Penerapan Algoritma DFS dalam Menyelesaikan Permainan Buttons & Scissors

Muhammad Ridwan / 13513008
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
muhammad_ridwan@students.itb.ac.id

***Abstract*—Suatu permainan pada umumnya diciptakan untuk menghilangkan stres atau mengisi waktu luang. Namun banyak pula permainan-permainan yang dibuat untuk mengasah kemampuan otak seperti kakurasu, sudoku, dll. Pada makalah ini, penulis akan memanfaatkan algoritma DFS untuk menyelesaikan permainan buttons & scissors. Penyelesaian yang akan dihasilkan merupakan salah satu solusi untuk menyelesaikan permainan tersebut.**

***Index Terms*—buttons & scissors, DFS, game, solution**

I. PENDAHULUAN

Permainan merupakan suatu alat hiburan yang diciptakan oleh manusia sejak zaman dahulu. Banyak permainan tradisional yang hingga saat ini masih dimainkan oleh anak-anak zaman sekarang bahkan orang dewasa. Permainan diciptakan dengan menciptakan cara bermain dan aturan atau batasan untuk permainan tersebut.

Menurut Kimpraswil (dalam As'adi Muhammad, 2009: 26) mengatakan bahwa definisi permaina adalah usaha olah diri (olah pikiran dan olah fisik) yang sangat bermanfaat bagi peningkatan dan pengembangan motivasi,kinerja, dan prestasi dalam melaksanakan tugas dan kepentingan organisasi dengan lebih baik.

Lain halnya dengan Joan Freeman dan Utami Munandar (dalam Andang Ismail, 2009: 27) mendefinisikan permainan sebagai suatu aktifitas yang membantu anak mencapai perkembangan yang utuh, baik fisik, intelektual, sosial, moral, dan emosional .Menurut beberapa pendapat para ahli tersebut peneliti menyimpulkan definisi permainan adalah suatu aktifitas yang dilakukan oleh beberapa anak untuk mencari kesenangan yang dapat membentuk proses kepribadian anak dan membantu anak mencapai perkembangan fisik,intelektuan, sosial, moral dan emosional.

Pada zama dahulu, permainan hanya dimainkan oleh anak-anak. Di Indonesia ada banyak permainan tradisional yang hingga saat ini masih dimainkan oleh masyarakat seperti congklak, bentengan, ular naga, lompat tali, petak umpet, dll.

Pada era teknologi sekarang, telah banyak jenis-jenis permainan baru yang marak dimainkan oleh anak-anak maupun orang dewasa melalui media komputer, *handphone*, dll. Ada permainan aksi, petualangan, teka-teki, olahraga, dll. Ada pula jenis permainan yang *online* atau *offline* tergantung permainan tersebut.

Untuk permainan teka-teki, banyak pula contoh dari jenis permainan tersebut yang telah ada sejak dahulu seperti sudoku, kakurasu, tic tac toe, dll. Namun, pada zaman sekarang telah bermunculan permainan-permainan teka-teki yang sangat menarik untuk dimainkan seperti 2048, *flow*, paperama, *buttons & scissors*, dll.

Tujuan dari makalah ini adalah memberikan algoritma program yang dapat diimplementasikan untuk menyelesaikan permainan *buttons & scissors*.

II. BUTTONS & SCISSORS

Buttons & scissors merupakan permainan baru yang muncul ketika telah banyak masyarakat yang menggunakan *smartphone* dan android. Permainan ini merupakan *puzzle game* yang hanya membutuhkan satu orang pemain, namun dapat dilakukan bersama-sama. Permainan ini membutuhkan taktik untuk menentukan urutan penyelesaian.

Permainan ini berisi sebuah kain yang memiliki banyak kancing. Kancing tersebut harus digunting dengan cara men-*tap* kancing yang ingin digunting kemudian men-*tap* kancing terakhir yang akan digunting. Kancing hanya dapat digunting ke arah kancing yang memiliki warna yang sama. Kancing tidak bisa digunting sendirian, pengguntingan kancing minimal harus melibatkan dua buah kancing. Kancing dapat digunting ke arah manapun asalkan tidak ada kancing berwarna lain yang menghalangin pengguntingan kancing tersebut.

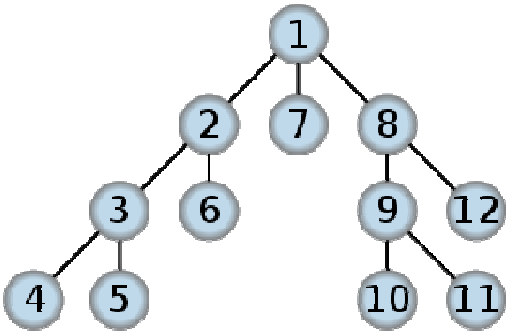
Tujuan akhir dari permainan ini adalah menghabiskan seluruh kancing yang ada pada kain. Permainan tidak memiliki batasan lain seperti waktu. Namun waktu penyelesaian untuk permainan ini akan menjadi faktor pemberian *score* untuk permainan tersebut.

Gambar 1. Permainan buttons & scissors (sumber: http://forum.xda-developers.com/showthread.php?t=2358867)

III. Depth-First Search (DFS)

Depth-First Search (DFS) adalah salah satu strategi pencarian dalam suatu graf. DFS menelusuri simpul yang bertetangga dengan simpul sekarang. Berikut adalah penjelasan tentang algoritma DFS. Penelusuran dimulai dari simpul akar. Dari simpul akar dicari satu simpul yang bertetangga dengannya. Saat ditemukan, penelusuran akan dilakukan pada simpul baru tadi dan mengulangi lagi pencarian simpul tetangga dan seterusnya.

Untuk lebih jelasnya, berikut adalah ilustrasi penjelasan algoritma DFS. Terdapat sebuah graf G dengan 12 simpul seperti Gambar 2. Penelusuran dimulai dari simpul nomor 1.



Gambar 2. Graf G (sumber: http://www.programmerinterview.com/images/DFS.png)

Penelusuran dilakukan mulai dari simpul nomor 1. Dari simpul nomor 1, ditemukan simpul tetangga berikutnya adalah nomor 2. Selanjutnya, penelusuran dilakukan di nomor 2 dan ditemukan simpul tetangganya adalah 3, begitu seterusnya. Perlu diperhatikan pada iterasi ke 3, saat penelusuran dilakukan pada simpul 4, tidak ditemukan lagi simpul tetangganya. Oleh karena itu, akan dilakukan backtrack dan pencarian kembali ke simpul nomor 3. Dari simpul nomor 3, ternyata ada simpul tetangga lain selain nomor 4, yaitu nomor 5. Maka, penelusuran dilanjutkan ke nomor 5.

Selanjutnya simpul nomor 3 tidak lagi memiliki simpul tetangga. Oleh karena itu dilakukan backtrack kembali ke simpul nomor 2 dan dilakukan pencarian kembali. Pencarian dan bactrack seperti ini terus dilakukan hingga seluruh simpul telah terjamah dan/atau solusi telah ditemukan. Sehingga, urutan penelusuran dari graf G di atas dengan menggunakan algoritma DFS adalah 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12.

Dalam implementasinya, struktur data algoritma DFS biasanya menggunakan struktur data rekursif atau tumpukan (stack). Struktur data stack biasanya digunakan untuk algoritma DFS yang iteratif. Berikut adalah pseudocode dari algoritma DFS dalam versi rekursif dan versi stack yang telah dijelaskan tadi.

procedure DFS (Graph G, Node s) // versi rekursif

```
visited[s] ← true
foreach simpul w bertetangga dengan
s in G do
        if not visited[w] then
                DFS(G,w)
        endif
endfor
```

procedure DFS (Graph G, Node s) // versi iteratif

```
S ← { }
PUSH(S,s)
while (S) not empty do
        POP(S,s)
        if not visited[s] then
                visited[s] ← true
                foreach simpul w bertetangga
                dengan s in G do
                        PUSH(S,w)
                endfor
        endif
endwhile
```

IV. Implementasi DFS dalam Penyelesaian Permainan Buttons & Scissors

Dalam makalah ini, kita akan menganggap kain dalam permainan buttons & scissors sebagai suatu matriks yang berisi integer. Element integer matriks tersebut merepresentasikan warna dari kancing. Contoh kain dan matriks.

Gambar 3. Contoh kain (sumber: http://www.techydad.com/wp-content/uploads/2014/01/buttons_and_scissors.jpg)

Berdasarkan kain pada gambar 3, representasi matriks adalah sebagai berikut.

1	2	3	3	3
4	5	3	5	5
6	2	1	2	2
6	6	4	6	3
3	3	3	4	1

Tabel 1. Representasi matriks untuk gambar 3

A. Menentukan apakah kancing mungkin digunting

Kancing pada indeks (i,j) mungkin digunting apabila ada kancing yang bersebelahan yang memiliki warna yang sama dengan kancing tersebut.

function IsMayCutUpRight(MATRIX M, int i, int j)

```
if (i != 0 and j != M.NKol – 1) then
        while(M[i – 1][j + 1] = 0) do
                i ← i – 1
                j ← j + 1
        endwhile
        if (M[i - 1][j + 1] = M[i][j]) then
                return true
        endif
endif
return false
```

function IsMayCutDown(MATRIX M, int i, int j)

```
if (i != M.NBrs – 1) then
        while(M[i + 1][j] = 0) do
                i ← i + 1
        endwhile
        if (M[i + 1][j] = M[i][j]) then
                return true
        endif
endif
return false
```

function IsMayCutRight(MATRIX M, int i, int j)

```
if (j != M.Nkol – 1) then
        while(M[i][j + 1] = 0) do
                j ← j + 1
        endwhile
        if (M[i][j + 1] = M[i][j]) then
                return true
        endif
endif
return false
```

function IsMayCutDownRight(MATRIX M, int i, int j)

```
if (i != M.NBrs – 1 and j != M.Nkol – 1) then
        while(M[i + 1][j + 1] = 0) do
                i ← i + 1
                j ← j + 1
        endwhile
        if (M[i + 1][j + 1] = M[i][j]) then
                return true
        endif
endif
return false
```

B. Menggunting kancing

Mengunting kancing dengan cara merubah element Matriks M pada indeks kuncing yang digunting menjadi 0. Kuncing digunting ke arah yang ditentukan sejumlah kancing yang berada pada arah kancing tersebut.

procedure CutUpRight(MATRIX M, int i, int j, sum)

```
sum ← 0
while (IsMayCutUpRight(M,i,j)) do
        M[i][j] ← 0
        sum ← sum + 1
        while(M[i – 1][j + 1] = 0) do
                i ← i – 1
                j ← j + 1
        endwhile
endwhile
M[i][j] ← 0
sum ← sum + 1
```

procedure CutDown(MATRIX M, int i, int j, sum)

```
sum ← 0
while (IsMayCutDown(M,i,j)) do
        M[i][j] ← 0
        sum ← sum + 1
        while(M[i + 1][j] = 0) do
                i ← i + 1
        endwhile
endwhile
M[i][j] ← 0
sum ← sum + 1
```

procedure CutRight(MATRIX M, int i, int j)

```
sum ← 0
while (IsMayCutRight(M,i,j)) do
        M[i][j] ← 0
        sum ← sum + 1
        while(M[i][j + 1] = 0) do
                j ← j + 1
        endwhile
endwhile
M[i][j] ← 0
sum ← sum + 1
```

procedure CutDownRight(MATRIX M, int i, int j)

```
sum ← 0
while (IsMayCutDownRight(M,i,j)) do
        M[i][j] ← 0
        sum ← sum + 1
        while(M[i + 1][j + 1] = 0) do
                i ← i + 1
                j ← j + 1
        endwhile
endwhile
M[i][j] ← 0
sum ← sum + 1
```

C. DFS dan Penentuan Solusi

Andaikan kita memiliki List of Solution yang setiap elemennya beranggotakan (ic = indeks i pengguntingan, jc = indkes j pengguntingan, ac = arah pengguntingan, jc = jumlah pengguntingan untuk arah tersebut). Arah pengguntingan yang mungkin adalah: 1. Kanan atas, 2. Kanan, 3. Kanan bawah, dan 4. Bawah. Pada setiap simpul diperiksa apakah simpul tersebut simpul akhir atau bukan, selanjutnya diperiksa apakah simpul mati, lalu baru diperiksa untuk setiap element yang masih tersedia apakah bisa digunting, jika bisa maka masuk ke simpul baru dan gunting kancing tersebut.

DFS (Matriks M, List Solution) //Solver

```
if seluruh element sudah habis then
    return dummy valid sol
endif
if element belum habis tapi tidak dapat lagi digunting
    return invalid sol
endif
foreach baris
    foreach kolom
        if (M[baris][kolom] != 0)
            if (IsMayCutUpRight(M,baris,kolom))
                M1 ← CopyMatriks(M)
                CutUpRight(M,baris,kolom,sum)
                DFS(M,sol)
                if (sol valid) then //kancing dapat digunting
dan sudah digunting
                Solution.addFirstElement(baris,kolom,1,sum)
                    return Solution
                else
                     return invalid sol
                endif
            endif
            if (IsMayCutDown(M,baris,kolom))
                M1 ← CopyMatriks(M)
                CutDown(M,baris,kolom,sum)
                DFS(M,sol)
                if (sol valid) then //kancing dapat digunting
dan sudah digunting
                Solution.addFirstElement(baris,kolom,4,sum)
                    return Solution
                else
                     return invalid sol
                endif
            endif
            if (IsMayCutRight(M,baris,kolom))
                M1 ← CopyMatriks(M)
                CutRight(M,baris,kolom,sum)
                DFS(M,sol)
                if (sol valid) then //kancing dapat digunting
dan sudah digunting
                Solution.addFirstElement(baris,kolom,2,sum)
                    return Solution
                else
                     return invalid sol
                endif
            endif
            if (IsMayCutDownRight(M,baris,kolom))
                M1 ← CopyMatriks(M)
                CutDownRight(M,baris,kolom,sum)
                DFS(M,sol)
                if (sol valid) then //kancing dapat digunting
dan sudah digunting
                Solution.addFirstElement(baris,kolom,3,sum)
                    return Solution
                else
                     return invalid sol
```

```
            endif
          endif
        endif
      endfor
    endfor
```

Berikut penggambaran mekanisme DFS di atas untuk persoalan pada gambar 3.

Simpul 1

1	2	3	3	3
4	5	3	5	5
6	2	1	2	2
6	6	4	6	3
3	3	3	4	1

Simpul 2

1	2	0	0	3
4	5	3	5	5
6	2	1	2	2
6	6	4	6	3
3	3	3	4	1

Simpul 3

1	2	0	0	3
4	5	3	0	0
6	2	1	2	2
6	6	4	6	3
3	3	3	4	1

Simpul 4

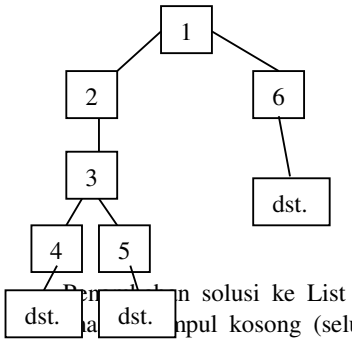
1	2	0	0	3
4	5	3	0	0
0	2	1	2	2
0	6	4	6	3
3	3	3	4	1

Simpul 5

1	2	0	0	3
4	5	3	0	0
0	2	1	2	2
6	0	4	6	3
3	3	3	4	1

Simpul 6

1	2	0	3	3
4	5	0	5	5
6	2	1	2	2
6	6	4	6	3
3	3	3	4	1



Pen[illegible]n solusi ke List dimulai ketika DFS telah [illegible]mpul kosong (seluruh elemen matriks telah 0). List akan ditambahkan dummy solution. Lalu DFS akan kembali ke simpul akar, menambahkan solution untuk simpul tersebut, lalu kembali ke simpul akar. Seterusnya, hingga sampai ke simpul awal.

Untuk persoalan tersebut, berikut salah satu solusi yang mungkin akan dikembalikan oleh fungsi DFS:

- (2,3,1,2)
- (2,2,2,3)
- (1,2,4,2)
- (2,1,3,3)
- (3,1,4,2)
- (3,4,2,2)
- (1,5,4,2)
- (4,2,2,2)
- (1,1,3,3)
- (1,3,4,2)
- (5,1,2,2)

V. KESIMPULAN

Algoritma DFS merupakan algoritma yang cukup baik untuk diterapkan pada persoalan buttons & scissors. Hal ini dikarenakan algoritma DFS menggunakan pencarian dengan cara menjalar ke akar. Sehingga apabila akar telah ditemukan, pencarian tidak perlu dilanjutkan ke akar selanjutnya.

Untuk persoalan buttons and scissors, Algoritma DFS juga lebih baik ketimbang algoritma BFS. Hal ini karena, Simpul solusi tidak mungkin ditemukan di awal-awal (untuk matriks 5 x 5 minimal ada 6 simpul) Sehingga akan terbilang boros apabila menggunakan algoritma BFS.

VI. UCAPAN TERIMAKASIH

Pertama, penulis mengucapkan terima kasih kepada Allah swt yang telah menyertai penulis sehingga berhasil menyelesaikan tugas makalah ini.

Penulis juga mengucapkan terima kasih kepada Bapak dan Ibu pengajar mata kuliah IF2211 Strategi Algoritma, yaitu Bapak Dr. Ir. Rinaldi Munir, M.T dan Ibu Dr. Nur Ulfa Maulidevi, S.T, M.Sc. yang telah mengajarkan banyak pelajaran kepada penulis di satu semester ini. Atas bimbingan Bapak dan Ibu, penulis akhirnya berhasil menyelesaikan mata kuliah ini.

Penulis juga ingin berterima kasih atas pelajaran eksplorasi strategi algoritma yang Bapak dan Ibu berikan yang dapat penulis bawa sebagai bekal di masa depan. Tanpa tugas eksplorasi yang Bapak dan Ibu berikan, penulis tidak dapat mendapatkan pengalaman lebih dari strategi algoritma saja. Semoga seluruh perjuangan yang telah penulis dapatkan di dalam mata kuliah ini dapat bermanfaat bagi penulis dan lingkungan sekitar.

DAFTAR PUSTAKA

[1] http://www.scribd.com/doc/172744204/Pengertian-Permainan-Menurut-Beberapa-Ahli#scribd diakses tanggal 4 Mei 2015 pukul 20.00

[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2013-2014-genap/Makalah2014/MakalahIF2211-2014-096.pdf diakses tanggal 4 Mei 2015 pukul 20.00

[3] Munir, Rinaldi. 2009. Diktat Kuliah Strategi Algoritma, Penerbit Informatika: Bandung.

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, 5 Mei 2015

Muhammad Ridwan
13513008