Algoritma Exhaustive Search

Dalam Permainan Congklak

Sigit Aji Nugroho (13510021)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13510021@stei.itb.ac.id

***Abstract*—Pada permainan congklak, saat pemain mendapatkan giliran bermain, maka harus memulai gilirannya dengan mengambil semua biji di salah satu lubang yang ada di sisinya. Salah satu cara termudah untuk memilih lubang yang mana untuk memulai adalah dengan memilih yang biji di dalamnya paling banyak. Dengan cara ini, jarak yang bisa ditempuh pada pengambilan pertama menjadi paling panjang. Dasar melakukan cara ini ialah semakin banyak langkah yang dibuat pemain dalam satu giliran, semakin banyak biji yang bisa dimasukkan dalam lubang induk dalam saat itu.**

Namun ternyata, cara ini belum tentu menjadi yang terbaik, karena saat biji yang digenggam habis di salah satu lubang, pemain masih bisa melanjutkan langkah dengan menggunakan biji yang ada di lubang tersebut. Jadi seharusnya sebelum memilih langkah pertama harus dimulai dari lubang yang mana, pemain sudah harus tahu jumlah biji yang bisa masuk ke dalam lubang induknya jika pemain memulai dari lubang tersebut. Untuk menentukan jumlah biji yang bisa di dapat dari masing-masing lubang, pencariannya bisa dikerjakan dengan algoritma *exhaustive search*.

***Index Terms*—Biji, *exhaustive search*, congklak, lubang sisi, lubang induk.**

I. PENDAHULUAN

Congklak adalah salah satu permainan tradisional yang ada di Indonesia. Ada beberapa nama daerah yang dimiliki oleh permainan ini, misal *dakon* di daerah Jawa, *congkak* di Sumatra, *makaotan* di Sulawesi [1]. Akibat perkembangan jaman, permainan-permainan modern mulai muncul, akibatnya permainan tradisional semacam congklak sudah mulai ditinggalkan.

Salah satu cara agar permainan ini tidak musnah adalah dengan cara terus dimainkan. Congklak adalah permainan untuk dua orang. Selain itu dibutuhkan papan permainan yang memiliki empat belas lubang kecil dan dua lubang besar. Lubang besar atau lubang induk berada di ujung papan. Sedangkan lubang kecil dibagi dua dan saling berhadapan sehingga masing-masing sisi ada tujuh [2]. Berikut adalah gambar dari papan congklak [3].

Gambar 1.1 Papan Permainan Congklak

Masing-masing lubang sisi harus diisi dengan tujuh buah biji. Total dibutuhkan 98 biji. Biji yang biasa digunakan adalah batu kecil, kulit kerang, atau biji buah-buahan.

Permainan dimulai dengan menentukan siapa pemain yang memulai terlebih dahulu. Pemain pertama memilih salah satu lubang yang ada di sisinya. Biji yang ada di dalam lubang itu di ambil kemudian di masukkan satu per satu ke dalam lubang setelahnya sesuai arah jarum jam. Semua lubang kecuali lubang induk lawan diisi. Jika biji terakhir jatuh pada suatu lubang yang ada biji-biji lain di dalamnya, maka biji di dalam lubang tersebut diambil lagi untuk diteruskan mengisi lubang-lubang selanjutnya.

Bila biji terakhir ternyata masuk ke dalam lubang induk sendiri, maka pemain bisa melanjutkan langkah dimulai dari salah satu lubang yang ada di sisinya. Saat biji terakhir berada di lubang sisi lawan yang kosong, maka gilirannya berakhir. Jika biji terakhir jatuh di lubang kosong yang ada di sisinya, maka gilirannya saat itu berakhir tetapi semua biji yang ada di lubang sisi lawan yang menjadi pasangan lubang terakhir tadi bisa dimasukkan ke lubang induk pemain [2].

Setelah semua lubang kosong maka permainan berakhir. Pemenang ditentukan dengan cara menghitung jumlah biji yang ada di masing-masing lubang induk. Pemain yang mengumpulkan biji lebih banyak akan menjadi pemenang.

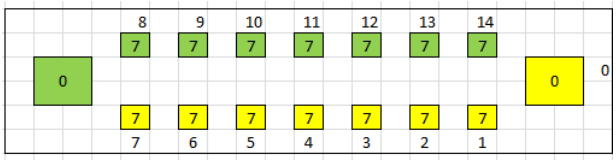
II. DASAR TEORI

Exhaustive search adalah salah satu jenis algoritma *brute force*. *Brute force* sendiri adalah algoritma dasar dalam memecahkan suatu masalah. Hampir sebagian besar masalah bisa diselesaikan dengan algoritma ini (wide applicability). Bahkan ada beberapa permasalahan yang hanya bisa di selesaikan dengan menggunakan *brute force*. Hal ini karena pola berfikirnya yang sederhana dan mudah dimengerti. Hanya saja algoritma ini tidak mangkus dan tidak kontruktif, akibatnya untuk permasalahan yang besar wantu pemecahan masalahnya jauh lebih lambat dari pada algoritma lain.

Exhaustive search adalah metode pencarian solusi untuk objek-objek kombinatorik, misalnya permutasi, kombinasi, atau himpunan bagian dari suatu himpunan. Langkah dari algoritma ini dimulai dengan mencari semua kemungkinan jalan. Selanjutnya untuk masing-masing jalan dievaluasi untuk mencari hasilnya. Jalan dengan solusi terbaik sementara disimpan. Jika jalan selanjutnya memiliki solusi yang lebih baik, maka jalan itu yang disimpan. Terakhir tinggal mengumumkan jalan dengan solusi terbaik [4].

III. METODE

A. Gambaran Model



Gambar 3.1 Model Congklak

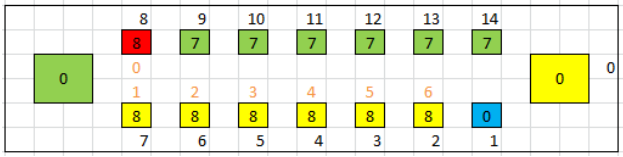
Gambar di atas menunjukkan kondisi awal papan congklak. Masing-masing lubang sisi berisi tujuh biji. Masing-masing lubang induk tidak berisi biji.

Struktur data yang digunakan bisa berupa *list*. Panjang *list* adalah lima belas. *List* indeks ke-0 digunakan sebagai lubang induk. *List* indeks ke-1 sampai ke-7 digunakan sebagai lubang sisi pemain. *List* indeks ke-8 sampai ke-14 digunakan sebagai lubang sisi lawan. Lubang induk lawan tidak dimasukkan ke dalam *list* karena tidak pernah diakses oleh pemain yang sedang jalan.

Tabel 3.1 Evaluasi Lubang Versi 1

Lubang	Penambahan	Lubang Induk
1		0
2		0
3		0
4		0
5		0
6		0
7		0

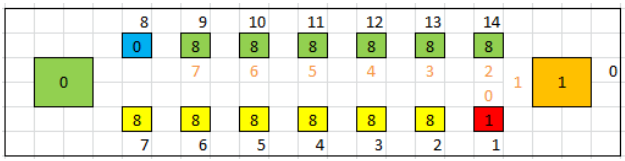
Tabel di atas hanya digunakan untuk mempermudah pencatatat. Untuk algoritma *exhaustive search* sendiri hanya perlu dicatat. Isi lubang induk maksimal (**max**) dan lubang yang menghasilkan nilai maksimal itu (**l**). Isi lubang induk maksimal diinisiasi dengan nilai nol (**max = 0**). Sementara lubangnya diinisiasi dengan nila satu (**l=1**).



Gambar 3.2 Model Congklak 2

Gambar di atas menunjukkan kondisi ketika pemain pertama menjalankan permainan mulai dari lubang sisi ke-1 miliknya. Pemain akan mengisi lubang di sebelahnya dengan satu biji sesuai arah jarum jam. Angka berwarna jingga adalah sisa jumlah biji yang masih dipegang oleh pemain setelah satu biji dimasukkan ke dalam lubang tersebut. Misal di atas lubang ke-2 tertulis angka jingga enam karena di lubang ke-1 pemain mengambil tujuh biji, kemudian dimasukkan ke lubang ke-2 satu, sehingga sisanya tinggal enam.

Akhirnya biji yang dipegang pemain habis di lubang ke-8. Pemain melanjutkan permainan dengan mengambil semua biji di lubang ke-8. Selanjutnya mengisi lubang setelahnya.



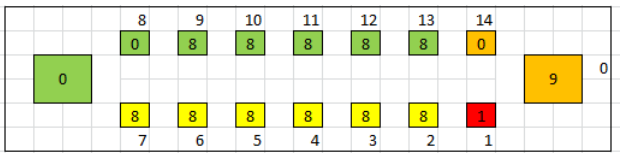
Gambar 3.3 Model Congklak 3

Pada kesempatan kedua ini ternyata pemain melalui lubang induknya. Akibatnya isi lubang induk bertambah menjadi satu. Pada tabel evaluasi lubang juga mengalami perubahan.

Tabel 3.2 Evaluasi Lubang Versi 2

Lubang	Penambahan	Lubang Induk
1	1	1
2		0
3		0
4		0
5		0
6		0
7		0

Padakesempatan kedua ini, biji terakhir jatuh pada lubang ke-1. Sesuai peraturan jika biji terakhir jatuh ke lubang kosong yang ada di sisi pemain, maka gilirannya berhenti. Namun semua biji yang ada di seberang lubang itu itu, lubang ke-14, bisa diambil dan dimasukkan ke lubang induknya.



Gambar 3.4 Model Congklak 4

Kondisi ini mengubah isi tabel evaluasi lubang lagi. Karena isi dari lubang ke-14 ada delapan biji, isi lubang induk ditambah delapan. Selain itu, **max** dan **l** juga bisa di-*update*. Nilai **max** menjadi sembilan sementara **l** masih satu.

Tabel 3.3 Evaluasi Lubang Versi 3

Lubang	Penambahan	Lubang Induk
1	1+8	9
2		0
3		0
4		0
5		0
6		0
7		0

Sebenarnya tabel evaluasi lubang bisa menjadi lebih panjang. Hal ini terjadi jika suatu saat biji terakhir berada di lubang induk pemain. Sesuai peraturan maka pemain bebas memulai lagi langkahnya dari salah satu lubang sisi miliknya. Akibatnya akan ada tujuh kemungkinan kombinasi baru. Dalam tabel di bawah, dicontohkan jika langkah awal adalah lubang ke-4 maka akan mengalami kondisi tersebut.

Tabel 3.3 Evaluasi Lubang Versi 3

Lubang	Penambahan	Lubang Induk
1	1+1	2
2	0	0
3	1+1+4	6
4-1		0
4-2		0
4-3		0
4-4		0
4-5		0
4-6		0
4-7		0
5		0
6		0
7		0

Kondisi ini tidak akan mempersulit pengelolaan struktur data karena memang tabel ini tidak perlu dibuat. Hanya saja, **max** perlu diakali. Caranya tinggal membuat max dalam tipe **string**.

B. Algoritma Evaluasi Lubang

Algoritma ini digunakan untuk menentukan isi dari lubang induk saat giliran berakhir jika langkah dimulai dari lubang ke-n. Nilai n berkisar antara satu hingg tujuh, yaitu indeks untuk lubang sisi milik pemain. Nilai keluarannya berupa integer lebih besar atau sama dengan nol.

Algoritma yang digunakan:

```
procedure Eval ( input n : integer, congklak: integer[15], output akhir: integer, jalur: string )
{menghitung jumlah akhir lubang induk giliran itu jika langkah dimulai dari lubang sisi ke-n
Masukan: n (salah satu indeks lubang sisi , 1-7)
Keluaran: isi lubang induk akhir
}

Deklarasi
        stop, i , idx, m, temp: integer
        List2 : integer[15]
        S1, S2, S3, S4, S5, S6, S7: string
        a, b, c, d, e, f, g: string
Algoritma
        stop ← 0
        idx ← 1
        m ← n
        List ← congklak
        Jalur←""
        while (stop =0 and n><0)do
                List[n] ← 0
                for i←0 to List[n]-1 do
                        temp ← (n+1+i) mod 15
                        List[temp]++
                        idx ← temp
                endfor

                if (List[idx]=1 and idx>7 ) then
{berhenti di kosong dan di sisi lawan}
                        Stop←1
                else if (List[idx]=1 and 0<idx<8 ) then
{berhenti di kosong dan di sisi sendiri}
                        List[0]
←List[0]+List[15-idx]
                        List[15-idx] ←0;
                else if (List[idx]><1 ) then
{berhenti di tidak kosong}
                        n=idx
                else {berhenti di lubang induk}
                        List2←List
                        Eval(1, List2, a, S1)
                        List2←List
                        Eval(7, List2, b, S2)
                        List2←List
                        Eval(6, List2, c, S3)
                        List2←List
                        Eval(5, List2, d, S4)
                        List2←List
                        Eval(4, List2, e, S5)
                        List2←List
                        Eval(3, List2, f, S6)
```

```
                    List2←List
                    Eval(2, List2, g, S7)

                    Max(a,b,c,d,e,f,g)
  {prosedur max akan mencari jalur mana
yang akan mencari nilai terbesar antara a-
g. jika yang terbesar c maka List[0] akan
ditambah c, dan jalur akan ditambah dengan
S3}
            endif
      endwhile
      akhir←List[0]
```

C. Algoritma Exhaustive Search

Algoritma ini akan memanggil fungsi Eval untuk lubang ke-1 hingga ke-7. Jika nilai akhir lubang induk baru lebih besar, maka **max** dan **l** akan diganti. Begitu seterusnya hingga lubang ke tujuh selesai dievaluasi.

Algoritma yang digunakan:

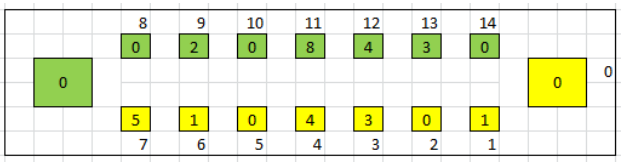
```
procedure ES( )
{menghasilkan jalur yang menghasilkan
lubang induk dengan isi terbanyak
List telah diketahui.
}

Deklarasi
      i, akhir, max: integer
      s,l: string
Algoritma
      max←0
      l←1
      for i←1 to 7 do
            s←integerToString(i)
            Eval(i,List, akhir, s)
            if(akhir>max)then
                  max←akhir
                  l←s
            endif
      endfor
      output (lubang maksimal: s)
```

D. Penggunaan Algoritma

Dua algoritma di atas hanya dipakai searah (dari satu sudut pandang pemain saja). Jika pemain lawan akan menggunakan algoritma yang sama maka dibutuhkan penyesuaian. Selain itu jika pemain lawan telah melakukan langkah yang berakibat perubahan posisi dan isi biji, maka *List* yang hendak digunakan dalam algoritma ES harus di-*update* terlebih dahulu.

Kondisi awal dari masing-masing lubang induk boleh tetap boleh dinolkan terlebih dahulu. Hal ini tidak akan mempengaruhi kinerja agoritma *exhaustive search* dalam menentukan lubang awal terbaik. Jika dinolkan terlebih dahulu maka jumlah biji yang akan didapatkan dari masing-masing lubang akan lebih mudah dilihat. Sedangkan jika tidak pernah dinolkan maka akan diketahui jumlah biji yang didapat pada akhir permainan.

IV. APLIKASI PENGGUNAAN EXHAUSTIVE SEARCH



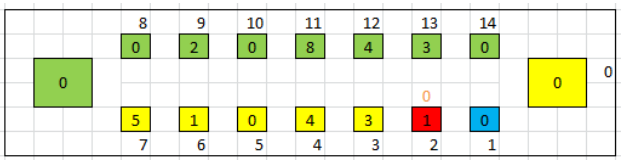
Gambar 4.1 Model Congklak 5

Algoritma *exhaustive search* dan evaluasi lubang di bagian sebelumnya akan coba diimplementasikan. Implementasinya akan dilakukan pada contoh kasus yang ada pada gambar di atas. Kedua lubang induk telah dinolkan sebelumnya. Isi dari masing-masing lubang sisi seperti yang tertera pada gambar.

Tabel 4.1 Evaluasi Lubang Versi 4

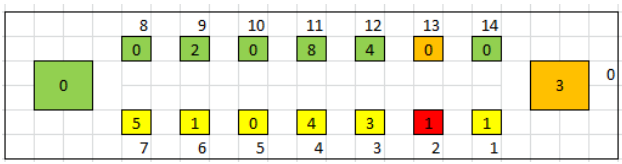
Lubang	Penambahan	Lubang Induk
1		0
2		0
3		0
4		0
5		0
6		0
7		0
	l= 1	max=0

Saat prosedur **ES** dijalankan maka **max** akan diisi 0 dan **l** diisi 1. Selanjutnya prosedur akan melakukan iterasi mulai dari **i=1**. Prosedur Eval untuk lubang ke-1 dibaca.



Gambar 4.2 Model Congklak 6

Isi lubang ke-1 hanya satu. Biji itu dimasukkan ke dalam lubang ke-2. Ternyata lubang ke-2 kosong. Artinya giliran telah selesai. Namun karena berhenti di lubang sisi pemain, sesuai algoritma isi lubang ke-13 dipindah ke lubang induk.



Gambar 4.3 Model Congklak 7

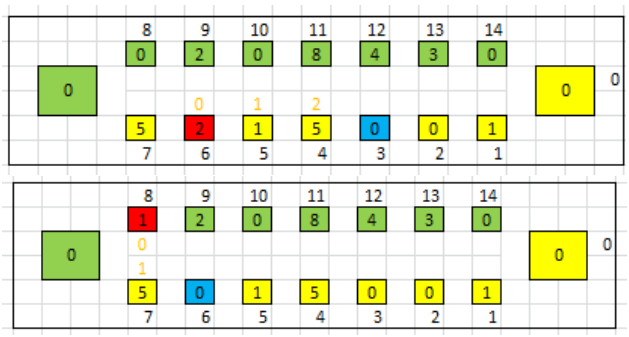
Nilai List[0] mengalami perubahan. Nilai yang baru adalah tiga. Dalam kedua prosedur nilai ini disimpan dalam parameter **akhir.** Karena nilai **akhir** lebih besar dari nilai **max,** nilai **akhir** di-*copy* ke nilai **max.** Nilai **s** juga di-*copy* ke **l**. Tapi nilainya tetap 1.

Tabel 4.2 Evaluasi Lubang Versi 5

Lubang	Penambahan	Lubang Induk
1	3	3
2		0
3		0
4		0
5		0
6		0
7		0
	l= 1	max=3

Selanjutnya iterasi dalam prosedur ES sampai ke lubang ke-2. Prosedur Eval di panggil. Namun karena isi dalr lubang ke-2 nol, akibatnya tidak terjadi apa-apa.

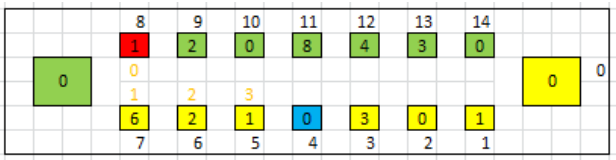
Iterasi kembali dilanjutkan ke lubang ke-3. Biji di lubang ke-3 diambil lalu diisikan ke lubang-lubang selanjutnya. Perngisian berakhir di lubang ke-6. Karena lubang ini masih ada isinya, maka perjalanan dilanjutkan dari lubang ini. Biji terakhir masuk ke lubang ke-8. Sebelum diisi, lubang ini kosong, maka menurut peraturan giliran berakhir.



Gambar 4.4 Model Congklak 8

Selama putaran yang dimulai dari lubang ke-3 di atas, tidak ada perubahan isi lubang induk. Artinya lubang induk masih tak berisi. Karena masih nol artinya lebih kecil dari **max** sebelumnya. Karena itu tidak ada perubahan nilai **max** dan **l**.

Iterasi dilanjutkan dari lubang ke-4. Langkahnya masih sama. Kali ini berakhir di lubang ke-8. Karena lubang ke-8 awalnya kosong, maka yang terjadi seperti akhir putaran yang dimulai dari lubang ke-3.

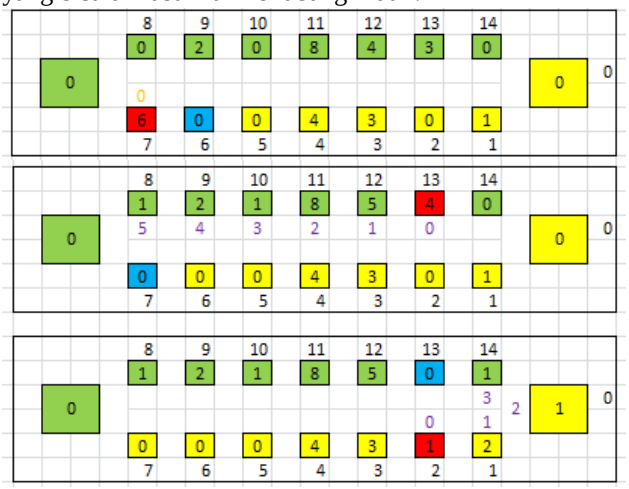


Gambar 4.5 Model Congklak 9

Iterasi selanjutnya adalah lubang ke-5. Namun ternyata lubang ke- tidak ada isinya. Alhasil tidak akan ada perubahan.

Iterasi dilanjutkan ke lubang ke-6. Langkah pertama berhenti di lubang ke-7. Namu karena masih ada isinya, dilanjutkan lagi. Langkah kedua berhenti di lubang ke-13. Karena masih ada isinya lagi maka dijalankan lagi hingga lubang ke-2. Saat langkah ini, pemain melalui lubang induk, sehingga isi lubang induk bertambah.

Lubang ke-2 sebelumnya tidak ada isinya. Artinya giliran pemain telah berakhir. Namun pemain berhenti di lubang sisi miliknya, sehingga boleh mengambil biji di lubang seberang. Akan tetapi, lubang seberangnya, yaitu lubang ke-13 tidak ada isinya. Sehingga tidak ada biji yang bisa dimasukkan ke lubang induk.



Gambar 4.6 Model Congklak 10

Pada giliran ini terjadi perubahan isi. Nilai **akhir** dibandingkan dengan nilai **max**. Akan tetapi nilai **akhir** lebih kecil. Artinya nilai max masih 3 dan terjadi melalui lubang ke-1.

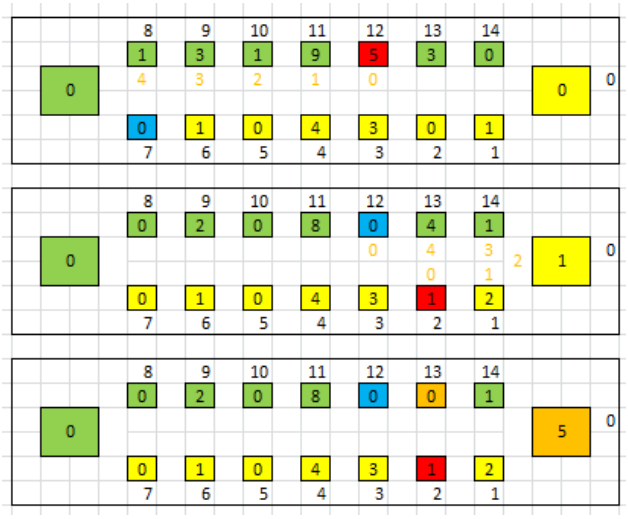
Tabel 4.3 Evaluasi Lubang Versi 6

Lubang	Penambahan	Lubang Induk
1	3	3
2		0
3		0
4		0
5		0
6	1	1
7		0
	l= 1	max=3

Terakhir iterasi sampai ke lubang ke-7. Isi lubang ke-7 ada lima biji. Masing-masing biji dibagikan hingga ke lubang ke-12. Lubang ke-12 juga ada isinya, yang kemudian juga dimasukkan satu-satu hingga ke lubang ke-2. Pada langkah ini, pemain melalui lubang induk, sehingga lubang induk berisi satu biji.

Pada langkah terakhir pemain berhenti di lubang ke-2. Ternyata lubang kedua sebelumnya tidak berisi. Artinya giliran pemain berakhir\. Namun karena lubang ke-2 ada di sisinya, maka pemain berhak mengambil biji yang ada di lubang seberangnya, yaitu lubang ke-13.

Lubang ke-13 berisi empat biji. Biji ini dimasukkan ke dalam lubang induk. Sehingga jumlah biji di lubang induk sekarang ada lima.



Gambar 4.6 Model Congklak 10

Terjadi perubahan nilai **List**[0]. Nilai ini disimpan dalam parameter **akhir** yang keluar dari prosedur **Eval**. Nilai ini dibandingkan dengan nilai **max**. ternyata lebih besar. Sehingga nilai **max** yang baru menjadi lima (max=5) dan nilai **l** menjadi "7" (**l="7"**).

Tabel 4.4 Evaluasi Lubang Versi 7

Lubang	Penambahan	Lubang Induk
1	3	3
2		0
3		0
4		0
5		0
6	1	1
7	1+4	5
	l= 7	max=5

Pada akhir prosedur ES ditampilkan sebuat *output*. *Output* ini menjelaskan lubang mana yang akan memberikan biji ke lubang induk terbanyak. Output yang tertulis adalah "**lubang maksimal: 7**".

V. KESIMPULAN

Algoritma *Exhaustive Search* sangat sederhana dan mudah dipahami. Hanya saja waktu eksekusinya cukup lama (kompleksitasnya tinggi). Dapat dibayangkan jika setiap hendak mengambil satu langkah awal pemain harus mengevaluasi semua langkah terlebih dahulu, tentu saja akan membutuhkan waktu yang sangat lama. Kecenderungan pemain akan lebih memilih untuk memilih secara asal atau melakukan evaluasi secara singkat saja tanpa menyeluruh.

Kondisi ini memang sesuai dengan karakter algoritma *exhaustive search* yang merupakan bagian dari algoritma *brute force*. Kelebihannya sederhana dan *wide applicability*. Namun tidak kreatif dan kompleksitasnya tinggi.

VI. ACKNOWLEDGMENT

Penulis ingin mengucapkan terima kasih yang paling besar kepada Allah SWT. Dialah yang senantiasa memberikan kesehatan dan kesempatan bagi penulis sehingga bisa menyusun makalah ini dan menyelesaikan mata kuliah Strategi Algoritma. Alhamdulillah.

Ucapan terima kasih selanjutnya diberikan kepada kedua dosen mata kuliah ini, Bapak Rinaldi Munir dan Ibu Masayu Leyla Khodra. Berkat bimbingan beliau akhirnya ilmu tentang Strategi Algoritma bisa dimengerti oleh penulis. Semoga ilmu ini akan terus bermanfaat kedepannya.

Ucapan terakhir diberikan kepada teman-teman sekelas penulis. Selain itu juga kepada rekan sekelompok tugas. Atas kerjasama dan dukungan dari mereka semua, segala tugas besar dan kecil dari mata kuliah ini bisa dilalui.

REFERENCES

[1] http://share.pdfonline.com/35c4afd73da64e2a80dfcf4382a8c78a/BAB%201%20%28Revisi%2012%20Des.%29.htm
Waktu akses: Jumat, 21 Desember 2012 pukul 07.27 WIB.

[2] http://gameduniaanak.blogspot.com/2011/10/cara-bermain-mainan-tradisional.html
Waktu akses: Jumat, 21 Desember 2012 pukul 07.39 WIB.

[3] http://4.bp.blogspot.com/-Uw1gv8wtjL0/Tk8qrXSFybI/AAAAAAAACQ4/r2GgQqkgTMo/s1600/congklak%2Bgame.jpg
Waktu akses: Jumat 21 Desember 2012 pukul 07.43 WIB.

[4] Munir, Rinaldi. *Diktat Kuliah IF3051 Strategi Algoritma*. Bandung: Program Studi Teknik Informatika STEI ITB, 2009, hal. 8-26.

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, 21 Desember 2012

Sigit Aji Nugroho
13510021