Aplikasi Algoritma Gugus A* Dua Arah dalam Pencarian Jarak Terdekat Antar Dua Buah Simpul Pada Sebuah Peta Digital

Putu Gery Wahyu Nugraha 13516100
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
gerywahyunugraha@gmail.com

***Abstract*—Peta digital merupakan sebuah komponen yang tidak dapat dilepaskan dari kehidupan bernavigasi kita. Untuk merancang sebuah piranti lunak peta digital diperlukan piranti keras yang cukup mumpuni karena proses dan memori yang diperlukan akan cukup besar. Perlu dicari sebuah teknik untuk dapat menjalankan sistem peta digital dalam perangkat yang tidak memiliki speisifkasi mumpuni**

Keywords—peta, digital, teknik, spesifikasi, sistem, piranti, lunak, keras

I. PENDAHULUAN

Peta baik fisik maupun digital merupakan sebuah gambaran yang berisi tentang informasi pada suatu daerah. Pada jaman sekarang sudah tidak lazim untuk menemukan peta yang berbentuk fisik karena hampir semuanya sudah di konversi menjadi bentuk digital. Namun peta fisik memiliki banyak kekurangan, salah satu contohnya adalah ukurannya yang konstan. Membaca peta fisik sembari mengendarai kendaraan merupakan hal yang sangat tidak nyaman untuk dilakukan.

Oleh karena itu diciptakan sebuah alternatif dari peta fisik, yaitu peta digital. Dengan adanya peta digital, peta bisa menjadi lebih membantu untuk kita. Dengan peta fisik, menentukan jalur terdekat dari dua buah simpul, asal dan tujuan, mungkin akan memerlukan konsentrasi yang luar biasa dari pembacanya. Namun dengan adanya peta digital kita dapat meminta bantuan komputer untuk melakukan komputasi jarak terdekat.

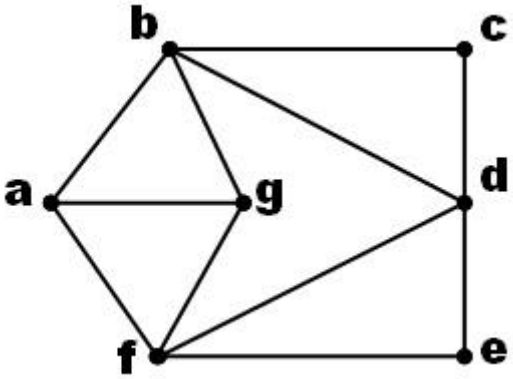
Tentunya pada peta yang ukurannya kecil, mencari jarak terdekat adalah hal yang wajar, dan implementasinya dapat dilakukan dengan melakukan iterasi ke semua kemungkinan jalur yang ada. Namun hal ini sangat tidak efektif jika digunakan pada peta dunia. Peta dunia sendiri memiliki jutaan bahkan miliaran simpul, dan untuk mengiterasi semua kemungkinan jalur pada simpul tersebut akan membutuhkan waktu yang sangat lama, bahkan untuk komputer tercanggih sekalipun.

Oleh karena itu dibutuhkan sebuah algoritma lain yang dapat menghitung jalur tercepat dalam waktu yang lebih singkat. Beberapa algoritma yang ada di pasaran dan mampu melakukan tugas ini adalah algoritma Djikstra dan A*. Namun terdapat satu masalah pada algoritma ini. Pada kasus peta dunia penghitungan jalur terdekat antar dua simpul yang cukup jauh jaraknya (jalur terdekat antara Jakarta dan London) sering kali tidak perlu mempedulikan simpul-simpul yang merepresentasikan jalan-jalan kecil di Jakarta. Karena yang diperlukan hanya gambaran secara umum jalur terdekat antar dua kota tersebut. Namun pada algoritma A* maupun Djikstra simpul-simpul tersebut tetap masuk ke dalam perhitungan sehingga terdapat penggunaan memori maupaun kemampuan komputasi dari komputer yang sebenarnya dapat dikurangi namun tidak pada implementasinya.

II. LANDASAN TEORI

A. Graf

Graf merupakan sebuah struktur data yang terdiri dari objek-objek bernama simpul. Simpul-simpul ini dapat dihubungkan ke satu sama lain melalui sebuah sisi [1]. Ketika dua buah simpul pada satu graf dihubungkan dengan sebuah sisi, kita mengatakan bahwa kedua simpul tersebut bertetangga. Sisi pada sebuah graf yang memiliki arah bernama graf berarah sedangkan sisi yang tidak memiliki arah disebut dengan graf tak berarah. Setiap sisi pada graf juga dapat memiliki suatu nilai yang dinamakan berat. Kalimat teknisnya, graf merupakan sebuah pasangan $G = (V, E)$ dari simpul dan sisi.



Gambar 1 Ilustrasi Graf
(http://1.bp.blogspot.com/_JeGWge2hu1c/TE8BZ_6oi-I/AAAAAAAAADA/313FNydAdyc/s1600/euler.jpg)

Beberapa kata kunci yang mungkin diasosiasikan dengan graf:

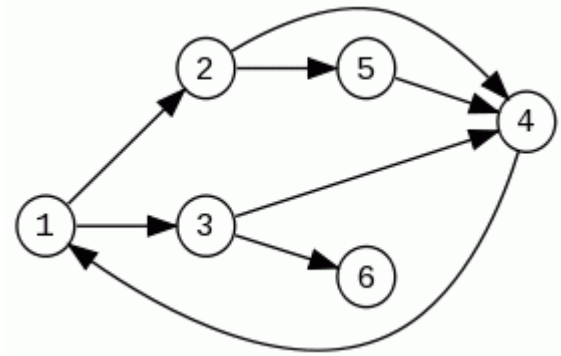
1. Simpul: sebuah titik atau objek dalam graf.
2. Sisi: sebuah garis yang menghubungkan dua simpul.
3. Berat: sebuah nilai yang merepresentasikan panjang dari satu sisi.
4. Tetangga: semua simpul yang dihubungkan oleh sebuah sisi.
5. Derajat: jumlah sisi yang terhubung pada simpul

B. *Graf Berarah*

Graf berarah merupakan salah satu tipe graf dimana pada sisinya terdapat sebuah data arah [2]. Data arah ini memberikan sebuah limitasi ketika penelusuran graf dilaksanakan. Tipe graf ini berguna ketika mencoba menggambarkan hubungan antar dua simpul dimana sebuah penelusur hanya dapat menelusuri dari simpul A ke simpul B tapi tidak dapat dari simpul B ke simpul A (contoh implementasinya dalam dunia nyata adalah jalan satu arah dimana suatu mobil dapat bergerak dari persimpangan A ke persimpangan B tapi tidak dari B ke A). Maka dari itu penggunaan graf bearah sendiri umum dijumpai pada penggambaran peta dunia.

C. *Best-First Search*

Best-first search (BFS) merupakan sebuah algoritma yang berfungsi untuk menelusuri pohon atau graf. Ketika diaplikasikan pada sebuah pohon, BFS akan menelusuri pohon tersebut dimulai dari akarnya, lalu penelusur akan menelusuri setiap anak dari akar tersebut. Urutan penelusuran dari BFS juga memperhitungkan seberapa menjanjikan sebuah simpul (seberapa keuntungan yang didapat ketika memilih simpul tersebut) [3].

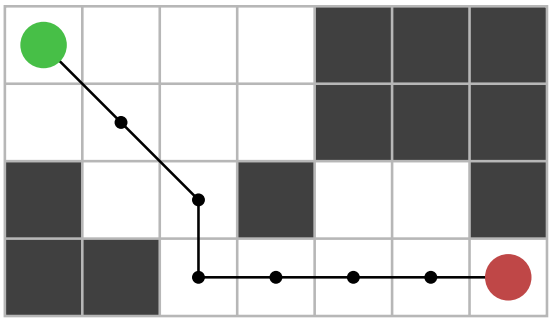


Gambar 2 Urutan Penelusuran pada BFS
(http://www.algorytm.org/images/grafy/przeszukiwanie.gif)

Algoritma ini juga memiliki alur kerja yang sama pada graf, yang membedakan hanya pada graf simpul mulanya ditentukan oleh pengguna. Untuk algoritma BFS kompleksitasnya dapat diekspresikan sebagai O ($|V| + |E|$). Dimana V adalah jumlah simpul dan E adalah jumlah sisi. Hal ini masuk akal karena untuk kasus terburuk semua simpul dan sisi akan ditelusuri oleh BFS, sehingga nilai kompleksitasnya akan berupa total jumlah dari keduanya.

D. *Algoritma A**

Pada ilmu komputer sains, A* merupakan sebuah algoritma komputer yang umum digunakan dalam pencarian jalur atau penelusuran graf. Algoritma ini umum digunakan karena keakuratan dan performanya yang jauh lebih baik dibandingkan algoritma lain, seperti Djikstra maupun *brute-force*. Namun pada praktiknya algoritma ini kalah saing dengan algoritma yang mampu melakukan *pre-process* data.



Gambar 3 Penelusuran menggunakan A* lazim digunakan untuk mencari jalur terdekat
(http://www.growingwiththeweb.com/images/2012/06/03/grid-example.svg)

Layaknya algoritma penelusuran graf lainnya, A* sendiri menelusuri setiap simpul pada graf, namun algoritma A* memprioritaskan simpul yang menurutnya akan mengarah ke simpul tujuan lebih cepat (pencarian seperti ini dinamakan dengan pencarian *best-first*). Pada setiap iterasi penelusuran A* harus menentukan apakah jalur yang saat ini ia telusuri akan ia berikan prioritas keberapa. Hal ini dia lakukan dengan menggunakan sebuah fungsi heuristik, secara spesifik A* mencari jalur yang mampu meminimumkan $f(n) = g(n) + h(n)$. Dimana n merupakan simput terakhir yang ada pada jalur, $g(n)$ merupakan *cost* untuk jalur dari simpul awal ke simpul n, dan $h(n)$ merupakan fungsi heuristic yang mengestimasi *cost* minimum untuk jalur dari simpul n ke simpul tujuan.

Perlu diketahui bahwa Algoritma A* sendiri tidak menghasilkan solusi yang selalu tepat, karena ketepatan solusinya sendiri sangatlah bergantung dengan fungsi heuristik yang ada. Namun dengan pemilihan fungsi heuristic yang tepat solusi yang didapatkan umumnya akurat.

E. *A* Dua Arah*

A* dua arah merupakan sebuah varian A* dimana selain melakukan penelusuran dari suatu simpul awal ke simpul tujuan komputer juga melakukan penelusuran dari simpul tujuan ke simpul awal. Harapan dari penelusuran dua arah ini adalah penelusur tersebut bertemu di suatu simpul di tengah pohon penelusuran. Hal ini mungkin terjadi karena jalur terbaik dari simpul awal ke simpul tujuan harusnya identic dengan jalur terbaik dari simpul tujuan ke simpul awal. Dengan ini penelusur tidak perlu lagi menelusuri setengah dari pohon penelusuran yang ada. Namun pada kasus terburuk, kedua penelusur tidak akan pernah bertemu. Hal ini dapat disebabkan oleh fungsi heuristik yang buruk menyebabkan jalur terbaik dari simpul

awal ke simpul tujuan berbeda dengan jalur terbaik dari simpul tujuan ke simpul awal, atau dapat juga disebabkan oleh adanya graf yang berarah. Pada kasus ini setiap penelusur akan tetap menelusuri semua pohon penelusuran dan kompleksitasnya akan sama dengan sebelumnya.

*F. Gugus A**

Gugus atau yang lebih familiar disebut dengan *cluster* merupakan suatu konsep dimana setiap simpul pada graf dapat dikelompokkan menjadi sebuah simpul yang ukurannya lebih besar. Penggugusan ini dapat digunakan sebagai bentuk generalisasi dari simpul-simpul yang berkaitan. Dengan melakukan penggugusan suatu algoritma tidak perlu menelusuri tiap simpul yang terdapat pada sebuah kumpulan simpul. Hanya jika diperlukan penelusuran yang lebih akuratlah, kelompok simpul tersebut baru dibuka.

III. Implementasi Graf Dan Algoritma Gugus A* Dua Arah Pada OpenStreetMap

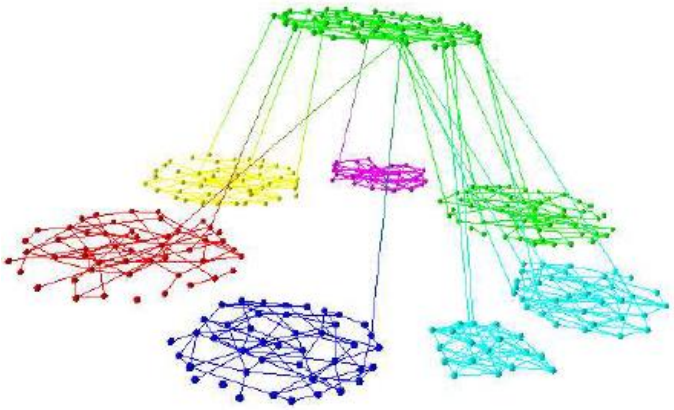
A. Implementasi Graf dan Gugus Graf pada Peta Digital

Seperti yang telah dijelaskan sebelumnya pada pendahuluan dan dasar teori peta digital yang baik adalah kumpulan simpul yang membentuk sebuah graf. Secara spesifik simpul pada graf sendiri melambangkan tiap persimpangan yang berada pada jalan. Setiap simpul mungkin mengandung beberapa informasi mengenai persimpangan jalan tersebut, seperti nama atau informasi apakah pada persimpangan tersebut ada lampu lalulintas atau tidak. Tentunya kumpulan persimpangan sendiri tidak dapat membentuk peta digital yang berguna, karena pada peta digital kita perlu juga mencari jalan antar satu persimpangan ke persimpangan yang lain. Jalan tersebut dilambangkan dengan sisi pada graf.

Gambar 4 Salah satu implementasi naif graf untuk peta digital dalam Google Map
(https://i1.wp.com/notes.zouhairj.com/wp-content/uploads/2018/01/p29018.jpg)

Setiap sisi pada graf pasti menghubungkan sebuah simpul ke simpul yang lain, sisi pada graf tersebut dapat berarah, penelusur hanya dapat menelusuri sisi tersebut pada satu arah tertentu (jalan satu arah), ataupun tidak berarah seperti jalan pada umumnya. Maka dari itu jika ada simpul A dihubungkan ke simpul B oleh sebuah sisi C itu berarti pada peta tersebut ada sebuah persimpangan A yang terhubung dengan persimpangan B oleh sebuah jalan C. Pada bahasa favorit anda, hal ini dapat diimplementasikan dengan membuat sebuah objek yang memiliki pointer ke objek lain, atau jika ingin lebih komples (sisi dapat mengandung sebuah informasi) pointer dapat digantikan dengan sebuah objek sisi yang memiliki informasi dua objek simpul yang dihubungkan.

Mengacu pada pendahuluan, implementasi graf diatas merupakan implementasi yang naif. Hal ini dikarenakan pada peta negara atau dunia akan terdapat jutaan bahkan miliaran simpul yang harus disimpan datanya. Namun pada penggunaanya tidak semua simpul tersebut perlu diproses oleh komputer. Contohnya adalah untuk seorang pengguna yang ingin berpergian disekitar Bandung, akan lazim baginya untuk melihat hanya jalan-jalan atau persimpangan-persimpangan di sekitar Bandung. Pada kasus seperti ini komputer perlu menampilkan semua informasi simpul yang ada karena semuanya signifikan bagi pengguna. Namun pada kasus lain, pengguna mungkin hanya ingin mengetahui jalur terdekat antar kota Jakarta dan Bandung. Pada konteks ini komputer tidak perlu mengetahui setiap simpul yang membentuk kota Jakarta maupun kota Bandung, yang komputer perlu ketahui adalah ada jarak dari satu simpul Jakarta ke simpul Bandung.



Gambar 5 Contoh gugus graf, setiap kumpulan simpul membentuk simpul baru lagi diatasnya
(http://www.it.usyd.edu.au/~shhong/img/cluster1.png)

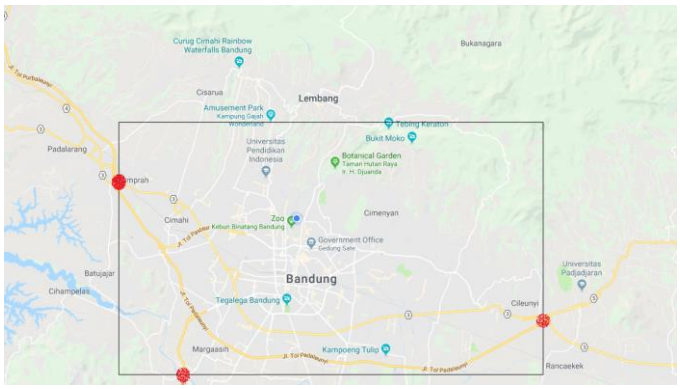
Tentunya hal ini tidak dapat diselesaikan dengan implementasi graf secara naif, karena jika mengimplementasikannya secara naif daerah dan menggunakan patokan persimpangan sebagai simpul, daerah Jakarta maupun Bandung tidak akan mampu direpresentasikan oleh sebuah simpul. Maka dari itu kita harus mengubah implementasinya. Dengan implementasi yang baru, kita akan mengelompokkan simpul-simpul yang berkaitan contohnya simpul-simpul yang terdapat pada kabupaten Bandung dan membuat sebuah simpul raksasa yang mampu merepresentasikan semua simpul dibawahnya. Tentu saja ukuran gugus ini kita bisa atur semau kita, jika menginginkan efisiensi memori tentunya akan lebih baik untuk mengelompokkan simpul tersebut berdasarkan kota maupun provinsi, namun hal ini mungkin memberikan generalisasi yang

kurang baik dalam konteks kelurahan atau desa. Maka dari itu layaknya implementasi gugus ini disesuaikan dengan kebutuhan pasar maupun klien dimana piranti lunak ini diperuntukkan.

B. *Penelusuran Graf dan Gugus Graf Menggunakan Algoritma A* pada Peta Digital*

Penelusuran Graf menggunakan algoritma A* telah dibahas sebelumnya di landasan teori. Secara singkat algoritma A* akan mengiterasikan semua simpul yang berdekatan dengan simpul saat ini lalu menghitung nilai setiap fungsi menggunakan sebuah fungsi heuristik. Kemudian algoritma akan mencoba untuk meminimumkan nilai dari fungsi heuristic sembari mengiterasikan simpul yang bersebelahan. Proses ini akan dilakukan berulang-ulang sampai penelusur sampai ke simpul tujuan. Sebenarnya algoritma A* ini dapat digunakan untuk gugus graf juga, selama pengguna tidak menghiraukan simpul-simpul yang membentuk gugusnya, contohnya pengguna hanya ingin melihat jarak antara Jakarta ke Bandung. Namun ketika pengguna ingin melihat lebih detil ke dalam simpul yang membentuk gugus diperlukan sebuah algoritma khusus yang mampu menggunakan jalur optimal sebelumnya dan lebih mendetilkannya lagi, contoh kasusnya adalah ketika pada Google Maps pengguna mencari jarak antar Jakarta dan Bandung namun setelah itu pengguna ingin mendetilkan hasil pencariannya ke daerah Cisitu.

Ada dua cara yang dapat dilakukan untuk menyelesaikan masalah tersebut. Cara pertama adalah ketika pengguna meminta informasi yang lebih detil, algoritma A* akan digunakan kembali namun tanpa menggunakan generalisasi untuk kota. Dengan ini semua gugus graf dipecah kembali menjadi simpul-simpulnya masing-masing. Tentunya hal ini sama saja dengan membuang informasi yang telah kita dapatkan sebelumnya, jalur optimal dari gugus graf satu ke gugus graf lainnya, dan mengulang semuanya kembali. Hal ini tidak disarankan karena menghilangkan tujuan penulis membuat penggugusan pada awalnya.



Gambar 6 Contoh desain gugus simpul yang menggunakan tiga simpul keluar

Terdapat cara alternatif yang dapat dilakukan. Namun sebelum melaksanakannya desain dari gugus graf itu sendiri harus diperbaiki. Semisal kita ingin membuat gugus graf kota Jakarta, daripada kita membuat keseluruhan kota Jakarta sebagai sebuah simpul besar akan lebih baik jika kita mencari titik keluar utama dari kota Jakarta dan membuatnya menjadi simpul. Jika kita merujuk pada gambar diatas, dapat ditemukan bahwa ada tiga titik keluar utama di Bandung (yang ditandai oleh titik merah). Ketiga titik ini dapat melambangkan kota Bandung. Ketika penelusur melakukan penelusuran jalur terdekat dari Kota Bandung ke Kota Jakarta, titik yang didapatkan (pada Kota Bandung) pastilah merupakan salah satu dari ketiga titik merah tersebut. Untuk mendapatkan informasi yang lebih detil, komputer dapat melanjutkan pencarian dengan menggunakan titik yang didapatkan sebelumnya sebagai titik asal/tujuan. Contohnya, A ingin mencari jarak terdekat antar Kota Bandung dengan Kota Jakarta, setelah pencarian, komputer menentukan bahwa titik keluar terdekat ke Kota Jakarta terdapat pada simpul tol Padalarang. Ketika A ingin melihat lebih detil dan mencari jarak terdekat antar Kota Jakarta ke daerah Cisitu di Bandung. Komputer dapat menggunakan informasi yang didapatkan tadi, bahwa titik terdekat ke Kota Jakarta terdapat pada simpul tol Padalarang, dan melanjutkan penelusuran dengan menggunakan tol Padalarang sebagai simpul asal dan daerah Cisitu sebagai simpul tujuan.

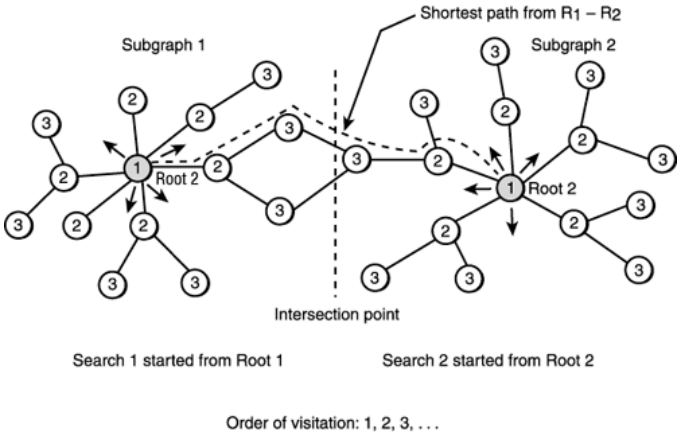
C. *Optimisasi Penelusuran Graf Menggunakan Algoritma A* Dua Arah*

Melakukan generalisasi pada simpul memang membantu terhadap performa namun walaupun kita menggunakan kota sebagai acuan gugus, akan masih terdapat banyak simpul yang harus ditelusuri menggunakan algoritma A*. Dari data sendiri didapatkan bahwa terdapat lebih dari 500.000 kota ada di dunia ini. Oleh karena itu diperlukan sebuah teknik baru yang dapat mempercepat algoritma A* lagi. Salah satunya adalah dengan menggunakan algoritma A* dua arah.

Algoritma ini merupakan salah satu bentuk modifikasi terhadap A* normal. Pada A* normal penelusuran hanya dilakukan dari simpul awal ke simpul tujuan, atau hanya ada satu penelusur yang menelusuri graf. Namun pada algoritma A* dua arah digunakan dua buah penelusur. Penelusur pertama akan menelusuri graf mulai dari simpul tujuan sedangkan penelusur kedua akan menelusuri graf dari simpul awal. Harapan dari teknik ini adalah kedua penelusur bertemu di tengah graf pencarian sehingga satu penelusur tidak perlu menelusuri seluruh graf untuk menemukan jalur terbaik.

Namun pada kasus terburuk, teknik ini menghasilkan hasil yang tidak lebih baik dibandingkan dengan A* normal. Hal ini terjadi karena pada kasus terburuk, kedua penelusur tersebut tidak akan pernah bertemu satu sama lain (hal ini mungkin disebabkan oleh fungsi heuristik yang buruk ataupun adanya sisi berarah yang menyebabkan satu penelusur tidak dapat mengambil jalur yang sama dengan penelusur lainnya).

Pada kasus terbaik, A* dua arah akan menghasilkan hasil yang jauh lebih baik dibandingkan A* biasa. Hal ini disebabkan oleh fakta bahwa penelusur tidak perlu menelusuri sisi lain pada graf. Dengan menggunakan fungsi herusitik yang baik ditambah dengan topografi jalan yang mendukung, dua penelusur pasti dapat dipastikan bertemu di satu titik.



Gambar 7 Contoh penelusuran menggunakan dua penelusur (https://infoarena.ro/blog/meet-in-the-middle?action=download&file=12fig26.gif&safe_only=true)

D. Analisis Performa Penelusuran Graf dan Gugus Graf Menggunakan A Normal dan A* Dua Arah*

Untuk menganalisa efek dari struktur data gugus graf dibandingkan dengan graf kita harus menggunakan beberapa asumsi. Asumsi yang pertama adalah pembuat program menggunakan abstraksi tingkat kota untuk grafnya. Asumsi yang kedua adalah jumlah jalan ataupun persimpangan pada sebuah kota adalah 1.500 dan kita akan membatasi sudut pandang kita pada pulau jawa saja. Di Jawa sendiri terdapat 42 kota. Dan kita berikan asumsi lagi bahwa pengelompokan kota sedemikian rupa sehingga terdapat 5 simpul keluar untuk setiap kota. Dari sini sesuai naluri dapat kita simpulkan bahwa kota yang dulunya direpresentasikan oleh 1.500 simpul dapat dengan mudah kita representasikan dengan 5 simpul saja. Jika misal simpul adalah sebuah pointer ke sebuah memori, karena pointer tersebut umumnya berukuran 8-byte (64-bit memori) Maka data tersebut yang dulunya disimpan dalam memori sebesar 504 KB dapat disimpan dengan mudah pada memori sebesar 1.68 KB saja dengan reduksi ukuran sebesar 99.67%

$$Memory = jumlah\ kota * banyak\ simpul * ukuran\ simpul$$

Angka diatas hanya benar adanya jika pengguna tidak mencoba untuk melihat informasi jalur secara lebih detil. Jika memang pengguna menginginkan hal tersebut, jumlah simpul yang diperlukan akan sama halnya dengan graf tanpa gugus.

Analisis terhadap performa akan dilakukan dengan memperhitungkan kompleksitas dari implementasi graf dan gugus graf dengan menggunakan algoritma BFS secara umumnya. Hal ini dilakukan karena pada algoritma A* sendiri kompleksitasnya sangatlah bergantung dengan fungsi heuristik yang ada. Dan hasil yang akurat tidak akan dapat ditemukan. Algoritma BFS pada umumnya memiliki kompleksitas sebesar

$$O(k^d)$$

Dengan melakukan abstraksi pada simpul yang tidak perlu, dan dengan mengambil asumsi bahwa pengguna hanya ingin melihat detil tingkat kota. Maka dengan menggunakan asumsi yang telah disebutkan pada paragraph sebelumnya, jumlah dari simpul dan sisi akan direduksi sebanyak.

$$jumlah\ simpul = \frac{jumlah\ simpul\ awal}{\frac{jumlah\ simpul\ per\ kota}{jumlah\ simpul\ keluar}}$$

Begitu juga dengan jumlah sisi pada graf akan direduksi sebanyak

$$jumlah\ sisi = \frac{jumlah\ sisi\ awal}{\frac{jumlah\ sisi\ per\ kota}{jumlah\ sisi\ keluar}}$$

Pada akhirnya algoritma BFS akan memiliki kompleksitas (big O) yang sama namun dengan jumlah sisi dan simpul yang lebih kecil, secara nilai kompleksitasnya akan jauh lebih kecil.

Untuk menghitung efek dari algoritma A* dua arah dibandingkan dengan algoritma A* pada umumnya kita harus melakukan asumsi bahwa fungsi heuristik yang digunakan pada algoritma A* ini akan menghasilkan kasus yang terburuk. Hal ini kita lakukan karena jika tidak algoritma A* akan memiliki kompleksitas yang hampir konstan, atau tak dapat ditentukan. Pada kasus terburuk, algoritma A* memiliki kompleksitas sama dengan BFS pada umumnya. Dengan mengaplikasikan algoritma A* dua arah, terdapat dua hasil yang mungkin didapatkan. Hasil yang pertama adalah kasus terbaik dari algoritma A* dua arah ditemukan, dimana setiap penelusur bertemu di tengah. Kasus ini akan menghasilkan kompleksitas sebesar

$$O(2k^{\frac{d}{2}})$$

Pada kasus terburuk kedua penelusur tidak akan bertemu sama sekali, kompleksitas yang dihasilkan akan sama dengan BFS.

$$O(\frac{2k^d + 2k^{\frac{d}{2}}}{2})$$

IV. Simpulan

Membuat sebuah piranti lunak yang mampu menyajikan peta digital diperlukan sebuah piranti keras yang mumpuni untuk mendukung kebutuhan performa dari piranti lunak. Tapi terdapat beberapa cara untuk mengakali batasan performa ini sehingga perangkat lunak dapat dijalankan pada piranti keras yang spesifikasinya tidak terlalu beras. Menggunakan abstraksi data adalah salah satu cara yang tepat untuk mengurangi penggunaan memori dari sistem peta digital. Sedangkan jika yang menjadi masalah adalah kekuatan prosesi maka akan jauh lebih baik untuk menerapkan A* dua arah dibandingkan A* yang biasa. Kombinasi kedua teknik ini juga dapat digunakan untuk mengurangi beban baik pada memori dan juga kemampuan proses.

V. Ucapan Terima Kasih

Saya ingin memberikan ucapan terima kasih kepada para penulis di Stack Overflow yang telah berkontribusi sangat banyak dalam memberikan saya ide untuk tulisan saya kali ini. Saya juga ingin mengucapkan terima kasih kepada semua teman saya yang telah membantu saya dalam pemilihan topik sehingga topik yang saya buat tetap orisinil sifatnya. Saya juga ingin berterimakasih kepada orang tua saya atas dukungan yang

mereka berikan selama masa pembelajaran saya. Dan terakhir saya ingin ucapkan terima kasih kepada Tuhan yang telah memberikan saya ketenangan selama proses pembuatan makalah ini.

Saya sadar bahwa makalah ini dipenuhi dengan kesalahan-kesalahan seperti simplifikasi yang saya lakukan maupun asumsi yang terlalu jauh dari fakta yang ada di lapangan. Saya mohon maaf bila kesalahan tersebut menganggu pembaca saya. Saya harap tulisan saya ini dapat digunakan sebagai bahan pembimbing untuk membantu mahasiswa lain yang ingin mempelajari ide alternatif untuk implementasi peta digital.

DAFTAR PUSTAKA

[1] Trudeau, Richard J. (1993). Introduction to Graph Theory (Corrected, enlarged republication. ed.). New York: Dover Pub. hal. 19. ISBN 978-0- 486-67870-2. Diakses Mei 14, 2018.

[2] Bang-Jensen & Gutin (2000). Diestel (2005), Bab 1.10. Bondy & Murty (1976), Section 10. Diakses Mei 14, 2018.

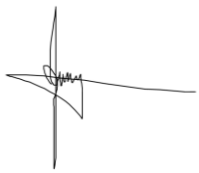
[3] Russell, Stuart J.; Norvig, Peter (2003), Artificial Intelligence: A Modern Approach (2nd ed.), Upper Saddle River, New Jersey: Prentice Hall, ISBN 0-13-790395-2. hal. 94 and 95 (note 3). Diakses Mei 14, 2018.

[4] Delling, D.; Sanders, P.; Schultes, D.; Wagner, D. (2009). *Algorithmics of Large and Complex Networks: Design, Analysis, and Simulation*. Springer. hal. 11. Diakses Mei 14, 2018.

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, 14 Mei 2018



Putu Gery Wahyu Nugraha 13516100