Penerapan Algoritma BFS dan DFS pada Permainan Logika *Wolf, Sheep, and Cabbage*

Sandy Socrates / 13508044[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*if18044@students.if.itb.ac.id*

***Abstrak*—Dalam permainan logika *Wolf, Sheep, and Cabbage*. Diceritakan ada seorang petani yang hendak menyeberangi sungai membawa hasil belanjanya dari pasar, yaitu sekeranjang penuh kubis, juga seekor serigala dan seekor domba. Pemain diminta untuk menyebrangkan petani, serigala, domba, dan keranjang kubis menggunakan sebuah perahu yang hanya muat ditempati oleh dua penumpang. Yang dapat menggunakan perahu hanya sang petani. Permasalahnya adalah pada saat petani tidak ada, serigala akan memakan domba, dan domba akan memakan kubis.**

Salah satu algoritma yang bisa digunakan untuk memecahkan permasalahan ini adalah *Breadth First Search* dan *Depth First Search*. Dengan Kedua algoritma ini kita bisa memodelkan permasalahan ini ke dalam sebuah struktur pohon yang dapat dijelajah untuk menemukan jawabannya.

***Kata Kunci*— BFS, DFS, logic, Wolf Sheep and Cabbage**

I. PENDAHULUAN

Permainan logika *Wolf, Sheep and Cabbage* ini adalah permainan logika penyebrangan sungai yang cukup popular di dunia maya sebagai permainan pengisi waktu sekaligus pengasah otak. Permainan ini memiliki banyak sekali versi, perbedaannya hanya terdapat dalam nama hewan—hewan yang terdapat dalam cerita, versi lain yang cukup dikenal adalah *Fox, Goose, and bag of beans.* Walaupun terdapat perbedaan dalam hewan yang berpartisipasi dalam cerita ini, logika yang digunakan tetap sama bahwa A, B, dan C harus menyebrangi sungai dengan sebuah perahu yang hanya dapat menampung si manusia yang harus selalu naik untuk mendayung perahu dan salah satu dari A, B, dan C; sementara A dan B serta C tidak boleh ditinggalkan berdua dengan pasangan yang dapat membunuhnya.

Permainan ini mulai berkembang pada sekitar awal abad ke Sembilan dan telah menjadi bagian dari cerita daerah di berbagai tempat di dunia.

Disebutkan bahwa penggagas cerita permainan logika ini adalah Alcuin of York (735-804) dalam suratnya kepada salah seorang muridnya yaitu Charlemagne[1]. Adapun pengagas pasti dari permainan ini tidak perlu diperdebatkan, karena fakta yang lebih menarik adalah bahwa permainan ini telah didistribusikan ke seluruh penjuru dunia selama ratusan tahun melalui mulut ke mulut dan tulisan.

Gambar 1. Contoh permainan Wolf, Sheep, and Cabbage dalam format flash

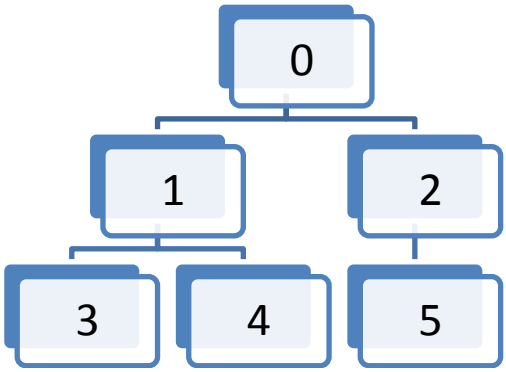
Gambar 2. Contoh tampilan mode permainan

Dalam permainan ini, kita bisa menggunakan algoritma *Breadth-First Search* (BFS) dan *Depth-First Search* (DFS) untuk mencari jawaban urutan pemindahan hewan-hewan tadi. BFS dan DFS adalah algoritma yang digunakan untuk mengunjungi simpul di dalam sebuah struktur pohon.

II. Algoritma Pencarian BFS dan DFS

A. Breadth-First Search

BFS adalah sebuah algoritma pencarian yang digunakan dalam sebuah struktur pohon. Pada algoritma ini pencarian dilakukan dimulai dari simpul akar, dan kemudian pencarian dilakukan secara menyebar sesuai dengan tingkat dari masing-masing simpul di dalam pohon.[2]



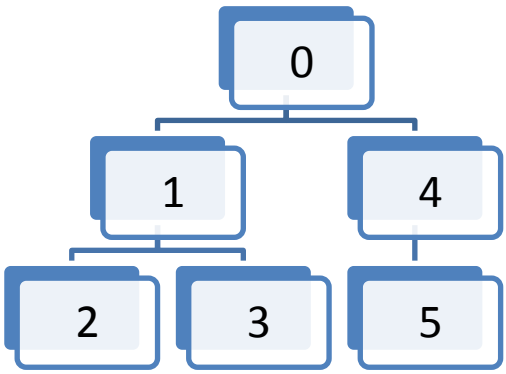
Gambar 3. Ilustrasi pencarian pada pohon dengan BFS

Kelebihan dari algoritma ini adalah jawaban yang dihasilkan selalu merupakan solusi optimal. Karena pencarian dilakukan dengan melihat ketinggian simpul.

Sayangnya pencarian optimal ini harus dibayar dengan jumlah memori yang diperlukan untuk komputasi yang akan banyak mengikuti jumlah simpul yang disimpan di memori.

B. Depth-First Search

DFS adalah sebuah algoritma pencarian yang digunakan dalam sebuah struktur pohon seperti BFS. Hanya saja pada algoritma ini setelah pencarian dilakukan di simpul akar,, pencarian kemudian dilakukan secara menurun sesuai urutan yang telah ditentukan (prioritas kiri ke kanan atau kanan ke kiri). Jika menemukan daun, pencarian dikembalikan ke simpul yang belum dikunjungi di atasnya mengikuti urutan tadi. [2]



Gambar 4. Ilustrasi pencarian pada pohon dengan DFS

Kelebihan DFS terletak pada jumlah memori yang diigunakan untuk melakukan komputasi sedikit, karena cukup satu simpul yang diingat untuk menjelajah isi pohon, yaitu simpul yang sedang digunakan.

Pencarian dengan Algoritma DFS juga memiliki kelemahan, yaitu solusi yang ditemukan tidak selalu optimal, karena yang didahulukan dalam pencarian adalah menuruni pohon.

III. Metode Pemecahan Masalah

Untuk memfasilitasi pemecahan masalah *Wolf, Sheep, dan Cabbage* menggunakan algoritma pencarian BFS dan DFS kita akan mendefinisikan permasalahan ini sebagai sebuah struktur pohon. Serta berikut adalah representasi permasalahan *Wolf, Sheep, and Cabbage* yang telah disesuaikan dengan struktur pohon yang dibangun:

1. Setiap *role* dalam permasalahan ini akan diwakilkan dengan sebuah karakter. Petani direpresentasikan dengan huruf F, serigala dengan huruf W, domba dengan huruf S, dan kubis dengan huruf C, kecuali perahu yang tidak perlu direpresentasikan karena sudah dapat diwakilkan oleh petani.
2. Kondisi awal permainan adalah *state* dengan semua *role* berada di sebelah kanan sungai.
3. Kondisi akhir permainan adalah *state* dengan semua *role* berada di sebelah kiri sungai tanpa ada satu pun *role* yang hilang karena dimakan.
4. Setiap *state* untuk *role* di sisi sungai disimpan ke dalam sebuah simpul dengan notasi berikut

   <{*role* di kiri}, {*role* di kanan}>*

   Contoh:
   Kondisi awal permainan
   <{}, { F,W,S,C }>
   Konsisi akhir permainan
   <{ F,W,S,C }, {}>

   *notasi { } menunjukkan sebuah himpunan, maka {F,W} = {W,F}

5. Petani yang membawa hewan dan barang akan dimasukkan ke dalam himpunan di mana sisi perahu menepi.
6. *State* yang terdapat salah satu dari {W,S} atau {S,C} akan dianggap tidak valid.

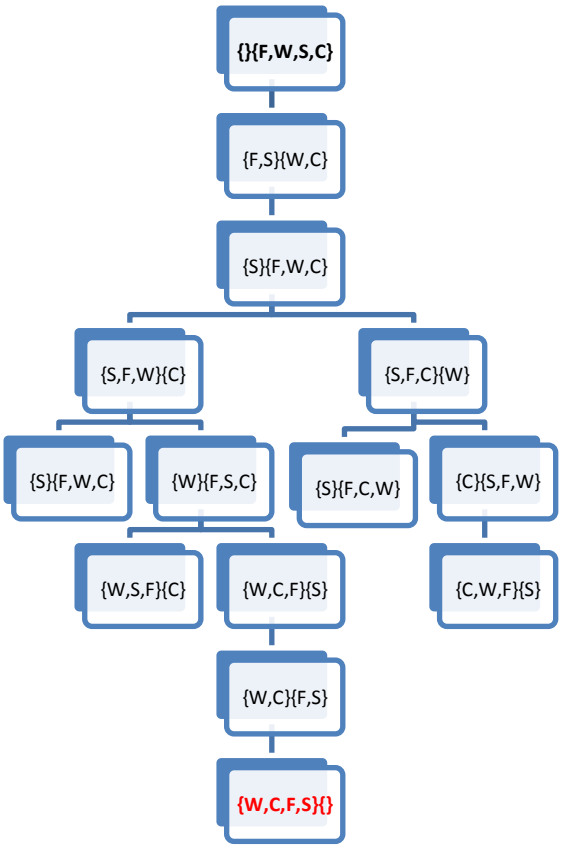
Berikut adalah batasan yang digunakan dalam pembangunan pohon,:

1. Simpul yang berulang akan digambarkan tetapi tidak akan diteruskan
2. Simpul yang tidak valid tidak akan digambarkan

Perlu diperhatikan bahwa pohon yang didapat pada bagian BFS dan DFS adalah pohon dengan simpul yang dilewati oleh pencarian BFS atau DFS

A. Pohon Ruang Status

Pohon minus state tidak valid yang dihasilkan jika dilakukan pencarian state secara manual.



Gambar 5. Pohon ruang status

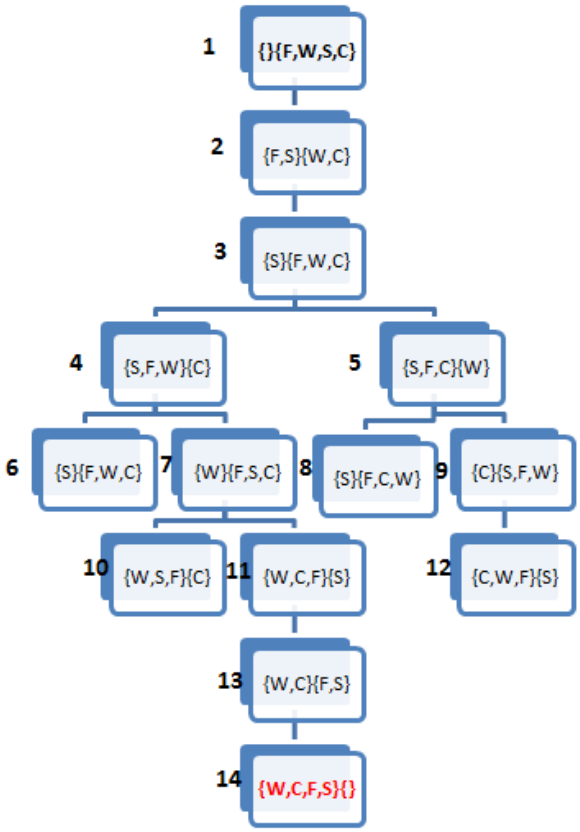
Dari pohon ruang status dapat dicari solusi pemecahan dengan menggunakan algoritma pencarian BFS atau DFS. Di atas terlihat ada sebuah status akhir. Kita akan melihat hasil pencarian dengan BFS dan DFS.

B. Pemecahan dengan BFS

Algoritma BFS yang digunakan :

1. Masukkan *state* awal ke dalam antrian
2. Cek apakah sudah memenuhi *state* akhir jika ya kembalikan solusi, jika tidak masukkan state yang mungkin dari state sebelumnya ke dalam antrian.
3. Cek antrian, jika kosong pencarian berakhir dengan dengan solusi kosong.
4. Kembali lagi ke 2.

Pada Gambar 5 ditunjukkan pohon hasil pencarian manual. Maka berikut adalah pohon yang dihasilkan dari pencarian dengan algoritma BFS



Gambar 6. Pohon ruang status dari pencarian BFS

Ternyata pohon yang didapatkan oleh BFS sama dengan pohon yang didapatkan dari pencarian manual. Keterangan angka pada gambar merupakan urutan pencarian yang dilakukan oleh BFS.

Jumlah penelusuran yang dilakukan oleh algoritma BFS adalah 14 (empat belas) kali. Pohon yang didapatkan sama dengan pohon dari pencarian manual.

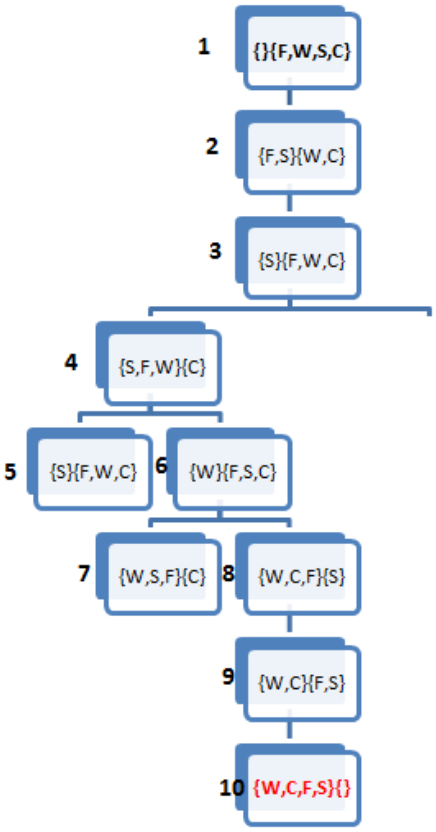
Jumlah state yang diperlukan untuk mencapai state akhir adalah delapan. Hasil yang didapatkan dari pencarian ini adalah optimal.

C. Pemecahan dengan DFS

Algoritma DFS yang digunakan :

- Masukkan *state* awal ke dalam tumpukan
- Cek apakah sudah memenuhi *state* akhir jika ya kembalikan solusi, jika tidak masukkan state yang mungkin dari state sebelumnya ke dalam antrian.
- Cek tumpukan, jika kosong pencarian berakhir dengan dengan solusi kosong.
- Kembali lagi ke 2.

Berikut adalah pohon yang dihasilkan dari pencarian dengan algoritma DFS dengan prioritas yang digunakan adalah kiri ke kanan



Gambar 7. Pohon ruang status dari pencarian DFS

Pohon yang didapatkan oleh DFS berbeda dengan pohon yang didapatkan dari pencarian manual dan pencarian BFS. Keterangan angka pada gambar merupakan urutan pencarian yang dilakukan oleh DFS.

Jumlah penelusuran yang dilakukan oleh algoritma DFS adalah tujuh kali.

Jumlah state yang diperlukan untuk mencapai state akhir adalah delapan. Hasil yang didapatkan dari pencarian ini sama dengan hasil yang didapatkan pencarian BFS, sehingga hasilnya adalah juga.

B. Penerapan solusi

Berikut adalah tampilan halaman yang permaianan yang selesai dengan solusi dari pencarian

Gambar 8. Tampilan antarmuka akhir permainan

IV. ANALISIS

Pada penggunaan algoritma pencarian DFS dan BFS di atas ditemukan hasil yang sama dan optimal. Tentu saja hal ini tidak selalu terjadi pada pencarian BFS dan DFS pada implementasinya.

Mengapa sampai terjadi kemunculan hasil yang sama untuk algoritma pencarian BFS dan DFS? Hal ini hanya akan terjadi pada saat *state* akhir yang dituju terletak di pinggiran dalam dari pohon yang dibentuk. Sehingga pada saat pencarian DFS tidak akan mencari terlalu jauh ke bawah.

Hal ini juga dipengaruhi oleh batasan yang diberlakukan yaitu pengulangan *state* tidak perlu dilanjutkan. Jika misalnya *state* tujuh pada pohon pencarian DFS dilanjutkann maka tentu DFS akan melanjutkan pencarian ke bawah dan hasil yang didapatkan berada di kedalaman yang lebih besar.

Hal lain yang mempengaruhi adalah prioritas yang digunakan dalam pemilihan simpul. Jika kita menggunakan prioritas kanan ke kiri untuk DFS maka rute yang dibangun akan berbeda dengan pohon pada Gambar 7.

V. KESIMPULAN

Algoritma pencarian *Breadth First Search* dan *Depth First Search* digunakan dalam penelusuran pohon, dan dengan memodelkan sebuah persoalan ke dalam sebuah pohon kita dapat mencari solusi yang mungkin dari persoalan tersebut menggunakan BFS dan DFS. Dan kita telah melihat bahwa permainan logika *Wolf, Sheep, and Cabbage* dapat diselesaikan dengan algoritma BFS dan DFS.

Berikut adalah tahapan yang dilakuan untuk mencapai *state* akhir. Baik BFS dan DFS memberikan solusi yang sama., yaitu:

1. Kondisi awal
2. Bawa domba ke kiri sungai
3. Kembali ke kanan
4. Bawa serigala ke kiri
5. Kembali ke kanan bersama domba
6. Bawa kubis ke kiri
7. Kembali ke kanan
8. Bawa domba ke kiri (selesai)

Pencarian dengan DFS dan BFS bisa mengembalikan solusi yang sama dan tentunya optimal. Hal ini dapat terjadi karena solusi terdapat pada sebelah pinggir pohon yang sesuai dengan prioritas yang digunakan.

Pengeliminasian *state* yang redundan terbukti dapat mengurangi jumlah pencarian dalam pohon pencarian dengan algoritma BFS dan DFS.

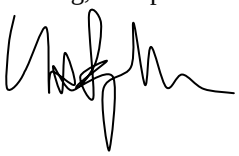
REFERENSI

[1] Ascher, Marcia (February 1990). "A River-Crossing Problem in Cross-Cultural Perspective". *Mathematics Magazine* (Mathematical Association of America) **63** (1): 26–29.doi:10.2307/2691506. Diakses 2010-12-8.
[2] Munir, Rinaldi. 2007. Strategi Algoritmik. Bandung : Teknik Informatika ITB.

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, 29 April 2010



Sandy Socrates
13508044