Pencarian Solusi Optimal Pemilihan Lubang pada Permainan Congklak dengan Algoritma Greedy dan Program Dinamis

Alifia[1], Frilla Ariani[2], Tania Krisanty[3]

Laboratorium Ilmu dan Rekayasa Komputasi
Departemen Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if14088@students.if.itb.ac.id[1], if14093@students.if.itb.ac.id[2], if14101@students.if.itb.ac.id[3]

Abstrak

Permainan congklak adalah permainan tradisional yang berasal dari Jawa Tengah. Permainan congklak terdiri dari papan dakon dan jumlah kerang atau biji yang dapat dihitung dengan rumus berikut :

Jumlah biji = (jumlah lubang keseluruhan - 2) x ((jumlah lubang keseluruhan – 2) ÷ 2) (1)

Permainan ini dimainkan oleh dua orang pemain. Keduanya bergiliran memasukkan biji-bijian itu satu persatu ke cekungan papan dakon hingga habis. Permainan congklak juga dikenal dengan nama *cangka* di Srilanka dan *tungkayon* di Thailand.

Untuk menyelesaikan makalah ini kami menggunakan berbagai sumber di internet yang berkaitan dengan permainan congklak dan sebagai sumber utama adalah diktat Kuliah IF2251 “Strategi Algoritmik” yang disusun oleh Ir. Rinaldi Munir, M.T.

Melalui makalah ini kami mencoba menelaah permainan congklak bukan dari sisi bagaimana menangnya, tapi kami mencoba mencari cara menemukan langkah terpanjang. Melalui diskusi kami, masalah tersebut dapat diselesaikan dengan algoritma greedy dan dapat disempurnakan lagi melalui algoritma program dinamis.

***Kata kunci**: congklak, algoritma greedy, program dinamis*

1. Pendahuluan

1.1 Sejarah permainan congklak

Permainan congklak merupakan permainan tradisional dari adat Jawa. Menurut sejarah permainan ini pertama kali dibawa oleh pendatang dari Arab yang rata-rata datang ke Indonesia untuk berdagang atau dakwah.

1.2 Cara bermain congklak

Pada umumnya jumlah lubang keseluruhan adalah 16, yang dibagi menjadi tujuh lubang kecil dan satu lubang tujuan untuk masing-masing pemain. Lubang tujuan merupakan lubang terkiri (biasanya diameternya lebih besar). Skor kemenangan ditentukan dari jumlah biji yang terdapat pada lubang tujuan tersebut.
Setiap pemain mengambil semua biji yang terdapat pada lubang kecil yang diinginkan, untuk disebar satu biji per lubang berurutan searah jarum jam. Langkah tersebut dilakukan berulang. Apabila pada lubang terakhir meletakkan biji masih ada isinya (lubang tersebut tidak kosong) maka pemain tersebut melanjutkan dengan mengambil semua biji yang terdapat pada lubang tersebut dan melanutkan permainan. Apabila peletakan biji terakhir berada pada lubang yang kosong maka pemain tidak dapat melanjutkan langkah. Giliran untuk bermain berpindah ke lawan. Keadaan ini disebut sebagai keadaan mati.
Permainan berakhir apabila seluruh biji sudah berada pada lubang tujuan masing-masing pemain, atau apabila salah satu pemain sudah tidak memiliki biji pada lubang-lubang kecilnya untuk dimainkan (disebut mati jalan). Pemenangnya adalah yang memiliki jumlah biji terbanyak pada lubangnya.

2. Ruang Lingkup Masalah

Pada makalah ini kami membatasi masalah permainan congklak pada pencarian langkah optimal yaitu langkah terpanjang atau paling lama mati.

3. Pencarian Langkah Optimal pada Permainan Congklak

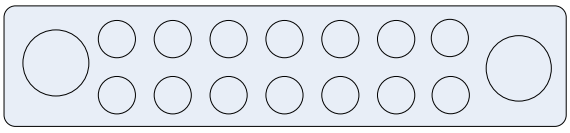
3.1 Deskripsi Umum Algoritma Greedy

Persoalan optimasi adalah persoalan yang mencari solusi terbaik. Solusi terbaik adalah solusi yang bernilai minimum atau maksimum dari sekumpulan alternatif solusi yang mungkin. Pada persoalan optimasi kita diberi sekumpulan kendala (*constraint*)

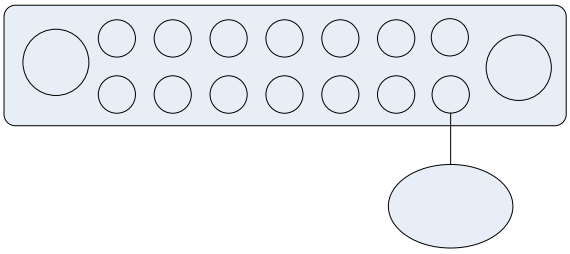
dan fungsi optimasi. Solusi yang memenuhi semua kendala disebut solusi layak (*feasible solution*). Solusi layak yang mengoptimumkan fungsi optimasi disebut solusi optimum.
Algoritma *greedy* mungkin merupakan metode yang baik untuk memecahkan persoalan optimasi. Secara harfiah *greedy* berarti rakus atau tamak. Prinsip *greedy* adalah "*take what you can get now*!". Ambil apa yang dapat diperoleh sekarang.

3.2 Pencarian Langkah Optimal dengan Algoritma *Greedy*

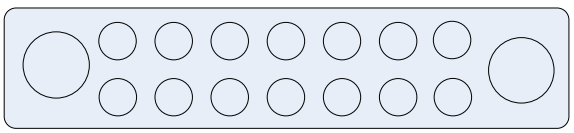
Penggunaan algoritma *greedy* tampak pada pemilihan lubang di sisi pemain yang akan diambil dan kemudian disebar bijinya. Metode *greedy* diterapkan dengan mengambil biji yang paling banyak.
Pemilihan lubang dilakukan secara iteratif untuk menemukan lubang yang mempunyai jumlah biji paling banyak.
Contoh pencarian pemilihan lubang dengan penerapan algoritma *greedy*:



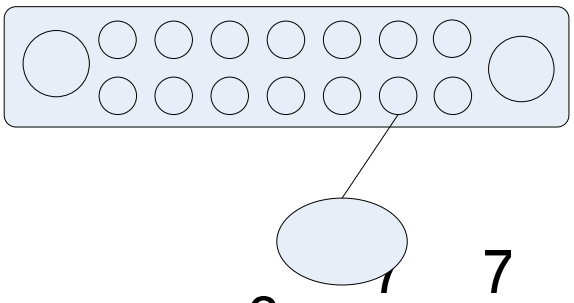
Gambar 3.2.a Kondisi awal



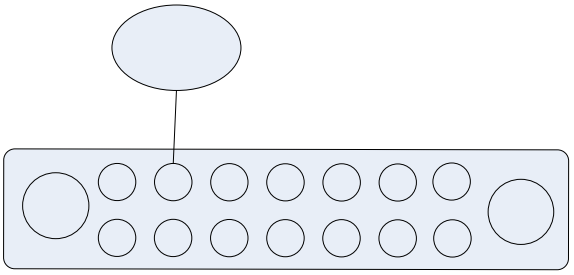
Gambar 3.2.b Kondisi memilih lubang pertama



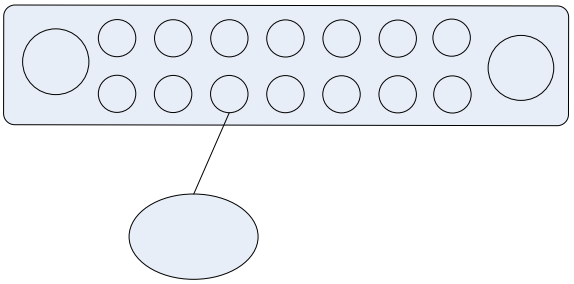
Gambar 3.2.c
Kondisi setelah biji dari lubang yang dipilih dari atas disebar



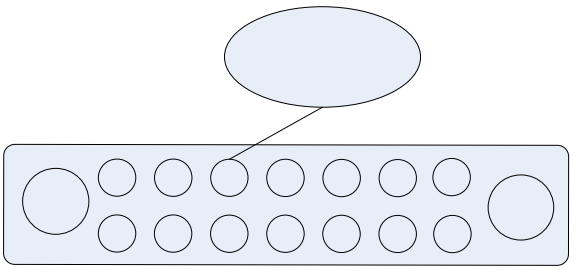
Gambar 3.2.d Kondisi memilih lubang ke dua



Gambar 3.2.e Kondisi setelah pemilihan lubang ke dua yang telah disebar dan belum mati



Gambar 3.2.f Kondisi akan mati



Gambar 3.2.g Kondisi mati

3.3 Deskripsi Umum Algoritma Program Dinamis

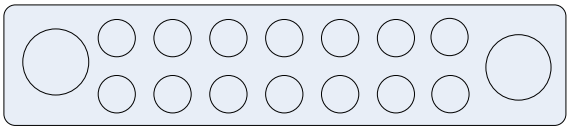
Program dinamis adalah metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan langkah dan tahapan sedemikian sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan. Program dinamis juga digunakan untuk memecahkan permasalahan optimasi yang memiliki banyak kemungkinan solusi.
Cara penyelesaian dengan program dinamis hampir sama dengan algoritma *greedy*. Pada algoritma *greedy*, keputusan diambil setiap tahap dengan cara mengambil pilihan yang memenuhi ukuran optimasi yang digunakan. Pengambilan keputusan pada tiap tahap didasarkan hanya pada informasi yang telah ditentukan dan pada setiap tahap tersebut kita tidak membuat keputusan yang salah. Pada kasus-kasus tertentu algoritma *greedy* bekerja dengan baik, tetapi pada persoalan lain tidak. Hal ini terjadi karena pengambilan keputusan pada setiap langkah *greedy*

0 7 7 7 7 7 7 0
7 7 7 7 7 7 7

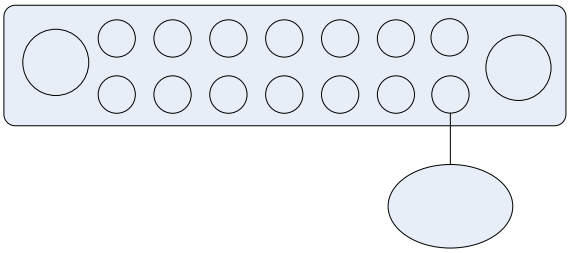
tidak pernah mempertimbangkan lebih jauh apakah pilihan tersebut pada langkah-langkah selanjutnya merupakan pilihan yang tepat. Dari sejumlah pilihan yang menarik, kita tidak tahu pilihan mana yang terbaik sampai kita menuju proses yang jauh ke depan.
Perbedaan mendasar antara algoritma *greedy* dengan program dinamis adalah bahwa metode *greedy* hanya satu rangkaian keputusan yang dihasilkan, sedangkan program dinamis lebih dari satu rangkaian keputusan yang dihasilkan.

3.4 Pencarian Langkah Optimal dengan Algoritma Program Dinamis

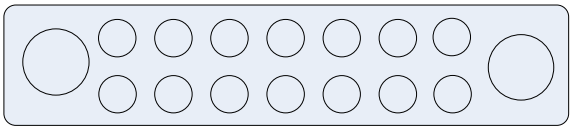
Penggunaan program dinamis tampak pada pemilihan lubang setelah terjadi keadaan berhenti.
Keadaan berhenti adalah keadaan yang mengharuskan pemain melanjutkan langkah dengan memilih lubang seperti awal permainan. Keadaan ini merupakan konsekuensi dari meletakkan biji terakhir pada lubang tujuan (lubang besar). Ketika terjadi keadaan berhenti inilah diperlukan program dinamis untuk menentukan lubang selanjutnya yang akan dipilih, agar didapat langkah terpanjang.
Contoh pemilihan langkah menggunakan program dinamis:



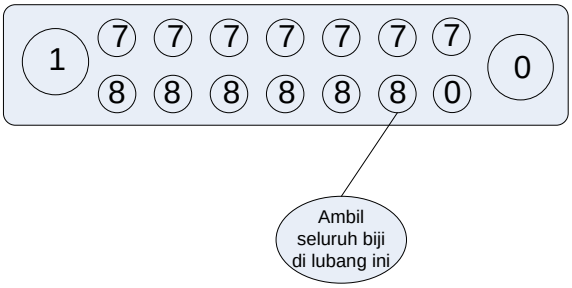
Gambar 3.4.a Kondisi awal



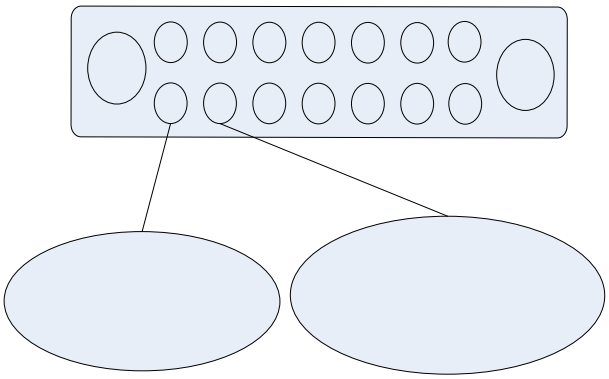
Gambar 3.4.b Kondisi memilih lubang pertama



Gambar 3.4.c
Kondisi setelah biji dari lubang yang dipilih dari atas disebar



Gambar 3.4.d Kondisi memilih lubang ke dua



Gambar 3.4.f Perbedaan pemilihan langkah antara algoritma *Greedy* dengan program dinamis.

Persoalan dibagi menjadi beberapa tahap, yang hanya akan diambil satu keputusan. Masing-masing tahap terdiri dari sejumlah status yang berhubungan dengan tahap tersebut. Hasil dari keputusan yang diambil pada setiap tahap ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya. Ongkos yang kami tentukan adalah jumlah biji yang ada di setiap lubang. Ongkos pada suatu tahap bergantung pada ongkos tahap-tahap yang sudah berjalan dan ongkos pada tahap tersebut.
Basis dari algoritma ini adalah langkah pertama dalam menentukan lubang yang akan diambil bijinya. Pemilihan tersebut harus memenuhi syarat bahwa jumlah biji yang akan diambil paling banyak dan langkah tersebut tidak akan menyebabkan kondisi mati. Seandainya setiap lubang yang dipilih akan menyebabkan kondisi mati, maka dipilih lubang dengan jumlah biji terbanyak. Kondisi mati ditentukan dengan memeriksa lubang yang jaraknya sebesar jumlah biji pada lubang yang dipilih (disebut *potential dead hole*). Jika *potential dead hole* memiliki nol biji, maka langkah ini akan menyebabkan kondisi mati, dan *potential dead hole* akan mengembalikan true Pada Gambar 3.4.f kita misalkan lubang terkanan memiliki indeks nol, dan seterusnya searah jarum jam. Kita ingin mengetahui apakah pemilihan lubang[7] yang berisi tujuh biji akan menyebabkan kondisi mati. *Potential dead hole* adalah lubang dengan indeks lubang awal ditambah

7 7 7 7 7 7 7
0 0

jumlah biji pada lubang awal tersebut. Pemeriksaan dilakukan dengan melihat jumlah biji pada *potential dead hole* (yaitu lubang[14]), karena lubang[14] berisi nol biji, maka langkah ini menyebabkan kondisi mati.
Rekurens adalah memilih biji terbanyak yang didapat dari tahap sebelumnya dan biji yang akan didapat dengan memilih suatu status (yaitu lubang selanjutnya yang optimal).

$R_k(p_k)$ = jumlah biji dari alternatif p_k pada tahap k
$f_k(x_k)$ = jumlah biji terbanyak dari tahap 1,2,..., dan k yang diberikan oleh status x_k

Relasi rekurens berikut menyatakan langkah optimal dari permainan congklak:

$$f_1(x_1) = \max_{\text{not } potential\ dead\ hole} \{R_1(p_1)\} \qquad \text{(basis)}$$

$$f_k(x_k) = \max_{\text{not } potential\ dead\ hole} \{R_k(p_k) + f_{k-1}(x_{k-1})\} \qquad \text{(rekurens)}$$

4. Kesimpulan

- Pencarian pemilihan lubang yang akan diambil melalui algoritma *greedy* belum seutuhnya optimal.
- Langkah optimal tidak bisa ditentukan hanya dengan melihat satu kali putaran.
- Solusi yang kami buat dengan algoritma ini hanyalah solusi untuk menemukan satu kali langkah sampai mati yang dilanjutkan dengan langkah lawan.

5. Daftar Pustaka

1. Expat Web Site Association, *Living in Indonesia, A Site for Expatriates*. Diakses tanggal 12 Mei 2006.
2. Munir, Rinaldi., *Strategi Algoritmik*, Teknik Informatika ITB, 2006
3. http://id.wikipedia.org/wiki/Congklak. Diakses tanggal 12 Mei 2006.