Penerapan Algoritma A* dalam Penentuan Rute Perjalanan untuk Menghindari Kemacetan

Siti Iedrania Azzariyat Akbar (*13519137*)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519137@std.stei.itb.ac.id

***Abstrak*—Pencarian rute dapat dilakukan dengan menggunakan berbagai macam algoritma, termasuk di antaranya Breadth-First Search (BFS), Depth-First Search (DFS), Greedy Best First Search, dan A* Search. Beberapa algoritma, termasuk A* Search, menggunakan heuristik dalam pencariannya, atau dapat disebut sebagai informed search. Ada beberapa hal yang dapat diperhitungkan secara heuristik dalam pencarian rute perjalanan, seperti jarak secara garis lurus antara dua titik simpul, banyaknya ruas jalan antara simpul akar dan simpul tujuan, sampai tingkat kemacetan suatu jalan. Dalam makalah ini, akan dibahas mengenai pencarian rute menggunakan algoritma A* dengan mempertimbangkan tingkat kemacetan sebagai data heuristik yang tersedia.**

Kata kunci—Algoritma A*; rute; kemacetan

I. PENDAHULUAN

Dalam menentukan rute perjalanan, ada banyak hal yang dapat diperhitungkan seseorang untuk mendapatkan rute yang paling sesuai dengan kebutuhan. Beberapa contohnya adalah buka atau tutupnya suatu jalan, jenis kendaraan yang bisa melalui suatu jalan, keamanan suatu jalan, tempat-tempat lainnya yang ingin disinggahi dalam perjalanan, dan tingkat kemacetan suatu jalan. Tidak jarang pengguna kendaraan memilih jalan yang lebih panjang untuk mencapai tempat tujuannya dengan alasan demikian.

Dalam berperjalanan, kemacetan adalah salah satu hal yang dapat memperlambat suatu perjalanan. Dengan rute yang memiliki jarak paling singkat sekalipun, kemacetan dapat menambah waktu yang diperlukan seseorang untuk mencapai tempat tujuannya. Bahkan, dalam beberapa keadaan, kemacetan dapat membahayakan orang yang sedang berkendaraan. Oleh karena itu, tidak jarang dalam proses penentuan rute dalam kehidupan nyata, orang yang berpergian memilih jalur yang sepertinya menambah jarak perjalanan namun memiliki tingkat kemacetan yang rendah, dibandingkan dengan jalan yang memperpendek rute perjalanan namun memiliki tingkat kemacetan yang lebih tinggi.

Persoalan pencarian rute yang menghindari kemacetan dapat dimodelkan dengan beberapa algoritma pencarian rute, salah satunya adalah algoritma A*. Algoritma A* dianggap sebagai salah satu algoritma dengan pencarian heuristik yang optimal. Dengan memilih tingkat kemacetan sebagai pertimbangan fungsi evaluasi heuristik, algoritma ini dapat digunakan untuk mencari rute optimal berdasarkan tingkat kemacetan.

II. DASAR TEORI

A. Kemacetan

Kemacetan didefinisikan sebagai hal (keadaan) macet. Kata "macet" sendiri didefinisikan sebagai "tidak dapat berfungsi dengan baik; sendat; serat".[1] Kemacetan lalu lintas adalah keadaan tersendat hingga terhentinya arus lalu lintas. Umumnya, kondisi ini disebabkan oleh jumlah kendaraan yang melebihi kapasitas yang tersedia di jalan. Kemacetan dapat terjadi di mana saja, dari kota-kota besar sampai kota-kota kecil. Kemacetan dapat disebabkan oleh banyak hal, di antaranya:

1. Jumlah kendaraan yang melebihi kapasitas jalan.
2. Terjadinya kecelakaan lalu lintas.
3. Adanya perbaikan jalan.
4. Adanya bencana alam yang memotong arus lalu lintas.[2]

Selain itu, ada juga penyebab kemacetan yang datangnya dari pengguna jalan itu sendiri. Di antara penyebab tersebut adalah pengguna jalan yang:

1. Berpindah jalur tidak pada tempatnya.
2. Bersikap ragu di jalanan.
3. Berkendara tidak pada jalurnya.
4. Parkir atau menaikkan/menurunkan penumpang sembarangan.

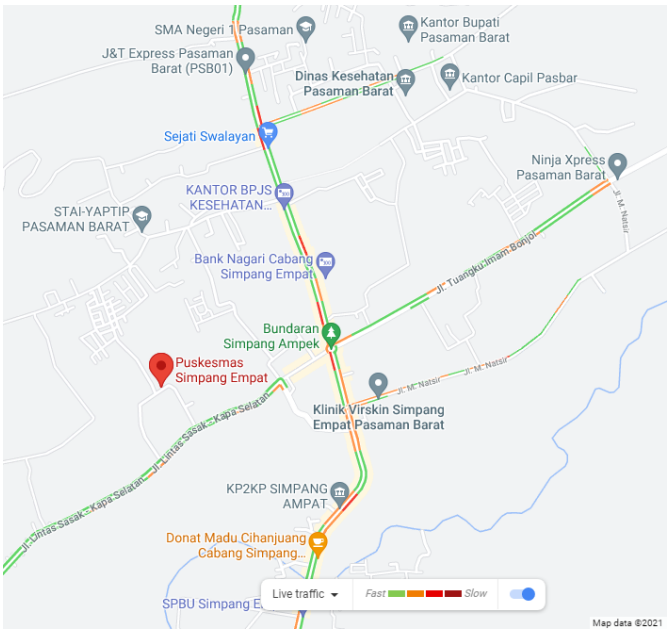
Meskipun ada penyebab yang datangnya dari pengguna jalan itu sendiri, ada banyak dampak negatif yang dapat dirasakan pengguna jalan ketika menghadapi kemacetan. Dampak negatif tersebut di antaranya:

1. Kerugian waktu. Hal ini karena seseorang akan membutuhkan waktu lebih lama di jalan ketika menghadapi kemacetan.
2. Kerugian ekonomi. Hal ini di antaranya karena seseorang biasanya akan menghabiskan BBM yang

lebih banyak dibandingkan dengan berkendara di jalanan yang tidak mengalami kemacetan.

3. Stres. Hal ini biasanya dialami oleh orang yang terburu-buru untuk mencapai tempat tujuan.
4. Kelelahan. Orang yang menghadapi kemacetan akan menggerakkan tubuhnya dengan lebih aktif dibandingkan yang tidak menghadapi kemacetan, terutama anggota tubuh bagian tangan dan kaki.[3]
5. Penyakit. Hal ini karena kebanyakan kendaraan bermotor mengeluarkan zat-zat yang menimbulkan pencemaran udara dan berbahaya bagi kesehatan manusia.
6. Pencemaran lingkungan. Nitrogen oksida dan sulfur dioksida dari knalpot kendaraan turut menyebabkan hujan asam yang mencemari air, membahayakan kehidupan di dalamnya, dan merusak beragam tanaman. Selain itu, kendaraan mengeluarkan karbon dioksida yang turut andil dalam pemanasan global.[4]

B. *Fitur Lalu Lintas di Google Maps*



Gambar 1. Contoh Visualisasi Tingkat Kemacetan di Google Maps[5]

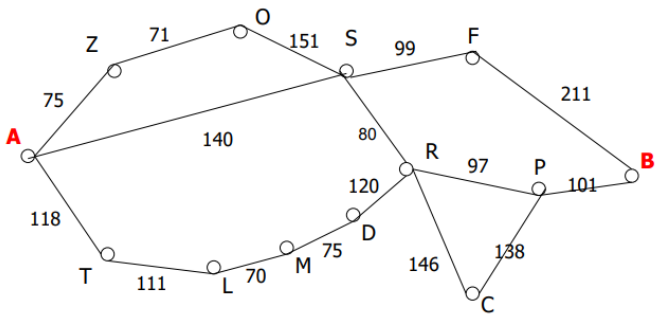
Sebagai salah satu aplikasi navigasi paling populer, Google Maps memiliki fitur yang memungkinkan pengguna untuk mengecek tingkat kemacetan di suatu jalan (Traffic/Lalu Lintas). Bahkan, fitur ini dijadikan salah satu pertimbangan dalam fitur lainnya pada Google Maps, yaitu pencarian rute. Dalam pencarian rutenya, jika menemukan kemacetan pada suatu jalan, Google Maps akan merekomendasikan kita untuk memilih jalur lain agar terhindar dari kemacetan.

Google Maps mengetahui kemacetan berdasarkan ponsel-ponsel penggunanya. Ratusan juta orang di dunia memberikan data *real-time* kepada Google, yang digunakan untuk menganalisis keadaan lalu lintas. Cara kerjanya yaitu semua ponsel yang sedang menggunakan Google Maps dan ponsel berbasis Android yang menghidupkan layanan lokasi mengirimkan data anonim ke Google. Lalu, Google akan menganalisis jumlah mobil dan kecepatan masing-masingnya, dalam waktu kapanpun pada suatu jalan manapun.

Selain itu, Google memiliki data yang mencakup waktu beberapa tahun mengenai keadaan lalu lintas pada jalan-jalan tertentu dan waktu-waktu tertentu. Dengan data itu, Google bisa memperkirakan bagaimana keadaan lalu lintas akan berubah dan berkembang sepanjang perjalanan yang dilalui. Hal ini memungkinkan Google untuk memperingatkan penggunanya jika keadaan lalu lintas lebih baik atau buruk daripada biasanya.[6]

C. *Algoritma Pencarian Rute*

Dalam algoritma pencarian rute, ada dua jenis pencarian yang bisa dilakukan, yaitu *uninformed search* dan *informed search*. Dalam algoritma *uninformed search*, tidak ada informasi tambahan tentang permasalahan yang ada, hanya definisi dari algoritma yang digunakan. *Uninformed search* sering juga disebut *blind search*. Istilah ini menggambarkan bahwa teknik pencarian ini tidak memiliki informasi tambahan di luar yang disediakan definisi masalah.[7]

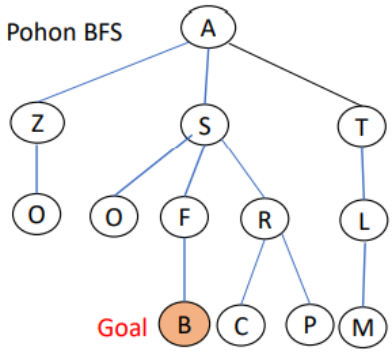


Gambar 2. Contoh Peta untuk Persoalan Pencarian Rute

Beberapa contoh *uninformed search* di antaranya:

1. Breadth-First Search (BFS)

   Pada BFS, dilakukan pencarian secara First In First Out (FIFO), dimana simpul-simpul pada graf diperlakukan seperti sebuah *queue* (antrian). Pada pencarian ini, simpul akar akan diekspansi, setelah itu dilanjutkan oleh semua *successor* (simpul anak/tetangga) dari simpul akar. Untuk setiap simpul yang ditelusuri, simpul-simpul tetangganya akan dimasukkan ke daftar simpul hidup yang akan ditelusuri, pada bagian belakang. Hal ini terus dilakukan berulang-ulang sampai semua simpul sudah diekspansi, yaitu mencapai daun paling bawah, atau ditemukan solusi.



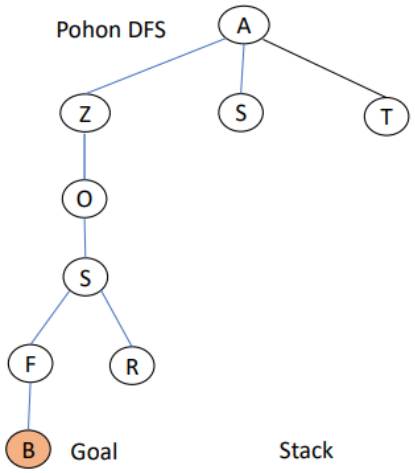
Gambar 3. Contoh Pohon pada Penelusuran Secara BFS

Simpul-E	Simpul Hidup
A	Z_A,S_A,T_A
Z_A	S_A,T_A,O_{AZ}
S_A	T_A,O_{AZ},O_{AS},F_{AS},R_{AS}
T_A	O_{AZ},O_{AS},F_{AS},R_{AS},L_{AT}
O_{AZ}	O_{AS},F_{AS},R_{AS},L_{AT}
O_{AS}	F_{AS},R_{AS},L_{AT}
F_{AS}	R_{AS},L_{AT},B_{ASF}
R_{AS}	L_{AT},B_{ASF},D_{ASR},C_{ASR},P_{ASR}
L_{AT}	B_{ASF},D_{ASR},C_{ASR},P_{ASR},M_{ATL}
B_{ASF}	Solusi ketemu

Gambar 4. Contoh Antrian Simpul Hidup pada Penelusuran Secara BFS

2. Depth-First Search (DFS)

   Pada DFS, kumpulan simpul diperlakukan seperti sebuah *stack*, yaitu secara Last In First Out (LIFO). Teknik yang digunakan pada pencarian ini adalah mengekspansi simpul yang ada sampai mencapai simpul yang paling dalam pada pohon, yaitu simpul yang tidak memiliki *successor*. Untuk setiap simpul yang ditelusuri, simpul-simpul tetangganya akan dimasukkan ke daftar simpul hidup yang akan ditelusuri, pada bagian depan daftar. Setelah simpul selesai diekspansi, yaitu telah ditemukan ujung dari pohon (daun), maka simpul akan ditinggalkan, dan dilanjutkan ke simpul berikutnya yang memiliki *successor* yang belum diekspansi.



Gambar 5. Contoh Pohon pada Penelusuran Secara DFS

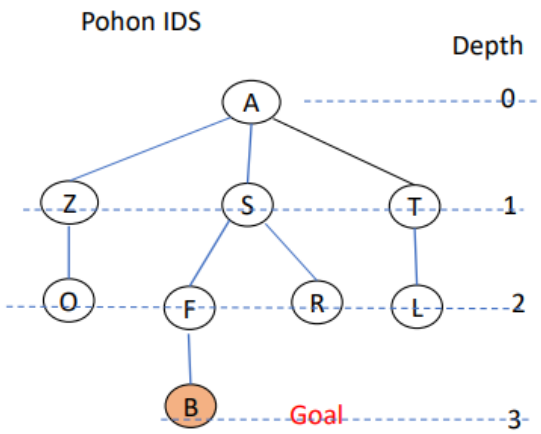
Simpul-E	Simpul Hidup
A	Z_A,S_A,T_A
Z_A	O_{AZ}, S_A,T_A
O_{AZ}	S_{AZO},S_A,T_A
S_{AZO}	F_{AZOS}, R_{AZOS},S_A,T_A
F_{AZOS}	B_{AZOSF}, R_{AZOS},S_A,T_A
B_{AZOSF}	Solusi ketemu

Gambar 6. Contoh Antrian Simpul Hidup pada Penelusuran Secara DFS

3. Depth Limited Search (DLS)

   Pencarian ini sama seperti DFS, namun dengan membatasi kedalaman pada pencarian, yaitu memberi batas pada kedalaman simpul yang akan ditelusuri, sejak awal pencarian, sehingga simpul yang berada pada batas tersebut akan diperlakukan seolah-olah mereka tidak memiliki *successor*. Pencarian ini dibuat untuk mengatasi permasalahan pada DFS, yaitu ketika proses pencarian menemui *infinite state space*. Namun, pencarian ini memiliki kekurangannya sendiri, yaitu adanya kemungkinan hasil tidak dapat ditemukan karena kedalaman simpul hasil tidak pernah tercapai.

4. Iterative Deepening Search (IDS)



Gambar 7. Contoh Pohon pada Penelusuran Secara IDS

IDS merupakan sebuah strategi umum yang biasanya dikombinasikan dengan DFS, yang akan menemukan berapa batas kedalaman terbaik untuk digunakan. Hal ini dilakukan dengan menambah batas kedalaman secara bertahap sampai ditemukan simpul hasil.

5. Uniform Cost Search (UCS)

Simpul-E	Simpul Hidup
A	Z_{A-75}, T_{A-118}, S_{A-140}
Z_{A-75}	T_{A-118}, S_{A-140}, O_{AZ-146}
T_{A-118}	S_{A-140}, O_{AZ-146}, L_{AT-229}
S_{A-140}	O_{AZ-146}, R_{AS-220}, L_{AT-229}, F_{AS-239}, O_{AS-291}
O_{AZ-146}	R_{AS-220}, L_{AT-229}, F_{AS-239}, O_{AS-291}
R_{AS-220}	L_{AT-229}, F_{AS-239}, O_{AS-291}, $P_{ASR-317}$, $D_{ASR-340}$, $C_{ASR-366}$
L_{AT-229}	F_{AS-239}, O_{AS-291}, $M_{ATL-299}$, $P_{ASR-317}$, $D_{ASR-340}$, $C_{ASR-366}$
F_{AS-239}	O_{AS-291}, $M_{ATL-299}$, $P_{ASR-317}$, $D_{ASR-340}$, $C_{ASR-366}$, $B_{ASF-450}$
O_{AS-291}	$M_{ATL-299}$, $P_{ASR-317}$, $D_{ASR-340}$, $C_{ASR-366}$, $B_{ASF-450}$
$M_{ATL-299}$	$P_{ASR-317}$, $D_{ASR-340}$, $D_{ATLM-364}$, $C_{ASR-366}$, $B_{ASF-450}$
$P_{ASR-317}$	$D_{ASR-340}$, $D_{ATLM-364}$, $C_{ASR-366}$, $B_{ASRP-418}$, $C_{ASRP-455}$, $B_{ASF-450}$
$D_{ASR-340}$	$D_{ATLM-364}$, $C_{ASR-366}$, $B_{ASRP-418}$, $C_{ASRP-455}$, $B_{ASF-450}$
$D_{ATLM-364}$	$C_{ASR-366}$, $B_{ASRP-418}$, $C_{ASRP-455}$, $B_{ASF-450}$
$C_{ASR-366}$	$B_{ASRP-418}$, $C_{ASRP-455}$, $B_{ASF-450}$
$B_{ASRP-418}$	Solusi ketemu

Gambar 8. Contoh Antrian Simpul Hidup pada Penelusuran Secara UCS

Selain menjalankan BFS, algoritma ini melakukan ekspansi pada simpul dengan nilai *path* yang paling kecil. Hal ini bisa dilakukan dengan membuat antrian yang berdasarkan pada nilai *path*-nya. Jadi, simpul hidup ditempatkan pada sebuah *priority queue*.

Informed search sering juga disebut dengan *heuristic search* atau pencarian heuristik. Pencarian pada algoritma ini memiliki informasi lain yang spesifik, selain definisi masalahnya itu sendiri.

Pada pencarian heuristik, heuristik yang digunakan mengestimasi nilai dari suatu simpul. Hal ini bisa berupa keberadaan simpul selanjutnya, tingkat kesulitan penyelesaian submasalah, kualitas dari solusi yang ditawarkan sebuah simpul, atau banyaknya informasi yang bisa diperoleh. Dalam menjalankan pencarian ini, dibutuhkan sebuah fungsi evaluasi heuristik yang bergantung pada simpul yang ditelusuri, simpul tujuan, pencarian sejauh ini, dan domain.

Beberapa contoh *informed search* di antaranya:

1. Greedy Best-First Search

   Pada algoritma ini, digunakan sebuah fungsi evaluasi pada setiap simpul. Fungsi f(n), yang setara dengan h(n), mengestimasi *cost* yang dibutuhkan dari suatu simpul n ke simpul tujuan. Sebagai contoh, dalam sebuah pencarian rute terpendek, fungsi ini dapat berupa jarak antara suatu simpul n ke simpul tujuan jika ditarik sebuah garis lurus yang menghubungkan kedua simpul tersebut. Namun, ekspansi yang dilakukan menggunakan evaluasi simpul hanya dengan melihat fungsi heuristiknya. Dengan kata lain, yang dibandingkan untuk penentuan ekspansi simpul hanya nilai estimasi saja, semisal pada contoh di atas, jarak secara garis lurus tersebut.

   Algoritma ini memiliki beberapa kekurangan, di antaranya hasil yang tidak komplit, kecenderungan berpaku pada minimum lokal, dan tidak *reversible*.[8]

2. A* Search

   Pada algoritma A*, pembangkitan simpul-simpul menghindari jalur yang sudah "mahal" atau memiliki *cost* yang besar. Pada algoritma ini digunakan fungsi evaluasi sebagai berikut:

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

   dengan n merupakan simpul yang sedang ditelusuri atau simpul terakhir pada jalur, g(n) merupakan *cost* dari simpul akar menuju simpul n, h(n) merupakan estimasi *cost* dari n menuju simpul tujuan, dan f(n) merupakan estimasi *cost* total yang melalui n dan menuju ke simpul tujuan.

   Algoritma A* adalah algoritma yang komplit, kecuali jika jumlah simpul tak hingga dengan $f \leq f(G)$. Waktu dan ruang memori yang dibutuhkan untuk melakukan pencarian dengan algoritma ini berkembang secara eksponensial. Hasil yang diperoleh adalah hasil yang optimal sesuai dengan heuristik yang digunakan.[9][10]

III. IMPLEMENTASI

Untuk menggunakan algoritma A* pada penentuan rute, pertama-tama, harus dirumuskan fungsi evaluasi heuristik. Rumus fungsi evaluasi pada algoritma A* terdiri dari dua bagian, yaitu g(n) yang menyatakan *cost* dari simpul awal sampai simpul n, dan h(n) yang merupakan estimasi *cost* dari simpul n sampai tujuan.

A. *Nilai g(n)*

Nilai g(n) diisi dengan *cost* sejauh ini (sampai simpul n), yaitu *cost* dari simpul akar sampai ke simpul n. Pada pembahasan ini, *cost* yang dimaksud adalah jarak antar simpul dalam satuan yang diseragamkan. Dalam penggunaannya, jarak ini dapat berupa jarak antar persimpangan jalan atau tempat-tempat pada sebuah peta.

B. *Nilai h(n)*

Nilai h(n) merupakan fungsi heuristik, yaitu perkiraan rute kedepannya yang telah mempertimbangkan tingkat kemacetan pada suatu jalan. Pada penerapan ini, tingkat kemacetan jalan dibuat mengikuti model visualisasi yang disediakan Google Maps. Pada fitur tersebut, kemacetan ditandai dengan skala warna, yaitu hijau, oranye, merah, dan merah gelap secara berurutan berdasarkan tingkat parahnya kemacetan suatu jalanan.

Dalam pembahasan ini, skala warna akan disesuaikan dengan faktor pengali yang angkanya sesuai dengan tingkat kemacetan. Warna hijau menandakan keadaan jalan yang normal tanpa kemacetan, jadi faktor pengali yang bersesuaian adalah 1. Warna oranye menandakan adanya kemacetan ringan, sehingga faktor pengali yang bersesuaian yang dipilih adalah 1.5. Warna merah menandakan adanya kemacetan sedang, dan disesuaikan dengan faktor pengali 2. Sedangkan untuk warna merah gelap, faktor pengali yang diberikan adalah 3.[11]

Pemberian nomor ini diasumsikan telah menyesuaikan dengan waktu tambahan yang diperlukan seseorang untuk berkendara ketika terjebak dalam kemacetan. Ketika pengguna kendaraan terjebak suatu kemacetan, dengan kecepatan yang sama seperti kecepatan dasar yang digunakan pengguna ketika tidak terjebak kemacetan, pengguna dapat menempuh jarak sejauh jarak sesungguhnya dikali dengan faktor pengali yang diberikan.

Pada pembahasan ini, nilai yang dimasukkan ke dalam rumus fungsi evaluasi total adalah nilai terkecil dari jarak sesungguhnya dari suatu jalan dikalikan dengan skala yang menyesuaikan dengan skala warna yang disebut di atas, yang memiliki tingkat kemacetan di atas hijau. Heuristik yang digunakan menganggap rute paling optimal adalah yang paling minim kemacetan, sehingga yang diperhitungkan dalam rumus adalah hanya yang memiliki tingkat kemacetan oranye ke atas.

Maka, h(n) dapat dirumuskan sebagai berikut:

h(n) = min(jarak sesungguhnya x faktor pengali)

dengan batasan h(n) hanya menyertakan jalur yang memiliki tingkat kemacetan di atas hijau, dan dengan nilai faktor pengali:

- 1.5 untuk oranye.
- 2 untuk merah.
- 3 untuk merah gelap.

C. *Penghitungan f(n) dan Pembentukan Jalur Pencarian*

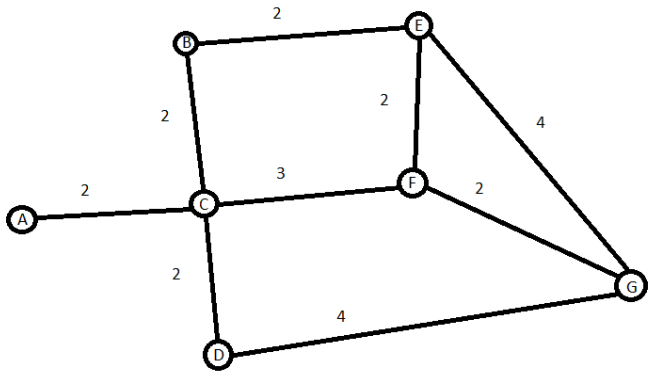
Dalam algoritma A*, nilai fungsi evaluasi total dihitung dengan menggunakan rumus sebagai berikut:

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

yang mengikuti nilai g(n) dan h(n) yang sudah disepakati di atas. Simpul selanjutnya yang akan ditelusuri akan didasarkan pada simpul hidup yang memiliki nilai f(n) terkecil, yang dianggap sebagai jalur dengan jarak terpendek dengan telah memperhitungkan kemacetan yang ada.

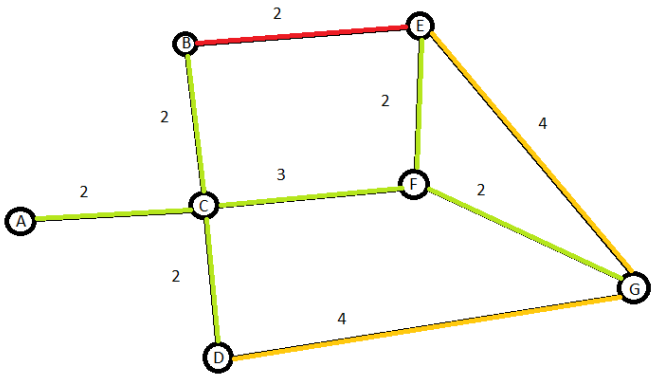
IV. PENGUJIAN

Pengujian akan dilakukan dengan sebuah data fiktif yang menggambarkan jalan-jalan pada peta beserta jarak dan keadaan kemacetannya. Peta fiktif yang digunakan adalah sebagai berikut:



Gambar 9. Peta Fiktif untuk Pengujian

Peta ini akan dilengkapi dengan keadaan kemacetan yang menyesuaikan dengan visualisasi kemacetan pada Google Maps. Lalu, akan dicari rute optimal yang mengikuti algoritma di atas.



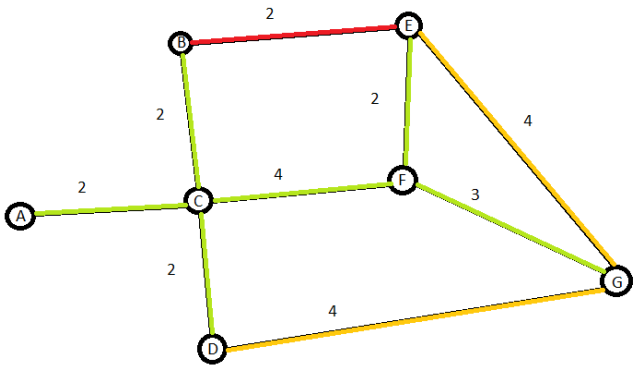
Gambar 10. Peta Fiktif yang Dilengkapi Keadaan Kemacetan

Pada percobaan pertama ini, akan dicari rute tercepat untuk menempuh perjalanan dari simpul C ke simpul G.

TABLE I. HASIL PENGUJIAN RUTE C-G

Iterasi	Simpul-Ekspan	Simpul Hidup	f(n)
1	C	B[C]	2 + 2(2) = 6
		D[C]	2 + 1.5(4) = 8
		F[C]	3 + 0 = 3
2	F[C]	E[F]	3 + 0 = 3
		G[F]	3 + 0 = 3

Dari tabel 1, dapat dilihat bahwa simpul tujuan G berhasil dicapai dengan jalur C-F-G, dengan total jarak 3, tanpa melalui kemacetan apapun. Dengan heuristik yang digunakan, dapat dilihat bahwa jalur B[C] berhasil dihindari meskipun jarak sesungguhnya adalah 2. Rute yang melalui B[C] minimal melalui satu jalan yang macet, yaitu jalan B-E. Artinya, kemacetan yang ada berhasil dihindari.



Gambar 11. Peta Fiktif yang Dilengkapi Keadaan Kemacetan (2)

Pada percobaan kedua ini, akan dicari rute tercepat untuk menempuh perjalanan dari simpul C ke simpul G.

TABLE II. HASIL PENGUJIAN RUTE C-G

Iterasi	Simpul-Ekspan	Simpul Hidup	f(n)
1	C	B[C]	2 + 2(2) = 6
		D[C]	2 + 1.5(4) = 8
		F[C]	4 + 0 = 4
2	F[C]	E[F]	4 + 0 = 4
		G[F]	4 + 0 = 4

Dari tabel 1, dapat dilihat bahwa simpul tujuan G berhasil dicapai dengan jalur C-F-G, dengan total jarak 3, tanpa melalui kemacetan apapun. Sama seperti pada percobaan pertama, jalur C-F-G tetap dipilih, walaupun jarak total sesungguhnya adalah 7, yang lebih besar daripada jarak pada jalur C-D-G, yaitu 6. Namun, dengan adanya kemacetan pada jalan D-G, jalur ini tidak dipilih. Artinya, kemacetan yang ada berhasil dihindari.

V. KESIMPULAN

Algoritma A* adalah salah satu algoritma pencarian rute yang memanfaatkan pencarian heuristik. Algoritma ini dapat digunakan untuk mencari rute yang menghindari kemacetan pada perjalanan. Keefektifan hasil yang diperoleh dari algoritma ini ditentukan oleh kriteria heuristik yang digunakan oleh penggunanya.

VIDEO LINK AT YOUTUBE

https://youtu.be/Nqk_gj8t5u8

ACKNOWLEDGMENT

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa karena hanya atas rahmat dan kurnia-Nya, penulis dapat menyelesaikan makalah ini. Penulis juga berterima kasih kepada Prof. Ir. Dwi Hendratmo Widyantoro, M.Sc., Ph.D. sebagai dosen pengampu mata kuliah Strategi Algoritma yang telah memberikan penulis pengetahuan mengenai materi ini, serta semua asisten mata kuliah yang turut membantu penulis memahami materi ini. Penulis juga mengucapkan terima kasih kepada semua pihak yang telah membantu proses penulisan baik secara langsung maupun tidak langsung, baik dukungan moral maupun pengetahuan tambahan, terutama keluarga dan teman-teman penulis. Penulis juga tak lupa menyampaikan terima kasih kepada seluruh pihak yang telah digunakan karyanya sebagai referensi pada pembuatan makalah ini.

Penulis berharap makalah ini dapat dimanfaatkan untuk sebesar-besarnya untuk kebutuhan pendidikan dalam bidang yang bersangkutan. Penulis menyadari bahwa makalah ini masih memiliki banyak ruang untuk pengembangan, seperti eksplorasi kemungkinan-kemungkinan model algoritma lain yang dapat digunakan pada persoalan yang sama dan penyempurnaan pertimbangan-pertimbangan yang digunakan dalam makalah ini.

REFERENCES

[1] Kbbi.web.id. Kamus Besar Bahasa Indonesia (KBBI). https://kbbi.web.id/macet/, diakses pada 11 Mei 2021.

[2] Merdeka.com. Penyebab Kemacetan Lalu Lintas yang Perlu Diketahui. https://www.merdeka.com/trending/penyebab-kemacetan-lalu-lintas-yang-perlu-diketahui-kln.html, diakses pada 11 Mei 2021.

[3] Sumbarprov.go.id. Sebab dan akibat Kemacetan dijalan Raya. https://sumbarprov.go.id/home/news/1848-sebab-dan-akibat-kemacetan-dijalan-raya.html, diakses pada 11 Mei 2021.

[4] Jabarprov.go.id. Dampak Buruk Kepadatan Lalu Lintas. http://dishub.jabarprov.go.id/artikel/view/114.html, diakses pada 11 Mei 2021.

[5] Maps.google.com. Google Maps. https://www.google.com/maps/@0.0961994,99.8249496,15z, diakses pada 11 Mei 2021.

[6] Businessinsider.com. Google has gotten incredibly good at predicting traffic — here's how. https://www.businessinsider.com/how-google-maps-knows-about-traffic-2015-11?r=US&IR=T, diakses pada 11 Mei 2021.

[7] Binus.ac.id. Searching: Uninformed & Informed. https://socs.binus.ac.id/2013/04/23/uninformed-search-dan-informed-search/, diakses pada 11 Mei 2021.

[8] Munir, Rinaldi. 2020. Penentuan Rute (Route/Path Planning) Bagian 1: BFS, DFS, UCS, Greedy Best First Search. [Internet].

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian1-2021.pdf, diakses pada 11 Mei 2021.

[9] Munir, Rinaldi. 2020. Penentuan Rute (Route/Path Planning) Bagian 2: Algoritma A*. [Internet]. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian2-2021.pdf, diakses pada 11 Mei 2021.

[10] Gatevidyalay.com. A* Algorithm | A* Algorithm Example in AI. https://www.gatevidyalay.com/a-algorithm-a-algorithm-example-in-ai/, diakses pada 11 Mei 2021.

[11] Support.google.com. Melihat tempat, lalu lintas, medan, bersepeda, dan transportasi umum. https://support.google.com/maps/answer/3092439#zippy=%2Clalu-lintas, diakses pada 11 Mei 2021.

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, 26 April 2021

Siti Iedrania Azzariyat Akbar
13519137