SNARTISI
Seminar Nasional Rekayasa Teknologi Informasi
02 – 03 November 2018
Fakultas Teknologi Informasi, Universitas Serang Raya

Pencarian Jalur Terdekat dengan Metode A*(Star) Studi Kasus Serang Labuan Provinsi Banten

Robby Rizky
[1)]Jurusan Sistem Informasi, Fakultas Informatika dan Komputer,Universitas Mathla'ul Anwar Banten
Jl. Raya Labuan Km 23 cikaliung - saketi 42273 Pandeglang
E-mail: robby.rizky@unmabanten.ac.id, robby_bae87@yahoo.com

ABSTRAKS

*Banten adalah provinsi yang bisa di bilang baru di bandingkan provinsi provinsi lainya, namun banten memiliki beberapa tempat tempat wisata yang menarik dan patut di kunjungi , banten pun memiliki beberapa tempat industri seperti indahkiat,nikomas di daerah banten ini ada beberpa titik tempat kemacetan seperti pasar baros dan pasar pandeglang, jika melintasi tempat ini di hari libur seperti sabtu dan minggu macet hingga berkilo kilo meter yang tadinya dapat di tempuh labuan serang hanya satu jam setengah, jika mengalami kemacetan bisa sampai 2 sampai 2 jam setengah,daerah banten memiliki jalan alternatif maka dari itu permasalahan tersebut dapat di pecahkan dengan metode A*STAR yang dapat membaca jalur terdekat. Algoritma A* (Astar) merupakan salah satu algoritma yang termasuk dalam kategori metode pencarian yang memiliki informasi (informed search method). Algortima ini sangat baik sebagai solusi proses pathfinding (pencari jalan). Algoritma ini mencari jarak rute tercepat yang akan ditempuh suatu point awal (starting point) sampai ke objek tujuan. Teknik pencarian yang digunakan dalam simulasi ini adalah menggunakan Algoritma A* dengan fungsi heuristic manhattan distance. Simulasi dilakukan dengan bahasa pemrograman Java. Tujuan utama penelitian ini mempelajari cara kerja algoritma A* dalam mencari jarak tercepat, yang disimulasikan seperti kondisi ketika 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.*

Kata Kunci: *Algoritma, A-star, simulasi, rute*

1. PENDAHULUAN

1.1 Latar Belakang

Banten memiliki beberapa tempat wisata yang sering di kunjungi oleh wisatawan dari pelosok kota bahkan mancanegara, daerah yang sering di kunjungi adalah daerah pesisir pantai yang berlokasi di kecamatan labuan kabupaten pandeglang, pantai carita ini adalah objek wisata yang sangat di gemari oleh wisatawan di karnakan akses masuk dengan biaya yang sangat murah dan memiliki panorama pantai yang sangat indah, banyak jalan menuju pantai carita/labuan tetapi jalan utama menuju labuan ini jaraknya jauh dan macet karna memiliki titik titik kemacetan melintasi pasar pasar besar seperti pasar baros dan pasar pandeglang apalagi menjelang hari raya idul fitri tamu dan wisatawan pasti memburu tempat pantai carita sebagai tempat wisata yang sangat ideal, di karnakan tujuan para wisata bertujuan ke pantai carita maka jalan utama dan kepadatan akan semakin meningkat. Maka dari itu peneliti ingin memecahkan permasalahan yang ada dengan menggunakan metode A* star algoritma A*star Dengan menerapkan suatu heuristik. Heuristik adalah nilai yang memberi nilai pada tiap simpul yang memandu A* mendapatkan solusi yang diinginkan. Dengan heuristik, maka A* pasti akan mendapatkan solusi (jika memang ada solusinya). Dengan kata lain, heuristik adalah fungsi optimasi yang menjadikan algoritma A* lebih baik dari pada algoritma lainnya.

1.2 Rumusan Masalah

Berdasarkan latar belakang penelitian jalur serang labuan telah dilakukan observasi dan pengidentifikasian masalah, adapun masalah-masalah yaitu :

a. Bangaimana cara kerja algoritma A* star?
b. Bagaimana menerapkan rumus heurustic pada metode A* agar dapat memecahkan permasalahan yang ada?

ISBN : 978-602-53437-0-4

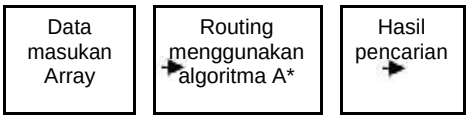
1.3 Tujuan

Tujuan dilaksanakan penelitian ini adalah:

a. Sebagai upaya menghindari jalur utama yang macet.
b. Sebagai sarana mempermudah dalam memilih jalur berkendara yang terdekat.

1.4 Rancangan Penelitian

Blok Diagram Keseluruhan Sistem



Gambar 1. Blok Diagram Keseluruhan Sistem

Blok diagram keseluruhan sistem dari penelitian ini seperti pada Gambar 1, dimana data masukan berupa array yang akan visualisasikan terlebih dahulu menjadi labirin yang selanjunya akan di hitung langkahnya sampai menemukan garis finish.

Algoritma A* menyelesaikan masalah yang menggunakan graf untuk perluasan ruang statusnya.

Dengan menerapkan suatu heuristik. Heuristik adalah nilai yang memberi nilai pada tiap simpul yang memandu A* mendapatkan solusi yang diinginkan. Dengan heuristik, maka A* pasti akan mendapatkan solusi (jika memang ada solusinya). Dengan kata lain, heuristik adalah fungsi optimasi yang menjadikan algoritma A* lebih baik dari pada algoritma lainnya.

Prinsip Algoritma 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* , dikarenakan simulasi berbentuk *grid* persegi , tiap koordinat antara titik koordinatberikutnya 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 *heuristicmanhattan distance*.

Algoritma A* secara garis besar dapat dijelaskan seperti berikut:

1. Masukkan node awal ke *openlist*
2. *Loop* langkah-langkah dibawah ini:
   a. Cari node (n) dengan nilai f(n) yang paling kecil dalam *open list*, dan *node* ini sekarang menjadi *current node*.
   b. Keluarkan *current node* dari *openlist* dan masukkan ke *open list*
   c. Untuk setiap tetangga dari *current node* lakukan berikut
      1) Jika tidak dapat dilalui atau sudah ada dalam *close list* , abaikan
      2) Jika belum ada di *open list* buat current *node parent* dari *node* tetangga ini, simpannilai f,g, dan h dari node ini.
      3) Jika sudah ada di *openlist* cek apabila *node* tetangga ini lebih baik menggunakan nilai g sebagai ukuran. Jika lebih baik ganti *parent* dari *node* ini di *opentlist* menjadi *current node*, lalu kalkulasikan ulang nilai g dan fdari *node* ini.
   d. Hentikan *looping* jika:
      1) *Node* tujuan telah ditambah ke *openlist* yang berarti rute ditemukan
      2) Belum menemukan *node* akhir (tujuan) sementara *openlist* kosong atau berarti tidak ada rute.

Simpan rute, lalu secara *backward* urut mulai dari *node* akhir (tujuan) sampai ke titik awal sambil menyimpan *node* ke dalam *array*.

2. PEMBAHASAN

2.1 Implementasi

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

Proses *Algoritma A**

{*Start point* dan *finish point* sudah dalam keadaan di *set*}

Input: *Set start , Set Finish*
Output: *Path Found or No Found*

Algoritma :
Function heuristic (awal, akhir , D)=

ISBN : 978-602-53437-0-4

```
      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

       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
```

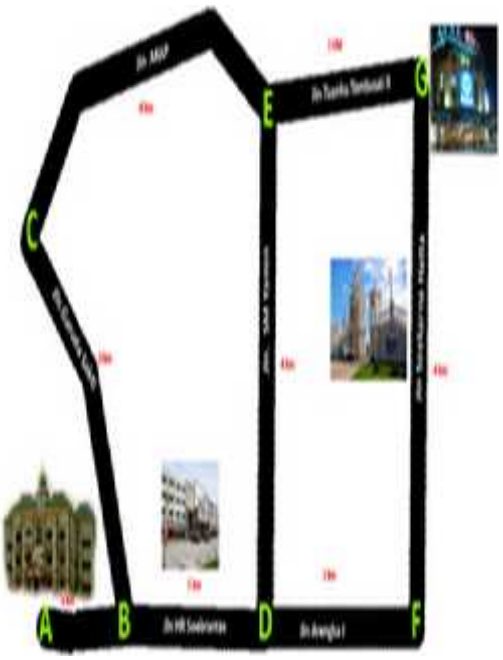
2.2 Pengujian Fungsi Pencarian Rute
Pengujian

Dalam pengujian ini akan dilakukan pegujian fungsi pencarian dengan pembuktian dari hasil pencarian rute. Dalam pembuktian ini dibuat sebuah jalur lalu lintas yang berbentuk grid yang akan dijalankan pada simulasi. Pertama kali dilakukan adalah mengambil (*load) mapgrid* yang sudah di *design* sebelumnya.

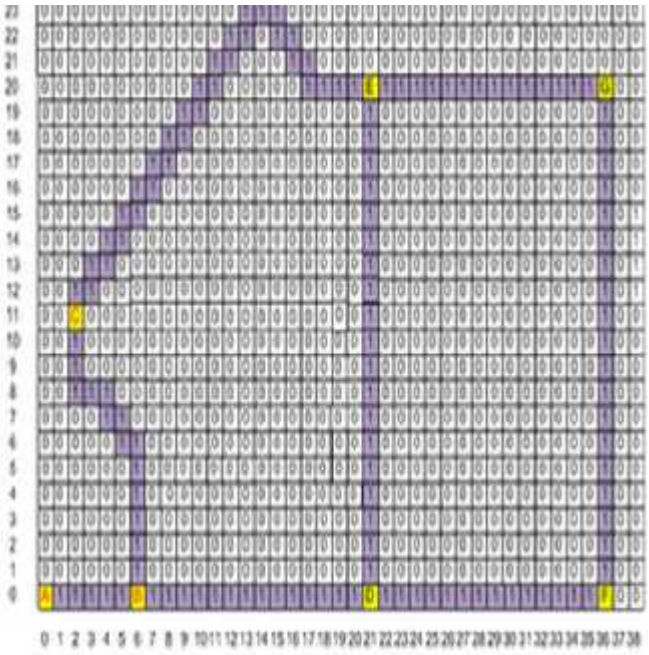
Setelah *MapGrid* ditampilkan kemudian dilakukan pengaturan titik awal (*Set Start*) yang dalam pengujian ini posisi titik awal (*start*) berada di salahsatu point Setelah itu tentukan titik akhir (*Set Finish*) yang dalam pengujian ini posisi titik akhir berada pada salahsatu point Kemudian dilakukan pencarian. Setelah ditemukan pencarian berikutnya dalam jalur pencarian.

Hasil dari proyeksi gambar *Google Earth* dapat dilihat sbb:

ISBN : 978-602-53437-0-4



A = SERANG
B = PALIMA
C = BAROS
D = PETIR
E = PANDEGLANG
F = CADASARI
G = LABUAN



Setiap Index mewakili jarak 200 meter dan perhitungan angka mewakili titik di atas.
A = SERANG (0,0)
B = PALIMA
(6,0)
C = BAROS
(2,11)
D = PETIR
(21,0)
E = PANDEGLANG
(21,20)
F = CADASARI
(36,0)
G = LABUAN
(36,20)

Menghitung:

Rumus jarak dua titik: $d(x,y) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

A (0,0) ke B (6,0)

$$d(x,y) = \sqrt{(0 \quad 6)^2 \; | \; (0 \quad 0)^2} = \sqrt{36} = 6$$

B (6,0) ke C (2,11)

$$d(x,y) = \sqrt{(6-2)^2 + (0-11)^2} = \sqrt{137} = 11.7$$

B (6,0) ke D(21,0)

$$d(x,y) = \sqrt{(6-21)^2 + (0-0)^2} = \sqrt{225} = 15$$

ISBN : 978-602-53437-0-4

C (2,11) ke E (21,20)

$$d(x,y) = \sqrt{(2-21)^2 + (11-20)^2} = \sqrt{442} = 21.$$

D (21,0) ke E (21,20)

$$d(x,y) = \sqrt{(21-21)^2 + (0-20)^2} = \sqrt{400} = 20$$

D (21,0) ke F(36,0)

$$d(x,y) = \sqrt{(21-36)^2 + (0-0)^2} = \sqrt{225} = 15$$

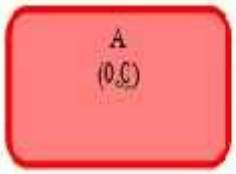
E (21,20) ke G (36,20)

$$d(x,y) = \sqrt{(21-36)^2 + (20-20)^2} = \sqrt{225} = 15$$

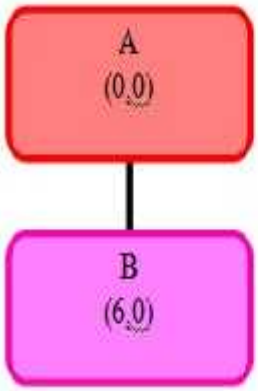
F(36,0) ke G (36,20)

$$d(x,y) = \sqrt{(36-36)^2 + (0-20)^2} = \sqrt{400} = 20$$

Langkah I



Langkah II



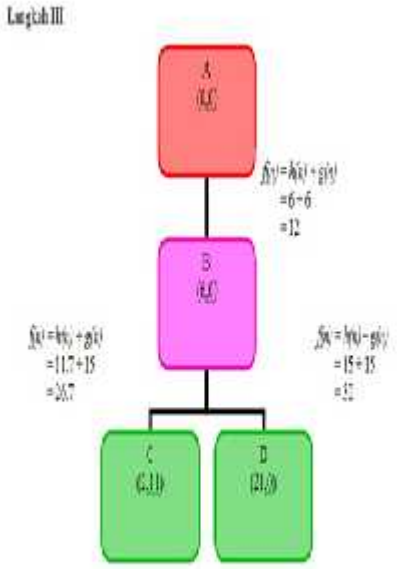
$f(n)=h(n)+g(n)$
$=6+6$
$=12$

2.3 Langkah-Langkah Pencarian Dalam Algoritma A*

Setelah nilai heuristik dari masing-masing node didapat maka kita akan mencari *f(n)* menggunakan algoritma A* dengan rumus:

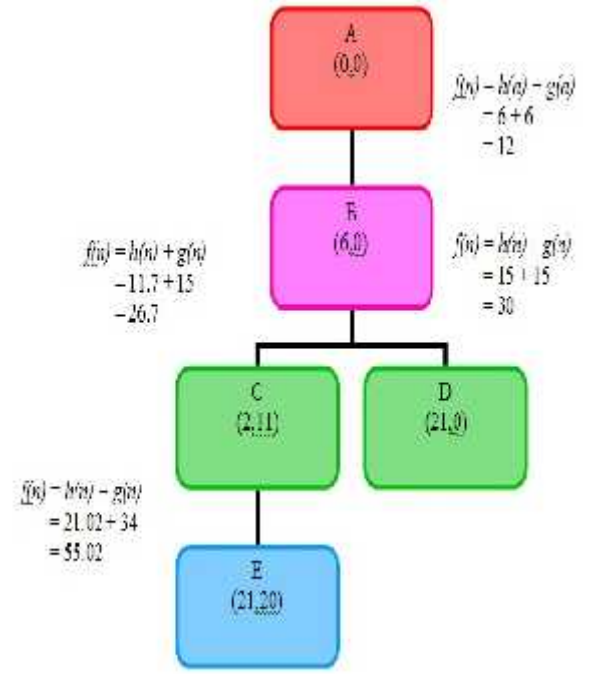
$$f(n) = h(n) + g(n)$$

Dimana:
h(n) = Nilai heuristik antar Koordinat
g(n) = Jarak Koordinat ke titik tujuan

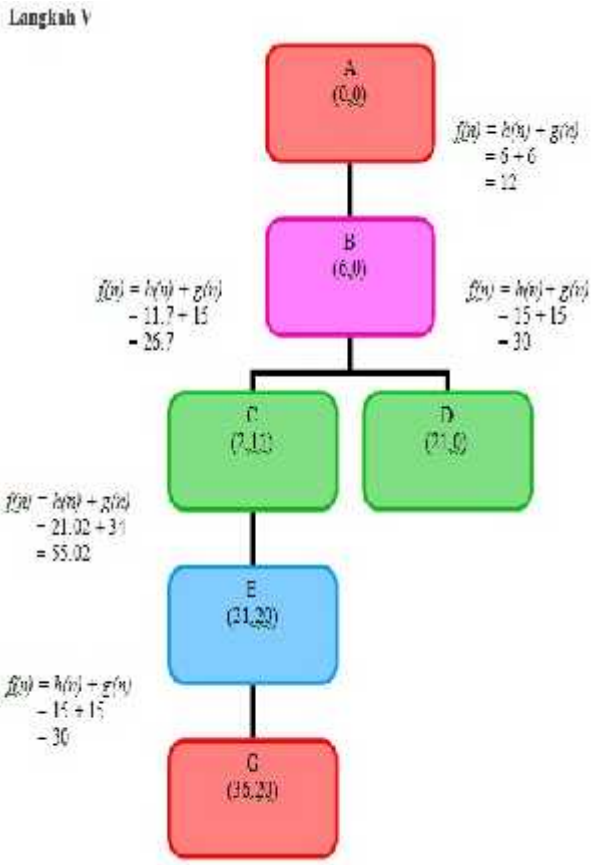


ISBN : 978-602-53437-0-4

Titik B memiliki 2 cabangan yaitu titik C dan titik D, maka *f(n)* yang harus dipilih adalah *f(n)* yang menghasilkan biaya paling kecil, yaitu titik C.



Langkah V



- Maka *f(n)* total yang didapat adalah 123.72, karena satu titik ordinat mewakili 200 meter maka jaraknya sebenarnya (dalam meter) adalah:
- 123.72 × 200 = 24744 meter,
  Dalam Kilometer = 24,744 km
- Jalur yang dilalui:
  A – B – C – E – G
- SERANG-PALIMA-BAROS-PANDEGLANG-LABUAN

3. KESIMPULAN

Berdasarkan hasil simulasi algoritma A* pada penelitian ini dapat disimpulkan hal-hal berikut:

1. Algoritma A* dapat diterapkan sebagai algoritma untuk menentukan *rute* (jalur) terbaik yang akan dilalui.
2. 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.
3. Untuk pengembangan lebih lanjut disarankan menggunakan algoritma lain selain algoritma A* untuk menentukan jalur (rute) yang terbaik. Dan juga dapat membandingkan algoritma lain tersebut apakah lebih baik dalam penentuan jalur tercepat

PUSTAKA

Brourg, David. Seeman,Glenn. (2004). *AI For GameDevelopers*.O'Reilly Inc : United States of America.

Madhav, Sanjay. *Game ProgrammingAlgorithms andTechniques*.Addison Wesley : United State.

Netbeans (2013). Documentation,Training,&Support.<Netbeans .org>"A* search algorithm". Wikipedia . Web . 22 May 2014.

http://en.wikipedia.org/wiki/A*_search_algorithm" Amit's Thoughts on Pathfinding. Web . 31 May 2014.

http://theory.stanford.edu/~amitp/Game Programming/AStarComparison

Eranki,Ranjiv. *Pathfinding using A* (A-Star)* . 2002 . Web. 31 May 2014. http://web.mit.edu/eranki/www/ "Simulation". Wikipedia . Web . 24 April 2014.

http://en.wikipedia.org/wiki/Simulation

Norvig,Peter. Russell,Stuart. *Artificial Intelligence AModern Approach Second Edition*. Prentice Hall :United State