Analisis dan Penyelesaian Permainan *River Crossing Ultimate* Menggunakan Algoritma BFS dan DFS

Ira Aprilia
Program Studi Teknik Elektro, Fakultas Teknik, Universitas Panca Marga
Jl. Yos Sudarso 107 Pabean Dringu Probolinggo 67271
Email : ira.aprilia11@gmail.com

Terima Naskah : 16 September 2016
Terima Revisi : 28 September 2016

ABSTRAK

Permainan merupakan salah satu implementasi dari bidang komputer. Permaiana *River crossing ultimate* terdapat berbagai macam chapter. Dalam jurnal ini akan dibahas mengenai chapter 8 yaitu permainan yang membantu semua karakter dalam permainan menyeberangi sungai dengan cara yang paling optimal dengan menggunakan perahu yang hanya cukup untuk dua karakter yaitu seorang pria dan tas yang berisi uang atau dua pria dengan peraturan-peraturan tertentu. Terdapat dua jenis algoritma yang akan digunakan yaitu *Breadth First Search* (BFS) dan *Depth First Search* (DFS). Kedua algoritma ini termasuk algoritma traversal dalam graf. Oleh karena itu, solusi akan dibuat dalam bentuk pohon pencarian yang dapat dijelajah untuk menemukan jawabannya secara optimal.
Kata Kunci : River Crossing Ultimate Chapter 8, Breadth First Search dan Depth First Search.

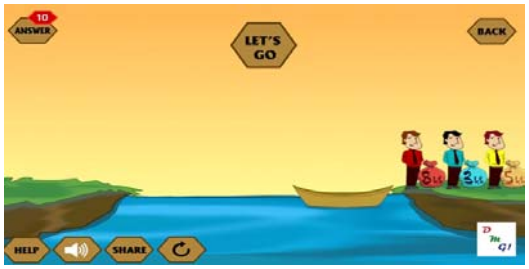
ABSTRACT

The game is one of the implementation of the computer field. Permaiana River crossing ultimate chapter there are various kinds. In this paper will be discussed on chapter 8 is a game that helps all the characters in the game to cross the river with the most optimal way by boat only enough for two characters are a man and a bag containing money or two men with specific regulations. There are two types of algorithms to be used, namely Breadth First Search (BFS) and Depth First Search (DFS). Both of these algorithms included in the graph traversal algorithms. Therefore, the solution will be made in the form of a search tree that can be explored to find the optimal answer.
Keywords : *River Crossing Ultimate Chapter 8, Breadth First Search dan Depth First Search.*

PENDAHULUAN

Perkembangan permainan pada masa kini sudah sangat pesat dan telah menjadi mode tersendiri di dunia karena mayoritas pengguna komputer menghabiskan sebagian besar waktu mereka di depan komputer dalam permainan komputer. Dalam permainan *River Crossing Ultimate Chapter* 8 ini diceritakan tiga orang dengan masing-masing membawa tas bawaannya yang berisi uang. Ketiga orang tersebut ingin menyebrangi sungai dengan sebuah perahu yang hanya cukup untuk dua karakter saja yaitu seorang pria dan tas berisi uang atau dua pria.

Pemain diminta untuk membantu menyebrangi ketiga orang tersebut dengan tas yang berisi uang secara korelatif pindah ke sisi lain sungai. Dengan catatan: Jika sewaktu-waktu di sisi sungai. Jumlah uang di dalam tas berisi uang yang lebih besar dari total nilai uang yang dimiliki tas pria ini. Maka pria-pria tersebut akan mencuri dan melarikan diri. Peraturan di atas berlaku untuk kedua sisi sungai. Berikut adalah tampilan awal permainannya diman posisi semua pria dan tas berisi uang berada di sisi sebelah kanan sungai beserta perahunya.



Gambar 1. Contoh tampilan awal permainan *River Crossing Ultimate Chapter 8.*

***Breadth First Search* (BFS)**

Breadth First Search (BFS) merupakan metode pencarian yang bertujuan untuk memperluas dan memeriksa semua node pada graph atau urutan kombinasi dengan pencarian secara sistematis melalui setiap solusi. BFS melakukan pencarian secara mendalam pada keseluruhan graph atau urutan tanpa memperhatikan tujuan sehingga menemukan tujuan tersebut. BFS tidak menggunakan algoritma heuristik. Adapun karakteristik yang dimiliki oleh Algoritma BFS yaitu:

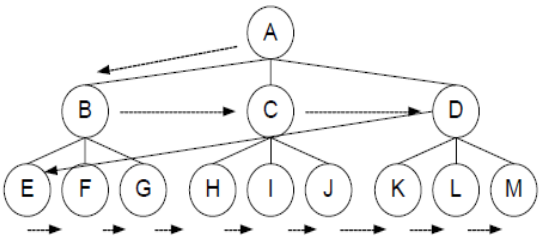
- Jika ada solusi, BFS akan menemukannya.
- BFS akan menemukan solusi dengan jalur terpendek.
- BFS tidak akan terjebak dalam "looping".
- BFS membutuhkan space untuk menyimpan node list antrian dan space yang dibutuhkan dan mungkin space yang dibutuhkan itu cukup besar.
- Asumsi :
  1. Ada solusi dalam pohon
  2. Pencarian tree adalah secara terurut.

BFS melakukan searching pada semua node yang berada pada level yang sama terlebih dahulu, sebelum melanjutkan proses searching pada node di level berikutnya.

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 node yang dikunjungi masuk ke antrian hanya untuk satu kali. Bentuk Algoritmanya adalah seperti berikut:

1. Idenya mirip dengan algoritma prim dan dijkstra.
2. Traversal dimulai dari node *v.*
   Algoritma :
   - Kunjungi node *v* (apabila node *v* ingin dikunjungi pertama kali).
   - Kunjungi semua node yang bertetangga dengan node v terlebih dahulu.
   - Kunjungi node yang belum dikunjungi dan bertetangga dengan node-node yang tadi dikunjungi, demikian seterusnya.

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



Gambar 2. Diagragm pohon dari BFS

Pencarian dimulai dari node A kemudian B, C dan seterusnya hingga berakhir pada node M yang merupakan solusi pencarian. Dapat dilihat bahwa pencarian dilakukan pertingkat pada pohon pencarian. Pencarian ini akan berakhir apabila algoritma BFS telah mencapai solusi akhir. Sedangkan apabila target yang dicari berada pada node L, maka node M tidak akan diperiksa karena solusi dari target telah ditemukan.

Kelebihan dari algoritma ini adalah jawaban yang dihasilkan selalu merupakan solusi optimal. Karena pencarian dilakukan dengan melihat ketinggian node. Sayangnya pencarian optimal ini harus dibayar dengan jumlah memori yang diperlukan untuk komputasi yang akan banyak mengikuti jumlah node yang disimpan di memori.

Depth-First Search (DFS)

Depth-first search (DFS) adalah proses searching sistematis buta yang melakukan ekpansi sebuah path (jalur) menuju penyelesaian masalah sebelum melakukan ekplorasi terhadap path yang lain. Proses searching mengikuti sebuah path tunggal sampai menemukan goal atau dead end. Apabila proses searching menemukan dead-end, DFS akan melakukan penelusuran balik ke node terakhir untuk melihat apakah node tersebut memiliki path cabang yang belum dieksplorasi.

Hanya saja pada algoritma ini setelah pencarian dilakukan di node 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 node yang belum dikunjungi di atasnya mengikuti urutan tadi.

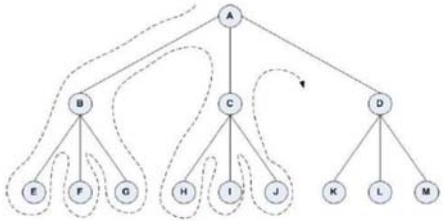
1. Traversal dimulai dari node *v.*
2. *Algoritma :*
   - Kunjungi node *v.*

- Kunjungi *w* yang bertetangga dengan node *v*.
- Ulangi DFS mulai dari node *w*.
- Ketika mencapai node *u* sedemikian sehingga semua node yang bertetangga dengannya telah dikunjungi sebelumnya dan mempunyai node *w* yang belum dikunjungi.
- Pencarian berakhir bila tidak ada lagi node yang belum dikunjungi yang dapat dicapai dari yang telah dikunjungi.

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

Berikut adalah contoh representasi DFS dalam bentuk pohon pencarian graf.



Gambar 3. Diagram pohon dari DFS

METODE

Untuk menyelesaikan pemecahan masalah pada permainan River *Crossing Ultimate chapter 8* dengan menggunakan algoritma pencarian BFS dan DFS kita akan mendefinisikan permasalahan ini sebagai sebuah struktur pohon. Berikut adalah representasi permasalahan yang telah disesuaikan dengan struktur pohon yang dibangun:

1. Setiap peraturan dalam permasalahan ini akan diwakilkan dengan sebuah karakter. Pria berpakaian berwarna merah direpresentasikan dengan huruf M, Pria berpakaian berwarna Biru dengan huruf B, Pria berpakaian berwarna kuning dengan huruf K, dan tas berisi uang delapan juta direpresentasikan sebagai angka 8, tas berisi uang tiga juta dengan angka 3 dan tas berisi uang lima juta dengan angka 5.
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 dicuri karena melarikan diri.
4. Setiap *state* untuk *role* di sisi sungai disimpan ke dalam sebuah node dengan notasi berikut :
   <{*role* di kiri}, {*role* di kanan}>
   Contoh representasi pada kondisi awal dan akhir permainan :
   Kondisi awal permainan:
   <{}, { M,8,B,3,K,5}>
   Konsisi akhir permainan:
   <{M,8,B,3,K,5}, {}>
   notasi {} menunjukkan sebuah himpunan, maka {M,8} = {8,M} dan Tiap pria dan tas yang berisi uang akan dipisahkan dengan koma.
5. Pria yang membawa tas berisi uang dan pria yang lain akan dimasukkan ke dalam himpunan di mana sisi perahu menepi.
6. *State* yang terdapat salah satu dari {8,K}, {8,B} dan {5,B} akan dianggap tidak valid dikarenakan jumlah uang di tas lebih besar dari total nilai uang yang dimiliki tas pria ini. Maka pria-pria tersebut akan mencuri dan melarikan diri.

Terdapat batasan yang digunakan dalam pembangunan pohon sebagai berikut :

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

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

A. Pemecahan dengan BFS

Algoritma BFS yang digunakan :

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.
5. Apabila tidak kosong, maka akan kembali lagi ke poin 2.

B. Pemecahan dengan DFS

Algoritma DFS yang digunakan :

1. Masukkan *state* awal ke dalam tumpukan
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 tumpukan, jika kosong pencarian berakhir dengan dengan solusi kosong.
4. Kembali lagi ke poin 2.

HASIL DAN PEMBAHASAN

Pada Penyelesaian permainan *River Crossing Ultimate chapter 8* dengan menggunakan 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. Hal ini dapat terjadi karena solusi yang berada condong di node kiri dan cukup dalam dan melebar untuk algoritma BFS sedangkan pada algoritma DFS meskipun telah mencapai solusi optimal ternyata hasil yang didapatkan mempunyai penulusuran solusi yang sama yaitu dua belas kali. Namun karena DFS akan menelusuri node mulai dari kiri dan secara acak, maka *state* tersebut dilakukan.

Begitu pula dengan BFS yang harus ditelusuri node secara melebar sehingga banyak state yang tidak perlu dilakukan. 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 node. Selain itu, apabila tidak dibuat asumsi node yang berulang dan node yang tidak valid tidak akan ditulis, maka pencarian dengan BFS dan DFS akan membutuhkan penelusuran yang lebih banyak lagi dan akan lebih sulit lagi.

Berikut adalah tampilan halaman permaianan hasil akhir dengan posisi tiga pria dan tas bawaannya serta perahunya berada di sebelah kiri sungai.

Gambar 6. Tampilan akhir penyelesaian permainan *River Crossing Ultimate chapter 8.*

SIMPULAN

Permainan *River Crossing Ultimate Chapter 8* ini dapat dicari solusinya dengan menggunakan algoritma BFS dan DFS. Algoritma pencarian ini digunakan dalam penelusuran pohon dan dengan memodelkan sebuah persoalaan ke dalam sebuah pohon.

Namun kedua algoritma yang digunakan tidak selalu bisa mendapatkan solusi secara cepat dan minimum. Tapi pada kali ini, kedua algortitma yang digunakan dalam permainan ini ternyata telah mencapai solusi optimal ternyata hasil yang didapatkan mempunyai penulusuran solusi yang sama yaitu dua belas kali.

DAFTAR PUSTAKA

[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.

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

[4] Syarif, Iwan. Pengantar ALgoritma & Pemograman. Politeknik Elektronika Negeri Surabaya (ITS). 2004