Pembuatan *Bot* pada Permainan *Connect Four* dengan menggunakan Strategi Algoritma *Greedy*

Mahesa Lizardy - 13520116
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520116@std.stei.itb.ac.id

***Abstract*—*Connect four* merupakan permainan *board game* dua pemain di mana pemain akan bergiliran memasukkan token yang berbeda warna, tujuannya adalah pemain yang mendapatkan *connect four* (4 buah token berurutan secara horizontal, vertikal, atau diagonal) terlebih dahulu akan memenangkan permainan. Pada permainan ini terdapat beberapa strategi yang dapat digunakan oleh pemain untuk mendapatkan kemenangan. Dari strategi-strategi tersebut akan dibuat sebuah *bot* dari sebuah *game connect four* yang mengimplementasikan strategi algoritma *greedy*.**

Keywords—algoritma; greedy; connect four

I. PENDAHULUAN

Gambar 1.1 *Game Connect Four (Sumber https://www.enasco.com/p/Connect-Four%C2%AE%2BSB19383)*

connect four merupakan *board game* dua pemain, di mana para pemain akan memilih warna dan kemudian bergantian menjatuhkan token berwarna secara vertikal ke dalam kotak berukuran 7 kolom dan 6 baris. Pada umumnya *connect four* klasik berukuran 7x6 namun terdapat variasi ukuran lainnya yaitu 8x7, 9x7, 10x7, 8x8, dan *connect four* dengan baris dan kolom yang tidak terbatas atau *infinite connect four*. Pada permainan *connect four*, Pemain yang pertama membentuk garis horizontal, vertikal, atau diagonal dari empat token milik sendiri akan memenangkan permainan. Jika papan terisi penuh pada semua posisi sebelum salah satu pemain mencapai empat diagonal berturut-turut maka permainan akan seri. Kompleksitas permainan *connect four* ini terdapat 4.531.985.219.092 kemungkinan posisi untuk semua papan permainan yang diisi dengan 0 hingga 42 buah[2]. Permainan ini telah diselesaikan dengan metode *brute force*. Namun Pada makalah ini akan digunakan pendekatan strategi algoritma *greedy* untuk menyelesaikan permainan *connect four*.

II. LANDASAN TEORI

A. Algoritma Greedy

Greedy dapat berarti rakus, tamak, dan loba. Algoritma *greedy* merupakan algoritma yang menggunakan pendekatan sederhana untuk memecahkan persoalan optimasi. persoalan optimasi(*optimization problems*) merupakan persoalan untuk mencari solusi yang optimal. Persoalan optimasi dapat berupa persoalan maksimasi (*maximization*) dan persoalan minimasi (*minimiuzation*). Algoritma *greedy* membentuk solusi langkah per langkah (*step by step*) mencari solusi yang paling optimal pada waktu itu atau optimum lokal. Algoritma *greedy* pada setiap langkah akan

1. Mengambil pilihan yang terbaik pada kondisi waktu itu tanpa memperhatikan konsekuensi kedepannya (prinsip "*take what you can get now!*")
2. dan "berharap" bahwa dengan memilih optimum lokal pada setiap langkahnya akan menjadi optimum global diakhir

Dalam menggunakan algoritma *greedy* terlebih dahulu dicari elemen-elemen-nya. Elemen-elemen algoritma *greedy* terdiri dari:

1. Himpunan Kandidat, C: berisi kandidat yang akan dipilih pada setiap langkah (misal: simpul/sisi di dalam graf, *job*, *task*, koin, benda, karakter, dll)
2. Himpunan solusi, S: berisi kandidat yang sudah dipilih
3. Fungsi solusi: menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi
4. Fungsi seleksi (*selection function*): memilih kandidat berdasarkan strategi *greedy* tertentu
5. Fungsi kelayakan (*feasible*): memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak)
6. Fungsi obyektif: memaksimumkan atau meminumkan

Dengan adanya elemen-elemen diatas maka dapat dikatakan bahwa Algoritma *greedy* melibatkan Algoritma

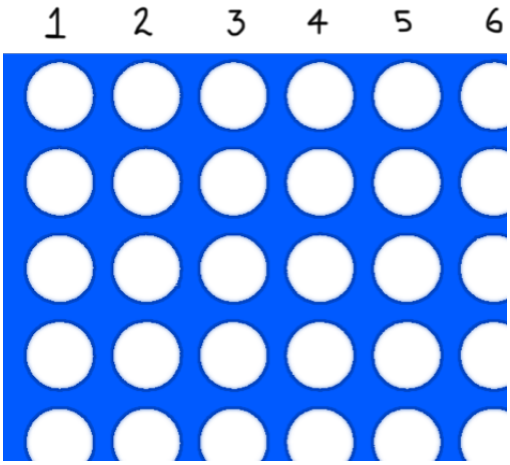
greedy melibatkan pencarian sebuah himpunan bagian, S, dari himpunan kandidat, C; yang dalam hal ini, S harus memenuhi beberapa kriteria yang ditentukan, yaitu S menyatakan suatu solusi dan S dioptimisasi oleh fungsi obyektif [1].

Perlu diperhatikan Algoritma *greedy* tidak selalu memberikan solusi yang optimal, tetapi mungkin dapat mendekati optimal. Seperti persoalan penukaran uang terdapat persoalan yang dapat diselesaikan secara optimal dan ada juga yang tidak, persoalan *knapsack problem*¸ Penjadwalan *job*, dll merupakan persoalan yang tidak selalu menghasilkan solusi optimal dengan menggunakan algoritma *greedy*. Sedangkan contoh algoritma *greedy* yang menghasilkan solusi optimal adalah *fractional knapsack problem*.

Walaupun algoritma *greedy* tidak selalu menghasilkan solusi yang optimal, namun algoritma ini dapat menyelesaikan permasalahan dengan cepat dibandingkan strategi algoritma lainya.

B. *Connect Four*

Pada sebelumnya sudah dijelaskan bahwa *connect four* merupakan permainan papan dua pemain setiap pemain akan memiliki token dengan warna berbeda setiap pemainnya yang nantinya akan dimasukkan secara vertikal ke dalam kotak permainan dengan ukuran 7 kolom dan 6 baris dengan pemain yang pertama membentuk empat diagonal, horizontal, atau vertikal dengan tokennya akan memenangkan permainan. Terdapat beberapa ukuran dari permainan *connect four*, namun pada makalah kali ini yang akan dibahas adalah permainan *connect four* dengan ukuran 7x6.



Gambar 2.1 *kolom yang dapat dipilih pemain.* (*source :* ark.io/blog/launching-html5-games-in-the-ark-desktop-wallet-part-four)

Pada permainan *connect four* pemain dapat memasukkan token dari kolom 1-7. pemain tidak bisa langsung menempatkan koin di tengah maupun di atas sesuai yang mereka inginkan dan jika masing-masing kolom sudah terisi penuh maka pemain tidak bisa lagi memasukkan token ke kolom tersebut. Meskipun banyak slot yang kosong, Pemain harus mengisi token terlebih dahulu mulai dari terbawah. Oleh karena itu diperlukan strategi yang tepat agar dapat memaksimalkan penempatan token pemain serta terhindar dari jebakan lawan dan menutup pergerakan lawan untuk menghasilkan *connect four* terlebih dahulu.

Pada permainan ini terdapat beberapa strategi yang dapat digunakan seperti dengan menaruh token sebanyak banyaknya di tengah untuk menguasai posisi, membuat *connect two*(2 token secara horizontal, vertikal, atau diagonal), sebanyak-banyaknya kemudian dikembangkan menjadi *connect three*(3 token secara horizontal, vertikal, atau diagonal), memasang jebakan pada posisi tertentu hingga posisi yang dapat digunakan lawan tinggal sedikit, dan masih banyak lagi. Permainan ini dapat diselesaikan dengan metode *brute force* namun konsekuensinya membutuhkan waktu yang lama. pada makalah ini Penulis akan menggunakan strategi *greedy* untuk membuat *bot* permainan *connect four* ini

III. Perancangan Algoritma

A. *Algoritma*

Pada makalah ini akan dibuat *bot* dan *game connect four* berupa CLI(*Command Line Interface*). Secara garis besar *greedy* yang dipakai adalah *greeedy* jumlah *connect three* lawan maupun bot, serta *greedy* dari penempatan posisi token. Pertama akan ditentukan elemen-elemen algoritma *greedy* yang terdapat pada *connect four*. Elemen-elemen algoritma *greedy* pada permainan *connect four* adalah sebagai berikut:

- Himpunan kandidat C, kandidat yang mungkin dipilih yaitu semua kolom 1-7 yang belum terisi penuh
- Himpunan solusi S, yaitu langkah yang sudah dipilih sebelumnya dalam hal ini merupakan kumpulan *slot* yang sudah terisi dengan token pemain
- Fungsi solusi, menentukan apakah himpunan kandidat yang dipilih sudah menghasilkan solusi, dalam hal ini akan diperiksa apakah dari himpunan solusi sudah terdapat susunan 4 token yang membentuk horizontal, vertikal, atau diagonal pada papan.
- Fungsi seleksi (*selection function*), akan dihitung cost setiap himpunan kandidat yang berupa kemungkinan untuk mendapatkan *connect three*, mencegah lawan *connect three*, dan cost dari posisi tersebut
- fungsi kelayakan, himpunan kandidat dianggap layak jika
- Fungsi objektif, mengambilkan gerakan yang menguntungkan bagi *bot* (memaksimumkan gerakan)

Secara garis besar algoritma *greedy* pada *bot* yang dibuat adalah sebagai berikut:

```
class Bot {
   int[][] cost
   int token_bot
}

function avoidTrap(int i)
{untuk menghindari jebakan yang dibuat lawan pada
posisi i}

function botConnect2(int i)
{ Menghitung jumlah connect two yang dapat dibuat bot
```

```
pada posisi i }

function opponentConnect2(int i)
{ Menghitung jumlah connect two yang dapat dibuat
lawan pada posisi i }


function count_cost()
{ menghitung cost dari semua himpunan kandidat, yaitu
kolom 1-7 yang hasilnya akan dimasukan ke dalam array
3-tuple cost }
Kamus

Algoritma
   costPosition()
   i interval 1...7
      opponentConnect2(i)
      botConnect2(i)
      avoidTrap(i)


function get_move()
{ mencari move yang akan diambil dengan memilih cost
yang memiliki value terbesar }
Kamus
   idxmax : int

Algoritma
   { inisialisasi }
   idxmax = 0

   { jika bot mendapatkan giliran ke 1 maka bot akan
agresif }
   if(token_bot = 1) then
      i interval 1...7
      if (cost[i][0] > cos[idxmax][0])
         idxmax = i
      else
         if (cost[i][0] = cos[idxmax][0]) then
            if (cost[i][1] > cost[idxmax[1]) then
                idxmax = i
            else
                if (cost[i][1] = cos[idxmax][1]) then
                   if (cost[i][2] > cost[idxmax[2])
                     then
                      idxmax = i

   { jika bot mendapatkan giliran ke 1 maka bot akan
defensif }
    else
      i interval 1...7
      if (cost[i][1] > cos[idxmax][1])
         idxmax = i
      else
         if (cost[i][1] = cos[idxmax][1]) then
            if (cost[i][0] > cost[idxmax[0]) then
                idxmax = i
            else
                if (cost[i][0] = cos[idxmax][0]) then
                   if (cost[i][2] > cost[idxmax[2])
                     then
                      idxmax = i
```

Class *Bot* akan memiliki *array* berukuran kolom 7 yang menunjukan cost setiap himpunan kandidat yang merupakan posisi kolom 1-7, dan baris 3 yang nantinya berisi *cost* untuk masing-masing kategori. kategori *cost* tersebut dapat dibagi menjadi:

1. Banyaknya *connect two* dari token *bot* disekitar posisi yang akan dipilih. Diharapkan jika dipilih langkah pada posisi tersebut *bot* dapat membuat *connect three* yang nantinya dapat dikembangkan lagi menjadi *connect four* untuk memenangkan permainan
2. Banyaknya *connect two* dari token player disekitar posisi yang akan dipilih. Diharapkanya *bot* dapat mencegah lawan membangun *connect three* dengan mengisi posisi tersebut dengan *bot*
3. Nilai posisi, posisi tengah pada papan lebih baik dari posisi yang berada di samping dikarenakan jika menempatkan token di tengah akan memberikan kesempatan yang lebih bagus untuk membentuk *connect four*, yaitu dapat dengan horizontal, vertikal, maupun diagonal, dibandingkan sisi samping yang kemungkinan lebih sulit untuk membentuk *connect four*.

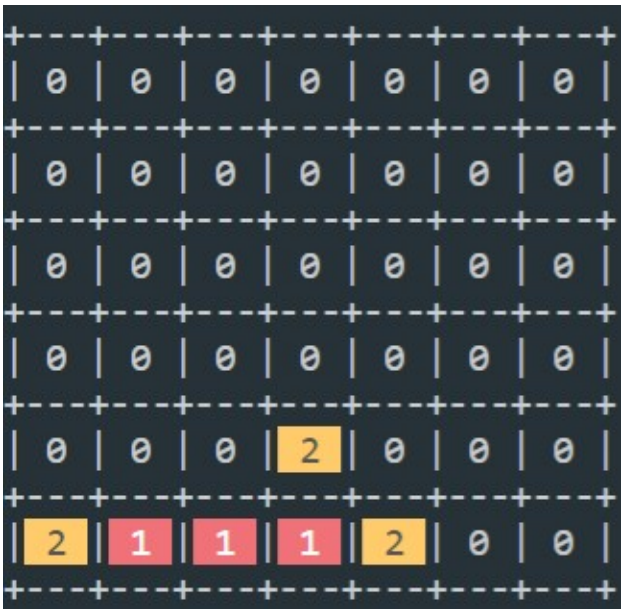
Setelah dihitung *cost* dari masing-masing maka selanjutnya akan dibandingkan *cost* dari setiap himpunan kandidat. penentuan strategi yang diambil dapat dibagi menjadi 2, jika *bot* merupakan pemain dengan giliran pertama maka *bot* akan bermain secara defensif, dikarenakan jika *bot* merupakan pemain ke dua otomatis *bot* akan ketinggalan satu langkah dari pemain ke satu. Oleh karena itu, *bot* akan bermain secara defensif yang berarti akan mencegah pergerakan-pergerakan sebelumnya dari pemain serta menunggu lawan melakukan kesalahan. Hal tersebut diimplementasikan dengan perbandingan *cost* akan mendahulukan perbandingan dari kategori *cost* ke dua yaitu banyaknya *connect two* dari token pemain di sekitar posisi yang akan dipilih. Jika perbandingan tersebut terdapat nilai yang sama maka akan dilanjutkan ke perbandingan kategori satu dan tiga. Kedua, jika *bot* merupakan pemain dengan giliran pertama maka *bot* akan agresif. Seperti yang sudah dijelaskan sebelumnya jika *bot* merupakan giliran pertama *bot* akan menang satu langkah dari lawan. Oleh karena itu, perbandingan akan dimulai dari kategori satu terlebih dahulu lalu jika terdapat cost yang memiliki nilai yang sama maka akan dilanjutkan perbandingan untuk kategori dua dan tiga. Posisi yang akan dipilih bot adalah posisi yang memiliki cost yang maksimum dengan urutan yang sudah dijelaskan sebelumnya.

Selain hal tersebut terdapat pendekatan *greedy* yang lainya yaitu untuk mencegah lawan menyelesaikan permainan dengan *connect four*. Ketika ditemukan posisi yang dapat menyebabkan lawan menyelesaikan permainan dengan *connect four* maka *bot* akan langsung meletakan token ke posisi tersebut untuk mencegah hal tersebut terjadi, selain itu juga terdapat strategi untuk menghindari jebakan lawan yang di mana jika kita menaruh token pada posisi tersebut dapat menguntungkan lawan untuk membuat *connect four* pada giliran-nya. Hal ini perlu dilakukan karena perlu diketahui sebelumnya *bot* hanya melihat keuntungan pada posisi tersebut dan mencegah posisi tersebut direbut lawan, namun hal tersebut dapat fatal jika posisi tersebut merupakan jebakan

yang dibuat lawan untuk melakukan *connect four* setelah giliran *bot* selesai. Intinya *bot* selain melihat posisi tersebut juga melihat satu langkah di depan yang akan terjadi selanjutnya.

B. Pengujian

Berikut akan dilakukan pengujian untuk lebih memahami mengenai strategi *greedy* yang telah dijelaskan sebelumnya. Pada pengujian pertama *bot* berperan sebagai pemain dengan giliran kedua.



Gambar 3.1 *Ilustrasi Bot bergerak defensif*

Pergerakan (move) yang dilakukan oleh pemain dan *bot* adalah sebagai berikut

	Move
Pemain	{4,3,2}
Bot	{4,5,1}

Tabel 3.1 *move dari pemain dan bot*

Sedangkan perhitungan cost untuk masing-masing move adalah sebagai berikut

Posisi	Move 1			Move 2			Move 3		
	c1	c2	c3	c1	c2	c3	c1	c2	c3
1	0	0	1	0	0	1	1	1	1
2	0	0	2	0	1	2	0	0	2
3	0	0	3	0	0	3	0	0	3
4	0	0	4	0	0	4	0	0	4
5	0	0	3	0	1	3	0	0	3
6	0	0	2	0	0	2	0	0	2
7	0	0	1	0	0	1	0	0	1

Tabel 3.2 *perhitungan cost di setiap move*

Pada *move Bot* ke-1 karena baru satu token yang keluar dari pemain maka untuk cost kategori pertama dan kedua tidak begitu membantu. pada waktu ini maka move akan ditentukan pada cost ke 3 yaitu berdasarkan posisi, telah diketahui sebelumnya bahwa posisi ditengah lebih utama daripada posisi di samping oleh karena itu *bot* akan mengambil move posisi 4 pada move ke 1.

Pada *move Bot* ke-2, pada tabel didapatkan nilai cost yang sama pada cost 2 dan cost 1 pada posisi 2 dan 5. Hal tersebut dikarenakan terdapat 2 token lawan secara berturut-turut horizontal pada posisi 4 dan 3 barisan ke-0. Hal tersebut dapat ditutup dengan meletakan token di samping kanan atau kiri dari token yang berjajar tersebut. Oleh karena itu perbandingan dilanjutkan oleh cost yang ke 3 di mana nilai posisi 5 lebih besar dari posisi 2, maka *bot* memilih posisi 5.

Pada *move Bot* ke-3, Berbeda dengan yang sebelumnya, pada langkah ini digunakan pendekatan greedy jika terdapat kemungkinan lawan melakukan connect 4 pada posisi tersebut maka *bot* akan menaruh token-nya untuk mencegah hal itu terjadi.

Gambar 3.2 *Menghindari jebakan*

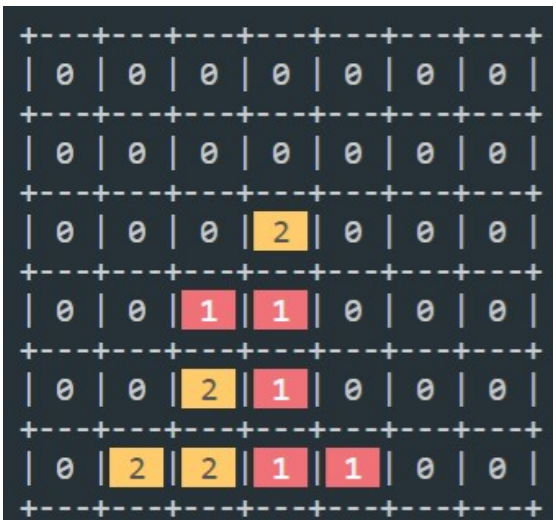
Sedangkan perhitungan cost pada *move* ke 20 adalah sebagai berikut

Posisi	cost1	cost2	cost3
1	0	0	1
2	0	0	2
3	0	0	0

4	0	0	0
5	0	1	3
6	0	2	2
7	0	0	1

Tabel 3.3 *perhitungan cost*

Pada gambar diatas digambarkan bahwa terdapat jebakan pada posisi ke-3 dimana jika *bot* menaruhkan token pada posisi tersebut meskipun kita mendapatkan *connect three*, namun pada giliran selanjutnya lawan akn mendapatkan *connect four*. Seperti yang sudah dijelaskan sebelumnya bahwa strategi greedy sebelumnya pada cost kategori pertama *bot* akan memilih posisi yang dapat membuat connect three paling banyak. Oleh karena itu perlunya strategi untuk menghindari jebakan tersebut dengan cara memperhitungkan langkah selanjutnya yang dilakukan lawan. Implementasi pada algoritma strategi tersebut akan mengubah semua cost pada posisi tersebut menjadi nol (dapat dilihat pada tabel posisi 4). Hal tersbut membuat *bot* tidak akan mungkin memilih posisi tersebut selama posisi lainnya tidak 0 semua juga. Dengan adanya hal tersebut dapat memperbaiki kualitas *bot* yang dibuat. Pada posisi ke 3 cost juga bernilai nol, namun hal tersebut bukan karena strategi sebelumnya, melkainkan dikarenakan posisi tersebut sudah terisi penuh. Oleh karena itu untuk menghindari bot memilih posisi yang sudah penuh semua kategori cost nya akan diganti menjadi 0.



Gambar 3.3 *Bot sebagai pemain 1*

	Move
Bot	{4,4,4,2,5}
Pemain	{3,3,4,1}

Tabel 3.4 *move bot sebagai pemain giliran pertama*

Pada bagian sebelumnya, *bot* berpean sebagai pemain kedua. Dimana saat *bot* berperan sebagai pemain kedua, *bot* akan lebih fokus untuk defensif. Pada gambar 3.3 menunjukan strategi agresif yang dilakukan *bot* ketika sebagai pemain dengan giliran pertama. Dapat dilihat pada langkah ke-3 *bot* memilih posisi 4 untuk menciptakan *connect four*. Jika menggunakan strategi defensif sebelumnya *bot* akan menutup pergerakan dari lawan dengan cara memilih posisi 3. Namun karena *bot* menang satu langkah maka *bot* lebih memilih melanjutkan membentuk *connect four* baru setelah pergerakanya ditutup *bot* akan menghalangi pergerakan lawan dikarenakan belum ada token yang bisa dikembangkan menjadi *connect four*. Selanjutnya *bot* akan memulai untuk agresif kembali dapat dilihat pada move 5, *bot* memilih posisi 5 untuk membentuk *connect three*

C. Analisis

Seperti yang telah diperlihatkan sebelumnya *bot* yang dibuat menggunakan strategi *greedy* tentu bukan merupakan solusi yang optimal untuk menyelesaikan permasalahan connect-4 ini namun cukup cepat untuk memperhitungkan langkah yang akan diambil selanjutnya dengan menggunakan strategi-strategi yang sudah dijelaskan sebelumnya.

Secara singkat *bot* memiliki 2 tipe yaitu agresif jika *bot* mendapat giliran pertama dan defensif jika mendapat giliran kedua. pada mode *defensif bot* akan fokus untuk bertahan dengan cara menghalangi lawan untuk mendapatkan *connect three* atau mendapatkan posisi-posisi yang strategis. Strategi defensif ini digunakan karena *bot* tertinggal satu langkah dibelakang karena menjadi pemain ke dua. Sedangkan pada mode agresif *bot* akan cenderung untuk membuat *connect three* dan akan melakukan defense jika memang sangat dibutuhkan atau tidak ada token yang dapat dikembangkan menjadi *connect three*. Strategi agresif ini digunakan karena *bot* memiliki keuntungan satu langkah lebih dahulu dari lawan karena sebagai pemain ke satu. *Bot* juga memiliki beberapa tambahan lain seperti untuk menghindari jebakan yang dibuat lawan dengan cara melihat satu langkah kedepan dari lawan, mencegah *connect four* dari lawan serta mencari posisi yang bisa menghasilkan *connect four* untuk *bot* dengan cara mencari *connect three* jika ada.

Namun demikian, *bot* memiliki kelemahan di mana masih belum cukup baik untuk menentukan jika terdapat poin yang sama hal ini terkadang membuat *bot* yang awalnya unggul menjadi tidak diuntungkan, kemudian pada mode defensif, *bot* terlalu fokus untuk mencegah lawan untuk mendapatkan keuntungan dengan meningkatkan jumlah *connect three*. Hal tersebut menyebabkan *bot* tidak bisa membangun *connect three*-nya sendiri atau menghubungkan antara token *bot* sehingga kemungkinan besar *bot* akan mengalami *draw* atau seri daripada memenangkan permainan. Sedangkan pada mode agresif *bot* cenderung untuk meningkatkan jumlah *connect* antara token, sehingga lupa untuk menghentikan pergerakan lawan, kasus terburuknya adalah ketika lawan mendapatkan *connect three* yang dapat *connect four* pada dua arah maka saat *bot* melakukan pertahanan akan menjadi sia-sia. Pembuatan *bot* ini dikhususkan untuk melawan orang awam untuk permainan *connect four*, sehingga *bot* dapat bersaing dengan orang tersebut.

IV. Kesimpulan dan Saran

Bot yang penulis buat tentu masih jauh dari kata sempurna, namun sudah cukup untuk menyelesaikan permasalahan permainan *connect four*. Meskipun algoritma yang dibuat masih jauh dari solusi optimal karena menggunakan strategi algoritma *greedy,* namun algoritma tersebut mungkin dapat melawan orang yang masih awam untuk permainan *connect four*. Program yang penulis buat merupakan *bot* serta permainan *connect four* yang berbasis CLI (*Command Line Interface*) yang dapat dilihat pada github dibawah ini. Penulis sangat terbuka menerima saran dan masukkan dari pembaca.

Video Link at Youtube

youtu.be/XhQKuxdEK20

Link Github

github.com/lizardyy/connect-4-greedy

Ucapan Terima Kasih

Penulis mengucapkan terima kasih serta puji dan syukur kepada Tuhan Yang Maha Esa karena berkat rahmatnya penulis dapat menyelesaikan makalah ini dengan tepat waktu. Penulis juga mengucapkan terima kasih kepada seluruh tim dosen mata kuliah IF2211 Strategi Algoritma yang telah dengan sabar membimbing penulis selama satu semester. Tanpa bimbingan dan ajaran dari tim dosen IF2211 makalah ini tidak mungkin dapat terselesaikan dengan baik.

Penulis juga berterima kasih kepada seluruh tim asisten mata kuliah IF2211 yang telah dengan sabar memberi ilmu serta membimbing selama mata kuliah IF2211 semester ini baik secara langsung saat demo maupun tidak langsung. Penulis juga berterima kasih kepada seluruh teman penulis di Teknik Informatika ITB angkatan 2020 yang telah membantu penulis menemukan judul makalah serta memberi masukkan dan saran pada proses pengerjaan makalah ini sehingga makalah ini dapat menjadi lebih baik lagi.

Terakhir, penulis juga berterima kasih kepada orang tua penulis yang berkat doa dan dukungannya memungkinkan penulis untuk menyelesaikan mata kuliah IF2211 Strategi Algoritma dan menyelesaikan makalah ini dengan tepat waktu.

Referensi

[1] Munir, Rinaldi. Algoritma Greedy. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf. 2022.

[2] Gambtier. Connect Four. https://gambiter.com/connection/Connect_four.html . 2022.

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, 20 Mei 2022

Mahesa Lizardy 13520116