# Penerapan Algoritma A* Untuk Menemukan Rute Optimal dalam Tugas Besar 1 Strategi Algoritma ITB 2021 “Worms”

Made Kharisma Jagaddhita/13519176
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519176@std.stei.itb.ac.id

***Abstrak*—Permainan Worms adalah permainan perang antara bot satu dengan bot lainnya. Dalam permainan, strategi yang dapat digunakan sangatlah bervariasi. Untuk melancarkan strategi tersebut, tentunya worms dari pemain harus dapat mencapai sel tertentu dengan cepat. Algoritma yang dapat digunakan untuk mencapai sel tertentu dengan cepat adalah algoritma A*. Algoritma ini akan menemukan rute yang dapat dilalui oleh worm untuk mencapai suatu sel tertentu.**

***Keywords—Entelect Challenge 2019 “Worms”, Algoritma A*,* Pathfinding Algorithm**

## I. Pendahuluan

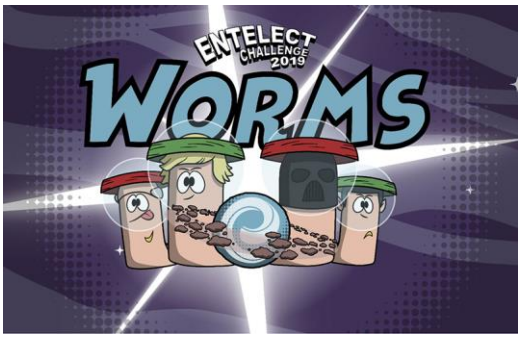
Teknologi yang berkembang baru-baru ini membuat jurusan Informatika di Indonesia menjadi populer. Informatika adalah disiplin ilmu komputer yaitu data maupun informasi pada mesin berbasis komputasi. Informatika mempelajari tentang struktur, sifat, dan interaksi dari beberapa sistem yang dipakai untuk mengumpulkan data, memproses dan menyimpan hasil pemrosesan data, serta menampilkannya dalam bentuk informasi

Pada jurusan Informatika Institut Teknologi Bandung, terdapat mata kuliah IF2211 Strategi Algoritma. Mata kuliah ini mempelajari tentang strategi-strategi algoritma yang dapat digunakan untuk menyelesaikan suatu masalah. Mata kuliah ini bertujuan agar mahasiswa dapat mengetahui algoritma-algoritma mangkus yang dapat digunakan untuk memecahkan masalah. Pada tahun 2021, untuk meningkatkan pemahaman mengenai algoritma greedy, dosen beserta asisten dari mata kuliah IF2211 merilis tugas besar yang berjudul “Pemanfaatan Algoritma Greedy dalam Aplikasi Permainan ‘Worms’”. Dalam tugas besar ini, mahasiswa diperintahkan untuk membuat sebuah bot dengan menggunakan algoritma greedy untuk mengimplementasikan strategi yang digunakan untuk mengalahkan lawan. Bot-bot yang dibuat mahasiswa ini nantinya akan dipertandingkan satu sama lain. Tim yang memenangkan kontes akan mendapatkan tambahan nilai dan gopay

Dalam permainan Worms, strategi yang dapat digunakan untuk memenangkan permainan sangat bervariasi. Mulai dari pengambilan *power up*, membantu worm yang sedang berperang, ataupun melarikan diri dari zona perang. Untuk melancarkan strategi-strategi tersebut, tentunya worms dari pemain harus dapat mencapai sel tertentu dengan cepat. Dengan menggunakan algoritma A*, worm dapat menentukan rute optimal untuk mencapai sel tersebut.

## II. Landasan Teori

### A. *Entelect Challenge 2019 “Worms”*




Gambar 1: Permainan Worms
Source: https://entelect.co.za/News/DisplayNewsItem.aspx?niid=63071

Permainan “Worms” merupakan sebuah permainan yang dilombakan pada *event* Entelect Challenge 2019. Permainan ini juga sempat menjadi salah satu tugas besar mata kuliah Strategi Algoritma ITB tentang penggunaan algoritma greedy.

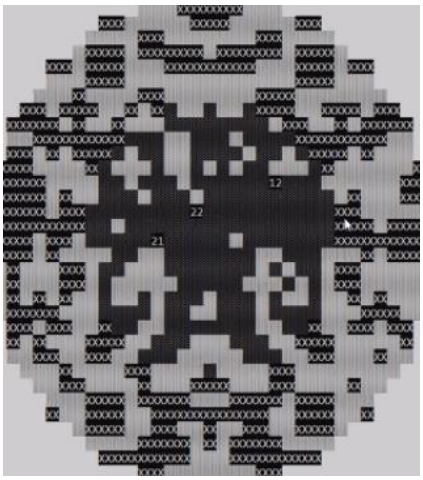
Permainan ini akan dimainkan 2 orang pemain yang akan bertanding satu sama lain. Setiap pemain akan memiliki 3 worms. Hasil akhir dari permainan akan ditentukan dari pemain terakhir yang memiliki worm yang masih hidup atau berdasarkan skor yang diperoleh.

Permainan dimainkan pada kotak yang di dalamnya terdapat 33 x 33 sel. Setiap sel adalah salah satu dari tipe berikut:

- *Air* – Setiap worm dapat bergerak dan menembak melalui sel ini. Pada peta, direpresentasikan dengan kotak berwarna hitam.
- *Dirt* – Setiap worm tidak dapat bergerak dan menembak melalui sel ini. Sel ini harus digali terlebih

dahulu. Pada peta, direpresentasikan dengan kotak berwarna abu-abu.

- *Deep Space* – Setiap worm tidak dapat berinteraksi dengan sel ini. Pada peta, direpresentasikan dengan abu muda putih yang terletak pada pojok peta
- *Lava* – Setiap worm dapat bergerak dan menembak melalui sel ini. Tetapi worm akan menerima *damage* setiap ronde jika berada di atasnya. Pada peta, direpresentasikan dengan huruf X.



Gambar 2: Peta pada permainan Worms
Source: Dokumen penulis

Setiap sel dapat berisi *powerup*. *Powerup* berupa *health pack* yang akan menambah *health* dari worm sebesar 10. Pada peta, direpresentasikan dengan huruf H.

Setiap ronde, pemain akan mengirimkan perintah untuk worm mereka yang sedang aktif. Perintah-perintah tersebut adalah sebagai berikut:

- *Move* – Perintah ini akan membuat worm bergerak ke sel yang dituju. Sel yang dituju adalah sel yang berada di sekitar worm tersebut, termasuk yang berada di diagonalnya. Worm dapat bergerak ke *Air* atau *Lava*, tetapi tidak dapat bergerak ke *Deep Space* atau *Dirt.*
- *Dig* – Perintah ini akan membuat worm menggali sel yang dituju. Sel yang dituju adalah sel yang berada di sekitar worm tersebut, termasuk yang berada di diagonalnya. Sel yang dapat digali hanya sel yang bertipe *Dirt*. Menggali sel akan mengubah sel tersebut menjadi sel bertipe *Air*.
- *Shoot* – Perintah ini akan membuat worm menembak ke arah yang dituju. Arah yang dituju berupa 8 arah mata angin. Tembakan worm dapat terhalang oleh sel *Dirt* atau *Deep Space.* Worm dari kedua pemain dapat terkena tembakan ini, kecuali worm yang menembak. Worm yang terkena tembakan akan dikurangi *health*-nya.
- *Do Nothing* – Perintah ini akan membuat worm tidak melakukan apa-apa.
- *Banana Bomb* – Perintah ini akan membuat worm menembakkan bom yang akan meledak dan mengurangi *health* dari worm dalam radius tertentu. Perintah ini hanya dapat digunakan oleh worm bertipe *Agent*.
- *Snowball* – Perintah ini akan membuat worm menembakkan bola salju yang akan meledak dan membuat worm yang berada dalam radius tertentu tidak dapat berbuat apa-apa untuk beberapa ronde. Perintah ini hanya dapat digunakan oleh worm bertipe *Technologist*
- *Select* – Perintah ini digunakan untuk mengganti worm yang sedang aktif. Perintah memiliki penggunaan yang terbatas.

Setiap worm yang dimiliki pemain memiliki profesi tertentu. *Commando* memiliki *health* yang lebih banyak dibandingkan worm lain. *Agent* dapat menggunakan perintah *Banana bomb.* dan *Technologist* dapat menggunakan perintah *Snowball.*

Jika setiap pemain masih memiliki worm yang masih hidup setelah 400 ronde, maka hasil permainan akan ditentukan dengan skor. Setiap perintah yang dikirimkan pemain kepada worm akan menghasilkan skor yang berbeda. Pemain yang memiliki skor tertinggi akan menjadi pemenang dari permainan.

### *B. Pathfinding Algorithms*

*Pathfinding Algorithms* adalah algoritma untuk menemukan rute dari dua node dengan kriteria yang ditentukan. Kriteria dapat berupa rute termurah, rute terpendek, dan-lain-lain. Terdapat *Pathfinding Algorithms*, diantaranya adalah:

- *Depth First Search* – Algoritma ini akan mengeksplorasi sejauh mungkin dari suatu cabang. Jika destinasi tidak ditemukan pada cabang tersebut, maka pencarian akan mundur ke node sebelumnya.
- *Breadth First Search* – Algoritma ini akan mengeksplorasi semua node tetangga pada kedalaman saat ini sebelum pindah ke node pada level kedalaman berikutnya. Algoritma ini akan selalu menemukan rute dengan langkah terpendek.
- *Djikstra Algorithm* – Algoritma ini agak mirip dengan algoritma BFS. Setelah mengeksplorasi semua node dengan kedalaman saat ini, algoritma ini akan memilih simpul terpendek saat ini ke simpul sumber di setiap langkahnya.
- *Greedy Best First Search* – Algoritma ini bekerja dengan memilih node terpendek saat ini ke node tujuan di setiap langkah. Jarak node ke tujuan dapat diperkirakan dengan menggunakan fungsi heuristik.
- *A* Algorithm* – Algoritma ini bekerja dengan mengunjungi node dengan jarak saat ini ditambah dengan perkiraan jarak ke tujuan yang terkecil dari semua node yang diekspansi. Algoritma ini akan selalu menemukan jalur terpendek dalam graf apapun jika fungsi heuristik yang digunakan *admissible*.

### C. *Algoritma A**

Algoritma A* digunakan untuk menemukan rute yang paling optimal untuk ditempuh. Algoritma ini termasuk ke dalam kategori metode *informed search method*. Ide dari algoritma ini adalah menghindari rute dengan biaya yang sudah terlalu besar. Penentuan node yang akan diekspansi ditetukan dengan menggunakan fungsi evaluasi. Fungsi evaluasi adalah sebagai berikut:

- Fungsi evaluasi f(n) = g(n) + h(n)
- g(n) = Biaya saat ini untuk mencapai node n
- h(n) = Perkiraan biaya dari node n ke node tujuan (nilai heuristik)
- f(n) = Perkiraan total biaya rute dari node n ke node tujuan.

Pseudocode dari algoritma A* adalah sebagai berikut:

```
function reconstruct_path(cameFrom, current)
    total_path := {current}
    while current in cameFrom.Keys:
        current := cameFrom[current]
        total_path.prepend(current)
    return total_path

function A_Star(start, goal, h)
    openSet := {start}
    cameFrom := an empty map

    gScore := map with default value of Infinity
    gScore[start] := 0

    fScore := map with default value of Infinity
    fScore[start] := h(start)

    while openSet is not empty
        current := the node in openSet having the lowest
          fScore[] value
        if current = goal
            return reconstruct_path(cameFrom, current)

        openSet.Remove(current)
        for each neighbor of current
            tentative_gScore := gScore[current] + d(current,
              neighbor)
            if tentative_gScore < gScore[neighbor]
                cameFrom[neighbor] := current
                gScore[neighbor] := tentative_gScore
                fScore[neighbor] := gScore[neighbor] + h
                  (neighbor)
                if neighbor not in openSet
                    openSet.add(neighbor)

    return failure
```

Gambar 3: Pseudocode algoritma A*
Source: Zeng, W.; Church, R. L. (2009). "Finding shortest paths on real road networks: the case for A*". International Journal of Geographical Information Science.

Optimalitas dari algoritma A* ditentukan dari fungsi heuristik yang *admissible* atau tidak. Fungsi heuristik dapat diterima jika untuk setiap node n, h(n) ≤ h*(n) dimana h(n) adalah perkiraan biaya dan h*(n) adalah biaya sebenarnya untuk mencapai node tujuan dari node n. Jika fungsi heuristik *admissible*, maka algoritma A* akan selalu dapat menemukan rute dengan biaya terkecil.

## III. PEMBAHASAN

### A. *Langkah-Langkah Algoritma A**

Algoritma A* menggunakan *open list* yang menyatakan node hidup, dan *closed list* yang menyatakan simpul-simpul yang telah diekspan. *Closed list* digunakan agar simpul-simpul yang telah terekspan tidak diekspan lagi sehingga algoritma tidak akan berlanjut jika tidak terdapat solusi.

Langkah-langkah dari algoritma A* adalah sebagai berikut:

1. Inisialisasi open list dan closed list. Open list nantinya akan berisi simpul-simpul yang masih hidup. Closed list akan berisi simpul-simpul yang telah dikunjungi.
2. Masukkan simpul awal ke dalam open list.
3. Iterasi semua simpul pada open list dan pilih simpul yang memiliki estimasi cost paling kecil. Perhitungan simpul yang ingin diekspansi menggunakan rumus f(n) = g(n) + h(n) dimana f(n) adalah estimasi cost dengan melalui jalur n ke tujuan, g(n) adalah cost sejauh ini untuk mencapai n, dan h(n) adalah perkiraan biaya n ke tujuan. Perhitungan h(n) dapat dilakukan secara heuristik. Dalam tugas ini akan menggunakan jarak euclidean.
4. Masukkan simpul yang dipilih tadi ke dalam closed list.
5. Ekspansi simpul yang dipilih. jika hasil ekspansi dari simpul tidak berada di closed list, maka masukkan simpul tersebut ke open list.
6. Ulangi langkah 3 sampai 5. Jika simpul yang diekspansi ternyata merupakan simpul tujuan, maka path telah ditemukan. Jika tidak ada lagi simpul di open list dan path belum ditemukan, maka simpul tujuan tidak dapat tercapai dari simpul awal.

### B. *Fungsi Evaluasi*

- f(n)

  Fungsi f(n) merupakan perkiraan biaya total yang ditempuh menuju simpul destinasi dengan melalui simpul n. Nilai dari fungsi f(n) didapatkan dengan menggunakan rumus f(n) = g(n) + h(n).

- g(n)

  Fungsi g(n) merupakan biaya yang diperlukan untuk mengunjungi node n dari node awal. Pada permainan Worms, perhitungan biaya untuk mengunjungi suatu sel bukanlah jumlah sel yang dilalui suatu worms untuk sampai ke node n. Melainkan banyak perintah minimal yang diperlukan untuk sampai ke node n. Hal ini karena halangan pada permainan Worms berbeda dengan maze biasa. Pada maze halangan tidak dapat dihancurkan, sedangkan pada permainan worms,

terdapat halangan, yaitu sel *Dirt* yang bisa dihancurkan. Oleh karena itu, didapatkan perhitungan sebagai berikut:

- Untuk mencapai sel *Air* atau *Lava* yang berada di sekitar worms, penambahan nilai g(n) adalah 1. Hal ini karena jumlah perintah yang dibutuhkan untuk mencapai sel tersebut hanya *move* untuk berpindah ke sel tersebut.
- Untuk mencapai sel *Dirt* yang berada di sekitar worms, penambahan nilai g(n) adalah 2. Hal ini karena jumlah perintah yang dibutuhkan untuk mencapai sel tersebut adalah *dig* untuk mengubah sel tersebut menjadi sel *Air*, dan *move* untuk berpindah ke sel tersebut.
- Untuk mencapai sel *Deep Space* yang berada di sekitar worms, penambahan nilai g(n) adalah tak hingga. Hal ini karena sel tersebut tidak bisa ditempati oleh worms.

- h(n)

Untuk mendapatkan solusi yang optimal, fungsi heuristik yang digunakan untuk memperkirakan biaya dari node yang dikunjungi saat ini ke node destinasi harus selalu kurang atau sama dengan biaya yang sebenarnya. Pada permainan Worms, nilai heuristik adalah nilai maksimal antara jarak horizontal dan jarak vertikal sel saat ini ke sel destinasi. Hal ini karena worms dapat bergerak secara diagonal dengan satu perintah sehingga jika digunakan jarak euclidean sebagai heurustik, terdapat kemungkinan nilai heuristik yang *over-estimated* sehingga fungsi heuristik yang menggunakan jarak euclidean tidak *admissible*. Misalkan terdapat potongan peta permainan Worms sebagai berikut:

P X - - O
X - X - -
X X - - -
X X X - D
X L L X X

Keterangan:

P : Letak worm saat ini

\- : sel *Air*

X : sel *Dirt*

O : sel *Deep Space*

L : sel *Lava*

D : Destinasi

Maka perhitungan nilai heuristik adalah:

- Hitung jarak horizontal P ke D, didapatkan nilai 4.
- Hitung jarak vertikal P ke D, didapatkan nilai 3.
- Ambil nilai maksimal antara jarak horizontal dan jarak vertikal P ke D, didapatkan nilai heuristik yang didapatkan adalah 4.

*C. Simulasi Algoritma A* Pada Permainan Worms*

Misalkan terdapat potongan peta permainan Worms sebagai berikut:

```
   0 1 2 3 4

0  X X D X -
1  X X X X -
2  L - - X -
3  X - - X -
4  O O O X P
```

Jika terdapat dua node dengan nilai f(n) yang sama, akan prioritas node dari yang terbesar adalah node yang berada di atas node yang diekspansi. Prioritas sel yang berada di sekitar node yang diekspansi akan semakin kecil dengan perputaran arah jarum jam. Sehingga prioritas tertinggi adalah node yang berada di atas node yang diekspansi saat ini dan prioritas terrendah adalah node yang berada di atas-kiri node yang diekspansi saat ini

Node tujuan adalah koordinat (0, 2). Pertama-tama akan mengekspansi node dengan koordinat tempat worm berada, yaitu (4, 4) sehingga didapatkan:

| Formula: f(n) = g(n) + f(n) | | |
|---|---|---|
| Simpul Ekspan | Simpul hidup | f(n) |
| (4, 4) | (3, 4) [(4, 4)] | 1+3=4 |
| | (3, 3) [(4, 4)] | 2+3=5 |
| | (4, 3) [(4, 4)] | 2+4=6 |
| (3, 4) [(4, 4)] | (3, 3) [(4, 4)] | 2+3=5 |
| | (4, 3) [(4, 4)] | 2+4=6 |
| | (2, 4) [(3, 4), (4, 4)] | 2+2=4 |
| | (4, 3) [(3, 4), (4, 4)] | 3+4=7 |

| | | |
|---|---|---|
| | (3, 3) [(3, 4), (4, 4)] | 3+3=6 |
| | (2, 3) [(3, 4), (4, 4)] | 3+2=5 |
| (2, 4) [(3, 4), (4, 4)] | (3, 3) [(4, 4)] | 2+3=5 |
| | (4, 3) [(4, 4)] | 2+4=6 |
| | (4, 3) [(3, 4), (4, 4)] | 3+4=7 |
| | (3, 3) [(3, 4), (4, 4)] | 3+3=6 |
| | (2, 3) [(3, 4), (4, 4)] | 3+2=5 |
| | (1, 4) [(2, 4), (3, 4), (4, 4)] | 3+2=5 |
| | (3, 3) [(2, 4), (3, 4), (4, 4)] | 4+3=7 |
| | (2, 3) [(2, 4), (3, 4), (4, 4)] | 4+2=6 |
| | (1, 3) [(2, 4), (3, 4), (4, 4)] | 4+1=5 |
| (1, 4) [(2, 4), (3, 4), (4, 4)] | (3, 3) [(4, 4)] | 2+3=5 |
| | (4, 3) [(4, 4)] | 2+4=6 |
| | (4, 3) [(3, 4), (4, 4)] | 3+4=7 |
| | (3, 3) [(3, 4), (4, 4)] | 3+3=6 |
| | (2, 3) [(3, 4), (4, 4)] | 3+2=5 |
| | (3, 3) [(2, 4), (3, 4), (4, 4)] | 4+3=7 |
| | (2, 3) [(2, 4), (3, 4), (4, 4)] | 4+2=6 |
| | (1, 3) [(2, 4), (3, 4), (4, 4)] | 4+1=5 |
| | (0, 4) [(1, 4), (2, 4), (3, 4), (4, 4)] | 4+2=6 |
| | (2, 3) [(1, 4), (2, 4), (3, 4), (4, 4)]7 | 5+2=7 |
| | (1, 3) [(1, 4), (2, 4), (3, 4), (4, 4)] | 5+1=6 |
| | (0, 3) [(1, 4), (2, 4), (3, 4), (4, 4)] | 5+1=6 |
| (2, 3) [(3, 4), (4, 4)] | (3, 3) [(4, 4)] | 2+3=5 |
| | (4, 3) [(4, 4)] | 2+4=6 |

| | | |
|---|---|---|
| | (4, 3) [(3, 4), (4, 4)] | 3+4=7 |
| | (3, 3) [(3, 4), (4, 4)] | 3+3=6 |
| | (3, 3) [(2, 4), (3, 4), (4, 4)] | 4+3=7 |
| | (2, 3) [(2, 4), (3, 4), (4, 4)] | 4+2=6 |
| | (1, 3) [(2, 4), (3, 4), (4, 4)] | 4+1=5 |
| | (0, 4) [(1, 4), (2, 4), (3, 4), (4, 4)] | 4+2=6 |
| | (2, 3) [(1, 4), (2, 4), (3, 4), (4, 4)] | 5+2=7 |
| | (1, 3) [(1, 4), (2, 4), (3, 4), (4, 4)] | 5+1=6 |
| | (1, 3) [(1, 4), (2, 4), (3, 4), (4, 4)] | 5+1=6 |
| | (0, 3) [(2, 3), (3, 4), (4, 4)] | 5+1=6 |
| | (3, 3) [(2, 3), (3, 4), (4, 4)] | 5+3=8 |
| | (3, 2) [(2, 3), (3, 4), (4, 4)] | 4+3=7 |
| | (2, 2) [(2, 3), (3, 4), (4, 4)] | 4+2=6 |
| | (1, 2) [(2, 3), (3, 4), (4, 4)] | 5+1=6 |
| (1, 3) [(2, 4), (3, 4), (4, 4)] | (3, 3) [(4, 4)] | 2+3=5 |
| | (4, 3) [(4, 4)] | 2+4=6 |
| | (4, 3) [(3, 4), (4, 4)] | 3+4=7 |
| | (3, 3) [(3, 4), (4, 4)] | 3+3=6 |
| | (3, 3) [(2, 4), (3, 4), (4, 4)] | 4+3=7 |
| | (2, 3) [(2, 4), (3, 4), (4, 4)] | 4+2=6 |
| | (0, 4) [(1, 4), (2, 4), (3, 4), (4, 4)] | 4+2=6 |
| | (2, 3) [(1, 4), (2, 4), (3, 4), (4, 4)] | 5+2=7 |
| | (1, 3) [(1, 4), (2, 4), (3, 4), (4, 4)] | 5+1=6 |
| | (0, 3) [(1, 4), (2, 4), (3, 4), (4, 4)] | 5+1=6 |

| | | |
|---|---|---|
| | (1, 3) [(2, 3), (3, 4), (4, 4)] | 5+1=6 |
| | (3, 3) [(2, 3), (3, 4), (4, 4)] | 5+3=8 |
| | (3, 2) [(2, 3), (3, 4), (4, 4)] | 4+3=7 |
| | (2, 2) [(2, 3), (3, 4), (4, 4)] | 4+2=6 |
| | (1, 2) [(2, 3), (3, 4), (4, 4)] | 5+1=6 |
| | (0, 3) [(1, 3), (2, 4), (3, 4), (4, 4)] | 6+1=7 |
| | (0, 4) [(1, 3), (2, 4), (3, 4), (4, 4)] | 5+2=7 |
| | (2, 2) [(1, 3), (2, 4), (3, 4), (4, 4)] | 5+2=7 |
| | (1, 2) [(1, 3), (2, 4), (3, 4), (4, 4)] | 6+1=7 |
| | (0, 2) [(1, 3), (2, 4), (3, 4), (4, 4)] | 5+0=5 |
| (0, 2) [(1, 3), (2, 4), (3, 4), (4, 4)] | Node tujuan sudah tercapai | 5 |

Tabel 1: Simulasi algoritma A* pada permainan Worms

Dari simulasi diatas didapatkan rute terpendek adalah:

(4, 4) → (3, 4) → (2, 4) → (1, 3) → (0, 2) dengan jumlah perintah yang diperlukan untuk mencapai node tujuan dari node awal = 5

## IV. KESIMPULAN DAN SARAN

### A. *Kesimpulan*

Hasil penerapan Algoritma A* untuk permainan Worms dapat menentukan rute dengan jumlah perintah terpendek dari sel satu ke sel lainnya. Dengan begitu, worm strategi-strategi yang digunakan menjadi lebih efektif. Worm dapat mengambil *power up* lebih cepat dibandingkan worm musuh, segera membantu teman, ataupun melarikan diri dengan rute yang lebih efektif.

### B. *Saran*

Pada pembahasan, sel *Lava* tidak diasumsikan tidak dapat dilewati. Dampaknya pada rute dengan jumlah perintah minimal, terdapat kemungkinan bahwa terdapat sel *Lava* sehingga dapat merugikan pemain. Tetapi jika asumsi tersebut ditetapkan, jika worm dikelilingi oleh sel *Lava*, maka tidak akan terdapat rute dengan jumlah perintah minimal. Oleh karena itu dibutuhkan fungsi untuk menentukan apakah untuk mencapai sel tujuan, worm dikelilingi oleh lava atau tidak sehingga tidak akan terdapat kasus tidak ada rute yang ditemukan.

## V. PRANALA VIDEO YOUTUBE

https://youtu.be/XvZKs3fRrCo

## VI. UCAPAN TERIMA KASIH

Segala puji dan syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa karena karunia-Nya, penulis dapat menyelesaikan makalah ini dengan baik dan tepat waktu.

Terima kasih yang sebesar-besarnya kepada semua pihak yang telah membantu penulis dalam menyelesaikan makalah ini. Terima kasih kepada kedua orang tua yang telah memberi dukungan kepada penulis. Terima kasih kepada Pak Rinaldi Munir selaku dosen pengampu mata kuliah Strategi Algoritma IF2211 kelas K4, dan juga kepada seluruh tim pengajar mata kuliah IF2211 yang telah mengajarkan ilmunya kepada penulis sehingga penulis mampu menyelesaikan makalah ini. Penulis juga menyampaikan terima kasih kepada semua teman penulis yang selalu memberikan motivasi dalam pengerjaan tugas makalah ini.

## REFERENSI

[1] Munir, Rinaldi, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian2-2021.pdf, diakses pada 11 Mei 2021

[2] Munir, Rinaldi, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Tugas-Besar-1-IF2211-Strategi-Algoritma-2021.pdf, diakses pada 10 Mei 2021

[3] Urna H., https://medium.com/@urna.hybesis/pathfinding-algorithms-the-four-pillars-1ebad85d4c6b, diakses pada 11 Mei 2021

[4] Roy, Baijayanta, https://towardsdatascience.com/a-star-a-search-algorithm-eb495fb156bb, diakses pada 11 Mei 2021

[5] Entelect Challenge, https://github.com/EntelectChallenge/2019-Worms/blob/develop/game-engine/game-rules.md, diakses pada 11 Mei 2021

[6] Zeng, W.; Church, R. L. (2009). "Finding shortest paths on real road networks: the case for A*". International Journal of Geographical Information Science.

## 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.

Badung, 11 Mei 2021

Made Kharisma Jagaddhita (13519176)