

BAB III ANALISIS

Bab ini akan membahas analisis dari pembangunan aplikasi yang akan dilakukan, yang meliputi pembangunan labirin, teknik penelusuran labirin, implementasi algoritma *Collision Solver*, serta implementasi sistem multiagen.

3.1 Analisis Pemilihan Algoritma untuk Aplikasi MAS dan Agen Tunggal Pemandang

Analisis yang terhadap metode-metode yang telah disebutkan pada subbab Algoritma Pencarian Rute Terdekat pada Labirin yakni sebagai berikut:

1. *Wall Follower* dan *Recursive Backtracker* tidak selalu memberikan solusi yang terbaik. Hal ini tidak sesuai dengan tujuan dari Tugas Akhir ini, dimana yang solusi diinginkan adalah solusi terbaik (terpendek).
2. Algoritma A^* tidak dapat diterapkan untuk sistem multiagen. Ada beberapa implementasi algoritma A^* yang digunakan untuk permainan komputer modern yang mengimplementasikan sistem multiagen, tetapi multiagen yang diimplementasikan pada aplikasi tersebut hanya dimaksudkan untuk tidak berbenturan satu sama lainnya, dan setiap agen yang ada memiliki posisi awal dan akhirnya masing-masing. Hal ini tidak sesuai dengan sifat labirin yang merupakan penelusuran dari posisi awal hingga akhir, dimana tidak memperhatikan benturan yang terjadi pada pemain yang berada dalam labirin tersebut.
3. Algoritma *Collision Solver* bekerja dengan prinsip *point-to-point* sesuai dengan batasan masalah. Selanjutnya algoritma ini juga dapat menemukan solusi terbaik dan dapat diterapkan dengan menggunakan sistem multiagen.

Berdasarkan analisis di atas, maka algoritma yang akan diimplementasikan pada aplikasi yang akan dibangun adalah *Collision Solver*.

Selanjutnya algoritma yang dipilih untuk aplikasi agen tunggal pemandang yaitu adalah A^* . Algoritma ini dipilih karena masih dapat menemukan solusi terbaik selama

nilai heuristiknya tidak melampaui batas *admissible heuristic* dan sifat pergerakannya masih bersifat *point-to-point*. Selain itu, algoritma *Collision Solver* adalah variasi algoritma A^* yang tidak menggunakan fungsi heuristic, jadi kedua algoritma tersebut sifatnya sejajar dan dapat diperbandingkan performansinya dengan baik [PUL07].

3.2 Analisis Pemanfaatan A^* dalam Pencarian Rute Terpendek

Algoritma A^* akan digunakan sebagai algoritma pembanding performansi untuk aplikasi agen tunggal terhadap penggunaan MAS dalam pencarian rute terdekat. Berdasarkan deskripsi dari prinsip kerja yang dilakukan oleh algoritma A^* pada subbab 2.4.1, maka metode implementasi algoritma tersebut yaitu sebagai berikut:

1. Pada setiap pemilihan grid berikutnya, maka nilai $f(n)$ akan dievaluasi dan dipilih bobot yang terkecil. Jika ada nilai yang sama, maka akan dipilih berdasarkan nilai $g(n)$ yang terbesar.
2. Nilai $f(n)$ adalah nilai yang merepresentasikan bobot yang diperlukan untuk perpindahan dari satu grid ke grid lainnya. Nilai ini didapatkan dengan menambahkan nilai $g(n)$ dan $h(n)$.
3. Nilai $g(n)$ adalah nilai yang merepresentasikan jarak yang telah ditempuh dari grid awal hingga mencapai grid saat ini. Nilai ini didapatkan dengan menghitung jarak yang telah dilalui dari grid awal hingga grid saat ini.
4. Nilai $h(n)$ adalah nilai yang merepresentasikan perkiraan jarak dari grid saat ini hingga grid akhir. Heuristik yang akan digunakan adalah *Manhattan Distance* karena paling cocok dengan lingkungan labirin dibandingkan dengan jenis heuristik lainnya.
5. Walaupun sudah mencapai grid akhir, jika masih ada grid yang memiliki nilai $f(n)$ yang lebih kecil, maka grid tersebut akan terus dipilih sampai bobotnya jauh lebih besar atau mencapai node akhir dengan bobot yang lebih kecil dibandingkan dengan yang telah mencapai grid akhir.

3.3 Analisis Pemanfaatan MAS dalam Pencarian Rute Terpendek

Implementasi MAS yang akan digunakan dalam Tugas Akhir ini yaitu dengan menggunakan algoritma *Collision Solver*. Berdasarkan deskripsi dari prinsip kerja

yang dilakukan oleh algoritma *Collision Solver* pada subbab 2.4.5, maka metode implementasi algoritma *Collision Solver* dengan menggunakan MAS yaitu sebagai berikut:

1. Pada tahap inisialisasi, hanya ada satu agen yang dibangkitkan. Agen tersebut akan berjalan mulai dari grid awal hingga menemui persimpangan.
2. Saat menemui persimpangan, maka agen tersebut akan memilih salah satu jalur yang tersedia, sedangkan untuk jalur lainnya akan dibangkitkan agen lain untuk menelusurinya. Hal ini dilakukan setiap menemui persimpangan. Agen yang dibangkitkan akan mewarisi seluruh atribut yang ada pada agen generasi sebelumnya. Dan agen yang membangkitkan agen baru tersebut akan dihentikan kerjanya.
3. Jika sebuah agen menemui jalan buntu, maka agen tersebut akan dipaksa untuk menghentikan kerjanya.
4. Jika sebuah agen menelusuri jalur yang sudah dilalui oleh agen lainnya, maka agen tersebut akan dihentikan kerjanya.
5. Data perjalanan akan dicatat untuk setiap agen ke dalam dirinya sendiri dan akan diwariskan kepada agen anaknya untuk penelusuran balik dari posisi tujuan ke posisi awal.
6. Saat sebuah agen mencapai posisi tujuan, maka agen tersebut akan menandai jalur yang telah dilaluinya dari posisi awal hingga akhir.

Algoritma yang menggambarkan kerja dari algoritma *Collision Solver* tersebut dapat dilihat pada Algoritma III-1.

3.4 Deskripsi Umum Aplikasi

Aplikasi yang dibangun dalam Tugas Akhir ini selanjutnya akan dinamakan *LabyrinthModel*. *LabyrinthModel* akan menghasilkan solusi jalur terpendek terhadap labirin yang menjadi input aplikasi. Aplikasi ini akan menampilkan jalur solusi untuk memperlihatkan kepada pengguna jalur solusi terdekat dari labirin tersebut. Data labirin yang digunakan diperoleh dengan terlebih dahulu dibuat oleh pengguna atau menggunakan file data labirin yang telah terintegrasi pada aplikasi.

Langkah pertama dalam penggunaan aplikasi ini yaitu dengan memasukkan parameter yang sesuai dengan data labirin yang digunakan. Selanjutnya, aplikasi akan memproses data labirin beserta parameter yang dimasukkan dan menghasilkan solusi jalur terdekat pada labirin tersebut. Solusi akan ditampilkan dengan menandai seluruh grid yang dilalui oleh jalur terpendek pada labirin tersebut. Deskripsi dari hal ini dapat dilihat pada Gambar III-1.

```

Procedure CollisionSolver (input: P: Point, D: Distance, LN:
    ListOfNode)
{
    Mencari jalur terdekat antara posisi awal hingga posisi akhir
}

Deklarasi:
LN      : ListOfNode
moveList : LinkedList
A       : Agent
P, F    : Point
i, D    : Integer

Algoritma
for (i == 0; i < agentList.size; i++)
    //Memeriksa 4 grid tetangga, atas=(x, y-1), kanan=(x+1, y),
    bawah=(x, y+1), kiri=(x-1, y)
    if (getObjectAt(Neighbor) == 8)
        moveList.add()

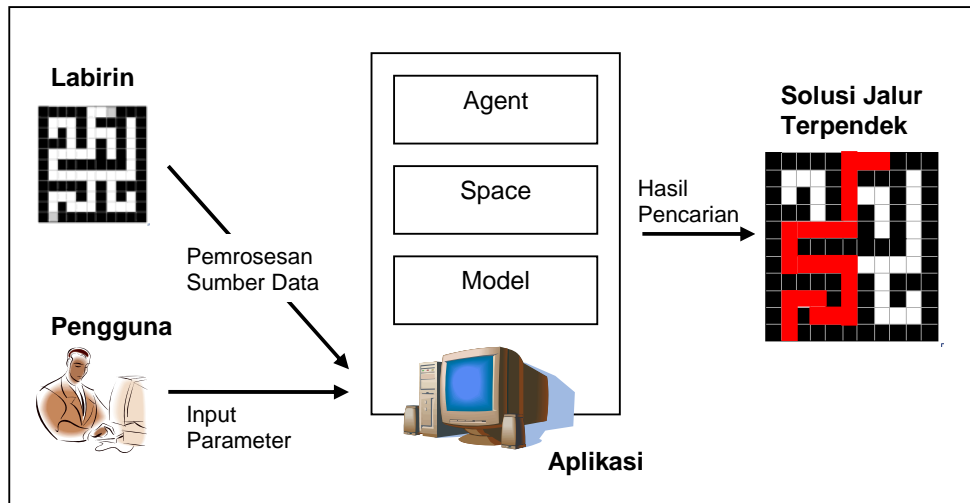
    if (moveList.size == 0)
        A.destroy()
    elseif (moveList.size > 1)
        A.createNewAgent(moveList)
        A.destroy()
    elseif (moveList.size == 1)
        A.move(moveList)

    D++

    //Jika agen telah mencapai posisi tujuan, maka proses kerja
model berakhir
    if (P == F)
        solution(L)
        finish()
    else
        CollisionSolver()

```

Algoritma III-1: Algoritma Collision Solver



Gambar III-1 : Deskripsi Umum Aplikasi

3.5 Analisis Struktur Data Agen

Berdasarkan Algoritma III-1, maka struktur data dari agen yang akan digunakan dalam pemodelan meliputi atribut sebagai berikut:

1. Posisi saat ini. Atribut ini diperlukan sebagai penanda posisi agen saat ini, dan terbagi menjadi dua nilai, yaitu x dan y, yaitu posisi relatif dengan posisi (x, y) pada grid labirin.
2. Jarak yang telah ditempuh. Atribut ini mencatat jarak yang telah ditempuh oleh sebuah agen dalam penelusuran labirin. Atribut ini diwarisi dari agen induknya, sehingga saat sebuah agen baru lahir, maka nilai dari atribut ini tidak akan bernilai nol dan akan meneruskan nilai jarak tempuh yang sudah ada.
3. Daftar node yang telah dikunjungi. Atribut ini berisi seluruh grid yang telah dikunjungi dalam penelusuran labirin. Atribut ini juga diwarisi dari agen induknya, sehingga saat sebuah agen baru lahir, isi dari atribut ini akan memuat daftar grid yang telah dikunjungi oleh agen induknya.

3.6 Analisis Struktur Data Labirin

Dalam pembangunan labirin, terdapat beberapa hal yang perlu dijadikan pertimbangan, yaitu antara lain:

1. Ukuran area labirin. Hal ini sangat penting karena menyangkut besar memori yang diperlukan dan lama pencarian solusi.
2. Struktur data labirin. Hal ini penting karena struktur data tersebut merupakan sumber informasi yang digunakan dalam pembangunan labirin serta pencarian solusi dari labirin tersebut.
3. Jenis *routing* labirin. Hal ini diperlukan untuk menentukan jenis labirin yang akan dibangun.

Ketiga hal di atas akan dijelaskan lebih lanjut pada bagian berikut ini.

3.6.1 Ukuran Area Labirin

Ukuran area dari sebuah labirin harus diperhitungkan dengan cermat. Hal ini menyangkut kepada jumlah memori yang diperlukan saat membangun labirin serta saat mencari solusi dari labirin tersebut. Selain itu, ukuran labirin yang terlalu besar akan memakan waktu yang terlalu lama dalam pencarian solusi, sedangkan ukuran labirin yang terlalu kecil akan menghilangkan efektivitas dari algoritma yang digunakan untuk mencari solusi.

Labirin yang digunakan akan berbentuk grid dua dimensi, dimana setiap grid adalah posisi yang berjarak 1 satuan dari posisi sebelumnya. Batasan ukuran area yang digunakan adalah 15x15 grid. Pertimbangan akan hal ini yaitu untuk menyamakan parameter ukuran area dengan kaskas pengujian yang memiliki ukuran area yang sama.

3.6.2 Representasi Data Labirin

Struktur data dari labirin akan menggunakan struktur data yang digunakan oleh citra PGM. Hal ini disebabkan karena Repast memiliki fungsi untuk memproses dan memanipulasi citra dalam struktur PGM.

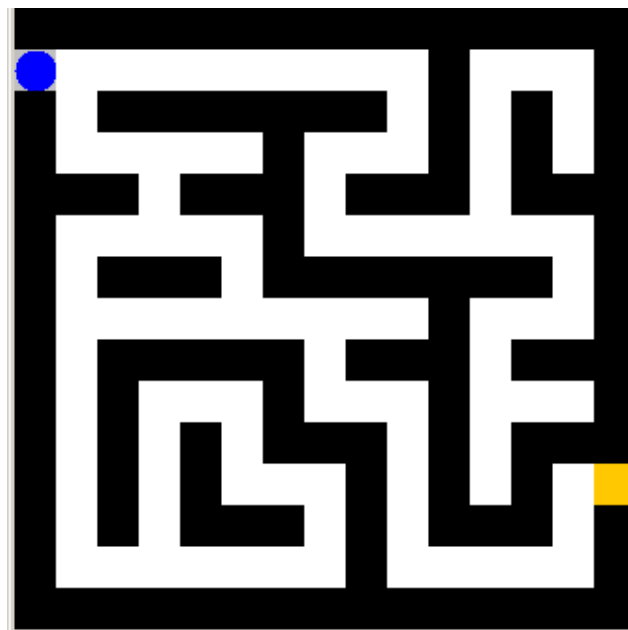
Contoh dari struktur data labirin yang akan dibuat dapat dilihat pada Gambar III-2 berikut, dan representasi visualnya pada Gambar III-3. Pada algoritma tersebut, data labirin pertama ditandai oleh kode khusus, selanjutnya ukuran panjang dan lebar area, dan intensitas maksimum dari pixel. Setelah itu barulah berisi data dari labirin yang akan dibangun.

Pada matriks data, angka 0 merepresentasikan dinding labirin, angka 1 merepresentasikan jalur labirin yang telah ditelusuri, angka 2 merepresentasikan jalur solusi labirin, angka 3 merepresentasikan posisi awal dan akhir dari labirin, dan angka 8 merepresentasikan jalur labirin yang belum ditelusuri.

Sedangka secara visual, warna hitam adalah dinding, putih adalah jalur yang belum ditelusuri, abu-abu adalah jalur yang telah ditelusuri, jingga adalah letak posisi awal/akhir.

P2
15 15
8
0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 8 8 8 0 8 8 8 8 8 8 8 8 8 8 0
0 8 0 8 0 8 0 8 0 0 0 0 0 0 8 0
0 8 0 8 8 8 0 8 0 8 8 8 8 8 8 0
0 8 0 8 0 8 0 8 0 8 0 8 0 0 0 8 0
0 8 0 8 0 8 8 8 0 8 8 8 0 8 8 0
0 8 0 0 0 0 0 8 0 0 0 8 0 8 0
0 8 0 8 8 8 0 8 8 8 0 8 8 8 8 0
0 8 0 8 0 8 0 8 0 8 0 8 0 0 0 0
0 8 8 8 0 8 0 8 0 8 0 8 8 8 8 8 0
0 0 0 0 0 8 0 0 0 0 0 0 0 0 8 0
0 8 8 8 8 8 0 8 8 8 8 8 8 8 8 8 0
0 8 0 0 0 8 0 8 0 8 0 8 0 0 0 8 0
0 8 8 8 0 8 8 8 0 8 0 8 8 8 8 8 0
0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0

Gambar III-2: Contoh Citra PGM



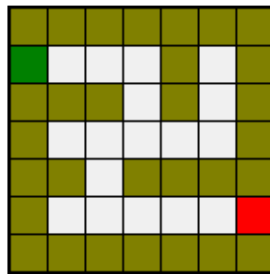
Gambar III-3: Representasi Visual dari Gambar III-2

3.6.3 Jenis Routing Labirin

Jenis routing labirin yang akan diimplementasikan yaitu bersifat *semi-perfect*. Yang dimaksud dengan *semi-perfect* yaitu merupakan gabungan antara *perfect* dan *braid*, dimana labirin masih memiliki kemungkinan adanya jalur buntu, loop dan daerah isolasi. Pertimbangan hal ini yaitu untuk menerapkan sebanyak mungkin kemungkinan situasi yang dihadapi oleh sistem multiagen yang akan dibangun tersebut.

3.7 Analisis Contoh Kasus

Gambar III-4 adalah gambar dari contoh kasus labirin untuk kedua jenis algoritma yang akan diimplementasikan:



Gambar III-4 : Contoh Kasus Labirin

3.7.1 Analisis Contoh Kasus untuk *Collision Solver*

Langkah-langkah yang terjadi pada algoritma shortest path finder untuk kasus labirin di atas adalah sebagai berikut ini:

1. Sebagai inisialisasi, sebuah agen (A1) akan dibangkitkan pada grid (0,1) sehingga $cost(A1) = 0$ dan $ListNode = \{(0,1)\}$.
2. Selanjutnya, A1 akan bergerak ke grid (1,1) sehingga $cost(A1) = 1$ dan $ListNode = \{(0,1), (1,1)\}$.
3. Selanjutnya, A1 akan bergerak ke grid (2,1) sehingga $cost(A1) = 2$ dan $ListNode = \{(0,1), (1,1), (2,1)\}$.
4. Selanjutnya, A1 akan bergerak ke grid (3,1) sehingga $cost(A1) = 3$ dan $ListNode = \{(0,1), (1,1), (2,1), (3,1)\}$.
5. Selanjutnya, A1 akan bergerak ke grid (3,2) sehingga $cost(A1) = 4$ dan $ListNode = \{(0,1), (1,1), (2,1), (3,1), (3,2)\}$.

6. Selanjutnya, A1 akan bergerak ke grid (3,3) sehingga $\text{cost}(A1) = 5$ dan $\text{ListNode} = \{(0,1), (1,1), (2,1), (3,1), (3,2), (3,3)\}$.
7. Karena terdapat dua kemungkinan langkah pada grid (3,3), maka akan dibangkitkan agen A2 dan A3:
 - a. A2 akan bergerak ke grid (2,3) sehingga $\text{cost}(A2) = 6$ dan $\text{ListNode} = \{(0,1), (1,1), (2,1), (3,1), (3,2), (3,3), (2,3)\}$.
 - b. A3 akan bergerak ke grid (4,3) sehingga $\text{cost}(A3) = 6$ dan $\text{ListNode} = \{(0,1), (1,1), (2,1), (3,1), (3,2), (3,3), (4,3)\}$.
 - c. Agen A1 akan dimatikan.
8. Karena terdapat dua kemungkinan langkah pada grid (2,3), maka akan dibangkitkan agen A4 dan A5:
 - a. A4 akan bergerak ke grid (1,3) sehingga $\text{cost}(A4) = 7$ dan $\text{ListNode} = \{(0,1), (1,1), (2,1), (3,1), (3,2), (3,3), (2,3), (1,3)\}$.
 - b. A5 akan bergerak ke grid (2,4) sehingga $\text{cost}(A5) = 7$ dan $\text{ListNode} = \{(0,1), (1,1), (2,1), (3,1), (3,2), (3,3), (2,3), (2,4)\}$.
 - c. A3 akan bergerak ke grid (5,3) sehingga $\text{cost}(A3) = 7$ dan $\text{ListNode} = \{(0,1), (1,1), (2,1), (3,1), (3,2), (3,3), (4,3), (5,3)\}$.
 - d. Agen A2 akan dimatikan.
9. Agen A4 pada grid (1,3) tidak dapat bergerak, maka akan dimatikan:
 - a. A3 akan bergerak ke grid (5,2) sehingga $\text{cost}(A3) = 8$ dan $\text{ListNode} = \{(0,1), (1,1), (2,1), (3,1), (3,2), (3,3), (4,3), (5,3), (5,2)\}$.
 - b. Agen A4 akan dimatikan.
 - c. A5 akan bergerak ke grid (2,5) sehingga $\text{cost}(A5) = 8$ dan $\text{ListNode} = \{(0,1), (1,1), (2,1), (3,1), (3,2), (3,3), (2,3), (2,4), (2,5)\}$.
10. Karena terdapat dua kemungkinan langkah pada grid (2,5), maka akan dibangkitkan agen A6 dan A7:
 - a. A3 akan bergerak ke grid (5,1) sehingga $\text{cost}(A3) = 9$ dan $\text{ListNode} = \{(0,1), (1,1), (2,1), (3,1), (3,2), (3,3), (4,3), (5,3), (5,2), (5,1)\}$.
 - b. A6 akan bergerak ke grid (1,5) sehingga $\text{cost}(A6) = 9$ dan $\text{ListNode} = \{(0,1), (1,1), (2,1), (3,1), (3,2), (3,3), (2,3), (2,4), (2,5), (1,5)\}$.
 - c. A7 akan bergerak ke grid (3,5) sehingga $\text{cost}(A7) = 9$ dan $\text{ListNode} = \{(0,1), (1,1), (2,1), (3,1), (3,2), (3,3), (2,3), (2,4), (3,5)\}$.

- d. Agen A5 akan dimatikan.
11. Agen A3 dan A6 sudah tidak dapat bergerak lagi, maka:
 - a. Agen A3 akan dimatikan.
 - b. Agen A6 akan dimatikan.
 - c. A7 akan bergerak ke grid (4,5) sehingga $\text{cost}(A7) = 10$ dan $\text{ListNode} = \{(0,1), (1,1), (2,1), (3,1), (3,2), (3,3), (2,3), (2,4), (3,5), (4,5)\}$.
 12. Selanjutnya, A7 akan bergerak ke grid (5,5) sehingga $\text{cost}(A7) = 11$ dan $\text{ListNode} = \{(0,1), (1,1), (2,1), (3,1), (3,2), (3,3), (2,3), (2,4), (3,5), (4,5), (5,5)\}$.
 13. Selanjutnya, A7 akan bergerak ke grid (6,5) sehingga $\text{cost}(A7) = 12$ dan $\text{ListNode} = \{(0,1), (1,1), (2,1), (3,1), (3,2), (3,3), (2,3), (2,4), (3,5), (4,5), (5,5), (6,5)\}$.
 14. Karena agen A7 sudah mencapai node akhir dan agen lainnya sudah tidak bekerja lagi, maka seluruh proses kerja akan selesai dengan total jarak tempuh 12 melalui jalur $\{(0,1), (1,1), (2,1), (3,1), (3,2), (3,3), (2,3), (2,4), (3,5), (4,5), (5,5), (6,5)\}$.

3.7.2 Analisis Contoh Kasus untuk A*

Langkah-langkah yang terjadi pada algoritma A* untuk kasus labirin di atas dengan menggunakan heuristik *Manhattan Distance* adalah sebagai berikut ini:

1. Sebagai inisialisasi, posisi awal adalah pada grid (0,1).
 $\{(0,1) \rightarrow (1,1): g(x) = 1; h(x) = (\text{abs}(1-6) + \text{abs}(1-5)) = 9; f(x) = 10\} \rightarrow \text{Go}$
2. Posisi saat ini adalah (1,1):
 $\{(1,1) \rightarrow (2,1): g(x) = 2; h(x) = (\text{abs}(2-6) + \text{abs}(1-5)) = 8; f(x) = 10\} \rightarrow \text{Go}$
3. Posisi saat ini adalah (2,1):
 $\{(2,1) \rightarrow (3,1): g(x) = 3; h(x) = (\text{abs}(3-6) + \text{abs}(1-5)) = 7; f(x) = 10\} \rightarrow \text{Go}$
4. Posisi saat ini adalah (3,1):
 $\{(3,1) \rightarrow (3,2): g(x) = 4; h(x) = (\text{abs}(3-6) + \text{abs}(2-5)) = 6; f(x) = 10\} \rightarrow \text{Go}$
5. Posisi saat ini adalah (3,2):
 $\{(3,2) \rightarrow (3,3): g(x) = 5; h(x) = (\text{abs}(3-6) + \text{abs}(3-5)) = 5; f(x) = 10\} \rightarrow \text{Go}$
6. Posisi saat ini adalah (3,3):
 $\{(3,3) \rightarrow (2,3): g(x) = 6; h(x) = (\text{abs}(2-6) + \text{abs}(3-5)) = 6; f(x) = 12\}$

$$\{(3,3) \rightarrow (4,3): g(x) = 6; h(x) = (\text{abs}(4-6) + \text{abs}(3-5)) = 4; f(x) = 10\} \rightarrow \text{Go}$$

7. Posisi saat ini adalah (4,3):

$$\{(3,3) \rightarrow (2,3): g(x) = 6; h(x) = (\text{abs}(3-6) + \text{abs}(2-5)) = 6; f(x) = 12\}$$

$$\{(4,3) \rightarrow (5,3): g(x) = 7; h(x) = (\text{abs}(5-6) + \text{abs}(3-5)) = 3; f(x) = 10\} \rightarrow \text{Go}$$

8. Posisi saat ini adalah (5,3):

$$\{(3,3) \rightarrow (2,3): g(x) = 6; h(x) = (\text{abs}(2-6) + \text{abs}(3-5)) = 6; f(x) = 12\} \rightarrow \text{Go}$$

$$\{(5,3) \rightarrow (5,2): g(x) = 8; h(x) = (\text{abs}(5-6) + \text{abs}(2-5)) = 4; f(x) = 12\}$$

9. Posisi saat ini adalah (2,3):

$$\{(2,3) \rightarrow (1,3): g(x) = 7; h(x) = (\text{abs}(1-6) + \text{abs}(3-5)) = 7; f(x) = 14\}$$

$$\{(2,3) \rightarrow (2,4): g(x) = 7; h(x) = (\text{abs}(2-6) + \text{abs}(4-5)) = 5; f(x) = 12\}$$

$$\{(5,3) \rightarrow (5,2): g(x) = 8; h(x) = (\text{abs}(5-6) + \text{abs}(2-5)) = 4; f(x) = 12\} \rightarrow \text{Go}$$

10. Posisi saat ini adalah (5,2):

$$\{(2,3) \rightarrow (1,3): g(x) = 7; h(x) = (\text{abs}(1-6) + \text{abs}(3-5)) = 7; f(x) = 14\}$$

$$\{(2,3) \rightarrow (2,4): g(x) = 7; h(x) = (\text{abs}(2-6) + \text{abs}(4-5)) = 5; f(x) = 12\} \rightarrow \text{Go}$$

$$\{(5,2) \rightarrow (5,1): g(x) = 9; h(x) = (\text{abs}(5-6) + \text{abs}(1-5)) = 5; f(x) = 14\}$$

11. Posisi saat ini adalah (2,4):

$$\{(2,3) \rightarrow (1,3): g(x) = 7; h(x) = (\text{abs}(1-6) + \text{abs}(3-5)) = 7; f(x) = 14\}$$

$$\{(2,4) \rightarrow (2,5): g(x) = 8; h(x) = (\text{abs}(2-6) + \text{abs}(5-5)) = 4; f(x) = 12\} \rightarrow \text{Go}$$

$$\{(5,2) \rightarrow (5,1): g(x) = 9; h(x) = (\text{abs}(5-6) + \text{abs}(1-5)) = 5; f(x) = 14\}$$

12. Posisi saat ini adalah (2,5):

$$\{(2,3) \rightarrow (1,3): g(x) = 7; h(x) = (\text{abs}(1-6) + \text{abs}(3-5)) = 7; f(x) = 14\}$$

$$\{(2,5) \rightarrow (1,5): g(x) = 9; h(x) = (\text{abs}(1-6) + \text{abs}(5-5)) = 5; f(x) = 14\}$$

$$\{(2,5) \rightarrow (3,5): g(x) = 9; h(x) = (\text{abs}(3-6) + \text{abs}(5-5)) = 3; f(x) = 12\} \rightarrow \text{Go}$$

$$\{(5,2) \rightarrow (5,1): g(x) = 9; h(x) = (\text{abs}(5-6) + \text{abs}(1-5)) = 5; f(x) = 14\}$$

13. Posisi saat ini adalah (3,5):

$$\{(2,3) \rightarrow (1,3): g(x) = 7; h(x) = (\text{abs}(1-6) + \text{abs}(3-5)) = 7; f(x) = 14\}$$

$$\{(2,5) \rightarrow (1,5): g(x) = 9; h(x) = (\text{abs}(1-6) + \text{abs}(5-5)) = 5; f(x) = 14\}$$

$$\{(3,5) \rightarrow (4,5): g(x) = 10; h(x) = (\text{abs}(4-6) + \text{abs}(5-5)) = 2; f(x) = 12\} \rightarrow \text{Go}$$

$$\{(5,2) \rightarrow (5,1): g(x) = 9; h(x) = (\text{abs}(5-6) + \text{abs}(1-5)) = 5; f(x) = 14\}$$

14. Posisi saat ini adalah (4,5):

$$\{(2,3) \rightarrow (1,3): g(x) = 7; h(x) = (\text{abs}(1-6) + \text{abs}(3-5)) = 7; f(x) = 14\}$$

$$\{(2,5) \rightarrow (1,5): g(x) = 9; h(x) = (\text{abs}(1-6) + \text{abs}(5-5)) = 5; f(x) = 14\}$$

$\{(4,5) \rightarrow (5,5): g(x) = 11; h(x) = (\text{abs}(5-6) + \text{abs}(5-5)) = 1; f(x) = 12\} \rightarrow \text{Go}$

$\{(5,2) \rightarrow (5,1): g(x) = 9; h(x) = (\text{abs}(5-6) + \text{abs}(1-5)) = 5; f(x) = 14\}$

15. Posisi saat ini adalah (5,5):

$\{(2,3) \rightarrow (1,3): g(x) = 7; h(x) = (\text{abs}(1-6) + \text{abs}(3-5)) = 7; f(x) = 14\}$

$\{(2,5) \rightarrow (1,5): g(x) = 9; h(x) = (\text{abs}(1-6) + \text{abs}(5-5)) = 5; f(x) = 14\}$

$\{(5,5) \rightarrow (6,5): g(x) = 12; h(x) = (\text{abs}(6-6) + \text{abs}(5-5)) = 0; f(x) = 12\} \rightarrow \text{Go}$

$\{(5,2) \rightarrow (5,1): g(x) = 9; h(x) = (\text{abs}(5-6) + \text{abs}(1-5)) = 5; f(x) = 14\}$

16. Karena sudah mencapai posisi tujuan dan nilai $f(x)$ -nya adalah yang paling minimum, maka proses pencarian selesai dengan jalur terdekat: $\{(0,1), (1,1), (2,1), (3,1), (3,2), (3,3), (2,3), (2,4), (3,5), (4,5), (5,5), (6,5)\}$ dengan jarak 12.

3.7.3 Analisis Perbandingan Hasil Contoh Kasus

Berdasarkan perbandingan kedua hasil algoritma di atas, sementara dapat disimpulkan bahwa algoritma *Collision Solver* memiliki performansi yang lebih baik daripada A^* . Berdasarkan jumlah langkah pengerjaan dapat dilihat bahwa algoritma *Collision Solver* akan menemukan solusi dengan jumlah langkah terpendek yang mungkin, sedangkan proses pengerjaan pada A^* dapat melebar karena menggunakan fungsi heuristik yang dapat memperlambat kinerjanya dalam menemukan solusi masalah.

Analisa yang disebutkan di atas akan diujicobakan dalam implementasi perangkat lunak yang akan dibangun untuk diuji kebenarannya.