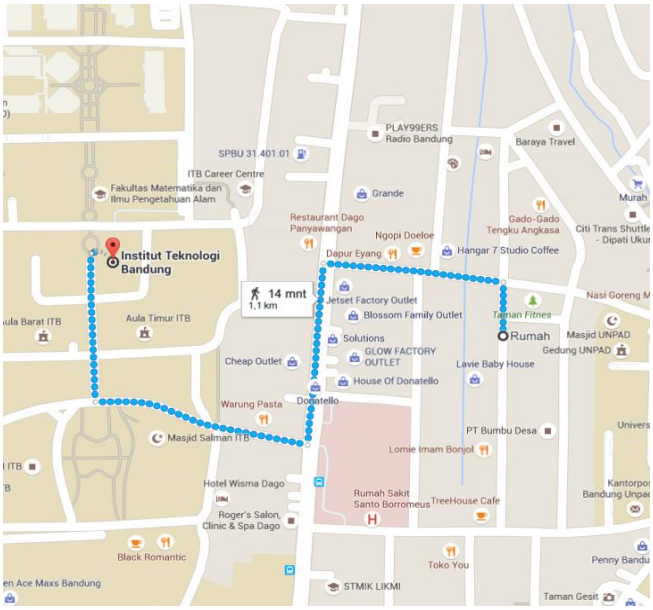
Lakukan langkah- langkah diatas untuk mendapatkan seluruh fungsi heuristik setiap simpul X lainnya.

Setelah mendapatkan seluruh fungsi heuristiknya jalankan algoritma A*.

Simpul tujuan direpresentasikan sebagai simpul_tujuan dan simpul awal direpresentasikan sebagai simpul_awal.
Jarak setiap simpul (bobot) merupakan panjang jalan dari suatu simpul X ke simpul X lainnya.
Terdapat dua buah list yaitu OPEN dan CLOSE.
OPEN adalah list yang mengandung simpul yang sudah dikunjungi namun belum di ekspansi, list ini merupakan antrian yang akan dikerjakan.
CLOSE adalah list yang mengandung simpul yang sudah dikunjungi dan sudah diekspansi.

Taruh simpul_awal ke OPEN list dengan
f (simpul_awal) = *h* (simpul awal) (inisialisasi)
while Open ≠ ***Empty***
Ambil simpul dari OPEN sebagai simpul_sekarang dengan *f* (simpul_sekarang) = *g* (simpul_sekarang) + *h* (simpul_sekarang)
if simpul_sekarang = simpul_tujuan maka kita telah menemukan solusinya **then break**
Ekspansi node_sekarang dan tambahkan simpul_penerus ke dalam OPEN list
for setiap node_penerus dari node_sekarang
simpul_penerus_biaya = *g* (simpul_sekarang) + *w* (simpul_sekarang, simpul_penerus)
if node_penerus ada di OPEN list **then**
if *g* (simpul_penerus) ≤ simpul_penerus_biaya **then continue**
else if simpul_penerus ada di CLOSE list **then**
if *g* (simpul_penerus) ≤ simpul_penerus_biaya **then continue**
pindahkan simpul_penerus dari CLOSE list ke OPEN list
else
tambahkan simpul_penerus ke OPEN list
h (simpul_penerus) menjadi jarak heuristik ke simpul_tujuan
g (simpul_penerus) = simpul_penerus_biaya
buat leluhur dari simpul_penerus menjadi simpul_sekarang
tambahkan simpul_sekarang ke CLOSE list
if simpul_sekarang ≠ simpul_ tujuan **then** tulis pesan kesalahan (OPEN list sudah kosong)

didapatkan hasil seperti berikut



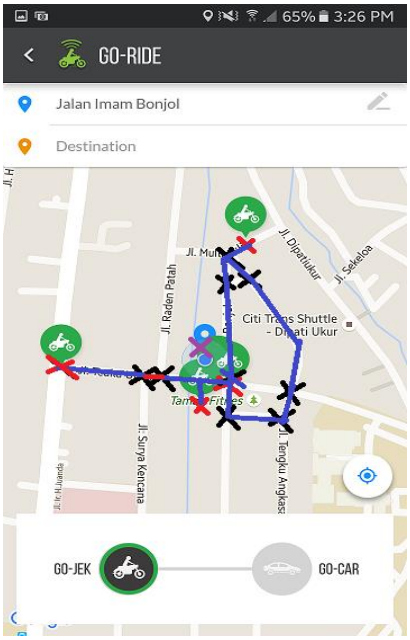
Gambar merupakan koleksi penulis

Terlihat bahwa dengan menggunakan fugnsi heuristik yang penulis jelaskan tidak melanggar aturan $h(n) < h^*(n)$ karena jarak terkecil sesungguhnya dari jalan Imam Bonjol ke Institut Teknologi Bandung ialah 1.100 m sedangkan heuristiknya hanya 519 m.

Dengan demikian telah didapatkan jalur terpendek dari jalan Imam Bonjol ke Institut Teknologi Bandung.

[7]Biaya yang diperlukan untuk membayar *GO-JEK* setelah didapatkan jalur terpendek ialah jika dibawah 15 km maka dikenakan biaya sebesar Rp. 15.000 sedangkan jika diatas 15 km maka dikenakan biaya Rp. 2.000/km.

Pencarian Supir *GO-JEK* terdekat juga menggunakan Algoritma A* yang telah dijelaskan seperti diatas.



Gambar merupakan koleksi penulis

Namun dengan menggangap setiap supir sebagai simpul X yang berwarna merah serta setiap persimpangan sebagai simpul X yang berwarna hitam. Posisi pengguna (*user*) sebagai simpul X berwarna ungu.
Simpul tujuan merupakan supir dan simpul awal adalah posisi pengguna sekarang. Dengan menggunakan algoritma A* seperti yang telah dijelaskan penulis akan mendapatkan jarak beberapa supir dengan posisi pengguna. Lalu memilih supir dengan posisi terdekat terlebih dahulu dan misalkan ia tidak mau maka jasa akan ditawarkan kepada supir yang kedua terdekat dari posisi pengguna.

IV. KESIMPULAN

Dengan mengggunakan algoritma A* (*a star)* untuk penemuan jalan atau *pathfinding* pasti didapatkan solusi yang optimum yaitu jalur terpendek (*shortest path*) asalkan tidak melanggar syarat heuristik yang telah disebutkan diatas. Namun Algoritma A* ini tidak dapat lebih cepat daripada algoritma *Greedy Best- First Search*. Algoritma A* dapat diterapkan dalam menentukan jalur minimum *(shortest path)* pada aplikasi seperti *Google Maps, GO-JEK*, dan aplikasi penyedia transportasi lainnya.

V. UCAPAN TERIMA KASIH

Pertama-tama penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa oleh karena anugerah-Nya penulis dapat menyelesaikan semua tulisan ini. Penulis ingin berterima kasih kepada dosen Strategi Algoritma IF2211 yaitu Pak Rinaldi Munir dan Ibu Nur Ulfa Maulidevi. Serta penulis juga mengucapakan terima kasih yang tidak terhingga kepada semua teman-teman seperjuangan yang membantu penulis untuk menyelesaikan tulisan ini. Penulis pun tidak lupa mengucapkan terima kasih kepada semua pembaca tulisan ini dan semoga tulisan ini dapat bermanfaat bagi para pembacanya.

REFERENSI

[1] informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).ppt 6 Mei 2016 pada jam 11:28
[2] informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/A-Star-Best-FS-dan-UCS-(2016).pptx 6 Mei 2016 pada jam 11:42
[3] http://grobvater.blogspot.co.id/2011/03/pathfinding-algoritma-pencarian-rute.html 6 Mei 2016 pada jam 11:45
[4] http://mat.uab.cat/~alseda/MasterOpt/AStar-Algorithm.pdf 6 Mei 2016 pada jam 12:54
[5] http://labanapost.com/2015/07/website/investor-go-jek-dan-sejarah-para-pendirinya/ 6 Mei 2016 pada jam 13:30
[6] https://www.scribd.com/doc/76276046/Menghitung-Jarak-Antar-Koordinat 6 Mei 2016 pada jam 14:43
[7] http://www.ojekganteng.com/2016/01/tarif-gojek-2016-dan-cara-melihat-tarif-gojek.html 6 Mei 2016 pada jam 15:19

V. 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, 6 Mei 2016

Joshua Salimin
13514001