Perbandingan Efisiensi antara Algoritma BFS dan DFS dalam Permainan *Battleship*

Ahmad Rizdaputra 13513027
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13513027@std.stei.itb.ac.id

***Abstract*—Ada banyak cara untuk memecahkan sebuah masalah maupun sebuah aplikasi. Salah satunya adalah algoritma *breadth first search* dan *depth first search*. Kedua algoritma ini berfokus kepada bentuk pencari dalam bentuk pohon. Algoritma ini dapat membentuk sebuah aplikasi yang biasa kita dengar dengan nama *battleship*. Permainan *battleship* ini cukup popular di kalangan masyarakat saat ini. Permainan ini termasuk ke dalam jenis permainan papan. Adapun beberapa aturan dasar dari permainan ini yaitu menghancurkan semua kapal musuh sebelum kapal kita sendiri semuanya hancur. Permainan ini dapat diselesaikan dengan algoritma BFS maupun DFS dengan cara memilih gerakan pertama secara acak lalu menembak bagian sekitar dari tempat pertama tersebut. Dari dua algoritma ini bisa salah satunya akan lebih efisien dari pada yang lainnya.**

Kata Kunci—*battleship*, BFS, DFS.

I. PENDAHULUAN

Startegi algoritma adalah cabang keilmuan dari jurusan informatika yang mempelajari berbagai macam algoritma pemecahan masalah dan perbandingannya dalam hal-hal tertentu.

Mata kuliah ini dapat diterapkan di seluruh bidang informatika maupun non-informatika seperti persoalan *travelling salesman problem* atau yang biasa kita kenal dengan TSP. Salah satu penerapan yang bisa kita dapat dari mata kuliah ini adalah pembuatan aplikasi *battleship* yang akan dijelaskan pada makalah ini.

Battleship adalah salah satu permainan papan yang butuh strategi dan cara tertentu untuk memenangkan permainan ini. Aturannya adalah Menghancurkan semua kapal musuh sebelum kapal kita dihancurkan olehnya. Permainan ini hanya bisa dimainkan oleh dua orang dan bisa dimainkan sendiri melawan inteligensi buatan yang telah dibuat oleh program.

Inteligensi buatan ini dapat dibuat dengan algoritma BFS dan DFS yang dapat memunculkan hasil yang cenderung optimal, namun akan dicari yang paling optimal untuk dijadikan sebagai inteligensi buatan.

II. *BATTLESHIP*

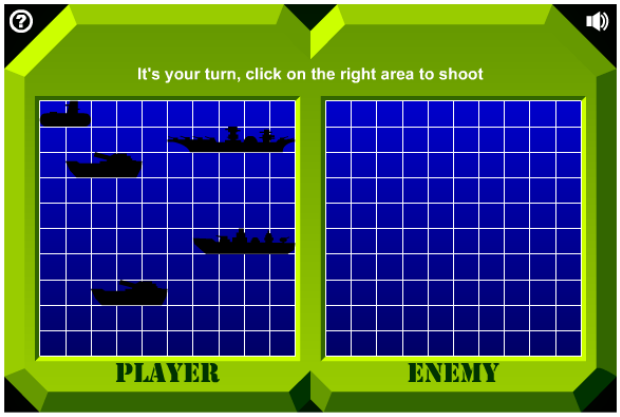
Permainan *battleship* memiliki banyak versi, adapaun beberapa peraturan dasar yang biasanya umum digunakan di banyak versi yaitu:

- Pada saat awal permainan kedua pemain menempatkan kapal-kapal miliknya di tempat yang ia inginkan.
- Pemain tidak dapat memempatkan satu kapal di atas kapal yang lain.
- Saat permainan dimulai pihak satu bertugas untuk menyerang dan yang satunya menunggu giliran.
- Akan dikeluarkan notif yang berbeda pada saat terjadi *hit* atau *miss*.
- Pemain yang berhasil terlebih dahulu menenggelamkan semua kapal musuh maka ialah pemenangnya.

Permainan *battleship* ini juga sudah sangat terkenal sejak dulu sekitar tahun-tahun perang dunia kedua. Adapun beberapa cara bermainnya yaitu:

- Dengan kertas dan digambar sendiri
- Dengan *kit* khusus permainan *battleship.*
- Dengan komputer.
- Bahkan sekarang bisa dimainkan *online* dengan orang lain melalui internet

Berikut illustrasi dari papan permainan *battleship*:



Gambar 2.1 Illustrasi papan permainan *battleship*
Sumber:
http://www.knowledgeadventure.com/games/battleship/
diakses pada tanggal 4-5-2015 pukul 21.55

Pada illustrasi di atas dapat dilihat bahwa danya lima

kapal untuk tiap pemain dan peletakan kapal kita hanya bisa dilihat oleh kita sendiri.

Untuk perilaku dari inteligensi buatan adalah penembakan awal dilakukan secara acak lalu jika ditemukan suatu kapal musuh maka ia akan menembak kotak-kotak yang berada di sekitar tempat ditemukannya kapal tersebut.

III. ALGORITMA BFS

Algoritma *breadth first search* atau yang biasa kita kenal dengan algoritma BFS adalah salah satu jenis algoritma pencarian yang dikategorikan sebagai *uninformed*/*blind search*. Hal ini berarti bahwa BFS adalah algoritma pencarian yang dilakukan tanpa mengetahui solusi yang akan kita temui ada di mana.

Algoritma ini melakukan pencarian secara melebar yang mengunjungi simpul secara preorder yaitu mengunjungi suatu simpul kemudian mengunjungi semua simpul yang bertetangga dengan simpul tersebut terlebih dahulu. Selanjutnya, simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya.

Gambar 3.1 illustrasi urutan kunjungan simpul pada BFS

Adapun cara kerja algoritma BFS sebagai berikut:

Dalam algoritma BFS, simpul anak yang telah dikunjungi disimpan dalam suatu antrian. Antrian ini digunakan untuk mengacu simpul-simpul yan bertetangga dengannya yang akan dikunjungi kemudian sesuai urutan pengantrian.

Untuk memperjelas cara kerja algoritma BFS beserta antrian yang digunakannya, berikut langkah-langkah algoritma BFS:

- Masukkan simpul ujung (akar) ke dalam antrian
- Ambil simpul dari awal antrian, lalu cek apakah simpul merupakan solusi
- Jika simpul merupakan solusi, pencarian selesai dan hasil dikembalikan.
- Jika simpul bukan solusi, masukkan seluruh simpul yang bertetangga dengan simpul tersebut (simpul anak) ke dalam antrian
- Jika antrian kosong dan setiap simpul sudah dicek, pencarian selesai dan mengembalikan hasil solusi tidak ditemukan
- Ulangi pencarian dari langkah kedua

Adapun beberapa property dari algoritma BFS yaitu:

- Algoritma ini bersifat *complete* berarti jawaban pasti ditemukan. Selama nilai jumlah akar dari suatu simpul itu terbatas.
- Jika tidak ada bobot maka algoritma ini optimal, karena dapat dikatakan bahwa langkah = biaya.
- Memilki kompleksitas waktu:
  $1+b+b^2+b^3+\ldots+b^d = O(b^d)$, di mana b adalah jumlah anak dari akar dan d adalah kedalaman pada saat ini.
- Kompleksitas ruang yang cenderung kurang baik yaitu:
  $O(b^d)$, di mana b adalah jumlah anak dari akar dan d adalah kedalaman pada saat ini.

IV. ALGORITMA DFS

Algoritma *depth first search* atau yang biasa kita kenal dengan DFS memiliki karakteristik yang mirip dengan algoritma BFS. Seperti yang dijelaskan di atas algoritma DFS juga termasuk ke kategori *uninformed*/*blind search.*

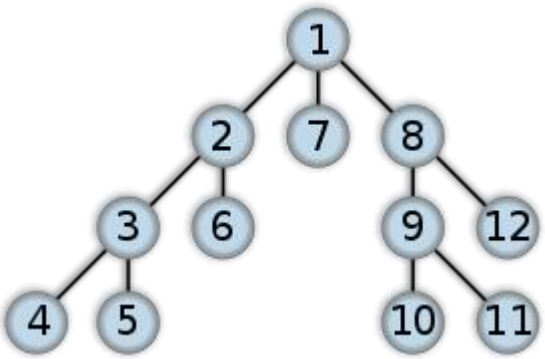
Pencarian dengan algoritma ini dilakukan pada satu node dalam setiap level dari yang paling kiri. Jika pada level yang paling dalam, solusi belum ditemukan, maka pencarian dilanjutkan pada node sebelah kanan. Node yang kiri dapat dihapus dari memori. Jika pada level yang paling dalam tidak ditemukan solusi, maka pencarian dilanjutkan pada level sebelumnya. Demikian seterusnya sampai ditemukan solusi. Jika solusi ditemukan maka tidak diperlukan proses backtracking.

Kelebihan DFS adalah:

- Pemakaian memori hanya sedikit, berbeda jauh dengan BFS yang harus menyimpan semua node yang pernah dibangkitkan.
- Jika solusi yang dicari berada pada level yang dalam dan paling kiri, maka DFS akan menemukannya secara cepat.

Kelemahan DFS adalah:

- Jika pohon yang dibangkitkan mempunyai level yang dalam (tak terhingga), maka tidak ada jaminan untuk menemukan solusi (Tidak *Complete*).
- Jika terdapat lebih dari satu solusi yang sama tetapi berada pada level yang berbeda, maka pada DFS tidak ada jaminan untuk menemukan solusi yang paling baik (Tidak Optimal).



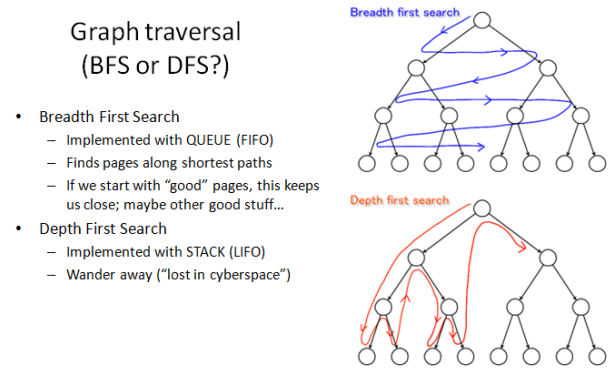
Gambar 3.2 illustrasi urutan kunjungan simpul pada DFS

Sumber: http://en.wikipedia.org/wiki/Depth-first_search

Diakses pada 4-5-2015 pukul 22.40

Adapun beberapa property dari algoritma BFS yaitu:

- Bersifat *complete* jika nilai kedalaman terbatas dan adanya penanganan terhadap jalur yang redundan dan *state* yang berulang
- Tidak optimal karena butuh melakukan perjalanan panjang sampai ke yang paling dalam terlebih dahulu.
- Kompleksitas ruang:
  O(bm), di mana b adalah banyaknya anak dari akar dan m adalah level terdalam yang pernah ditelusuri.
- Kompleksitas waktu yang cenderung kurang baik yaitu:
  $O(b^m)$, di mana b adalah banyaknya anak dari akar dan m adalah level terdalam yang pernah ditelusuri.



Gambar 3.3 Perbandingan alur mencari BFS dan DFS dalam bentuk graf
Sumber: File "Tubes-2-IF2211-Strategi-Algoritma.doc"

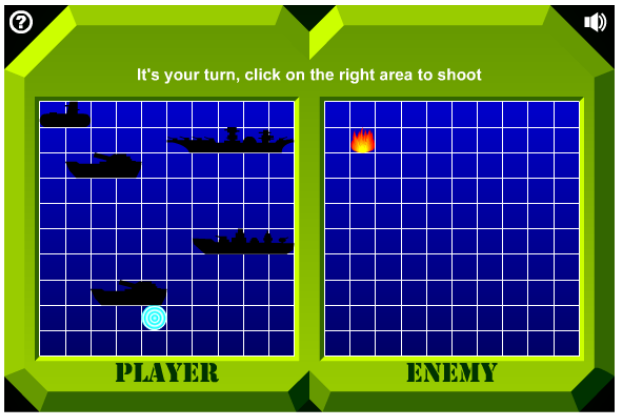
V. ANALISIS

Dalam permainan *battleship*, algoritma DFS dan BFS ini akan dipakai sebagai algoritma pencarian kapal dari sisi musuh dari pemain.

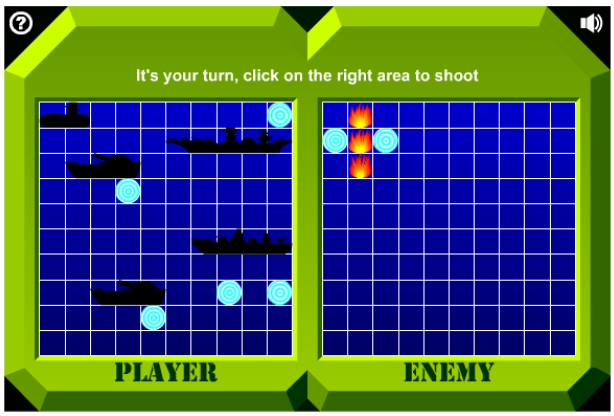
Pertama adalah algoritma BFS. Berikut urutan pemecahan masalah inteligensi buatan ini dengan menggunakan algoritma BFS:

1. Mencari kapal secara acak
2. Jika tidak ditemukan kapal lakukan kembali langkah 1.
3. Jika ditemukan maka pertama serang bagian bawah dari tempat tersebut, lalu kiri atas dan bawah. Hal ini dilakukan karena ini adalh algoritma BFS maka kita akan menelusuri semua bagian yang berada di sekitar tempat ditemukannya kapal terlebih dahulu, lalu baru ke tempat yang lain.
4. Ulangi langkah 3 sampai sekitar tempat ditemukannya kapal musuh sudah tidak ditemukan bagian lagi
5. Ulangi langkah 1 kembali

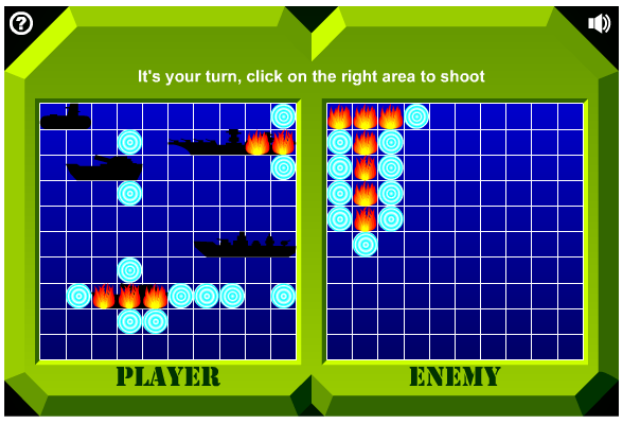
Berikut adalah illustrasi langkah di atas dalam bentuk gambar:



Gambar 5.1 Gerakan pertama dari program
Sumber:
http://www.knowledgeadventure.com/games/battleship/
diakses pada tanggal 4-5-2015 pukul 23.00



Gambar 5.2 Hasil gerakan nomor 3 dari program
Sumber:
http://www.knowledgeadventure.com/games/battleship/
diakses pada tanggal 4-5-2015 pukul 23.00



Gambar 5.3 Hasil gerakan nomor 4 dari program
Sumber:
http://www.knowledgeadventure.com/games/battleship/
diakses pada tanggal 4-5-2015 pukul 23.00

Bisa dilihat bahwa dengan algoritma BFS pencarian cenderung tidak efisien karena jelas bahwa kapal

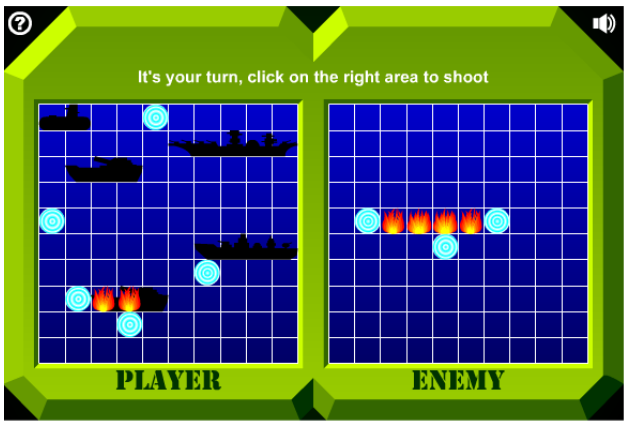
berbentuk lurus maka tidak diperlukan mencari secara horizontal jika secara vertical sudah ditemukan.

Selanjutnya yang kedua adalah dengan algoritma DFS. Berikut penjelasannya dalam bentuk langkah-langkah:

1. Mencari kapal secara acak
2. Jika tidak ditemukan kapal lakukan kembali langkah 1.
3. Jika ditemukan maka serang bagian bawah dari tempat ditemukannya kapal tersebut dahulu.
4. Jika tidak ditemukan maka cari bagian kiri, jika masih tidak ditemukan maka cari bagian atas, jika masih tidak ditemukan maka cai bagian kanan, dan jika tidak ditemukan kembali ke langkah 1.
5. Jika ditemukan pada bagian bawah maka ulangi langkah 3. Jika ditemukan di kiri maka lakukan hal yang sama ke kiri, jika ditemukan di atas maka lakukan hal yang sama ke atas, dan jika ditemukan di kanan lakukan hal yang dama ke kanan.

Berikut illustrasi dalam bentuk gambar:

Gambar 5.4 Gerakan pertama dari program
Sumber:
http://www.knowledgeadventure.com/games/battleship/
diakses pada tanggal 4-5-2015 pukul 23.00



Gambar 5.1 Hasil gerakan 3, 4, dan 5 dari program
Sumber:
http://www.knowledgeadventure.com/games/battleship/
diakses pada tanggal 4-5-2015 pukul 23.00

Dapat dilihat bahwa dengan algoritma DFS pencarian cenderung lebih efisien karena kita dapat mengikuti sampai ke bagian paling dalam terlebih dahulu. Namun ini masih belum optimal, karena memang seharusnya ada yang lebih efisien

Setelah membandingkan kedua hasil dari algoritma BFS dan DFS di atas dapat diketahui bahwa pada kasus permainan *battleship* ini algoritma DFS sedikit lebih efisien disbanding algoritma BFS.

Algortima DFS lebih efisien dikarenakan jika sudah ditemukan arah dari kapal tersebut (horizontal atau vertical) maka ia akan melanjutkan pencarian berdasarkan arah tersebut.

VI. SIMPULAN

Untuk menyelesaikan masalah inteligensi buatan dalam permainan *battleship* ini dapat digunakan algoritma DFS maupun BFS.

Algoritma DFS terbukti sedikit leboh efisien dibandingkan dengan algoritma BFS untuk kasus ini. Sehingga dapat disimpulkan bahwa jika ingin membuat program yang ingin memecahkan maslaah ini maka sebaiknya menggunakan algoritma DFS.

Secara teoritis algoritma BFS lebih efisien namun dikarenakan terbatasnya nilai kedalaman dan jumlah anak untuk setiap akar maka algoritma BFS cenderung boros dan algoritma DFS menjadi lebih efisien.

Akan tetapi algoritma DFS bukanlah yang terbaik untuk aplikasi ini, masih ada lagoritma lain yang lebih efisien seperti algoritma *backtracking* yang dicampur dengan BFS.

VII. SARAN

Di saat kita ingin membuat satu program, kita tidak boleh hanya memikirkan satu tipe algoritma kaarena kita bisaa menggabungkan beberapa algoritma agar llebih efisien dan lebih cepat.

REFERENSI

[1] http://www.knowledgeadventure.com/games/battleship/
diakses pada 4-5-2015 pada pukul 21.55
[2] Slide Kuliah “BFS dan DFS (2015)”.
[3] http://www.wikihow.com/Play-Battleship
diakses pada 4-5-2015 pada pukul 23.00
[4] http://boardgames.about.com/od/battleship/a/Rules-of-Battleship.htm
diakses pada 4-5-2015 pada pukul 23.00
[5] http://www.hasbro.com/common/instruct/battleship.pdf
diakses pada 4-5-2015 pada pukul 23.00
[6] http://www.kirupa.com/developer/actionscript/depth_breadth_search.htm
diakses pada 4-5-2015 pada pukul 23.00

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, 5 Mei 2015

Ahmad Rizdaputra 13513027