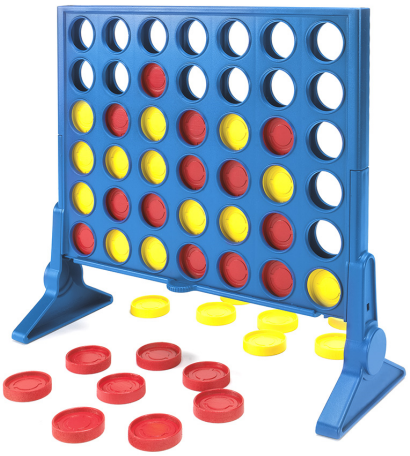
Penerapan Strategi Algoritma *Greedy* pada Permainan Connect Four

Aditya Bimawan (13519064)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): aditditto@gmail.com

***Abstract*—Untuk mengisi waktu luang, banyak orang yang memainkan *board game* atau permainan papan. Salah satu dari banyak *board game* yang ada adalah *connect four*, sebuah permainan yang dapat dimainkan oleh dua pemain. Pada permainan ini kedua pemain bergantian menjatuhkan "piringan" masing masing pemain ke papan dan berusaha membuat 4 koin berposisi berurutan secara horizontal, vertikal, ataupun diagonal. Pada makalah ini akan dibahas sebuah strategi bermain *connect four* yang memanfaatkan algoritma *greedy* untuk berusaha mencapai kemenangan. Dengan algoritma *greedy*, didapat sebuah strategi bermain connect four yang meskipun tidak menjamin kemenangan, tetapi juga sulit untuk dikalahkan sehingga cocok untuk diaplikasikan sebagai *bot* dari sebuah *game* komputer.**

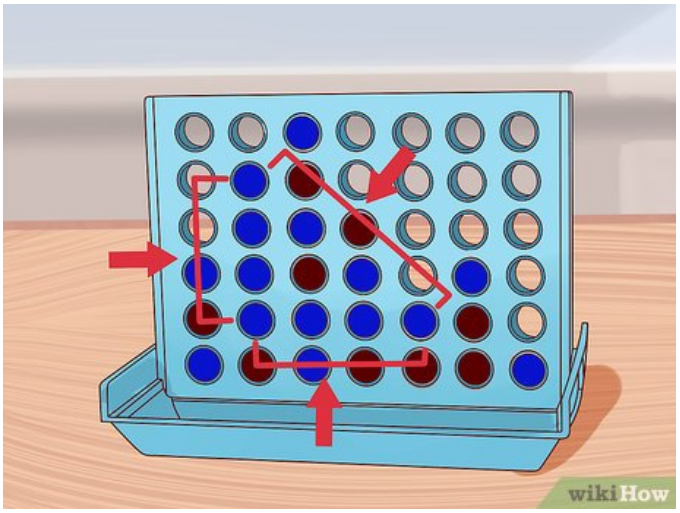
Keywords—connect four; algoritma greedy

I. Pendahuluan



Gambar 1. Contoh papan *connect four* (Sumber: https://www.shopbecker.com/Connect-Four-Game-MB4430/)

Connect Four adalah *board game* untuk dua orang pemain yang pertama kali dibuat pada tahun 1974, dimana masing-masing pemain mewakili satu warna dan memasukkan piringan ke salah satu jalur secara bergantian. Pada umumnya, jalur yang ada pada papan sejumlah 7 jalur. Piringan yang dimasukkan akan jatuh dan bertumpuk dengan piringan lain apabila sudah ada yang menempati. Seorang pemain dapat dikatakan memenangkan permainan apabila pemain tersebut berhasil menyusun piringannya sedemikian sehingga didapat satu baris dengan 4 piringan berturut-turut yang tersusun baik secara horizontal, vertikal, ataupun diagonal.

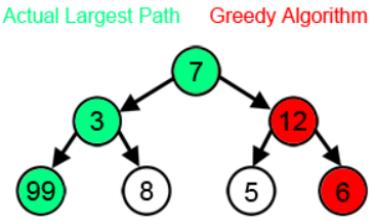


Gambar 2. Beberapa cara untuk memenangkan *connect four* (Sumber: https://www.wikihow.com/Play-Connect-4)

Pada makalah ini, akan dibuat sebuah strategi bermain permainan *connect four* dengan memanfaatkan strategi algoritma greedy. Diharapkan dengan membuat sebuah strategi greedy dapat dihasilkan sebuah strategi yang dapat menghasilkan kemenangan.

II. Landasan Teori

A. Algoritma Greedy



Gambar 3. Ilustrasi algoritma *greedy* yang gagal untuk menemukan jalur maksimum pada suatu struktur data *tree*. (Sumber: https://vikram-bajaj.gitbook.io/cs-gy-6033-i-design-and-analysis-of-algorithms-1/chapter1)

Algoritma greedy adalah sebuah algoritma yang cukup populer dan dapat digunakan untuk menyelesaikan berbagai macam masalah jenis optimasi. Masalah optimasi sendiri terdiri dari dua jenis, yaitu maksimalisasi dan minimalisasi. Algoritma greedy mengikuti heuristik tertentu yaitu mengambil pilihan maksimum yang dapat diambil pada setiap langkah. Dengan mengambil pilihan dengan nilai maksimum pada tiap langkah, algoritma greedy berharap untuk mendapatkan hasil yang optimal. Karena hal inilah algoritma ini dinamakan *greedy* atau rakus.

Salah satu penerapan algoritma greedy yang dapat menggambarkan sifat-sifatnya adalah masalah penukaran koin, atau *coin exchange problem*. Pada persoalan ini diberikan uang senilai A, dan disediakan beberapa jenis koin yang jumlahnya sangat banyak. Yang harus dilakukan adalah menukar uang yang sejumlah A tadi dengan koin yang ada, dengan jumlah yang minimal.

Misal jumlah uang A yang harus ditukar adalah 32, maka dengan strategi greedy yaitu mengambil nilai terbaik yang ada saat ini atau "*take what you can get now*", akan diambil koin dengan nilai terbesar pada tiap langkah, sehingga apabila tersedia nominal koin 1, 5, 10, dan 25 maka akan diambil yang terbesar terlebih dahulu, yaitu 25, kemudian dari sisa uang yaitu 32-25 = 7, diambil lagi koin dengan nominal terbesar yaitu 5, dan seterusnya hingga selesai, dengan hasil akhir koin yang diambil adalah nominal 25, 5, 1, 1. Pada kasus ini, solusi yang didapat adalah solusi yang optimal, yang didapat dengan mengambil nilai optimal yang ada pada tiap langkah.

Meskipun dengan mengambil optimum lokal algoritma greedy mungkin mendapat optimum global, tetapi hal itu tidak dijamin. Salah satu kasus dimana algoritma greedy tidak dapat mencapai optimum global adalah pada persoalan yang sama, yaitu masalah penukaran koin, tetapi dengan nominal koin 10, 7, dan 1, dan nilai uang A yang harus ditukar adalah 15. Pada kasus ini, algoritma greedy pertama-tama akan mengambil nominal terbesar yaitu 10, tetapi kemudian hanya bisa mengambil nominal 1 sebanyak 5 kali, sehingga hasil yang didapat adalah 10, 1, 1, 1, 1, 1 atau sebanyak 6 koin. Sedangkan solusi optimal global yang benar adalah 7 + 7 + 1, atau hanya 3 koin.

Secara umum, algoritma greedy terdiri dari beberapa elemen, yaitu:

- Himpunan kandidat C, yaitu kandidat yang dapat dipilih pada setiap langkah.
- Himpunan solusi S, yaitu kandidat-kandidat yang sudah dipilih.
- Fungsi solusi, menentukan apakah kandidat yang sudah dipilih menghasilkan solusi.
- Fungsi seleksi (*selection function*), fungsi yang memilih kandidat berdasarkan strategi greedy yang bersifat heuristik.
- Fungsi kelayakan, memeriksa apakah kandidat tertentu layak atau dapat dimasukkan ke himpunan solusi.
- Fungsi objektif, memaksimumkan atau meminimumkan.

Sehingga dapat dibilang algoritma greedy adalah algoritma yang melibatkan pencarian sebuah himpunan solusi S, dari himpunan kandidat C, dimana S memenuhi beberapa kriteria yang ditentukan, yaitu S menyatakan suatu solusi dan S di optimisasi oleh fungsi objektif.

Dari definisi di atas, dapat dibuat sebuah skema umum untuk implementasi algoritma greedy yaitu sebagai berikut:

```
function greedy(C : himpunan_kandidat) → himpunan_solusi
{ Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy }
Deklarasi
  x : kandidat
  S : himpunan_solusi

Algoritma:
  S ← { }     { inisialisasi S dengan kosong }
  while (not SOLUSI(S)) and (C ≠ { } ) do
      x ← SELEKSI(C)     { pilih sebuah kandidat dari C}
      C ← C – {x}        { buang x dari C karena sudah dipilih }
      if LAYAK(S ∪ {x}) then      { x memenuhi kelayakan untuk dimasukkan
          S ← S ∪ {x}     { masukkan x ke dalam himpunan solusi }
      endif
  endwhile
  {SOLUSI(S) or C = {} }

  if SOLUSI(S) then     { solusi sudah lengkap }
    return S
  else
    write('tidak ada solusi')
  endif
```

Dengan algoritma *greedy* yang sudah dijelaskan di atas, dapat dilihat bahwa algoritma ini sebenarnya tidak menjamin optimalitas hasil yang didapat. Meskipun demikian, algoritma *greedy* masih memiliki kelebihan, yaitu kesederhanaan dan kemudahan dalam pembuatan algoritmanya, yang dapat menghasilkan algoritma yang memiliki kompleksitas rendah, tetapi hasil yang mendekati optimal. Sehingga, algoritma *greedy* ini baik untuk permasalahan dimana optimalitas tidak diwajibkan, sehingga dapat dihasilkan algoritma yang cepat dan sederhana tetapi dengan hasil yang cukup mendekati hasil optimal.

Meskipun algoritma greedy tidak menjamin optimalitas pada setiap permasalahan, ada juga kasus dimana algoritma greedy dapat mencapai hasil optimal pada setiap percobaan. Contoh dari permasalahan dimana algoritma greedy dapat menjamin hasil optimal adalah *Fractional Knapsack Problem*, yaitu variasi dari *Integer Knapsack Problem* dimana solusi boleh berbentuk pecahan atau fraksi.

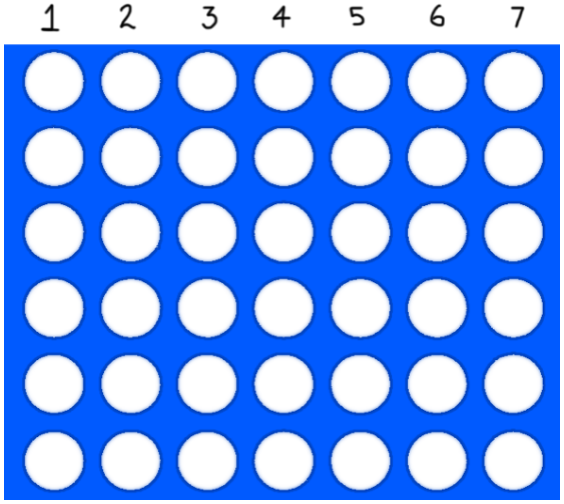
Permasalahan *Fractional Knapsack* adalah maksimasi dari jumlah total objek yang diambil, tetapi objek yang diambil boleh berupa pecahan.

Properti objek				*Greedy by*		
i	w_i	p_i	p_i/w_i	*profit*	*weight*	*density*
1	18	25	1,4	1	0	0
2	15	24	1,6	2/15	2/3	1
3	10	15	1,5	0	1	1/2
Total bobot				20	20	20
Total keuntungan				28,2	31,0	31,5

Pada gambar di atas dapat dilihat, bahwa apabila strategi yang diterapkan berupa *greedy by density*, maka hasil dari algoritma greedy akan selalu optimal. Tetapi, hal ini hanya terjadi pada beberapa permasalahan, tidak semua.

B. *Connect Four*

Seperti yang sudah dijelaskan pada bagian I, connect four adalah permainan papan yang dimainkan dua orang, dimana orang pertama yang berhasil menyusun piringannya sehingga terbentuk satu barisan dengan panjang empat piringan baik secara horizontal, vertikal, ataupun diagonal.



Gambar 4. Gerakan yang dapat dipilih dalam permainan *connect four*. (Sumber: https://ark.io/blog/launching-html5-games-in-the-ark-desktop-wallet-part-four)

Sejatinya gerakan yang dapat dilakukan pada tiap giliran pemain di *connect four* hanya sejumlah baris yang ada (umumnya 7), karena meskipun ada banyak slot yang kosong pada papan, tetapi yang bisa diisi hanyalah yang slot kosong yang berada pada posisi paling bawah. Karena ini, yang perlu dilakukan untuk mencari gerakan optimal pada tiap giliran adalah membandingkan tiap baris dan mengambil yang paling optimal.

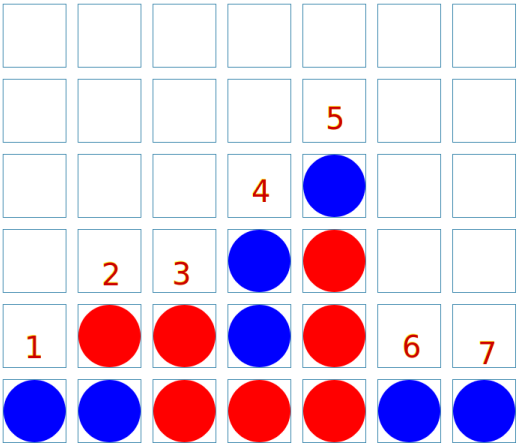
Untuk menyederhanakan permasalahan, makalah ini hanya akan membahas permainan *connect four* dengan tujuh jalur dan peraturan kemenangan yang mengharuskan menyusun empat piringan berturut-turut, tetapi sebenarnya strategi yang dijelaskan pada makalah ini dapat diterapkan untuk papan dengan ukuran yang berbeda dan peraturan untuk menyusun N piringan berturut-turut.

Dalam bermain *connect four,* ada beberapa hal yang harus diperhatikan untuk memaksimalkan kemungkinan menang, yaitu memperhatikan gerakan lawan supaya tidak terjebak dan kalah, kemudian selalu berusaha untuk mengambil gerakan yang memenangkan permainan.

III. Pembahasan

Dari landasan teori yang sudah dijelaskan di atas, kita dapat membuat sebuah strategi berdasarkan algoritma greedy untuk permainan *connect four*. Secara garis besar, strategi yang akan kita buat adalah strategi yang mengambil gerakan terbaik pada tiap giliran, dengan gerakan yang terbaik dinilai dari apakah gerakan ini membuat menang ataupun apabila tidak diambil akan membuat kalah. Apabila tidak ada gerakan tertentu yang akan membuat kalah atau menang, maka diambil gerakan yang akan membuat baris paling panjang. Dengan ini, dapat dibuat elemen-elemen dari strategi greedy pada connect four sebagai berikut:

- Himpunan kandidat C, kandidat yang mungkin, yaitu semua kolom 1-7 yang belum terisi penuh.
- Himpunan solusi S, yaitu kandidat-kandidat yang sudah dipilih.
- Fungsi solusi, menentukan apakah kandidat yang sudah dipilih menghasilkan solusi. Memeriksa apakah dari himpunan solusi terbentuk sebuah barisan horizontal, vertikal, ataupun diagonal pada papan.
- Fungsi seleksi (*selection function*), fungsi yang memilih kandidat berdasarkan strategi greedy yang bersifat heuristik. Fungsi inilah yang dapat divariasikan sehingga menjadi strategi *greedy* yang baik.
- Fungsi kelayakan, memeriksa apakah kandidat tertentu layak atau dapat dimasukkan ke himpunan solusi. Pada permainan *connect four* berarti apakah suatu kolom sudah penuh apa belum.
- Fungsi objektif, memaksimumkan atau meminimumkan. Memaksimalkan bobot gerakan yang diambil.



Gambar 5. Contoh permainan *connect four* yang sedang berlangsung.

Pada contoh diatas, apabila tiap kolom dinomori dengan nomor 1 untuk kolom paling kiri, 2 untuk kedua dari kiri, dan seterusnya, maka bobot tiap gerakan untuk pemain dengan warna merah adalah sebagai berikut:

Kolom	Bobot vertikal	Bobot horizontal	Bobot diagonal	Bobot akhir
1	0	2	0	2
2	1	0	1	1
3	1	0	1	1
4	0	0	0	0
5	0	0	0	0
6	0	1	1	1
7	0	0	0	0

Tabel 1. Tabel bobot tiap kolom untuk strategi *greedy* agresif.

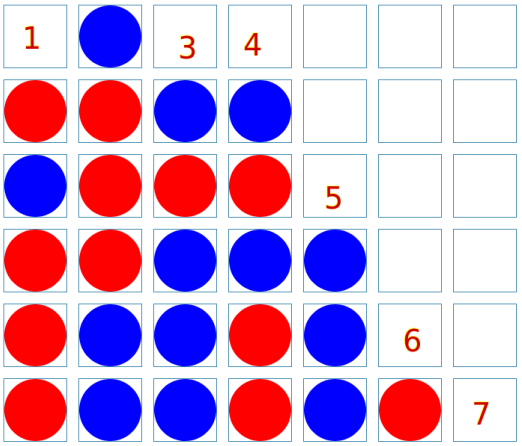
Dengan algoritma greedy, pemain dengan warna merah akan mengambil kolom 1, karena dengan menempatkan piringan di kolom 1, akan terbentuk barisan sepanjang 3 piringan, meskipun barisan 3 piringan ini tidak dapat diperpanjang menjadi sepanjang 4 piringan pada giliran selanjutnya.

Pada contoh di atas dapat dilihat, bahwa dengan algoritma greedy yang berusaha mencapai kemenangan dengan membuat barisan yang paling panjang, tidak dijamin bahwa gerakan yang dilakukan adalah gerakan yang akan membuat menang, karena belum tentu hasil dari susunan yang dibuat dapat dilanjutkan untuk menjadi baris dengan panjang 4 piringan.

Dari pembahasan di atas, disimpulkan bahwa strategi greedy yang mengejar kemenangan belum tentu menjamin kemenangan. Hal ini juga berhubungan dengan strategi bermain *connect four* secara umum, yaitu harus memiliki *foresight* atau "melihat ke masa depan". Sehingga tidak cocok diimplementasikan dengan strategi *greedy*.

Lantas, bagaimana strategi *greedy* yang lebih baik? Karena strategi *greedy* yang agresif tidak terlalu efektif, maka alternatif yang dapat diambil untuk implementasi algoritma *greedy* adalah membuat algoritma *greedy* untuk bermain *connect four* secara pasif atau mengutamakan pertahanan. Dalam kasus bermain *connect four*, bermain secara defensif berarti menghindari kekalahan. Maka, fungsi seleksi yang dibuat akan berupa fungsi yang menghitung bobot dari gerakan tersebut bagi musuh.

Strategi defensif ini akan dicoba untuk kasus sebagai berikut:



Gambar 6. Contoh permainan *connect four* yang sedang berlangsung.

Dengan strategi defensif, tabel bobot yang dihasilkan untuk pemain dengan piringan berwarna merah adalah sebagai berikut:

Kolom	Bobot vertikal	Bobot horizontal	Bobot diagonal	Bobot akhir
1	0	1	0	1
2	-	-	-	- (penuh)
3	1	1	1	1
4	1	0	1	1
5	3	0	3	3
6	0	1	1	1
7	0	0	0	0

Tabel 2. Bobot tiap kolom untuk strategi *greedy* defensif dalam permainan *connect four*.

Dari tabel tersebut, dapat dilihat bahwa algoritma *greedy* yang bersifat defensif akan menaruh piringan pada kolom 5, karena apabila menaruh piringan pada kolom lain, pemain lawan akan mengisi kolom 5 dan mencapai kemenangan.

Perhatikan juga bahwa dengan mengisi kolom 5 pemain dengan strategi *greedy* yang bersifat defensif akan mencapai kemenangan, tetapi hal ini hanya berupa kebetulan dan bukan efek dari strategi *greedy* yang diterapkan, karena strategi yang diterapkan hanya menimbang bobot dari gerakan musuh, tidak memedulikan apakah gerakan tersebut membuat diri sendiri menang atau tidak. Apabila digunakan strategi agresif pada kasus ini, maka gerakan yang diambil juga sama, yaitu kolom

5, tetapi untuk alasan yang berbeda, yaitu karena ia memiliki piringan tetangga berwarna merah yang paling banyak.

Karena strategi yang diterapkan pada kasus ini bersifat defensif, maka hasil akhir dari strategi ini pada umumnya adalah seri atau kalah, dengan kemenangan hanya muncul karena kebetulan.

IV. Analisis

Dengan melihat hasil penerapan algoritma *greedy* pada permainan *connect four*, didapatkan dua variasi dari algoritma *greedy,* yaitu algoritma *greedy* yang bersifat agresif dan algoritma *greedy* yang bersifat defensif.

Untuk algoritma *greedy* yang bersifat agresif, dapat diperhatikan bahwa algoritma bersifat naif dan "buta" terhadap gerakan musuh, sehingga mudah untuk dijebak dan dikalahkan. Selain itu, dengan strategi *greedy*, maka algoritma yang bersifat agresif akan selalu mengambil kolom yang menghasilkan barisan paling panjang pada saat itu, meskipun barisan tersebut tidak dapat diperpanjang menjadi barisan yang membawa kemenangan. Kasus ini dapat dilihat pada contoh permainan *connect four* yang sedang berlangsung pada Gambar 4.

Di sisi lainnya, untuk algoritma *greedy* yang bersifat defensif, dapat dibilang bahwa algoritma ini berupa kebalikan dari *greedy* yang bersifat agresif, karena ia bergerak hanya berdasarkan bobot keuntungan lawan, dan "buta" terhadap potensi dirinya sendiri untuk menang. Dengan strategi yang defensif ini, algoritma akan lebih sulit dikalahkan, tetapi juga sangat jarang menang, dimana ia hanya akan menang apabila pihak lawan melakukan sebuah blunder dan membiarkan algoritma *greedy* defensif meletakkan piringan di tempat yang membuatnya menang.

Secara keseluruhan penulis rasa bahwa strategi bermain *connect four* dengan algoritma *greedy* yang lebih baik adalah strategi *greedy* yang bersifat defensif, karena meskipun kemungkinan menangnya lebih kecil, tetapi kemungkinan kalahnya juga kecil, dan memiliki kemungkinan lebih besar untuk mendapat hasil akhir seri. Hal ini lebih baik dari *greedy* yang bersifat agresif, karena kemungkinan menang dari strategi *greedy* yang bersifat agresif hanya berbeda sedikit, tetapi kemungkinan untuk mendapat hasil akhir seri jauh lebih kecil.

Dapat dilihat bahwa strategi *greedy* yang dibuat, baik yang agresif maupun defensif, kurang efektif dalam memenangkan permainan *connect four*. Hasil ini didapat karena *connect four* merupakan permainan strategi yang membutuhkan *foresight* atau perencanaan untuk masa depan, sehingga dapat membuat "jebakan" untuk musuh dan memenangkan permainan. Strategi algoritma *greedy* tidak menjamin untuk menghasilkan hasil yang optimal untuk persoalan seperti ini, karena yang dilakukan oleh algoritma *greedy* adalah mengambil gerakan yang paling optimal untuk tiap giliran, dan tidak memikirkan langkah selanjutnya atau *big picture*, karena terkadang lebih baik mengambil gerakan yang bukan berupa gerakan "terbaik" untuk giliran itu, tetapi membantu untuk mencapai kemenangan di giliran selanjutnya.

V. Kesimpulan

Setelah berusaha untuk membuat algoritma yang mampu memenangkan permainan connect four dengan memanfaatkan strategi greedy, didapatkan dua buah variasi strategi, yaitu strategi greedy yang bersifat agresif dan yang bersifat defensif.

Di antara kedua strategi tersebut, dipilih strategi yang bersifat defensif sebagai strategi yang lebih baik, karena memiliki kemungkinan kalah yang lebih kecil. Tetapi, perlu diperhatikan bahwa kedua algoritma yang dihasilkan tidak menjamin kemenangan. Hal ini disebabkan algoritma greedy yang bersifat "take what you can get now" sehingga algoritma greedy tidak mampu untuk merencanakan strategi-strategi yang dapat "menjebak" lawan sehingga sulit untuk mencapai kemenangan dalam permainan strategi yang membutuhkan perencanaan seperti connect four.

Meskipun hasil algoritma yang dibuat tidak dapat menjamin kemenangan, algoritma ini masih memiliki kegunaan, misalnya sebagai bot dalam game komputer connect four, sehingga pemain dapat bermain sendiri tanpa harus mencari lawan. Dengan strategi defensif yang tidak menjamin kemenangan tetapi juga cukup sulit dikalahkan, algoritma ini dapat menjadi bot yang baik sehingga menarik perhatian pemain.

VI. Ucapan Terima Kasih

Puji syukur penulis panjatkan kepada Allah SWT, karena atas berkat dan rahmatnya makalah ini dapat dibuat dan diselesaikan tepat waktu, dan juga telah menjaga penulis, keluarga, dan para dosen yang sudah mengajarkan materi dengan sepenuh hati selama pandemi COVID-19 ini. Penulis juga tidak lupa untuk mengucapkan terima kasih kepada kedua orang tua yang sudah mendukung dalam segala cara selama ini, dan kepada keluarga yang sudah membuat penulis ingin menjadi lebih baik.

Penulis juga berterima kasih kepada Dr. Nur Ulfa Maulidevi, ST., M.Sc. selaku dosen pengampu mata kuliah Strategi Algoritma pada kelas K2, yang sudah mengajarkan materi dengan sepenuh hati dan kesabaran. Tanpa ilmu yang beliau berikan makalah ini tidak dapat diselesaikan.

Video link at Youtube

Makalah ini juga dipublikasikan dalam bentuk video yang di-upload pada platform YouTube. Video tersebut dapat ditonton pada link berikut: https://youtu.be/7q3u0EJILAM

Referensi

[1] Munir, Rinaldi. Diktat Kuliah IF2211 Strategi Algoritma. Bandung: Teknik Informatika Institut Teknologi Bandung, 2009.

[2] Wikipedia. https://en.wikipedia.org/wiki/Connect_Four, diakses pada 10 Mei 2021, pukul 20:33

[3] https://connect-4.org/ , diakses pada 10 Mei 2021, pukul 21:44

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, 26 April 2021

Aditya Bimawan
13519064