ANALISIS ALGORITMA MINIMAX DENGAN OPTIMASI ALPHA-BETA PRUNING TERHADAP WAKTU PADA *GAME* CONGKLAK



Oleh : Sarah Nurhasanah(10107493)

Penguji 1 : Utami Dewi W, S.Kom.

NIP. 41277006035

Penguji 2 : Mira Kania Sabariah, S.T.,M.T.

NIP. 41277006008

Penguji 3 : Ednawati Rainarli, S.Si,. M.Si

NIP. 41277006125

5 februari 2013

Latar belakang

- Permainan congklak merupakanperhitungan matematika perjumlahan dan pengurangan yang dibutuhkan kecepatan dan strategi yang cepat dan tepat dalam memilih biji yang akan dimainkan agar mendapatkan langkah optimum yang tepat dalam waktu yang cepat.

- Sedangkan berdasarkan penelitian sebelumnya yang dilakukan Citra Anindya[1] terdapat kelemahan yaitu penelitiannya hanya mencari solusi optimum dalam permainan congklak, namun tidak mempertimbangkan waktu komputasi atau waktu proses yang dibutuhkan komputer untuk mengambil solusi

Rumusan masalah

- Bagaimana menganalisis algoritma *Minimax* dengan dioptimasi *Alpha-Beta Pruning* untuk menghemat waktu komputasi dalam mengambil sebuah solusi dalam permainan congklak.

Maksud

- **Maksud** dari penelitian ini adalah untuk menganalisis Algoritma *Minimax* dengan optimasi *Alpha-Beta Pruning* pada *Game* Congklak terhadap waktu.

Tujuan

- **Tujuan** dari penelitian ini adalah untuk membuktikan algoritma *Minimax* dengan optimasi *Alpha-Beta Pruning* dalam menghemat waktu komputasi komputer untuk mengambil solusi dalam permainan congklak.

Batasan Masalah

- Algoritma yang digunakan adalah *Minimax dengan Alpha beta pruning*, algoritma ini digunakan untuk komputer.
- Pemain bisa memilih jumlah lubang dan biji yang diinginkan.
- Pemain bisa memilih siapa yang akan bermain pertama.
- Permainan akan dibatasi dengan level permainanan, yaitu mudah, sedang, sulit.
- Aturan permainan disesuaikan dengan permainan congklak versi jawa yaitu dakon, hanya saja akan dimodifikasi beberapa peraturannya.
- Model analisis yang digunakan adalah model kompleksitas waktu (Tn).
- Notasi asimtotik yang digunakan adalah O (big oh).

Metode Penelitian

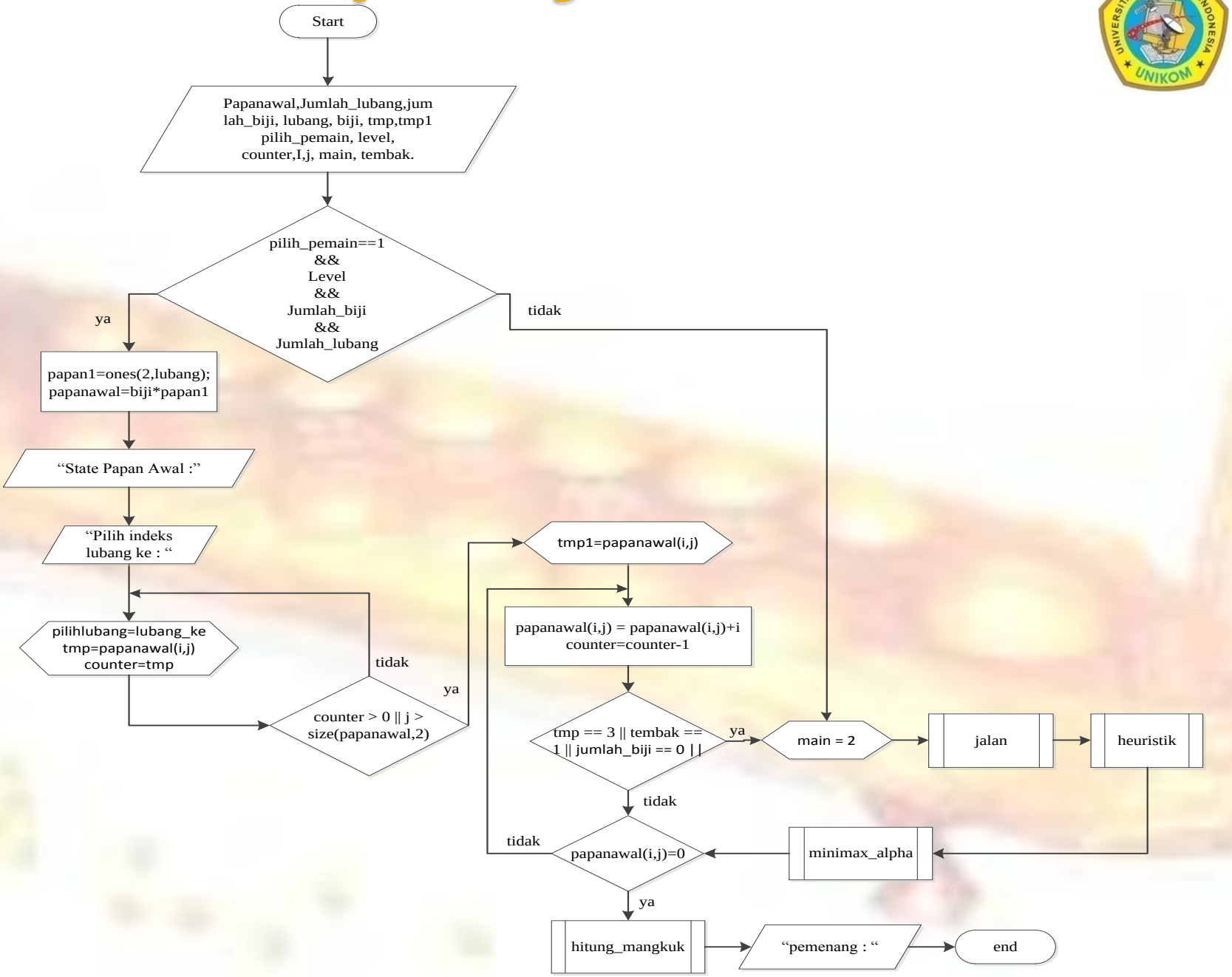
Metode penelitian yang digunakan dalam penulisan penelitian ini adalah sebagai berikut :

1. Pengumpulan data :
    a. Studi Literatur.
    b. Observasi.
2. Eksperimen

Analisis Masalah

- Pada penelitian ini pemain diijinkan untuk memilih jumlah lubang yang diinginkan yang akan dibatasi hingga 7 lubang, dengan banyaknya lubang maka akan semakin lama permainan berlangsung dan akan lebih diperlukan kemampuan yang lebih dalam perhitungan matematika untuk memilih biji mana yang akan diambil, dan pada penelitian sebelumnya pun menjelaskan bahwa komputer akan sangat lama merespon.

Alur Permainan pada Congklak



Aturan Permainan Congklak

1. Permainan dimainkan oleh dua pemain yaitu, pemain *human* dan pemain *computer*.
2. Awal permainan :
    a. Pemain bisa memilih level yang akan dimainkan
    b. Pemain bisa memilih jumlah lubang
    c. Pemain bisa memilih siapa yang akan bermain pertama
3. Ketika permainan dimulai pemain memilih biji pada lubang miliknya dan menjatuhkan satu persatu biji tersebut ke lubang miliknya searah jarum jam
4. Proses “Tembak”

Lanjutan Aturan Permainan Congklak

5. Akhir giliran pemain terdapat 3 kondisi yaitu :

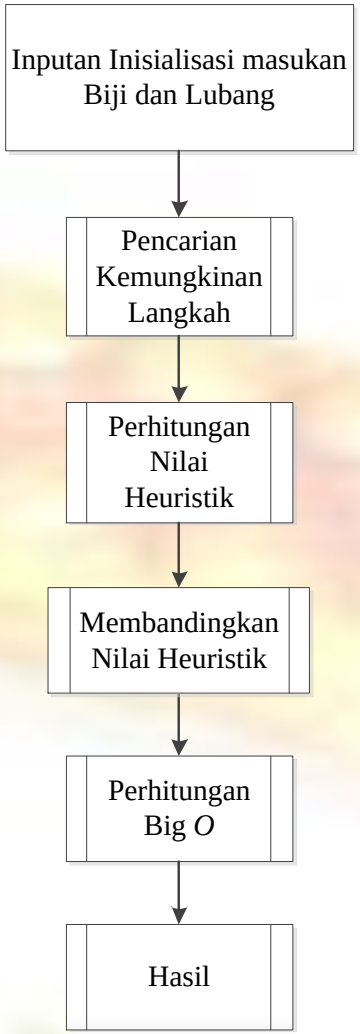
a. Jika biji terakhir yang dijatuhkan pemain berada pada lubang kosong miliknya dan sudah mengambil 3 kali biji maka gilirannya berakhir.

b. Jika biji terakhir yang dijatuhkan pemain berada pada lubang kosong milik lawan

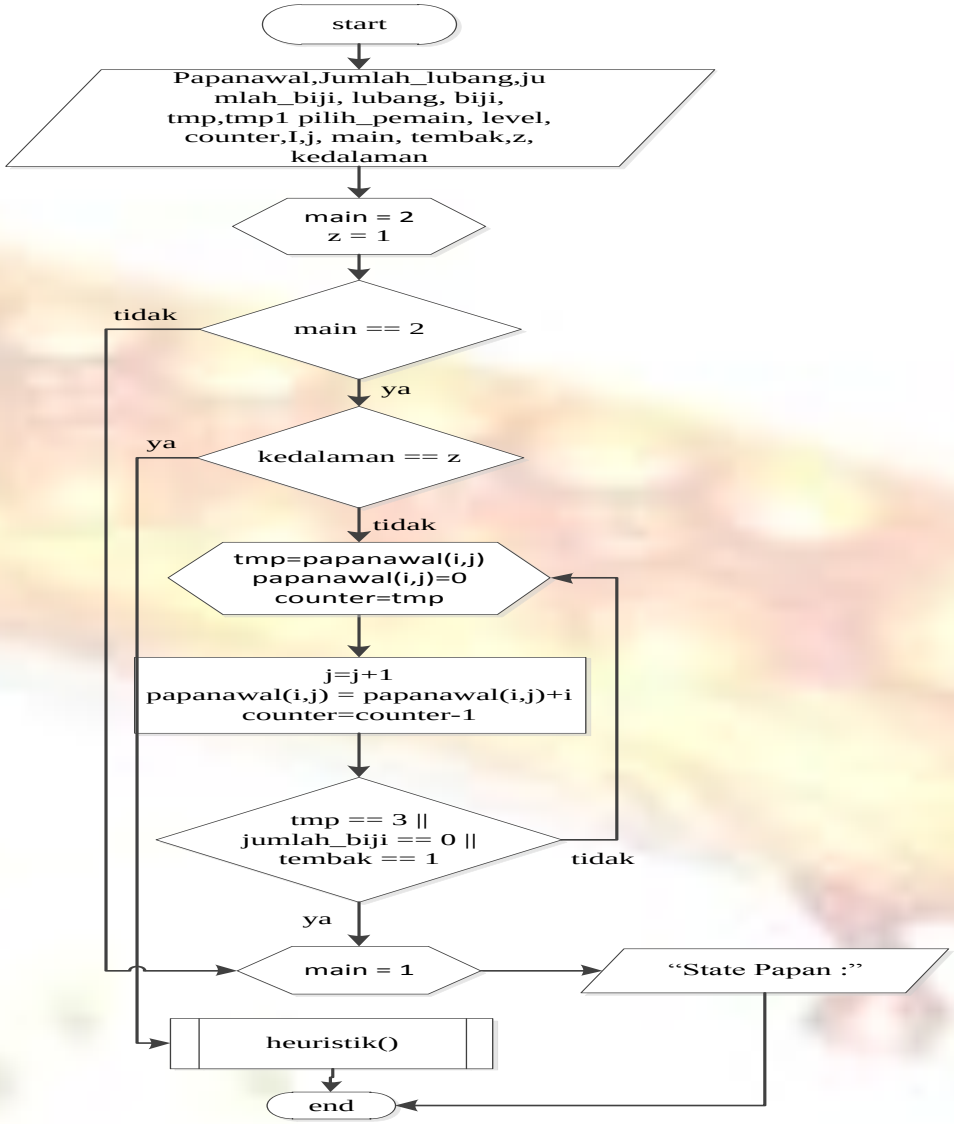
c. Pemain sudah mengambil biji sebanyak 3 kali

6. Akhir permainan, permainan dikatakan selesai jika tidak adalagi biji

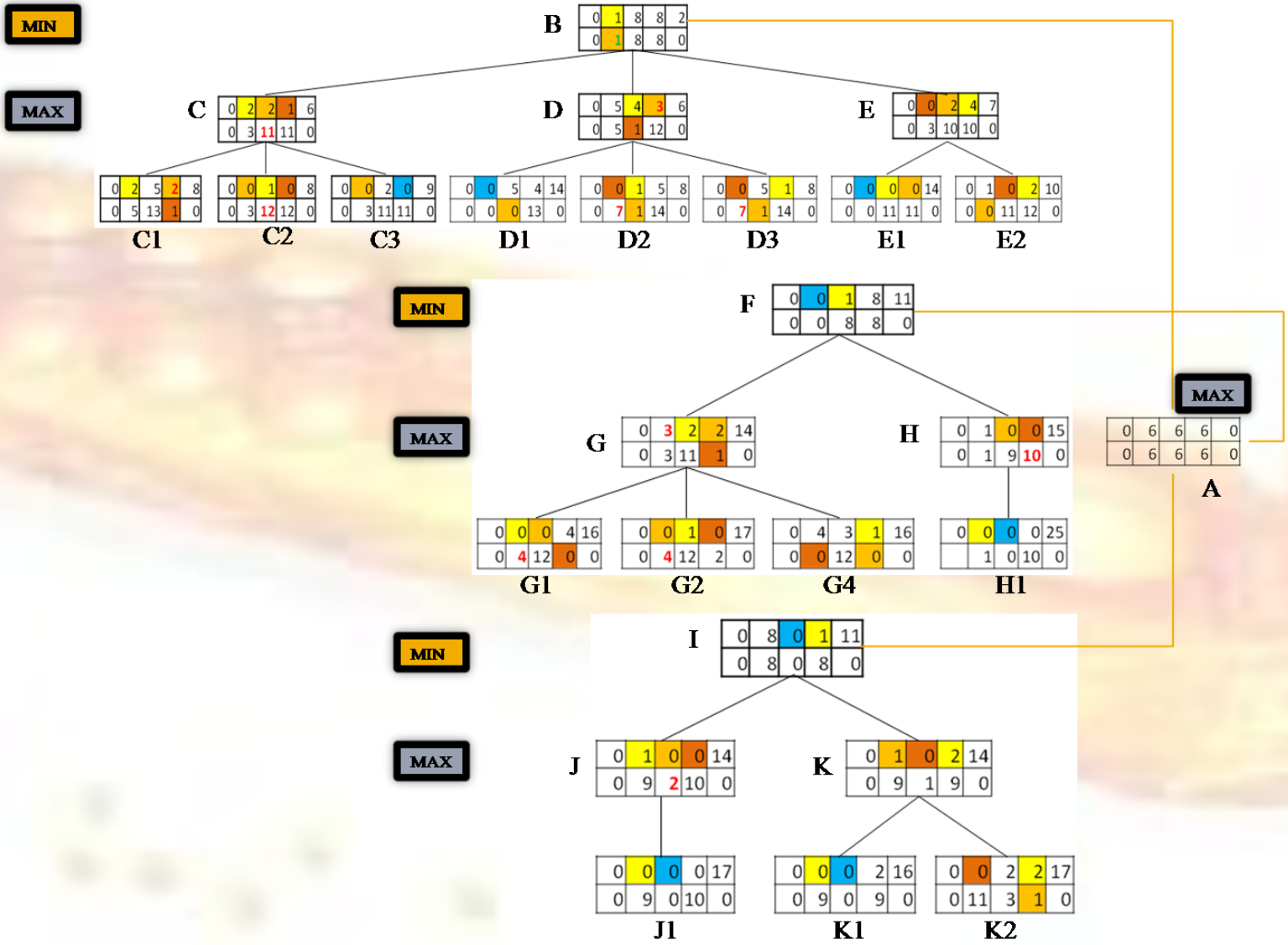
Tahapan Analisis Algoritma



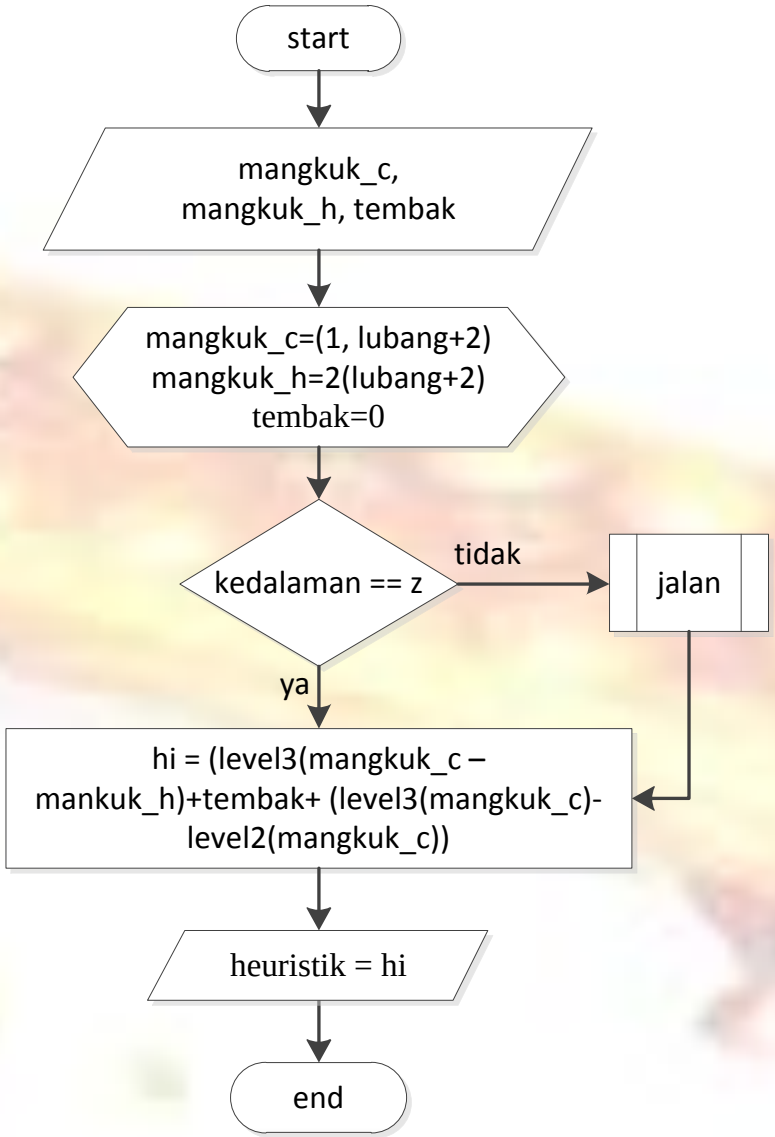
Pencarian Kemungkinan-kemungkinan langkah



Analisis Pohon Pencarian 3 lubang 6 Biji



Perhitungan Nilai Heuristik



Lanjutan Perhitungan Nilai Heuristik

Proses menghitung fungsi evaluasi heuristik dilakukan dengan melihat 3 faktor penting dalam congklak yaitu:

1. Jumlah biji yang terdapat dalam mangkuk pemain komputer dan lawan.
2. Kondisi kemungkinan tembak dalam satu state
3. Dan biji komputer sebelum permainan berjalan

Lanjutan Perhitungan Nilai Heuristik

Rumus fungsi evaluasi heuristik pada Minimax untuk permainan congklak mengacu pada 3 faktor di atas yang merupakan total dari beberapa heuristik, rumusnya yaitu sebagai berikut :

$$H = \sum_{i=1}^{n} hi$$

Lanjutan Perhitungan Nilai Heuristik

Berikut adalah rumus-rumus heuristik dari beberapa kondisi :

1. Total jumlah biji di mangkuk komputer

h1 = mangkuk_c – mangkuk_h

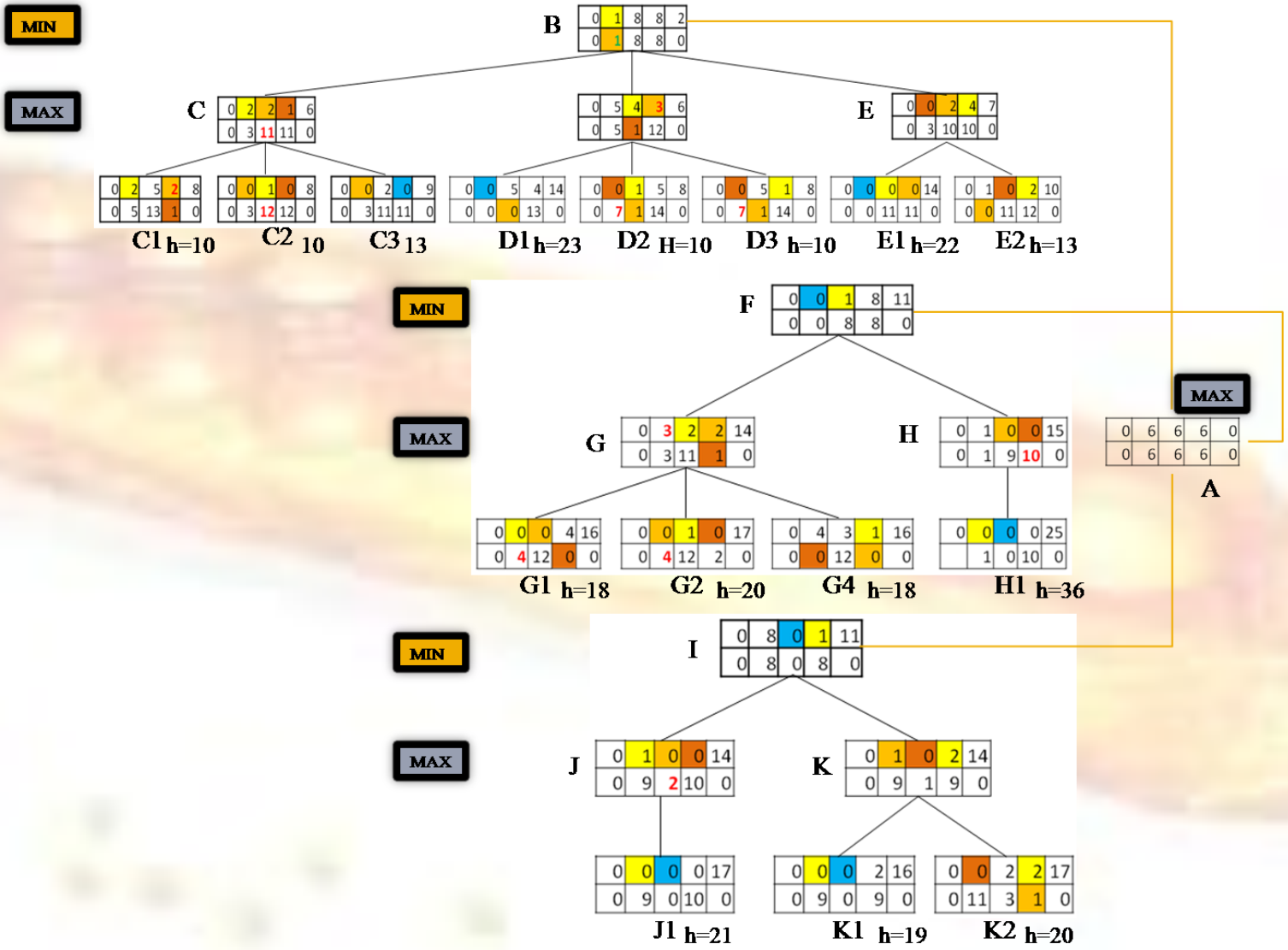
2. Kondisi Tembak

h2 = Kondisi tembak

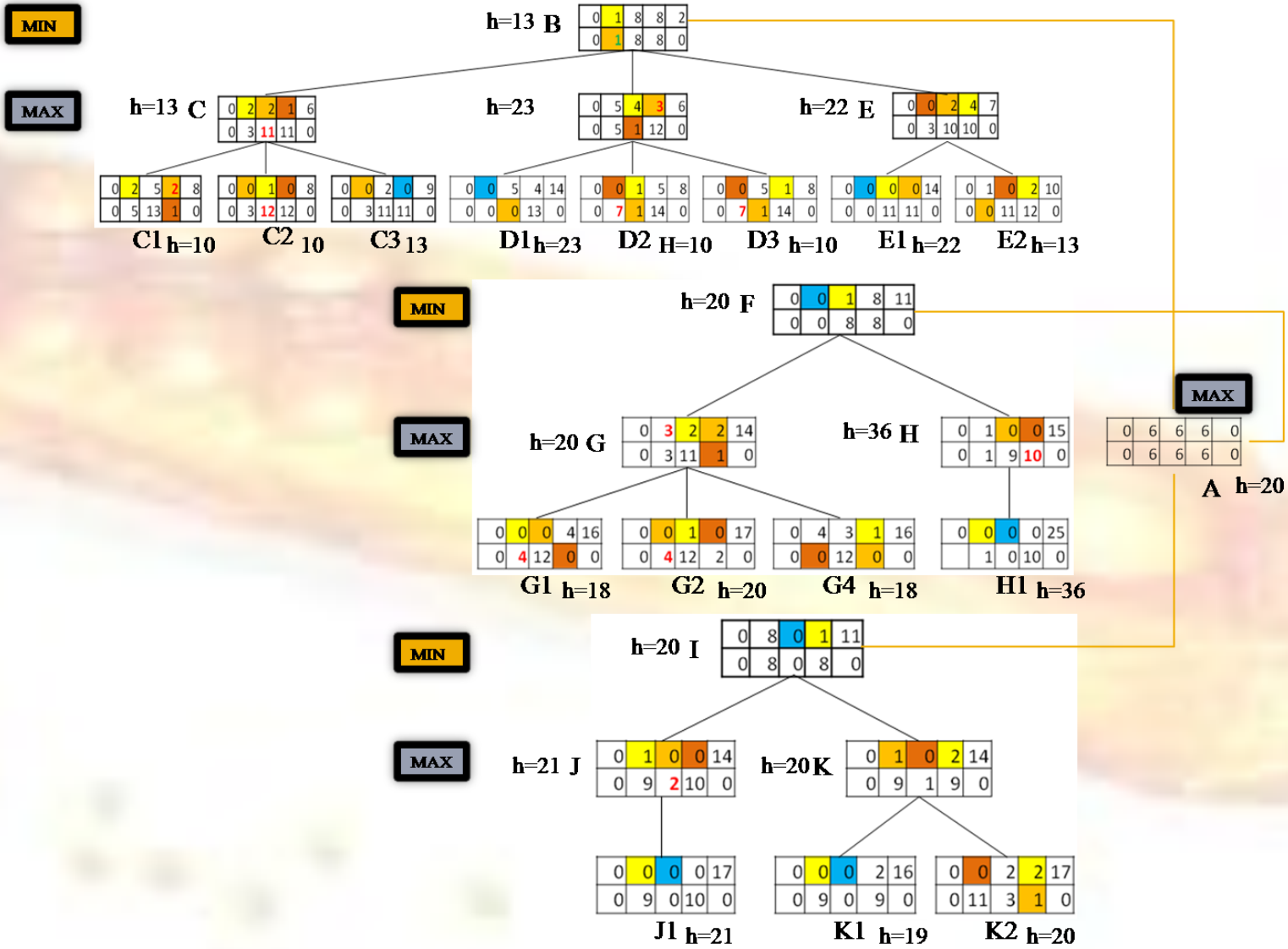
3. Jumlah mangkuk komputer pada level sebelumnya

h3 = levelhi(mangkuk_c – mangkuk_c)

Lanjutan Perhitungan Nilai Heuristik



Membandingkan Nilai Heuristik Minimax



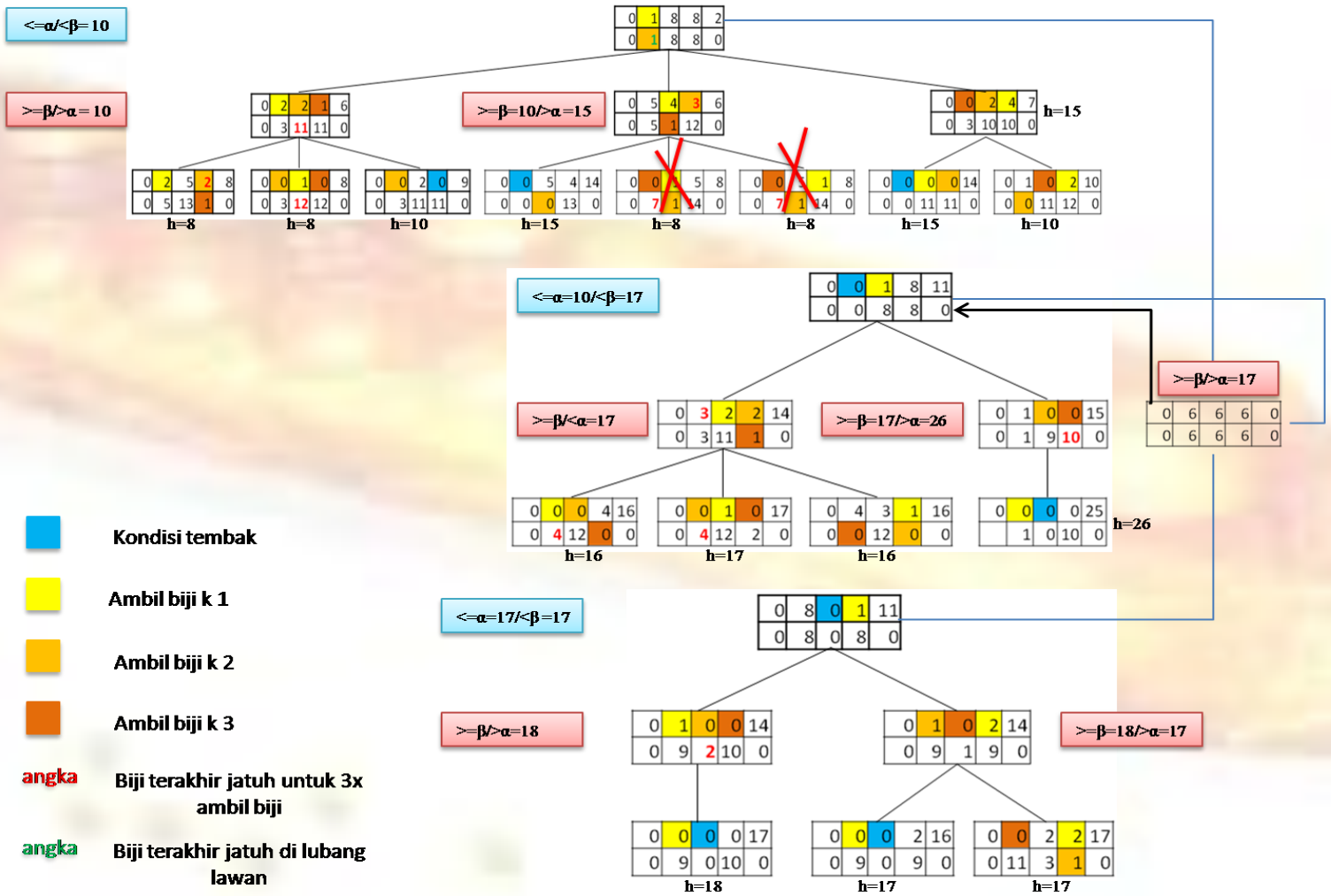
Hasil Permainan Congklak Algoritma *Minimax* 3 lubang 6 biji

No	Kasus Congklak Lubang 3 Biiji 6 pada Level Mudah							
	Pemain	State Matrik	∑ jumlah langkah congklak	∑ pengambilan biji	Total biji di mangkuk	∑ langkah algoritma	Nilai Heuristik
1	komputer	0 0 1 8 11 0 0 8 8 0	13	2x	11	50	20
	human	0 0 4 11 11 3 1 3 0	21	3x	3	-	-
2	komputer	0 0 0 12 17 3 0 0 4 0	6	2x	17	26	19
	human	0 1 0 12 17 4 1 1 0 0	4	1x	4	-	-
3	komputer	0 0 0 3 21 4 3 3 2 0	17	3x	21	14	24
	human	1 1 0 0 1 5 1 21 6 0	10	3x	6	-	-

Lanjutan Hasil Permainan Congklak Algoritma *Minimax* 3 lubang 6 biji

No	Kasus Congklak Lubang 3 Biiji 6 pada Level Mudah									
	Pemain	State Matrik					∑ jumlah langkah congklak	∑ pengambilan biji	Total biji di mangkuk	∑ langkah algoritma	Nilai Heuristik
4	komputer	0	1	0	0	23	1	1x	23	6	24
		6	1	5	0						
	human	0	2	1	1	23	6	2x	8	-	-
		8	1	0	0						
5	komputer	0	2	0	0	24	3	2x	24	12	19
		8	1	0	1						
	human	0	2	0	0	24	2	2x	10	-	-
		10	0	0	0						
6	komputer	0	0	1	0	25	2	1x	25	2	16
		10	0	0	0						
	human	-					-	-	-	-	-
7	komputer	0	0	0	0	26	1	1x	26	1	16
		10	0	0	0						

Membandingkan Nilai Heuristik Algoritma *Minimax* Optimasi *Alpha-Beta Pruning*



Hasil Permainan Congklak Algoritma Algoritma *Minimax* Optimasi *Alpha-Beta Pruning* 3 lubang 6 biji

No	Kasus Congklak Lubang 3 Biiji 6 pada Level Mudah								
	Pemain	State Matriks					∑ jumlah langkah congklak	∑ pengambilan biji	Total biji di mangkuk	∑ langkah algoritma	Nilai Heuristik
1	komputer	0	0	1	8	11	13	2x	11	46	20
		0	0	8	8	0					
	human	0	0	4	11	11	21	3x	3	-	-
		3	3	1	3	0					
2	komputer	0	0	0	12	17	6	2x	17	18	19
		3	0	0	4	0					
	human	0	1	0	12	17	4	1x	4	-	-
		4	1	1	0	0					
3	komputer	0	0	0	3	21	17	3x	21	14	24
		4	3	3	2	0					
	human	0	1	1	0	21	10	3x	6	-	-
		6	1	5	1	0					

Lanjutan Hasil Permainan Congklak Algoritma Algoritma *Minimax* Optimasi *Alpha-Beta Pruning* 3 lubang 6 biji

No	Kasus Congklak Lubang 3 Biiji 6 pada Level Mudah								
	Pemain	State Matriks	∑ jumlah langkah congklak	∑ pengambilan biji	Total biji di mangkuk	∑ langkah algoritma	Nilai Heuristik
4	komputer	0 1 0 0 23 6 1 5 0 0	1	1x	23	6	24
	human	0 2 1 1 23 8 1 0 0 0	6	2x	8	-	
5	komputer	0 2 0 0 24 8 1 0 1	3	2x	24	12	19
	human	0 2 0 0 24 10 0 0 0	2	2x	10	-	-
6	komputer	0 0 1 0 25 10 0 0 0	2	1x	25	2	16
	human	-	-	-	-	-	-
7	komputer	0 0 0 0 26 10 0 0 0	1	1x	26	1	16

Kompleksitas Waktu Algoritma T(n)

Dalam praktek, kompleksitas waktu dihitung berdasarkan jumlah operasi abstrak yang *mendasari* suatu algoritma, dan memisahkan analisisnya dari implementasi. Kompleksitas waktu yang dilakukan adalah pada algoritma *Minimax* dan *Minimax* dengan optimasi *Alpha-Beta Pruning*.

Pseudo-code Algoritma Minimax

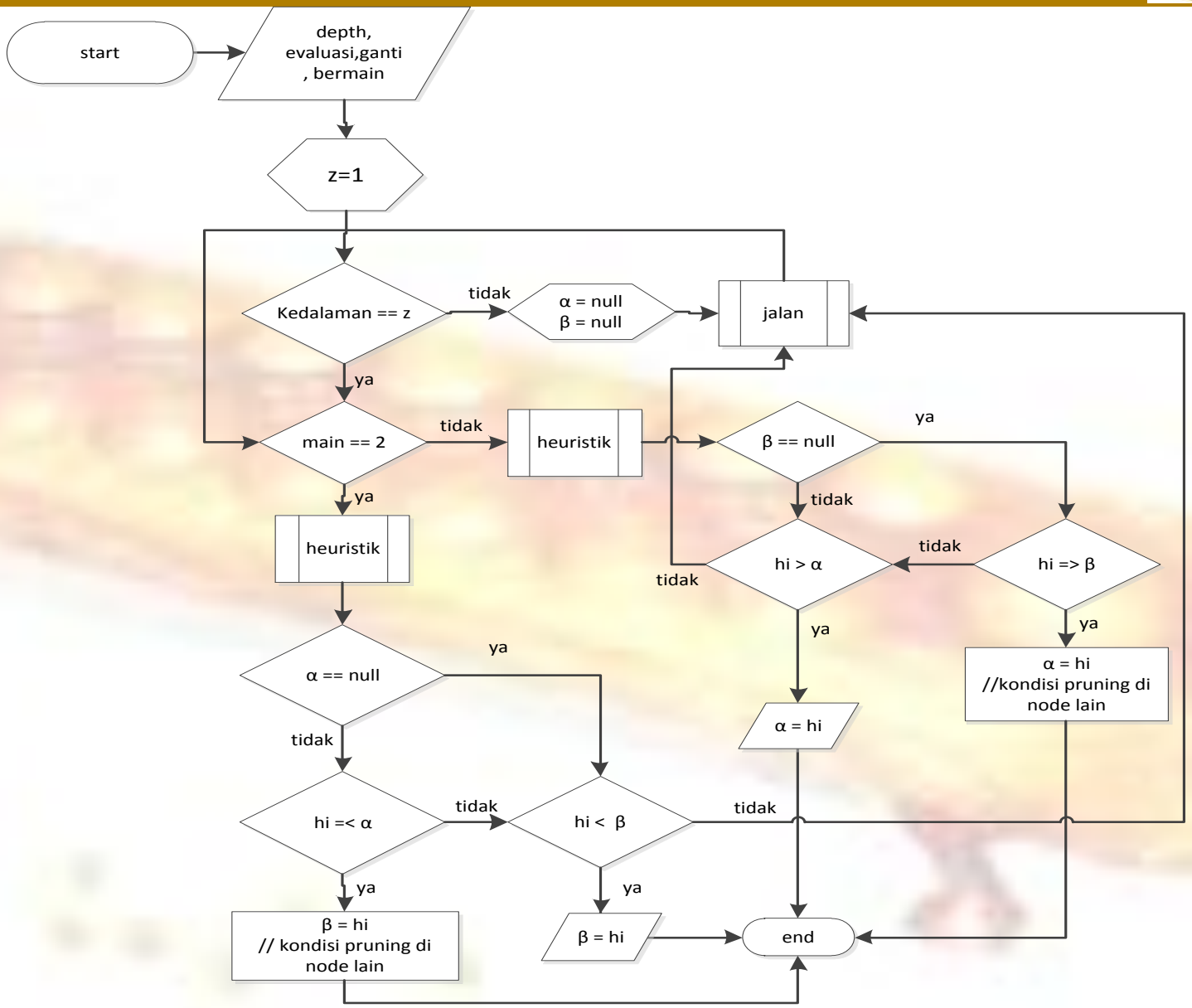
```
1. Function Minimax( input file_arr, endarr
2.                   output papan, chosen_score, chosen_move, ct, jmlkl)

   Deklarasi :
3.      a, child_awal, chosen_score : integer
4.      chosen_move, ct, tmp_score, tmp_move : integer

Algooritma :
5.   while N <= end_arr do                                   o(n)
6.   tmpScore ← -99999                                      o(1)
7.   for a=1 to child_awal do                               o(1)
8.        chosen_score, chosen_move                         o(1)
9.        if chosen_score > tmpScore then                   o(1)
10.            ct = ct + 1                                  o(1)
11.            tmpScore = chosen_score                      o(1)
12.            tmpMove = chosen_move                        o(1)
13.       endif
14. chosen_score = tmpScore                                 o(1)
15. chosen_move = tmpMove                                   o(1)
16. papan = papanawal;                                      o(1)
17. Endfor
18. Endwhile

                                                  Jumlah : O(n)
```

Flowchart Algoritma Minimax Optimasi Alpha-Beta Pruning



Pseudo-code Algoritma Minimax Optimasi Alpha-Beta Pruning

```
1. Function AlphaBeta(input file_arr, endarr
2.          output papan, chosen_score, chosen_move, ct, jmlkl)

Deklarasi :
3. parent, chosen_score, tmpCounterVisit, comp : integer
4.  chosen_move, ct, tmp_score, tmp_move, counterVisit : integer

Algoritma :
5. tmpScore = chosen_score;                              o(1)
6. tmpMove = chosen_move;                                o(1)
7. while (parent, 'awal.mat') == 0 do                    o(1)
8.    tmpCounterVisit = counterVisit                     o(1)
9.    if comp==1 then                                    o(1)
10.       if tmpScore > chosen_score then                 o(1)
11.          Pruning(next) // di pruning                  o(1)
12.          chosen_score = tmpScore                      o(1)
13.          chosen_move = tmpMove                        o(1)
14.          ct = ct + 1                                  o(1)
15.       endif
16.       else
17.       if tmpScore < chosen_score then                 o(1)
18.          Pruning(next) // di pruning                  o(1)
19.          chosen_score = tmpScore                      o(1)
20.          chosen_move = tmpMove                        o(1)
21.          ct = ct + 1                                  o(1)
22.       endif
23.      counterVisit = tmpCounterVisit                  o(1)
24.      tmpScore = chosen_score                         o(1)
25.      tmpMove = chosen_move                           o(1)
29.    endif
30. Endwhile
                                              Jumlah : O(n)
```

Lanjutan Kompleksitas Waktu T(n)

1. **Algoritma Minimax**

   Dari *pseudo-code* di atas operasi dasar dari *Minimax* adalah yang diberi tulisan warna merah dengan Kompleksitas *running time* ***T(n)=n***, karena operasi dasar diulang dari i=1 sampai panjang langkah.

2. **Algoritma *Minimax* optimasi Alpha Beta**

   Dari *pseudo-code* di atas operasi dasar dari *Minimax* Optimasi *Alpha-Beta pruning* adalah yang diberi tanda kotak merah dengan Kompleksitas *running time* ***T(n)=n-1***, Kompleksitas waktu algoritma dihitung berdasarkan jumlah operasi perbandingan *tmpscore* > *chosen score* dan *tmpscore* < *chosen score*

Perhitungan Notasi Asimptotik Big (*O*)

1. **Algoritma Minimax**

Dari *pseudo-code* di atas Notasi Asimptotiknya adalah *O*(n), karena pada pengulangan `while N <= end_arr do` tidak dapat ditentukan panjangnya maka kompleksitasnya diambil dari kompleksitas waktu terburuk **(Tmax(n))** atau ***O(n)***

2. **Algoritma *Minimax* optimasi Alpha Beta**

Dari *pseudo-code* di atas Notasi Asimptotiknya :

$= O(1) + O(1) + (n\text{-}1)\ \{\ O(1) + O(1) + O(1) + O(1) + O(1) + O(1) +$
$\quad O(1) + O(1) + O(1) + O(1) + O(1) + O(1) + O(1) + O(1) + O(1) + O(1)$
$= O(1) + (n\text{-}1)\ O(1)$
$= O(1) + O(n\text{-}1)$
$= O(1) + O(n)$
$= \mathbf{O(n)}$

Kesimpulan Kompleksitas Waktu dan Big *(O)*

Hasil dari analisis Big *O* untuk *Minimax* dan *Minimax* optimasi Alpha-Beta Pruning yang sudah digambarkan dan dihitung menggunakan Big *O* dengan notasi asimptotik pada penjelasan sebelumnya bahwa Big *O* *Minimax* termasuk ke dalam *time complexity function* "n" atau Big *O* nya adalah Big *O*(n) dan untuk *Minimax* optimasi Alpha-Beta Pruning termasuk ke dalam *time complexity function* "n" atau Big *O* nya adalah Big *O*(n), artinya algoritma tersebut termasuk ke dalam *time complexity function* "n", sehingga hasil dari perbedaan waktu antara *Minimax* dan *Minimax* optimasi Alpha-Beta Pruning tidak akan terlalu berbeda jauh atau signifikan. Kelompok Algoritma untuk *Minimax* dan *Minimax Alpha-Beta pruning* adalah sebagai berikut.

Lanjutan Kesimpulan Kompleksitas Waktu dan Big *(O)*

Algoritma	Kelompok Algoritma	Nama
Minimax	*O*(n)	Lanjar
Minimax Optimasi *Alpha-Beta Pruning*		

Pengujian

Pengujian adalah untuk mengetahui, apakah algoritma *Minimax* untuk mendapatkan solusi pada congklak berhasil atau tidak, serta apakah algoritma *Alpha-Beta Pruning* dapat menghemat waktu pencarian atau tidak

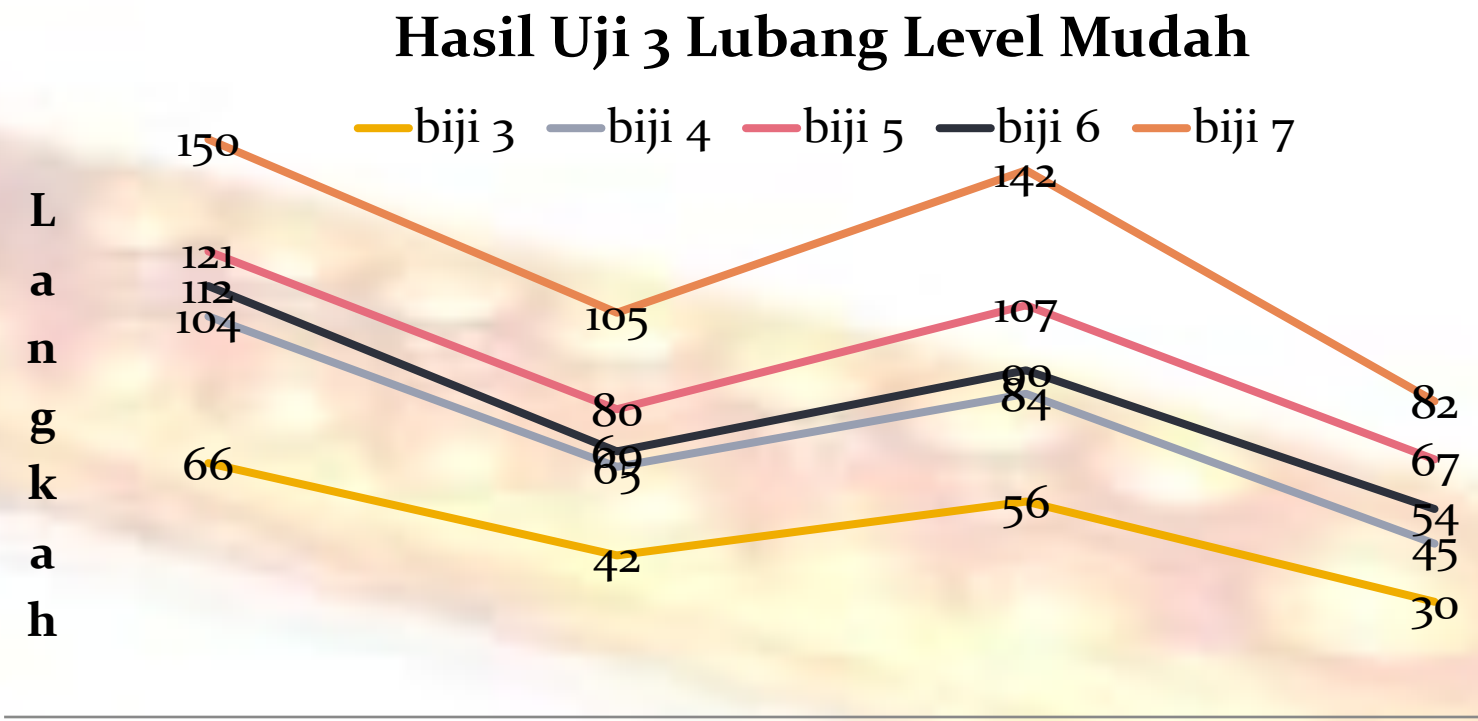
Untuk mencapai tujuan tersebut, dilakukan tahapan-tahapan sebagai berikut :

1. Pengujian hasil *sample* dengan pohon pencarian.
2. Pengujian waktu pada *Minimax* dan *Minimax* yang sudah dioptimasi dengan *Alpha-Beta Pruning*

Hasil Uji pada 3 Lubang pada Level Mudah

No	Biji	Minimax		Minimax Alpha-Beta Pruning	
		∑Langkah	**∑Langkah Dasar**	**∑Langkah**	**∑Langkah Dasar**
1	3	66	42	56	30
2	4	104	65	84	45
3	5	121	80	107	67
4	6	112	69	90	54
5	7	150	105	142	82

Hasil Uji pada 3 Lubang pada Level Mudah



(a) ∑Langkah Minimax

(b) ∑Langkah Dasar Minimax

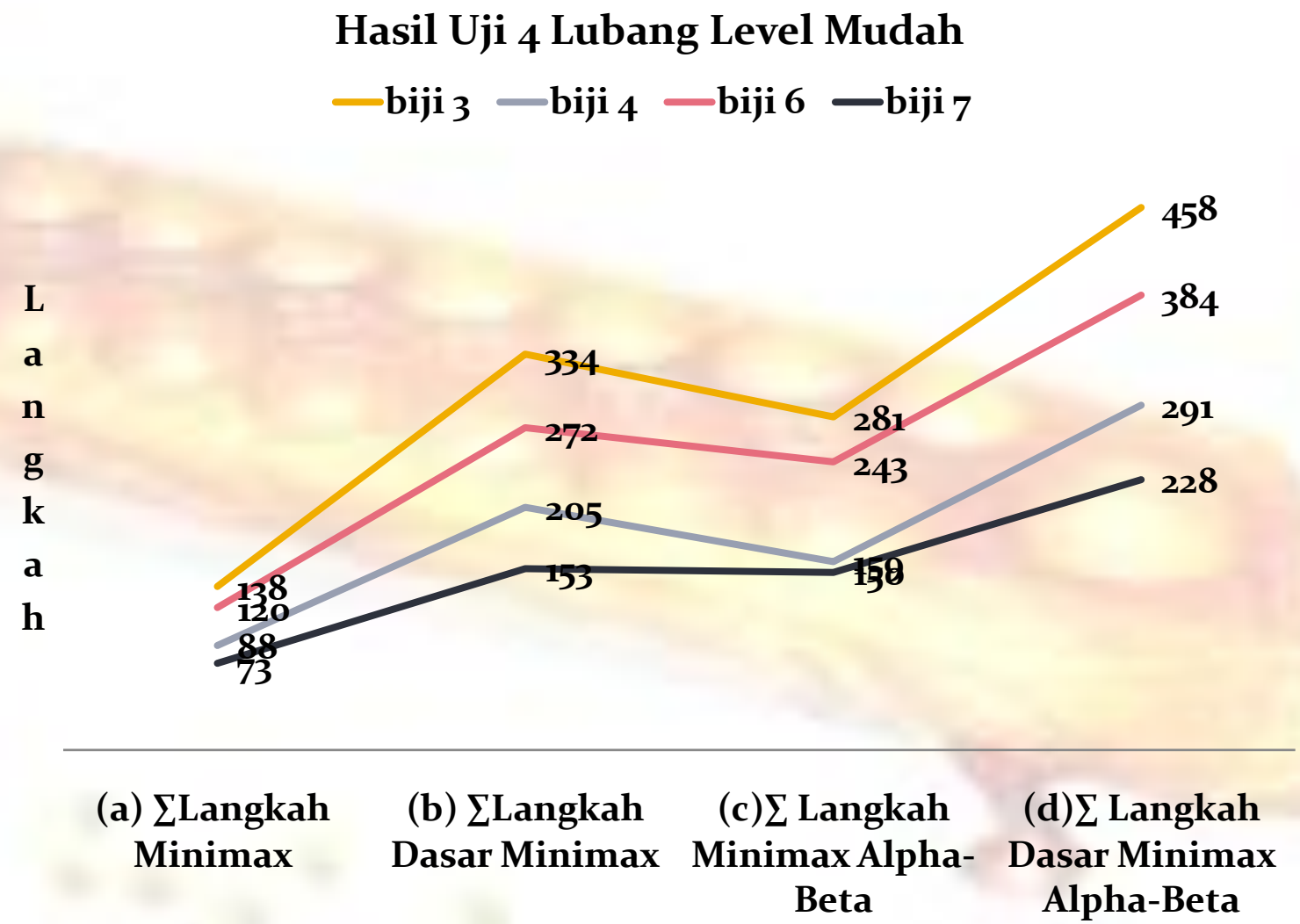
(c)∑ Langkah Minimax Alpha-Beta

(d)∑ Langkah Dasar Minimax Alpha-Beta

Hasil Uji pada 4 Lubang pada Level Mudah

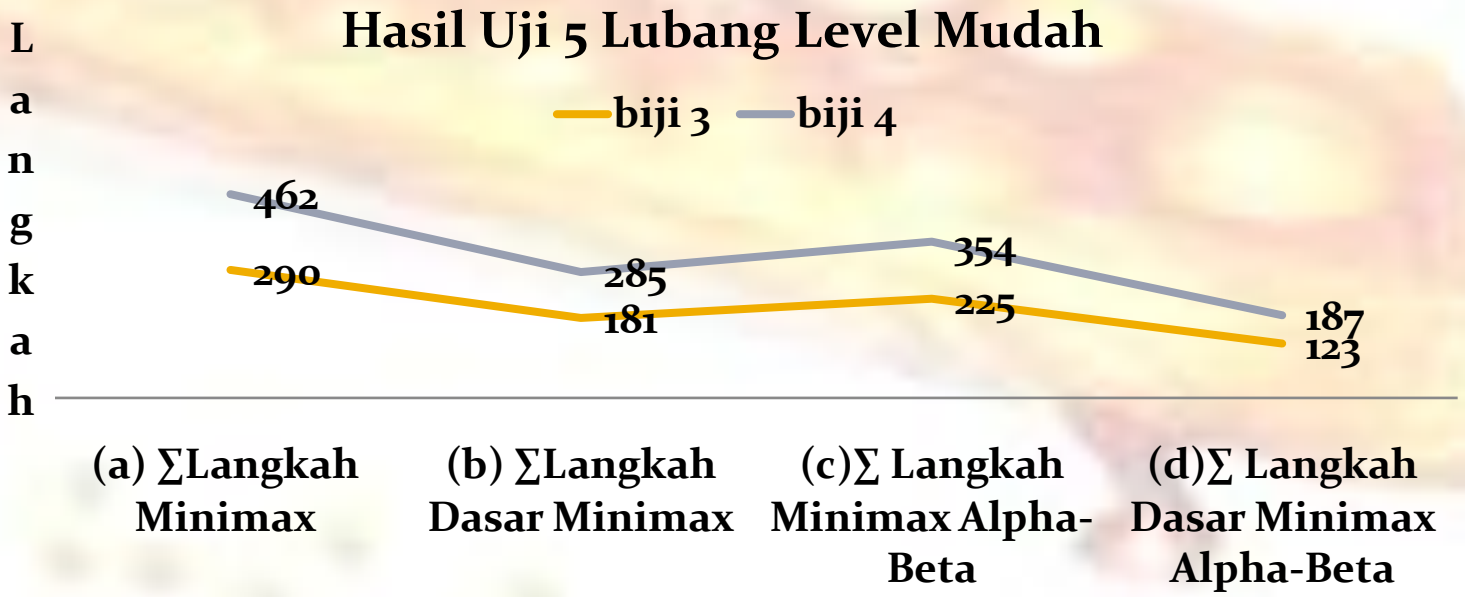
No	Biji	Minimax		Minimax Alpha-Beta Pruning	
		∑Langkah	**∑Langkah Dasar**	**∑Langkah**	**∑Langkah Dasar**
1	3	138	88	120	73
2	4	334	205	272	153
4	6	281	159	243	150
5	7	458	291	384	228

Hasil Uji pada 4 Lubang pada Level Mudah



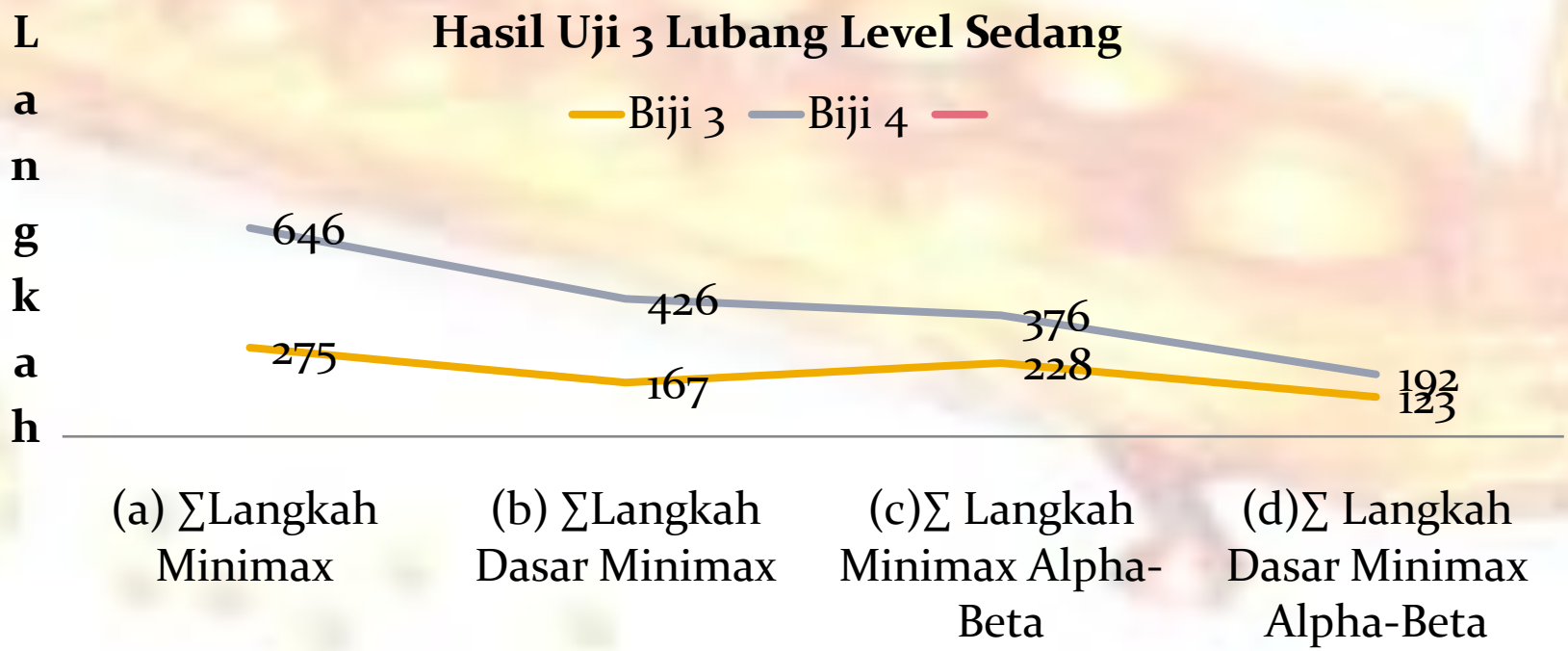
Hasil Uji pada 5 Lubang pada Level Mudah

No	Biji	Minimax		Minimax Alpha-Beta Pruning	
		∑Langkah	**∑Langkah Dasar**	**∑Langkah**	**∑Langkah Dasar**
1	3	290	181	225	123
2	4	462	285	354	187

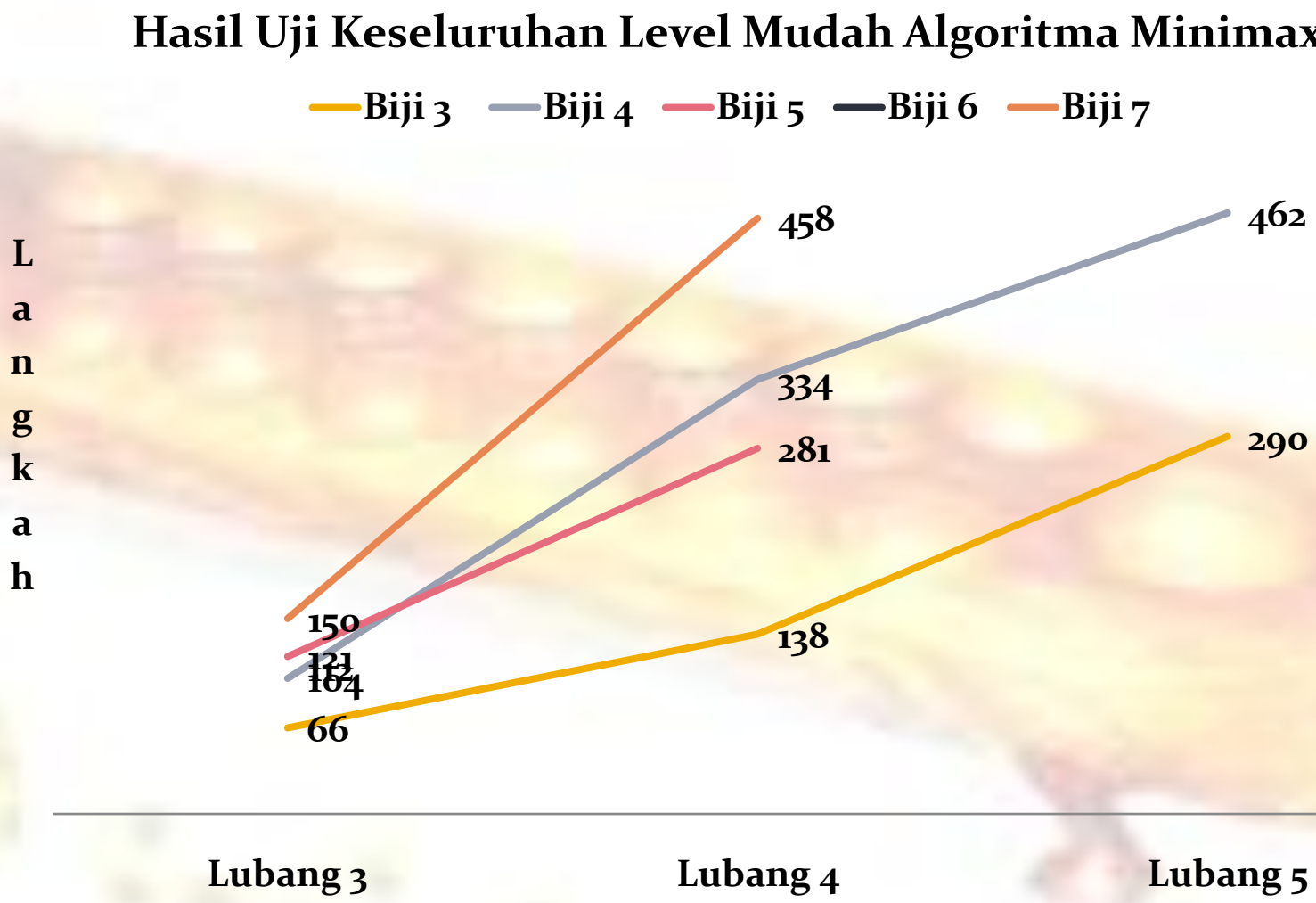


Hasil Uji pada 3 Lubang pada Level Sedang

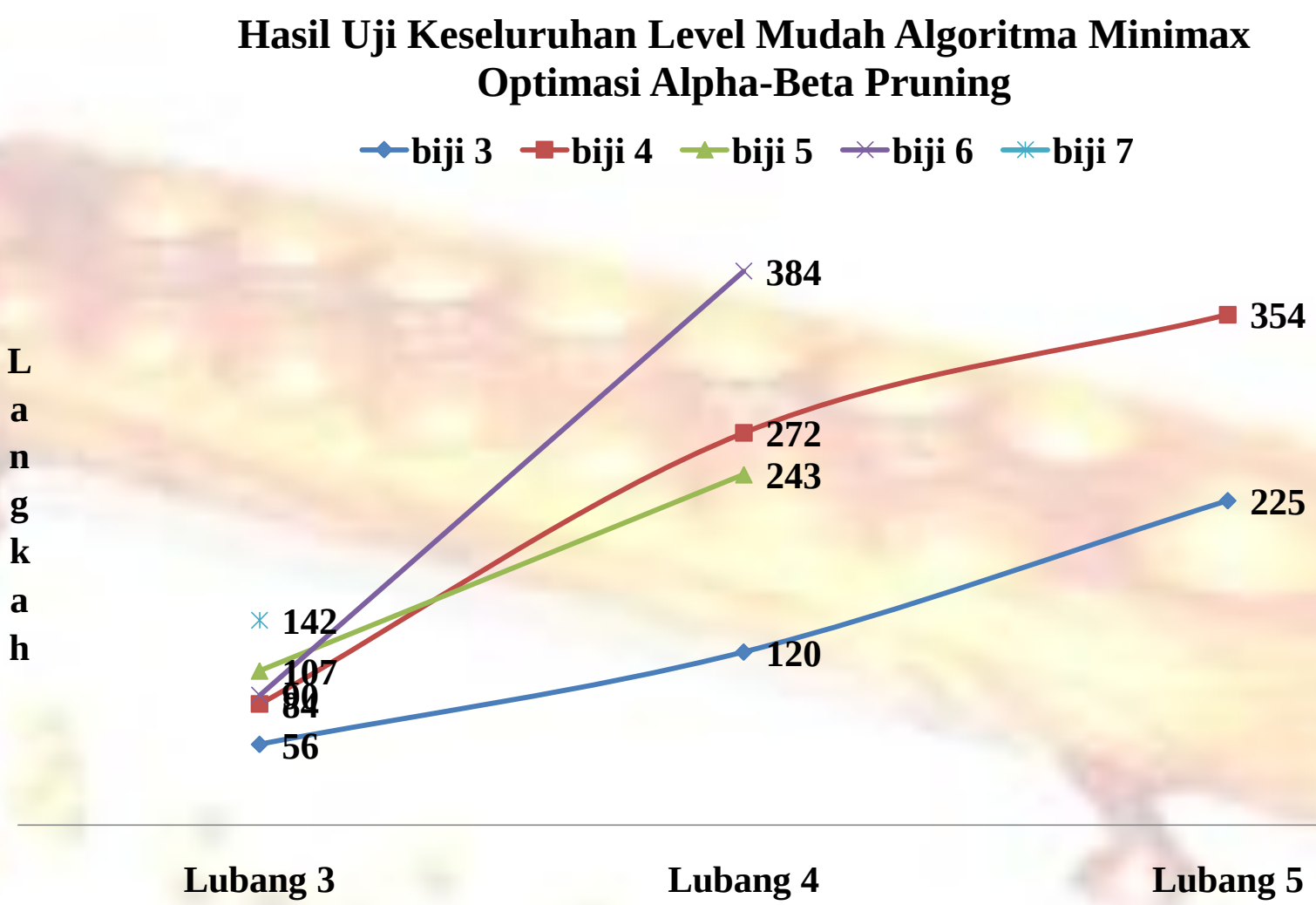
No	Biji	Minimax		Minimax Alpha-Beta Pruning	
		∑Langkah	**∑Langkah Dasar**	**∑Langkah**	**∑Langkah Dasar**
1	3	275	167	228	123
2	4	646	426	376	192



Hasil Uji Keseluruhan Level Mudah Minimax



Hasil Uji Keseluruhan pada Level Mudah Minimax Optimasi Alpha-Beta Pruning



Kesimpulan Pengujian

1. Dari hasil pengujian yang dilakukan terhadap 25 sampel untuk setiap lubang dan biji, diperoleh bahwa langkah yang ditelusuri *Minimax* adalah lebih banyak dari *Alpha-Beta Pruning*, pada hasil uji di atas, langkah paling banyak di telusuri pada Algoritma *Minimax* adalah *lubang dengan jumlah 5* pada *biji 4* dan pada jumlah lubang *4 pada biji 7*, yaitu "462" dan "458" yang terdapat pada tabel 4.10 dan tabel 4.13, sedangkan pada Algoritma *Alpha-Beta pruning* adalah "384" dan "354" dalam level mudah, sedangkan pada level sedang pada lubang 4 dengan biji 5 berjumlah "646" dengan begitu dengan semakin banyak jumlah lubang, biji, dan level maka akan semakin banyak penelusuran langkah yang dibutuhkan oleh komputer

2. Dari langkah tersebut terlihat perbeedaan dari *Minimax dan Alpha-Beta Pruning*, bahwa *Alpha-Beta Pruning* lebih sedikit penelusuran langkahnya yaitu rata-rata "30%" dari *Minimax*. Penelitian ini terbukti bahwa *Minimax yang di optimasi Alpha-Beta Pruning* lebih cepat dari pada *Minimax* tanpa Optimasi.

Kesimpulan

Algoritma Minimax optimasi Alpha-Beta yang dianalisis dan di uji coba pada BAB IV telah membuktikan bahwa algoritma Minimax dalam penelusuran langkahnya berbeda dalam jumlah langkahnya dengan Minimax optimasi Alpha-beta pruning, dengan semakin banyaknya lubang maka pencarian kemungkinan-kemungkinan langkah akan semakin banyak sehingga akan memakan waktu lama, Dari hasil pengujian yang dilakukan terhadap 25 sampel untuk setiap lubang dan biji, dengan begitu dengan semakin banyak jumlah lubang dan biji maka akan semakin banyak penelusuran langkah yang dibutuhkan oleh komputer dan dari langkah tersebut terlihat perbeedaan dari *Minimax dan Alpha-Beta Pruning* bahwa *Alpha-Beta Pruning* lebih sedikit penelusuran langkahnya yaitu sebesar "30%" dari *Minimax*. Penelitian ini terbukti bahwa *Minimax yang di optimasi Alpha-Beta Pruning* lebih cepat dari pada *Minimax* tanpa Optimasi.

Saran

Dalam analisis Algoritma penelitian ini masih jauh dari sempurna dan masih banyak kekurangan. Oleh karena itu perlu dilakukan analisis yang lebih dalam. Adapun sarannya, yaitu Untuk penelitian selanjutnya akan lebih baik jika menggunakan algoritma yang lebih cepat dari Minimax atau Alpha-beta pruning agar permainan komputer lebih handal dan cepat.



TERIMA KASIH