Penerapan Algoritma Greedy pada Permainan Othello

Ahmad Ghulam Ilham - 13521118
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13521118@std.stei.itb.ac.id

***Abstrak*—Othello merupakan permainan strategi dua pemain pada papan berukuran 8x8 kotak menggunakan sejumlah koin yang memiliki warna sisi berbeda, terang dan gelap. Umumnya, warna papan adalah hijau, jumlah koin adalah 64, warna kedua sisi koin adalah hitam dan putih. Kedua pemain menentukan warna sisi koin yang dimainkan dan berusaha untuk mempertahankan jumlah koin sesuai warna sisi yang ditentukan. Pemenang permainan adalah pemain yang memiliki jumlah koin lebih banyak pada akhir permainan.**

Keywords—Othello, Strategi, Algoritma Greedy

I. PENDAHULUAN

Permainan papan strategi tidak asing di kalangan masyarakat. Othello merupakan salah satu permainan strategi papan yang cukup dikenal. Othello sendiri merupakan pengembangan dari permainan bernama "Reversi".

Sekitar 140 tahun yang lalu, lebih tepatnya pada 1883, Othello, versi modern Reversi, dipatenkan oleh Lewis Waterman. Pada 1971, perusahaan Jepang Tsukuda Original merilis Othello yang menyebabkan popularitas permainan ini meningkat.

Othello pun telah menjadi permainan kompetitif yang diakui secara internasional. Pada 1977, International Othello Association didirikan untuk mengoordinasikan kompetisi Othello di tingkat internasional.

Aturan Othello memang sederhana, tetapi strategi yang dapat digunakan dalam permainan ini cukup kompleks. Algoritma dan teknik memainkan Othello menjadi salah satu hal yang dikembangkan, terutama dalam bidang kecerdasan buatan dan komputasi.

Makalah ini bertujuan untuk menerapkan salah satu materi Strategi Algoritma, algoritma *greedy*, dalam mendapatkan langkah atau gerakan terbaik dalam satu giliran pemain pada permainan Othello.

II. LANDASAN TEORI

A. *Algoritma Greedy*

Algoritma *greedy* merupakan metode sederhana untuk memecahkan persoalan optimasi, yaitu mencari solusi optimal dari suatu persoalan. Hanya ada dua macam persoalan optimasi, yaitu:

1. Maksimasi (*maximization*)
2. Minimasi (*minimization)*

Contoh persoalan optimasi adalah persoalan penukaran uang (*coin exchange problem*): diberikan uang senilai *A*. Tersedia uang koin-koin dalam jumlah yang banyak. Tukar *A* dengan koin-koin uang yang ada. Berapa jumlah minimum koin yang diperlukan untuk penukaran tersebut. Berdasarkan deskripsi tersebut, persoalan penukaran uang termasuk persoalan minimasi.

Greedy, diambil dari bahasa Inggris yang berarti rakus, atau tamak. Prinsip *greedy* adalah "*take what you can get now!*". Algoritma *greedy* membentuk solusi Langkah per Langkah.

Pada setiap Langkah, terdapat banyak pilihan yang perlu dievaluasi. Oleh karena itu, pada setiap Langkah harus dibuat keputusan terbaik dalam menentukan pilihan dan tidak bisa mundur ke Langkah sebelumnya.

Pada setiap Langkah, akan dipilih optimum lokal. Harapannya, setiap pilihan optimum lokal akan mengarah menuju solusi optimum global.

Elemen-elemen algoritma *greedy*:

1. Himpunan kandidat, *C*: berisi kandidat yang akan dipilih pada setiap Langkah (misal: simpul/sisi di dalam graf, job, task, koin, benda, karakter)
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. Strategi *greedy* ini bersifat heuristic
5. Fungsi kelayakan (*feasible*): memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak)
6. Fungsi obyektif: memaksimumkan atau meminimumkan

Berikut merupakan skema umum algoritma *greedy*:

```
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 ke dalam himpunan solusi }
          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
```

Gambar 2.1. Skema Umum Algoritma Greedy

Sumber: Slide Strategi Algoritma

Hal yang perlu diingat pada algoritma *greedy*, optimum global yang dicapai belum tentu merupakan solusi optimum (terbaik), bisa jadi hanya solusi *sub-optimum* atau *pseudo-optimum*. Alasan yang menyebabkan hal tersebut adalah:

1. Algoritma *greedy* tidak beroperasi secara menyeluruh terhadap semua kemungkinan solusi yang ada (sebagaimana pada metode *exhaustive search*)
2. Terdapat beberapa fungsi seleksi yang berbeda, sehingga kita harus memilih fungsi yang tepat jika kita ingin menghasilkan solusi optimal.

Pada Sebagian persoalan, algoritma *greedy* tidak selalu berhasil memberikan solusi yang optimal, namun sub-optimal. Jika solusi terbaik mutlak tidak terlalu diperlukan, maka algoritma *greedy* dapat digunakan untuk menghasilkan solusi hampiran (*approximation*) daripada menggunakan algoritma yang kebutuhan waktunya eksponensial untuk menghasilkan solusi yang eksak.

Sebagai contoh, mencari tur dengan bobot minimal pada persoalan TSP untuk jumlah simpul (n) yang banyak dengan algoritma *brute force* dibutuhkan waktu komputasi yang lama untuk menemukannya. Dengan algoritma *greedy*, meskipun tur dengan berbobot minimal tidak dapat ditemukan, namun solusi dengan algoritma *greedy* dianggap sebagai hampiran solusi optimal.

Namun, bila algoritma *greedy* dapat menghasilkan solusi optimal, maka keoptimalannya itu harus dapat dibuktikan secara matematis. Membuktikan optimalitas algoritma *greedy* secara matematis adalah tantangan tersendiri. Lebih mudah memperlihatkan algoritma *greedy* tidak selalu optimal dengan menunjukkan *counterexample* (contoh kasus yang menjukkan solusi *greedy* yang diperoleh tidak optimal).

Contoh-contoh persoalan yang dapat diselesaikan dengan algoritma *greedy* adalah sebagai berikut:

1. Persoalan penukaran uang (*coin exchange problem*)
2. Persoalan memilih aktivitas (*activity selection problem*)
3. Minimisasi waktu di dalam sistem
4. Persoalan *knapsack* (*knapsack problem*)
5. Penjadwalan *Job* dengan tenggat waktu (*job scheduling with deadlines*)
6. Pohon merentang minimum (*minimum spanning tree*)
7. Lintasan terpendek (*shortest path*)
8. Kode Huffman (*Huffman code*)
9. Pecahan Mesir (*Egyptian fraction*)

B. *Othello*

Othello merupakan permainan papan strategi yang dimainkan oleh dua pemain. Othello biasa dimainkan pada papan berukuran 8x8 kotak berwarna hijau. Baris pada papan dinyatakan menggunakan angka (1 s.d. 8), sedangkan kolom pada papan dinyatakan menggunakan huruf (A s.d. H).

Permainan menggunakan sejumlah koin yang memiliki warna sisi berbeda, hitam dan putih. Jumlah koin maksimal yang dapat digunakan adalah 64 koin.

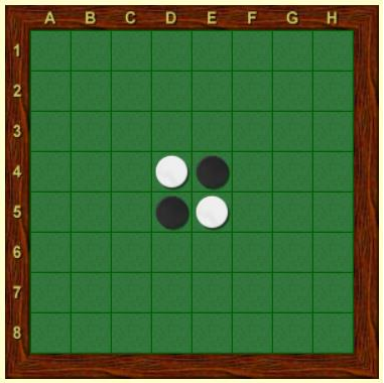
Secara umum, permainan dimulai dengan memberikan masing-masing pemain 30 koin. Kedua pemain menentukan warna yang ingin dimainkan. Warna hitam selalu mendapat giliran pertama.

Permainan berakhir ketika kedua pemain tidak memiliki Langkah atau Gerakan yang *legal*, biasanya ketika seluruh 64 kotak telah terisi. Pemenang permainan adalah pemain yang memiliki jumlah koin sesuai warna lebih banyak.

Berikut adalah gambaran alur permainan Othello berdasarkan sebuah *strategy guide* yang dibuat oleh Emmanuel Lazard dan French Othello Federation:

1. Starting Position

   Pada awal permainan, dua koin hitam diletakkan pada e4 dan d5, dua koin putih diletakkan pada d4 dan e5.



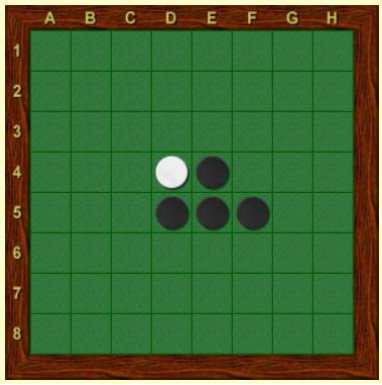
Gambar 2.2. Posisi Awal Permainan

Sumber: Strategy guide (radagast.se)

2. Making a Move

   Pada masing-masing giliran, pemain dapat meletakkan koin sesuai warnanya pada salah satu kotak kosong papan Othello. Dengan meletakkan koin, pemain harus menjepit satu atau beberapa koin lawan dengan koin yang sudah terletak pada papan sebelum giliran tersebut. Kemudian, ia akan membalikkan warna koin

lawan yang dijepit di antara koin yang baru diletakkan dan koin yang sudah berada di papan tersebut. Koin pada papan tidak dapat berpindah dari kotak koin tersebut diletakkan. Perlu diperhatikan Kembali bahwa syarat pemain dapat menaruh koin pada papan adalah ketika koin yang ia letakkan pada giliran tersebut menjepit satu atau koin lawan dengan koin yang sudah terdapat pada papan. Dengan kata lain, definisi langkah atau gerakan yang *legal* pada permainan Othello hanyalah langkah atau gerakan yang menyebabkan terjadinya pembalikan warna koin melalui penjepitan koin lawan.

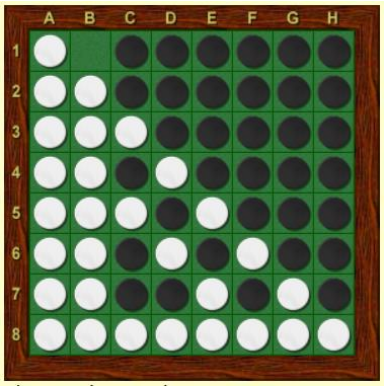


Gambar 2.3. Hitam Memainkan f5

Sumber: Strategy guide (radagast.se)

3. End of the Game

   Permainan berakhir ketika kedua pemain tidak memiliki Langkah atau Gerakan yang *legal*, tidak lagi terdapat koin yang dapat diletakkan pada papan sehingga dapat terjadi pembalikan warna koin akibat penjepitan koin. Biasanya, hal tersebut terjadi ketika seluruh 64 kotak papan Othello sudah terisi. Namun, dalam beberapa posisi tertentu permainan mungkin berakhir sebelum seluruh kotak terisi koin.



Gambar 2.4. Posisi Akhir Permainan

Sumber: Strategy guide (radagast.se)

III. PEMBAHASAN

Permainan Othello memiliki strategi yang cukup kompleks dalam menentukan langkah atau gerakan yang akan diambil pada giliran pemain. Berdasarkan deskripsi permainan yang sudah diberikan, secara singkat dapat disimpulkan bahwa persoalan optimasi pada permainan Othello termasuk persoalan maksimasi.

Pada permainan Othello, pemenang adalah pemain yang memiliki jumlah koin sesuai dengan warna terbanyak. Dengan kata lain, pada setiap giliran pemain ingin melakukan langkah atau gerakan yang mampu memberikan jumlah koin terbanyak.

Sebelum menerapkan alternatif solusi menggunakan algoritma *greedy*, berikut adalah dua dari strategi yang sering digunakan pada permainan Othello.

A. *Strategi Umum Othello*

1. *Stable disc* (*positional strategy*): selama permainan berlangsung, perlu diingat bahwa seorang pemain hanya dapat menaruh koin jika ia menjepit satu atau lebih koin lawan. Dengan begitu, prioritas utama pemain adalah mendapatkan posisi kotak yang aman dari jepitan koin lawan. Berdasarkan deskripsi tersebut, posisi-posisi pojok (*corner*) dan pinggir (*edge*) papan merupakan posisi yang penting untuk diambil terlebih dahulu.
2. *Tempo* (*waiting moves*): jika pemain hanya memiliki langkah atau gerakan yang terbatas dan tidak mengarah pada pojok atau pinggir papan, maka pemain perlu menentukan langkah yang tidak memberikan kesempatan musuh untuk menguasai pojok dan pinggir. Dengan kata lain, hindari meletakkan koin pada daerah di sekitar pojok dan pinggir papan, kecuali pada pojok atau pinggir papan itu sendiri.

Berdasarkan dua strategi tersebut, prioritas utama pemain adalah memiliki posisi kotak yang stabil, yaitu kotak yang sulit untuk dijepit oleh lawan sehingga koin yang diletakkan tidak dapat dibalik menjadi warna lawan. Posisi tersebut adalah pojok dan pinggir papan.

Namun, pada beberapa kondisi, mungkin pemain tidak dapat menaruh koinnya pada pojok atau pinggir papan. Diperlukan alternatif solusi yang mampu menghampiri solusi pertama. Berdasarkan tujuan permainan, akan diambil alternatif solusi *greedy* terbaik adalah langkah atau gerakan yang menghasilkan koin balikan terbanyak.

Dengan demikian, terdapat dua solusi *greedy* yang akan dibahas, yaitu:

1. *Greedy by Corners and Edges*
2. *Greedy by Coin Count*

B. *Greedy by Corners and Edges*

Sebelum membuat skema umum implementasi algoritma *greedy*, akan dilakukan pemetaan terhadap persoalan yang dihadapi.

1. Himpunan kandidiat, *C*: langkah-langkah yang menghasilkan jumlah koin yang diapit.

2. Himpunan solusi, *S*: langkah-langkah dari himpunan kandidat yang memberikan pemain kotak pada posisi pojok atau pinggir papan
3. Fungsi seleksi: pilih langkah yang memberikan posisi pojok atau pinggir papan
4. Fungsi kelayakan: semua langkah adalah layak
5. Fungsi obyektif: memaksimumkan stablitas koin pemain

Dengan pemetaan di atas, berikut adalah skema umum dari pendekatan *Greedy by Corners and Edges.*

```
function greedyByCnE(C: himpunan_kandidat) -> himpunan_solusi
{ menghasilkan langkah-langkah terbaik menggunakan algoritma
greedy terhadap pojok dan pinggir papan }
Deklarasi
  c: kandidat
  S: himpunan_solusi

Algoritma:
  S <- {}  { inisialisasi S dengan kosong }
  for c in C do
    if ((c[x] == 1) || (c[x] == 8)) then
      if c not in S then
        S <- S + c
    if ((c[y] == a) || (c[y] == h)) then
      if c not in S then
        S <- S + c
  if (S != {})
    return S
  else
    write('Tidak terdapat langkah yang memenuhi greedy by
corners and edge')
```

Gambar 3.1. Skema Umum Greedy by Corners and Edges

Sumber: Pribadi

Skema umum pendekatan *Greedy by Corners and Edges* di atas menerima sebuah himpunan kandidat kotak yang valid untuk diletakkan koin. Informasi kotak mencakup posisi baris dan kolom dari kotak tersebut. Fungsi akan mengecek apakah kotak tersebut terletak pada posisi baris 1, baris 8, kolom a, atau kolom h, yaitu posisi-posisi pojok dan pinggir pada papan. Jika iya, maka kandidat kotak tersebut akan ditambahkan pada himpunan solusi.

C. *Greedy by Coin Count*

Pada alternatif solusi ini, juga terlebih dahulu dilakukan pemetaan terhadap persoalan yang dihadapi.

1. Himpunan kandidiat, *C*: langkah-langkah yang menghasilkan jumlah koin yang diapit.
2. Himpunan solusi, *S*: langkah-langkah dari himpunan kandidat yang memberikan pemain jumlah koin Balika terbanyak
3. Fungsi seleksi: pilih langkah yang memberikan jumlah koin balikan terbanyak
4. Fungsi kelayakan: semua langkah adalah layak
5. Fungsi obyektif: memaksimumkan jumlah koin balikan

Dengan pemetaan di atas, berikut adalah skema umum dari pendekatan *Greedy by Corners and Edges.*

```
function greedyByCount(C: himpunan_kandidat)->himpunan_solusi
{ menghasilkan langkah-langkah terbaik menggunakan algoritma
greedy terhadap pojok dan pinggir papan }
Deklarasi
  c: kandidat
  S: himpunan_solusi

Algoritma:
  S <- {}  { inisialisasi S dengan kosong }
  for c in C do
    count <- c.getCoinCount()
    if S == {} then
      S <- {c}
    else
      if count > S[0].getCoinCount() then
        S <- {c}
    return S
```

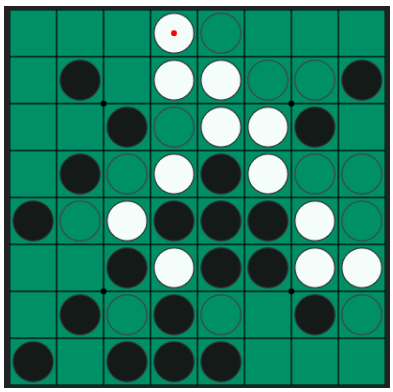
Gambar 3.2. Skema Umum Greedy by Coin Count

Sumber: Pribadi

Skema umum pendekatan *Greedy by Coin Count* di atas menerima sebuah himpunan kandidat kotak yang valid untuk diletakkan koin. Informasi kotak mencakup posisi baris dan kolom dari kotak tersebut. Diasumsikan sudah terdefinisi sebauh fungsi untuk mendapatkan jumlah koin yang dijepit oleh kotak valid tersebut. Untuk setiap kandidat, fungsi akan mengecek berapa banyak kotak lawan yang dijepit. Jika kandidat memiliki jumlah kotak lebih banyak disbanding yang terdapat pada himpunan solusi, maka kandidat tersebut akan menggantikan himpunan solusi.

D. *Analisis Aplikasi Strategi Greedy*

Berdasarkan skema umum yang sudah dibuat, penerapan algoritma *greedy* belum tentu memberikan hasil yang selalu optimal, terutama dalam kasus-kasus tertentu. Karena *greedy* hanya mengambil kemungkinan terbaik untuk langkah atau gerakan pada satu giliran tersebut, algoritma tidak memperhitungkan efek-efek yang mungkin muncul akibat langkah atau gerakan yang mungkin diambil berdasarkan himpunan solusi yang diberikan. Berikut adalah beberapa kasus yang mungkin terjadi ketika menggunakan strategi *greedy* pada permainan Othello:

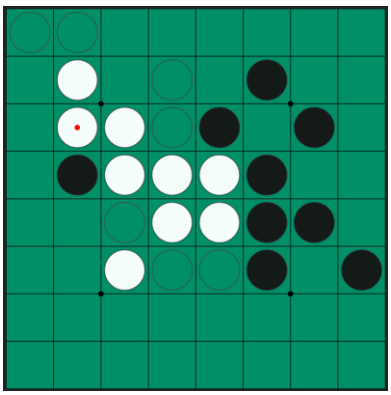


Gambar 3.3. Test Case 1

Sumber: Pribadi

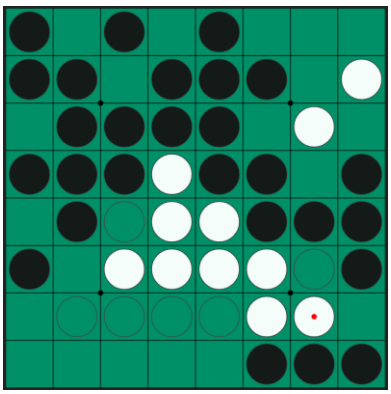
Pada gambar di atas, ketika algoritma *greedy* dijalankan, maka pemain (koin hitam) akan meletakkan koinnya pada salah satu kotak di pinggir papan, yaitu kotak yang

terdapat pada kolom h, yang merupakan himpunan solusi dari pendekatan *greedy by corners and edges*. Namun, hal tersebut belum tentu memberikan posisi yang dipastikan stabil karena sudah terdapat koin berwarna putih pada kolom h. Dengan kata lain, terdapat kemungkinan pada giliran lawan selanjutnya untuk justru membalikkan koin-koin hitam yang diletakkan pada kolom h.



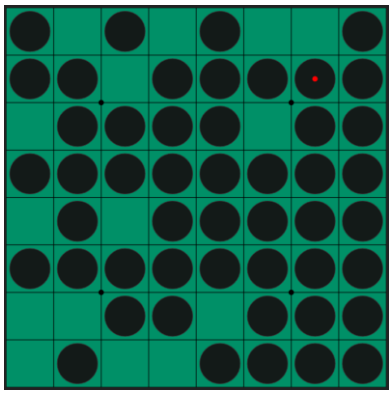
Gambar 3.4. Test Case 2
Sumber: Pribadi

Pada gambar kasus kedua, pendekatan *greedy by corners and edges* akan memberikan hasil yang dipastikan optimal karena salah satu kandidat pada himpunan solusi adalah posisi kotak a1, yaitu kotak pojok papan. Posisi pojok pada papan merupakan posisi yang dipastikan stabil karena tidak bisa dijepit oleh lawan, baik secara horizontal, vertikal, maupun diagonal. Dengan begitu, stabilitas kotak pemaain hitam dapat dipastikan optimal.



Gambar 3.5. Test Case 3
Sumber: Pribadi

Pada gambar kasus ketiga, pendekatan *greedy* yang digunakan adalah *greedy by coin count*. Hal tersebut karena tidak terdapat posisi kotak pada pojok atau pinggir papan sebagai kandidat dalam himpunan solusi *greedy by corners and edges*. Meskipun tidak dapat dilakukan *greedy by corners and edges*, terdapat alternatif solusi *greedy by coin count.* Terdapat 6 himpunan kandidat, yaitu { c5, g6, b7, c7, d7, e7 }. Berdasarkan keenam himpunan kandidat tersebut, kandidat dengan jumlah koin balikan terbanyak adalah kandidat posisi kotak d7, yang memliki koin balikan sebanyak 3. Sementara kandidat lainnya memiliki koin balikan 2 atau 1 saja. Dengan begitu, pemain akan meletakkan koin pada posisi kotak d7.



Gambar 3.6. Test Case 4
Sumber: Pribadi

Pada gambar kasus keempat, dapat dilihat bahwa penerapan algoritma *greedy by corners and edges* dan *greedy by coin count* yang efektif akan menghasilkan permainan Othello yang lebih mudah. Pada gambar di atas, terlihat permainan berakhir dengan kemenangan hitam tanpa perlu memenuhi seluruh 64 kotak papan. Hal tersebut dapat dicapai dengan penggunaan pendekatan algoritma *greedy* yang sudah dibahas di atas. Seperti yang sudah dijelaskan, prioritas pemain adalah mendapatkan posisi-posisi penting pada papan, yaitu pojok dan pinggir papan. Jika tidak terdapat posisi kotak pojok atau pinggir sebagai dalam himpunan kandidat, pilih kandidat posisi kotak yang memberikan jumlah koin balikan terbanyak. Dengan demikian, penerapan algoritma *greedy* dalam permainan Othello terbukti dapat membantu memenangkan permainan Othello dengan mudah. Namun, perlu diingat bahwa terkadang akan terdapat solusi sub-optimal sehingga tidak selalu memenangkan permainan.

IV. KESIMPULAN

Algoritma *greedy* merupakan salah satu teknik yang dapat digunakan untuk menentukan langkah atau gerakan terbaik pada satu giliran permainan Othello. Kedua implementasi algoritma *greedy* yang dibahas pada makalah ini, yaitu *greedy by coin count* dan *greedy by corners and edges* merupakan pengambilan keputusan maksimasi yang melebihi kelebihan dan kekurangannya masing-masing. Dengan memanfaatkan algoritma *greedy* pada permainan Othello, diharapkan pemain mendapat beberapa gambaran atau ide strategi efektif untuk dapat memenangkan Othello dengan lebih mudah.

PENUTUP

Puji dan syukur penulis panjatkan kepada Allah SWT. karena berkat rahmat dan karunia-Nya penulis dapat menyelesaikan makalah dengan tepat waktu. Penulis juga menyampaikan terima kasih kepada Ibu Dr. Nur Ulfa Maulidevi, S.T, M.Sc. selaku dosen mata kuliah Strategi Algoritma yang telah membimbing penulis selama perkuliahan strategi algoritmas.

VIDEO LINK AT YOUTUBE

https://youtu.be/Y7Ex_XV8SBk

REFERENSI

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf, diakses pada 21 Mei 2023 pukul 19.32

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf, diakses pada 21 Mei 2023 pukul 19.45

[3] https://www.chessprogramming.org/Othello, diakses pada 21 Mei 2023 pukul 19.59

[4] How to play Othello - YouTube, diakses pada 21 Mei 2023 pukul 20.13

[5] How to win at Othello: Corner & Edge Strategies - YouTube, diakses pada 21 Mei 2023 pukul 20.26

[6] How to win at Othello almost every time! - YouTube, diakses pada 21 Mei 2023 pukul 20.40

[7] http://radagast.se/othello/Help/strategy.html, diakses pada 21 Mei 2023 pukul 21.03

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, 22 Mei 2023

Ahmad Ghulam Ilham 13521118