IMPLEMENTASI ALGORITMA *GREEDY* PADA PERMAINAN OTHELLO

Nur Fajriah Rachmah – NIM 13506091

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jalan Ganesha nomor 10
e-mail: if16091@students.if.itb.ac.id

ABSTRAK

Algoritma *Greedy* merupakan metode yang paling populer untuk memecahkan masalah optimasi. Secara harfiah, *greedy* berarti tamak. Seorang yang tamak akan berusaha mengambil sebanyak mungkin kesempatan tanpa memikirkan konsekuensi ke depan. Prinsip *greedy* ialah : "*take what you can get now*".

Permainan othello dimainkan pada arena papan kotak-kotak persegi dengan koin hitam dan putih di atas arena. Pada awal permainan diletakkan dua koin hitam dan dua koin putih pada pusat arena. Koin warna hitam harus melewati koin warna putih agar koin putih dapat diubah menjadi koin hitam, dan sebaliknya. Permainan akan berakhir jika semua kotak arena sudah terisi koin, atau seluruh koin yang ada di atas arena berwarna sama. Pemenang adalah pemain yang memiliki jumlah koin lebih banyak di atas arena. Permainan ini di luar negeri lebih dikenal dengan nama *Reversi.* Saat ini Othello tidak hanya dimainkan secara tradisional, namun sudah banyak dibuat dalam bentuk animasi pada komputer.

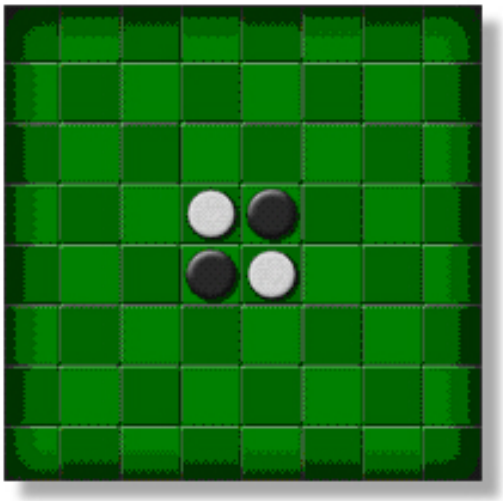
Dalam makalah ini, penulis bermaksud untuk mengimplementasikan algoritma *greedy* pada permainan othello untuk membuktikan solusi optimum yang dapat dihasilkan.

Kata kunci: *greedy*, othello, optimasi.

1. PENDAHULUAN

1.1 Peraturan Permainan

Permainan Othello dimainkan oleh dua orang pemain. Permainan ini dimainkan di atas papan arena permainan persegi yang terdiri dari kotak-kotak kecil, biasanya berukuran 8x8 kotak. Peralatan lain yang dibutuhkan ialah koin hitam dan koin putih masing-masing sebanyak 64 buah. Pada awal permainan akan diletakkan dua koin hitam dan dua koin putih pada pusat arena.



Gambar 1. Tampilan Othello pada *online game*

Misalkan pemain pertama menggunakan koin hitam dan pemain kedua menggunakan koin putih. Koin hitam harus dapat 'melompati' koin putih agar koin putih dapat menjadi hitam. Setelah koin hitam berpindah posisi dan koin-koin yang dilalui diubah warnanya menjadi hitam, koin putih yang mendapat giliran untuk berpindah. Koin putih harus 'melompati' koin hitam agar koin tersebut dapat diganti menjadi koin putih.

Kedua pemain terus menerus secara bergantian memindahkan letak koin masing-masing. Hanya 1 koin yang dapat dipindahkan setiap kali giliran jalan. Gerakan koin dapat lurus maupun diagonal asalkan setiap pergerakan membentuk satu garis lurus.

Permainan akan berakhir jika seluruh kotak-kotak kecil pada arena permainan terisi koin atau seluruh koin di atas arena berwarna sama. Pemenang ialah pemain yang memiliki jumlah koin paling banyak di atas arena permainan.

1.2 Skema Umum Algoritma *Greedy*

Algoritma *greedy* merupakan metode yang paling populer untuk memecahkan masalah optimasi. Algoritma *greedy* membentuk solusi langkah per langkah. Pendekatan yang digunakan di dalam algoritma *greedy* adalah membuat pilihan yang tampak memberi perolehan terbaik, yaitu dengan membuat pilihan optimum lokal pada setiap langkah dengan harapan akan mengarah ke solusi optimum global.

Prinsip algoritma *greedy* pada setiap langkah ialah mengambil pilihan terbaik yang dapat diperoleh saat itu tanpa memperhatikan konsekuensi ke depan, dan berharap bahwa dengan memilih optimum lokal pada setiap langkah akan menghasilkan optimum global pada akhir proses.

Persoalan optimasi algoritma *greedy* disusun oleh elemen-elemen berikut :

1. Himpunan kandidat, yang berisi elemen-elemen pembentuk solusi.
2. Himpuan solusi, berisi kandidat-kandidat yang terpilih sebagai solusi persoalan.
3. Fungsi seleksi, dinyatakan dengan predikat SELEKSI memilih kandidat yang paling memungkinkan mencapai solusi optimal pada setiap langkah.
4. Fungsi kelayakan, dinyatakan dengan predikat LAYAK, memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak dengan tidak melanggar *constraints* yang ada.
5. Fungsi objektif, yang memaksimumkan atau meminimumkan nilai solusi.

Prinsip algoritma *greedy* pada setiap langkah ialah mengambil pilihan terbaik yang dapat diperoleh saat itu tanpa memperhatikan konsekuensi ke depan, dan berharap bahwa dengan memilih optimum lokal pada setiap langkah akan menghasilkan optimum global pada akhir proses.

Skema umum algoritma *greedy* dapat digambarkan melalui *pseudo-code* berikut,

```
function greedy(input C:h_kandidat)→h_kandidat
{
   Mengembalikan solusi dari persoalan optimasi
   dengan algoritma greedy. Masukan : himpunan
   kandidat C. Keluaran : himpunan solusi yang
   bertipe himpunan_kandidat.
}


Deklarasi
   x : kandidat
   S : h_kandidat

   function SELEKSI(C : h_kandidat) → kandidat
   {me-return sebuah kandidat yang dipilih dari
   C berdasarkan kriteria tertentu}

   function SOLUSI(S : h_kandidat) → boolean
   {true jika S adalah solusi dari persoalan;
   false jika S belum menjadi solusi}

   function LAYAK(S : h_kandidat) → boolean
   {true jika S merupakan solusi yang tidak
   melanggar kendala ; false jika S melanggar
   kendala}


Algoritma
   S ← {}      {Inisialisasi S dengan kosong}
   while (not SOLUSI(S)) and (C ≠ {}) do
      x ← SELEKSI(C) {pilih 1 kandidat dari C
      C ← C - {x}
      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
```

2. RUANG LINGKUP PERSOALAN

Ruang lingkup pembahasan pada makalah ini ialah beberapa strategi yang dapat memecahkan persoalan optimasi jumlah koin pada akhir permainan.

Terdapat dua strategi *greedy* heuristik yang akan diimplementasikan untuk menghasilkan sebanyak mungkin koin sehingga dapat memenangkan permainan, yaitu :

1. *Greedy by* jumlah koin
   Pada setiap langkah, koin pemain akan menuju koordinat yang menghasilkan sebanyak mungkin koin lawan. Strategi ini berusaha memaksimumkan jumlah koin pada akhir permainan dengan menghasilkan koin sebanyak-banyaknya pada setiap langkah.
2. *Greedy by* jarak ke tepi
   Pada setiap langkah, koin pemain akan menuju ke koordinat yang semakin dekat dengan tepi arena permainan. Strategi ini berusaha memaksimumkan jumlah koin pada akhir permainan dengan menguasai daerah tepi yang sulit untuk dilangkahi koin lawan. Bahkan untuk bagian pojok arena, sama sekali tidak dapat dilangkahi oleh lawan.

3. *GREEDY BY* JUMLAH KOIN PADA PERMAINAN OTHELLO

3.1 Representasi Masalah

Untuk merepresentasikan masalah digunakan sebuah larik (*array*) dua dimensi yang diberi nama isi[i][j]. Elemen *array* isi akan bernilai 1 jika pada koordinat tersebut terdapat koin berwarna hitam, dan akan bernilai 0 jika pada koordinat tersebut terdapat koin berwarna putih. Indeks i dan j masing-masing antara 1-8 yang menunjukkan koordinat terhadap sumbu X dan sumbu Y.

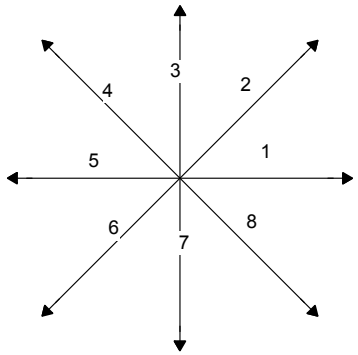
Nilai himpunan kandidat disimpan di dalam sebuah *array* satu dimensi yang diberi nama jumlah[k]. *Range* k berkisar antara 1-8. Angka 1-8 menunjukkan arah dari koordinat asal ke koordinat tujuan, 1 menunjukkan arah kanan, kemudian urut berlawanan arah dengan jarum jam, hingga angka 8 yang menunjukkan arah kanan bawah.

Selain *array*, terdapat pula dua buah variabel bertipe integer yang diberi nama JHitam dan JPutih. JHitam akan berisi jumlah koin hitam di atas arena permainan, sedangkan JPutih berisi jumlah koin putih di atas arena permainan.

3.2 Pemilihan Langkah

Sebelum memulai setiap langkah, pemain akan menghitung nilai yang akan didapatkan pada setiap kemungkinan langkah. Kemudian pemain akan memilih langkah yang menghasilkan nilai terbesar dengan harapan akan menghasilkan solusi yang optimum.

Jika terdapat lebih dari satu jumlah[k] yang bernilai paling besar, prioritas pemilihan langkah ialah ke arah kanan, kanan atas, atas, dan terus memutar berlawanan arah dengan jarum jam, sehingga prioritas terakhir ialah ke arah kanan bawah.



Gambar 2. Prioritas pergerakan koin jika jumlah[k] sama

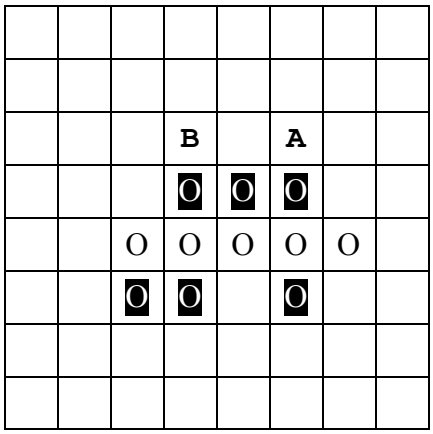
Pada suatu permainan Othello sangat mungkin pada satu giliran jalan terdapat banyak koin yang mungkin untuk dijalankan. Jika hal ini terjadi, prioritas pemilihan langkah ialah arah 1 dari seluruh koin, dengan prioritas letak koin di kanan-atas. Kemudian dilanjutkan dengan urutan koin yang sama dengan arah 2, dan seterusnya.

3.3 Pemecahan Masalah

Persoalan optimasi dengan *greedy by* jumlah koin disusun oleh elemen-elemen berikut :

1. Himpunan kandidat, seluruh jumlah[k] yang memungkinkan dimana k berada pada *range* 1-8. Jika pada arah tertentu tidak terdapat koin lawan untuk dilangkahi, tidak ada pergerakan yang dapat dilakukan sehingga arah tersebut tidak masuk ke dalam himpunan kandidat.
2. Himpunan solusi, langkah-langkah pada himpunan kandidat yang memiliki jumlah[k] terbesar.
3. Fungsi seleksi, pilihlah langkah yang menghasilkan jumlah[k] terbesar. Jika terdapat lebih dari satu jumlah[k] memiliki nilai paling besar, pilih langkah sesuai urutan prioritas.
4. Fungsi kelayakan, periksa apakah k masih dalam range yang telah ditetapkan.
5. Fungsi objektif, langkah yang dipilih menghasilkan jumlah[k] maksimum.

Pemilihan jumlah[k] terbesar merupakan pemilihan optimum lokal yang diharapkan akan menjadi optimum global. Pemain yang menggunakan koin hitam akan memenangkan permainan jika JHitam > JPutih, dan sebaliknya jika JPutih lebih besar nilainya, pemain yang menggunakan koin putih yang akan memenangkan permainan.



Gambar 3. Contoh ilustrasi permainan Othello dengan *greedy by* jumlah koin, A menunjukkan langkah selanjutnya

Huruf A pada Gambar 3 menunjukkan langkah selanjutnya yang dipilih jika bermain menggunakan *greedy by* jumlah koin. Koordinat yang ditandai dengan huruf A dan huruf B sama-sama menghasilkan dua koin putih, namun koordinat B memiliki prioritas lebih rendah sehingga koordinat A yang dipilih sebagai langkah selanjutnya.

4. *GREEDY BY* JARAK KE TEPI PADA PERMAINAN OTHELLO

4.1 Representasi Masalah

Dalam permainan Othello, koin yang terletak di tepi arena permainan lebih sulit untuk dilangkahi karena hanya mungkin dilangkahi dari 2 arah saja. Bahkan koin yang terletak di pojok arena tidak dapat dilangkahi dari arah manapun. Karena itulah para pemain seringkali memiliki prioritas untuk melangkah ke tepi.

Jarak ke tepi yang menjadi fokus batasan pada algoritma ini ialah jarak ke tepi kiri dan tepi kanan. Jarak ke tepi atas dan bawah tidak ikut diperhitungkan.

Serupa dengan *greedy by* nilai koin, pada *greedy by* jarak ke tepi juga menggunakan sebuah larik (*array*) dua dimensi yang diberi nama isi[i][j] untuk merepresentasikan masalah. Elemen *array* isi akan bernilai 1 jika pada koordinat tersebut terdapat koin berwarna hitam, dan akan bernilai 0 jika pada koordinat tersebut terdapat koin berwarna putih. *Range* indeks i dan j berada diantara 1-8 yang menunjukkan koordinat terhadap sumbu X dan sumbu Y.

Jarak ke kedua tepi disimpan dalam dua buah array satu dimensi. Jarak ke kanan disimpan dalam array kanan[p] dan jarak ke kiri dalam array kiri[p]. p menunjukkan prioritas arah seperti yang ditunjukkan Gambar 2. Terdapat pula dua variabel bertipe integer yang diberi nama JHitam dan JPutih untuk menyimpan jumlah koin.

Misalkan, untuk pergerakan ke titik (3,5),

```
kanan[p]  =  8 - 3
kiri [p]  =  3 - 1
```

Konstanta 8 merupakan koordinat paling kanan, sedangkan konstanta 1 merupakan koordinat paling kiri.

4.2 Pemilihan Langkah

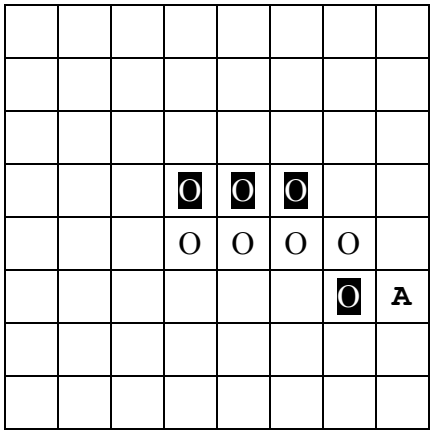
Sebelum memilih langkah, pemain akan menghitung jarak ke kanan dan jarak ke kiri pada setiap kemungkinan langkah. Pemain akan mengambil jarak terkecil yang didapat untuk mencapai tepi arena sesegera mungkin. Kemungkinan dan prioritas langkah dibagi menjadi delapan arah seperti pada Gambar 2. Jika jarak ke kanan dan ke kiri sama, akan diprioritaskan langkah ke arah kanan.

4.3 Pemecahan Masalah

Persoalan optimasi dengan *greedy by* jarak ke tepi disusun oleh elemen-elemen berikut :

1. Himpunan kandidat, seluruh kanan[p] dan kiri[p] yang mungkin dimana p menunjukkan arah langkah seperti ditunjukkan Gambar 2. Jika pada arah tertentu tidak terdapat koin lawan untuk dilangkahi, tidak ada pergerakan yang dapat dilakukan sehingga arah tersebut tidak masuk ke dalam himpunan kandidat.
2. Himpunan solusi, kanan[p] dan kiri[p] yang memiliki nilai paling minimum, yang berarti jarak paling dekat dengan tepi arena.
3. Fungsi seleksi, pilihlah langkah yang menghasilkan kanan[p] atau kiri[p] terkecil. Jika terdapat lebih dari 1 nilai terkecil, pilih berdasarkan urutan prioritas.
4. Fungsi kelayakan, periksa apakan p dan q masih termasuk dalam range yang telah ditetapkan.
5. Fungsi objektif, langkah yang dipilih menghasilkan jarak minimum dengan tepi arena.

Pemilihan kanan[p] atau kiri[p] terkecil merupakan pemilihan optimum lokal yang diharapkan akan menjadi optimum global. Pemain yang menggunakan koin hitam akan memenangkan permainan jika JHitam > JPutih, dan sebaliknya jika JPutih lebih besar nilainya, pemain yang menggunakan koin putih yang akan memenangkan permainan.



Gambar 4. Contoh ilustrasi permainan Othello dengan *greedy by* jarak ke tepi, A menunjukkan langkah selanjutnya

Huruf A pada Gambar 4 menunjukkan langkah selanjutnya yang akan dipilih jika bermain menggunakan *greedy by* jarak ke tepi. Semakin dekat jarak ke tepi, semakin besar prioritas koordinat tersebut untuk dipilih.

5. KESIMPULAN

Kesimpulan yang dapat diambil dari analisis permainan Othello menggunakan algoritma *greedy* ialah :

1. Algoritma *greedy* dapat memecahkan masalah optimasi, namun tidak selalu menghasilkan solusi yang optimum.
2. Pada kenyataannya, umumnya pemain mengkombinasikan kedua algoritma di atas untuk

mendapatkan hasil yang optimum. Pada awal permainan, pemain diharapkan lebih fokus untuk mengincar daerah tepi arena agar dapat menyusun 'benteng pertahanan'. Namun sebaliknya, saat telah menguasai daerah tepi pemain diharapkan dapat melompati koin sebanyak mungkin agar dapat memenangkan permainan.

3. Jika algoritma *greedy* tidak dapat menghasilkan solusi yang optimum, metode *exhaustive search* pasti dapat memberikan solusi yang optimum. Namun waktu yang dibutuhkan akan lebih lama karena *exhaustive search* mempertimbangkan seluruh alternatif solusi.

REFERENSI

[1] Munir, Rinaldi, "Diktat Kuliah IF2251 Strategi Algoritmik", Program Studi Teknik Informatika ITB, 2006.

[2] Vijayan, Smitha. *http://www/codeproject.com/KB/game/ReversiAIProject.aspx*. 2008. Diakses pada 18 Mei 2008 pukul 07.42

[3] *http://www.filedudes.com/smart_reversi-download-24861.html*. 2005. Diakses pada 18 Mei 2008 pukul 07.48.

[4] *http://www.rainfall.com/othello/rules/othellorules.asp.* 2000. Diakses pada 20 Mei 2008 pukul 18.08.

[5] *http://www.site-constructor.com/othello/Present/Basic_Strategy.html.* 2001. Diakses pada 20 Mei 2008 pukul 18.10.

[6] *http://www.cs.man.ac.uk/~graham/cs2022/greedy/*. Diakses pada 20 Mei 2008 pukul 20.12.

As