Penerapan Algoritma *Greedy* pada Permainan Congklak

Micky Yudi Utama - 13514011
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
micky.yu@students.itb.ac.id

Abstrak—Makalah ini akan membahas mengenai penerapan algoritma *greedy* dalam permainan congklak. Algoritma *greedy* merupakan algoritma yang umumnya digunakan dalam persoalan optimasi. Algoritma *greedy* memiliki prinsip *"take what you can get now!"* yang berarti mengambil setiap kesempatan pada saat itu juga tanpa memperhatikan konsekuensi ke depannya. Salah satu implementasi algoritma *greedy* adalah dalam permainan congklak. Permainan congklak merupakan permainan tradisional Indonesia yang dimainkan oleh dua orang pemain. Permainan congklak menggunakan sejenis cangkang kerang sebagai biji congklak dan papan congklak sebagai tempat permainan. Permainan ini sudah mendunia dan dimainkan hampir oleh semua orang di dunia.

Kata kunci: algoritma, congklak, *greedy*, permainan.

1. PENDAHULUAN

Seiring dengan berjalannya waktu dan perkembangan zaman, *game* juga semakin berkembang. *Game* merupakan permainan terstruktur yang biasanya dibuat untuk menghilangkan stress ataupun untuk tujuan pendidikan sehingga orang yang sedang belajar tidak merasa bosan. Komponen dari *game* yaitu tujuan, aturan, tantangan, dan interaksi. *Game* memiliki beberapa jenis, atau biasanya disebut *genre game*. Beberapa jenis *game* yang ada seperti RPG (*Role Playing Game)*, FPS (*First Person Shooter)*, *Strategy*, *Sports*, *Simulation*, *Tycoon*, *Racing*, *Action*, *Arcade*, *Puzzle*, *Board Game*, dll. Setiap jenis *game* tersebut tentunya memiliki cara bermainnya tersendiri. Pada makalah ini, penulis akan membahas mengenai permainan congklak yang memiliki *genre board game*. Permainan dengan *genre board game* pada umumnya dimainkan oleh lebih dari satu orang. Permainan ini menggunakan kemampuan berpikir dalam proses permainannya. Untuk memenangkan permainan, pemain tidak bisa memainkannya secara asal-asalan melainkan terdapat teknik yang dapat digunakan untuk memenangkan permainan. Tentunya untuk *game* yang berbeda, terdapat strategi yang berbeda pula. Dalam permainan congklak, terdapat beberapa strategi yang dapat digunakan untuk memenangkan permainan, salah satunya yaitu strategi dengan menggunakan algoritma *greedy*. Algoritma *greedy* merupakan algoritma yang digunakan untuk pengambilan keputusan pada persoalan optimasi. Pada permainan congklak, persoalannya adalah bagaimana mendapatkan biji congklak sebanyak mungkin pada tempat penyimpanan di akhir permainan.

Gambar 1.1 Kumpulan *game*

2. PERMAINAN CONGKLAK

Permainan congklak merupakan permainan tradisional Indonesia yang dimainkan oleh dua orang pemain. Permainan congklak dipercayai berasal dari daerah Afrika atau Arab yang disebar ke daerah Asia melalui pedagang-pedagang Arab. Nama "congklak" berasal dari Bahasa Melayu yaitu "conglak" yang berarti pengiraan mental.

Gambar 2.1 Permainan congklak

Permainan congklak menggunakan sejenis cangkang kerang sebagai biji congklak dan papan congklak yang terdiri dari 14 lubang kecil dan 2 lubang besar yang biasa disebut lubang penyimpanan. Tujuan dari permainan ini adalah mengisi lubang penyimpanan milik sendiri dengan biji congklak sebanyak mungkin. Pemain yang pada akhir

permainan memiliki biji congklak lebih banyak dinyatakan sebagai pemenang dari permainan. Permainan ini dimainkan secara bergantian oleh dua pemain. Pada awal permainan, masing-masing lubang kecil diisi dengan 7 biji congklak. Pemain yang mendapatkan giliran pertama berhak untuk memilih salah satu lubang pada sisinya untuk dimainkan. Setelah pemain memilih salah satu lubang, pemain harus mengambil semua biji pada lubang tersebut dan kemudian menaruh biji tersebut satu persatu pada lubang-lubang selanjutnya dengan arah searah jarum jam kecuali lubang penyimpanan milik lawan. Setelah pemain selesai menaruh biji-biji tersebut, terdapat beberapa kasus:

1. Jika biji terakhir berhenti pada lubang penyimpanan milik sendiri, maka pemain mendapatkan giliran sekali lagi.
2. Jika biji terakhir berhenti pada lubang tidak kosong, maka pemain mengambil semua biji pada lubang tersebut dan melanjutkan permainan.
3. Jika biji terakhir berhenti pada lubang kosong daerah milik lawan, maka giliran pemain tersebut telah selesai dan dilanjutkan oleh pemain lawan.
4. Jika biji terakhir berhenti pada lubang kosong daerah sendiri dan pemain sudah mengitari papan permainan minimal satu kali, maka pemain mengambil biji terakhir tersebut dan semua biji pada lubang seberang (jika ada) dan menyimpannya pada lubang penyimpanan sendiri. Kondisi seperti ini sering kali dikatakan sebagai "menembak". Giliran pemain tersebut berakhir dan dilanjutkan oleh pemain lawan.

Setelah giliran pemain pertama selesai, permainan dilanjutkan oleh pemain kedua dengan peraturan yang sama dengan pemain pertama. Permainan ini dilakukan secara bergantian hingga permainan selesai. Permainan dinyatakan selesai jika semua biji congklak pada lubang kecil habis. Pemain yang memiliki jumlah biji congklak pada lubang penyimpanan lebih banyak dinyatakan sebagai pemenang. Jika pada akhir permainan, jumlah biji congklak pada lubang penyimpanan kedua pemain sama, maka permainan dinyatakan sebagai seri (tidak ada pemenang).

3. ALGORITMA *GREEDY*

Algoritma *greedy* merupakan algoritma yang paling populer digunakan dalam memecahkan persoalan optimasi. Kata "*greedy*" berasal dari bahasa Inggris yang berarti rakus, tamak, atau serakah. Prinsip algoritma *greedy* adalah "*take what you can get now!"* yang berarti mengambil setiap kesempatan yang ada pada saat itu juga tanpa memperhatikan konsekuensi ke depannya. Algoritma ini membentuk solusi dari langkah demi langkah. Pada setiap langkah tersebut akan dipilih keputusan yang memberikan hasil paling optimal pada langkah tersebut, biasanya disebut solusi optimum lokal. Keputusan yang diambil tidak memperhatikan keputusan yang akan diambil pada langkah selanjutnya dan keputusan tersebut tidak dapat diubah lagi pada langkah selanjutnya. Kemudian kumpulan keputusan inilah yang diharapkan dapat menghasilkan solusi optimum di akhir proses, biasanya disebut solusi optimum global.

Algoritma *greedy* tidak selalu memberikan solusi terbaik pada akhir proses dikarenakan pengambilan keputusan pada setiap langkah hanya memperhatikan kondisi langkah tersebut saja. Akan tetapi, algoritma *greedy* merupakan algoritma yang baik dikarenakan algoritma ini bekerja dengan cepat dan sering memberikan perkiraan nilai optimum yang baik pada setiap langkahnya. Algoritma ini juga tidak jarang memberikan solusi optimum global yang paling baik pada akhir proses.

Elemen-elemen yang digunakan dalam penerapan algoritma *greedy* antara lain:

1. Himpunan kandidat
   Himpunan ini berisi elemen pembentuk solusi.
2. Himpunan solusi
   Himpunan ini berisi kandidat-kandidat yang terpilih sebagai solusi persoalan. Himpunan solusi merupakan himpunan bagian dari himpunan kandidat.
3. Fungsi seleksi
   Fungsi yang digunakan untuk memilih keputusan yang akan diambil pada setiap langkah. Keputusan yang diambil tidak memperhatikan keputusan yang akan diambil pada langkah selanjutnya dan keputusan tersebut tidak dapat diubah lagi pada langkah selanjutnya. Biasanya keputusan yang diambil merupakan sebuah nilai numerik dan merupakan nilai terbesar/terkecil dari himpunan nilai yang ada pada suatu langkah.
4. Fungsi kelayakan
   Fungsi yang memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, di mana 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 objektif
   Fungsi yang mengoptimalkan solusi.

Dengan kata lain, algoritma *greedy* melibatkan pencarian sebuah himpunan bagian S, dari himpunan kandidat C, yang dalam hal ini, S harus memenuhi beberapa kriteria yang ditentukan, yaitu menyatakan suatu solusi dan S dioptimasi oleh fungsi obyektif.

Berikut merupakan skema umum algoritma *greedy*:

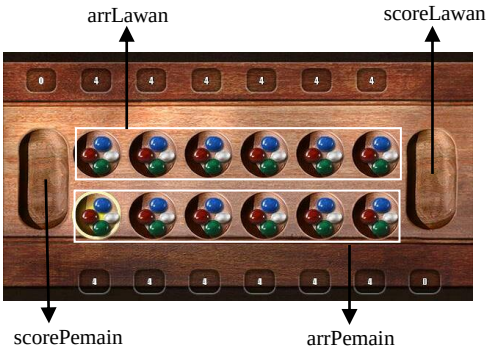
```
function greedy(input C: himpunan_kandidat) →
himpunan_kandidat
{ Mengembalikan solusi dari persoalan optimasi dengan
algoritma greedy
Masukan: himpunan kandidat C
Keluaran: himpunan solusi yang bertipe himpunan_kandidat}
Deklarasi
x : kandidat
S : himpunan_kandidat
Algoritma:
S → {} { inisialisasi 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
```

Gambar 3.1 Skema umum algoritma *greedy*

4. Representasi Algoritmik

Bentuk representasi permainan congklak dalam bahasa algoritmik:

1. arrPemain merupakan larik dengan tipe integer yang berfungsi untuk merepresentasikan jumlah batu yang terdapat pada lubang daerah pemain. Larik ini memiliki indeks dari satu hingga tujuh, dengan indeks satu merupakan lubang paling kiri dari sudut pandang pemain.
2. arrLawan merupakan larik dengan tipe integer yang berfungsi untuk merepresentasikan jumlah batu yang terdapat pada lubang daerah lawan. Larik ini memiliki indeks dari satu hingga tujuh, dengan indeks satu merupakan lubang paling kanan dari sudut pandang lawan.
3. scorePemain merupakan variabel bertipe integer yang berfungsi untuk merepresentasikan jumlah score/jumlah biji congklak pada lubang penyimpanan milik pemain.
4. scoreLawan merupakan variabel bertipe integer yang berfungsi untuk merepresentasikan jumlah score/jumlah biji congklak pada lubang penyimpanan milik lawan.



Gambar 4.1 Representasi permainan congklak dalam bahasa algoritmik

Berikut merupakan elemen-elemen algoritma *greedy* dalam permainan congklak:

1. Himpunan kandidat merupakan semua lubang pada daerah pemain kecuali lubang penyimpanan.
2. Himpunan solusi merupakan kumpulan lubang yang diambil yang diputuskan merupakan keputusan dengan hasil paling optimal pada langkah tersebut.
3. Fungsi seleksi merupakan fungsi yang digunakan untuk pemilihan lubang pada langkah tertentu, di mana lubang yang dipilih diharapkan memberikan hasil yang paling optimal pada langkah tersebut. Fungsi ini membutuhkan prioritas pemilihan lubang dalam penerapannya.
4. Fungsi kelayakan merupakan pengecekan apakah lubang yang dipilih memiliki biji. Lubang yang dipilih menjadi solusi harus memiliki biji.
5. Fungsi obyektif merupakan pengambilan keputusan yang memberikan poin paling banyak kepada pemain pada langkah tertentu.

Dalam menggunakan pendekatan *greedy*, diperlukan penentuan prioritas solusi untuk menentukan keputusan yang akan diambil pada tiap langkah. Penentuan prioritas solusi tersebut dilakukan dengan melakukan analisa pada peraturan permainan. Berikut merupakan penentuan prioritas yang digunakan oleh penulis:

1. Prioritas pertama: memilih lubang yang dapat memberikan pemain giliran lagi jika dimainkan. Pemain akan mendapat giliran lagi jika biji congklak terakhir berhenti pada lubang penyimpanan milik pemain. Jika diperoleh lebih dari 1 lubang yang memenuhi persyaratan maka lubang yang diambil adalah lubang yang paling kiri. Hal ini cukup logis, di mana dapat dilihat ketika lubang yang lebih kiri dimainkan, maka biji yang dimainkan tidak akan mempengaruhi lubang di bagian kanannya, sehingga lubang yang di bagian kanan dapat dimainkan lagi dengan syarat pertama. Sebaliknya, jika lubang yang di kanan yang dipilih, biji congklak yang diambil akan

mengisi semua lubang pada bagian kirinya dan lubang yang seharusnya memenuhi persyaratan menjadi tidak memenuhi lagi.

2. Prioritas kedua: memilih lubang di mana biji congklak terakhir akan jatuh pada lubang kosong. Jika pemain menjatuhkan biji congklak terakhir pada lubang kosong daerah pemain, maka pemain dapat mencuri semua biji congklak pada lubang seberang daerah lawan untuk memasukkannya ke lubang penyimpanan, biasanya disebut sebagai "menembak". Jika pemain menjatuhkan biji congklak terakhir pada lubang kosong daerah lawan, maka pemain dapat mencegah lawan untuk menembak. Jika terdapat lebih dari satu lubang yang memenuhi persyaratan, maka pemain perlu melakukan perbandingan. Perbandingan dilakukan terhadap jumlah biji congklak yang dicuri (ketika menembak) atau diselamatkan (ketika mencegah musuh menembak). Jika jumlah biji congklak yang dicuri lebih banyak dibandingkan jumlah biji congklak yang diselamatkan, maka pilih lubang yang memberikan pemain kesempatan untuk mencuri. Sebaliknya jika jumlah biji congklak yang dicuri lebih sedikit dibandingkan jumlah biji congklak yang diselamatkan, maka pilih lubang yang memberikan pemain kesempatan untuk menyelamatkan. Selain itu, terdapat sebuah kondisi di mana semua lubang yang memenuhi persyaratan memberikan hasil pencurian atau hasil penyelamatan nol. Jika kondisi ini ditemukan, maka lanjut pada prioritas ketiga.
3. Prioritas ketiga: memilih lubang yang memberikan pemain jumlah poin paling banyak pada giliran tersebut. Hal ini dilakukan dengan menghitung jumlah poin yang didapatkan pada masing-masing lubang. Lubang yang memberikan jumlah poin paling besar akan dipilih.

Pseudo-code penerapan algoritma *greedy* dalam permainan congklak:

```
procedure greedy (input arrPemain[1..7] :
array of integer, input arrLawan[1..7] :
array of integer)
{ prosedur yang digunakan pemain dalam
  melakukan langkah }

Kamus Lokal
  function cekPrio1 (input arrPemain[1..7] :
  array of integer, input arrLawan[1..7] :
  array of integer) -> boolean
  { fungsi yang mengembalikan nilai true
    apabila kondisi lubang daerah pemain
    memenuhi persyaratan prioritas pertama,
    false jika tidak }

  function cekPrio2 (input arrPemain[1..7] :
  array of integer, input arrLawan[1..7] :
  array of integer) -> boolean
  { fungsi yang mengembalikan nilai true
    apabila kondisi lubang daerah pemain
    memenuhi persyaratan prioritas kedua,
    false jika tidak }

  function indeksSelamat (input
  arrPemain[1..7] : array of integer, input
  arrLawan[1..7] : array of integer) ->
  integer
  { fungsi yang mengembalikan indeks lubang
    yang mengakibatkan pencegahan lawan
    melakukan menembak }

  function indeksCuri (input arrPemain[1..7]
  : array of integer, input arrLawan[1..7] :
  array of integer) -> integer
  { fungsi yang mengembalikan indeks lubang
    yang mengakibatkan pemain menembak }


  function indeksPrio1 (input arrPemain[1..7]
  : array of integer, input arrLawan[1..7] :
  array of integer) -> integer
  { fungsi yang mengembalikan indeks lubang
    yang memenuhi persyaratan prioritas
    pertama }

  function indeksPoinMaks (input
  arrPemain[1..7] : array of integer, input
  arrLawan[1..7] : array of integer) ->
  integer
  { fungsi yang mengembalikan indeks lubang
    yang memberikan poin paling maksimum pada
    kondisi persyaratan prioritas ketiga }

  function hasilCuriMax (input
  arrPemain[1..7] : array of integer, input
  arrLawan[1..7] : array of integer) ->
  integer
  { fungsi yang mengembalikan bilangan
    integer berupa hasil curi yang paling
    maksimum }

  function hasilSelamatMax (input
  arrPemain[1..7] : array of integer, input
  arrLawan[1..7] : array of integer) ->
  integer
  { fungsi yang mengembalikan bilangan
    integer berupa hasil curi yang paling
    maksimum }

  procedure melakukanLangkah (input
  arrPemain[1..7] : array of integer, input
  arrLawan[1..7] : array of integer, input
  indeks : integer)
  { prosedur untuk melakukan langkah
    berdasarkan indeks yang dipilih }
```

```
  indeks : integer
  nCuri, nSelamat : integer
  giliranBerakhir : boolean

Algoritma
  giliranBerakhir <- false
  while (giliranBerakhir = false) do
    if (cekPrio1(arrPemain, arrLawan) = true)
    then
      indeks <- indeksPrio1(arrPemain,
      arrLawan)
    else if (cekPrio2(arrPemain, arrLawan) =
    true) then
      nCuri <- hasilCuriMax(arrPemain,
      arrLawan)
      nSelamat <- hasilSelamatMax(arrPemain,
      arrLawan)
      if (nCuri ≠ 0) or (nSelamat ≠ 0)
        if (nSelamat > nCuri) then
          indeks <- indeksSelamat(arrPemain,
          arrLawan)
        else
          indeks <- indeksCuri(arrPemain,
          arrLawan)
        endif
        giliranBerakhir <- true
      endif
    else
      indeks <- indeksPoinMax(arrPemain,
      arrLawan)
      giliranBerakhir <- true
    endif
    melakukanLangkah(arrPemain, arrLawan,
    indeks)
  endwhile
```

Gambar 4.2 *Pseudo-code* algoritma *greedy* dalam permainan congklak

5. IMPLEMENTASI ALGORITMA

Untuk membuktikan keefektifan dari algoritma *greedy* dalam permainan congklak. Penulis mengimplementasikan algoritma *greedy* dengan memainkan permainan congklak melawan kecerdasan buatan pada situs http://www.games.co.id/permainan_/congklak. Hasilnya penulis memenangkan 3 set permainan dengan skor 38-10, 36-12, dan 30-18 meskipun pada situs tersebut permainan congklak yang digunakan memiliki sedikit peraturan yang berbeda di mana jumlah biji congklak yang digunakan lebih sedikit dan jika biji terakhir berhenti pada lubang yang masih terdapat biji, maka giliran pemain langsung dinyatakan berakhir.

Gambar 5.1 Permainan congklak pada situs http://www.games.co.id/permainan_/congklak

Selain itu penulis juga mencoba untuk melawan orang dengan bantuan situs http://dakon-the-game.appspot.com/. Hasilnya penulis memenangkan 3 set permainan dengan skor 67-31, 52-46, dan 80-18. Pada situs tersebut juga permainan congklak yang digunakan memiliki peraturan yang berbeda di mana tidak ada kondisi menembak.

Meskipun permainan congklak memiliki variasi peraturan yang berbeda-beda saat ini, algoritma *greedy* tetap dapat digunakan dalam permainan congklak. Pemain hanya perlu mengerti urutan prioritas yang harus digunakan dalam melangkah ketika menggunakan algoritma *greedy*.

6. KESIMPULAN

Algoritma *greedy* merupakan algoritma yang cukup sederhana, cepat, dan praktis dalam menyelesaikan persoalan optimasi, meskipun solusi yang diberikan dengan algoritma *greedy* belum tentu merupakan solusi terbaik.

Salah satu implementasi algoritma *greedy* adalah dalam permainan congklak. Algoritma *greedy* diimplementasikan dengan menentukan prioritas terlebih dahulu. Dalam implementasinya, algoritma *greedy* memberikan hasil yang cukup memuaskan meskipun belum tentu memberikan solusi terbaik.

REFERENSI

[1] http://parkshinheru.blogspot.co.id/2014/01/sejarah-congklak-permainan-tradisional.html

[2] http://www.ayobekasi.com/asal-usul-mainan-congklak/

[3] https://harlyez.wordpress.com/2014/05/10/nostalgia-yuk-ini-dia-permainan-paling-ngehits-di-tahun-90-an/congklak/

[4] http://dolananndeso.blogspot.co.id/p/congklak.html

[5] http://www.it-jurnal.com/2015/09/pengertian-algoritma-greedy.html

[6] http://www.expat.or.id/info/congklak.html

[7] http://www.expat.or.id/info/congklakinstructions.html

[8] http://ajiraksakumala.blogspot.co.id/2015/03/sejarah-dan-perkembangan-game.html

[9] http://snareproject.blogspot.co.id/2013/06/makalah-game.html

[10] http://www.games.co.id/permainan_/congklak

[11] http://dakon-the-game.appspot.com/

[12] Munir, Rinaldi, "Strategi Algoritma", Informatika, Bandung: 2014.

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, 7 Mei 2016

Micky Yudi Utama - 13514011