Login : Jurnal Teknologi Komputer
ISSN : 2302-9692 (print) | 2723-8695 (online)
Vol. 17, No. 1, June 2023, pp. 23-32
http://login.seaninstitute.org/index.php/Login

Simulasi Pencarian Rute Terpendek dengan Metode Algoritma A* (A-Star)

Oslan Juliana Simbolon
Kepala Tata Usaha, Fakultas Ilmu Komputer, Universitas Katolik Santo Thomas, Medan, Indonesia

Article Info

Keywords:

Rute Terpendek,
Simpul(node),
A*(A-Star),
Heuristik.

ABSTRACT

Pencarian rute terpendek merupakan suatu permasalahan yang sering muncul pada pengguna sarana transportasi, karena para pengguna sarana transportasi dalam melakukan perjalanan memerlukan solusi untuk mendapatkan rute atau jalur tempuh terpendek. Hal ini eratkaitannya dengan efisiensi waktu, biaya, serta tenaga yang dikeluarkan. Algoritma A* menggunakan estimasi jarak terdekat untuk mencapai tujuan (goal) dan memiliki nilai heuristik yang digunakan sebagai dasar pertimbangan. Heuristik adalah kriteria, metoda, atau prinsip-prinsip untuk menentukan pilihan sejumlah alternatif untuk mencapai sasaran dengan efektif. Hasil pada penelitian iniadalah berupa realisasi program simulasi pencarian rute terpendek dari posisi asal ke posisi yang dituju (goal) dengan menggunakan bahasa pemrograman Visual 2008. Representasi visual dari Graf adalahdengan menyatakan obyek sebagai simpul, dan hubungan antara simpul dinyatakan dengan Titik-titik. Program simulasi ini memberikan kemudahan untuk menentukan rute terpendek yanga kan dilalui dari posisi asal ke tujuan.

Corresponding Author:

Oslan Juliana Simbolon,
Kepala Tata Usaha, Fakultas Ilmu Komputer, Universitas Katolik Santo Thomas, Medan, Indonesia,
Jl. Setia Budi No.479-F Medan Sumatera Utara.
Email: oslan.juliana.sbln@gmail.com

1. INTRODUCTION

Persoalan jalur terpendek (Shortest Path) merupakan suatu jaringan pengarahan perjalanan dimana seseorang pengarah jalan ingin menentukan jalur terpendek antara dua kotak yang tersedia yaitu titik awal dan titik tujuan, dimana kota tujuan hanya satu. Masalah ini sendiri menggunakan representasi graph untuk memodelkan persoalan yang diwakili sehingga lebih memudahkan penyelesaiannya. Masalahnya adalah bagaimana cara mengunjungi vertek pada graph dari vertek awal ke vertek akhir dengan bobot minimum, dimana dalam hal ini bobot yang digunakan adalah jarak dan kotak-kotak yang dikunjungi diasumsikan sebagai graph yang saling terhubung (connected graph) antar suatu kota dengan kota yang lainnya. Suatu graph G disebut terhubung jika untuk setiap vertek dari graph terdapat jalur yang menghubungkan kedua verteks tersebut, atau dengan kata lain graph terhubung jikkra setiap dua vertek yaitu vi dan vj dalam suatu graph terdapat sedikitnya sebuah edge. edge pada graph berarah disebut arc.

Simulasi adalah model dari suatu sistem yang nyata, melakukan pengolahan data terhadap model tersebut dan mengevaluasi hasil simulasi tersebut. Simulasi juga merupakan teknik untuk menyelesaikan persoalan melalui pengolahan data yang nyata untuk memperoleh output berupa solusi persoalan ataupun sebagai bahan masukan dalam rangka pengembangan dan perbaikan struktur dan operasi sistem yang nyata.

Algoritma A* adalah suatu metode yang sudah banyak digunakan dalam game yang menggunakan kecerdasan buatan terutama pada game yang membutuhkan metode pencarian dan memiliki papan permainan yang berupa graph (memiliki node dan edge).Algoritma A* (adalah algoritma pencarian yang merupakan pengembangan dari algoritma Best First Search (BFS). Seperti halnya pada BFS, untuk menemukan solusi, A* (juga 'dituntun' oleh fungsi heuristik, yang menentukan urutan titik mana yang akan dikunjungi terlebih dahulu. Heuristik merupakan penilai yang memberi harga pada tiap verteks yang memandu A* mendapatkan solusi yang diinginkan.

Pada penelitian ini penulis akan menyelesaiakan masalah pencarian jalur terpendek menggunakan kotak- kotak, dimana user akan membuat jalur sendiri yang akan dilalui dengan menggunakkan kotak-kotak berwarna hitam, untuk membuatkan posisi awal user dapat menentukan posisi awal menggunakan kotak berwarna biru, dan user juga menentukan posisi tujuan yang ditentukan user menggunakan kotak berwarna merah. Setelah user menentukan posisi awal, posisi akhir dan menentukan jalur-jalur yang akan dilalui maka aplikasi ini nantinya akan menampilkan titik-titik jalur terpendek menuju posisi akhir. Algoritma yang digunakan untuk pathfinding adalah algoritma A*. Maka, penulis mengambil judul "SIMULASI PENCARIAN JALUR TERPENDEK DENGAN ALGORITMA A* (STAR)''.

2. RESEARCH METHOD

a. Tempat dan Waktu Penelitian
   Penelitian dilakukan di lingkungan kampus Universitas Katolik Santo Thomas dan akan dilaksanakan pada bulan April sampai dengan September 2022.
b. Tahapan dalam Penelitian
   Adapun tahapan metode penelitian yang dilakukan diantaranya:
   Studi Literatur
   1. Pada tahap ini, penelitian dilakukan dengan mengumpulkan data-data terkait perancangan aplikasi yang bersumber dari buku, jurnal, artikel, serta informasi yang bersumber dari internet yang menjadi pendukung informasi perancangan aplikasi
   2. Observasi
      Pada tahap ini, pengamatan secara langsung ke lokasi dilakukan untuk menemukan kebutuhan akan perancangan aplikasi dan keakuratan data. Adapun lokasi observasi dilakukan di Universitas Katolik Santo Thomas
c. Perancangan Kebutuhan
   Pada tahap ini, dilakukan pendefinisian kebutuhan yang didapat dari studi literatur serta observasi untuk memudahkan proses perancangan dengan perancangan kebutuhan mulai dari perangkat yang digunakan,basis data serta tim yang akan membantu proses perancangan aplikasi.
d. Analisis dan Desain sistem
   Pada tahap ini, perancangan aplikasi yang akan dibuat dilakukan berdasarkan analisis kebutuhan sistem yang akan dibangun. Tahap ini berkaitan dengan evaluasi perencanaan kedalam sebuah desain, pola, dan komponen yang diperlukan yang menghasilkan prototype. Adapun beberapa aktivitas dalam tahap ini diantaranya : perancangan interaksi fungsi serta objek sistem,perancangan skema dan database, dan perancangan user interface.
e. Implementasi
   Pada tahap ini, setiap rancangan yang dibuat diubahkan dalam bentuk code dengan menggunakan beberapa bahasa pemrograman, sehingga menghasilkan output berupa sistem aplikasi yang dapat dioperasikan.
f. Pembuatan Laporan
   Tahap terakhir penelitian, melakukan pembuatan laporan berupa dasar teori dan metode yang digunakan.

3. RESULTS AND DISCUSSION

1. Spesifikasi Perangkat Keras (Hardware) dan Perangkat Lunak (Software)

Dalam penerapan sistem yang dibuat tidak terlepas dari perangkat keras dan perangkat lunak. Untuk menguji program atau sistem informasi, digunakan komputer dengan spesifikasi sebagai berikut:

a. Perangkat Keras (Hardware)

a) Minimal Intel (R) Pentium (R) III CPU
b) RAM 128 MB
c) Hardisk Minimal 20 GBs
d) Monitor 12,1”

b. Perangkat Lunak (Software)
a) Sistem Operasi Microsoft Windows 7
b) Bahasa Pemrograma VB.NET 2008

2. Tampilan Hasil Program

Form utama akan tampil pertama sekali. Pada form utama ini terdapat beberapa toolbox. diantaranya yaitu :

a. Option Starting Tile : Starting tile adalah untuk menentukan posisi awal dimana disimbolkan dengan warna hijau
b. Option Ending Tile : Ending tile adalah untuk menentukan posisi akhir dimana di simbolkan dengan warna merah
c. Option Wall Tile : Wall tile adalah sebagai penghalang dari posisi tujuan ke posisi akkhir, yang di simbolkan silver
d. Button Run : Button run ini untuk menjalankan aplikasi atau untuk mencari rute terdekat dari posisi awal ke posisi akhir
e. Button Clear : Button clear untuk membersihkan jalur.

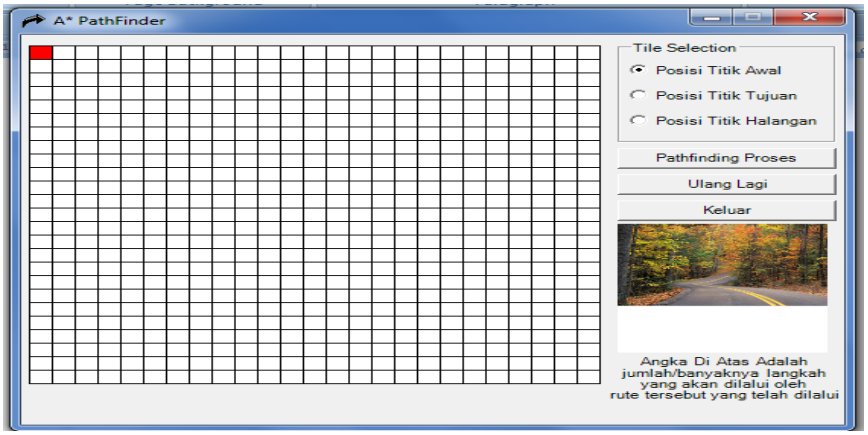

Figure 1. Form Utama

3. Uji Coba Program

a. Menentukan Posisi Awal Dan Posisi Akhir

Pada form ini akan di uji coba menentukan posisi awal dan posisi tujuan dimana warna hijau sebagai posisi awal dan warna merah sebagai posisi tujuan yang terlihat pada Figure 2

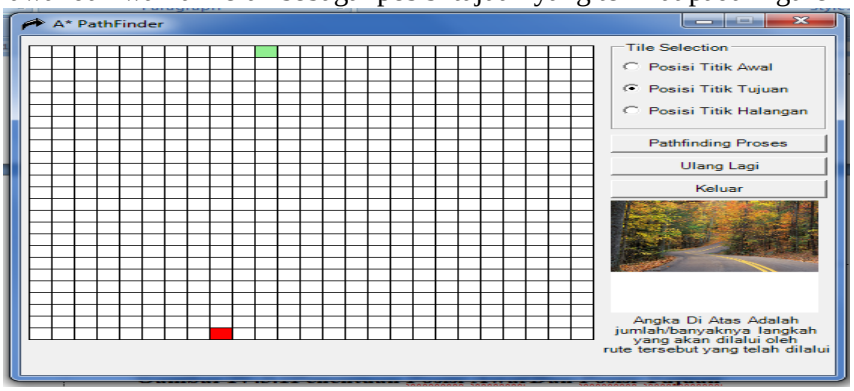

Figure 2. Penentuan Posisi Awal Dan Posisi Tujuan

b. Membuat Wall tile

Membuat wall tile adalah membuat penghalang dari posisi awal ke posisi tujuan pada penentun sebelumnya, untuk membuat penghalang user harus memilih optionwall tile, dan memilih dimana kotak – kotak tersebut di tempatkan sebagai penghalang yang disimbolkan dengan warna blue, untuk menempatkan penghalang tersebut user tinggal mengclick dimana titik-titik penghalang tersebut ingin ditempatkan yang terlihat pada Figure 3

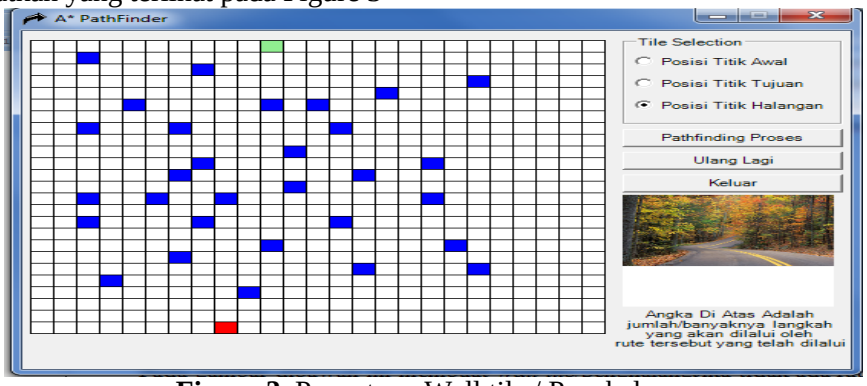

Figure 3. Penentuan Wall tile / Penghalang

c. Membuat Wall tile Jika tidak ada rute yang tidak dilewati

Pada gambar dibawah ini membuat wall tile/penghalangJika tidak ada rute yang tidak dilewati dari posisi awal ke posisi tujuan, untuk membuat penghalanguser harus memilih optionwall tile, dan memilih dimana kotak – kotak tersebut di tempatkan sebagai penghalang yang disimbolkan dengan warna silver, Yang tidak bias dilalui/dilewatkan oleh titik-titik tersebut jika dirun. Akan muncul perintah tersebut solusi tidak ditemukan, akan terlihat pada Figure 4

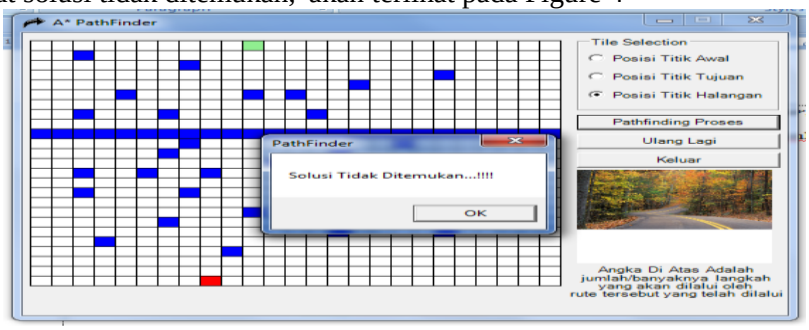

Figure 4 Penentuan Wall tileJika tidak ada rute yang tidak dilewati

d. Membuat Keterangan/hasil pencarian jalur terpendek

Pada gambar di bawah ini akan ditampilkan beberapa titik-titik berwarna biru , titik-titik tersebut adalah rute yang di lalui menuju posisi akhir berwarna merah dari posisi awal berwarna hijau, kotak-kotak yang dilalui akan terlihat beberapa kotak yang telah dilaluidan menunjukkan nilaia Diagonal, Fertikal Dan Horizontaldimana nilai untuk pergerakan Diagonal adalah empat belas(14), dan nilai untuk pergerakan Vertikal Dan Horizontal adalah sepuluh(10), dapat terlihat pada Figure 5

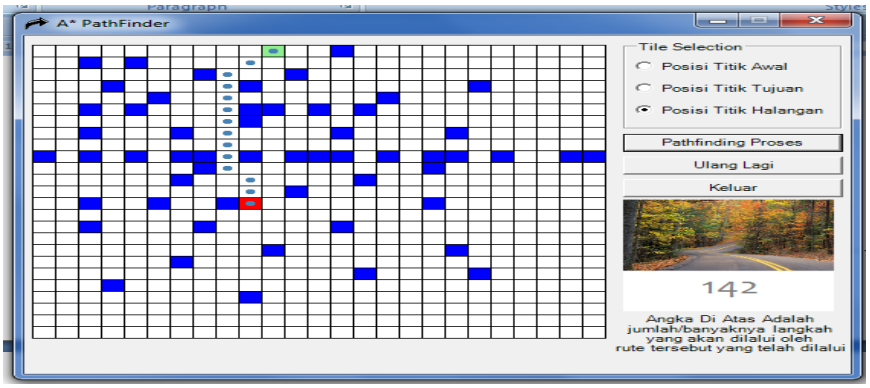

Figure 5. Penenetuan keterangan/hasil pencarian

4. V.2 Pembahasan

Aplikasi pathfinding akan berupa kotak-kotak dengan ordo X x Y, supaya pengguna bisa menentukan sendiri orde yang diinginkan. Pada ordo X x Y, 2 kotak akan dipakai oleh titik awal dan titik tujuan, sisanya untuk menghasilkan jalan agar terlihat jalurnya yang pada akhirnya akan menentukan jalur terpendek ke titik tujuan. Maksimal penghalang pada tiap ordo dapat kita tentukan yaitu dengan rumus pada tabel 1

Tabel 1. Ordo

Ordo	Maksimal Penghalang
X x Y	(M x N) – 2
3 x 3	(3 X 3) – 2 = 7
4 x 4	(4 x 4) – 2 = 14
5 x 5	(5 x 5) – 2 = 23
… x …	(… x …) – 2 = …

Keterangan :
X : Nilai baris
Y : Nilai kolom

Dengan maksimal penghalang (X x Y)-2, maka akan menghasilkan jalan yang tidak terlihat disebabkan jarak antara titik awal dan titik tujuan saling berdekatan, tapi dengan kondisi titik awal dan titik tujuan saling berjauhan maka akan mengahasilkan jalan tidak akan ditemukan. Agar terlihat jalan yang dihasilkan maka diusahakan titik awal dengan titik tujuan tidak saling berdekatan dan jumlah penghalang yang akan dipasangkan kurang dari maksimal, implementasinya dapat dilihat dengan contoh ordo 3x3 pada Figure 6

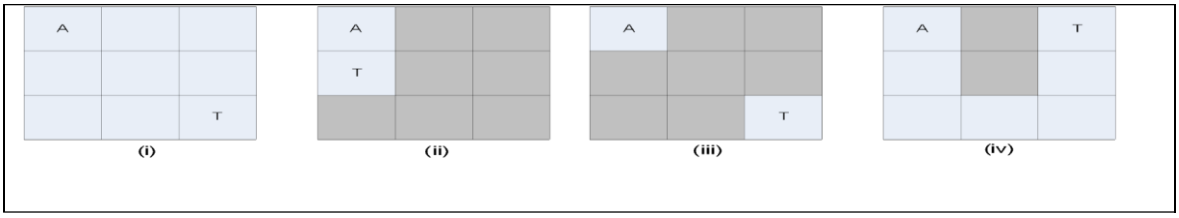

Figure 6. Ruang peta (map) ordo 3 x 3 dengan tiga kondisi

Keterangan :
A : Titik awal
T : Titik Tujuan

Maksimal penghalang yang akan dipasangkan pada ordo lainnya adalah sebanyak ordo tersebut dikurangi dua untuk menempatkan titik awal dan titik tujuan. Pada gambar III.3 menunjukkan suatu ruang (map) dengan ordo 3 x 3 di dalam Aplikasi yang akan dibangun. Setiap kotak mempresentasikan simpul(node). Setiap kotak terhubung ke delapan kotak yang paling dekat, artinya setiap simpul (node) terhubung ke simpul lain yang berada di sebelah kanan, kiri, atas-kanan, bawah-kanan, bawah-kiri, dan atas-kiri dari simpul tersebut. Kotak warna orange dan warna hijau diimplementasikan sebagai penghalang, yaitu kotak yang tidak dapat dilalui oleh titik awal.

Sekarang, akan mencari jalan terpendek dari posisi titik awal ke posisi titik tujuan. Karena titik A tidak terhubung langsung ke titik T, maka perlu melewati simpul-simpul tertentu yang pada akhirnya akan mengantarkan ke titik T dengan jarak seminimal mungkin.

Terdapat beberapa hal yang perlu didefinisikan terlebih dahulu dalam kasus Aplikasi pathfinding dengan penerapan algoritma A*. Adapun istilah-istilah yang akan dibahas yaitu path, open list, closed list, nilai f, g dan n.

Algoritma A* menggunakan dua senarai yaitu *OPEN* dan *CLOSED. OPEN* adalah senarai (list) yang digunakan untuk menyimpan simpul-simpul yang pernah dibangkitkan dan nilai heuristiknya telah dihitung tetapi belum terpilih sebagai simpul terbaik (best node) dengan kata lain, OPEN berisi simpul-simpul masih memiliki peluang untuk terpilih sebagai simpul terbaik, sedangkan CLOSED adalah senarai untuk menyimpan simpul-simpul yang sudah pernah dibangkitkan dan sudah pernah

terpilih sebagai simpul terbaik. Artinya, *CLOSED* berisi simpul-simpul yang tidak mungkin terpilih sebagai simpul terbaik (peluang untuk terpilih sudah tertutup).

1) *OPEN LIST* adalah list yang menyimpan kemungkinan path yang akan diperiksa. *OPEN LIST* dibuat terurut berdasarkan nilai f. *OPEN LIST* digunakan untuk menentukan secara selektif (berdasarkan nilai f) jalan yang dikira lebih dekat menuju pada path tujuan. *OPEN* berisi simpul-simpul yang masih memiliki peluang untuk terpilih sebagai simpul terbaik (best node).
2) CLOSED adalah senarai(list) untuk menyimpan simpul-simpul yang sudah pernah dibangkitkan dan sudah pernah terpilih sebagai simpul terbaik (best node) atau senarai yang menyimpan jalan yang sudah diperiksa dari open list. Artinya, *CLOSED* berisi simpul-simpul yang tidak mungkin terpilih sebagai simpul terbaik (peluang untuk terpilih sudah tertutup).Kedua list (*OPEN LIST* dan *CLOSED LIST*) ini bertujuan juga untuk menghindari penelusuran berkali-kali jalan (rute) yang memang sudah diidentifikasi agar tidak masuk kembali ke dalam *OPEN LIST*.
3) Nilai F adalah cost perkiraan suatu path yang teridentifikasi. Nilai F merupakan hasil dari f(n).
4) Nilai G hasil dari fungsi g(n), adalah banyaknya langkah yang diperlukan untuk menuju ke path sekarang.
5) Setiap simpul(node) harus memiliki informasi nilai h(n), yaitu estimasi harga simpul tersebut dihitung dari simpul tujuan yang hasilnya menjadi nilai H. Fungsi f sebagai estimasi fungsi evaluasi terhadap node, dapat dituliskan :

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

dengan keterangan:
f(n)= fungsi evaluasi (jumlah g(n) =dengan h (n))
g(n)= biaya (cost) yang dikeluarkan dari keadaan awal sampai keadaan n
h(n)= estimasi biaya untuk sampai pada suatu tujuan mulai dari n

Pergerakan diagonal pada map diperbolehkan, maka digunakan fungsi heuristic *Non-Manhattan* Distance. Maka fungsi *heuristic* yang digunakan adalah sebagai berikut :

$$h\_diagonal(n) = min(abs(n.x-goal.x) + abs(n.y-goal.y))$$
$$h\_orthogonal(n) =(abs(n.x-goal.x) + abs(n.y-goal.y))$$
$$h(n) = h\_diagonal(n) + (h\_orthogonal\ (n)–(2 * h\_diagonal(n)))$$

Sudah dijelaskan pada analisis masalah bahwa ordo dapat disesuaikan dengan rentang X x Y, salah satu contoh perhitungan pada Aplikasi pathfinding ini akan dijelaskan dengan ordo minimal yaitu 3x3 dikarenakan perhitungan pada ordo berapa pun akan sama. Pada penentuan bobot setiap node akan diberikan nilai sesuai dengan jarak terdekat ke tujuan, missal node yang terjauh dari tujuan maka diberi bobot yang kecil sedangkan node yang terdekat dengan tujuan diberi bobot yang lebih besar. Contoh perhitungannya seperti pada Figure 7 yang diberi nilai bobot yang terkecil yaitu 1 dan bobot selanjutnya ditambahkan 1.

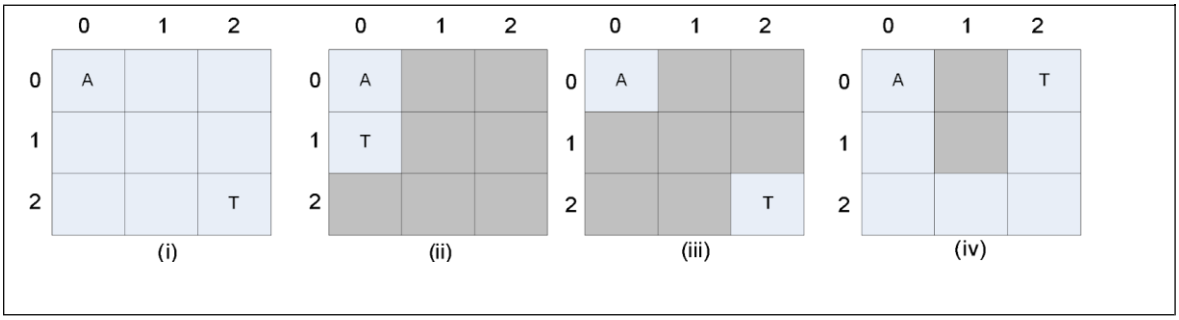

Figure 7 Contoh kondisi ruang map yang akan dihitung dengan A*

Perhitungan yang dilakukan dengan algoritma A* dengan kondisi tanpa penghalang yang terlihat pada Figure 8

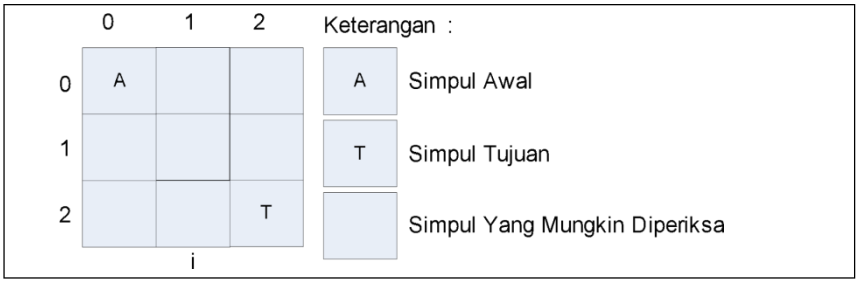

Figure 8 Contoh kondisi (i) tanpa penghalang dengan pencarian A*

Posisi simpul awal = Ax : 0, Ay : 0
Posisi simpul tujuan = goal x : 2, goal y : 2
Langkah ke satu
n (1,1) : g (1,1)=1
h_orthogonal(n) = (abs(n.x-goal.x) + abs(n.y-goal.y))
h_orthogonal(1,1) = (abs(1-2) + abs(1-2))
= (abs(-1) + abs(-1))
= 2
h_diagonal(n) = min(abs(n.x-goal.x) + abs(n.y-goal.y))
h_diagonal(1,1) = min(abs(1-2)+abs(1-2))
= min(abs(-1)+abs(-1))
= min 2
h(n) = h_diagonal(n) + (h_orthogonal(n)–(2 * h_diagonal(n)))
h(1,1) = (-2) + (2-(2*(-2)))
=(-2)+(2-(-4))
=-2 + 6
=4
f (1,1) =g (1,1) + h (1,1)
=1 + 4
=5
n (1,0) : g (1,0) =1
h_orthogonal(n) =(abs(n.x-goal.x) + abs(n.y-goal.y))
h_orthogonal(1,0) =(abs(1-2) + abs(0-2))
= (abs(-1) + abs(-2))
= 3
h_diagonal(n) = min(abs(n.x-goal.x), abs(n.y-goal.y))
h_diagonal(1,0) = min(abs(1-2)+abs(0-2))
= min(abs(-1)+abs(-2))
= min 3
h(n) = h_diagonal(n) +(h_orthogonal(n)–(2 * h_diagonal(n)))
h(1,0) = (-3) + (3-(2*(-3)))
=(-3)+(3-(-6))
=-3 + 9
=6
f (1,0) =g (1,0) + h (1,0)
=1 + 6
=7
n (0,1) : g (0,1) =1
h_orthogonal(n) = (abs(n.x-goal.x) + abs(n.y-goal.y))
h_orthogonal(0,1) =(abs(0-2) + abs(1-2))
= (abs(-2) + abs(-1))

= 3
h_diagonal(n) = min(abs(n.x-goal.x) + abs(n.y-goal.y))
h_diagonal(0,1) = min(abs(0-2)+abs(1-2))
= min(abs(-2)+abs(-1))
= min 3
h(n) = h_diagonal(n) +(h_orthogonal(n)–(2 * h_diagonal(n)))
h(0,1) =(-3) + (3-(2*(-3)))
=(-3)+(3-(-6))
=-3 + 9
=6
f (0,1) = g (0,1) + h (0,1)
= 1 + 6
= 7

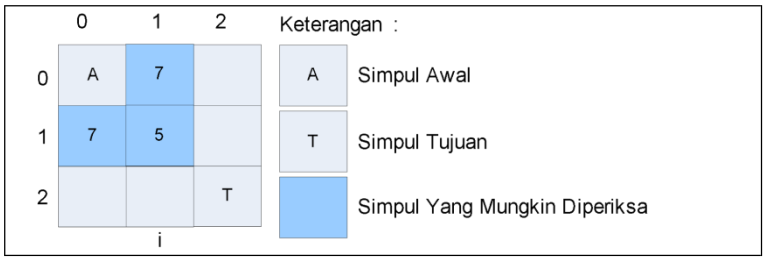

Figure 9. Langkah pertama pencarian BestNode pada kondisi (i)

Pada Figure 9 terdapat tiga simpul yang mungkin menjadi Best Node yaitu (1,0) dengan f(n)=7, (1,1) dengan f(n)=5 dan (0,1) dengan f(n)=7. Dari ke tiga simpul yang mungkin maka dipilihlah simpul (1,1) dengan biaya terkecil yaitu 5.

Langkah ke dua
n (2,2) : g (2,2) = 2
h_orthogonal(n) = (abs(n.x-goal.x) + abs(n.y-goal.y))
h_orthogonal(2,2)
= (abs(2-2) + abs(2-2))
= (abs(0) + abs(0))
= 0
h_diagonal(n) = min(abs(n.x-goal.x) + abs(n.y-goal.y))
h_diagonal(2,2) = min(abs(2-2)+abs(2-2))
= min(abs(0)+abs(0))
= min 0
h(n) = h_diagonal(n) + (h_orthogonal(n)–(2 * h_diagonal(n)))
h(0,1) = (-0) + (0-(2*(-0)))
= 0 + 0
= 0
f (2,2) = g (2,2) + h (2,2)
= 2 + 0
= 2

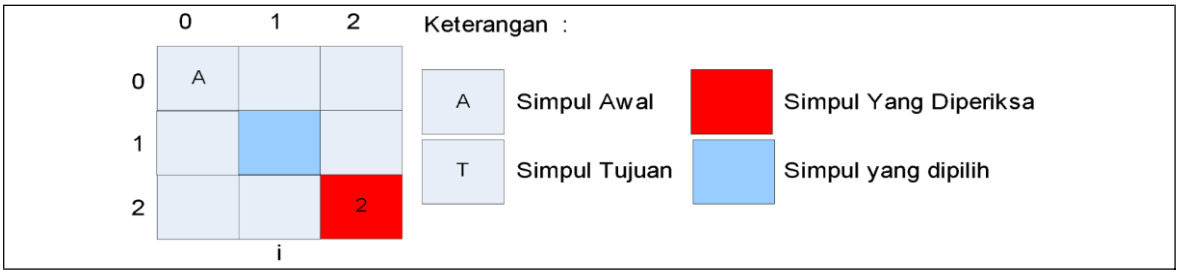

Figure 10. Langkah ke dua pencarian BestNode pada kondisi (i)

Pada Figure 10 terdapat satu simpul yang mungkin menjadi BestNode yaitu (2,2) dengan f(n)=2, dan dikenali sebagai simpul tujuan yaitu (2,2) berarti solusi telah ditemukan.

Figure 11 Hasil pencarian jalur dengan Algoritma A* pada kondisi (i)

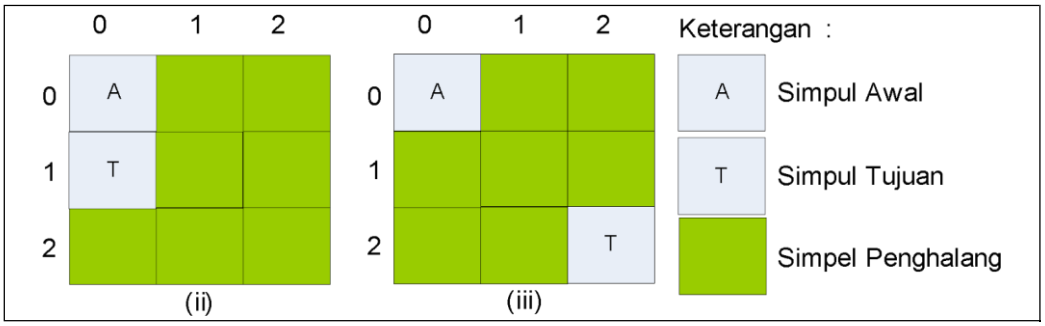

Figure 12 Contoh kondisi(ii) dan (iii) dengan maksimal penghalang

4. CONCLUSION

Dari semua perhitungan yang telah dilakukan dipilihlah biaya/cost terkecil pada setiap langkahnya sehingga akan menghasilkan jalur terpendek yang terlihat pada Figure 11. Pada contoh ordo 3x3 dengan kondisi penghalang maksimal dan simpul awal dengan simpul tujuan saling berdekatan pada kondisi (ii) dengan posisi simpul awal yaitu (0,0) dan posisi simpul tujuan yaitu (0,1), maka pencarian simpul yang diperiksa hanya satu simpul saja yaitu simpul (0,1) karena simpul lain yang berdekatan dengan posisi simpul awal merupakan penghalang yang tidak bisa dilewati, kemudian simpul (0,1) dikenali sebagai simpul tujuan yang berarti solusi telah ditemukan. Jika kondisi simpul awal dengan simpul tujuan saling berjauhan pada kondisi (iii) dimanapun posisinya dengan penghalang maksimal, maka solusi tidak akan ditemukan sebab tidak ada jalan yang dapat dilewati untuk menuju tujuan. Kondisi (ii) dan (iii) dapat dilihat pada Figure 12.

REFERENCES

[1] Adhiwidjaja, S. B., Aditung, P., Rustan, A. D., Pranata, T. P., & Lisangan, E. A. (2020). Prototipe Aplikasi Peringatan DIni Covid-19 Berbasis Location Based Service. Konferensi Nasional Ilmu Komputer (KONIK), 682-686.

[2] Aditya, R., Pranatawijaya, V. H., & Putra, B. P. (2021). Rancang Bangun Aplikasi Monitoring Kegiatan Menggunakan Metode Prototype. JOINTECOMS (Journal of Information Technology and Computer Science), 47-57.

[3] Budiaji, W. (2013). Skala Pengukuran dan Jumlah Respon Skala Likert. Jurnal Ilmu Pertanian dan Perikanan Desember , 127-133.

[4] Joshi, A., Kale, S., Chandel, S., & Pal, D. (2015). Likert Scale: Explored and Explained. British Journal of Applied Science & Technology, 397-403.

[5] Pressman, R. S., & Maxim, B. R. (2015). Software Engineering: A Practitioner's Approach. New York: McGraw-Hill Education.

[6] Purnomo, D. (2017). Model Prototyping Pada Pengembangan Sistem Informasi. JIMP - Jurnal Informatika Merdeka Pasuruan, 54-61.

[7] Syarifudin, A., & Ani, N. (2019). Rancangan Sistem Informasi Pengajuan dan Pelaporan Tunjangan Kinerja Kementerian Keuangan Menggunakan Metode Prototype. SISFOKOM, 149-158.
[8] Thohari, A. N., & Vernandez, A. B. (2020). Aplikasi Monitoring Kasus Coronavirus Berbasis Android. JTET (Jurnal Teknik Elektro Terapan), 12-17.
[9] Bambang Sridadi (2009), Pemodelan dan Simulasi Sistem, Penerbit Informatika Bandung
[10] Pranat, 2000, Algoritma dan Pemrograman dengan bahasa java, Penerbit Grahayu Ilmu, Yogyakarta.
[11] Dalem, I. B. G. W. A. (2018). Penerapan algoritma A*(Star) menggunakan graph untuk menghitung jarak terpendek. Jurnal RESISTOR (Rekayasa Sistem Komputer), 1(1), 41-47.
[12] Rizky, R. (2018, November). Pencarian Jalur Terdekat dengan Metode A*(Star) Studi Kasus Serang Labuan Provinsi Banten. In Prosiding Seminar Nasional Rekayasa Teknologi Informasi| SNARTISI (Vol. 1).
[13] Senn, 2005, Analysis And Design Of Information System International, Penerbit McGraw-HILL, Singapore
[14] Suarga, (2006), Algoritma Pemrograman, Penerbit Andi, Yogyakarta.