Penerapan Algoritma *Greedy* dalam Permainan Bantumi

Andi Setiawan

Program Studi Teknik Informatika
Institut Teknologi Bandung
Jalan Ganesha 10 Bandung
e-mail: andise@students.itb.ac.id

ABSTRAK

Algoritma *greedy* adalah salah satu algoritma yang sederhana untuk menyelesaikan persoalan optimasi. Idenya sangatlah sederhana, yaitu '*take the best, what you can get now*', yang intinya adalah mengambil pilihan terbaik yang ada pada setiap tahap dalam suatu proses.
Bantumi adalah sebuah permainan yang sederhana yang terdapat dalam telepon genggam Nokia seri 3 monokromik. Permainan ini sama seperti permainan tradisional yang terdapat pada beberapa negara Asia yaitu Indonesia (congklak) dan Malaysia(Congkak). Namun negara-negara Eropa dan Amerika lebih banyak mengenal permainan Bantumi, yang namanya berasal dari Afrika.
Makalah ini akan membahas hasil penerapan algoritma *greedy* untuk memenangkan permainan Bantumi melawan kecerdasan buatan tingkat tersulit yang dibuat pada permainan Bantumi, baik yang terdapat pada telepon genggam Nokia, maupun perangkat lunak permainan Bantumi yang dapat diunduh pada beberapa situs internet.
Makalah ini juga membahas tentang permainan bantumi, baik sejarah permainan ini, cara bermain, dan peraturan-peraturan dalam permainan ini. Kemudian, makalah ini juga membahas prinsip dasar algoritma *greed,* penerapan algoritma greedy pada permainan ini, dan hasil tes algoritma ini pada perangkat lunak permainan bantumi.

Kata kunci: Bantumi, Penerapan *Greedy*, Congklak

1. Pendahuluan

Permainan Bantumi berasal dari Afrika yang biasa dimainkan oleh dua orang, nama lain dari Bantumi adalah Mancala. Permainan sejenis ini juga terdapat di beberapa negara Asia seperti Indonesia (congklak) dan Malaysia (Congkak). Namun, belum ada informasi yang jelas mengenai tempat asal mula permainan ditemukan untuk pertama kali.

Permainan Bantumi biasanya menggunakan bijih atau batu yang dimainkan pada sejenis papan yang memiliki lubang dengan jumlah baris dua buah dan jumlah kolom enam buah (jumlah lubang dapat bervariasi). Selain papan, permainan ini juga membutuhkan dua buah mangkuk. Dalam beberapa versi, permainan ini juga dapat dimainkan dengan menggali lubang pada tanah, atau dengan mengukir batu untuk menandai lubang-lubang dan mangkuk tersebut. Kedua buah mangkuk disebut 'tempat penyimpanan', yaitu tempat untuk menyimpan bijih atau batu.

Gambar 1. Papan Bantumi dari Afrika Barat

Inti dari permainan ini adalah mendapatkan batu sebanyak-banyaknya pada mangkuk yang terletak di sebelah kanan pemain. Pemain yang mendapatkan jumlah batu terbanyak pada mangkuk yang terletak di sebelah kanannya adalah pemenang.

Cara bermain Bantumi sangat sederhana. Setiap lubang pada papan diisi batu sebanyak empat buah. Sedangkan kedua buah mangkuk dikosongkan. Kedua pemain duduk berhadapan. Pemain yang mendapat giliran, dapat memilih salah satu dari enam lubang berisi batu yang ada pada sisinya. Semua batu yang terdapat pada lubang itu diambil, dan dijatuhkan satu persatu pada lubang lainnya. Dijatuhkan dengan berlawanan arah jarum jam. Lubang yang dapat menerima batu tersebut adalah lubang pada sisi pemain, lubang pada sisi lawan, dan mangkuk yang terdapat di sebelah kanan pemain. Jadi, mangkuk yang berada di sisi kiri pemain tidak dapat menerima batu, hanya dilewati saja. Aturan pada permainan ini adalah:

1. **Jika batu terakhir dijatuhkan pada mangkuk.** Pemain akan mendapatkan gilirannya kembali. Pemain bisa mendapatkan giliran sebanyak mungkin jika ia menjatuhkan batu terakhir pada mangkuk.

2. **Jika batu terakhir dijatuhkan pada lubang kosong pada sisi pemain.** Pemain akan mendapatkan batu terakhir yang dijatuhkan pada lubang kosong. Selain itu, pemain juga dapat mencuri semua batu pada lubang yang sejajar lubang kosong tersebut di sisi lawan. Lalu semua batu tersebut diletakkan di mangkuk sebelah kanan pemain.
3. **Jika batu terakhir dijatuhkan di tempat lain (selain dua tempat diatas).** Maka giliran akan berpindah ke lawan.
4. Permainan dilanjutkan hingga semua lubang pada sisi salah satu pemain tidak memiliki batu lagi. Batu yang terdapat pada sisi lawan akan dimasukkan ke mangkuk lawan.

2. Algoritma *Greedy*

Algoritma *Greedy* adalah salah satu algoritma yang membentuk solusi langkah per langkah. Ada banyak langkah yang harus dieksplorasi pada setiap solusinya. Oleh karena itu, pada setiap langkah harus mengambil keputusan yang terbaik dari setiap pilihan. Dan keputusan yang telah diambil tidak dapat diubah lagi pada langkah selanjutnya. Pendekatan algoritma *greedy* adalah mengambil keputusan yang tampaknya terbaik. Dengan mengambil solusi optimum lokal pada setiap langkah, diharapkan dari solusi optimum lokal tersebut didapatkan solusi global (solusi akhir) yang optimum.

Algoritma *greedy* adalah algoritma yang memecahkan masalah langkah per langkah dengan mengambil pilihan yang terbaik yang dapat diperoleh saat itu tana memperhatikan konsekuensi ke depan, prinsip dari algoritma ini adalah "*take the best what you can get now!*". Dan berharap bahwa dengan memilih solusi optimum lokal pada setiap langkah akan berakhir dengan optimum global.

2.1 Skema Umum Algoritma *Greedy*

Elemen-elemen dalam persoalan optimasi algoritma *greedy*:

1. **Himpunan kandidat, C**.
   Himpunan ini berisi elemen pembentuk solusi.
2. **Himpunan solusi, S.**
   Berisi kandidat yang terpilih sebagai solusi persoalan. Dengan kata lain, himpunan solusi adalah himpunan bagian dari setiap himpunan kandidat.
3. **Fungsi seleksi**
   Fungsi yang pada setiap langkah memilih pilihan paling memungkinkan mencapai solusi yang paling optimal. Pilihan yang sudah dipilih, tidak pernah dipertimbangkan lagi pada masalah selanjutnya. Biasanya setiap kandidat x menunjuk sebuah nilai numerik, dan fungsi seleksi memilih x yang mempunyai nilai terbesar atau memilih nilai x yang memiliki nilai terkecil.
4. **Fungsi kelayakan**
   Fungsi yang memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk dan 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 memaksimumkan atau meminimumkan nilai solusi.

Dengan kata lain, persoalan optimasi yang diselesaikan dalam 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 objektif. Secara umum, skema algoritma *greedy* dirumuskan:

1. Inisialisasi S dengan kosong
2. Pilih kandidat dari C (dengan fungsi seleksi).
3. Kurangi c dengan kandidat yang sudah dipilih dari langkah 2.
4. Periksa apakah kandidat yang dipilih tersebut bersama-sama dengan himpunan solusi membentuk solusi yang layak. Jika ya, masukan kandidat tersebut kedalam himpunan solusi; jika tidak, buang kandidat tersebut dan tidak perlu dipertimbangkan lagi.
5. Periksa apakah himpunan solusi sudah memberikan solusi yang lengkap dengan menggunakan fungsi solusi. Jika ya, berhenti; jika tidak, ulangi langkah 2.

Algoritma *greedy* tidak selalu memberikan solusi yang optimal pada setiap masalah. Namun pada beberapa permasalahan optimasi, algoritma *greedy* terbukti selalu memberikan solusi yang optimal contohnya: pengaturan jadwal, permasalahan knapsack yang fraksional, dan lain-lain.

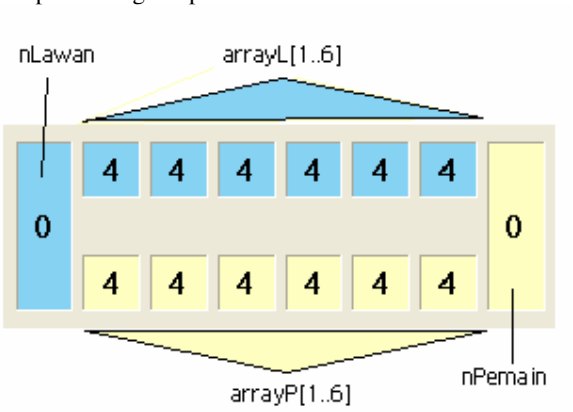
2.2 Representasi Bantumi

Bentuk representasi permainan bantumi dalam bahasa pemrograman:

1. ArrayP: adalah larik yang berisi jumlah batu yang terdapat pada lubang di sisi pemain. Indeks dari larik ini adalah dari satu hingga enam. Indeks larik terkecil berada di sebelah kiri ditinjau dari sudut pandang pemain.
2. ArrayL: adalah larik yang berisi jumlah batu yang terdapat pada lubang di sisi lawan. Indeks dari larik ini adalah dari satu hingga enam. Indeks larik terkecil

berada di sebelah kiri ditinjau dari sudut pandang pemain.

3. nLawan: adalah variabel yang berisi jumlah batu pada mangkuk lawan.
4. nPemain: adalah variabel yang berisi jumlah batu pada mangkuk pemain.



Gambar 2. Representasi Permainan

2.3 Penerapan Algoritma *Greedy*

Untuk menggunakan pendekatan *greedy* untuk menyelesaikan permainan Bantumi, prioritas solusi terbaik harus ditentukan terlebih dahulu. Penentuan prioritas tersebut dapat dilakukan dengan menganalisa peraturan permainan.

Berikut adalah asumsi yang digunakan dalam penentuan prioritas solusi yang terbaik hingga yang kurang baik:

1. **Prioritas pertama**: **memilih lubang yang dapat memberikan giliran.** Lubang terpilih adalah lubang dengan jumlah batu yang bila dijalankan, lubang terakhir yang menerima batu adalah mangkuk (berarti mendapatkan giliran). Jika terdapat dua lubang yang memenuhi prioritas pertama, maka lubang yang pertama kali dipilih adalah lubang yang paling kanan. Pertimbangan ini cukup logis, karena jika lubang yang dipilih adalah lubang yang paling kiri, hal tersebut dapat merubah kondisi lubang di sebelah kanannya. Sedangkan jika lubang yang dipilih adalah lubang solusi yang paling kanan, maka lubang solusi yang lebih kiri tidak akan terpengaruh, sehingga keduanya dapat dijalankan.
2. **Prioritas kedua: memilih lubang yang dapat berhenti pada lubang kosong di sisi pemain (*mencuri* batu lawan yang terbanyak), atau *menyelamatkan* sebanyak-banyaknya batu sendiri (jika lawan dapat berhenti pada lubang kosong di sisi lawan)**. Jika terdapat dua buah pilihan untuk menyelamatkan atau mencuri, maka untuk menentukan pilihan, terlebih dahulu dilakukan perbandingan antara jumlah keuntungan yang akan didapat atau jumlah kerugian batu yang dapat dicuri lawan. Jika jumlah kerugian lebih kecil dibandingkan jumlah keuntungan. Maka jalankan lubang yang memberi keuntungan. Dan sebaliknya. Apabila terdapat dua atau lebih lubang yang memberikan keuntungan atau kerugian yang sama, maka lubang yang dijalankan adalah lubang solusi yang terletak paling kanan. Hal ini cukup logis pada kasus apabila lawan dapat mencuri batu pemain. Bila lubang yang dijalankan adalah lubang yang lebih kiri, maka dapat menambah jumlah batu pada lubang yang sebelah kanan. Sehingga kerugian bertambah. Jika terdapat lubang yang bila dijalankan berhenti pada lubang kosong, namun lubang lawan tidak terisi batu, dan jika lawan dapat berhenti pada lubang kosong, namun lubang pemain pada arah sejajar juga kosong, maka lanjut ke prioritas ketiga.
3. **Prioritas ketiga:**
   Bila tidak terdapat pilihan lubang yang memberikan solusi prioritas pertama dan kedua. Maka jalankan lubang yang berisi batu, yang paling kanan. Secara logika, lubang tersebut lebih dapat menjangkau mangkuk dan menambah poin.

Penerapan algoritma *Greedy* pada kasus ini adalah dengan memilih pilihan terbaik berdasarkan prioritas pada setiap giliran

Himpunan kandidat C adalah semua lubang pada sisi pemain yang ada batunya. (arrayP[i] ≠ 0, untuk i =1..6). dan himpunan solusi, S adalah salah satu dari himpunan kandidat (arrayP[i] yang dapat memberikan solusi maksimal).

Pseudo-code dari penerapan algoritma *greedy* untuk mencari solusi Bantumi:

```
function greedyBantumi (input
arrayP[1..6]: array of integer)-> indeks
array

deklarasi
  untung, rugi: integer
  indeks: integer

Algoritma

if (cekPrio1(arrayP)= true) then
   indeks <- cariIndeksPrio1(arrayP)
else if (cek_Prio2(arrayP = true)) then
   untung <- cariUntungMax(arrayP)
   rugi <- cariRugiMax(arrayP)
   if (untung ≠ 0)and(rugi≠ 0) then
     if (untung > rugi) then
        indeks<-cariIndeksUntung(arrayP)
     else
        indeks <- cariIndeksRugi(arrayP)
     endif
   endif
else
   indeks <- cariIndeksKanan(arrayP)
   endif
return indeks
```

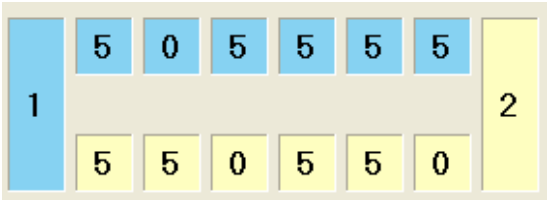
Fungsi greedyBantumi adalah fungsi untuk memilih indeks dari arrayP yang akan memberikan solusi optimum. Pertama-tama akan dilakukan pengecekan apakah solusi untuk prioritas pertama ada (dengan fungsi cekPrio1) apabila solusi ada, maka dicari indeks dari larik yang memiliki solusi tersebut. Apabila solusi untuk prioritas pertama tidak ada, maka dilakukan pencarian untuk priroritas kedua, bila terdapat solusi untuk prioritas kedua, selanjutnya mencari keuntungan terbesar (dengan fungsi cariUntungMax) yaitu jumlah batu terbesar yang dapat dicuri dari lawan, dan kemudian cari kerugian maksimum (dengan fungsi cariRugiMax). Jika keuntungan dan kerugian maksimum tersebut tidak nol, maka keuntungan dan kerugian dibandingkan, lalu jika untung lebih besar dari rugi, indeks adalah indeks larik yang memberikan keuntungan, dan sebaliknya. Jika untung atau rugi sama dengan nol atau tidak ditemukan larik yang memenuhi prioritas pertama atau kedua, maka indeks dimasukan nilai indeks larik paling kanan yang isinya tidak nol(dengan menggunakan fungsi cariIndeksKanan).

2.4 Hasil Uji Melawan Komputer

Pengujian dilakukan dengan melawan kecerdasan buatan dari perangkat lunak permainan Bantumi yang diunduh dari situs penyedia permainan bantumi secara gratis dengan alamat www.andibell.ch/download. Dari hasil pengujian, dengan algoritma *greedy* ini dapat memenangkan permainan melawan komputer tingkat tersulit dengan skor telak 36-12. Apabila pemain memulai permainan dengan giliran pertama. Namun masih belum berhasil memenangkan permainan bila pemain memulai permainan dengan giliran kedua setelah computer. Selain itu, pengujian juga dilakukan pada permainan Bantumi yang terdapat pada telepon genggam nokia seri 3330. Namun, hasil tes ini tidak dimuat dalam makalah, karena terlalu panjang.

2.5 Analisis Ketidakoptimalan Algoritma

Algoritma yang digunakan bukanlah algoritma yang sangat cerdas untuk menyelesaikan setiap kasus, ada beberapa kasus algoritma ini menghasilkan solusi yang kurnag optimal. Kasus ini dapat dilihat di gambar 3. dengan menggunakan fungsi greedyBantumi, larik yang dipilih adalah arrayP[2] yang dapat memberikan giliran. Namun jika pemain memilih arrayP[1] pemain dapat mencuri lima buah batu dari lawan.

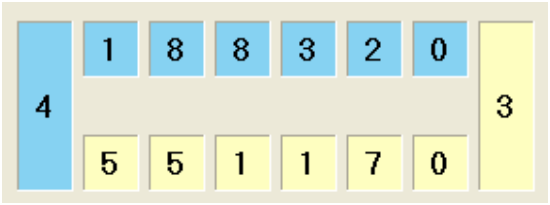


Gambar 3. Kasus tidak optimal

Namun kasus ini jarang terjadi apabila melawan komputer dengan tingkat kesulitan tinggi.

Ketidakoptimalan ini dapat diperbaiki dengan merubah fungsi prioritas, namun dengan mengubah fungsi prioritas pastinya akan ada contoh kasus ketidakoptimalan lainnya.

Selain itu algoritma ini masih terdapat kekurangan lainnya. Seperti yang terlihat pada gambar 4.



Gambar 4. Kasus tidak optimal (2)

Pada kasus gambar 4, dengan algoritma greedyBantumi akan mengeksekusi arrayP[2], lalu arrayP[5]. Padahal komputer dapat mengeksekusi arrayL[1] lalu mengeksekusi arrayL[4] yang dapat mencuri lima buah batu dari pemain.

Kasus ini dapat diperbaiki dengan penambahan pengecekan pada prioritas kedua, yaitu dengan mengecek kemungkinan lawan dapat berjalan lebih dari satu kali, lalu melakukan pengecekan nilai kerugian apabila lawan mejalankan lubang tersebut. Namun, hal ini memiliki kekurangan karena algoritma greedyBantumi menjadi sangat rumit.

IV. Kesimpulan

Algoritma *greedy* adalah algoritma yang sederhana yang dapat menyelesaikan permasalahan optimasi dengan baik pada beberapa kasus.

Penerapan algoritma greedy pada permainan bantumi dapat mengalahkan kecerdasan buatan komputer tingkat tersulit pada beberapa perangkat lunak permainan bantumi.

Ada beberapa kasus ketidakoptimalan dari penerapan algoritma *greedy*. Hal tersebut dapat diperbaiki dengan merubah prioritas atau dengan melakukan pengecekan lebih teliti.

REFERENSI

[1] Bantumi the Game, http://www.andybell.ch/e_bantumi.htm, diakses 19 Mei 2008, pukul 15.00 WIB.
[2] Bantumi(Thing), http://everything2.com/e2node/bantumi, diakses 19 Mei 2008, pukul 15.00.
[3] Congklak, http://id.wikipedia.org/wiki/Congklak , diakses 19 Mei 2008, pukul 15.00.
[4] Mancala, http://en.wikipedia.org/wiki/Bantumi , diakses 19 Mei 2008, pukul 15.00.
[5] Munir, Rinaldi, "Diktat Kuliah IF2251 Strategi Algoritmik", Program Studi Teknik Informatika, ITB, 2007.