Perbandingan BFS dan DFS pada Pembuatan Solusi Penyelesaian Permainan Logika

Geraldi Anggapardana (13511097)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13511097@std.stei.itb.ac.id

ABSTRAK

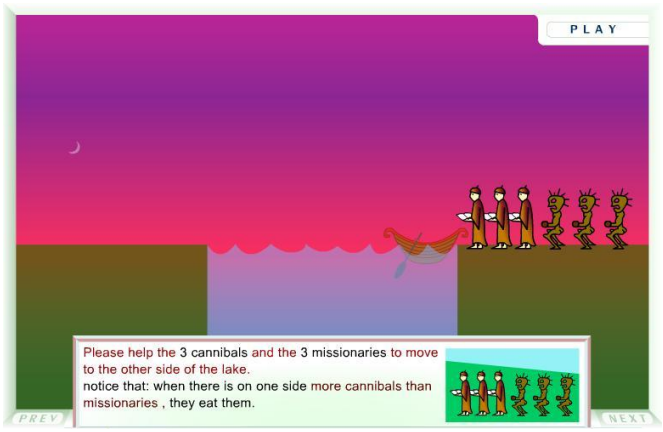
Makalah ini menjelaskan tentang cara menyelesaikan sebuah permainan logika menggunakan teknik algoritma* Breadth First Search (BFS) dan Depth First Search (DFS) *dan memberikan solusinya. Akar pohon yang merupakan status awal permainan disimpan menjadi sebuah akar pohon aras 0. Kemudian dengan metode BFS dan DFS dilakukan pembuatan pohon dinamis sampai ditemukan sebuah solusi dan kemudian hasilnya akan dicatat dan ditampilkan dalam layar. Kemudian kedua metode ini dibandingkan mana yang lebih baik.State yang sudah ada didalam pohon diberi tanda agar state yang sama tidak akan terjadi/terbentuk pada aras yang lebih tinggi.

Kata kunci*—pohon, solusi, *solusion mark.

I.PENDAHULUAN

Flash game adalah kumpulan permainan-permainan yang sudah populer sejak beberapa tahun lalu dan sampai sekarang jenis permainan ini tetap banyak diminati sebagai pengisi waktu luang atau waktu istirahat karena permainan ini berskala kecil dan bisa diulang-ulang tidak seperti *game online* atau permainan skala besar lainnya. *Flash game* memiliki banyak ragam permainan, ada permainan olahraga, menembak, balapan, *entertainment*, strategi, logika, dan lain-lain. Setiap kategori pun terbagi lagi menjadi banyak jenis permainan, misalnya sepak bola, basket, tenis dengan berbagai macam judul, *snot put, hobo, NY Shark, PvZ*, dan masih banyak lagi. Dan salah satu permainan *flash* ini adalah *Missionary and Cannibals.*

Permainan ini adalah salah satu permainan *flash* bergenre logika yang sudah ada sejak beberapa tahun yang lalu. *Missionary and Cannibals* ini adalah sebuah permainan berskala kecil dimana terdapat tiga orang misionaris dan tiga orang kanibal yang bertujuan menyeberangi sebuah sungai agar mereka semua tiba di pulau sebelah dengan selamat. Disini hanya tersedia sebuah perahu dengan kapasitas dua penumpang saja. Dan jumlah misionaris tidak boleh lebih sedikit dari jumlah kanibal dalam satu pulau, jika terjadi maka misionaris akan dimakan oleh kanibal dan permainan selesai. Berikut gambar *interface* dari permainan ini :

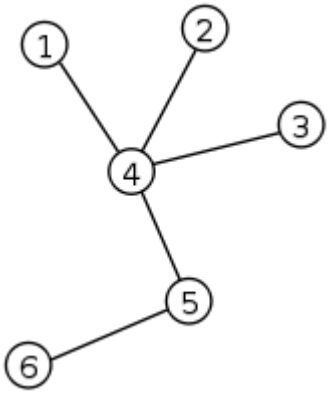


Pada makalah kali ini akan dijelaskan, bagaimana menyelesaikan problem pada permainan ini secara algoritmik dan secara "otomatis". Dengan menggunakan salah satu metode strategi algoritma, yaitu BFS (*Breadth First Searching*) akan dicari sebuah strategi dan solusi dari permainan ini dan akan dijelaskan mengenai cara penyelesaian dari algoritma ini.

II. DASAR TEORI

II.a POHON

Pohon adalah graf khusus. Graf khusus disini berarti graf tak-berarah terhubung yang tidak mengandung sirkuit. Contoh dari pohon adalah sebagai berikut :



Gambar 2.1

Pohon juga bersifat unik untuk setiap lintasan pasang simpul yang ada. Dari contoh gambar 2.1, dari 1 ke 4 hanya satu lintasan, dari 2 ke 4 hanya satu lintasan,

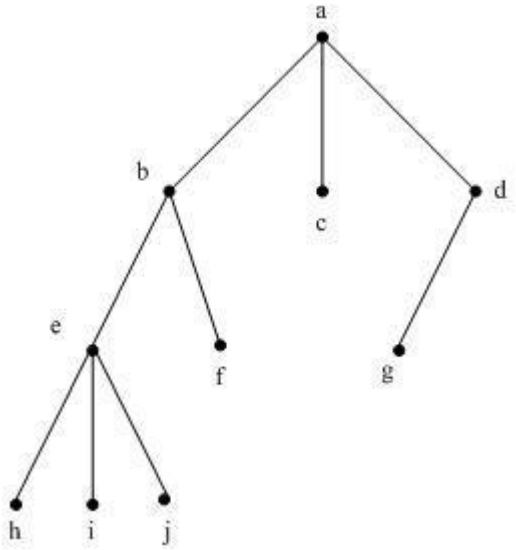
dan seterusnya. Berikut ini adalah sifat-sifat dari pohon[1] yang dimisalkan oleh G = (V,E) adalah:

a. G adalah pohon
b. setiap pasang simpul di dalam G terhubung dengan lintasan tunggal.
c. G terhubung dan memiliki $m = n - 1$ buah sisi
d. G tidak mengandung sirkuit dan memiliki $m = n - 1$ buah sisi
e. G tidak mengandung sirkuit dan penambahan satu sisi pada graf akan membuat hanya satu sirkuit.
f. G terhubung dan semua sisinya adalah jembatan.

Jembatan adalah sisi yang bila dihapus menebabkan graf terpecah menjadi dua komponen.

II.a.2 Pohon Berakar

Pohon yang sebuah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah menjauh dari akar dinamakan pohon berakar. Bisa dilihat pada gambar berikut :



Gambar 2.2

II.b BFS

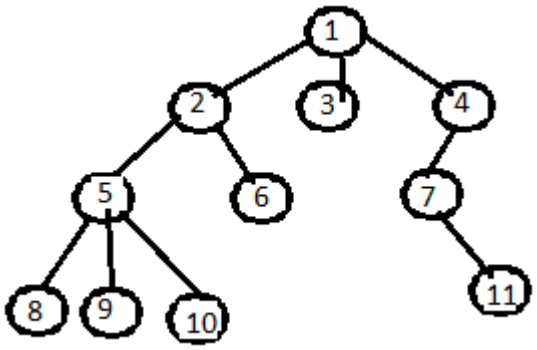
BFS adalah metode pencarian melebar atau secara kepanjangan adalah *Breadth First Search*, yaitu adalah metode strategi algoritma yang bersifat traversal dan berupa pohon. Misalkan kita memunyai graf G yang memunyai n buah simpul. Kita akan melakukan traversal di dalam graf, dan misalkan traversal dimulai dari simpul v. maka algoritma BFS adalah sebagai berikut :

1. Kunjungi simpul v
2. Kunjungi semua simpul yang bertetangga dengan simpul v terlebih dahulu.
3. Kunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya.

Algoritma BFS memerlukan tabel Boolean yang bernama ‘dikunjungi’ untuk menyimpan simpul-simpul yang telah dikunjungi (ini dimaksudkan agar tidak ada simpul yang dikunjungi lebih dari satu kali).

Gambar dan Ilustrasi algoritma BFS :

1. gambar



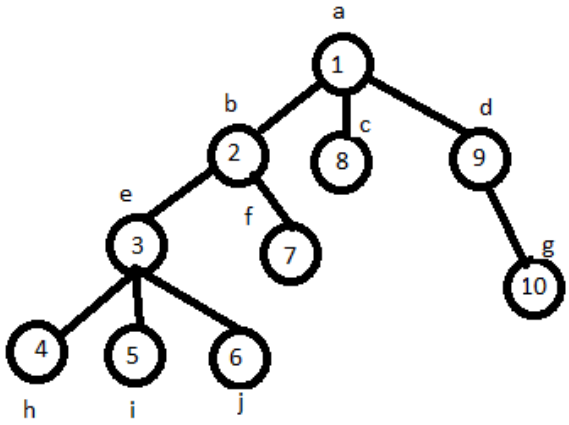
Gambar 2.3

Angka yang ada pada gambar menunjukkan urutan pencarian dari metode BFS. Algoritma BFS bersifat traversal biasa. Algoritma ini akan terus mengulangi prosesnya terurut membesar.

II.c DFS

DFS(*Depth First Search*) adalah metode pencarian menggunakan pohon yang bergerak secara mendalam. Pada umumnya, DFS melakukan pencarian secara prefix. Bila kita melihat gambar 2.2 disamping, maka bentuk path dari DFS adalah a-b-e-h-i-j-f-c-d-g secara berturut-turut.

Ilustrasi gambar proses DFS dapat dilihat pada sebagai berikut untuk memperjelas pengertian dari DFS yaitu :



Gambar 2.3

Misalkan kita memiliki graf G yang memiliki n buah simpul. Traversal dimulai dari simpul v. Berikut merupakan algoritma umum dari DFS :

1.Kunjungi simpul v

2. Kunjungi simpul w yang bertetangga dengan simpul v.

3. Ulangi DFS dari simpul w.

4. Ketika mencapai simpul u sedemikian rupa sehingga semua simpul yang bertetangga dengannya telah dikunjungi, pencarian dirunut-balik ke simpul terakhir

yang dikunjungi sebelumnya dan memunyai simpul w yang belum dikunjungi.

5. pencarian berakhir bila tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi.

Algoritma DFS ini bersifat rekursif, karena dia mengulangi proses saat menemui state baru.

IIIPENENTUAN STATE PADA POHON DINAMIS DAN ALGORITMA BFS DAN DFS

Pada persoalan ini, pada permainan logika *Missionary and Cannibal*, akan dibuat state awal terlebih dahulu. Permainan ini punya beberapa peraturan yaitu :

1. Pada awalnya terdapat tiga orang dan tiga orang kanibal pada pulau sebelah kanan

2. tujuan permainan ini adalah memindahkan ke enam orang ini ke pulau sebelah, melewati laut dengan menaiki perahu.

3. kapasitas perahu maksimal dua orang.

4. perahu dapat bergerak jika ada penumpangnya minimal satu.

5. jumlah orang dalam satu pulau tidak boleh lebih sedikit daripada jumlah kanibal, karena akan dimakan dan permainan akan selesai.

Dari kelima peraturan ini kemudian terbentuk sebuah kondisi state, dimana setiap state dibedakan dengan perbedaan *move* atau pergerakkan tiap karakter (orang/kanibal). Yang dimaksud oleh *move* state ini adalah :

1. orang menaiki perahu
2. kanibal menaiki perahu
3. orang menuruni perahu
4. kanibal menuruni perahu
5. perahu menyeberangi sungai

Kelima *move* ini yang menggerakkan dan membedakan tiap state yang terbentuk dalam sebuah pohon yang akan dibentuk dengan metode BFS ini.

Pada metode BFS ini juga diperlukan adanya suatu *Boolean* yang menyatakan bahwa suatu state pernah didefinisikan pada state di aras yang sama/sebelumnya. Metode Boolean tersebut memiliki kriteria sebagai berikut agar state selanjutnya dapat tercantum pada pohon:

1. membandingkan state yang baru sama dengan state yang sudah ada sebelumnya.

2. memeriksa apakah perahu dapat menyeberangi sungai asalkan isi perahu tidak kosong. (peraturan no. 4)

3. memeriksa apakah jumlah orang dalam suatu pulau tidak kurang dari jumlah kanibal yang ada di situ.

Kemudian pada persiapan algoritma BFS yang akan digunakan adalah pohon dinamis, yaitu pohon yang dibuat langsung bersamaan dengan berjalannya algoritma BFS.

III.b. Algoritma BFS

Berikut adalah algoritma BFS yang digunakan pada percobaan kali ini:

1. bentuk simpul akar pada sebuah simpul dinamis

2. mencoba state dengan mengubah keadaan state sesuai dengan *move* state yang sudah dijabarkan pada bab sebelumnya.

3. lakukan validasi dengan *Boolean* yang sudah didefinisikan

4. jika valid, maka terbentuk simpul dengan state baru yang berisi keadaan baru setelah ditambah dengan *move* state.

5. jika tidak, maka state baru tidak akan dimasukkan ke dalam pohon.

6. ulangi dari langkah 2 sebanyak *move* state yang didefinisikan

7. ulangi dari langkah satu dengan akar baru adalah saudara dari simpul/anak pertama dari simpul(dalam hal ini anak paling kiri) jika sudah tidak ada lagi saudara yang bisa diakses

8. proses berhenti bila sudah ketemu state solusinya, yaitu 0;0;3;0;0;3;kiri

III.c Algoritma DFS

Berikut adalah algoritma dari DFS pada permainan ini :

1. bentuk simpul akar pada sebuah simpul dinamis.

2. bentuk node anak dari mulai state 1 pada kondisi *move* state. Proses diulangi secara rekursif. Ulangi terus langkah 2 sampai kondisi *invalid* atau state sudah semua dikunjungi.

3. jika pembentukan bernilai *invalid*, maka akan dibentuk anak dengan state selanjutnya. Ulangi langkah 2 dengan state selanjutnya.

4. jika semua state telah dikunjungi, pencarian di runut balik pada simpul orang tuanya dan ulangi langkah 2.

5. proses berhenti bila sudah tidak ada state yang bisa dibuat/ketemu solusinya, , yaitu 0;0;3;0;0;3;kiri

IV. HASIL DAN ANALISIS METODE PEMBENTUKKAN SOLUSI DENGAN ALGORITMA BFS DAN DFS

IV.a. Pembentukan state awal

Dibentuk sebuah state yang merupakan akar awal dari sebuah pohon dinamis dimana dalam suatu state tersebut mengandung atribut sebagai berikut :

1. jumlah orang disebelah kanan
2. jumlah orang di perahu
3. jumlah orang disebelah kiri
4. jumlah kanibal disebelah kanan
5. jumlah kanibal di perahu
6. jumlah kanibal disebelah kiri
7. posisi perahu saat ini (kiri/kanan)

Seperti dilihat pada gambar diatas, maka state awal yang terbentuk bila sesuai dengan urutan atribut yang dijabarkan diatas adalah 3;0;0;3;0;0;kanan (sesuai urutan atribut di atas, **cara penulisan state akan dijadikan acuan pada bab IV**).

IV.b proses dan hasil pada algoritma BFS

Pada awalnya, dibentuk sebuah simpul pertama dengan state 3;0;0;3;0;0;kanan (subbab IV.a) dan disimpan sebagai akar pada sebuah pohon dinamis. Kemudian dibuat langkah selanjutnya yaitu membangun simpul selanjutnya, dimana disini adalah anak dari simpul pertama. Diiterasi sebanyak lima state, kemudian didapatkan anak baru yaitu dengan state sebagai berikut :

1. 2;1;0;3;0;0;kanan (state 1 , orang menaiki perahu)
2. 3;0;0;2;1;0;kanan (state 2 , kanibal menaiki perahu).

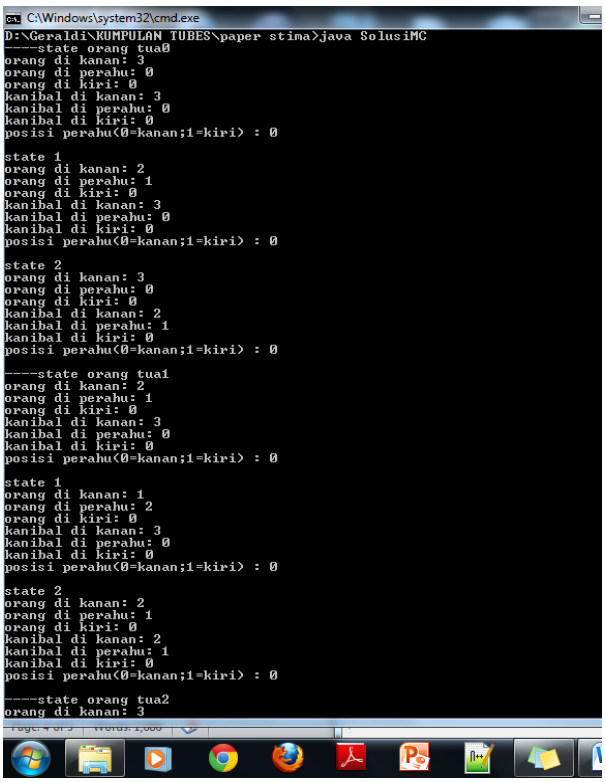
State 3, 4, 5 tidak terbentuk karena dia tidak dianggap valid oleh fungsi Boolean yang ada, seperti yang sudah dikutip di bab III. State 3 dan 4 tidak terbentuk karena perahu dalam keadaan kosong, begitu pula dengan state 5 karena kosong, sehingga tidak bisa untuk dipakai untuk menyeberang.Kemudian pada state yang nomor 1, dilakukan iterasi pengembangan pohon dan mendapat data sebagai berikut :

1. 1;2;0;3;0;0;kanan
2. 2;1;0;2;1;0;kanan
3. 2;1;0;3;0;0;kiri (state 5, perahu menyeberang sungai)

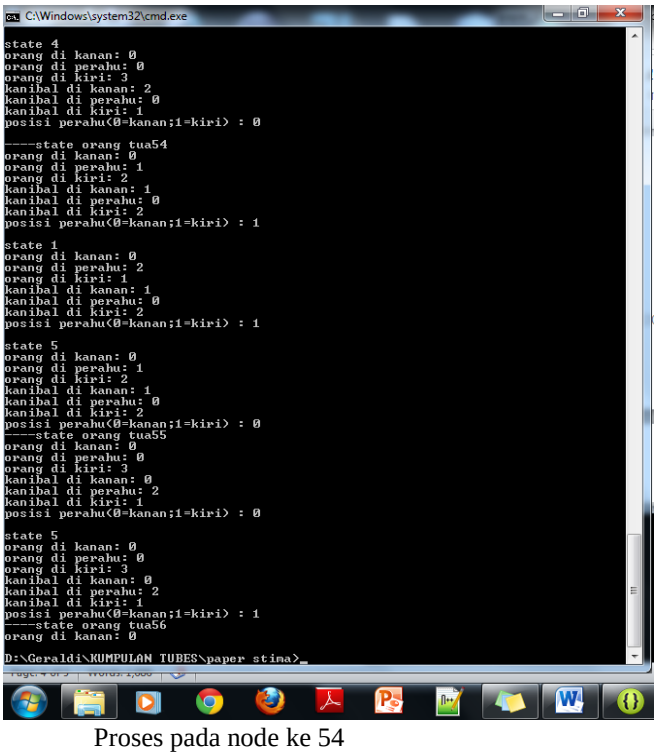
State 3 dan 4 tidak terbentuk, karena pada state 3, akan menjadi sama dengan state sebelumnya, yaitu 3;0;0;3;0;0;kanan, sehingga tidak valid. Pada state 4 belum ada kanibal yang ada di perahu.

Kemudian proses dilanjutkan sampai solusi ditemukan.

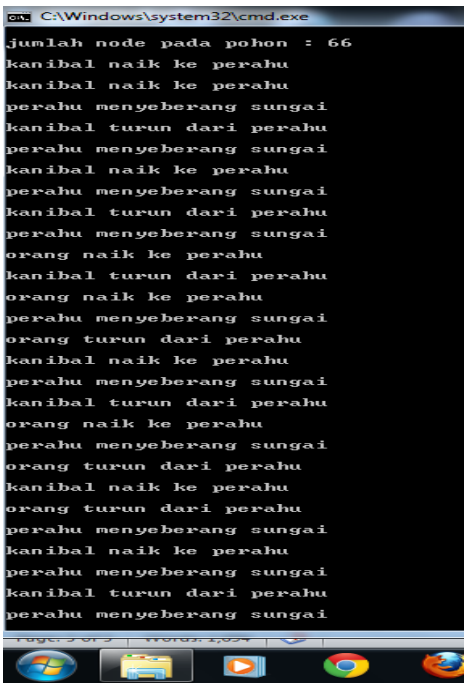
Berikut *screenshot* dari hasil dari BFS yang diimplementasikan :



Awal permulaan proses



Proses pada node ke 54



Solusi dari algoritma yang dibuat

Dari data yang ada di atas, terlihat bahwa jumlah node yang terbentuk ada 66 buah. State yang terbentuk hanya state yang tertulis dilayar seperti pada gambar diatas, sehingga metode BFS ini singkat dan relatif cepat, dengan banyak kondisi yang terbentuk.

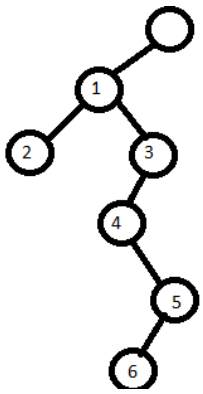
IV.c Proses dan hasil pada algoritma DFS

Pada awalnya, dibentuk sebuah simpul pertama dengan state 3;0;0;3;0;0;kanan (subbab IV.a) dan disimpan sebagai akar pada sebuah pohon dinamis. Kemudian dibuat langkah selanjutnya yaitu membangun simpul selanjutnya, dimana disini adalah anak dari simpul pertama. Proses DFS dilakukan secara rekursif dan didapatkan potongan data sebagai berikut dilihat dari state pertama :

1. 2;1;0;3;0;0;kanan (state1)
2. 1;2;0;3;0;0;kanan (state1)
3. 2;1;0;2;1;0;kanan(state2)
4. 3;0;0;2;1;0;kanan(state4)
5. 3;0;0;1;2;0;kanan(state2)
6. 3;0;0;1;2;0;kiri(state5)
7. … dst

Pada data yang didapat diatas, ‘1’ adalah anak dari simpul pertama, ‘2’ adalah anak dari ‘1’ , ‘3’ adalah anak dari ‘1’, karena ‘2’ sudah tidak bisa membentuk anak karena semua kondisi sudah *invalid*. Pada state 1 dan 2 *invalid* karena perahunya sudah penuh, state 3 akan membuat state sebelumnya terulang, state 4 tidak bisa karena tidak ada kanibal di perahu dan state 5 tidak bisa karena sisa orang di pulau kanan < kanibal yang ada di pulau kanan.

Selanjutnya ‘4’ adalah anak dari ‘3’, ‘5’ adalah anak dari ‘4’, ‘6’ adalah anak dari ‘5’. Berikut dilampirkan bentuk pohon yang terbentuk agar lebih mudah dimengerti:



Keterangan : angka = urutan pembentukan DFS

Penomoran dilakukan sesuai dengan state yang sudah didefinisikan di atas.

IV.d Perbandingan BFS dan DFS

Pada kasus ini, DFS menghasilkan/menciptakan node pada pohon dinamis lebih sedikit daripada BFS, dimana sebanyak 66 buah node. Pencarian menggunakan lebih cepat menggunakan DFS daripada BFS. Tapi aras/level yang dihasilkan menuju solusi jauh lebih panjang daripada pada pembuatan pohon dinamis BFS.

Dapat terlihat dari kasus berikut. Pada subbab IV.e , state ‘4’ bernilai 3;0;0;2;1;0;kanan berada di aras 3, nilai ini sama dengan hasil BFS yang dilihat pada *screenshot* pertama dimana nilai state ‘2’ juga bernilai 3;0;0;2;1;0;kanan dan berada di aras pertama pada pohon. Berikut hasil dari DFS bila dituliskan secara solusi dimulai dari pohon :

1. orang menaiki perahu
2. kanibal menaiki perahu
3. orang menuruni perahu
4. .. DST

Dari awal solusi saja sudah menunjukkan bahwa solusi yang dihasilkan oleh DFS tidak mangkus, karena pada BFS dapat ditulis langsung ‘kanibal menaiki perahu’ tanpa harus menaikkan dan menurunkan orang terlebih dahulu. Solusi yang dihasilkan DFS juga jauh lebih panjang dari BFS, sehingga dalam kasus ini BFS lebih mangkus dari DFS dalam menyelesaikan permainan logika.

V. KESIMPULAN

Algoritma BFS dapat digunakan untuk memecahkan masalah-masalah banyak permainan logika, salah satunya adalah *missionary and canibals* ini karena permainan logika membutuhkan banyak state yang harus dicari secara bertahap. Algoritma BFS juga dikatakan efektif karena dengan metode melebar ini, pohon tidak memerlukan aras yang terlalu banyak dan dapat dengan cepat diperoleh solusi serta solusi yang memiliki aras minimum dengan akar daripada menggunakan metode DFS. Dalam kasus ini penggunaan DFS tidak efektif karena hasil akhirnya yang kurang mangkus jika digunakan dalam permainan dan solusinya yang panjang sehingga dapat membingungkan pemain yang membutuhkan bantuan.

DAFTAR PUSTAKA

[1] Munir, Rinaldi.*Strategi Algoritma*.Informatika Bandung, 2009

[2] Munir, Rinaldi.*Slide BFS-DFS edisi 2013-14.* Informatika Bandung. 2013

[3] Anggapardana,Geraldi.*Pencarian dan Pengolahan File Menggunakan Metode Pohon Biner pada Program*.Informatika Bandung,2012.

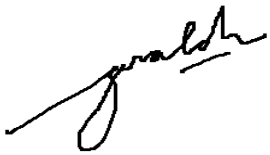
[4] http://xaestanapatela.blogspot.com/ , 14 Desember , pk 18.57 WIB

[5] http://www.8flashgames.com/ , 14 Desember 2013 pk 18.44 WIB

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, 20 Desember 2013



Geraldi Anggapardana / 13511097