Penerapan Algoritma BFS/DFS dalam Logic Game: Crossing the Bridge

Rikysamuel/13512089
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13512089@std.stei.itb.ac.id

***Abstract*—Melalui makalah ini, dibahas satu topik dari Strategi Algoritma, yaitu Algoritma Pencarian Melebar dan Algoritma Pencarian Mendalam. Topik tersebut kemudian dikaitkan dalam pencarian solusi dari suatu permasalahan dalam sebuah permainan logika. Permainan logika yang dimaksud tersebut adalah Crossing the Bridge. Dari permainan ini, akan dicari solusinya dengan kedua algoritma yang telah disebutkan sebelumnya.**

***Index Terms*—BFS, DFS, Cross Bridge, Logic game,**

I. PENDAHULUAN

Siapa yang tidak mengenal Crossing the Bridge problem. Permainan logika ini sudah sangat terkenal. Permainan yang tujuannya adalah untuk menyebrangkan sejumlah orang dari sisi "kanan" ke sisi "kiri". Tapi untuk menyebrangkan sejumlah orang tersebut, ada batas waktu tertentu sebelum lampu lentera yang digunakan untuk menyebrang kehabisan bahan bakarnya.

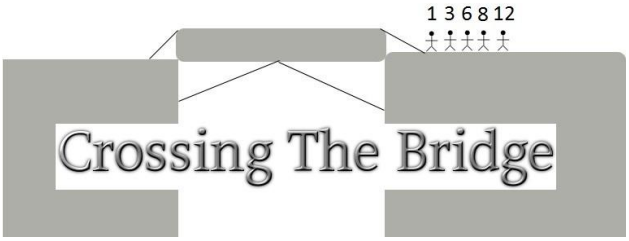
Permainan ini juga kerap kali dikenal sebagai Midnight Train atau Dangerous Crossing.

Jika lampu lentera kehabisan bahan bakarnya, maka jalan menjadi gelap dan tidak kelihatan. Jika jalan gelap, terlalu beresiko jika melanjutkan perjalanan dan misi kemudian dianggap gagal.

Dalam makalah ini, ditentukan jumlah penyebrang adalah 5 orang. Anggap saja kelima orang ini memiliki nama A,B,C,D, dan E. Dan kelima orang tersebut masing-masing dari A ke E memiliki waktu untuk menyebrang selama 1 menit, 3 menit, 6 menit, 8 menit, dan 12 menit. Sementara waktu paling lama untuk lampu lentera dapat menyala adalah hanya 30 menit.

Kelima orang tersebut harus dapat menyebrang sebelum lampu lentera tersebut kehabisan bahan bakar. Untuk dapat menyebrangkan semua orang ke sebrang, tiap kali menyebrang harus minimal ada 2 orang yang menyebrang. Sehingga salah satu dari yang menyebrang tersebut dapat kembali lagi ke tempat semula untuk kemudian mengoper lampu lentera tersebut kepada temannya yang lain.

Aturan lainnya adalah jembatan hanya dapat digunakan oleh maksimal 2 orang karena jembatan tersebut sudah sangat tua dan rapuh.



Gambar 1 : Crossing the Bridge

Untuk menyelesaikan permasalahan Crossing the Bridge ini tentu banyak caranya. Banyak teknik dan metode yang dapat dilakukan untuk dapat menyelesaikan masalah Crossing the Bridge. Tapi dalam makalah ini, akan dibahas cara penyelesaikan permasalah Crossing the Bridge ini melalui cara/metode Algoritma Pencarian Melebar *(Breadth First Search)* atau dengan cara Algoritma Pencarian Mendalam *(Depth First Search)*.

Dengan kedua algoritma ini, permasalahan harus dapat disederhanakan agar dapat dibuat menjadi simpul-simpul. Hal tersebut perlu dilakukan karena kedua algoritma tersebut mencari solusi dari suatu persoalan dengan mengunjungi simpul-simpul dalam suatu graf.

II. TRAVERSAL GRAF

Ada banyak cara untuk mencari solusi pada suatu graf. Biasanya tujuan yang ingin kita capai dari suatu permasalahan dari graf adalah mencapai suatu simpul atau memunculkan seluruh simpul dari sebuah simpul akar yang diberikan. Solusi yang dihasilkannya pun bisa merupakan solusi yang optimal atau tidak.

Algoritma yang digunakan itulah disebut algoritma traversal dalam graf. Algoritma traversal ini sendiri ada beberapa macam, yaitu *Tricolor Algorithm, Breadth-First Search, Depth-First Algorithm, Cycle Detection, Topological Sort,* dan *Connected Component.*

Yang kemudian akan dijelaskan lebih rinci ialah *Breadth-First Search Algorithm* dan *Depth-First Search Algorithm.* Tapi akan sedikit dibahas seluruh algoritma sebagai pelengkap saja

1. *Tricolor Algorithm*

"In this algorithm, graph nodes are assigned one of three colors that can change over time" – Dikutip dari http://www.cs.cornell.edu/courses/CS2112/2012sp/lectures/lec24/lec24-12sp.html

Dengan algoritma ini, tiap simpul pada graf diberi warna tertentu(putih, abu, hitam) dan dapat berubah-ubah seiring waktu.

2. *Topological Sort*

"the nodes of an acyclic graph are placed in an order consistent with the edges of the graph" – Dikutip dari http://www.cs.cornell.edu/courses/CS2112/2012sp/lectures/lec24/lec24-12sp.html

Dengan algoritma ini, simpul-simpul dari graf yang asiklik ditempatkan sesuai urutan yang konsisten dari sisi graf.

3. *Cycle Detection*

Algoritma ini menjalankan algoritma DFS untuk mengecek apakah suatu graf merupakan graf siklik atau bukan.

4. Connected Component

Algoritma ini digunakan hanya jika ada simpul pada suatu graf yang tidak terhubung.

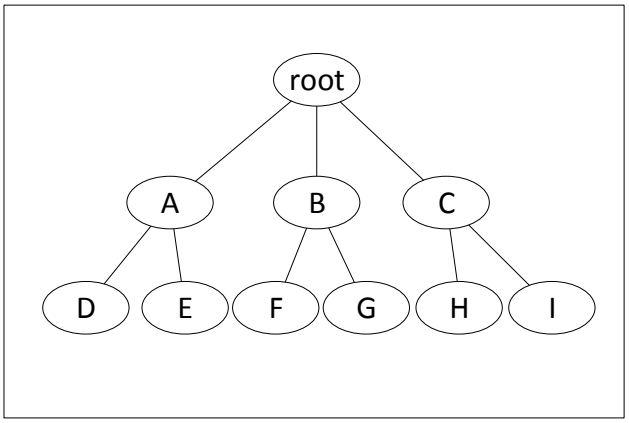
5. Breadth-First Search/BFS

BFS sendiri adalah sebuah teknik/algoritma pencarian yang jalan mengunjungi simpulnya sendiri ialah secara melebar. Maksudnya adalah, simpul-simpul yang memiliki aras yang sama akan dibangkitkan terlebih dahulu. Kemudian jika setelah semua simpul pada aras yang sama telah bangkit semua, baru dibangkitkan anak simpulnya.

Cara pembangkitan simpulnya pun sama. Yaitu membangkitkan semua simpul lagi yang memiliki aras yang sama. Pembangkitan simpul akan terus dilakukan selama simpul-simpul belum dikunjungi.

Algoritma BFS mempunyai beberapa keuntungan, yaitu tidak akan pernah "hilang" saat menjelajah simpul[2]. Lalu BFS juga akan dapat cepat mencapai suatu solusi jika memang ada solusinya dalam suatu permasalahan dan bisa menemukan solusi yang optimal[2]. Oleh karena itu solusi dapat ditemukan dengan langkah yang lebih sedikit[2].

Algoritma ini dalam "penjelajahannya" butuh mengingat keseluruhan simpul. Sehingga kerugian dari algoritma ini adalah butuh jumlah memori yang bergantung dari jumlah simpul yang dibangkitkan[2]. Selain itu, jika solusi yang didapatkan berada jauh dari akar, maka akan dibutuhkan waktu yang tidak sebentar juga untuk mencapai solusi tersebut[2].



Gambar 2 : Algoritma BFS

6. Depth-First Search/DFS

Pada algoritma pencarian mendalam, simpul-simpul tidak dibangkitkan secara per-aras seperti pada BFS. Tapi pembangkitan dilakukan terus dilakukan dari satu simpul, turun ke simpul anaknya.

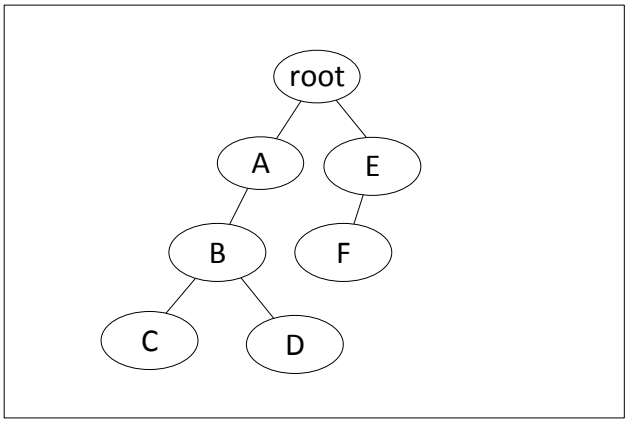
Kemudian jika kemudian tidak ditemukan solusi, akan dilakukan *backtracking* menuju simpul lain yang "belum" memiliki anak-simpul atau yang anak-simpul nya belum keluar semua. Algoritma pencarian mendalam kemudian akan diulang kembali terhadap simpul yang bersangkutan.

Keuntungan menggunakan algoritma ini adalah penggunaan memorinya yang tidak besar jika dibandingkan dengan BFS. DFS menggunakan *stack of nodes* untuk menyimpan langkahnya hanya dari akar ke simpul saat ini[3]. Selain itu juga, dengan DFS, solusi dari suatu permasalahan dapat langsung ditemukan tanpa harus membangkitkan lebih banyak simpul[3].

Tapi metode DFS ini juga memiliki kerugiannya sendiri juga. Algoritma ini ada kemungkinan bisa "hilang" ditengah penjelajahan. Maksudnya struktur pohon yang dihasilkan dari metode pencarian ini bisa saja tak hingga walaupun grafnya sendiri berhingga[3].

Solusi dari permasalahan tersebut adalah dengan membatasi kedalaman penjelajahan. Tapi untuk memberi batasan yang ideal, sebaiknya diberi kedalaman 'd' dimana d adalah kedalaman dari solusi. Jika 'd' terlalu kecil, maka penelusuran tidak akan pernah mencapai solusi. Dan sebaliknya juga jika 'd' terlalu besar, waktu akan banyak terbuang dalam mencari sesuatu yang tidak penting[3].

Selain itu, metode DFS juga tidak menjamin solusi yang dihasilkan adalah solusi yang optimal. Dan jika terdapat lebih dari satu solusi, tidak dijamin pula solusi yang didapat adalah solusi minimal[3].



Gambar 3 : Algoritma DFS

III. PEMECAHAN MASALAH

Untuk dapat menyelesaikan permasalahan Crossing the Bridge, harus terlebih dahulu membuat pendefinisan yang lebih rinci terhadap permasalahan dan membuat batasan-batasan agar tiap langkah yang diambil untuk menuju solusi tidak menjadi banyak dan sulit.

Menggunakan algoritma traversal graf, berarti harus didefinisikan dahulu simpul/state-nya. Berikut adalah pendefinisian dari masalah Crossing the Bridge:

1. Lima orang dengan waktu menyebrang : A(1), B(3), C(6), D(8), dan E(12). Seluruhnya harus dapat menyebrang dalam waktu paling lama dalam 30 satuan waktu.
2. Setiap orang yang belum menyebrang berada dalam suatu penampung NCR. Oleh karena itu sebagai initial, NCR [] ={A,B,C,D,E}.
3. Setiap orang yang sudah menyebrang berada dalam suatu penampung CR. Oleh karena itu sebagai initial, CR [] = {}.
4. Misinya adalah untuk memindahkan seluruh isi NCR kedalam CR.
5. Proses penyebrangan hanya boleh dilakukan oleh maksimal dua orang dalam satu waktu. Itu artinya tidak boleh ada yang menyebrang sebanyak 3 orang atau bahkan lebih.
6. Waktu akan berkurang sebanyak waktu orang yang paling lama menyebrang

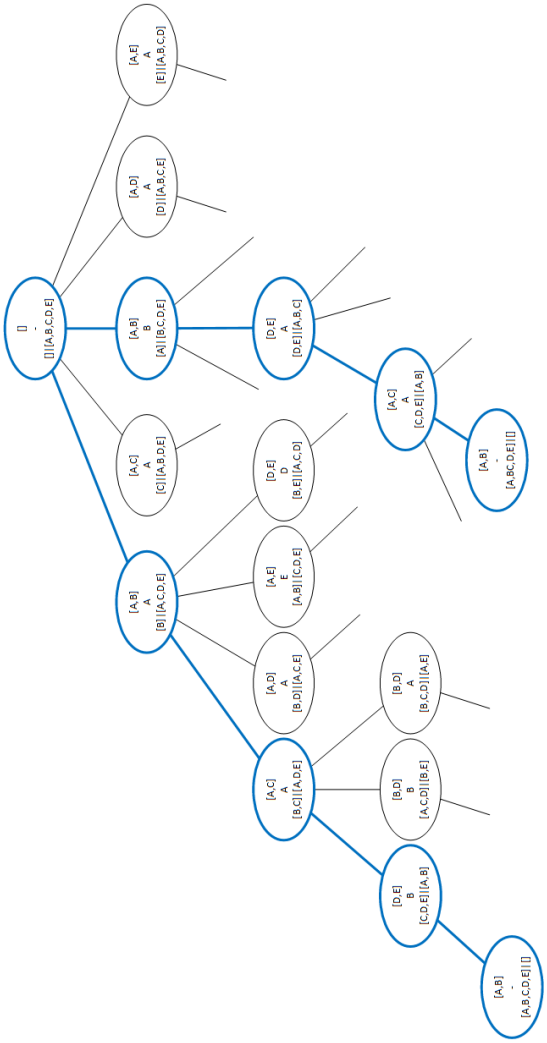
Untuk pembatasannya sendiri dalam algoritma BFS/DSF adalah:

1. Tidak seluruh simpul akan digambarkan. Hanya simpul-simpul yang menuju simpul solusi yang akan digambar, dan beberapa simpul lain sebagai penjelas/pelengkap
2. Jika ada simpul yang berulang, tidak akan diteruskan.

A. Solusi dengan BFS

Jika menggunakan BFS, berarti harus mengunjungi dahulu seluruh simpul yang memiliki aras yang sama. Notasi pada simpul []-var-[..]|[..] dimana [] pertama menunjukan "siapa" yang sedang menyebrang, diikuti var-menunjukan siapa yang kembali dan [] berikutnya yang sebelah kiri menandakan CR dan [] sebelah kanan menandakan NCR.

Demikian pohon ruang status dengan metode BFS :



Gambar 4 : Solusi Crossing Bridge dengan BFS

Jka dilihat dari gambar, bagian yang berwarna biru merupakan solusi. Jadi kita dapat solusi pertama yaitu
[A,B] – A kembali
[A,C] – B kembali
[D,E] – A kembali
[A,B]

Pembahasan

1. Akar

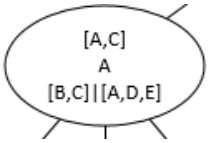
Pemecahan masalah dimulai dari root yang menunjukan bahwa belum ada aksi yang dilakukan. Belum ada yang menyebrang, ditulis di [] yang paling atas. Lalu tidak ada yang kembali juga dan CR dan NCR berada pada posisi initial.

2. Aras-1

Pada aras ini sudah dimulai proses penyebrangan. Terlihat pada state,

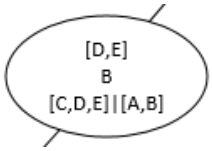
yang menyebrang adalah A(1) bersama B(3). Lalu kemudian A kembali lagi bersama C,D, dan E. Sampai dengan aras ini, waktu yang sudah dipakai untuk A dan B adalah sebanyak 4 satuan waktu (A&B kemudian A).

3. Aras-2



Pada aras ini kemudian dilanjutkan kembali proses penyebrangan. Saat ini A menyebrang bersama C. Lalu A kembali lagi bersama D dan E. Waktu yang dihabiskan sampai dengan simpul ini adalah 4 + 7 satuan waktu(A&C kemudian A). Total waktu 11 satuan waktu.

4. Aras-3



Pada aras ini proses penyebrangan dilanjutkan lagi. Sekarang D dan E yang menyebrang. Lalu kemudian B kembali menjemput A. Waktu yang dihabiskan sampai dengan simpul ini adalah 11 + 15 satuan waktu(D&E kemudian A). Total waktu 26 satuan waktu.

5. Aras-4

Pada aras ini proses penyebrangan yang dilakuka adalah yang terakhir. Waktu yang digunakan untuk menyebrang adalah 26+3 satuan waktu(A&B). Sehingga total waktu menjadi 29 satuan waktu.

Demikianlah solusi pertama yang ditemukan dari BFS dengan total waktu 29 satuan waktu, yaitu:
[A,B]-A, [A-C]-A, [D,E]-B, dan [A,B]

Terlihat juga dengan algoritma BFS, dimungkinkan ditemukannya solusi yang lebih dari satu. Kedua hasil ini kebetulan berada pada aras yang sama. Jika dihitung juga, sisa waktu yang dihasilkan dari solusi kedua ini pun sama seperti solusi pertama, yaitu 29 satuan waktu. Dan urutan langkah dari solusi kedua ini adalah :
[A,B]-B, [D,E]-A, [A,C]-A, dan [A,B]

Kedua solusi ini menghasilkan waktu 29 satuan waktu < 30 satuan waktu. Sehingga misi dinyatakan berhasil.

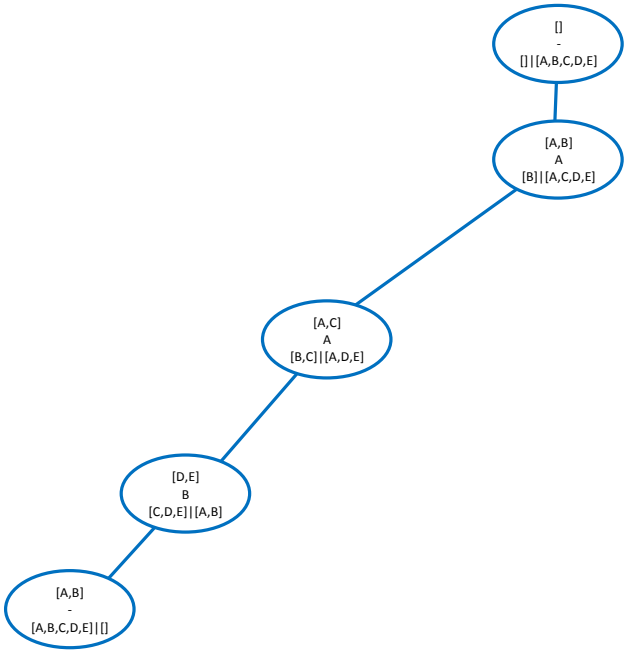
B. Solusi dengan DFS

Jika pada BFS, pembentukan ruang status adalah dengan mengunjungi terlebih dahulu simpul-simpul yang memiliki aras yang sama, maka pada DFS penelusuran akan dilakukan kedalam terlebih dahulu.

Pendefinisian persoalan sama seperti pendefinisian pada BFS. Pendefinisian ruang status juga sama menggunakan penampung NCR, CR, dan status penyebrang. Hanya ditambah satu batasan untuk DFS, yaitu pencarian akan dihentikan jika telah ditemukan satu solusi.

Selain itu, harus ditentukan juga batas kedalaman daripada DFS, agar tidak mencari terus kedalam dan malah membuang-buang waktu. Kedalaman yang efisien adalah kedalaman dimana solusi dapat ditemukan. Dari BFS kita tahu kalau kedalaman solusi adalah saat d=4(akar memiliki d=0). Oleh karena itu, didefinisikan kedalaman untuk permasalahan ini adalah d=4.

Penggambaran pohon ruang status menjadi:



Gambar 5 : Solusi Crossing the Bridge dengan DFS

Penjelasan mengenai masing-masing simpul pada DFS sama seperti penjelasan sebelumnya, pada BFS. Pertama-tama dahulukan A dan B untuk menyebrang. Untuk menyebrangkan A dan B dibutuhkan waktu 3 satuan waktu. Kemudian kembalikan A ke bagian NCR, membutuhkan waktu 1 satuan waktu. Total waktu adalah 4 satuan waktu

Lalu pilih A dan C untuk menyebrang selanjutnya. Untuk meneyebrangkan keduanya, membutuhkan waktu selama 6 satuan waktu. Kemudian A kembali lagi ke NCR, membutuhkan waktu 1 satuan waktu lagi. Total waktu menjadi 11 satuan waktu.

Kemudian saatnya menyebrangkan 2 orang yang paling lama, yaitu D dan E. Untuk menyebrangkan keduanya membutuhkan waktu selama 12 satuan waktu. Lalu kemudian B kembali ke NCR menjemput A, membutuhkan waktu 3 satuan waktu. Sehingga total waktu saat ini 26 satuan waktu.

Langkah yang paling terakhir adalah untuk menyebrangkan kedua orang yang tersisa di NCR yaitu A dan B. Untuk menyebrangkan keduanya, dibutuhkan waktu 3 satuan waktu. Sehingga total waktu saat ini ialah 29 satuan waktu.

Kelima orang tersebut dapat dipindahkan seluruhnya kebagian NCR tanpa kehabisan waktu.

Solusi menghasilkan waktu 29 satuan waktu < 30 satuan

waktu. Sehingga misi penyebrangan ini dikatakan berhasil.

IV. PEMBANDINGAN BFS DAN DFS

Dalam permasalahan crossing the bridge, dapat diselesaikan dengan algoritma traversal graf. Algoritma yang lebih ditekankan adalah algoritma BFS/DFS. Telah dibahas pada bagian sebelumnya bahwa baik menggunakan BFS maupun DFS, keduanya dapat mencapai solusi.

Kedua algoritma tersebut memiliki nilai plus/minus masing-masing. Jika diperhatikan, untuk mencapai simpul solusi pada BFS, diperlukan untuk melakukan pembangkitan simpul yang sangat banyak. Pembangkitan simpul menjadi sangat banyak karena pada metode BFS, pembangkitan dilakukan per level terlebih dahulu.

Karena pembangkitan per level inilah, menajdi dimungkinkan untuk dapat menemukan adanya lebih dari satu solusi. Tapi tentunya sebagai akibatnya, perlu diingat setiap simpul. Semakin banyak jumlah simpul yang dibangkikan, semakin besar memori yang dibutuhkan untuk menyimpan state-state tersebut.

Untuk mempermudah pembuatan, biasanya dalam representasi BFS, dalam mengingat simpul, jenis penampung yang digunakan adalah penampung queue. Sehingga perancang memiliki queue of state.

Sedangkan pada DFS, dapat dilihat untuk mendapatkan solusi dari permasalahan ini, pergerakannya tidak bertele-tele. Langsung menuju sasaran. Itu dikarenakan arah gerak pertamanya langsung kearah simpul yang menuju solusi. Jika simpul yang menuju simpul solusi tersebut berada di simpul terakhir yang dibangkitkan, maka jumlah simpul yang dibangkitkan oleh DFS pun tidak akan beda jauh dari BFS.

Waktu untuk mencapai solusinya pun tidak akan berbeda jauh dari BFS. Itu untuk kasus yang tidak optimalnya. Selain itu juga, jumlah kedalaman juga menjadi salah satu factor yang sangat menentukan. Juga sudah dibahas sebelumnya jika kedalaman dair DFS terlalu dangkal, maka algoritma tidak akan pernah dapat menemukan solusinya. Dan jika kedalaman yang diberikan terlalu jauh, maka algoritma akan terlalu sia-sia untuk dibiarkan "berputar-putar".

Apalagi jika algoritma DFS yang digunakan, tidak diberikan batas kedalaman. Membutuhkan waktu yang sangat lama untuk akhirnya dapat mencapai solusi. Atau malah algoritma tersebut tidak akan pernah sampai pada solusi atau dengan kata lain menghasilkan pohon tak hingga.

Kemungkinan untuk hilang itulah yang sangat dihindari. Sehingga penting untuk menentukan sebuah kedalaman. Telah dibahas juga sebelumnya bahwa pada kedalaman yang ditentukan agar menghasilkan solusi yang optimal ialah kedalaman 'd' dimana solusi ditemukan,

Sehingga jika tidak ditemukan solusi pada kedalaman 'd', maka pencarian mendalam terhadap suatu simpul dipotong, dan kemudian melakukan *backtracking* sampai ke simpul *parent* yang bersesuaian dan kemudian dilakukan kembali DFS.

Jika memang pada suatu permasalahan ada solusinya, solusi secara DFS akan ditemukan pada kedalaman 'd' dari suatu simpul yang aras-1.

V. KESIMPULAN

Sekarang telah diketahui pencarian solusi dengan metode BFS/DFS beserta keuntungan/kerugian masing-masing algoritma. Dengan algoritma DFS dan BFS keduanya menhasilkan solusi yang sama.

Pada BFS, banyak simpul yang dibangkitkan sebelum akhirnya menuju solusi. Sehingga lebih banyak mengkonsumsi memori, Sedangkan pada DFS, simpul yang dibangkitkan lebih sediti dan lebih tidak boros memori. Selain itu juga lebih cepat,

Untuk permasalahan Crossing the Bridge, solusi yang dihasilkan dari DFS mapun BFS adalah solusi yang optimal.

Langkah-langkah untuk menyelesaikan permasalahan Crossing the Bridge adalah:

1. Sebrangkan A dan B.
2. Kembalikan A.
3. Sebrangkan A dan C.
4. Kembalikan A.
5. Sebrangkan D dan E.
6. Kembalikan B
7. Sebrangkan A dan B

Serangkaian aksi tersebut menghabiskan waktu:

1. 3 satuan waktu
2. 1 satuan waktu
3. 6 satuan waktu
4. 1 satuan waktu
5. 12 satuan waktu
6. 3 satuan waktu
7. 3 satuan waktu

Sehingga total waktu yang digunakan untuk menyelesaikan permasalahan ini adalah 3+1+6+1+12+3+3 = 29 satuan waktu. Hal ini sama baik untuk BFS dan untuk DFS.

Pada DFS, jumlah simpul yang dibangkitkan adalah sebanyak 5 buah simpul. Sedangkan pada BFS, ada banyak sekali simpul yang harus dibangkitkan.

Tapi tentunya dengan DFS, waktu eksekusi pencarian solusi menjadi jauh lebih cepat jika dibandingkan dengan BFS.

REFERENCES

[1] Munir, Rinaldi. 2009. Diktat Kuliah IF2211 Strategi Algoritma. Bandung : Teknik Informatika ITB

[2] http://intelligence.worldofcomputing.net/ai-search/breadth-first-search.html#.U3XvPvmSxeQ diakses pada 16 Mei 2014

[3] http://intelligence.worldofcomputing.net/ai-search/depth-first-search.html#.U3X0I_mSxeQ diakses pada 16 Mei 2014

[4] http://www.cs.cornell.edu/courses/CS2112/2012sp/lectures/lec24/lec24-12sp.html diakses pada 16 Mei 2014

[5] www.novelgames.com/en/spgames/bridge/ diakses pada 16 Mei 2014

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, 16 Mei 2014

Rikysamuel
13512089