Penerapan Algoritma Pencarian A* dalam Sistem Pathfinding Mob pada Game Minecraft

Aryo Bama Wiratama - 13523088[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]aryobama09@gmail.com, [2]13523088@std.stei.itb.ac.id

Abstract --- Minecraft merupakan permainan video bergenre sandbox 3D yang populer di kalangan pemain game. Salah satu fitur menarik dalam permainan ini adalah keberadaan mob (karakter non-pemain) yang dapat berinteraksi dan bergerak secara otomatis. Makalah ini membahas implementasi algoritma pencarian A* (A star) dalam menemukan jalur terpendek pergerakan mob pada Minecraft. Analisis dilakukan dengan memodelkan area permainan sebagai graf berbobot, di mana setiap simpul merepresentasikan posisi yang dapat dilalui dan sisi merepresentasikan jalur yang menghubungkan antar posisi. Fungsi heuristik pada algoritma A* dihitung menggunakan jarak Manhattan. Hasil eksperimen menunjukkan bahwa implementasi algoritma A* berhasil menemukan jalur optimal yang identik dengan pergerakan mob dalam permainan. Algoritma A* lebih eifisen dari Djikstra dan BFS dalam melakukan pencarian. Namun, dalam hal kompleksitas perhitungan algortima A* lebih rumit dari kedua algoritma tersebut.

Keywords --- algoritma A star, jarak Manhattan, Minecraft, pencarian jalur, fungsi heuristik

I. PENDAHULUAN

Minecraft adalah sebuah permainan video yang cukup populer di seluruh dunia. Minecraft merupakan permainan video yang bertipe sandbox 3D. Game sandbox adalah game yang tidak memiliki tujuan dan biasanya memberikan kebebasan pemainnya untuk memainkan permainan dengan cara mereka sendiri. Game sandbox biasanya memiliki dunia yang luas sehingga memungkinkan pemainnya untuk menjelajah, membangun, dan berkreasi tanpa ada batasan. Hal itulah yang membuat permainan video minecraft diminati di semua kalangan umur.

Minecraft dikembangkan oleh Markus "Notch" Persson, seorang pengembang game asal Swedia. Minecraft pertama kali dirilis dalam versi alpha pada tanggal 17 Mei 2009 dan pada tanggal 18 November 2011 versi lengkapnya dirilis. Pada tahun 2009, Persson mendirikan Mojang untuk memfasilitasi pengembangan permainan video ini. Pada tahun 2014, Microsoft mengakuisisi Mojang dan hak atas Minecraft dengan nilai sekitar $2,5 miliar. Hingga kini, minecraft menjadi game populer yang berhasil mencapai 300 juta kopi penjualan di seluruh dunia.

Minecraft memiliki dua mode utama, yaitu mode *creative* dan mode *survival*. Dalam mode *creative*, fokus utama pemain adalah berkreasi dan membangun dengan bebas tanpa batasan. Pemain dapat melakukan ekspolarasi, eksperimen, membangun struktur berkreatif yang diinginkan oleh pemain. Mode *creative* memungkinkan pemain menggunakan fitur terbang sehingga pemain memudahkan pemain dalam berkreasi. Sedangkan dalam mode *survival*, fokus utama pemain adalah bertahan hidup. Di mode ini pemain dapat mengumpulkan sumber daya dengan berbagai cara mulai dari menebang pohon, berkebun, hingga menambang. Dalam mode ini terdapat mob yang dapat menyerang pemain. Mob ini biasanya akan keluar pada malam hari sehingga pemain diharuskan untuk membangun tempat perlindungan sebelum malam hari. Pemain juga dapat melakukan ekspolarasi dan petualangan. Terdapat banyak *structure* di minecraft yang dapat dieksplorasi oleh pemain.

Gambar 1.1 Permainan Video Minecraft
(Sumber: https://www.minecraft.net/en-us)

Makalah ini bertujuan untuk menganalisis sistem pathfinding pada mob minecraft menggunakan algoritma pencarian A*.

II. LANDASAN TEORI

A. *Graf*

Graf secara sederhana adalah kumpulan titik yang dihubungkan satu sama lain melalui garis. Titik pada graf biasanya disebut dengan simpul (node) dan garis pada graf biasa disebut dengan sisi/busur (edge). Graf biasanya digunakan untuk menggambarkan elemen – elemen diskrit serta hubungan antara elemen – elemen tersebut.

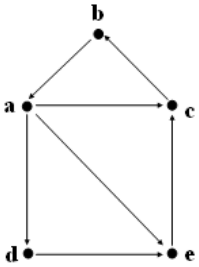
Dalam konteks matematika diskrit, graf G didefinisikan sebagai $G = (V, E)$ di mana:

V = Himpunan tak kosong dari elemen – elemennya yang disebut dengan simpul atau vertex
$= \{v_1, v_2, ..., v_n\}$

Himpunan V tidak boleh kosong, artinya graf G harus memiliki setidaknya satu buah simpul.

E = Himpunan pasangan – pasangan simpul yang disebut dengan sisi atau edge
$= \{e_1, e_2, ..., e_n\}$

Himpunan *E* boleh kosong, artinya graf *G* tidak harus memiliki sisi satu buahpun.



Gambar 2.1 Graf Sederhana *G*
(Sumber: https://mti.binus.ac.id/2018/03/05/teori-graph-sejarah-dan-manfaatnya/)

Gambar 2.1 merupakan contoh graf sederhana *G* dengan komponen sebagai berikut:

G = (V, E)

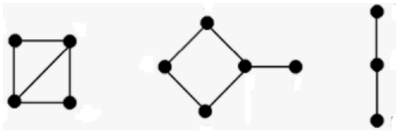
V = {**a, b, c, d, e**}

E = {(**a, b**)**,** (**a, c**)**,** (**a, e**)**,** (**a, d**)**,** (**b, c**)**,** (**c, e**)**,** (**d, e**)}

Berdasarkan hubungan antar simpul, graf dibagi menjadi 2, yaitu graf sederhana dan graf tidak sederhana.

1. Graf sederhana

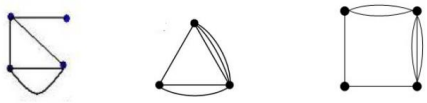
   Graf sederhana adalah graf yang tidak mengandung sisi ganda (satu pasangan simpul dihubungkan dengan lebih dari satu sisi) dan tidak mengandung sisi gelang atau *loop* (sisi yang menghubungkan simpul dengan dirinya sendiri).



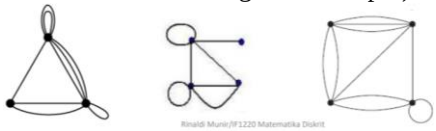
Gambar 2.2 Contoh Graf Sederhana
(Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf)

2. Graf tidak sederhana

   Graf tidak sederhana adalah kebalikan dari graf sederhana, yaitu graf yang memiiliki sisi ganda ATAU sisi gelang. Graf tidak sederhana dibagi lagi menjadi dua, yaitu graf ganda dan graf semu (*pseudograph)*. Graf ganda merupakan graf yang memiliki sisi ganda, sedangkan graf semu merupakan graf yang memiliki sisi gelang.



Gambar 2.3 Contoh Graf Ganda
(Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf)

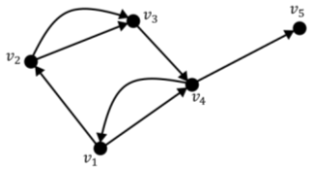


Gambar 2.4 Contoh Graf Semu
(Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf)

Berdasarkan arah sisinya, graf dibedakan menjadi dua, yaitu graf berarah dan graf tak berarah.

1. Graf berarah

   Graf berarah merupakan graf yang setiap sisinya memiliki arahnya masing – masing.



Gambar 2.5 Contoh Graf Berarah
(Sumber: https://mathcyber1997.com/materi-soal-dasar-graf-dan-terminologi/)

2. Graf tak berarah

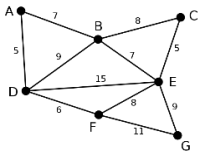
   Berkebalikan dengan graf berarah, graf tak berarah merupakan graf yang tidak memiliki arah, artinya hubungan antar simpul bersifat simetris. Contohnya {**a, b**} = {**b, a**}

Gambar 2.6 Contoh Graf Tak Berarah
(Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

Berdasarkan bobot sisinya, graf dibedakan menjadi dua, yaitu graf berbobot dan graf tak berbobot

1. Graf berbobot

   Graf berbobot merupakan graf yang sisi – sisinya memiliki bobot. Bobot ini biasanya nilai yang menyatakan suatu intensitas, seperti jarak, biaya, dan waktu.



Gambar 2.7 Contoh Graf Berbobot
(Sumber:
https://matrisnowei.blogspot.com/2015/01/algoritma-prim.html)

2. Graf tak berbobot

   Graf tak berbobot merupakan graf yang sisi - sisinya tidak memiliki bobot. Sisinya hanya menunjukkan ada atau tidaknya hubungan antarsimpul. Gambar 2.6 merupakan contoh graf tak berbobot.

Terdapat dua representasi graf yang biasa dipakai, yaitu ketetanggaan (*Adjacent*) dan bersisian (*Incidency*).

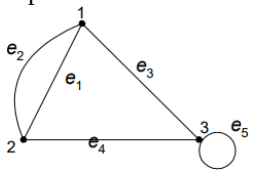
1. Ketetanggan (*Adjacent*)

   Dua buah simpul dikatakan bertetangga (*Adjacent*) apabila terdapat minimal satu buah sisi yang secara langsung menghubungkan kedua simpul tersebut. Graf *G* pada gambar 2.1. terlihat bahwa simpul **d** bertetangga dengan simpul **a** dan **e**, tetapi tidak bertetangga dengan simpul **b** dan **c**.

Terdapat dua cara menyimpan graf dengan rerpresentasi *adjacent*, yaitu *adjacency matrix*, dan *adjacency list.*

2. Bersisian (*Incidency*)

Sebuah sisi dikatakan bersisian atau *incident* dengan sebuah simpul apabila sisi tersebut bertemu atau menjadi penghubung simpul tersebut.



Gambar 2.8 Contoh Graf G_1
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf)

Pada graf G_1, sisi $\mathbf{e_3}$ (1,3) bersisian dengan simpul 1 dan simpul 3, tetapi tidak bersisian dengan simpul 2.

Terdapat dua cara menyimpan graf dengan rerpresentasi *adjacent*, yaitu *incidence matrix*, dan *incidence list.*

B. Jalur Terpendek

Dalam representasi graf, jalur atau lintasan atau rute merupakan urutan simpul yang dihubungkan oleh sisi pada graf. Simpul pada merepresentasikan suatu tempat dan sisi pada graf merepresentasikan suatu jalan yang menghubungkan kedua tempat. Biasanya graf yang digunakan untuk merepresentasikan suatu jalur adalah graf berbobot dengan nilai atau bobot merepresentasikan waktu atau jarak tempuh antartempat. Dalam graf, posisi awal disebut dengan *start node* dan posisi tujuan disebut dengan *goal node*.

Jalur terpendek (*shortest path)* dalam graf adalah jalur yang menghubungkan kedua node, yaitu *start node* dan *goal node* dengan bobot yang paling minimum. Masalah jalur terpendek adalah masalah dalam menemukan jalur antara *start node* dengan *goal node* sedemikian rupa sehingga bobot yang diperlukan seminimal mungkin. Jalur terpendek berfungsi mencari jalur mana yang paling efisien untuk dilalui. Di dunia nyata, jalur terpendek digunakan dalam beberapa sistem, seperti navigasi GPS, jaringan komunikasi dan routing data, transportasi dan logistik, serta pencarian jalur karakter pada permainan video.

*C. Algoritma A**

Algoritma A* (dibaca A star) merupakan salah satu algoritma pencarian jalur terpendek. Pertama kali ditemukan pada tahun 1968 oleh Peter Hart, Nils Nilsson, dan Betram Raphael di Stanford Research Institute (sekarang SRI International).

Algoritma A* merupakan pengembangan dari algoritma pencarian sebelum A*, yaitu algoritma best first search (BFS) dan algortima Dijkstra. A* star menggabungkan dua konsep utama pada Dijkstra dan BFS, yaitu biaya yang telah dikeluarkan dari sumber ke simpul **n** (g(n)) dan perkiraan biaya dari simpul **n** menuju simpul tujuan yang biasanya disebut fungsi heuristik (h(n)). Algoritma A* mengevaluasi kedua biaya tersebut untuk menentukan simpul mana yang akan dilalui. Setiap simpul akan memiliki total biaya yang dapat ditulis sebagai berikut:

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

Catatan:
f(n) = total biaya
g(n) = biaya dari simpul awal ke simpul n
h(n) = perkiraan biaya dari simpul n ke simpul akhir

D. Fungsi Heuristik

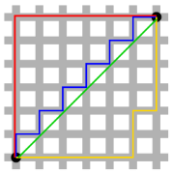
Heuristic merupakan kata yang berasal dari bahasa Yunani yang artinya "mencari atau menemukan". Dalam pemrograman, heuristik adalah teknik yang digunakan untuk memecahkan masalah secara efisien dengan membuat perkiraan. Fungsi heuristik sangat berguna ketika membutuhkan suatu solusi dalam waktu yang cepat. Namun, karena hanya perkiraan heuristik tidak menjamin solusi yang ditemukan adalah yang paling optimal. Dalam algoritma pencarian A* fungsi heuristik merupakan fungsi yang memberikan biaya perkiraan dari suatu solusi. Salah satu cara mandapat nilai dari fungsi heuristik pada algortima A* adalah menghitung nilai jarak manhattan (*manhattan distance*).

E. Jarak Manhattan

Jarak manhattan adalah jarak antara dua titik yang diukur sepanjang sumbu tegak lurus. Berbeda dengan jarak euclidean yang mengukur jarak garis lurus antara titik awal dengan titik akhir. Jarak manhattan antara dua titik A(x_1, y_1) dan B(x_2, y_2) dapat dihitung dengan rumus berikut.

$$Jarak\ Manhattan = |x1 - x2| + |y1 - y2|$$

Berikut merupakan gambaran perbedaan jarak manhattan dengan jarak euclidean.



Gambar 2.9 Jarak manhattan dan euclidean pada grid
(Sumber: https://p2k.stekom.ac.id/ensiklopedia/Jarak_Manhattan)

Pada gambar 2.9 garis merah, biru, dan kuning merupakan jarak manhattan. Apabila dihitung menggunakan rumus jarak manhattan, garis merah, kuning, biru memiliki jarak yang sama, yaitu 12. Sedangkan garis hijau merupakan jarak euclidean. Apabila dihitung menggunakan rumus jarak euclidean, garis hijau bernilai $6\sqrt{2}$.

Jarak manhattan biasanya digunakan pada permainan video jika karakternya hanya bisa bergerak ke atas, bawah, kiri, dan kanan tidak untuk permainan video yang bisa bergerak miring. Apabila karakter bisa bergerak miring, jarak euclidean lebih tepat untuk kasus tersebut. Pada implementasi, diasumsikan karakter tidak bisa bergerak miring sehingga pada implementasi akan

digunakan jarak manhattan untuk digunakan pada fungsi heuristik.

III. ANALISIS

A. *Visualisasi Graf pada Permainan Minecraft untuk Penerapan Algoritma Pencarian A**

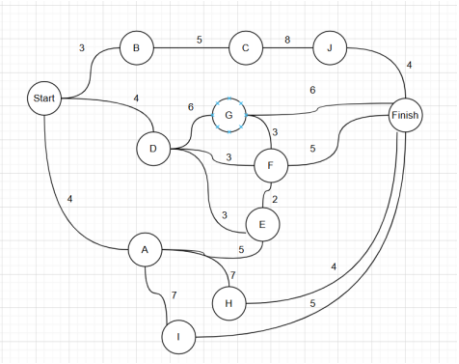
Graf digunakan agar memudahkan menganilisis jalur terpendek yang akan dipilih mob pada minecraft. Untuk menganlisis masalah, dibuat labirin sederhana yang berisi dua mob minecraft, yaitu vindicator dan villager. Akan diletakkan vindicator dan villager pada dua titik yang berbeda. Secara alami, vindicator akan menyerang villager. Untuk mendekati villager, vindicator akan mencari jalur terpendek meunuju villager yang akan diserang. Berikut adalah labirin sederhana yang telah dibuat.

Gambar 3.1 Labirin Sederhana
(Sumber: arsip penulis)

Labirin pada gambar 3.1 dapat dijadikan dalam representasi graf dengan membaginya menjadi beberapa wilayah sehingga terlihat seperti gambar berikut.

Gambar 3.2 Wilayah pada Labirin
(Sumber: arsip penulis)

Pada gambar 3.2 kotak merupakan titik tuju dan garis biru merupakan jalur yang dilalui oleh pillager. Kotak hijau merupakan titik awal dan kotak merah muda merupakan titik akhir Gambar 3.2 bisa kita representasikan sebagai graf berbobot dengan bobot pada graf menggambarkan jarak tempuh satu simpul ke simpul lain. Asumsikan bahwa satu blok pada minecraft jaraknya adalah 1 sehingga akan didapatkan graf sebagai berikut.



Gambar 3.3 Labirin Sederhana
(Sumber: arsip penulis)

Pada graf tersebut simpul menggambarkan kotak dan sisi menggambarkan garis biru serta bobot menggambarkan jarak antar kotak.

B. *Penerapan Algoritma dalam Menentukan Jalur Mob*

Dalam makalah ini, untuk menentukan jalur terpendek menggunakan algoritma A* digunakan fungsi evaluasi yang dapat dihitung berdasarkan rumus berikut.

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

Catatan:
f(n) = total biaya
g(n) = biaya dari simpul awal ke simpul n
h(n) = perkiraan biaya dari simpul n ke simpul akhir

Untuk memperoleh h(n) atau fungsi heuristik, gunakan jarak manhattan. Berikut adalah tabel jarak manhattan setiap simpul.

Tabel 3.1 Daftar jarak manhattan tiap simpul

Simpul	Jarak Manhattan
A	12
B	9
C	12
D	8
E	7
F	5
G	6
H	5
I	5
J	4

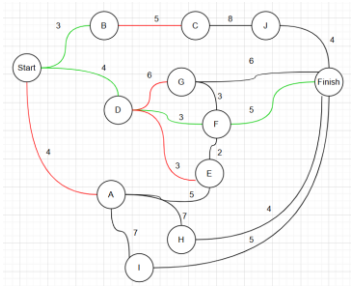
Untuk mendapatkan jalur terpendek kita bisa mengevaluasi simpul mulai dari simpul awal. Simpul dengan nilai f(n) paling kecil akan dipilih sebagai jalur dan sisanya tidak akan dipertimbangkan lagi.

Berikut adalah langkah – langkah pencarian jalur terpendek pada algoritma A*.

1. Simpul awal akan membangkitkan tiga simpul yang terhubung langsung, yaitu simpul A, B, D.
2. Akan dievaluasi nilai f(n) untuk menentukan simpul yang akan diambil.
   Untuk simpul A, nilai g(n) = 3, nilai h(n) = 12 sehingga nilai f(n) = 3 + 12 = 15.
   Untuk simpul B, nilai g(n) = 3, nilai h(n) = 9 sehingga nilai f(n) = 3 + 9 = 12.
   Untuk simpul D, nilai g(n) = 4, nilai h(n) = 8 sehingga nilai f(n) = 4 + 8 = 12.

Simpul A adalah simpul yang memiliki nilai f(n) paling besar sehingga simpul A tidak akan dipertimbangkan lagi. Sedangkan simpul D dan B memiliki nilai f(n) yang sama sehingga akan akan dilanjutkan pada simpul selanjutnya yang terhubung dengan simpul D dan B.

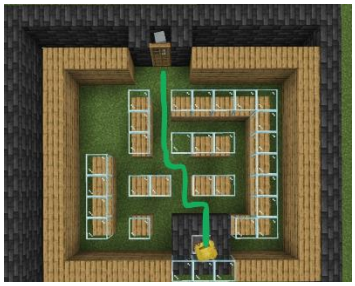
3. Bangkitkan simpul E, F, dan G yang terhubung langsung dengan simpul D. Bangkitkan simpul C yang terhubung dengan simpul B. Kemudian lakukan evaluasi pada simpul E, D, dan C.
   Untuk simpul E, nilai g(n) = 3 + 4 = 7, nilai h(n) = 7 sehingga nilai f(n) = 7 + 7 = 14.
   Untuk simpul F, nilai g(n) = 3 + 4 = 7, nilai h(n) = 5 sehingga nilai f(n) = 7 + 5 = 12.
   Untuk simpul G, nilai g(n) = 4 + 6 = 10, nilai h(n) = 6 sehingga nilai f(n) = 10 + 6 = 16.
   Untuk simpul C, nilai g(n) = 3 + 5 = 8, nilai h(n) = 12 sehingga nilai f(n) = 8 + 12 = 20.
   Nilai f(n) terkecil dimiliki oleh simpul F sehingga simpul yang akan dipilih adalah simpul F, simpul C dan E tidak dipertimbangkan lagi.
4. Kemudian hanya ditemukan satu jalur yang terhubung dengan simpul F. Jalur tersebut ternyata langsung menghubungkan simpul F pada simpul akhir sehingga didapat jalur terpendek yang dapat dilalui oleh vindicator (mob minecraft).
   Berikut adalah jalur yang ditemukan berdasarkan algoritma A*.



Gambar 3.4 Rute Terpendek yang Ditemukan
(Sumber: arsip penulis)

Pada Gambar 3.4 jalur hijau adalah jalur yang bisa dilalui, sedangkan jalur merah adalah jalur yang telah dieliminasi saat melakukan evaluasi. Jalur tercepat ditunjukkan dengan urutan simpul start – D – F – Finish.

Sehingga apabila digambarkan pada labirin pada minecraft, rute yang harus dilalui akan tampak sebagai berikut.



Gambar 3.5 Rute Terpendek pada Labirin
(Sumber: arsip penulis)

C. Hasil Sebenarnya

Untuk membuktikan bahwa hasil dari algortima A* ini benar, akan dibuka pintu yang mengunci vindicator. Setelah pintu dibuka vindikator akan mulai mencari keberadaan villager. Vindicator akan menganalisis rute terpendek yang mungkin kemudian mendekati villager. Berikut adalah pergerakan dari vindicator.

Gambar 3.6 Pergerakan Vindicator Sebenarnya
(Sumber: arsip penulis)

Pada gambar 3.6 terlihat bahwa hasil dari analisis menggunakan algortima A* terbukti sesuai dengan hasil sebenarnya.

D. Pembahasan

Algoritma pencarian A* merupakan algoritma yang paling efisien dalam melakukan pencarian jalur. Pada algortima A*, tidak semua simpul diperiksa, hanya simpul relevan saja yang diperiksa. Hal ini terlihat bahwa terdapat simpul yang sama sekali tidak dievaluasi, seperti simpul J, H, dan I. Berbeda dengan algortima Djikstra memeriksa semua simpul yang ada. Hal itu terjadi karena algortima A* memanfaatkan fungsi heuristik pada BFS untuk fokus pada simpul tujuan dan memanfaatkan fungsi biaya sebernarnya pada Djikstra untuk keakuratan. Akan tetapi, fungsi algoritma A* memiliki kekurangan dalam hal kompleksitas perhitungan. Hal ini terjadi karena fungsi algoritma A* membutuhkan fungsi heuristik dan biaya sebenarnya sekaligus, tidak seperti Dijkstra dan BFS yang hanya memerlukan salah satunya saja.

IV. IMPLEMENTASI DAN EKSPERIMEN

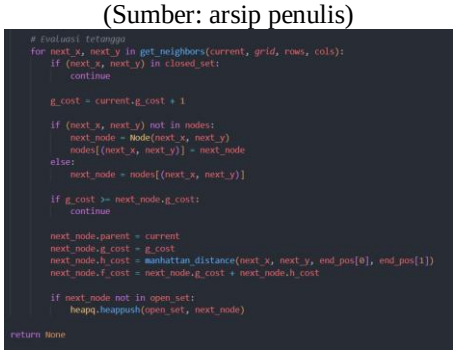
A. Hasil Implementasi

Berikut adalah hasil implementasi algoritma pencarian A* dalam bentuk kode python.

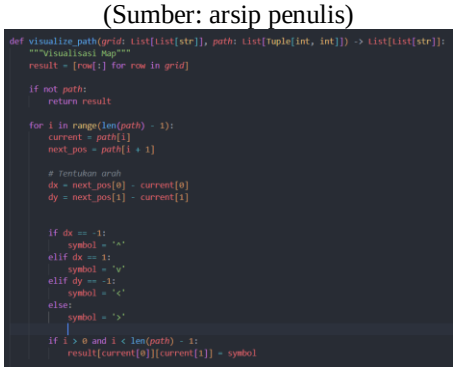
```
import heapq
from typing import List, Tuple

class Node:
    def __init__(self, x: int, y: int, g_cost: float = float('inf'), h_cost: float = 0):
        self.x = x
        self.y = y
        self.g_cost = g_cost
        self.h_cost = h_cost
        self.f_cost = g_cost + h_cost
        self.parent = None

    def __lt__(self, other):
        return self.f_cost < other.f_cost

    def __eq__(self, other):
        return self.x == other.x and self.y == other.y

def manhattan_distance(x1: int, y1: int, x2: int, y2: int) -> float:
    return abs(x1 - x2) + abs(y1 - y2)

def get_neighbors(node: Node, grid: List[List[str]], rows: int, cols: int) -> List[Tuple[int, int]]:
    """Mendapatkan tetangga yang valid dari sebuah node"""
    neighbors = []
    directions = [(-1, 0), (0, 1), (1, 0), (0, -1)]

    for dx, dy in directions:
        new_x, new_y = node.x + dx, node.y + dy
        if (0 <= new_x < rows and 0 <= new_y < cols and
            grid[new_x][new_y] != '#'):
            neighbors.append((new_x, new_y))

    return neighbors
```

Gambar 4.1 Implementasi kode bagian 1
(Sumber: arsip penulis)

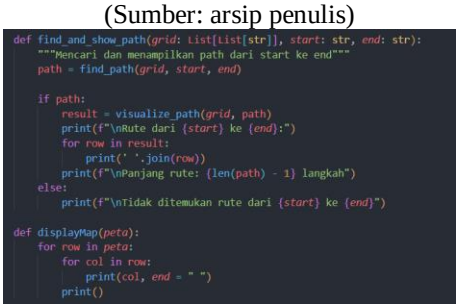
Gambar 4.2 Implementasi kode bagian 2
(Sumber: arsip penulis)



Gambar 4.3 Implementasi kode bagian 3
(Sumber: arsip penulis)



Gambar 4.4 Implementasi kode bagian 4
(Sumber: arsip penulis)



Gambar 4.5 Implementasi kode bagian 5
(Sumber: arsip penulis)

B. Eksperimen

Untuk membuktikan kebeneran dari kode tersebut, akan dilakukan eksperimen. Eksperimen akan dilakukan dengan cara membuat map pada kode yang sama persis dengan labirin pada gambar 3.1, rute yang didapat seharusnya sama seperti pada gambar 3.5. Berikut adalah output dari implementasi.

Gambar 4.6 Input pada program
(Sumber: arsip penulis)

Gambar 4.7 Output yang dikeluarkan program
(Sumber: arsip penulis)

Terlihat pada gambar 4.7, rute yang dipilih sama persis dengan rute sebenarnya. Hal ini ditunjukkan oleh simbol ‘v’ dan ‘>’. Jarak yang ditempuh juga sama persis dengan total bobot simpul start – D – F – finish, yaitu 12.

V. KESIMPULAN

Algoritma A* merupakan salah satu algoritma yang digunakan untuk mencari rute terpendek yang bisa dilalui. Algortima A* merupakan gabungan dari algoritma BFS dan Dijksta sehingga hasil yang diberikan lebih efisien dan akurat. Pada algoritma A*, tidak semua simpul dievaluasi, hanya simpul yang relevan saja sehingga algortima ini dikatakan lebih efisien dari BFS dan Djikstra. Hal ini dapat kita buktikan pada bab analisis. Oleh karena itu, algoritma ini sering dipakai untuk menentukan rute gerak mob pada permainan video, seperti pada permainan video minecraft. Pada eksperimen terbukti bahwa output dari implementasi algortima A* sama persis seperti pada permainan video minecraft. Akan tetapi, algoritma ini membutuhkan kompleksitas perhitungan yang lebih rumit dari algortima lainnya.

VI. LAMPIRAN

Berikut adalah source code yang digunakan untuk implementasi.

https://github.com/AryoBama/ImplementasiMatdis/blob/main/AStar.py

VII. UCAPAN TERIMAKASIH

Puji syukur dipanjatkan kepada Tuhan Yang Maha Esa yang telah melimpahkan rahmat dan karuniaNya sehingga penyusunan makalah ini dapat diselesaikan dengan lancar. Penulis ingin menyampaikan rasa terima kasih kepada keluarga serta teman – teman karena atas dukungan mereka selama proses

pengerjaan makalah sehingga makalah dapat selesai dengan baik. Ucapan terima kasih juga diucapkan kepada para pengajar mata kuliah IF1220, khususnya Ir. Rila Mandala, M.Eng., Ph.D. selaku dosen pengampu kelas 02 yang telah memberikan ilmu yang bermanfaat dalam penyusunan makalah ini. Harapan penulis, makalah ini dapat menjadi sumber referensi yang berguna, baik bagi para pembelajar yang tertarik dengan bidang ilmu ini maupun sebagai acuan bagi penulis sendiri di masa mendatang.

REFERENSI

[1] Rinaldi Munir, "Graf (bagian 1)," https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf (diakses 07 Januari 2025, pukul 19.23).

[2] Rinaldi Munir, "Graf (bagian 2)," https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/21-Graf-Bagian2-2024.pdf (diakses 07 Januari 2025, pukul 19.35).

[3] Trivusi, "Algoritma A* (A Star): Pengertian, Cara Kerja, dan Kegunaannya," *https://www.trivusi.web.id/2023/01/algoritma-a-star.html* (diakses 07 Januari 2025, pukul 20.05).

[4] Mojang Studios, https://www.minecraft.net/en-us (diakses 07 Januari 2025, pukul 18.30).

[5] Binus, "TEORI GRAPH, SEJARAH DAN MANFAATNYA," https://mti.binus.ac.id/2018/03/05/teori-graph-sejarah-dan-manfaatnya/ (diakses pada 07 Januari 2025, pukul 18.58).

[6] Sukardi, "Materi, Soal, dan Pembahasan – Dasar-Dasar Graf dan Terminologinya," *https://mathcyber1997.com/materi-soal-dasar-graf-dan-terminologi/* (diakses 07 Januari 2025, pukul 20.28).

[7] Anonymous , "Algoritma Prim," https://matrisnowei.blogspot.com/2015/01/algoritma-prim.html (diakses tanggal 07 Januari 2025, pukul 21.47).

[8] Stekom, "Jarak Manhattan," https://p2k.stekom.ac.id/ensiklopedia/Jarak_Manhattan (diakses 07 Januari 2025, pukul 21.58).

[9] Gramedia, "Apa Itu Heuristik? Metode Penelitian Sejarah, Pengertian, & Contohnya,", https://www.gramedia.com/literasi/heuristik/ (diakses 07 Januari 2025, pukul 21.53).

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, 08 Januari 2025

Aryo Bama Wiratama, 13523088