Penerapan Algoritma DFS pada Permainan Othello

Anasthasia Amelia / 13510093
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13510093@std.stei.itb.ac.id

***Abstraksi*—Othello merupakan salah satu permainan papan yang banyak digemari orang dari berbagai usia. Perkembangan teknologi membawa permainan ini ke dalam dunia virtual. Hal tersebut mengakibatkan munculnya kebutuhkan artificial intelligence dalam menangani pilihan mana yang akan diambil oleh pemain tertentu. Salah satu algoritma pencarian adalah algoritma depth-first search yang melakukan pencarian secara mendalam dengan menggunakan struktur pohon dinamis. Dalam pembangunan pohon dinamis tersebut, dibuat keputusan pengambilan langkah dengan menggunakan bantuan algoritma Minimax. Algoritma Minimax merupakan algoritma berbasiskan depth-first search yang cocok diaplikasikan pada permainan yang terdiri dari dua orang. Algoritma ini mencari nilai maksimum dan minimum dari setiap simpul yang ada.**

***Kata kunci*—simpul, *depth-first*, max, min.**

I. PENDAHULUAN

Permainan papan (board games) merupakan salah satu jenis permainan yang diminati banyak orang hingga saat ini. Pemain permainan papan tidak hanya berasal dari kalangan anak-anak saja, tetapi juga orang dewasa. Hal tersebut dikarenakan alur permainan yang sederhana dan mudah dimengerti.

Mengapa permainan ini disebut permainan papan? Ya, karena permainan ini dimainkan di atas sebuah papan. Ada banyak sekali macam permainan papan. Beberapa di antaranya adalah catur, *backgammon*, *ludo*, *scrabble*, dan masih banyak lagi yang lainnya. Pada makalah ini, akan dibahas permainan papan bernama *Othello*.

Dengan berkembangnya teknologi, permainan papan sekarang tidak hanya dapat dimainkan di atas papan saja, tetapi dapat dimainkan di berbagai perangkat keras seperti komputer, telepon genggam, ataupun *tablet*. Sudah banyak pengembang permainan yang membuat perangkat lunak permainan sehingga dapat dengan mudah dibawa dan dimainkan kapan saja. Dengan adanya perangkat lunak ini, pemain dapat bermain sendiri melawan mesin permainan tersebut. Mesin ini akan melawan pemain selayaknya manusia yang memainkannya.

Pertanyaannya adalah, bagaimana mesin tersebut dapat berpikir dan menentukan jalan permainan? Mesin tersebut menggunakan *artificial intelligence* untuk menjalankan permainan. Banyak algoritma yang digunakan untuk menyelesaikan masalah tersebut. Dalam makalah ini, akan dibahas peneraman algoritma *depth-first search* untuk membangun pohon status permainan yang membantu komputer menentukan pilihan langkah dalam permainan papan *Othello*.

II. OTHELLO

Othello merupakan sebuah permainan yang ditemukan oleh Goro Hasegawa pada tahun 1971. Beliau meminta bantuan James R. Becker untuk mengembangkan dan memasarkan game ini. Pada tahun 1973, sebuah perusahaan *game* Jepang, Tsukuda Original, meregistrasikan *game* ini dengan nama Othello. Nama Othello sendiri diambil dari sebuah drama Shakespeare, yaitu "*Othello, the Moor of Venice*" yang menceritakan kisah antara Othello yang berkulit hitam dan Desdemona yang berkulit putih.

Permainan Othello yang sekarang banyak dimainkan menggunakan aturan yang dikeluarkan oleh Pressman Toy Corporation, Amerika Utara. Tujuan dari permainan ini adalah untuk memiliki jumlah terbanyak disk warna tertentu, hitam atau putih, pada akhir permainan.

A. Gambaran Permainan

Dalam sebuah set permainan Othello, terdapat papan permainan othello dan 64 disk dengan 2 sisi berwarna hitam dan putih. Permainan Othello dimainkan oleh 2 orang. Setiap pemain mendapatkan 32 buah disk dan memilih warna yang diinginkan, hitam atau putih. Pada awal permainan, pada papan diletakkan 4 buah disk, 2 hitam dan 2 putih seperti pada gambar di bawah ini.

Gambar 1 – kondisi awal permainan Othello

Yang harus dilakukan pemain adalah meletakkan sebuah disk pada papan sehingga barisan milik lawan dibatasi oleh disk milik pemain. Kemudian membalikkan semua disk milik lawan yang telah dibatasi dengan disk milik pemain sehingga menjadi nilai tambah bagi pemain tersebut. Langkah ini disebut dengan "*outflank*".

B. Peraturan Othello

Berikut ini adalah peraturan permainan Othello yang dirilis oleh Pressman Toy Corporation:

1. Hitam selalu mendapat giliran pertama.
2. Jika dalam sebuah *turn* (giliran) pemain tidak menemukan *move* (langkah), dengan kata lain pemain tidak dapat melakukan "*outflank*", giliran akan diberikan kepada lawan. Jika ada langkah yang tersedia, pemain tidak boleh memberikan gilirannya pada lawan.
3. Sebuah disk dapat membalikkan banyak disk dengan berbagai arah, baik secara horizontal, vertikal, maupun diagonal.
4. Pemain tidak boleh melewati warnanya sendiri saat membalikkan disk.
5. Disk hanya dapat dibalik pada jalurnya sesuai dengan tempat di mana disk pembatas diletakkan.
6. Semua disk yang terbatasi (*outflanked*) harus dibalik, walaupun ada kasus yang menguntungkan pemain untuk tidak membalikkan semuanya.
7. Jika pemain salah membalikkan sebuah disk, pemain diperbolehkan untuk membalikkannya seperti semula selama lawan belum membuat langkah baru.
8. Jika sebuah disk telah diletakan pada sebuah kotak, disk tersebut tidak dapat dipindahkan ke kotak lainnya pada langkah-langkah selanjutnya.
9. Jika pemain kehabisan disk tetapi masih punya kesempatan untuk membalikkan disk lawan, maka lawan harus memberikan sebuah disk kepada pemain.
10. Jika sudah tidak ada langkah yang mungkin untuk kedua pemain, maka permainan telah berakhir. Jumlah disk untuk setiap warna dihitung dan pemain dengan jumlah tertinggi adalah pemenangnya.
11. Permainan mungkin berakhir walaupun keenampuluhempat kotak belum terisi penuh.

III. ALGORITMA PENCARIAN MENDALAM (DFS)

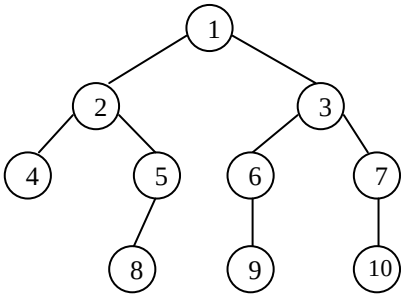
Algoritma pencarian mendalam (DFS) merupakan salah satu algoritma traversal dalam graf. Algoritma traversal dalam graf bekerja dengan cara menelusuri simpul-simpul yang ada secara sistematis.

Berbeda dengan algoritma BFS yang menelususi seluruh simpul dalam graf secara melebar, algoritma DFS menelusuri simpul-simpul dalam graf secara mendalam. Jadi tidak semua simpul dalam graf ditelusuri, hanya simpul-simpul yang menuju solusi yang dibangkitkan. Hal ini menyebabkan berkurangnya simpul yang harus diperiksa sehingga penelusuran graf dengan menggunakan algoritma DFS menjadi lebih efektif dan cepat.

Penelusuran graf *G* dengan menggunakan algoritma DFS adalah sebagai berikut:

1. Penelusuran (traversal) dimulai dari simpul *s*.
2. Mengunjungi simpul *t* yang merupakan anak dari simpul *s* dan bertetangga dengan simpul *u*.
3. Penelusuran mendalam dimulai lagi secara rekursif dari simpul *t*.
4. Penelusuran mendalam dilakukan hingga mencapai simpul *v*, sesuai dengan kedalaman maksimum pohon ruang status yang telah ditentukan.
5. Penelusuran dirunut-balik ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai simpul *t* yang belum dikunjungi.
6. Penelusuran dihentikan bila tidak ada lagi simpul yang belum dikunjungi, yang dapat dicapai dari simpul yang telah dikunjungi.

Berikut ini adalah ilustrasi sebuah graf G dengan penelusuran menggunakan algoritma DFS:



Gambar 2 – graf G

Dari gambar di atas, urutan simpul yang dikunjungi bila dilakukan penelusuran dengan algoritma DFS adalah: 1,2,4,5,8,3,6,9,7,10.

Dalam mencari solusi sebuah persoalan, digunakan sebuah struktur untuk memudahkan pencarian. Struktur yang sering digunakan adalah struktur pohon berakar. Pencarian solusi ini dilakukan dengan menelusuri simpul-simpul yang dibangun pada pohon tersebut. Bila belum menemukan solusi, maka simpul berikutnya diperiksa.

Pembentukan pohon pencarian solusi ini pada umumnya dilakukan secara dinamis. Pembangunan pohon dilakukan bersamaan dengan pencarian solusi. Apabila simpul yang dibangun tidak mengarah ke solusi, maka simpul berikutnya dibangun dan ditelusuri hingga solusi ditemukan.

Pohon ruang status merupakan pohon dinamis yang dibangun saat pencarian solusi. Beberapa komponen yang ada dalam pohon ruang status adalah:

- Status persoalan: simpul-simpul dalam pohon yang memenuhi *constraints*.
- Status solusi: pernyataan untuk solusi persoalan (daun).
- Status tujuan: status solusi yang merupakan simpul daun.
- Ruang solusi: himpunan semua status solusi.
- Ruang status: seluruh simpul dalam pohon dinamis.

- Status awal: akar pada phon ruang status.

Pada kebanyakan persoalan, pohon ruang status mungkin tidak berhingga kedalamannya. Oleh karena itu, biasanya ditentukan batas maksimum kedalaman pohon yang diperbolehkan. Simpul yang mempunyai kedalaman maksimum tersebut diperlakukan seolah-olah tidak memiliki anak lagi (tidak dibangun simpul baru).

Pada kasus terburuknya, DFS harus membangkitkan semua simpul pada pohon ruang status untuk menemukan solusi. Jika setiap simpul membangkitkan *b* buah simpul baru, dan batas maksimum kedalaman pohon ruang status adalah *m*, maka kompleksitas waktu algoritma DFS pada kasus terburuk adalah $O(b^m)$. Sedangkan kompleksitas ruang algoritma DFS adalah $O(bm)$ karena hanya perlu menyimpan satu lintasan tunggal dari akar sampai daun, ditambah simpul yang belum dikembangkan.

IV. MENENTUKAN LANGKAH TERBAIK

Dalam sebuah permainan yang diikuti oleh 2 orang pemain (2 *players*), seorang pemain tentunya menginginkan langkah terbaik agar dapat memenangkan permainan. Untuk menentukan langkah terbaik tersebut, pemain tidak dapat hanya dengan melihat kondisi saat itu. Pemain juga harus dapat memperkirakan langkah apa yang akan diambil oleh lawan jika pemain mengambil suatu langkah. Dengan asumsi bahwa lawan akan mengambil langkah yang mengakibatkan langkah pemain selanjutnya menjadi sulit atau buruk. Selain itu, pemain juga harus memperkirakan langkah apa yang dapat diambilnya setelah lawan mengambil langkahnya. Pemain pun harus menentukan apakah langkahnya di masa depan tersebut merupakan langkah yang terbaik yang dapat menghasilkan skor tertinggi. Dengan berbagai perkiraan tersebut, pemain dapat menentukan langkah apa yang akan diambilnya.

A. Algoritma Minimax

Algoritma Minimax merupakan algoritma pencarian yang berbasiskan algoritma DFS dalam membangun pohon dinamisnya. Algoritma ini merupakan salah satu cara dalam menentukan langkah optimal dalam permainan yang terdiri dari dua pemain. Di dalam pohon pencarian yang dibangun, terdapat dua macam simpul. Kedua simpul tersebut adalah simpul yang merepresentasikan langkah pemain dan simpul yang merepresentasikan langkah yang diambil oleh lawan.

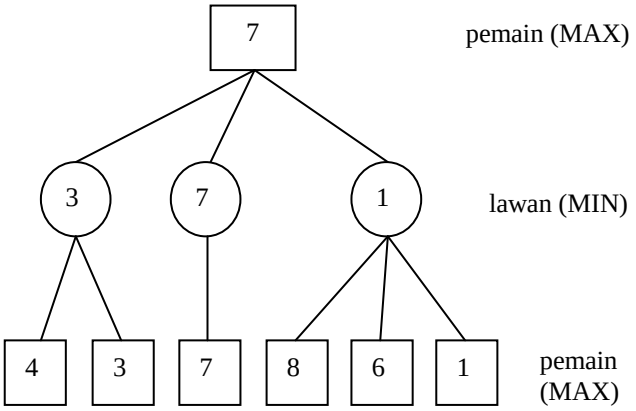
Seperti pada bahasan sebelumnya, untuk memperkecil ruang pencarian dalam algoritma DFS, perlu ditentukan batas maksimum kedalaman pohon ruang status. Dalam pengaplikasian algoritma DFS pada algoritma Minimax, batas kedalaman maksimum pohon ruang status yang ditentukan adalah 4. Masing-masing kedalaman tersebut adalah:

- Akar, yaitu kondisi dari permainan saat itu.
- *Move*: langkah yang mungkin diambil pemain.
- *Opponent's move*: langkah yang mungkin diambil oleh lawan setelah pemain mengambil langkah.
- *Player's next move*: langkah berikutnya yang mungkin diambil pemain setelah lawan mengambil langkah berikutnya.

Pada algoritma Minimax, terdapat nilai minimum (MIN) dan maksimum (MAX). Simpul maksimum adalah simpul yang memilih simpul anaknya dengan nilai terbesar, sehingga menjadi nilai dari simpul maksimum tersebut (simpul MAX). Sedangkan simpul minimum adalah simpul yang memilih simpul anak dengan nilai terkecil, yang kemudian menjadi nilai dari simpul minimum tersebut (simpul MIN).

Simpul MAX merupakan simpul yang merepresentasikan langkah terbaik yang dapat diambil oleh seorang pemain. Sedangkan simpul MIN merupakan simpul yang merepresentasikan langkah yang dapat diambil oleh lawan yang dianggap dapat meminimalkan skor lawan dan memaksimalkan skor pemain dengan menimbang langkah yang dapat diambil oleh pemain pada giliran berikutnya.

Berikut ini adalah gambaran dari algoritma Minimax yang mengimplementasikan algoritma DFS dalam pembangunan pohon dinamisnya:



Gambar 3 – pohon dinamis algoritma Minimax

Keterangan gambar di atas:

simpul MAX

simpul MIN

Dari pohon di atas, diperoleh keputusan bahwa pemain akan mengambil langkah yang mengarah ke simpul bernilai 7 karena simpul tersebut menuju ke arah kemungkinan langkah yang terbaik.

B. Menentukan Langkah Terbaik pada Othello

Berikut ini adalah pseudo code implementasi algoritma DFS dan Minimax pada permainan Othello.

```
integer countDisk(input: integer disk_color,
path current_path)
for (int i=0; i<n; i++)
   for (int j=0; j<m; j++)
      if(warna disk == disk_color) then
        count++;
      end if
   end for
end for
return count

procedure findPossiblePath(input: path
current_path)
   mencari path yang mungkin (simpul anak)
   dibangun dari state sebelumnya

procedure findBestPath(input: path current_path)
int depth = 1;
int i = 0;

if (game over) then
   player_count = countDisk(player_color)
   opponent_count = countDisk(opp_color)
   winner = max(player_count, opponent_count)
else if (tidak ada langkah untuk current player)
      skip()
      findBestPath()
   else
      findPossiblePath()
      for each possible_path do
         if(depth = 4) then
            node_val = countDisk(cur_plyr,
            current_path)
         else
            node_val = - max(node_val dari
            depth+1)
         end if
      end for
   end if
end if


procedure findBestPath(input: path current_path)
mengunjungi[current_path] = true
findPossiblePath(current_path)

for each possible_path do
   if (mengunjungi[possible_path] == false)
      findBestPath(possiblepath)
   else
      if(depth = 4) then
         node_val = countDisk(cur_plyr,
         current_path)
      else
         node_val = - max(node_val dari depth+1)
      end if
   end if
end for
```

Fungsi *countDisk()* digunakan untuk menghitung banyaknya disk *current player* pada node tertentu. Prosedur *findPossiblePath()* mencari simpul anak yang mungkin dibangun dari *current node*. Sedangkan prosedur *findBestPath()* mencari path / langkah terbaik yang dipertimbangkan dari nilai setiap simpul.

Berikut ini adalah gambaran langkah program yang dibuat:

1. Diberikan sebuah kondisi papan permainan Othello. Permainan ditinjau dari sisi pemain.
2. Membuat pohon bayangan sebagai tempat pencarian solusi.
3. Pada pohon dinamis, dibangun simpul anak pertama yang merupakan langkah yang dapat diambil oleh pemain.Jika terdapat lebih dari satu simpul anak, telusuri simpul yang paling kiri terlebih dahulu.
4. Dari simpul tersebut, dibangun simpul anak kedua yang merupakan langkah yang mungkin diambil oleh lawan setelah pemain mengambil langkah pada simpul sebelumnya. Telusuri dari simpul paling kiri.
5. Bangun simpul anak ketiga yang merupakan langkah yang dapat diambil oleh pemain setelah lawan mengambil langkah pada simpul sebelumnya.
6. Hitunglah nilai dari simpul tersebut dengan fungsi *countDisk()*.
7. Carilah negasi dari nilai maksimum simpul anak, kemudian simpan nilainya pada simpul orang tua.
8. Ulangi langkah 6 hingga semua simpul telah dikunjungi.
9. Pemain memilih langkah sesuai dengan nilai maksimum yang didapat pada simpul anak pertama.
10. Pada pohon yang sebenarnya, simpul berikutnya dibangun berdasarkan simpul yang dipilih oleh pemain, yaitu simpul dari hasil pencarian solusi pada langkah-langkah sebelumnya.

Permainan Othello memiliki banyak kemungkinan solusi pada suatu kondisi papan. Dengan menggunakan algoritma *depth-first search*, ruang solusi dipersempit karena hanya simpul yang mengarah ke solusi yang

ditelusuri. Simpul yang mengarah ke solusi adalah simpul (langkah) yang diambil oleh pemain setelah dianalisa dengan algoritma Minimax.

V. KESIMPULAN

Algoritma *depth-first search* merupakan algoritma pencarian solusi yang menggunakan struktur pohon dinamis dalam proses pencarian solusi. Proses pembangunan pohon dilakukan selama proses pencarian solusi masih berlangsung.

Algoritma *dept-first search* menelusuri pohonnya secara mendalam. Pencarian dilakukan terhadap simpul yang mengarah ke solusi saja, sehingga algoritma ini menjadi lebih efektif dalam mencari solusi karena tidak semua simpul ditelusuri.

Algoritma *depth-first search* dapat diterapkan dalam algoritma lain yang mendukung dalam pencarian solusi. Salah satunya adalah algoritma Minimax yang membantu pencarian solusi dengan menghitung "untung-rugi" dari sebuah pilihan.

Permainan Othello yang memiliki banyak kemungkinan solusi dapat dipecahkan dengan menggunakan kombinasi algoritma *depth-first search* dan Minimax.

REFERENSI

[1] Rinaldi Munir. Diktat Kuliah IF2251 Strategi Algoritmik. Bandung, Januari 2006.

[2] "Othello: The World's Best Selling Licensed Strategy Game Lands in MDI Entertainment's Portfolio of Lottery Game Properties" , 4 December 2002.
Tanggal akses: 16 Desember 2012.

[3] http://www.pressmantoy.com/instructions/instruct_othello.html
Tanggal akses: 16 Desember 2012.

[4] http://www.cs.ucla.edu/~rosen/161/notes/minimax.html
Tanggal akses: 16 Desember 2012.

[5] http://www.cs.cmu.edu/~./adamchik/15-121/lectures/Game%20Trees/Game%20Trees.html
Tanggal akses: 16 Desember 2012.

[6] http://cs.brown.edu/courses/csci0150/finalprojects/othello/handouts/Othello_AI_2012.pdf
Tanggal akses: 16 Desember 2012.

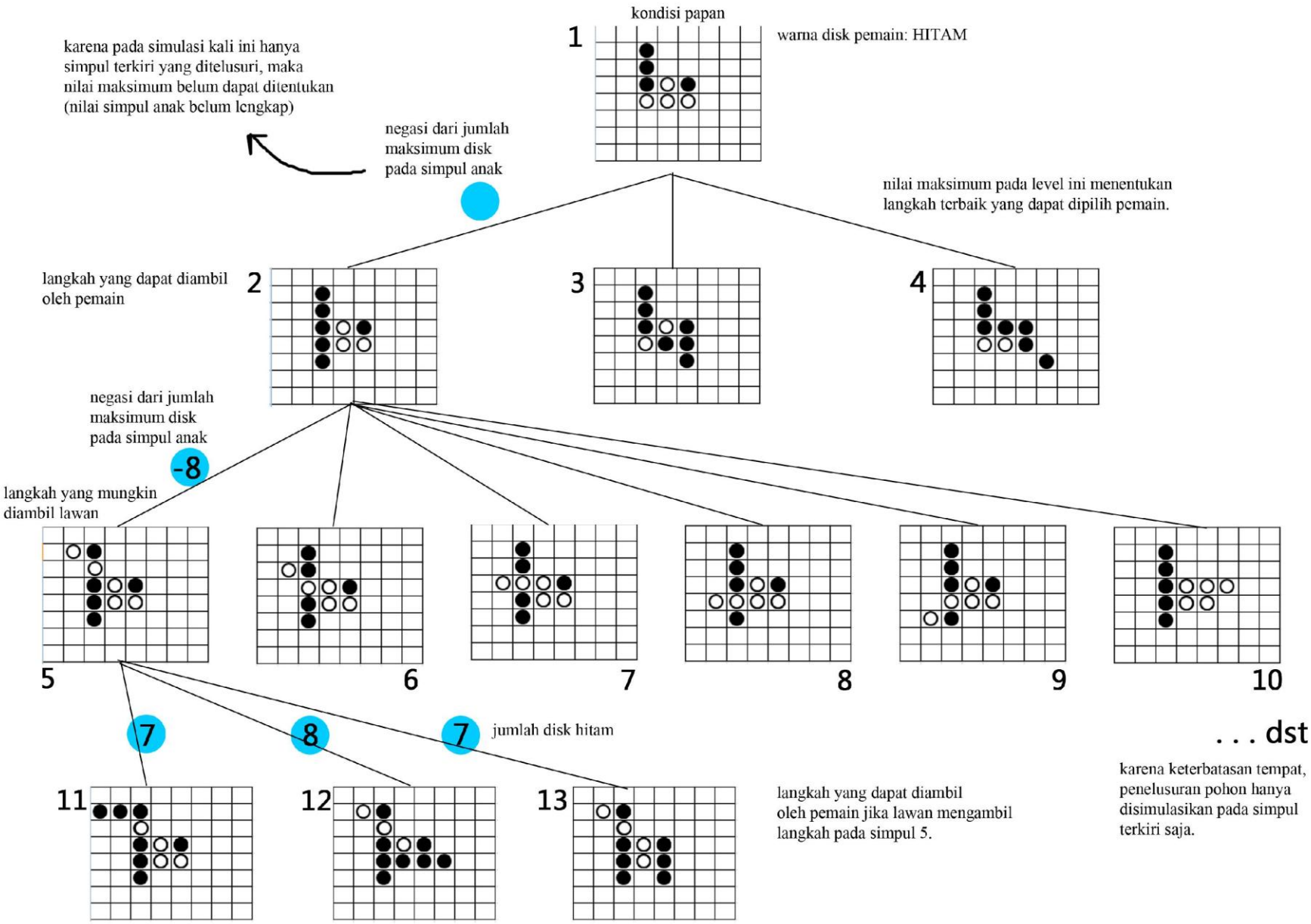
[7] othelloacademy.blogspot.com
Tanggal akses: 21 Desember 2012.

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

Anasthasia Amelia
13510093


kondisi papan
1
warna disk pemain: HITAM
karena pada simulasi kali ini hanya simpul terkiri yang ditelusuri, maka nilai maksimum belum dapat ditentukan (nilai simpul anak belum lengkap)
negasi dari jumlah maksimum disk pada simpul anak
nilai maksimum pada level ini menentukan langkah terbaik yang dapat dipilih pemain.
langkah yang dapat diambil oleh pemain
2
3
4
negasi dari jumlah maksimum disk pada simpul anak
-8
langkah yang mungkin diambil lawan
5
6
7
8
9
10
7
8
7
jumlah disk hitam
. . . dst
11
12
13
langkah yang dapat diambil oleh pemain jika lawan mengambil langkah pada simpul 5.
karena keterbatasan tempat, penelusuran pohon hanya disimulasikan pada simpul terkiri saja.