Penerapan DFS dan BFS dalam Pencarian Solusi Game "Japanese River IQ Test"

Hanif Eridaputra / 13510091
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13510091@std.stei.itb.ac.id

***Abstract*—Japanese River IQ Test adalah permainan yang berasal dari Jepang. Tujuan utama dari permainan ini adalah menyebrangkan semua orang melalui sungai dengan menggunakan perahu yang hanya cukup untuk dua orang dengan peraturan-peraturan tertentu. Pada makalah ini, akan dibahas mengenai pencarian solusi yang tepat untuk menyelesaikan permainan Japanese River IQ Test ini. Terdapat dua jenis algoritma yang akan digunakan yaitu Breadth First Search (BFS) dan Depth First Search (DFS). Kedua algoritma ini termasuk algoritma travelsal dalam graf. Maka dari itu, solusi akan dibuat dalam bentuk pohon pencarian.**

***Index Terms*—Japanese River IQ Test, Breadth First Search, Depth First Search, traversal.**

I. PENDAHULUAN

Dalam permainan Japanese River IQ Test, diceritakan bahwa terdapat delapan orang yang ingin menyebrangi sungai. Delapan orang ini adalah satu orang ayah, satu orang ibu, dua anak laki-laki, dua anak perempuan, satu orang pencuri dan satu orang polisi. Pemain diminta untuk menyebrangkan delapan orang tersebut dengan sebuah perahu yang hanya cukup untuk dua orang saja. Peraturannya adalah sebagai berikut:

1. Perahu hanya dapat dioperasikan oleh ayah, ibu atau polisi dan tidak dapat digerakkan tanpa ada orang yang mengoperasikannya.
2. Anak laki-laki tidak boleh ditinggal bersama ibu kecuali apabila ada ayah yang bersamanya.
3. Anak perempuan tidak boleh ditinggal bersama ayah kecuali apabila ada ibu yang bersamanya.
4. Semua orang tidak boleh ditinggal bersama pencuri kecuali ada polisi yang bersamanya. Pencuri boleh ditinggal sendirian.

Keempat peraturan di atas berlaku untuk kedua sisi sungai. Berikut adalah tampilan permainannya.

Teknis permainannya adalah memasukkan orang-orang ke perahu dengan cara mengklik salah satu orang. Apabila ingin menggerakkan perahu untuk menyebrang, klik pada tombol merah di samping perahu. Begitu juga sebaliknya apabila ingin mengembalikan perahu. Permainan Japanese River IQ Test ini dapat diselesaikan dengan algoritma Brute Force yaitu mencoba semua kemungkinan yang ada hingga. Dengan menggunakan algoritma Brute Force, solusi dari permainan ini pasti bisa didapatkan. Namun, karena algoritma Brute Force akan mencoba semua kemungkinan yang ada, maka dibutuhkan waktu yang cukup lama untuk mencari solusi yang tepat sehingga algoritma Brute Force jarang digunakan.

II. DASAR TEORI

A. Breadth First Search (BFS)

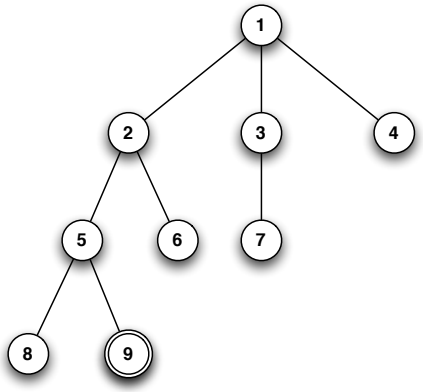
Algoritma *Breadth First Search* (BFS) atau pencarian melebar merupakan salah satu algoritma traversal dalam graf. BFS (*Breadth First Search*) adalah algoritma yang melakukan pencarian secara melebar yang mengunjungi simpul secara preorder yaitu mengunjungi suatu simpul kemudian mengunjungi semua simpul yang bertetangga dengan simpul tersebut terlebih dahulu. Dari sudut pandang algoritma, semua simpul anak didapatkan dengan memperluas simpul yang ditambahkan pada queue FIFO (*First In First Out*).

Pencarian selalu mengunjungi node-node pohon secara melebar, berawal dari level dengan depth 0 ke depth maximum. Hal ini dapat dinyatakan dalam suatu antrian. Tiap simpul yang dikunjungi masuk ke antrian hanya

untuk satu kali. Bentuk Algoritmanya adalah seperti berikut:

1. Kunjungi simpul v (apabila simpul v ingin dikunjungi pertama kali).
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.

Jika graf berbentuk pohor berakar, maka semua simpul pada aras *d* dikunjungi lebih dahulu sebelum mengunjungi simpul-simpul pada aras *d* + 1. Dibawah ini adalah merupakan contoh dari BFS dalam bentuk pohon pencarian graf.



Pencarian dimulai dari simpul 1 kemudian simpul 2 dan seterusnya hingga berakhir pada simpul 9 yang merupakan solusi pencarian. Dapat dilihat bahwa pencarian dilakukan per tingkat pada pohon pencarian. Pencarian ini akan berakhir apabila algoritma BFS telah mencapai solusi. Apabila solusi yang dicari berada pada simpul 8, maka simpul 9 tidak akan diperiksa karena solusi telah ditemukan.

Berikut di bawah adalah *presudo-code* dari algoritma BFS.

```
procedure BFS(input v:integer)
{ Traversal graf dengan algoritma pencarian BFS.

    Masukan: v adalah simpul awal kunjungan
    Keluaran: semua simpul yang dikunjungi dicetak ke layar
}
Deklarasi
    w : integer
    q : antrian;

    procedure BuatAntrian(input/output q : antrian)
    { membuat antrian kosong, kepala(q) diisi 0 }

    procedure MasukAntrian(input/output q:antrian, input
v:integer)
    { memasukkan v ke dalam antrian q pada posisi belakang }

    procedure HapusAntrian(input/output q:antrian,output
v:integer)
    { menghapus v dari kepala antrian q }

    function AntrianKosong(input q:antrian) ® boolean
    { true jika antrian q kosong, false jika sebaliknya }

Algoritma:
    BuatAntrian(q) { buat antrian kosong }

    write(v) { cetak simpul awal yang dikunjungi }
    dikunjungi[v]<-true { simpul v telah dikunjungi, tandai
                          dengan true}
    MasukAntrian(q,v) { masukkan simpul awal kunjungan ke dalam
                        antrian}

    { kunjungi semua simpul graf selama antrian belum kosong }
    while not AntrianKosong(q) do
        HapusAntrian(q,v) { simpul v telah dikunjungi, hapus
                            dari antrian }
    for w¬1 to n do
        if A[v,w] = 1 then { v dan w bertetangga }
            if not dikunjungi[w] then
                write(w) {cetak simpul yang dikunjungi}
                MasukAntrian(q,w)
                dikunjungi[w]¬true
            endif
        endif
    endfor
endwhile
{ AntrianKosong(q) }
```

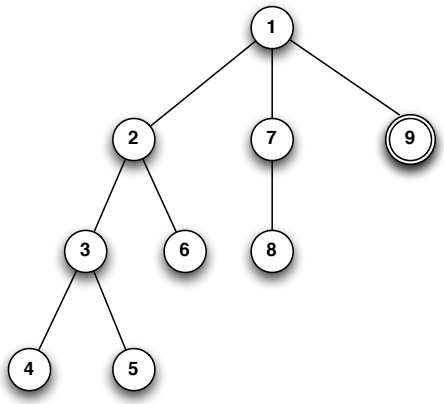
B. Depth First Search (DFS)

Algoritma DFS juga merupakan salah satu algoritma traversal dalam graf. DFS (*Depth First Search*) artinya adalah pencarian mendalam. *Depth-first search* adalah sebuah teknik pencarian dengan menelusuri titik yang terdalam dari sebuah pohon. Teknik ini mengunjungi seluruh simpul pada pohon yang ada terlebih dahulu tanpa melihat bobot yang ada pada masing-masing simpul. Setelah simpul pada bagian tertentu telah dikunjungi dan belum mendapatkan tujuan yang dikehendaki, maka akan dilakukan *backtracking* menuju simpul lainnya yang belum dikunjungi. Teknik ini dapat diimplementasi menggunakan pencarian berbasis pohon dengan antrian *last in first out* (LIFO) pada *stack* atau menggunakan fungsi rekursif.

Bentuk Algoritmanya adalah sebagai berikut:

1. Kunjungi simpul v (apabila simpul v ingin dikunjungi pertama kali).
2. Kunjungi simpul w yang bertetangga dengan simpul v.
3. Ulangi DFS mulai dari simpul w.
4. Ketika mencapai simpul u sedemikan sehingga semua simpul yang bertetangga dengannya telah dikunjungi, pencarian dirunut balik (*backtrack)* ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai simpul w yang belum dikunjungi.
5. Pencarian berakhir apabila tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi.

Dibawah ini adalah merupakan contoh dari DFS dalam bentuk pohon pencarian graf.



Pencarian dimulai dari simpul 1 kemudian simpul 2 dan seterusnya hingga berakhir pada simpul 9 yang merupakan solusi pencarian. Berbeda dengan algoritma BFS, dapat dilihat bahwa pencarian dengan algoritma DFS dilakukan secara mendalam mulai dari simpul paling kiri. Pencarian ini akan berakhir apabila algoritma DFS telah mencapai solusi. Apabila solusi yang dicari berada pada simpul 8, maka simpul 9 tidak akan diperiksa karena solusi telah ditemukan. Metode pencarian dengan menggunakan DFS dapat menjadi solusi yang lebih baik daripada BFS apabila simpul yang dicari berada di sebelah kiri pohon. Namun pencarian menggunakan BFS akan lebih unggul apabila simpul yang dicari berada di sebelah kanan.

Berikut di bawah adalah *presudo-code* dari algoritma BFS.

```
procedure DFS(input v:integer)
{Mengunjungi seluruh simpul graf dengan algoritma pencarian DFS

Masukan: v adalah simpul awal kunjungan
Keluaran: semua simpulyang dikunjungi ditulis ke layar
}

Deklarasi
    w : integer

Algoritma:
    write(v)
    dikunjungi[v]¬true
    for w¬1 to n do
        if A[v,w]=1 then {simpul v dan simpul w bertetangga }
            if not dikunjungi[w] then
                DFS(w)
            endif
        endif
    endfor
```

III. Analisis Pemecahan Masalah

Untuk menemukan solusi dari permainan Japanese River IQ Test dengan algoritma BFS dan DFS, maka permasalahan ini akan didefinisikan dengan sebuah pohon pencarian. Berikut adalah representasi dari orang-orang yang ada di permainan yang telah disesuaikan dengan struktur pohon pencarian yang akan dibangun.

1. Setiap orang yang ada di dalam permainan akan direpresentasikan dengan satu atau dua buah karakter. Ayah direpresentasikan dengan huruf A, ibu dengan huruf I, anak laki-laki dengan huruf L, anak perempuan dengan huruf P, polisi dengan huruf PL dan pencuri dengan huruf PN.
2. Kondisi awal permainan adalah semua orang berada di sebelah kiri sungai dan perahu juga berada di sebelah kiri.
3. Kondisi akhir permainan adalah semua orang berada di sebelah kanan sungai dan perahu juga berada di sebelah kanan.
4. Setiap simpul akan direpresentasikan seperti berikut

   {bagian kiri} {tempat perahu} {bagian kanan}
   Tempat perahu berada akan direpresentasikan dengan huruf R (Right) atau L (Left). Tiap orang akan dipisahkan dengan koma.

   Contoh:
   Kondisi awal permainan:
   {A, L, L, I, P, P, PL, PN} {L} {}
   Kondisi akhir permainan:
   {} {R} {A, L, L, I, P, P, PL, PN}

   Notasi {} menyatakan himpunan sehingga *{A, I} = {I, A}*.

5. Peraturan permainan telah disebutkan pada bab pendahuluan di atas. Berikut adalah contoh simpul yang salah.

   {L, L, I, P, P, PL, PN} {R} {A}
   Simpul ini salah karena disebelah kiri terdapat anak laki-laki yang bersama ibu tetapi tidak ada ayah disampingnya.
   {A, L, L, I, P, P, PN} {R} {PL}
   Simpul ini salah karena disebelah kiri terdapat pencuri yang bersama orang-orang tetapi tidak ada polisi disampingnya.

   {A, L, I, P, PL, PN} {R} {L, P}
   Simpul ini tidak mungkin terjadi karena anak laki-laki ataupun anak permpuan tidak dapat mengoperasikan perahu. Yang dapat mengoperasikan perahu hanya ayah, ibu dan polisi.

Dalam pembentukan pohon pencarian ini, terdapat beberapa asumsi dan batasan yaitu sebagai berikut.

1. Simpul yang berulang (sudah ada sebelumnya) tidak akan ditulis kembali.
2. Simpul salah atau tidak valid tidak akan dituliskan.
3. Dua anak laki-laki adalah sama. Artinya tidak ada perbedaan anak laki-laki yang mana yang akan disebrangkan lebih dulu. Begitu pula dengan dua anak perempuan.

Pencarian solusi akan dilakukan dengan algoritma DFS dan BFS.

A. Solusi dengan Breadth First Search (BFS)

Bentuk algoritma BFS yang digunakan adalah sebagai berikut.

1. Masukkan state awal ke dalam antrian.
2. Ambil antrian yang paling dulu dimasukkan.
3. Cek apakah sudah memenuhi state akhir atau tidak. Apabila telah memenuhi state akhir, kembalikan solusi. Apabila tidak memenuhi state akhir, maka masukkan state yang mungkin dicapai dari state tersebut ke dalam antrian (apabila ada dan urutan antrian yang dimasukkan secara random).
4. Cek antrian. Apabila antrian kosong, maka kembalikan solusi kosong. Apabila tidak kosong, kembail ke poin 2.

Berikut adalah pohon yang dihasilkan dari penggunaan algoritma BFS:

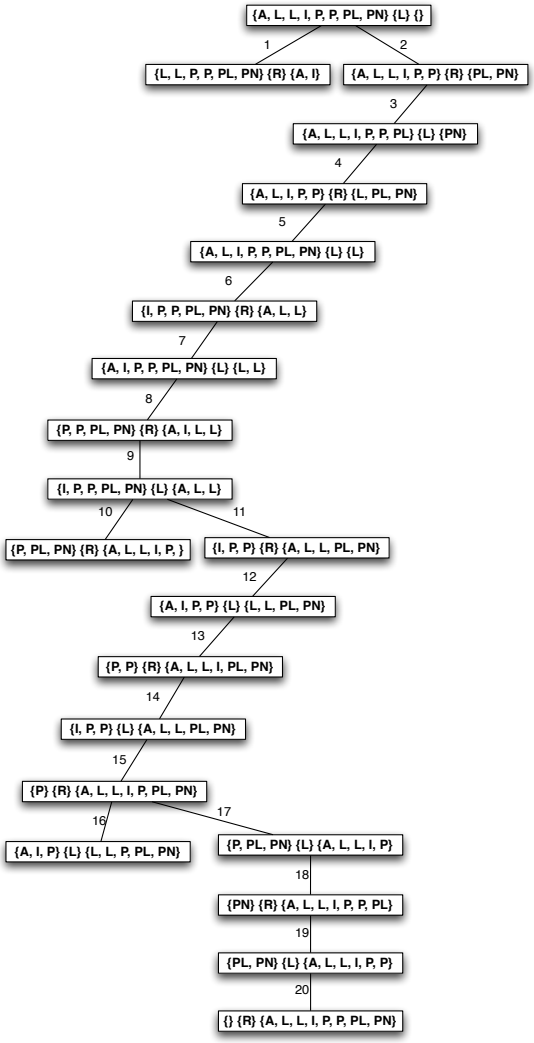
Jumlah penelusuran yang dilakukan oleh pohon algoritma BFS di atas adalah 46. Ini merupakan jumlah yang cukup banyak. Hal ini dikarenakan solusi yang cukup dalam dan BFS harus memeriksa tiap simpul secara melebar.

B. Solusi dengan Depth First Search (DFS)

Bentuk algoritma DFS yang digunakan adalah sebagai berikut.

1. Masukkan state awal ke dalam tumpukan.
2. Ambil tumpukan yang paling atas.
3. Cek apakah sudah memenuhi state akhir atau tidak. Apabila telah memenuhi state akhir, kembalikan solusi. Apabila tidak memenuhi state akhir, maka masukkan state yang mungkin dicapai dari state tersebut ke dalam tumpukan (apabila ada dan urutan tumpukan yang dimasukkan secara random).
4. Cek tumpukan. Apabila tumpukan kosong, maka kembalikan solusi kosong. Apabila tidak kosong, kembai ke poin 2.

Berikut adalah pohon yang dihasilkan dari penggunaan algoritma DFS:



Jumlah penelusuran yang dilakukan oleh pohon algoritma DFS di atas adalah 20. Jumlah ini jauh lebih sedikit dibandingkan dengan algoritma BFS. Hal ini dikarenakan solusi dari permasalahan lebih condong berada di kiri dan algoritma DFS akan mencari solusi mulai dari simpul paling kiri dan secara mendalam.

C. Analisis Solusi

Pada penggunaan algoritma BFS dan DFS diatas dapat dilihat bahwa kedua algoritma berhasil mendapatkan solusi dengan jumlah penelusuran algoritma BFS sebanyak 46 dan algoritma DFS sebanyak 20. Hal ini dikarenakan solusi yang berada condong di simpul kiri dan cukup dalam. Meskipun DFS telah mencapai solusi dengan hanya 20 kali penelusuran, hasil yang didapatkan seberanya masih belum minimum. State pertama, ke-10 dan ke-16 sebenarnya tidak perlu dilakukan. Namun karena DFS akan menelusuri simpul mulai dari kiri dan secara acak, maka state tersebut dilakukan. Begitu pula dengan BFS yang harus menelusuri simpul secara melebar sehingga banyak state yang tidak perlu dilakukan. Selain itu, apabila tidak dibuat asumsi simpul yang berulang dan simpul yang tidak valid tidak akan ditulis, maka pencarian dengan BFS dan DFS akan

membutuhkan penelusuran yang lebih banyak lagi dan akan lebih rumit.

IV. KESIMPULAN

Algoritma Breadth First Search (BFS) dan Depth First Search (DFS) merupakan contoh algoritma traversal dalam graf yang memodelkan sebuah pencarian dengan pohon pencarian. Algoritma tersebut dapat digunakan dalam pencarian solusi yang dalam hal ini adalah pencarian solusi permaian Japanese River IQ Test. Namun kedua algoritma tersebut tidak selalu bisa mendapatkan solusi secara cepat dan minimum.

Dari hasil analisis solusi di atas, didapatkan bahwa pencarian solusi dengan algoritma BFS membutuhkan penelusuran sebanyak 46 dan algoritma DFS sebanyak 20. Algoritma DFS tidak akan selalu lebih baik daripada algoritma BFS. Apabila solusi berada di simpul kanan dan solusi berada tidak terlalu dalam, maka algoritma DFS akan membutuhkan lebih banyak penelusuran daripada algoritma BFS.

Hasil di atas juga bukan merupakan hasil minimum. Berikut adalah hasil minimum yang seharusnya didapat dari pencarian solusi.

0. Kondisi awal
1. Polisi dan pencuri ke kanan sungai
2. Polisi kembali ke kiri sungai
3. Polisi dan anak laki-laki ke kanan sungai
4. Polisi dan pencuri kembali ke kiri sungai
5. Ayah dan anak laki-laki ke kanan sungai
6. Ayah kembali ke kiri sungai
7. Ayah dan ibu ke kanan sungai
8. Ibu kembali ke kiri sungai
9. Polisi dan pencuri ke kanan sungai
10. Ayah kembali ke kiri sungai
11. Ayah dan ibu ke kanan sungai
12. Ibu kembali ke kiri sungai
13. Ibu dan anak perempuan ke kanan sungai
14. Polisi dan pencuri kembali ke kiri sungai
15. Polisi dan anak perempuan ke kanan sungai
16. Polisi kembali ke kiri sungai
17. Polisi dan pencuri ke kanan sungai
18. Kondisi akhir

Pencarian solusi di atas hanya membutuhkan 17 kali penelusuran. Ini merupakan solusi minimum dari permainan Japanese River IQ Test. Namun, meskipun algoritma BFS dan DFS tidak selalu menghasilkan solusi minimum, algoritma ini dapat menjadi pilihan untuk mencari sebuah solusi dibandingkan dengan metode pencarian dengan algoritma brute force.

REFERENCES

[1] Rinaldi Munir, "Diktat Kuliah IF3051 Strategi Algoritma", Bandung: Penerbit ITB.

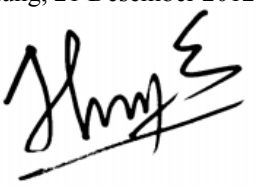
[2] "Japanese River IQ Test"
URL : http://www.arcadetemple.com/game/21345/Japanese-IQ-Test.html, diakses tanggal 20 Desember 2012 pukul 21.12 WIB

[3] "BFS dan DFS"
URL : http://www.docstoc.com/docs/80422515/DFS-dan-BFS, diakses tanggal 21 Desember 2012 pukul 00.09 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, 21 Desember 2012



Hanif Eridaputra, 13510091