Pencarian Pohon Solusi Permainan Alchemy Menggunakan Algoritma BFS dan DFS

Emil Fahmi Yakhya - 13509069
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
emilfahmi@students.itb.ac.id

***Abstract*—*Alchemy* adalah salah satu permainan yang paling populer bagi para pemilik *Device* Mobile dengan Sistem Operasi Android. Permainan ini merupakan permainan dimana pemain harus menggabungkan dua elemen yang sudah ada menjadi elemen baru. Permainan ini adalah permainan yang sederhana dimana pemain tidak membutuhkan keahlian khusus atau ketangkasan dalam memainkannya. Pemain hanya membutuhkan semangat dan kreativitas untuk menemukan elemen-elemen baru.**

Algoritma DFS dan BFS dipakai untuk penelusuran solusi yang akan digunakan saat menemukan dan mencari elemen pada permainan ini. Setiap elemen baru yang ditemukan akan dikumpulkan dalam sebuah *list* kemudian *list* tersebut akan dijadikan *prunning* untuk penelusuran solusi berikutnya.

Index Terms*—Permainan, DFS, BFS, *list.

I. PENDAHULUAN

Alchemy adalah sebuah permainan yang menggunakan prinsip kimia sederhana. Dalam permainan ini pemain akan diberikan empat buah elemen dasar yakni air, angin, api dan tanah untuk digabungkan dan dikembangkan menjadi elemen yang lebih kompleks.

Dalam permainan ini, pemain dapat menggunakan kreativitas untuk menemukan elemen baru. Setiap elemen yang sudah ditemukan oleh pemain juga dapat digabungkan dengan elemen yang sudah ada untuk menjadi elemen yang lebih kompleks lagi.

Walaupun pemain harus mengembangkan permainan dengan kreativitas seluas-luasnya, permainan ini memiliki batasan jumlah elemen maksimal yang ada dan komponen penyusun yang pasti untuk setiap elemen (selain elemen dasar).

Berikut adalah beberapa *screenshot* dari permainan Alchemy yang diambil dari perangkat Android:

Gambar 1: Layar Utama Permainan Alchemy

Gambar 2: *Interface* Penambahan Elemen(*)

Dari *screenshot* tersebut dapat dilihat beberapa elemen yang sudah ditemukan oleh pemain. Elemen tersebut dapat disusun ke dalam layar utama permainan untuk digabungkan dan dieksplorasi lebih jauh oleh pemain.

Pada layar utama permainan ini dapat dilihat jumlah elemen yang sudah ditemukan dan jumlah maksimal elemen yang dapat dieksplorasi. Dan kemudian pada bagian bawah layar ada tiga tombol navigasi utama dimana tombol pertama yang berbentuk "?" berguna untuk mencari informasi dan *track record* dari suatu elemen. Tombol "+" berguna untuk menambahkan elemen yang sudah ada ke dalam layar utama dan tombol yang paling kanan berguna untuk melakukan "undo".

Kemudian, apabila kita menekan tombol "+" pada layar utama, maka yang akan ditampilkan adalah *screenshot* yang ada pada gambar sebelah kanan. Dalam layar ini, pemain dapat menambahkan elemen satu per satu atau pun sekaligus banyak.

Dan yang paling menarik dari permainan ini adalah ketika pemain menemukan suatu elemen dan mengarahkannya ke tombol "?", maka pemain dapat menemukan informasi-informasi mengenai elemen tersebut yang sudah terintegrasi dengan wikipedia sehingga permainan ini tidak hanya memberi kesenangan kepada pemain namun juga bisa memberi banyak pengetahuan baru.

(*): Pada *screenshot* dapat dilihat ada elemen yang diberi tanda "*" yang berarti elemen tersebut adalah

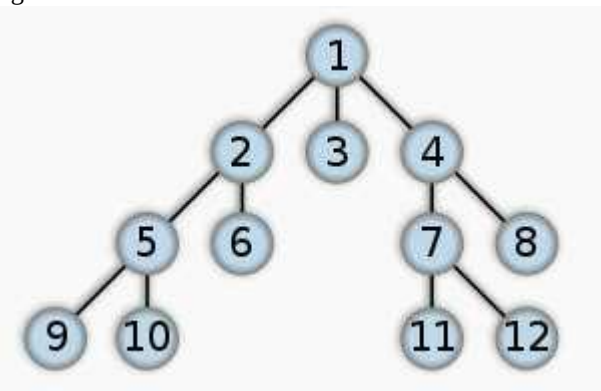
II. Dasar Teori

A. Algoritma BFS (*Breadth First Search)*

Dalam teori graf, BFS adalah algoritma pencarian graf vang melakukan pencarian dari simpul akar. Kemudian penelusuran diteruskan dengan mencari ke semua simpul yang bertetangga.

Untuk setiap simpul-simpul tetangga yang telah dikunjungi, setelahnya akan ditelusuri simpul-simpul lain yang belum dikunjungi yang bertetangga dengan simpul tersebut. Mekanisme tersebut akan terus dilakukan sampai pencarian berakhir.

Berikut adalah contoh persoalan graf yang diselesaikan dengan metode BFS.



Gambar 3: Contoh urutan penelusuran dalam algoritma BFS

Dalam algoritma ini, semua simpul tetangga atau simpul anak yang didapat dengan mengembangkan sebuah simpul akan dimasukkan ke dalam FIFO *queue*. Pada implementasinya, informasi simpul-simpul akan disimpan pada sebuah kontainer yang berupa *queue* atau *linked list*.

Secara umum, algoritma BFS adalah sebagai berikut:

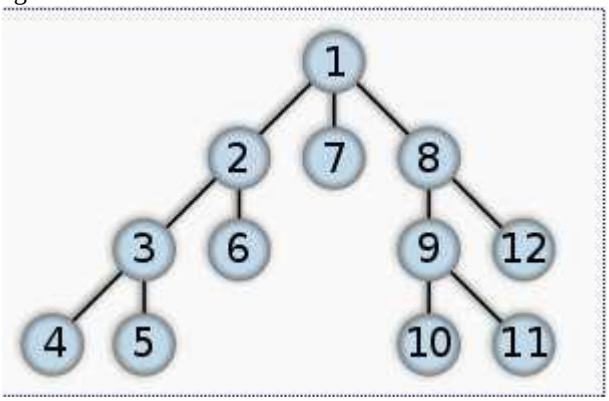
1. Masukkan simpul akar ke dalam antrian.
2. Ambil sebuah simpul pada antrian kemudian diperiksa.
    a. Jika elemen yang dicari ditemukan pada simpul ini, hentikan pencarian.
    b. Jika tidak, masukkan simpul-simpul anak simpul ini ke dalam antrian.
3. Jika antrian kosong, semua simpul graf telah diperiksa dan elemen tidak ditemukan. Hentikan pencarian.
4. Ulangi mulai langkah 2.

B. Algoritma DFS (*Depth First Search)*

DFS adalah algoritma untuk melakukan traversal ataupencarian pada sebuah graf atau pohon. Dimulai dari simpul akar, pencarian akan dilakukan dan mengeksplorasi sejauh mungkin pada tiap cabang sebelum akhirnya melakukan backtracking.

DFS bekerja dengan mengembangkan simpul anakpertama pada pohon kemudian mencari lagi lebih dalam pada anak simpul tersebut sampai simpul yang dicari ditemukan atau sampai tidak ada lagi simpul anak yang bisa dikunjungi. Setelah itu, backtrack dilakukan, kembali ke simpul terakhir yang masih memiliki simpul anak yang belum dikunjungi, dan seterusnya.

Berikut adalah contoh persoalan graf yang diselesaikan dengan metode DFS.



Gambar 4: Contoh urutan penelusuran dalam algoritma DFS

Pada implementasi non-rekursif, simpul-simpul yang baru dikembangkan dimasukkan pada sebuah *stack* atau *linked list*.

Secara umum, algoritma DFS adalah sebagai berikut:

1. Masukkan simpul akar ke dalam stack.

2. Ambil sebuah simpul pada antrian kemudian diperiksa.
    a. Jika elemen yang dicari ditemukan pada simpul ini, hentikan pencarian.
    b. Jika tidak, masukkan simpul-simpul anak simpul ini ke dalam antrian.
5. Jika antrian kosong, semua simpul graf telah diperiksa dan elemen tidak ditemukan. Hentikan pencarian.
6. Ulangi mulai langkah 2.

III. METODE PENERAPAN ALGORITMA

A. Algoritma Bruteforce

Pencarian menggunakan algoritma Bruteforce dilakukan dengan cara melakukan kombinasi secara random. Pencarian ini mengumpulkan semua elemen yang sudah ada pada layar utama dan mengacak seluruh elemen sehingga didapatkan elemen-elemen baru secara acak.

B. Pencarian Heuristik

Pencarian menggunakan prinsip heuristik dilakukan dengan cara mengumpulkan semua kemungkinan elemen yang dapat digabungkan kemudian dicek satu per satu. Setiap elemen yang terlihat sudah tidak mungkin untuk digabungkan dipisahkan dan tidak akan digabungkan sehingga pencarian akan lebih efektif.

C. Algoritma BFS

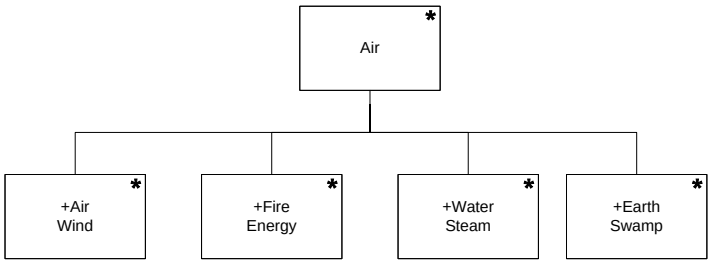
Pencarian menggunakan algoritma BFS memanfaatkan dua *tools.* Yang pertama adalah *list* yang berisi kumpulan-kumpulan elemen yang sudah ditemukan dan yang kedua adalah empat buah pohon BFS yang masing-masing berawal dari elemen dasar (*fire, air, earth, water).*

Mekanisme pencarian dilakukan dengan cara membuat akar pohon kombinasi dari masing-masing elemen dasar tersebut untuk dicari elemen baru yang mungkin tercipta.

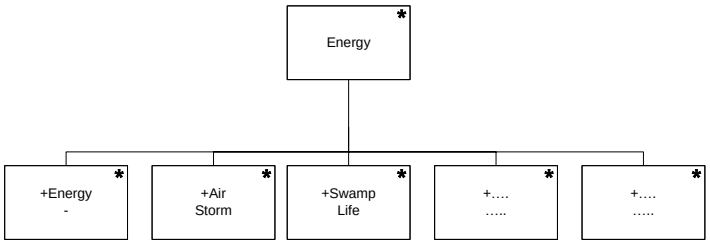
Ketika kemungkinan kombinasi untuk elemen dasar ditemukan seluruhnya (telah dikombinasikan dengan seluruh elemen yang ada pada list), maka pencarian akan dilakukan pada akar baru (pada gambar contoh di bawah ini dilakukan untuk *air & energy).*

Setelah elemen baru ditemukan, maka elemen tersebut akan dimasukkan ke dalam list solusi dan digunakan untuk rekursi pada akar berikutnya.

Pencarian akan dihentikan ketika bertemu dengan *terminal element* atau seluruh elemen yang ada pada jalur *air* telah ditemukan.



Gambar5: Ilustrasi Implementasi Pohon BFS untuk elemen dasar *Air*



Gambar 5: Ilustrasi Implementasi Pohon BFS untuk elemen *Energy*

D. Algoritma DFS

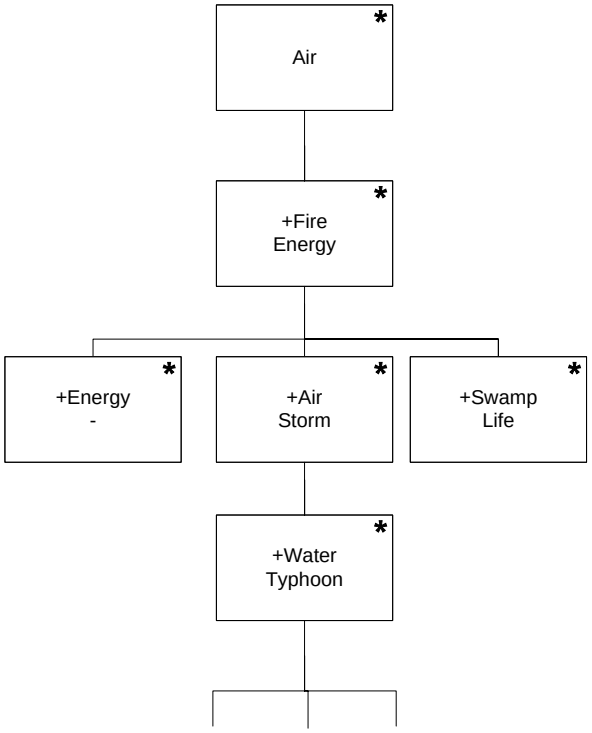
Pencarian menggunakan algoritma DFS juga memanfaatkan dua *tools*. Yang pertama adalah *list* yang berisi kumpulan-kumpulan elemen ayng sudah ditemukan dan yang kedua adalah empat buah pohon DFS yang masin-masing berawal dari elemen dasar (*fire, air, earth, water).*

Mekanisme pencarian dilakukan dengan cara membuat akar pohon kombinasi yang ditelusuri secara mendalam.

Pada gambar di bawah ini, diperlihatkan mekanisme pencarian untuk elemen *air*. Pada pohon ini elemen *air* digabungkan dengan elemen pertama yang berada di *list* solusi sehingga menghasilkan *energy.*

Kemudian setelahnya energy tersebut akan dikombinasikan dengan *energy* yang dimasukkan ke dalam *list* yang ternyata gagal.

Ketika terjadi kegagalan seperti ini, maka pencarian akan kembali ke induknya dan mencoba melakukan dengan kombinasi lain, *air*, yang ternyata menghasilkan *storm.* Ketika kombinasi ini berhasil, maka penelusuran akan langsung mencoba membuat pohon anaknya dan melakukan kombinasi kembali. Pencarian ini akan berhenti ketika menemukan *terminal element* atau seluruh elemen pada jalur *air* telah ditemukan.



Gambar 6: Ilustrasi Implementasi Pohon DFS untuk *Air*

IV. Analisis Penerapan Algoritma

A. Algoritma Bruteforce

Pada pencarian dengan menggunakan algoritma Bruteforce, kemungkinan ditemukan elemen-elemen akan lebih cepat pada awal permainan. Hal ini disebabkan karena elemen-elemen yang digabungkan secara acak adalah elemen dasar yang masih memiliki kemungkinan ditemukannya elemen baru sangat tinggi.

Namun, ketika elemen yang ditemukan telah banyak (>100), maka algoritma ini akan sangat tidak efektif karena kemungkinan ditemukannya elemen baru menjadi semakin kecil.

B. Pencarian Heuristik

Pencarian heuristik ini efektifitasnya tidak bisa ditentukan secara empiris karena kemungkinan ditemukannya elemen baru sangat bergantung kepada kreativitas dan keuletan pemain dalam mencari elemen baru dari elemen-elemen yang sudah ada.

Meskipun begitu, ada satu hal yang dapat dipastikan bahwa metode ini lebih efektif dibanding brute force ketika jumlah elemen yang ditemukan sudah banyak (>100).

C. Algoritma BFS

Pencarian solusi menggunakan algoritma BFS dirasa sangat lambat pada awal pencarian. Hal ini disebabkan algoritma BFS membutuhkan usaha yang lebih pada awal permainan untuk mencoba semua kemungkinan yang ada pada *list* solusi pemain.

Namun, algoritma ini akan efektif ketika jumlah elemen yang ditemukan sudah cukup banyak (>50) karena algoritma ini menjanjikan kepastian akan ditemukannya elemen baru setiap pencarian suatu *node* pada pohon.

Dapat dikatakan algoritma ini adalah algoritma yang paling cepat untuk mengumpulkan elemen sebanyak-banyaknya karena sifat pencariannya yang “melebar”.

D. Algoritma DFS

Pencarian solusi menggunakan algoritma DFS juga dirasakan sangat lmabat pada awal pencarian. Hal ini disebabkan algoritma DFS membutuhkan usaha yang lebih pada awal permainan untuk mencoba semua kemungkinan yang ada pada *list* solusi.

Namun, algoritma ini akan efektif ketika jumlah elemen yang ditemukan sudah cukup banyak (>50) karena algoritma ini juga menjanjikan kepastian akan ditemukannya elemen baru pada setiap tahap pencarian.

Kekurangan algoritma ini dibandingkan dengan algoritma BFS adalah tidak memiliki kecepatan mencari elmeen sebanyak-banyaknya karena pencariannya yang bersifat mendalam sehingga kemungkinan menemukan variasi elemen akan lebih sedikit.

Namun dapat dikatakan bahwa algoritma ini adalah algoritma yang paling cepat untuk menemukan *terminal element* karena sifat pencariannya yang mendalam.

V. Program yang Telah Diimplementasi dengan Algoritma BFS (Tambahan)

Berikut adalah program “Alchemy Helper” yang telah diimplementasikan dengan algoritma BFS.

Gambar 7: Program "Alchemy Helper"

Dalam program ini, terdapat dua buah kolom. Kolom yang pertama adalah kumpulan-kumpulan elemen yang telah ditemukan oleh user. Elemen ini juga disimpan sebagai list solusi.

Kemudian, kolom pada sebelah kanan merupakan solusi-solusi yang mungkin didapatkan oleh pemain yang dicari menggunakan algoritma algoritma BFS berdasarkan elemen-elemen yang berada pada kolom sebelah kiri.

V. KESIMPULAN

1. Algoritma Bruteforce merupakan algoritma yang efektif digunakan pada awal permainan Alchemy, namun buruk untuk digunakan pada tahap selanjutnya.
2. Pencarian Heuristik merupakan pencarian yang paling asyik untuk digunakan karena menggali kreativitas sebesar-besarnya dari pemain.
3. Algoritma BFS merupakan algoritma yang paling cepat untuk mengumpulkan seluruh elemen yang ada permainan Alchemy.
4. Algoritma DFS merupakan algoritma yang paling efektif untuk menemukan *terminal element* yang ada pada permainan Alchemy.

REFERENSI

[1] [1] Munir, Rinaldi, "Diktat Kuliah IF3051 Strategi Algoritma", Program Studi Teknik Informatika, 2009.

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, 8 Desember 2011

Emil Fahmi Yakhya - 13509069