Penerapan Algoritma Simplified-Memory-Bounded A* dan Algoritma Greedy-Best First Search dalam Pencarian Lintasan Terpendek dan Efisiensi Tarif Perjalanan Antar Kota

Yongke Yoswara - 13508034
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18034@students.if.itb.ac.id

***Abstraksi*—saat ini, masyarakat di berbagai penjuru dunia seringkali menggunakan kendaraan umum untuk berpergian dari satu kota menuju kota lainnya. Untuk sampai dari satu daerah/kota ke daerah/kota tujuan, seringkali melewati berbagai kota dan harus melakukan pergantian kendaraan. Sebagai contoh, masyarakat akan pergi dari kota A menuju kota G. maka masyarakat ingin memilih jalur yang pendek dan efisien supaya menghemat biaya dan juga waktu. Untuk mendapatkan solusi jalur yang terpendek dan efisie, dapat digunakan algoritma SMA* dan algoritma greedy-best first search. Kedua algoritma tersebut merupakan bagian dari metode Heuritic dalam algoritma pencarian. Metode ini menggunakan sebuah fungsi untuk melakukan penghitungan biaya perkiraan dari satu simpul ke simpul lainnya.**

***Kata Kunci*—Greedy-Best First Search, SMA*, jalur pendek-efisien, Heuristic**

I. PENDAHULUAN

Sekarang ini, masyarakat di Indonesia seringkali menggunakan angkutan umum (bus, angkot, ojek, dll) untuk berpergian. Untuk perjalanan di dalam kota, masyarakat sering menggunakan angkutan kota(angkot), bus dalam kota, dan ojek. Sedangkan untuk berpergian antar kota antar provinsi, masyarakat menggunakan bus.

Masyarakat biasanya sudah di-doktrin untuk menggunakan jalur yang sama setiap akan berpergian. Misalkan, apabila di dalam kota, ada seorang karyawan yang selalu pergi menuju kantor menggunakan angkot setiap harinya. Dia selalu menggunakan angkota yang sama setiap harinya. Tetapi apabila dilihat lebih jauh lagi, ada jalur angkot lain yang akan membuat karyawan itu mengeluarkan uang lebih sedikit atau justru ada jalur lain yang lebih pendek daripada jalur yang biasanya ditempuh oleh karyawan tersebut (terlepas dari macet atau tidaknya perjalanan).

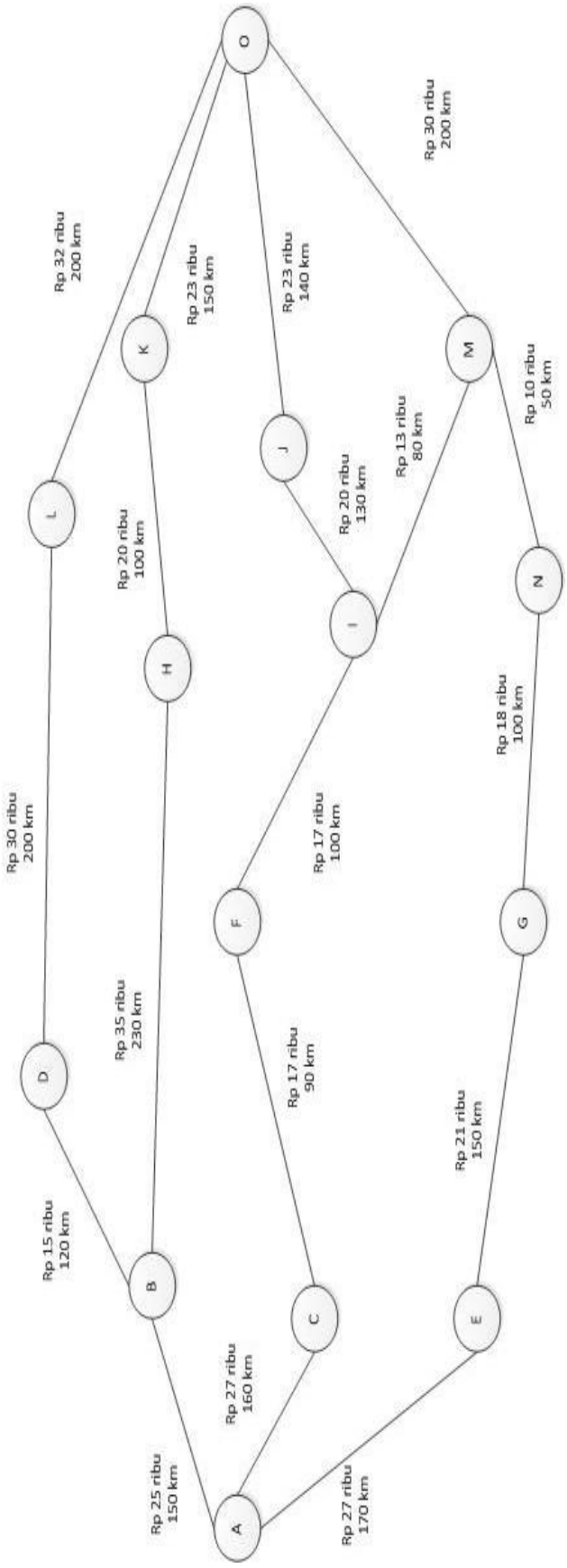
Begitu pula dengan perjalanan menggunakan bus antar kota. Seperti yang kita ketahui, banyak sekali kota yang terdapat di Indonesia. Oleh sebab itu, makalah kali ini akan membahas mengenai permasalahan biaya dan jalur terpendek untuk berpergian dari satu kota menuju kota lainnya.

Permasalahan yang ada adalah bagaimana kita mengetahui jalur yang paling efisien untuk pergi dari satu kota ke kota lainnya. Untuk memudahkan pengertian mengenai makalah ini, penulis akan memberikan gambaran singkat dimana apabila kita akan pergi dari kota A menuju kota O, maka akan ada banyak jalur yang bisa ditempuh dengan berbagai macam jarak dan tarif antar kotanya.

Setiap perjalanan dari simpul satu ke simpul lainnya, akan ada dua variable. Variable pertama adalah jarak yang menghubungkan kedua simpul. Sedangkan variable kedua adalah biaya yang harus dibayarkan oleh seseorang apabila ingin berpergian dari satu simpul ke simpul lainnya.

Jalur-jalur yang dapat ditempuh dari kota A menuju kota O adalah sebagai berikut :

1. A > B > D > L > O
2. A > B > H > K > O
3. A > C > F > I > J > O
4. A > E > G > I > J > O
5. A > E > G > N > M > O



Gambar 1 Gambar contoh rute perjalanan

II. PENCARIAN

Dalam ilmu komputer, sebuah algoritma pencarian dijelaskan secara luas adalah sebuah algoritma yang menerima masukan berupa sebuah masalah dan menghasilkan sebuah solusi untuk masalah tersebut, yang biasanya didapat dari evaluasi beberapa kemungkinan solusi. Sebagian besar algoritma yang dipelajari oleh ilmuwan komputer adalah algoritma pencarian. Himpunan semua kemungkinan solusi dari sebuah masalah disebut ruang pencarian. Algortima pencarian brute-force atau pencarian naif/uninformed menggunakan metode yang sederhana dan sangat intuitif pada ruang pencarian (metode blind), sedangkan algoritma pencarian informed menggunakan heuristik untuk menerapkan pengetahuan tentang struktur dari ruang pencarian untuk berusaha mengurangi banyaknya waktu yang dipakai dalam pencarian.

II.1 Metode Blind

Sebuah algoritma pencarian uninformed adalah algoritma yang tidak mempertimbangkan sifat alami dari permasalahan. Oleh karena itu algoritma tersebut dapat diimplementasikan secara umum, sehingga dengan implementasi yang sama dapat digunakan pada lingkup permasalahan yang luas, hal ini berkat abstraksi. Kekurangannya adalah sebagian besar ruang pencarian adalah sangat besar, dan sebuah pencarian uninformed (khususnya untuk pohon) membutuhkan banyak waktu walaupun hanya untuk contoh yang kecil. Sehingga untuk mempercepat proses, kadang-kadang hanya pencarian informed yang dapat melakukannya.

II.2 Metode Heuristik

Metode pencarian heuristic merupakan teknik yang digunakan untuk meningkatkan efisiensi dari proses pencarian. Dalam pencarian state space, heuristik adalah aturan untuk memilih cabang-cabang yang paling mungkin menyebabkan penyelesaian permasalahan dapat diterima. Algoritma yang termasuk ke dalam metode ini diantaranya adalah algoritma Greedy-Best First Search dan algoritma A*.

III. PEMBAHASAN

III.1 Best First Search

Algoritma BFS membangkitkan simpul berikutnya dari sebuah simpul yang sejauh ini terbaik di antara semua leaf nodes yang pernah dibangkitkan. Penentuan simpul terbaik tersebut dilakukan dengan menggunakan informasi berupa biaya perkiraan dari suatu simpul menuju simpul tujuannya atau gabungan dari biaya sebenarnya dengan biaya perkiraan tersebut. Biaya perkiraan tersebut didapat dengan sebuah fungsi yang disebut fungsi heuristic. Terdapat dua jenis algoritma Best-First Search, yaitu :

Greedy-Best First Search yang hanya memperhitungkan

biaya perkiraan tersebut

A star (A*) yang memperkirakan biaya gabungan antara biaya sebenarnya dengan biaya perkiraan tersebut

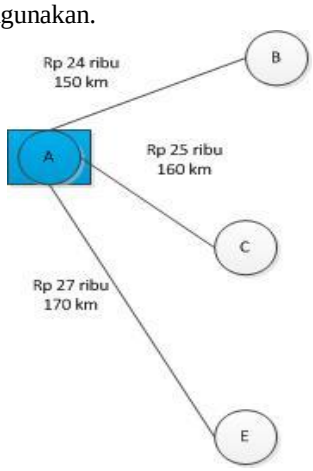
Berikut ini pseudocode untuk algoritma BFS

OPEN = [initial state]
CLOSED = []
While OPEN tidak kosong or goal tidak ditemukan do
i. Buang best node dari OPEN, sebut saja n
ii. apabila n adalah goal state, maka return path-nya
iii. apabila n bukan goal state, maka tambahkan ke dalam CLOSED
iv. bangkitkan semua suksesor dari simpul n
v. untuk setiap suksesornya, kerjakan :
i. apabila suksesor tersebut tidak ada di CLOSED (belum pernah dibangkitkan), evaluasi suksesor tersebut, dan catat 'parent'-nya
ii. jika suksesor tersebut ada dalam CLOSED, ubah 'parent'-nya jika jalur melalui parent ini lebih baik daripada jalur melalui 'parent' sebelumnya. Selanjutnya perbaharui biaya untuk suksesor tersebut dan nodes lain yang berada di level bawahnya

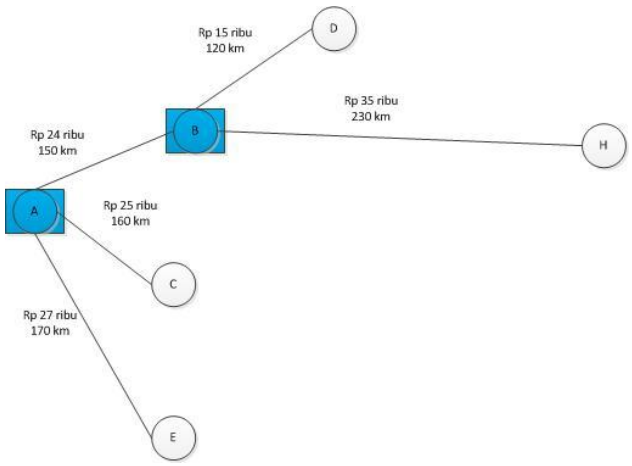
Pada pseudocode di atas, OPEN adalah list yang berisi semua simpul yang pernah dibangkitkan dan nilai heuristiknya telah dihitung tapi belum terpilih sebagai simpul terbaik. Sedangkan CLOSED juga merupakan sebuah list yang berisi simpul-simpul yang tidak mungkin terpilih lagi sebagai simpul terbaik

III.1.1 Greedy-Best First Search

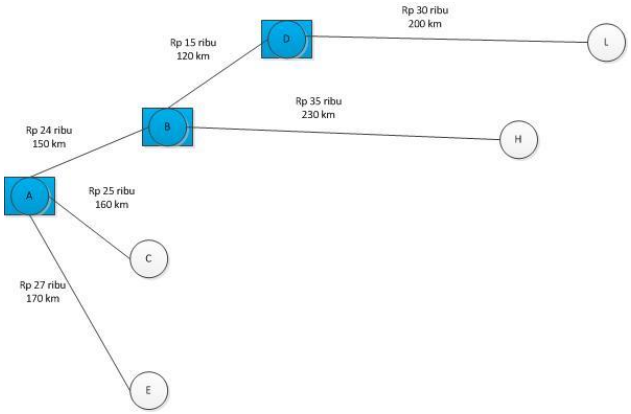
Algoritma greedy merupakan algoritma yang paling popular untuk memecahkan masalah optimasi. Algoritma ini membentuk solusi langkah per langkah. Pada setiap langkah, banyak pilihan yang perlu dieksplorasi. Oleh karena itu, pada setiap langkah harus dibuat keputusan terbaik dalam menentukan pilihan. Pada setiap langkah kita membuat pilihan optimum local dengan harapan bahwa langkah sisanya mengarah ke solusi optimum global. Algoritma Greedy-Best First Search memecahkan masalah optimasi dengan hanya mempertimbangkan harga perkiraan. Sedangkan harga sesungguhnya tidak digunakan.



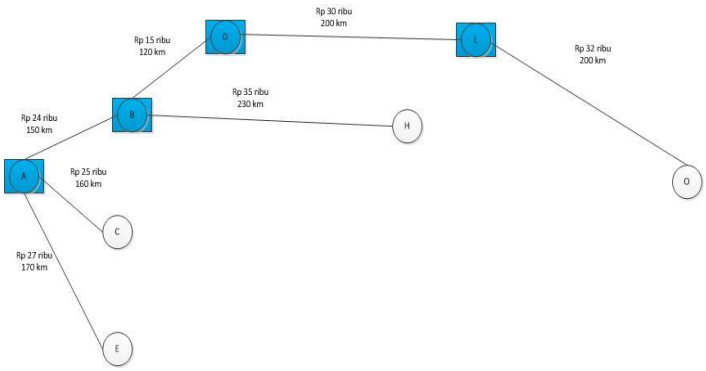
Gambar 2 Gambar langkah 1



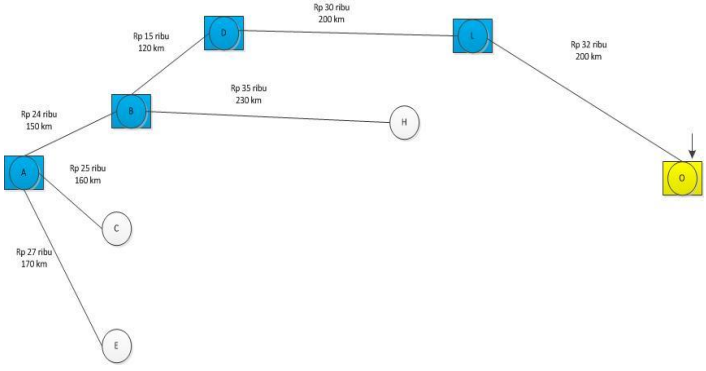
Gambar 3 Gambar langkah 2



Gambar 4 Gambar langkah 3



Gambar 5 Gambar langkah 4



Gambar 6 gambar langkah 5

Dalam graph seperti ini, dalam setiap langkah perlu dilakukan pengecekan berulang untuk menuju ke goal state sehingga mengakibatkan greedy menjadi tidak efektif. Untuk melakukannya dengan greedy, diperlukan strategy greedy :

Lintasan dibentuk satu per satu dari state awal sampai state akhir. Lintasan berikutnya adalah lintasan yang meminimumkan jumlah jaraknya. Berikut ini adalah rute yang dapat digunakan dari state awal sampai state akhir beserta biaya-nya

Rute Biaya
A>B>D>L>O Rp 102 ribu
A>B>H>K>O Rp 103 ribu
A>C>F>I>J>O Rp 104 ribu
A>C>F>I>M>O Rp 104 ribu
A>E>G>N>M>O Rp 106 ribu

Hasil di atas hanya menggunakan biaya saja, bukan biaya sebenarnya. Biaya sebenarnya yaitu empertimbangkan juga jarak antar state. Berdasarkan biaya perkiraan, rute dengan hasil terbaik adalah A>B>D>L>O. untuk mendapatkan biaya sebenarnya, akan digunakan pemodelan kasus serupa tetapi menggunakan algoritma SMA* yang merupakan variasi dari algoritma A*. Algoritma A* sendiri merupakan bagian dari algoritma Branch and Bound.

III.2 Simplified Memory-Bounded A (SMA*)*

Algoritma SMA* adalah sebuah algoritma yang dikembangkan dari algoritma A* (A Star). Algoritma A* sendiri adalah variasi dari algoritma Branch and Bound. Jadi secara tidak langsung, algoritma SMA* itu sendiri merupakan salah satu variasi pengembangan dari algoritma Branch and Bound.

Menggunakan algoritma SMA* berarti kita memperhitungkan biaya sebenarnya ditambah dengan biaya perkiraan. Biaya sebenarnya dinotasikan dengan g(n). sedangkan biaya perkiraan dinotasikan dengan h(n). jadi kita memperhitungkan g(n) + h(n).

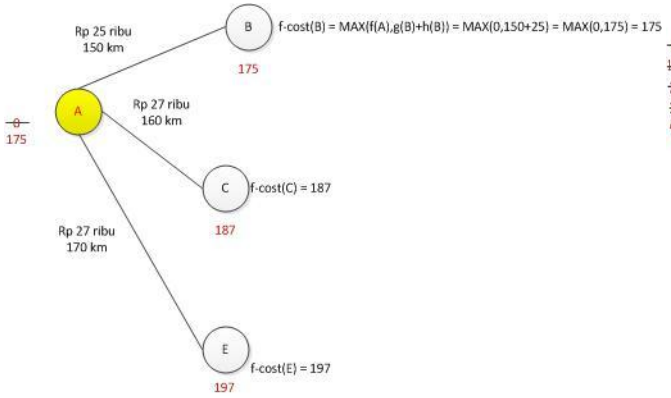
Dalam algoritma SMA* yang digunakan untuk mencair lintasan yang paling efisien ini, dibutukan sebuah Queue yang digunakan untuk memanipulasi antrian simpul yang terurut berdasarkan f-cost-nya. f-cost itu sendiri adalah g(n)+h(n).

Berikut ini adalah pseudocode algoritma A*

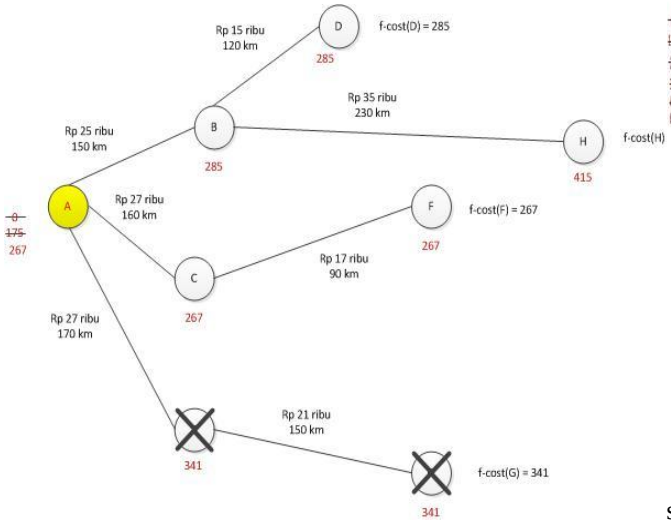
```
function A*(start,goal)
    closedset := the empty set                 // The set of nodes already evaluated.
    openset := set containing the initial node // The set of tentative nodes to be evaluated.
    came_from := the empty map                 // The map of navigated nodes.
    g_score[start] := 0                        // Distance from start along optimal path.
    h_score[start] := heuristic_estimate_of_distance(start, goal)
    f_score[start] := h_score[start]           // Estimated total distance from start to goal through y.
    while openset is not empty
        x := the node in openset having the lowest f_score[] value
        if x = goal
            return reconstruct_path(came_from, came_from[goal])
        remove x from openset
        add x to closedset
        foreach y in neighbor_nodes(x)
            if y in closedset
                continue
            tentative_g_score := g_score[x] + dist_between(x,y)

            if y not in openset
                add y to openset
                tentative_is_better := true
            elseif tentative_g_score < g_score[y]
                tentative_is_better := true
            else
                tentative_is_better := false
            if tentative_is_better = true
                if came_from[y] is set
                    if f_score[x] < f_score[came_from[y]]
                        came_from[y] := x
                else
                    came_from[y] := x

                g_score[y] := tentative_g_score
                h_score[y] := heuristic_estimate_of_distance(y, goal)
                f_score[y] := g_score[y] + h_score[y]
    return failure

function reconstruct_path(came_from, current_node)
    if came_from[current_node] is set
        p = reconstruct_path(came_from, came_from[current_node])
        return (p + current_node)
    else
    return current_node
```

Berikut ini adalah langkah-langkah penyelesaian masalah dengan menggunakan algoritma SMA*

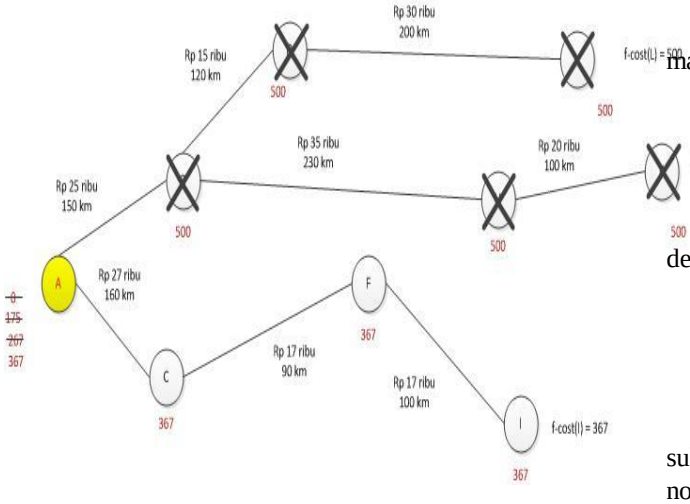
Gambar 7 Gambar langkah 1



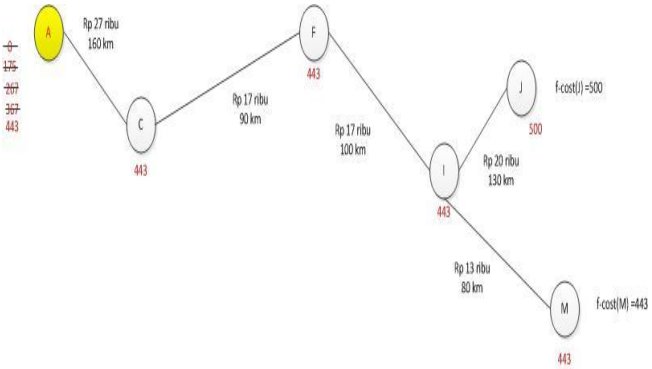
Gambar 8 Gambar langkah 2



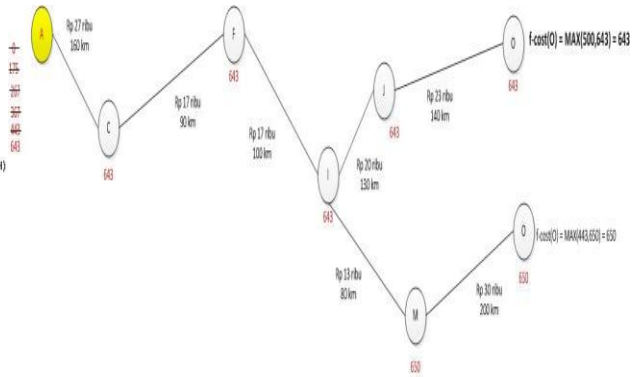
Gambar 9 Gambar langkah 3



Gambar 10 Gambar langkah 4



Gambar 11 Gambar langkah 5



Gambar 12 Gambar langkah 6

Pada langkah 0, nilai f(A) = 0

Pada langkah 1, dicari nilai dari setiap simpul dari suksesor (A), yaitu simpul B,C,E dan didapatkan hasil :

f-cost(B) = 175
f-cost(C) = 187
f-cost(E) = 197

Pada langkah 2, didapat nilai f-cost untuk masing-masing simpul sebagai berikut :

f-cost(D) = 285
f-cost(H) = 415
f-cost(F) = 267
f-cost(G) = 341

pada langkah ini, simpul G dan E dihapus karena sesuai dengan atursan SMA* no 2 dan 3 yang ada di atas.

Pada langkah 3, didapat nilai f-cost sebagai berikut :

f-cost(L) = 500
f-cost(K) = 500
f-cost(I) = 367

pada langkah ketiga ini, simpul B dan semua suksesornya dihapus, karena sesuai dengan aturan SMA* no 2 dan 3, sehingga kita tinggal menyisakan rute A>C>F>I

Pada langkah 4, didapat dua simpul suksesor dari simpul I dengan nilai f-cost masing-masing sebagai berikut :

f-cost(J) = 500
f-cost(M) = 443

Kemudian pada langkah terakhir, kita telah sampai simpul tujuan, dan didapatkan hasil yang minimum adalah

dengan nilai f-cost = 643 dan memiliki jalur A>C>F>I>J>O

IV. KESIMPULAN

Algoritma Simplified-Memory A* bias mengoptimalkan pencarian. Hal ini dikarenakan algoritma tersebut dapat menyelesaikan solusi dengan menggunakan solusi yang telah ada sebelumnya.

Sedangkan algoritma greedy tidak dapat memperhitungkan kebenaran biayanya. Secara optimality, algoritma greedy tidak selalu dapat menemukan solusi yang terbaik karena diperlukan pengecekan langkah-langkah algoritma yang berulang-ulang, sehingga tidak menghemat waktu untuk mendapatkan solusi yang terbaik dari beberapa solusi yang telah terbentuk.

Walaupun greedy cukup terkenal dan favorit untuk kasus pencarian optimasi, tetapi dalam kasus di atas, greedy dengan Greedy-Best First search-nya masih kurang efektif dibandingkan dengan algoritma Simplified-Memory A*.

Algoritma hanyalah algoritma, yaitu hanyalah sebuah proses yang ‘mungkin’ dapat menyelesaikan sebuah masalah. Seperti pengertian heuristik yang telah dijelaskan sebelumnya. Mungkin jika itu masalah perhitungan soal dapat diselesaikan. Tetapi jika itu masalah dunia nyata yang tidak akan lepas dari perubahan secara dinamis, solusi pun akan cepat bergantian.

Oleh karena itu, pengembangan dan modifikasi algoritma dalam beberapa kasus sangatlah penting dan patut untuk dicoba.

V. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa karena atas rahmat-Nya, penulis dapat menyelesaikan makalah ini. Terima kasih juga penulis ucapkan kepada Bapak Rinaldi Munir, selaku dosen Strategi Algoritma semester ganjil tahun ajaran 2010/2011 atas bimbingan yang diberikan selama ini. Makalah ini juga dapat terselesaikan berkat Diktat Kuliah IF3051 karangan Bapak Rinaldi Munir.

REFERENCES

[1] Munir, Rinaldi. *Diktat Kuliah IF3051 Strategi Algoritma*. 2009, Hal. 26–29.

[2] http://www. andraseptian.blog.upi.edu/files/2009/06/heuristic-searching.pdf
Tanggal Akses: 20 November 2010, 18:00 WIB

[3] http://www.id.wikipedia.org/
Tanggal Akses: 20 November 2010, 18:30 WIB.

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, 29 November 2010

Yongke Yoswara - 13508034