Penerapan Algoritma A Star Dalam Penentuan Jalur Evakuasi Bencana Gedung Bertingkat

Gama Pradipta Wirawan / 13517049
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung. Jl. Ganesha 10 Bandung 40132, Indonesia
gamapradipta88@gmail.com

***Abstrak*—Jalur evakuasi, merupakan salah satu aspek yang cukup penting bagi pengelola gedung dalam melakukan manajemen bencana. Jalur evakuasi berfungsi untuk membantu setiap orang yang berada dalam gedung tersebut untuk menyelamatkan diri saat terjadi bencana atau kejadian yang tidak diinginkan. Penelitian ini dimaksudkan untuk membantu menentukan jalur evakuasi yang efektif sehingga setiap orang yang berada dalam gedung dapat melakukan evakuasi dengan cepat dan aman. Penentuan jalur evakuasi bergantung pada posisi awal dari penghuni atau pengujung dan posisi pintu keluar gedung. Pada makalah ini, akan dijelaskan bagaimana algoritma A* (*A Star*) dapat membantu pihak pengelola gedung dalam menentukan jalur evakuasi yang aman dan efektif dari posisi pengunjung atau penghuni menuju pintu keluar gedung.**

Keywords—Algoritma A*, Evakuasi Bencana, Gedung Bertingkat, Jalur Evakuasi

I. PENDAHULUAN

Salah satu faktor yang cukup penting dalam manajemen bencana suatu gedung adalah jalur evakuasi untuk keluar gedung saat terjadi bencana alam atau kejadian-kejadian yang dapat mengancam hidup baik penghuni maupun pengujung gedung tersebut. Tanpa adanya jalur evakuasi, korban dari bencana alam yang terjadi dapat bertambah menjadi sangat banyak. Dengan adanya jalur penyelamatan keluar dari gedung, diharapkan bahwa korban dari bencana yang terjadi dapat dikurangi.

Ketika bencana terjadi, baik penghuni maupun pengunjung gedung belum tentu dapat mengetahui lokasi-lokasi yang berbahaya untuk dilewati. Salah satu contohnya adalah gedung yang mengalami kebakaran. Pada saat kebakaran terjadi, baik pengunjung dan penghuni tidak mengetahui letak kebakaran itu terjadi. Akibatnya apabila jalur evakuasi yang disediakan melewati lokasi berbahaya, pengunjung dan penghuni dapat mengambil jalur evakuasi yang berbahaya dan kasus terburuknya adalah bertambahnya jumlah korban dari bencana tersebut.

Selain itu jalur evakuasi haruslah merupakan jalur tercepat untuk keluar dari gedung dari posisi penghuni atau pengunjung menuju pintu keluar darurat gedung tersebut. Dengan adanya jalur evakuasi yang cepat maka semakin banyak pula orang-orang yang dapat menyelamatkan diri mereka dari bencana alam yang tengah terjadi. Walaupun bencana alam atau kejadian - kejadian buruk tidak dapat dihindari namun dengan adanya jalur evakuasi yang efektif dan aman, maka korban bencana yang terjadi dapat diminimalisir.

Namun meski demikian, untuk menentukan jalur evakuasi yang aman dan cepat dibutuhkan informasi yang tepat mengenai lokasi pintu keluar darurat dan juga lokasi-lokasi yang berbahaya untuk dilewati oleh penghuni dan pengunjung gedung tersebut. Salah satu contohnya, kita dapat memanfaatkan sensor kebakaran untuk mendeteksi lokasi-lokasi yang berbahaya pada saat terjadi kebakaran di dalam gedung tersebut. Dan dengan adanya informasi yang cukup maka penentuan jalur evakuasi dapat memberikan jalur penyelamatan keluar dari gedung yang efektif dan aman. Salah satu algoritma yang dapat memberikan jalur yang efektif adalah algoritma A*.

Algoritma ini bertujuan untuk memberikan suatu jalur terpendek dari suatu titik awal menuju titik akhir. Dengan asumsi bahwa waktu bagi pengunjung dan penghuni untuk keluar dari gedung berbanding lurus dengan jarak mereka dari pintu keluar maka algoritma ini cocok untuk menyelesaikan permasalahan ini.

II. DASAR TEORI

A. *Path Finding*

Path Finding merupakan proses pencarian jalur dari suatu titik atau node menuju titik atau node lainnya. Proses pencarian ini biasa dilakukan untuk melakukan pencarian pada graf. Pada umumnya proses ini bertujuan untuk mencari rute terpendek dari satu node ke node lainnya.

B. *Shortest Path*

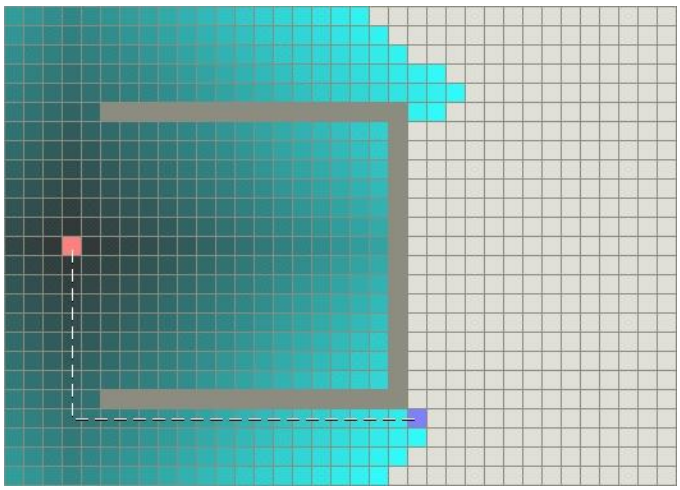
Shortest Path atau jalur terpendek merupakan optimalisasi dari proses pencarian rute (*path finding*). Terdapat banyak metode untuk menemukan jalur terpendek, dan pada makalah

ini penulis memakai algoritma A* untuk menemukan jalur terpendek dari suatu titik menuju titik lainnya.

*C. Algoritma A**

Algoritma A* adalah salah satu algoritma pencarian rute yang cukup sering digunakan untuk mencari rute terdekat dari suatu titik menuju titik lainnya. Algoritma A* termasuk dalam algoritma pencarian rute bertipe *informed search*. Algoritma ini dapat digunakan untuk mencari rute terdekat dari suatu lokasi menuju lokasi lainnya yang dimana terdapat suatu lokasi yang tidak dapat dilewati dalam melakukan pencarian.

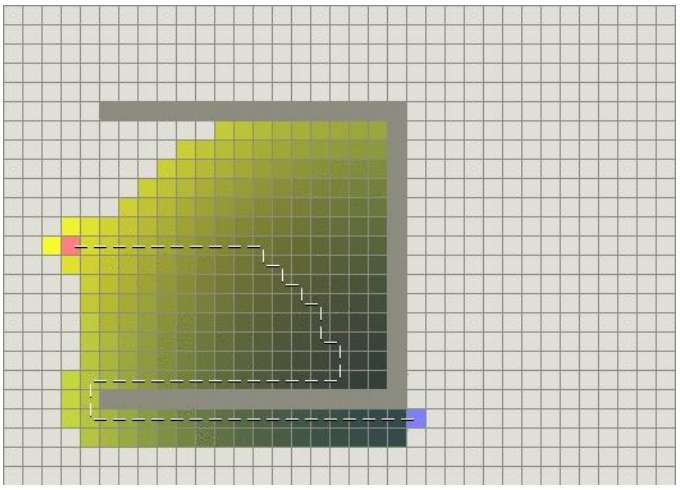
Algoritma ini menerapkan 2 algoritma sekaligus yaitu algoritma *Greedy-Best First Search* dan algoritma *Dijkstra*. Algoritma *Dijkstra* memiliki kelebihan yaitu algoritma ini dapat mencari rute terpendek dari suatu titik ke titik lainnya. Namun kelemahannya adalah algoritma ini membutuhkan waktu yang cukup lama untuk menjelajahi semua jalur yang ada untuk menemukan jalur terpendek.



Gambar 1. Proses pencarian algoritma *Dijkstra*
Source:http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

Pada Gambar 1. diatas, warna abu-abu melambangkan jalur yang tidak dapat dilalui, warna merah melambangkan titik awal pencarian dan warna ungu melambangkan titik akhir pencarian. Sedangkan warna biru melambangkan jalur pencarian dari algoritma *Dijkstra*. Dari Gambar 1. diatas dapat dilihat bahwa rute yang dihasilkan merupakan rute terdekat dari titik awal menuju titik akhir, namun proses pencariannya membutuhkan waktu yang cukup lama.

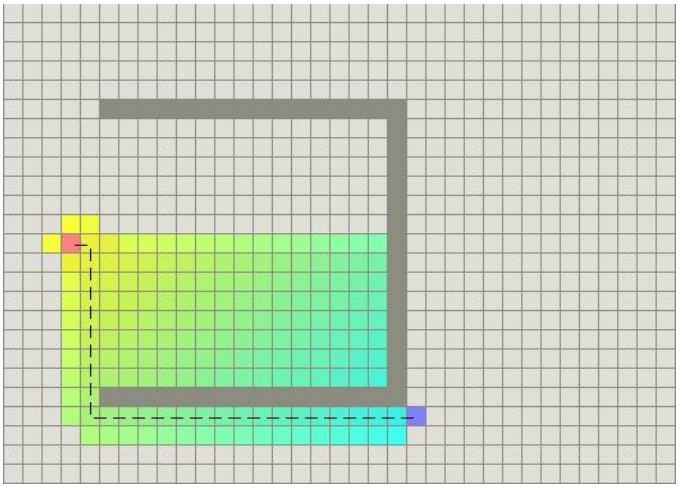
Sedangkan algoritma *Greedy-Best First Search* memiliki kelebihan yaitu lama proses yang dibutuhkan untuk mencari rute dari suatu titik ke titik lainya relatif lebih cepat daripada algoritma *Dijkstra*, namun rute yang dihasilkan oleh algoritma ini tidak dapat menjamin bahwa rute itu merupakan rute terdekat dari satu titik ke titik lainnya.



Gambar 2. Proses pencarian algoritma *Greedy-Best First Search*
Source:http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

Pada Gambar 2. diatas, warna abu-abu melambangkan jalur yang tidak dapat dilalui, warna merah melambangkan titik awal pencarian dan warna ungu melambangkan titik akhir pencarian. Sedangkan warna kuning melambangkan jalur pencarian dari algoritma *Greedy-Best First Search*. Dari Gambar 2. diatas dapat dilihat proses pencarian yang dilakukan algoritma ini lebih sedikit dibanding algoritma *Dijkstra*, tapi jalur yang dihasilkan bukan merupakan jalur terpendek dari titik awal ke titik akhir.

Algoritma *A**, yang merupakan gabungan dari kedua algoritma *Greedy-Best First Search* dan algoritma *Dijkstra* mengambil kelebihan dari kedua algoritma tersebut, sehingga dengan waktu pencarian yang cukup cepat algoritma ini dapat menjamin bahwa rute yang dihasilkan merupakan rute terdekat dari suatu titik menuju titik lainnya.



Gambar 3. Proses pencarian algoritma *A**
Source:http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

Pada Gambar 3. diatas, warna abu-abu melambangkan jalur yang tidak dapat dilalui, warna merah melambangkan titik awal pencarian dan warna ungu melambangkan titik akhir pencarian.

Sedangkan warna kuning-hijau melambangkan jalur pencarian dari algoritma *A**. Dari Gambar 3. diatas dapat dilihat proses pencarian yang dilakukan algoritma ini mengkombinasikan algoritma *Dijkstra* dan algoritma *Greedy-Best First Search* yang dimana algoritma ini dapat memberikan rute jalur terpendek dengan proses pencarian yang relatif cepat.

Algoritma *A** dapat memberikan rute terdekat dengan waktu proses pencarian yang cukup singkat dengan menggunakan fungsi *f(n)*. Rumus dari fungsi *f(n)* dituliskan sebagai persamaan sebagai berikut :

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

Fungsi *heuristic* ini didapatkan dengan menjumlahkan fungsi *g(n)* yang merupakan harga atau bobot dari titik awal hingga ke titik ke-n dan fungsi *h(n)* yang merupakan estimasi harga dari titik ke-n. Semakin tinggi keakuratan fungsi *h(n)* (*heuristic*) maka semakin cepat dan tepat rute yang akan dihasilkan oleh algoritma A* ini.

Dalam implementasinya algoritma A* menggunakan 2 buah senarai yaitu *Open List* dan *Closed List*, dan terdapat 3 buah kondisi bagi setiap simpul yang dibangkitkan yaitu sudah berada di *Open List*, sudah berada di *Closed List* dan tidak berada di keduanya. Pada ketiga kondisi tersebut diberikan penanganan yang berbeda-beda.

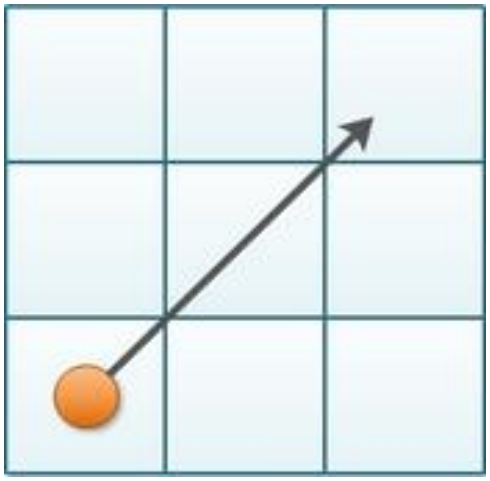
Open List adalah senarai yang digunakan untuk menyimpan simpul-simpul yang sudah pernah dibangkitkan dan telah dihitung nilai heuristiknya namun belum dipilih sebagai simpul terbaik. Hal ini dapat diartikan bahwa senarai ini berisi simpul-simpul yang memiliki peluang untuk dipilih menjadi simpul terbaik. Sedangkan *Closed List* adalah senarai yang digunakan untuk menyimpan simpul-simpul yang sudah pernah dibangkitkan dan sudah pernah dipilih menjadi simpul terbaik. Hal ini dapat diartikan bahwa senarai ini berisi simpul-simpul yang tidak dapat dipilih kembali sebagai simpul terbaik atau dapat dikatakan bahwa simpul-simpul dalam senarai ini tidak memiliki peluang lagi untuk terpilih kembali sebagai simpul terbaik. Dalam proses pemilihan simpul terbaik dari *Open List*, digunakan fungsi *f(n)* sebagai parameter pengambilan keputusan simpul terbaik. Dengan memakai fungsi *f(n)* ini maka pengambilan solusi dengan metode algoritma *A** dapat menjadi efektif secara waktu dan dapat memberikan solusi yang terbaik.

D. Fungsi Heuristic

Fungsi *heuristic* adalah fungsi yang digunakan dalam algoritma *A** untuk mengevaluasi perkiraan harga (*cost*) dari suatu simpul dalam graph menuju simpul lainnya.

Dalam permasalahan dengan jarak terdapat 3 jenis fungsi *heuristic* yang dapat digunakan yaitu *Euclidean*, *Chebyshev*, dan *Manhattan distance*.

1. Euclidean Distance

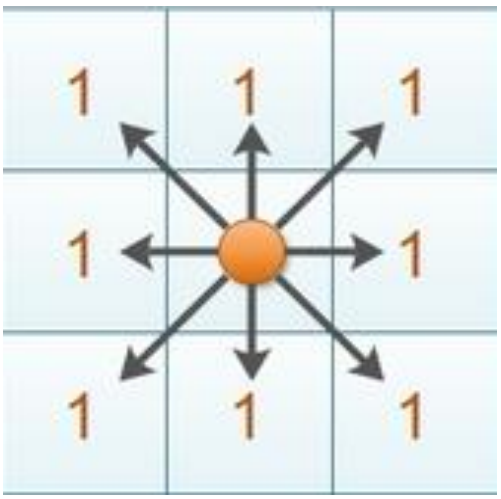


Gambar 4. *Euclidean Distance*
Source:https://lyfat.wordpress.com/2012/05/22/euclidean-vs-chebyshev-vs-manhattan-distance/

Euclidean Distance memperkirakan *cost* yang dibutuhkan dengan cara menghitung jarak secara horizontal, vertikal, atau diagonal dari suatu simpul menuju simpul lainnya. Formula dari *Euclidean Distance* adalah sebagai berikut.

$$h(i) = sqrt[(X_o - X_i)^2 + (Y_o - Y_i)^2]$$

2. Chebyshev Distance



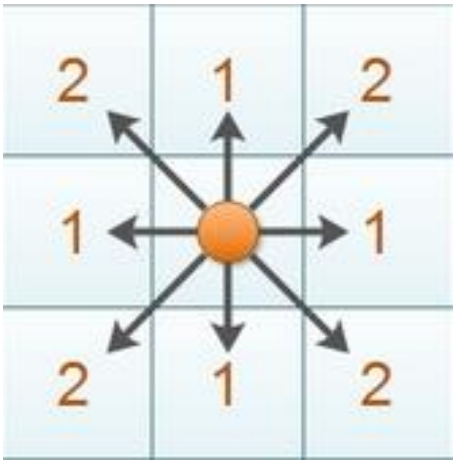
Gambar 5. *Chebyshev Distance*
Source:https://lyfat.wordpress.com/2012/05/22/euclidean-vs-chebyshev-vs-manhattan-distance/

Chebyshev Distance memperkirakan *cost* yang dibutuhkan dengan cara menghitung jarak dari suatu simpul menuju simpul lainnya dengan asumsi bahwa *cost* dari simpul tersebut ke semua arah bernilai sama

(secara diagonal, vertikal dan horizontal). Formula dari *Chebyshev Distance* adalah sebagai berikut.

$$h(i) = max[(X_o - X_i), (Y_o - Y_i)]$$

3. *Manhattan Distance*



Gambar 6. *Chebyshev Distance*
Source:https://lyfat.wordpress.com/2012/05/22/euclidean-vs-chebyshev-vs-manhattan-distance/

Manhattan Distance memperkirakan *cost* yang dibutuhkan dengan cara menghitung jarak dari suatu simpul menuju simpul lainnya dengan asumsi bahwa jumlah jarak dari kedua simpul merupakan penjumlahan nilai mutlak dari perbedaan jarak berdasarkan sumbu X dan Y. Formula dari *Manhattan Distance* sebagai berikut.

$$h(i) = | X_o - X_i | + | Y_o - Y_i |$$

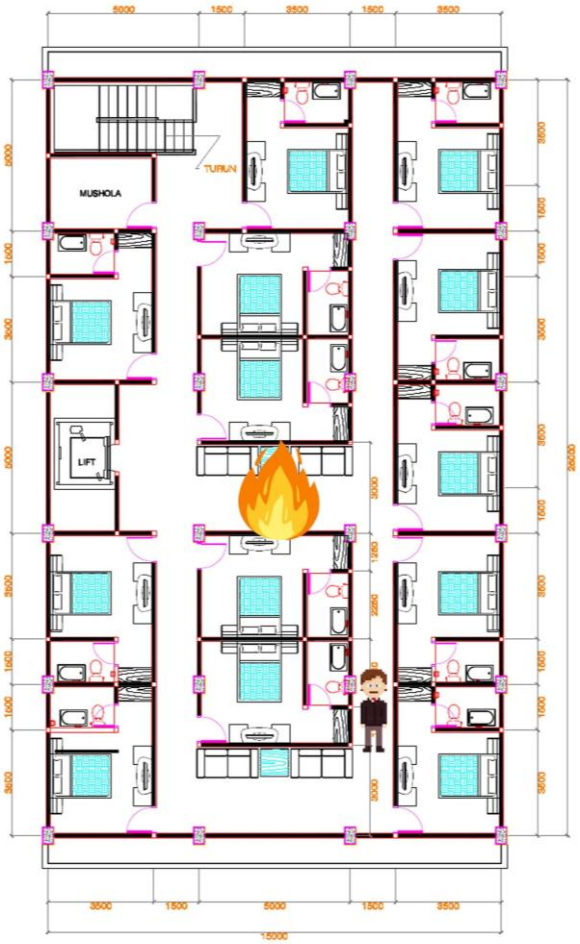
E. *Jalur Evakuasi*

Jalur evakuasi merupakan suatu rute yang digunakan untuk menghubungkan seluruh area menuju tempat yang aman. Jalur evakuasi dibuat untuk membantu semua orang yang berada di dalam lokasi untuk dapat menyelamatkan diri dari bahaya bencana ataupun kejadian-kejadian yang tidak diinginkan. Dengan adanya jalur evakuasi, korban-korban dari bencana atau kejadian yang tidak diinginkan diharapkan dapat diminimalisir. Tanpa adanya jalur evakuasi, orang-orang yang berada di lokasi kejadian akan semakin susah untuk menemukan jalur keluar atau tempat aman.

III. PERANCANGAN ALGORITMA

Untuk merancang dan menerapkan algoritma pada persoalan rute evakuasi ini, terdapat beberapa tahapan yang perlu dilakukan terlebih dahulu. Pertama, kita harus dapat memetakan denah lantai dari gedung tersebut, lokasi mana saja yang bisa kita lewati dan lokasi mana yang tidak bisa kita lewati seperti tembok, pilar dan semacamnya. Selain itu kita juga perlu memetakan lokasi pintu darurat pada lantai gedung tersebut.

Untuk menggambarkan permasalahan ini, akan digunakan sebuah denah dari sebuah lantai pada suatu gedung. Gambar orang melambangkan posisi dari penghuni atau pengunjung gedung, tangga melambangkan pintu darurat yang menjadi titik akhir dari pencarian rute ini dan api melambangkan posisi kebakaran yang terdeteksi oleh sensor kebakaran gedung tersebut.



Gambar 7. Denah Lantai 1 beserta lokasi api dan penghuni
Source:pribadi

A. *Proses Pemetaan Kondisi Gedung*

Untuk mendapatkan kondisi yang akurat pada proses pemetaan kondisi gedung maka terlebih dahulu denah kondisi

terkini dari lantai gedung tersebut diubah menjadi Grid 2D seperti berikut ini.

T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
				E		T			T		T			T
T	T	T	T	T		T		T	T		T		T	T
T									T					T
T	T	T	T		T	T	T	T	T		T	T	T	T
T			T						T					T
T	T		T		T		T		T		T		T	T
T					T	T	T	T	T		T			T
T	T	T	T		T		T		T		T	T	T	T
T		T							T		T			T
T					T	T	T	T	T		T		T	T
T		T					A							T
T	T	T	T		T	T	T	T	T		T	T	T	T
T									T					T
T	T		T		T		T		T		T		T	T
T			T		T	T	T	T	T		T			T
T	T	T	T		T		T		T	P	T	T	T	T
T			T						T		T			T
T	T		T		T	T	T	T	T		T		T	T
T														T
T	T	T	T	T	T	T	T	T	T	T	T	T	T	T

Gambar 8. Representasi Denah Gedung dalam Grid *2D*
Source:pribadi

Pada Gambar 8. diatas dapat dipahami bahwa T melambangkan lokasi yang tidak dapat dilewati sebab terdapat tembok disana, E melambangkan posisi pintu keluar yang digunakan untuk lokasi akhir dalam pencarian rute evakuasi, A melambangkan lokasi yang tidak dapat dilewati karena adanya bencana atau kejadian yang tidak diinginkan dan dalam kasus kali ini A melambangkan lokasi api berada. Dan terakhir adalah P yang melambangkan lokasi penghuni itu berada.

Kemudian usai memetakan seluruh komponen dari denah lantai gedung tersebut dalam representasi *matrix of character*, untuk mempermudah dalam pembacaan matriks oleh program maka matriks pada Gambar 8. diubah kembali dengan ketentuan angka 1 sebagai tembok, 0 sebagai tempat yang dapat dilewati dan 2 sebagai tempat pintu keluar dari gedung tersebut. Sedangkan 3 melambangkan tempat dimana lokasi tersebut tidak dapat dilewati dikarenakan bencana atau kejadian yang tidak diinginkan Matriks tersebut kemudian berubah menjadi berikut ini.

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	2	0	1	0	0	1	0	1	0	0	1
1	1	1	1	1	0	1	0	1	1	0	1	0	1	1
1	0	0	0	0	0	0	0	0	1	0	0	0	0	1
1	1	1	1	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	0	0	0	1	0	0	0	0	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	1
1	0	0	0	0	1	1	1	1	1	0	1	0	0	1
1	1	1	1	0	1	0	1	0	1	0	1	1	1	1
1	0	1	0	0	0	0	0	0	1	0	1	0	0	1
1	0	0	0	0	1	1	1	1	1	0	1	0	1	1
1	0	1	0	0	0	0	3	0	0	0	0	0	0	1
1	1	1	1	0	1	1	1	1	1	0	1	1	1	1
1	0	0	0	0	0	0	0	0	1	0	0	0	0	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	1
1	0	0	1	0	1	1	1	1	1	0	1	0	0	1
1	1	1	1	0	1	0	1	0	1	0	1	1	1	1
1	0	0	1	0	0	0	0	0	1	0	1	0	0	1
1	1	0	1	0	1	1	1	1	1	0	1	0	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Gambar 9. Representasi Denah Gedung dalam Matriks Angka
Source:pribadi

B. Penentuan Fungsi Heuristic

Fungsi heuristik sudah dijelaskan sebelum bahwa fungsi *heuristic* ini sangat menentukan keefektifan dari algortima *A** ini. Oleh sebab itu diperlukan fungsi *Heuristic* yang cocok untuk permasalah kali ini.

Sudah diketahui bahwa terdapat 3 buah jenis fungsi *heuristic* yang berhubungan dengan permasalahan pencarian jarak terdekat yaitu *Euclidean*, *Chebyshev*, dan *Manhattan distance*.

Euclidean akan menghasilkan jalur yang tepat apabila jalur penelusuran tidak hanya bergerak secara vertikal dan horizontal, akan tetapi dapat bergerak secara diagonal.

Chebyshev akan menghasilkan jalur yang tepat apablia jalur penelurusan tidak hanya bergerak secara vertikal dan horizontal, akan tetapi juga dapat bergerak secara diagonal dan *cost* untuk setiap langkahnya sama, berbeda dengan *Euclidean*, dimana untuk *cost* untuk melakukan gerak diagonal lebih besar daripada melakukan gerak vertikal dan horizontal.

Manhattan akan menghasilkan jalur yang tepat apabila jalur penelusuran hanya dapat bergerak secara vertikal dan horizontal saja.

Setelah mengetahui dan memahami ketiga fungsi *heuristic*, penulis memutuskan untuk menggunakan *Manhattam Distance* untuk melengkapi algoritma *A** ini.

C. Algortima Penentuan Rute Evakuasi

Setelah mendapatkan semua data berkaitan dengan data peta, titik akhir dan lokasi yang tidak dapat dilewati karena adanya bencana atau kejadian yang tidak diingini, tahap berikutnya adalah merancang algoritma A* untuk menentukan jalur evakuasi yang cepat dan aman bagi penghuni atau pengunjung gedung.

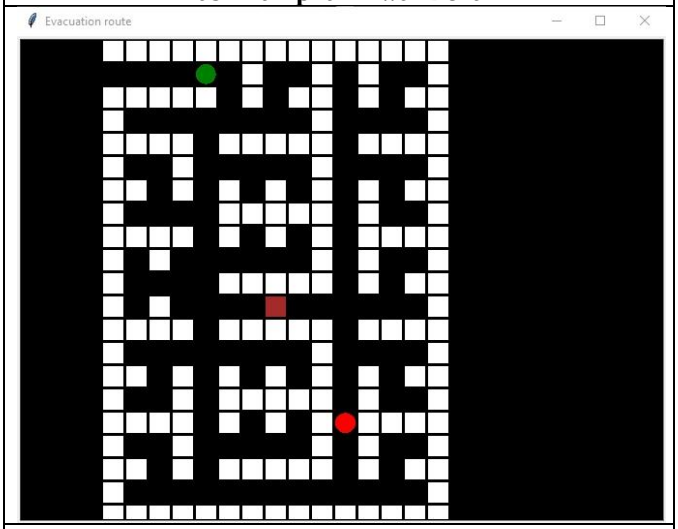
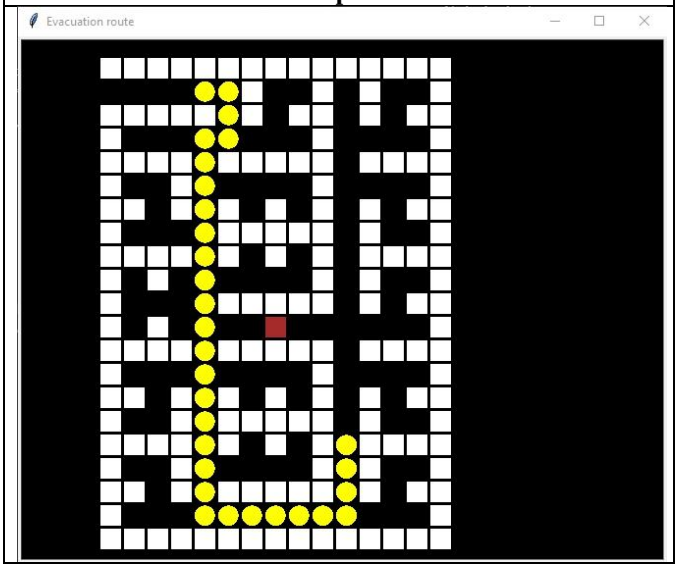
Algoritma A* pada penentuan rute evakuasi ini akan menelusuri setiap kotak dari titik awal penghuni atau pengunjung berada hingga ke titik akhir yaitu pintu keluar darurat dan akan mengembalikan jalur yang terpendek. Berikut adalah kode dari algoritma A* dalam pencarian rute terpendek untuk melakukan evakuasi.

```
FUNCTION AStarSearch(self, start, goal):
    g <- {start:0}
    temp_f <-  heuristic(start,goal) + g[start]
    f <- {start:temp_f}
    closedSet <- []
    openSet <- []
    openSet.append(start)
    cameFrom <- {}
    route <- []
    directions <- [(1,0),(0,1),(-1,0),(0,-1)]
    while (openSet):
      currentPost <- openSet[0]
      for post in openSet :
        IF (f[post]<f[currentPost]):
          currentPost= post
        ENDIF
      ENDFOR
      IF(currentPost==goal):
        route <-  construct_route(cameFrom, currentPost)
        break
      ENDIF
      openSet.remove(currentPost)
      closedSet.append(currentPost)
      for direction in directions:
        tempPost <- tuple(map(operator.add,currentPost,direction))
        IF ( isInMaze(tempPost)):
          IF ( isNotWall(tempPost)):
            IF (tempPost in closedSet):
              continue
            ENDIF
            temp_g <- g[currentPost]+1
            IF (tempPost not in openSet):
              openSet.append(tempPost)
            ELSE:
              continue
            ENDIF
            cameFrom[tempPost] <- currentPost
            g[tempPost] <- temp_g
            f[tempPost] <- g[tempPost] + heuristic(tempPost,goal)
          ENDIF
        ENDIF
    ENDWHILE
    ENDFOR
    RETURN route

  FUNCTION construct_route(self,cameFrom, currentPost):
    route <- [currentPost]
    while currentPost in cameFrom:
      currentPost <- cameFrom[currentPost]
      route.append(currentPost)
    ENDWHILE
    RETURN route
  ENDFUNCTION

  FUNCTION heuristic(self,post,goal):
    RETURN abs(post[0]-goal[0]) + abs(post[1]-goal[1])
  ENDFUNCTION
```

*D. Pengujian Algoritma A**

Hasil dari pengujian diatas dengan contoh persoalan yang sudah disebutkan sebelumnya, yaitu perancangan jalur evakuasi dari gedung bertingkat saat terjadi bencana berupa kebakaran menghasilkan hasil sebagai berikut.

Matriks Peta
111111111111111
000020100101001
111110101101011
100000000100001
111101111101111
100100000100001
110101010101011
100001111101001
111101010101111
101000000101001
100001111101011
101000030000001
111101111101111
100000000100001
110101010101011

100101111101001 111101010101111 100100000101001 110101111101011 100000000000001 111111111111111
Lokasi Penghuni
Lokasi penghuni terletak di (16,10)
Hasil Tampilan Awal Denah

Hasil Tampilan Jalur


Dapat dilihat pada tabel diatas bahwa Algortima A* berhasil membuat suatu jalur evakuasi yang cepat dan aman dimana rute yang dihasilkan berhasil menghindari titik berbahaya yang dilambangkan denga kotak berwarna coklat. Sebagai tambahan lingkaran hijau melambangkan pintu keluar darurat dan lingkaran berwanra merah melambangkan posisi awal penhuni ataupun pengunjung gedung tersebut.

IV. KESIMPULAN

Kesimpulan yang dapat diambil dalam makalah ini adalah Algoritma A* terbukti dapat membantu menyelesaikan pembuatan jalur evakuasi bencana pada suatu gedung dengan baik. Algoritma A* juga dapat berjalan dengan baik ketika fungsi *heuristik* yang dipakai sesuai dengan permasalahan yang ada. Jadi dalam penentuan jalur evakuasi bencana pada gedung terutama gedung-gedung yang rawan terjadi bencana disarankan menerapkan algoritma A* dalam penentuan jalur evakuasi yang cepat dan aman.

REFERENCES

[1] Route/Path Planning Using A Star dan UCS
http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/A-Star-Best-FS-dan-UCS-(2018).pdf
Diakses pada tanggal 25 April 2019

[2] A* Algorithm
http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html
Diakses pada tanggal 25 April 2019

[3] Euclidean vs Chebyshev vs Manhattan Distance
https://lyfat.wordpress.com/2012/05/22/euclidean-vs-chebyshev-vs-manhattan-distance/
Diakses pada tanggal 25 April 2019

[4] Is A*, BFS, Dijkstra, or DFS the best? Why?
https://www.quora.com/Is-A*-BFS-Dijkstra-or-DFS-the-best-Why
Diakses pada tanggal 25 April 2019

[5] Penerapan Algoritma A* (A Star) Sebagai Solusi Pencarian Rute Terpendek Pada Maze
https://www.academia.edu/29753048/Penerapan_Algoritma_A_A_Star_Sebagai_Solusi_Pencarian_Rute_Terpendek_Pada_Maze
Diakses pada tanggal 25 April 2019

[6] Algoritma A* (A-Star)
https://piptools.net/algoritma-a-star/
Diakses pada tanggal 25 April 2019

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, 25 April 2019

Gama Pradipta Wirawan