Pencarian Rute Pada Lalu Lintas Dengan Kemacetan Menggunakan Algoritma A*

Dandi Agus Maulana 13517077
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl Ganesha 10 Bandung 40132, Indonesia
13517077@std.stei.itb.ac.id

***Abstract*— Menavigasi rute paling efektif untuk menghadapi kemacetan sangat diperlukan untuk semua pengguna jalan. Metode yang berurusan dengan penemuan rute efektif dari kemacetan masih terlalu rumit. Kondisi jalan kapanpun selalu berubah seiring waktu. Untuk menavigasi rute terbaik, perhitungan selalu dilakukan berulang. Masalah itu mengharuskan algoritma yang efisien untuk menangani rute. Dalam tulisan ini, saya menawarkan pendekatan menggunakan algoritma A* dalam memecahkan masalah ini.**

Keywords—Algoritma, A*, rute, kemacetan.

I. PENDAHULUAN

Di kota-kota besar di dunia yang jumlah penduduk dan pengendaranya besar seperti Paris, Bombay, dan Jakarta, kemacetan merupakan hal yang selalu menjadi sumber permasalahan. Kemacetan yang terjadi biasanya disebabkan oleh jumlah kendaraan yang melebihi kapasitas. Orang-orang cenderung menggunakan kendaraan pribadi dibandingkan menggunakan kendaraan umum. Bahkan tak jarang kita melihat orang mengendarai mobil seorang diri tanpa membawa satu penumpang pun. Hal itu tentunya memakan banyak ruang di jalan raya dan merupakan hal yang sangat tidak efektif.

Kemacetan yang merupakan sebuah permasalahan ternyata melahirkan berbagai macam permasalahan lainnya. Mulai dari pemborosan waktu, stres, hingga polusi. Oleh karena itu pemerintah selalu mencoba berbagai solusi untuk melenyapkan masalah macet. Namun, langkah-langkah yang diambil oleh pemerintah seakan-akan tidak akan pernah cukup untuk menghentikan kemacetan. Jumlah kendaraan terus bertambah dan bertambah seiring perkembangan zaman.

Untuk menangani masalah tersebut, setidaknya ada hal yang dapat kita lakukan untuk membantu para pengguna jalan , yaitu dengan memberikan rute terbaik dalam kemacetan tersebut.

Algoritma pencarian jarak tercepat ataupun jarak terpendek adalah bidang penelitian algoritma yang menarik. Terdapat beberapa algoritma pencarian untuk menemukan solusi pencarian jarak tercepat, diantaranya adalah algoritma breadth first search, depth first search, best first search , A* , dan lain lain. Algoritma A* (Astar) adalah salah satu algoritma yang termasuk dalam kategori metode pencarian yang memiliki informasi (informed search method). Algoritma ini sangat baik sebagai solusi proses pathfinding (pencari jalan), mencari jarak rute tercepat yang akan ditempuh suatu poin awal (starting point) sampai ke objek tujuan sehingga cukup popular dikalangan pemrogram.

Algoritma A* menggunakan estimasi jarak terdekat untuk mencapai tujuan (goal) dan memiliki nilai heuristik yang digunakan sebagai dasar pertimbangan.

Penelitian ini bertujuan untuk mempelajari cara kerja algoritma A* dalam mencari jarak tercepat, yang disimulasikan dengan kondisi seorang mencari rute dalam keadaan jalanan macet. Simulasi ini memberikan gambaran yang lebih realistis terhadap perilaku algoritma A dalam pencarian jarak tercepat, dan untuk itu, akan dibangun sebuah aplikasi sebagai pendukung proses simulasi tersebut.

Penelitian ini diselesaikan dengan memperhatikan batasan-batasan sebagai berikut :

1. Teknik pencarian yang digunakan dalam simulasi ini adalah dengan menggunakan Algoritma A* dengan menggunakan fungsi heuristic manhattan distance
2. Rute jalan disimulasikan dengan bentuk mapgrid
3. Jarak tempuh jalan akan diasumsikan dalam bentuk banyaknya grid yang dilalui jalur

II. LANDASAN TEORI

2. 1. Definisi A*

Algoritma A* adalah salah satu algoritma pathfinding dengan jalur terpendek seperti algoritma Greedy Breadth First Search, Djikstra. Algoritma ini memecahkan masalah dengan mencari seluruh jalur yang mungkin dengan ongkos yang paling kecil (jarak paling pendek, waktu tercepat, dll.), dan dari jalur-jalur tersebut akan dipilih jalur yang paling cepat menemukan tujuan.

Algoritma A* digunakan pada graf berbobot, mulai dari sebuah simpul pada graf dan menuju ke simpul yang ingin dituju.

2. 2. Deskripsi Algoritma A*

Ide utama dari Algoritma A* adalah algoritma ini mencegah untuk menelusuri jalur yang sudah dapat dikatakan besar bobotnya. Karakteristik dari Algoritma A* ini adalah pembuatan sebuah list tertutup untuk mencatat simpul-simpul yang telah ditelusuri, dan list pinggir untuk mencatat simpul-simpul di sekitar simpul yang telah ditelusuri untuk nantinya akan menjadi kandidat simpul pilihan selanjutnya.

Karakteristik yang membedakan algoritma ini dengan algoritma lain adalah adanya perhitungan jarak yang telah dilalui dari simpul mulai dan jarak perkiraan ke simpul tujuan. Pada setiap iterasi, Algoritma A* akan berusaha untuk meminimalkan fungsi :

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

- g(n) : cost (ongkos) pada jalur dari simpul mulai ke simpul n.
- h(n) : perkiraan cost dari simpul n ke tujuan (heuristik)

Jika fungsi heuristik, h(n), selalu dapat menghasilkan ongkos paling minimal yang ada dari ongkos sesungguhnya (h(n) lebih kecil dibanding h*(n) yang berarti lebih kecil dari ongkos n ke tujuan yang sesungguhnya), maka A* dijamin akan menemukan sebuah solusi yang optimal yang admissible dan harus konsisten.

Sebuah heuristik dapat dikatakan admissible apabila untuk setiap simpul n, $h(n) \leq h^*(n)$ dimana h*(n) adalah ongkos sebenarnya untuk mencapai tujuan dari simpul n. Ciri-ciri lainnya dari algoritma A* adalah algoritma A* merupakan algoritma yang complete, yang berarti pasti akan ditemukan solusi apabila ada, kecuali ada tak hingga jumlah simpul yang ada pada graf.

Kompleksitas waktu algoritma A* dalam notasi Big-Oh adalah eksponensial, yaitu O(bm). Kompleksitas ruang pada algoritma A* adalah O(bm) karena algoritma A* menyimpan seluruh simpul pada memori. Algoritma A* dijamin optimal untuk semua kasus, apabila syarat $h(n) \leq h^*(n)$ terpenuhi. Oleh karena hasilnya yang hampir selalu optimal, algoritma ini sangat unggul dibandingkan dengan algoritma pathfinding lainnya seperti Greedy Breadth First Search dan Djikstra. Algoritma A* merupakan kombinasi yang mengambil kelebihan-kelebihan antara algoritma-algoritma lain.

2. 3. Langkah Algoritma A* Secara Umum

Langkah-langkah algoritma A* secara umum adalah sebagai berikut :

1. Memasukkan simpul akar ke dalam antrian Q. Jika simpul akar adalah simpul solusi, maka solusi telah ditemukan dan hentikan pencarian.
2. Jika Q kosong, maka tidak terdapat solusi dan hentikan pencarian.
3. Jika Q tidak kosong, maka pilih simpul i dari antrian Q yang mempunyai nilai f(n) paling kecil. Jika terdapat beberapa simpul n yang memenuhi, maka dipilih satu yang berada dalam antrian terlebih dahulu (atau juga dapat dipilih secara sembarang).
4. Jika simpul n adalah simpul solusi, berarti simpul solusi telah ditemukan, hentikan pencarian. Jika simpul n bukan sebuah simpul solusi, maka ekspansi semua simpul anak dari simpul n. Jika simpul n tidak memiliki simpul anak, maka kembali ke langkah 2.
5. Untuk setiap anak m pada simpul n, dihitung f(m), dan masukkan semua anak-anak tersebut ke dalam antrian Q terurut dari nilai terkecil.
6. Kembali ke langkah 2.

III. METODE

3. 1. Prinsp Pembuatan

Sistem simulasi yang akan dibangun berupa sebuah tampilan grid yang berbentuk maze . Di dalamnya akan ditentukan titik awal (start point) perjalanan, titik akhir/tujuan perjalanan setelah itu pencarian dengan algoritma A* diproses untuk menemukan rute. Setelah rute ditemukan akan dicoba untuk melakukan pencarian lagi tetapi dengan memberikan beban atau penghalang disalah satu grid yang tadi rutenya sudah ditemukan untuk melihat pencarian baru. Penghalang ini merupakan asumsi dari kemacetan yang terjadi di jalan. Setelah dilakukan pencarian maka rute terbaik akan ditemukan.

Simulasi ini akan digunakan untuk menentukan pencarian rute (path finding) yang dibentuk dengan

model mapgrid yang diasumsikan grid tersebut sebagai sebuah jalan, dengan asumsi tersebut dapat dilihat apakah dengan melewati kemacetan ataupun menghindari kemacetan dengan mencari rute terbaru merupakan hal yang tepat untuk menghemat waktu perjalanan. Sistem Algoritma A* akan mencari solusi yang terbaik.

3. 2. Kebutuhan Data

Dalam pembuatan simulasi ini digunakan algoritma A* dengan fungsi heuristik, oleh karena itu ada beberapa data-data yang diperlukan :

1. Titik awal, ditentukan oleh pengguna dengan memilih titik koordinat yang akan dijadikan titik awal. Titik awal ini akan diberikan tanda berupa warna hijau
2. Grid adalah petak-petak kecil yang di representasi dari area path finding yang berupa persegi panjang. Grid yang ada disimulasi ini 20x20 = 400 grid.
3. Titik akhir, ditentukan oleh pengguna dengan memilih titik koordinat yang akan dijadikan titik akhir. Titik akhir ini akan diberikan tanda berupa warna hijau.
4. Harga fungsi F, nilai yang diperoleh dari penjumlahan nilai G (movement cost) dan nilai h (heuristic). Dikarenakan simulasi ini berbentuk gridcell maka nilai jarak tempuh per grid (per koordinat) memiliki nilai 1.
5. Nilai hambatan kemacetan (penghalang), diberikan dengan memilih titik koordinat yang akan dijadikan titik kemacetan. Untuk grid yang macet koordinatnya yang dipilih akan diberikan tanda berupa warna merah
6. Pembatas jalan, merupakan batas jalan kiri dan kanan yang tidak dapat dilalui. Batas jalan ini diberi sebuah tanda berupa warna hitam.

3. 3. Prinsip A*

Prinsip algoritma ini adalah mencari jalur terpendek dari sebuah titik awal menuju titik akhir dengan memperhatikan harga F terkecil. Algoritma ini memperhitungkan nilai dari current state ke tujuan dengan fungsi heuristik, dan juga mempertimbangkan nilai yang telah ditempuh selama ini dari initial state ke current state. Jadi jika ada jalan yang telah ditempuh sudah terlalu panjang dan ada jalan lain yang nilainya lebih kecil tetapi memberikan posisi yang sama dilihat dari goal, jalan yang lebih pendek yang akan dipilih.

Rumus pencarian F (n) = h(n) + g (n) dimana : g (n) adalah movecost dari titik awal , dikarenakan simulasi berbentuk grid persegi , tiap koordinat antara titik koordinat berikutnya sama bernilai satu. Lalu h(n) dapat dicari dengan rumus sebagai berikut :

```
function heuristic(node) :
    dx = abs(node.x - goal.x)
    dy = abs(node.y - goal.y)
    return D * (dx + dy)
```

Rumus ini digunakan dikarenakan pengembangan aplikasi simulasi ini menggunakan fungsi heuristic manhattan distance.

IV. IMPLEMENTASI

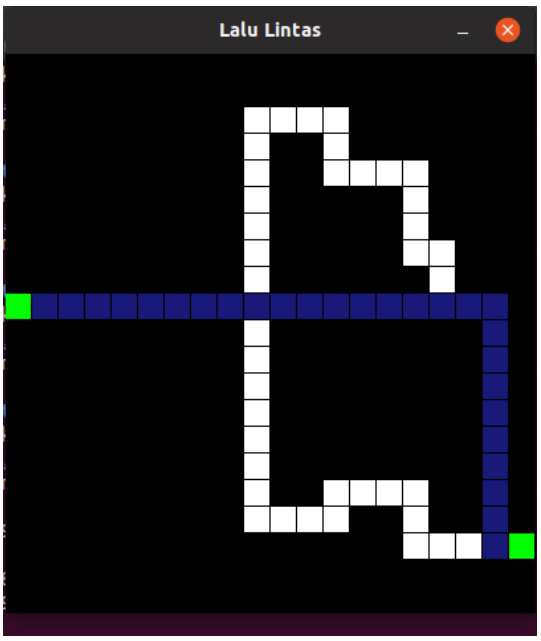
Pada simulasi ini, algoritma A* diimplementasikan pada saat pengguna melakukan pencarian rute. Deskripsi di bawah ini menjelaskan implementasi algoritma A* yang diterapkan pada simulasi ini.

Algoritma :

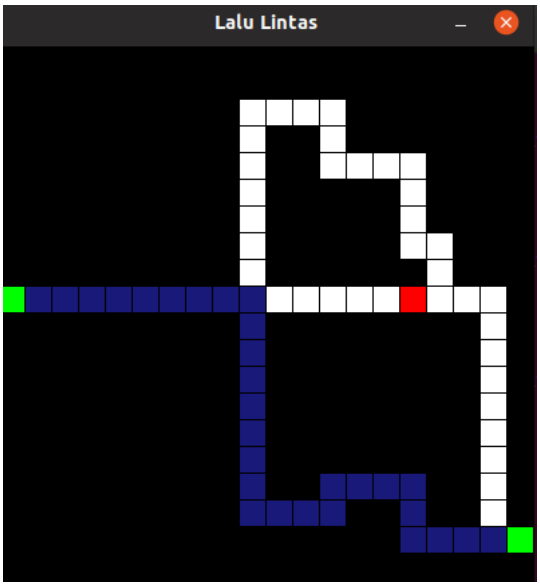
```
Function heuristic (awal, akhir , D) :
        dx = abs (nilai x awal – nilai x akhir)
        dy = abs (nilai y awal - niai y akhir )
        return nilai D * (dx + dy )
EndFunction

Function FindPath :
        For i to total x grid
                For j to total y grid
                minCost = Math.min (nilai gridcell [j][i],
                minCost)
                        increment j
                        increment i
                end for
        end for
EndFunction

Function Langkah :
        For i temp
                Set now gridcell temp elementAt (i)
                If now sudah terisi
                        score = ambil nilai now dari start
                score = score + Function heuristic (posisi
                sekarang (), posisi akhir , minCost)

                If score kurang dari min
                        Min = score
                        Best = now Endif
                Endif
                Increment i
        Endfor
        Now = best
        Edge.removeElement(now)
        Done.AddElement(now)
        Set Gridcell next [] = map.getAdjacent(now)
        For i to 4
                If next [i] != null
                        If next [i] == finish
                                Ditemukan
```

```
                    Endif
                    If next [i] totalblock
                        Add next[i] to path from start
                        If!edge.contains (next[i]) &&
                      !done.contains(next[i])
                            Add element next[i] ;
                           grownt =true;
                      Endif
                    Endif
 Endif
If ditemukan
        Call ditemukan (PATH FOUND)
Endif

Urutkan secara backward dari finish (tujuan)
ke titik semula (start) lalu beri tanda   berupa
warna
If edgesize bernilai 0
        Call tidak ditemukan (No_Found)
EndFunction
```

Gambar 1 adalah tampilan aplikasi simulasi pencarian rute



1.a Tanpa Penghalang



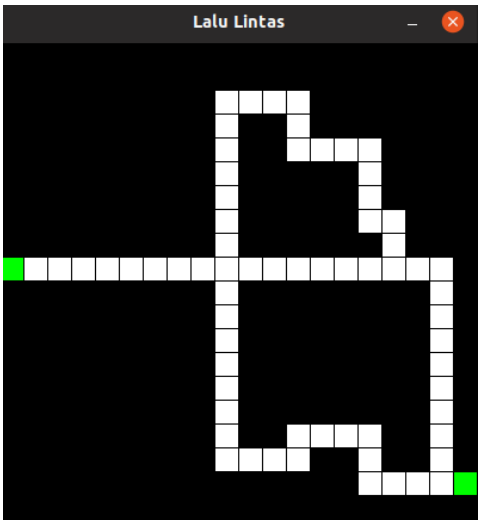
1.b Dengan penghalang

Gambar 1. Tampilan pencarian rute

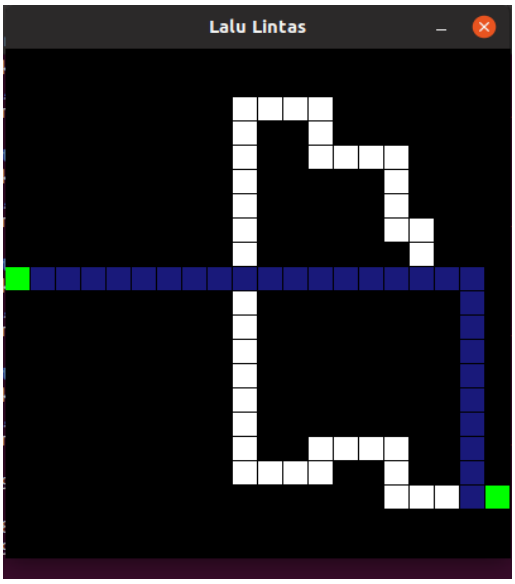
V. PERCOBAAN

5. 1. Pengujian

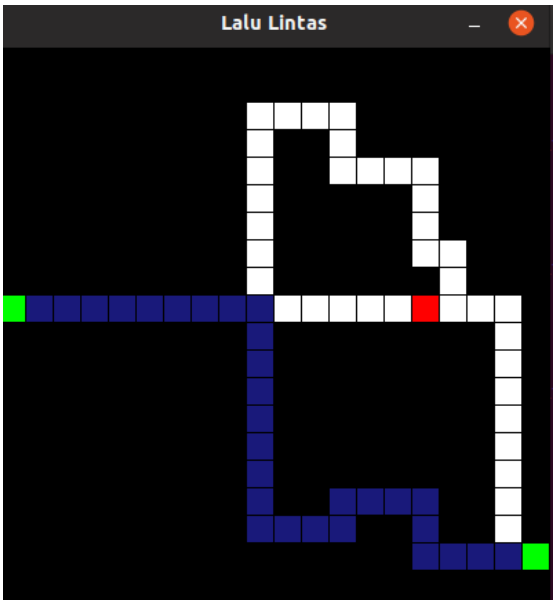
Setelah Grid ditampilkan kemudian dilakukan pengaturan titik awal (Set Start) yang dalam pengujian ini posisi titik awal (start) berada di point (x=0,y=9). Setelah itu tentukan titik akhir (Set Finish) yang dalam pengujian ini posisi titik akhir berada pada point (x=18,y=19).Kemudian dilakukan pencarian. Setelah ditemukan pencarian berikutnya dalam jalur pencarian tersebut diberi hambatan kemacetan parah di titik (x=9,y=15).



2.a Tampilan awal



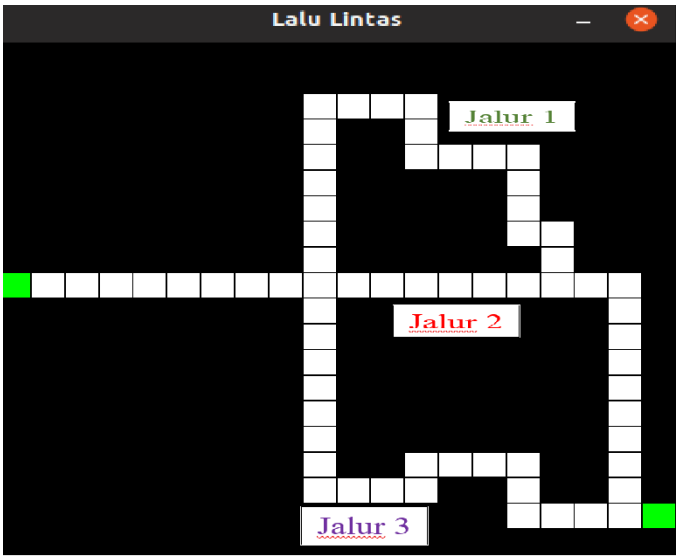
2.b Tanpa Penghalang



2.c Dengan penghalang

5. 2. Pembuktian

Berikutnya dilakukan pembuktian perhitungan dari hasil yang didapat. Rute yang akan dibuktikan adalah hasil pencarian rute dengan adanya hambatan kemacetan yang terlihat pada Gambar 2. Pembuktian hasil pencarian rute akan dilakukan dengan perhitungan yang dilakukan secara manual.



Gambr 3. Kemungkinan jalur-jalaur tanpa kemacetan

NO	X	Y	G(n)	H (n)	F(n)
1	1	9	1	27	28
2	2	9	2	26	28
3	3	9	3	25	28
..	..	..	..	..	..
27	17	19	27	1	28
28	18	19	28	0	28

Tabel 1. Perhitungan jalur 2

NO	X	Y	G(n)	H (n)	F(n)
1	1	9	1	27	28
2	2	9	2	26	28
3	3	9	3	25	28
..	..	..	..	..	..
29	17	19	29	1	30
39	18	19	30	0	30

Tabel 2. Perhitungan jalur 3

NO	X	Y	G(n)	H (n)	F(n)
1	1	9	1	27	28
2	2	9	2	26	28
3	3	9	3	25	28
..	..	..	..	..	..
15	15	9	15	13 +10	38
27	17	19	27	1 +10	38
28	18	19	28	0 + 10	38

Tabel 3. Perhitungan jalur 2 dengan kemacetan

Dari tabel hasil perhitungan manual didapat bahwa nilai f(n) terbaik terdapat pada setiap cell yang dilalui oleh jalur 2 yaitu bernilai 28 sedangkan jalur 3 bernilai 30. Dengan demikian jalur terbaik terdapat pada jalur 2, karena memiliki nilai f(n) yang kecil daripada jalur-jalur lainnya Kemudian dilakukan pembuktian secara manual untuk melihat nilai dari jalur, tetapi kali ini jalur yang terbaik yang sudah dibuktikan sebelumnya diberi beban (penghalang kemacetan) dengan beban 10. Dari tabel pembuktian 3 dapat dilihat nilai f(n) bertambah menjadi 38. Dengan demikian jalur terbaik menjadi jalur 3 yang bernilai 30.

Dari pembuktian dan pengujian yang dilakukan diatas dapat dilihat bahwa simulasi dapat menentukan jalur terbaik yang akan dilalui . Dengan ini simulasi yang menerapkan Algoritma A* dengan fungsi heuristic manhattan distance dapat menentukan rute (jalur) terbaik dengan membandingkan nilai f(n) terkecil.

VI. KESIMPULAN

Algoritma A* merupakan algoritma pathfinding yang sering digunakan untuk mencari jarak terdekat suatu titik ke titik lainnya. Algoritma A* telah terbukti efektif dapat menyelesaikan salah satu permasalahan di bidang transportasi, yaitu masalah kemacetan.

Simulasi ini dapat menentukan rute (jalur) terbaik dari titik awal (start) menuju titik akhir (finish) dengan hambatan-hambatan yang diberikan disetiap rute. Dari hasil pengujian , rute yang ditemukan merupakan rute yang terbaik dengan nilai f(n) terkecil dibandingkan dengan rute-rute (jalur-jalur) lainnya.

VII. UCAPAN TERIMA KASIH

Terima kasih kepada Tuhan Yang Maha Esa, karena atas karunia-Nyalah makalah berjudul “Pencarian rute pada lalu lintas dengan kemacetan menggunakan algoritma A* “ ini dapat diselesaikan dengan baik. Penulis mengucapkan terima kasih kepada bapak Dr. Ir. Rinaldi Munir MR, ibu Masayu Leyla Khodra ST., MT., dan ibu Dr.Nur Ulfa Maulidevi atas bimbingan dalam mengajarkan saya mata kuliah IF2211 Strategi Algoritma. Tak lupa penulis juga mengucapkan terima kasih kepada keluarga dan rekan-rekan penulis atas bantuannya dalam pembuatan makalah ini.

REFERENSI

[1] Munir, Rinaldi. Diktat Kuliah Strategi Algoritma
[2] Heuristics
http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html
[3] A* Search | Brilliant Math & Science Wiki
https://brilliant.org/wiki/a-star-search/
[4] Introduction to A*
http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison. html
[5] Munir, Rinaldi, 2004, Diktat Strategi Algoritmik. Bandung: Departemen Teknik Informatika Institut Teknologi Bandung

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 2019

Dandi Agus Maulana
13517077