Shortest Path Algorithm Pada Aplikasi Google Maps

Ahmad Nadil
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
16521516@std.stei.itb.ac.id

Justin Yusuf Abidjoko
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
16521494@std.stei.itb.ac.id

Kelvin Rayhan Alkarim
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
16521496@std.stei.itb.ac.id

Adrian Fahri Affandi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
16521503@std.stei.itb.ac.id

Ditra Rizqa Amadia
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
16521531@std.stei.itb.ac.id

Raditya Naufal Abiyu
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
16521534@std.stei.itb.ac.id

ABSTRAK

***Abstrak*— Pada abad ke 21 indonesia bahkan dunia sedang mengalami pembangunan besar-besaran. Terkhususnya Indonesia pada era digital di abad 21 ini, pemerintah mengeklaim telah membangun jalan tol hingga 1900 KM dalam 5 tahun terakhir. Semakin banyaknya jalan dan rute yang ada, semakin banyak juga opsi yang dapat diambil mencapai suatu tujuan tertentu. Hal ini menimbulkan suatu permasalahan bagi para pengendara untuk memilih jalan terpendek. Seperti contoh, pemadam kebakaran atau ambulan yang harus mencapai tujuan secara cepat. Hal itu dilakukan untuk meminimalisir baik kerugian atau korban yang ada. Sehingga, diperlukan suatu sistem yang dapat mencari rute terdekat lokasi kejadian. Pada penilitian ini algoritma yang dipakai adalah Algoritma A Star(A*). Algoritma A Star(A*) mengestimasikan jarak terdekat untuk mencapai tujuan dan memiliki nilai heuristik yang digunakan sebagai dasar pertimbangan untuk mencari rute terdekat. Fungsi heuristik yang digunakan adalah funsi yang menghitung selisih koordinat titik awal dan akhir. Hasil dari penelitian ini adalah pengaplikasian rute terdekat yang diinginkan pengguna.**

***Keywords*—A Star(A*), Rute Terdekat, Path Finding**

I. PENDAHULUAN

Pada revolusi teknologi 4.0 yang berjalan saat ini, hampir setiap manusia yang berada di muka bumi ini memiliki telepon genggam milik pribadi. Salah satu fitur yang sering kali digunakan adalah peta digital yang bernama Google Maps. Google Maps adalah layanan pemetaan dunia secara digital yang dikembangkan oleh Google yang memiliki fitur untuk memberikan citra satelit, peta jalan, panorama 360°, kondisi lalu lintas, dan perencanaan rute untuk bepergian dengan berjalan kaki, mobil, sepeda, atau angkutan umum.

Permasalahan utama mobilitas manusia yang sangat tinggi memerlukan jarak terpendek yang akan ditempuh untuk menghemat waktu perjalanan. Rute terpendek juga tergantung dengan kendaraan apa yang digunakan. Fitur Google dalam perencanaan rute oleh Google Maps tentunya dapat memilih rute terpendek yang dapat dilalui oleh pengguna. Pengguna juga dapat menentukan rute terpendek yang sesuai dengan kendaraan pengguna yang pakai. Misalnya, rute tercepat dengan menggunakan sepeda motor, mobil atau bahkan rute tercepat dengan menggunakan kendaraan umum. Dalam makalah ini penulis akan membahas bagaimana fitur-fitur dari Google tersebut dapat bekerja dengan baik dan efisien.

II. METODOLOGI

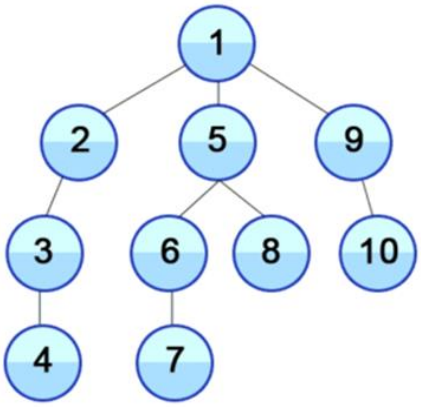
A. *Input dan Output Solusi*

Dalam algoritma *shortest path* pada Google Maps program membutuhkan input berupa lokasi awal. Input awal pengguna akan dijadikan sebagai node pertama. Lalu akan diminta pula tujuan akhir dari pengguna sebagai node terakhir. Output dari program ini adalah jalur/rute terpendek dari kedua titik tersebut.

B. *Karakteristik Solusi*

Depth First Search (DFS), merupakan salah satu algoritma *searching* dalam graf struktur data untuk mencapai tujuan yang diinginkan. Algoritma ini mulai pada *node* pertama dan melakukan *searching* sejauh mungkin pada suatu cabang, jika titik temu tidak ditemukan, maka akan balik ke *node* sebelumnya, atau yang dikenal dengan *BackTracking.* DFS dapat digunakan untuk memecahkan berbagai masalah algoritma, misalnya mendeteksi sebuah cycle dalam sebuah graf, mencari jalur/path, Topological Sorting, menemukan komponen yang memiliki koneksi kuat dalam sebuah graf, bahkan menyelesaikan sebuah puzzle atau maze dengan satu solusi. Secara lebih lanjut, berikut cara melakukan pencarian menggunakan metode DFS :

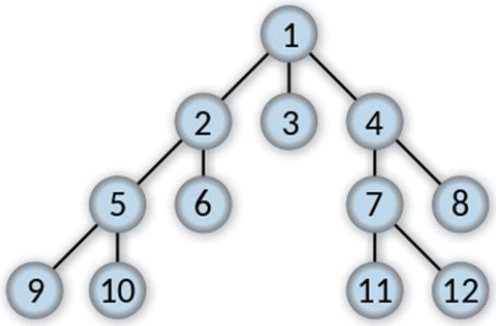
- Mulai pada Node pertama
- Anggap semua *Edge* meninggalkan *Node* pertama dengan suatu urutan
- Ikuti *Edge* pertama dan periksa apakah sudah mencapai *Node* tujuan
- Jika tidak, ulangi proses dengan pindah ke *Node* baru
- Lanjutkan proses tersebut sampai mencapai *Node* tujuan, atau sampai kehabisan opsi. Jika kehabisan opsi, lakukan *BackTrack* ke *Node* sebelumnyam dan gunakan *Node* selanjutnya.



Breadth First Search (BFS), merupakan salah satu algoritma *searching* dalam graf struktur data untuk mencapai tujuan yang diinginkan. Algoritma ini mulai pada sebuah

Node awal, lalu lanjut ke *Node* selanjutnya dan melakukan *Searching* pada *Node* yang setara, lalu akan diteruskan ke tingkat *Node* yang selanjutnya. BFS dapat digunakan untuk berbagai masalah teknologi, misalnya, menemukan jalur terpendek atau *shoretest path*, menemukan sesuatu pada *Search Engine*, *Web Social Networking*, hingga Navigasi GPS. Secara lebih lanjut, berikut cara melakukan pencarian menggunakan metode BFS:

- Mulai pada Node pertama
- Anggap semua Edge meninggalkan Node pertama dengan suatu urutan tertentu
- Ikuti Edge pertama dan periksa apakah itu merupakan Node tujuan
- Jika tidak, coba gunakan Edge selanjutnya dari Node yang sedang digunakan
- Lanjutkan sampai menemukan Node tujuan, atau kehabisan opsi
- Jika kehabisan opsi, pindah ke Node selanjutnya yang memiliki jarak yang sama dengan Node awal
- Jika kehabisan opsi, pindah ke tingkat Node selanjutnya.



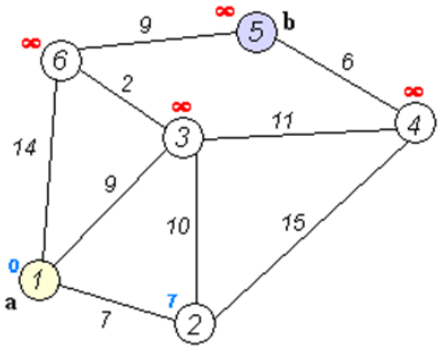
Algoritma Dijkstra, merupakan salah satu dalam algoritma searching dalam graf struktur data yang dikembangkan oleh Edsger.W. Dijkstra. Algoritma ini memiliki penggunaan yang mirip dengan BFS, akan tetapi algoritma ini lebih efektif dalam pencarian jalur/jarak terpendek antara dua titik, karena algoritma ini mengkalkulasikan juga weight atau bobot yang ada untuk mencapai dari satu Node ke Node yang lain. Secara lebih lanjut, berikut cara melakukan pencarian menggunakan metode Dijkstra :

- Pilih Node awal yang tersedia pada graf yang ada bobotnya (Weighted Graph)
- Jarak antar Node diinisiasi dengan nominal tak hingga (infinity)
- Anggap Node yang berada di sebelah Node aktif sebagai Node tetangga, lalu jumlahkan jarak dengan bobot garis pada Node tersebut
- Jika Node tetangga sudah memiliki jarak, maka update jarak yang telah didapatkan pada step sebelumnya jika jarak sekarang lebih kecil dari jarak sebelumnya
- Pilih Node yang memiliki jarak terkecil, dan ulangi step sampai bertemu dengan Node tujuan

Algoritma A* (A Star), merupakan algoritma yang memerika node dengan menggabungkan g(n), yaitu cost yang dibutuhkan untuk mencapai sebuah node dan h(n), yaitu cost yang didapat dari node ke tujuan. Sehingga didapatkan rumus dasar dari algoritma A* ini adalah :

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

Beberapa terminologi dasar yang terdapat pada algoritma ini adalah starting point, simpul (nodes) A, open list, closed list, harga (cost), dan halangan (unwalkable). Prinsip algoritma ini adalah mencari jalur terpendek dari sebuah simpul awal (starting point) menuju simpul tujuan dengan memperhatikan harga (F) terkecil. Prinsip kerja algoritma A* dapat dijelaskan dengan :



- Masukkan node awal ke openlist
- Loop langkah - langkah di bawah ini :
  - Cari node (n) dengan nilai f(n) yang paling rendah dalam openlist. Node ini sekarang menjadi current node.
  - Keluarkan current node dari openlist dan masukan ke closelist.
  - Untuk setiap tetangga dari current node lakukan berikut :
    - Jika tidak dapat dilalui atau sudah ada dalam closelist, abaikan.
    - Jika belum ada di openlist. Buat current node parent dari node tetangga ini. Simpan nilai f,g, dan h dari node ini.
    - Jika sudah ada di openlist, cek bila node tetangga ini lebih baik, mengunakan nilai g sebagai ukuran. Jika lebih baik ganti parent dari node ini di openlist menjadi current node, lalu kalkulasi ulang nilai g dan f dari node ini.
  - Hentikan loop jika :
    - Node tujuan telah ditambahkan ke openlist, yang berate rute telah ditemukan.
    - Belum menemukan node goal sementara openlist kosong atau berarti tidak ada rute.
- Simpan rute. Secara ‘backward’, urut mulai dari node goal ke parent-nya terturs sampai mencapai node awal sambil menyimpan node ke dalam sebuah array.

III. HASIL PENELITIAN

A. Concept Selection

Untuk menentukan algoritma terbaik untuk menentukan shortest path, digunakan metode concept selection terhadap 5 algoritma dengan beberapa kriteria, dan berikut merupakan hasilnya :

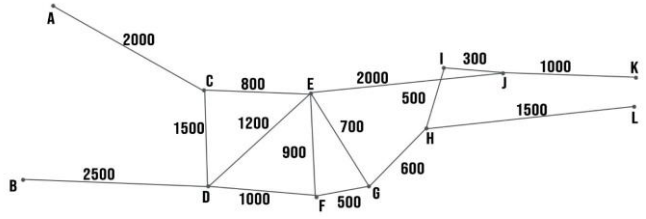
Concept Selection

Criteria	BFS	DFS	Dijkstra's	A*	Manual
Kecepatan Memperoleh Data	-	-	0	+	-
Tingkat akurasi data yang diperoleh	0	-	+	+	0
Tingkat kemudahan penyusunan algoritma	+	+	-	-	0
Tingkat Efisiensi Penggunaan Memori	-	-	+	+	0
Tingkat kemudahan pengaplikasian	+	+	+	+	-
Total +	2	2	3	4	0
Total 0	1	0	1	0	3
Total -	2	3	1	1	2
Final Score	0	-1	2	3	-2
Rank	3	4	2	1	5
Choice	**No**	**No**	**No**	**Yes**	**No**

Jadi berdasarkan penelitian dan referensi yang kami dapatkan, dengan menggunakan analisa concept selection kami mendapatkan hasil bahwa pilihan terbaik yang dapat diambil adalah menggunakan algoritma A*. Hal ini dikarenakan algoritma A* berdasarkan kecepatan memperoleh data, tingkat akurasi data yang diperoleh, tingkat efisiensi penggunaan memori, tingkat kemudahan pengaplikasian mendapatkan hasil yang positif, dan memiliki hasil negatif hanya pada tingkat kemudahan penyusunan algoritma. Walaupun begitu dibanding algoritma yang lainnya, algoritma A* memiliki nilai paling tinggi.

B. Perhitungan Algoritma

Berikut merupakan graph yang kami gunakan, graph ini merupakan peta bagian tengah kota Bandung



Keterangan :

A = Exit tol pasteur
B = Bandara husein
C = Pasteur
D = Pajajaran
E = Cihampelas
F = Cicendo
G = Wastukencana
H = Sultan Agung
I = Dago
J = Gasibu
K = Pusdai
L = Diponegoro

Rumus perhitungan *weight* antar node :

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

Pada graph tree ini, setiap node memiliki atau g(n) sebesar 999. Lalu, h(n) merupakan jarak antar node. Berikut merupakan perhitungan lebih lanjut mengenai *shorets path* dari node A ke L. Karena g(n) bernilai sama di setiap node, maka untuk perhitungannya dapat diabaikan.

Inisiasi pertama pada Node A
A => C : 2000

Lalu, temukan jarak dari node C ke node tetangga

A => C => E : 2800 ☑
A => C => D : 3500 (HOLD) (1)

Karena E memiliki nilai f(n) terkecil, kita telusuri dari Node E dan telusuri sampai bertemu Node terakhir yaitu L
A => C => E => D : 4000 (HOLD) (2)
A => C => E => F : 3700 (HOLD) (3)
A => C => E => G : 3500 ☑
A => C => E => J : Path Not Found

Lanjut penelusuran dari Node G
A => C => E => G => H => L : 5600 ☑
Karena dari Node G ke L hanya ada satu path, dapat langsung diselesaikan.

Lalu kita akan memeriksa nilai Hold yang lebih kecil dari penelusuran Node pertama.
Hold 1 :
A => C => D => F => G => H => L : 7100
Hold 2:
A => C => E => D => F => G => H => L : 7600
Hold 3 :
A => C => E => F => G => H => L : 6300

Karena dari nilai Hold tidak ada yang lebih rendah dari penelusuran pertama, maka penelusuran pertama dianggap menjadi rute paling efisien, atau dapat disebut *Shortest Path*.

C. Source Code

Berikut link github kami yang berisi source code lengkap tentang implementasi algoritma A star dalam bahasa Python :
https://github.com/ditramadia/astar-demo

IV. KESIMPULAN

Pada aplikasi Google Maps, untuk menemukan rute terpendek yang dilalui, dapat menggunakan perhitungan menggunakan algoritma A Star. Dengan menggunakan algoritma ini, kita dapat memperhitungkan beban antar Node yang ada, sehingga membuat perhitungan lebih akurat dengan kondisi di dunia aslinya

V. REFERENSI

[1] H. Agung, "Sistem Penentuan Jarak Terpendek Berdasarkan data koordinat Menggunakan Algoritma Dijkstra Dalam Kasus Pengantaran Barang Se-Jabodetabek," *Jurnal Sisfo Kom (Sistem Informasi dan Komputer)*, vol. 8, no. 1, pp. 14–23, 2019.

[2] A. Candra, M. A. Budiman, and R. I. Pohan, "Application of A-star algorithm on Pathfinding Game," *Journal of Physics: Conference Series*, vol. 1898, no. 1, p. 012047, 2021.

[3] [3] H. Mehta, P. Kanani, and P. Lande, "Google maps," *International Journal of Computer Applications*, vol. 178, no. 8, pp. 41–46, 2019.

[4] K. S. Ali and N. M. Abid, "The Importance of Google Maps for Traffic in Calculating the Level of Service for the Road and Traffic Delay" IOP Conference Series: Materials Science and Engineering, vol. 1076, no. 1, p. 012015, 2020.

[5] Fahui Wang and Yanqing "Estimating O-D Travel Time Matrix by Google Maps API: Implementation, Advantages, and Implications" Annals of GIS, vol. 17, no. 4, p. 199-209

[6] Eric Grimson, MIT Department of Electrical Engineering and Computer Science, 2010.

[7] "Dijkstra's algorithm," Wikipedia, 01-Mar-2022. [Online]. Available: https://en.wikipedia.org/wiki/Dijkstra's_algorithm. [Accessed: 06-Mar-2022].

[8] "Depth-first search," Wikipedia, 02-Mar-2022. [Online]. Available: https://en.wikipedia.org/wiki/Depth-first_search. [Accessed: 06-Mar-2022].

[9] "Breadth-first search," Wikipedia, 04-Mar-2022. [Online]. Available: https://en.wikipedia.org/wiki/Breadth-first_search. [Accessed: 06-Mar-2022].

VI. HAL YANG DIPELAJARI

1. Mempelajari bagaimana *shortest path* dalam sebuah aplikasi atau program dapat ditemukan.
2. Dapat membuat implementasi dari algoritma A star dalam program berbahasa Python.
3. Teamwork dan time management merupakan point yang penting dalam menyelesaikan makalah ini

VII. PEMBAGIAN KERJA ANGGOTA KELOMPOK

Kami membuat laporan ini secara bersama-sama, akan tetapi terdapat bagian yang salah satu dari kami lebih dominan dalam pengerjaannya, yaitu:

1. Justin : Membuat penjelasan mengenai algoritma BFS dan DFS
2. Kelvin : Merumuskan input dan output solusi, hal-hal yang dipelajari, serta membuat powerpoint presentasi
3. Adrian : Membuat abstrak serta pendahuluan
4. Nadil : Membuat penjelasan tentang algoritma Dijkstra dan A star serta merumuskan perhitungan algoritma AStar secara manual.
5. Ditra : Membuat dan mendemonstrasikan algoritma A* dalam bahasa Python
6. Radit : Membuat dan merancang tabel concept selection, graph tree yang digunakan, dan segala hal yang berkaitan dengan design