Penerapan Algoritma *Greedy* untuk Permainan Halma

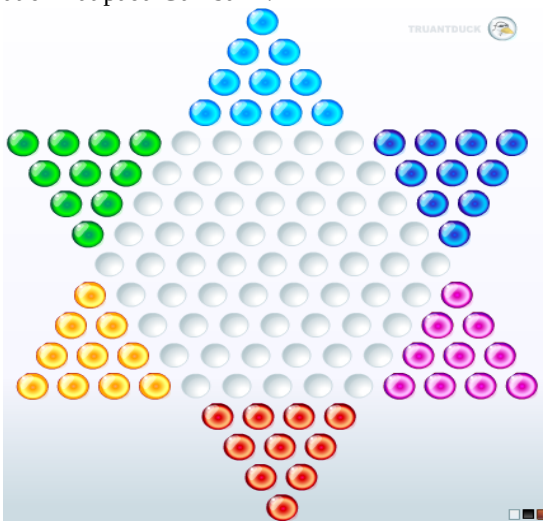
Vivi Lieyanda / 13509073[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13509073@std.stei.itb.ac.id

***Abstrak*—Halma adalah salah satu permainan yang bertujuan memindahkan sekelompok pion menuju daerah tujuan di seberangnya. Pemain dinyatakan menang apabila dapat memindahkan sekelompok pion ini terlebih dahulu. Permainan ini membutuhkan solusi optimal untuk mencari jumlah pergerakan paling sedikit untuk memindahkan pion menuju daerah tujuan. Salah satu algoritma yang banyak digunakan untuk masalah optimasi adalah algoritma *greedy*. Algoritma *greedy* memiliki prinsip untuk memilih langkah terbaik di tiap pergerakan untuk menemukan hasil yang optimal. Makalah ini akan membahas mengenai penerapan algoritma *greedy* dalam permainan halma.**

***Kata Kunci*—*Greedy*, Halma**

I. PENDAHULUAN

Halma merupakan salah satu jenis permainan yang menggunakan papan permainan sebagai salah sau komponennya (*board game*). Halma diciptakan pada tahun 1883 atau 1884 oleh seorang ahli bedah plastik Amerika di *Harvard Medical School* yang bernama Dr. George Howard Monks. Dr Thomas Hill, seorang ahli matematika, membantu mengembangkan permainan ini dan memberi nama "Halma". Halma dalam bahasa Yunani berarti "lompat". Seiring dengan berkembangnya teknologi, maka halma telah dibuat sebagai permainan komputer. Salah satu bentuk tampilan dari permainan ini dapat dilihat pada Gambar 1.



***Gambar 1** Halma untuk 6 pemain*

Halma dimainkan di dalam daerah berbentuk bintang berkaki enam. Permainan ini dapat dimainkan oleh 2 hingga 6 pemain. Tiap pemain diwakili oleh warna yang berbeda dengan pemain lainnya. Masing-masing pemain memiliki 10 atau 15 buah pion tergantung ukuran halma yang digunakan. Objektif dari permainan halma adalah memindahkan semua pion berwarna ini dari tempat asal ke tempat tujuan tang terletak di seberangnya. Pemain yang dinyatakan menang adalah permain pertama yang berhasil memindahkan semua pion ke daerah tujuan.

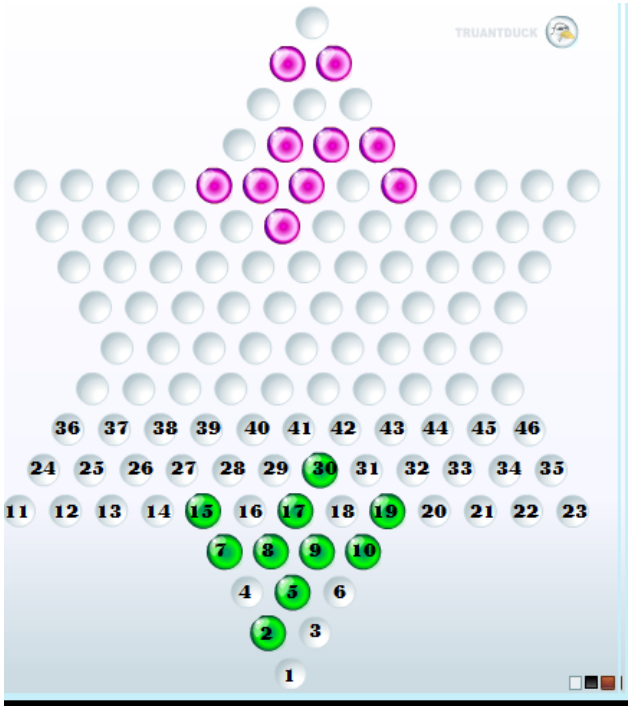
***Gambar 2** Halma dengan 15 pion untuk tiap pemain*

II. DASAR TEORI

A. Aturan Permainan Halma

Cara permainan halma adalah sebagai berikut:

- Mulai melangkahkan pion dengan arah yang diizinkan secara bergiliran
- Pion boleh hanya bergerak satu langkah ke kotak kosong ke arah kiri atas, kiri bawah, kanan atas, kanan bawah, samping kanan, samping kiri, atau melompati pion di depannya. Lompatan dapat terus dilakukan selama masih ada pion yang dapat dilompati.
- Langkahkan pion menuju ke arah segitiga seberang dan disusun secara tepat agar area segitiga terisi dengan penuh.
- Pemenang adalah yang paling cepat memindahkan semua pion ke segitiga seberang.



Gambar 3 *Contoh keadaan posisi pion*

Misalkan kita ingin menggerakkan pion pada posisi 17 dengan posisi pion-pion lain seperti pada gambar 3. Adapun posisi-posisi yang dapat dicapai oleh pion ini adalah sebagai berikut :

- Posisi 29 yang merupakan posisi pion setelah dipindahkan ke arah kiri atas.
- Posisi 16 yang merupakan posisi pion setelah dipindahkan ke arah samping kiri.
- Posisi 18 yang merupakan posisi pion setelah dipindahkan ke arah samping kanan.
- Posisi 4 yang merupakan posisi pion setelah melompat ke arah kiri bawah melewati pion posisi 8.
- Posisi 6 yang merupakan posisi pion setelah melompat ke arah kanan bawah melewati pion posisi 9.
- Posisi 42 yang merupakan posisi pion setelah melompat ke arah kanan atas melewati pion posisi 30.
- Posisi 1 yang merupakan posisi pion setelah melompat ke posisi 4 kemudian melompat lagi ke posisi 1 melewati pion posisi 2 pada arah kiri bawah.

B. Algoritma Greedy

Algoritma *greedy* merupakan metode yang paling populer untuk memecahkan persoalan optimasi. Optimasi adalah persoalan yang tidak hanya mencari sekedar solusi namun juga mencari solusi terbaik. Algoritma *greedy* menggunakan prinsip "*take what you can get now*!". Algoritma *greedy* adalah algoritma yang membentuk solusi langkah per langkah. Pada setiap langkah, terdapat banyak pilihan yang perlu dieksplorasi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya. Pada setiap langkah kita membuat pilihan optimum lokal dengan harapan bahwa langkah sisanya mengarah ke solusi optimum global.

Elemen-elemen algoritma *greedy*:

1. Himpunan kandidat, *C*
   Himpunan kandidat adalah himpunan yang berisi elemen-elemen pembentuk solusi.
2. Himpunan solusi, *S*
   Himpunan solusi adalah himpunan yang berisi kandidat-kandidat yang terpilih sebagai solusi persoalan.
3. Fungsi seleksi (*selection function*)
   Fungsi seleksi adalah fungsi yang memilih kandidat-kandidat yang paling memungkinkan mencapai solusi optimal. Kandidat yang telah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya.
4. Fungsi kelayakan (*feasible*)
   Fungsi ini memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala yang ada. Kandidat yang layak dimasukkan ke dalam himpunan solusi, sedangkan kandidat yang tidak layak dibuang dan tidak pernah dipertimbangkan lagi.
5. Fungsi obyektif
   Fungsi ini memaksimumkan atau meminimumkan nilai solusi.

Dengan kata lain algoritma *greedy* melibatkan pencarian sebuah himpunan bagian dari himpunan kandidat, dimana yang dalam hal ini, himpunan solusi harus memenuhi beberapa kriteria yang ditentukan, yaitu menyatakan suatu solusi dan himpunan solusi dioptimisasi oleh fungsi obyektif. Adapun *pseudo-code* untuk algoritma *greedy* adalah sebagai berikut:

```
function greedy(input C:
himpunan_kandidat)→
himpunan_kandidat
{ Mengembalikan solusi dari
persoalan optimasi dengan
algoritma greedy
Masukan : himpunan kandidat C
Keluaran: himpunan solusi yang
bertipe himpunan_kandidat
}
Deklarasi
   x : kandidat
   S : himpunan_kandidat
Algoritma:
   S ← {} {inisialisasi S dengan
        kosong}
   while (not SOLUSI(S)) and (C
   !={} ) do
      x ← SELEKSI(C) { pilih
           sebuah kandidat dari C}
      C ← C - {x} { elemen
           himpunan kandidat
           berkurang satu }
      if LAYAK(S ∪{x}) then
         S ← S ∪ {x}
```

```
      endif
   endwhile
   {SOLUSI(S) or C = {} }

   if SOLUSI(S) then
      return S
   else
      write('tidak ada solusi')
endif
```

III. APLIKASI

Dari berbagai penjelasan di atas, terlihat bahwa algoritma *greedy* cocok digunakan untuk mencari solusi dalam permainan halma. Dalam penggunaan algoritma ini, kita gunakan beberapa pendekatan, yaitu *greedy* berdasarkan lompatan dan *greedy* berdasarkan pion belakang. Arah pergerakan dari tiap pion terdiri atas 6 arah, yaitu kiri atas, kanan atas, samping kanan, samping kiri, kiri bawah, dan kanan bawah.

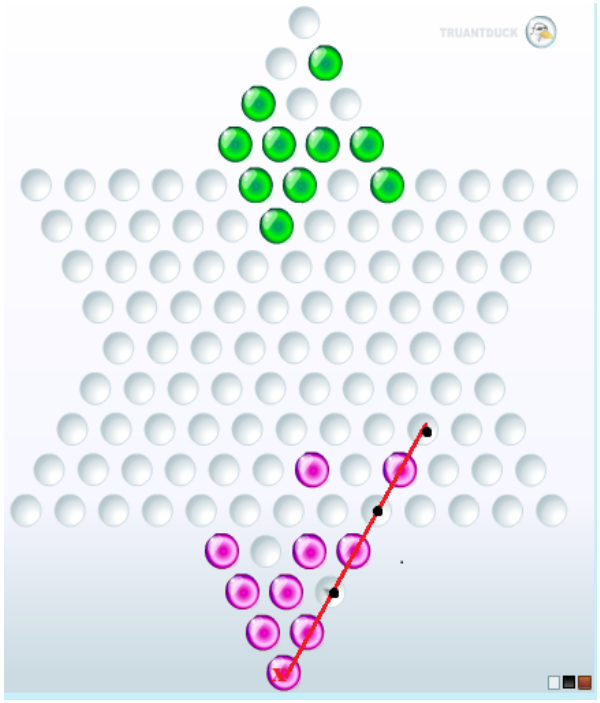
A. Greedy berdasarkan Lompatan

Algoritma ini mengutamakan pion yang dapat melompat sebanyak mungkin menuju arah segitiga di seberangnya. Di dalam algoritma ini, kita akan menggerakkan pion yang dapat melompat sebanyak mungkin. Dalam menggunakan algoritma ini, kita akan memprioritaskan beberapa arah pergerakan, yaitu

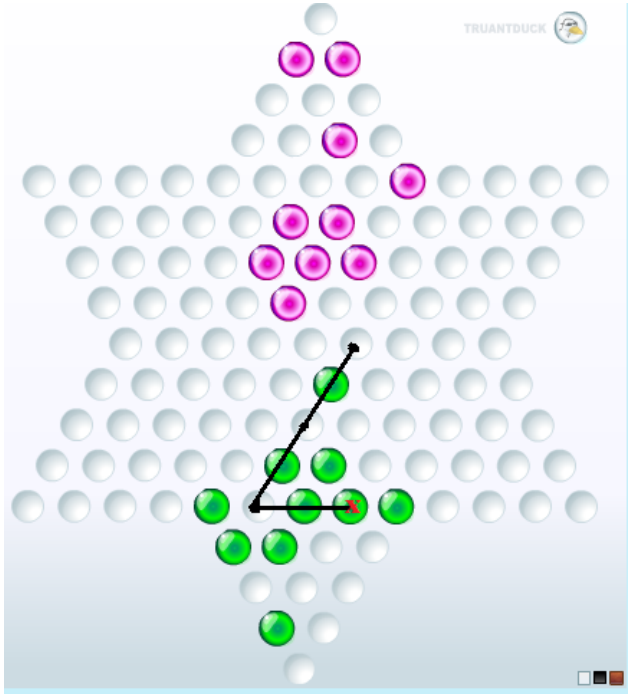
1. Lompat atas berkali-kali
   Yang dimaksud disini adalah pion dapat bergerak ke arah atas (bergerak ke daerah tujuan) dengan melompat berkali-kali. Lompatan ini dapat dilakukan ke arah kanan ataupun kiri.
2. Lompat samping kemudian lompat atas berkali-kali
   Pergerakan ini dilakukan dengan mengecek ke arah samping kanan atau kiri. Apabila pion dapat bergerak ke samping kanan atau kiri, maka akan diperiksa apakah pion setelah lompat dapat bergerak ke arah atas berkali-kali.
3. Lompat bawah kemudian lompat atas berkali-kali
   Pergerakan ini akan mengecek ke arah kanan bawah atau kiri bawah. Setelah melakukan pergerakan ke bawah, maka akan diperiksa apakah pion dapat bergerak ke atas lebih dari satu kali. Hal ini dilakukan karena apabila pion hanya dapat bergerak ke atas sebanyak satu kali, maka pergerakan ini akan menempati posisi di level yang sama dengan sebelum pergerakan.
4. Lompat atas sekali
   Pergerakan ini diperiksa dengan melihat apakah pion dapat melompat ke arah kanan atas atau kiri atas sejumlah satu lompatan tergantung prioritas kiri atau kanan yang duluan.
5. Lompat samping kemudian lompat atas sekali
   Pergerakan ini akan mengecek ke arah samping kanan atau kiri. Apabila setelah pergerakan ke kanan, pion dapat melompat ke arah atas sebanyak sekali, maka pergerakan ini akan dilakukan. Jika tidak, algoritma akan mengecek ke arah kiri. Jika setelah pindah ke samping kiri, pion tidak dapat melompat ke atas, maka akan digunakan prioritas pergerakan berikutnya.
6. Apabila dari semua pergerakan di atas tidak ada yang dipenuhi, maka pion akan berpindah ke arah atas.

Dalam algoritma greedy berdasarkan lompatan, adapun elemen-elemen yang digunakan adalah sebagai berikut:

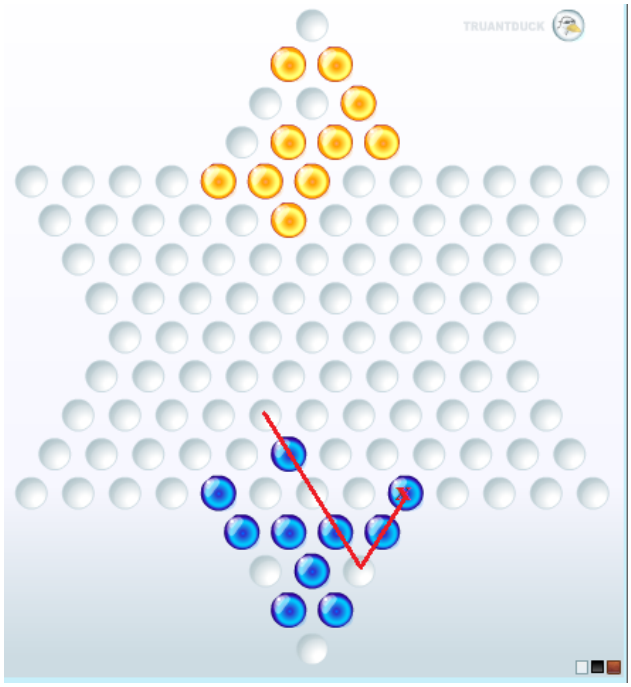
- Himpunan kandidat : himpunan semua pion-pion dari pihak CPU yang berada di atas bidak. Pion ini berjumlah 10 atau 15 sesuai dengan jenis halma yang digunakan.
- Himpunan solusi : pion yang memenuhi prioritas-prioritas dalam algoritma ini. Pion yang memenuhi prioritas disini maksudnya adalah pion yang dapat melompat berkali-kali ke arah atas, pion yang dapat melompat ke samping dan melompat berkali-kali ke arah atas, dan lain sebagainya.
- Fungsi seleksi : pilihlah pion yang memenuhi prioritas tertinggi.
- Fungsi layak : memeriksa apakah pion-pion yang dipilih dapat digerakkan ke posisi tujuan.
- Fungsi objektif : pion yang dijalankan adalah pion yang berada pada prioritas tertinggi.



Gambar 4 *Perpindahan pion dengan melompat berkali-kali ke arah kanan atas*



Gambar 5 *Pergerakan pion ke arah samping kiri kemudian lompat berkali-kali ke arah kanan atas*



Gambar 6 *Pergerakan pion ke arah kiri bawah kemudian melompat ke kiri atas berkali-kali*

B. Greedy berdasarkan Pion Belakang

Algoritma ini mengutamakan pergerakan pion-pion yang berada di baris belakang untuk maju terlebih dahulu. Hal ini dikarenakan apabila kita selalu medahulukan pion-pion yang berada pada barisan depan, maka pion-pion yang tertinggal di baris belakang akan kesusahan atau lambat pergerakannya. Pergerakan yang lambat diakibatkan sedikitnya jumlah pion yang dapat dilompati, sehingga pion hanya dapat bergerak maju satu langkah di tiap pergerakannya. Jadi, algoritma ini dijalankan dengan memperhatikan posisi setiap pion. Nilai dari tiap posisi pion dapat diambil dari koordinat tiap pion pada *board* yang digunakan.

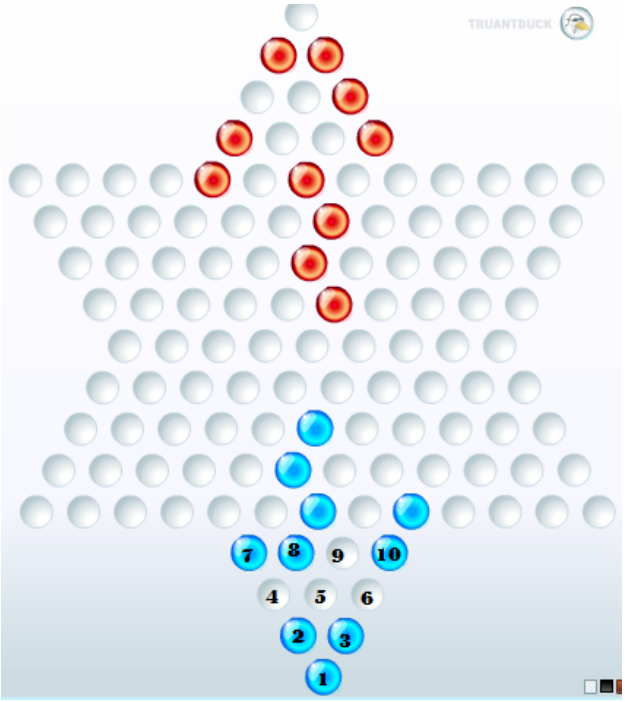
Dalam algoritma ini, pion tidak menutup kemungkinan melompat apabila pion dapat melompat sejauh mungkin untuk menghemat jumlah pergerakan. Namun lompatan disini hanyalah lompatan ke arah atas. Berikut adalah algoritma dalam memilih urutan pion.

```
Function PionBelakang (input C : himpunan_pion) -> himpunan_pion
{mengembalikan solusi urutan pion yang harus dijalankan dalam permainan halma dengan meneraplan greedy berdasarkan pion belakang
Masukan    : himpunan pion C
Keluaran   : himpunan solusi yang bertipe himpunan pion}

Deklarasi
S : himpunan_pion
X : kandidat

Algoritma
S ← {}
while (Jumlah pion dalam S) <> (jumlah pion dalam C)  and (C <> {}) do
X ← objek di dalam C yang memiliki posisi paling belakang
C <- C - {X}
   if CanMove(Pion) then
         S ← S U {X}
   endif
endwhile
{ C = {} }
return S
```

Pada algoritma *greedy* berdasarkan pion belakang di atas, elemen-elemen yang digunakan adalah sebagai berikut :

- Himpunan kandidat : himpunan semua pion-pion dari pihak CPU yang berada di bidak. Pion ini berjumlah 10 atau 15 untuk tiap jenis halma.
- Himpunan solusi : urutan pion-pion yang dijalankan dimana akan menghasilkan jumlah langkah paling sedikit sehingga dapat memindahkan pion dengan cepat daripada musuh. Pada algoritma ini, jumlah langkah dipercepat dengan mengutamakan pergerakan pion bagian belakang agar dapat melompat.
- Fungsi Seleksi : pilihlah pion pada bidak dimana pion berada pada posisi paling belakang.
- Fungsi Layak : memeriksa apakah pion yang dipilih dapat berpindah posisi ke arah depan mendekati daerah tujuan.

- Fungsi Objektif : pion yang dijalankan adalah pion yang berada pada posisi terakhir dan apabila dapat melompat, maka diizinkan untuk melompat.



Gambar 7 *Contoh keadaan posisi pion-pion dalam board*

Dari gambar berikut, apabila kita menerapkan algoritma *greedy* berdasarkan pion belakang, maka algoritma akan menjalankan pion yang berada pada posisi 1. Pion pada posisi 1 memenuhi fungsi kelayakan dimana ia dapat berpindah ke posisi bernomor 4 ataupun 6. Setelah itu, kita akan menggerakan pion pada posisi 2 ataupun 3 dengan memeriksa apakah pion tersebut memenuhi fungsi kelayakan.

IV. HASIL DAN PEMBAHASAN

Setelah melakukan ujicoba terhadap algoritma yang dibuat, maka algoritma *greedy* berdasarkan lompatan lebih baik daripada algoritma *greedy* berdasarkan pion belakang. Hal ini dikarenakan pergerakan dengan lompatan jauh menghemat jumlah pergerakan yang akan dilakukan berikutnya. Dibandingkan dengan algoritma *greedy* berdasarkan pion belakang, maka pion-pion memang akan melompat juga, namun tidak memperhitungkan pion dapat bergerak ke samping atau ke bawah terlebih dahulu untuk dapat melompat ke arah depan.

Solusi yang terbaik adalah dengan mencari lintasan terpanjang tiap pergerakan pion. Lintasan terpanjang yang dimaksud adalah lintasan yang terbentuk dari pion yang dapat melompat lebih dari satu kali sehingga mendekati daerah tujuan. Semakin banyak lompatan yang dapat dilakukan menuju daerah tujuan, maka akan menghemat jumlah pergerakan yang akan dilakukan.

Selain itu, diusahakan untuk tidak membiarkan ada pion yang tertinggal di belakang dalam jumlah yang sedikit. Hal ini akan menyebabkan pion-pion tersebut sangat lambat untuk bergerak. Hal ini dikarenakan hanya sedikit pion yang dapat digunakan untuk melompat. Solusi yang baik adalah dengan mengatur susunan pion agar dapat digunakan dilakukan loncatan yang terus menerus. Namun hal ini bisa saja digagalkan oleh pion lawan yang melompat ke tengah dari dua buah pion yang telah kita buat.

Algoritma *greedy* merupakan salah satu algoritma yang dapat diimplementasikan di sebagain besar masalah dalam kehidupan. Meskipun algoritma *greedy* belum dapat memberikan solusi yang optimal dalam permainan halma ini, algoritma ini dapat memberikan solusi yang lumayan optimal.

V. SIMPULAN DAN SARAN

Dari pembahasan di atas, dapat disimpulkan :

- Algoritma *greedy* digunakan untuk mencari solusi optimum lokal dengan harapan akan memperoleh solusi optimum global
- Penggunaan algoritma *greedy* untuk permainan halma tidaklah memberikan hasil yang cukup optimal.
- Algoritma *greedy* berdasarkan lompatan memberikan hasil yang lebih optimal dibandingkan algoritma *greedy* berdasarkan pion belakang.

Adapun saran untuk pengembangan selanjutnya adalah sebagai berikut :

- Dengan adanya kelebihan dan kekurangan dari masing-masing algoritma *greedy* berdasarkan beberapa pendekatan, dalam hal ini *greedy* berdasarkan lompatan dan *greedy* berdasarkan pion belakang. Kedua algoritma tadi mungkin dapat dikombinasikan untuk memberikan solusi yang optimal.

DAFTAR PUSTAKA

[1] Rinaldi Munir, "Diktat Kuliah IF3051 Strategi Algoritma", Institut Teknologi Bandung, 2009, hal 26–68.

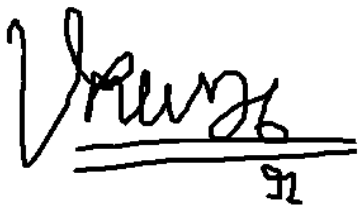
[2] http://www.online-games-zone.com/pages/classic/halma.php, 7 Desember 2011 20:05.

[3] http://upload.wikimedia.org/wikipedia/commons/thumb/f/fe/Halma-Spielfeld.jpg/769px-Halma-Spielfeld.jpg 7 Desember 2011 20:10

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, 8 Desember 2011



Vivi Lieyanda
13509073