Perbandingan Penyelesaian Permainan Bejeweled dengan Algoritma *Branch and Bound* dan *Greedy*

Rahmad Yesa Surya / 13515088
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jalan Ganesha Nomor 10, Bandung
13515088@std.stei.itb.ac.id

Abstrak— **Makalah ini menunjukkan perbandingan penggunaan algoritma *greedy* dan *branch and bound* dalam menyelesaikan permainan *puzzle* populer bernama Bejeweled. Di dalam makalah ini akan dijelaskan bagaimana algoritma *greedy* dan *branch and bound* mengambil keputusan di setiap langkah. Keputusan yang diambil oleh kedua algoritma tersebut berdampak pada waktu eksekusi sehingga keduanya akan memiliki waktu eksekusi yang berbeda. Setelah dilakukan pengujian terhadap kedua algoritma, disimpulkan bahwa algoritma *greedy* memiliki performansi waktu yang lebih baik dibandingkan *branch and bound* untuk menyelesaikan permainan Bejeweled dalam mode normal.**

Kata kunci—rantai; perhiasan; optimasi; pohon;

I. PENDAHULUAN

Bejeweled merupakan salah satu permainan ber-*genre puzzle* yang populer dimainkan pada sistem operasi Windows. Permainan ini dikembangkan pertama kali oleh PopCap Games pada tahun 2001. Pada awalnya, permainan ini hanya dikembangkan dalam bentuk aplikasi *web* sehingga hanya bisa dimainkan pada *web browser*. Sampai pada saat ini, sudah lebih dari 75 juta aplikasi Bejeweled yang terjual. Hal lain yang perlu diketahui adalah permainan ini telah diunduh lebih dari 150 juta kali sejak ia diciptakan[1].

Dalam perkembangannya, Bejeweled dikembangkan menjadi aplikasi *desktop* dan mulai dijadikan aplikasi bawaan pada sistem operasi Windows. Pada sistem operasi Windows 8 atau yang lebih baru, aplikasi ini tidak lagi dijadikan aplikasi bawaan, namun tersedia dan dapat diunduh pada Windows Store. Dengan alasan popularitasnya yang semakin baik, Bejeweled kemudian dikembangkan supaya berjalan di sistem operasi lain, misalnya Mac OS.

Bejeweled dimainkan dengan cara menukar sebuah perhiasan (sesuai namanya *jewel*) dari sebuah titik (koordinat) dengan perhiasan yang berada di sebelahnya secara vertikal atau horizontal. Perhiasan tersebut hanya dapat tertukar jika membentuk rantai yang panjangnya minimal tiga perhiasan yang sama. Jika syarat tersebut tidak dipenuhi, maka perhiasan tidak dapat ditukar. Pemain akan mendapatkan skor jika berhasil membentuk rantai perhiasan yang sama dengan panjang minimal tiga. Rantai tersebut kemudian akan hilang dan perhiasan di bagian atas rantai tersebut akan bergeser kebawah mengisi ruang yang kosong. Kadangkala, dengan menukar sekali, pemain dapat menjumpai *cascades*, yakni rantai perhiasan yang sama dengan panjang minimal tiga terbentuk sendiri setelah rantai sebelumnya hilang.

Gambar 1.1 Tampilan permainan Bejeweled

Dalam *normal mode*, pemain akan mengumpulkan skor sebanyak-banyaknya hingga tercapai skor target. Pemain yang mampu mencapai skor target memenangkan permainan. Terdapat modus permainan yang lain yakni *time-trial mode* yang mana pemain beradu dengan waktu untuk memperoleh skor tertentu. Pemain yang mampu mencapai skor target dalam waktu yang ditentukan memenangkan pertandingan.

Penulis menyusun makalah ini untuk membandingkan penyelesaian permainan Bejeweled dengan menggunakan dua pendekata algoritma, yakni algoritma *branch and bound* dan algoritma *greedy*. Permainan Bejeweled yang digunakan berapa pada modus normal sehingga parameter terselesaikannya permainan adalah jika pemain berhasil mencapai sebuah skor target.

II. DASAR TEORI

A. Bejeweled

Bejeweled memiliki papan utama yang berbentuk persegi dengan ukuran tertentu. Tingkat kesulitan Bejeweled ditentukan oleh persebaran perhiasan pada papan tersebut. Pada level yang mudah, perhiasan-perhiasan yang sama terletak pada koordinat yang tidak terlalu jauh pada papan, sehingga pemain dapat dengan cepat mencapai skor target. Pada permainan ini, selain pemain diharuskan membentuk rantai perhiasan yang sama dengan panjang minimal tiga untuk mendapatkan poin, pemain dapat memanfaatkan fitur lain seperti power-up dan sebagainya. Fitur -fitur tersebut mempercepat pemain mendapatkan skor tinggi. Namun demikian, penggunaan fitur-fitur seperti power-up tersebut tidak digunakan dalam pembahasan makalah ini.

Gambar 1.2 Ilustrasi penukaran perhiasan untuk membentuk rantai perhiasan yang sama dengan panjang minimal tiga

B. Algoritma *Branch and Bound*

Branch and bound adalah algoritma yang digunakan untuk mengoptimasi sebuah permasalahan. Algoritma ini bekerja dengan meminimumkan atau memaksimumkan fungsi objektif dengan tujuan setiap langkah yang diambil merupakan langkah yang optimum untuk mencapai solusi. *Branch and bound* digambarkan secara lojik pada struktur data pohon yang mana setiap simpul pada pohon tersebut memiliki biaya (*cost*). Biaya ini kemudian menentukan simpul manakah yang harus dituju dan diekspan selanjutnya oleh pemain.

Pada penyelesaian permainan Bejeweled ini, *branch and bound* akan membangun pohon ruang status. Pada setiap kedalaman pohon ruang status, untuk sebuah papan permainan yang berukuran n x n, maka jumlah simpul yang dibangkitkan adalah

$$4 \cdot ((n-2) \cdot (n-2)) + 3 \cdot (4 \cdot (n-2)) + 2 \cdot 4$$

Formula 2.1 Jumlah simpul yang dibangkitkan pada setiap langkah di papan permainan berukuran n x n

Pada papan berukuran n x n, perhiasan yang dapat melakukan gerakan ke empat arah yang berbeda (kanan, kiri, bawah, dan atas) adalah perhiasan yang terletak di tengah atau tidak terletak di pinggiran papan permainan. Kemudian, perhiasan yang terletak di pinggiran papan namun tidak di pojok papan mampu bergerak ke tiga arah yang berbeda. Perhiasan yang tidak terletak pada pojok papan dan terletak di:

- pinggiran kanan dapat bergerak ke arah kiri, bawah dan atas
- pinggiran kiri dapat bergerak ke arah kanan, bawah, dan atas
- pinggiran bawah dapat bergerak ke arah kanan, kiri, dan atas
- pinggiran atas dapat bergerak ke arah kanan, kiri, dan bawah

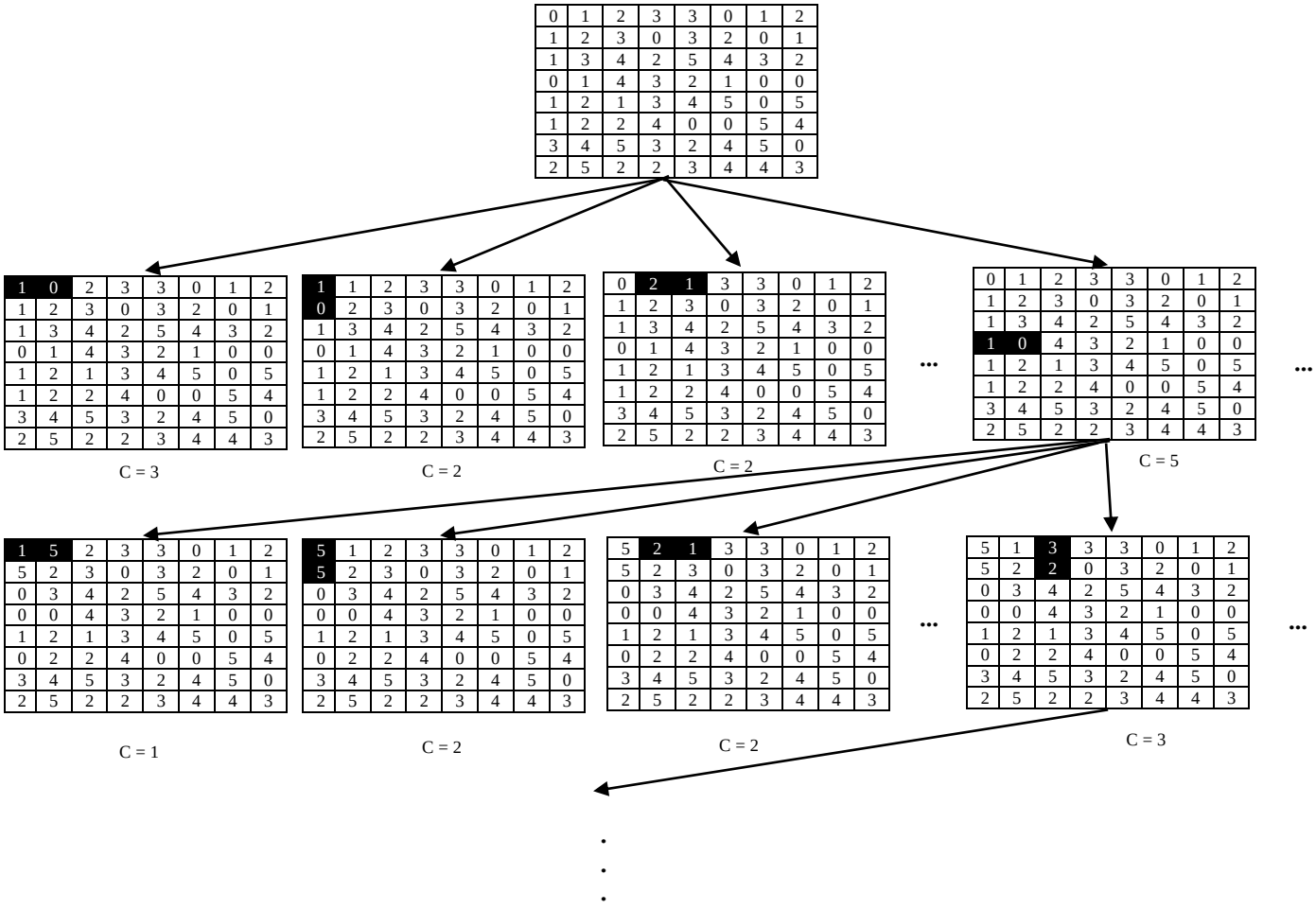
Kemudian, terdapat empat perhiasan yang masing-masing terletak di pojok papan. Keempat perhiasan tersebut hanya mampu bergerak ke dua arah berbeda. Perhiasan yang terletak di:

- pojok kanan atas dapat bergerak ke kiri dan bawah
- pojok kiri atas dapat bergerak ke kanan dan bawah
- pojok kanan bawah dapat bergerak ke kiri dan atas
- pojok kiri bawah dapat bergerak ke kanan dan atas

Pohon ruang status yang dibangkitkan oleh *branch and bound* dimulai pola papan awal permainan sebagai akar pohon. Setiap simpul selanjutnya dibangkitkan dengan cara melakukan iterasi terhadap seluruh perhiasan pada koordinat yang berbeda-beda dan mencoba penukaran atau gerakan ke seluruh arah yang memungkinkan. Bersamaan dengan dibangkitnya simpul tersebut, dihitung pula biayanya menggunakan sebuah fungsi. Fungsi ini bertujuan untuk mencari gerakan sebuah perhiasan yang dapat menghasilkan rantai perhiasan sama yang terpanjang. Dengan demikian, diharapkan setiap langkah yang diambil selalu menghasilkan rantai perhiasan sama terpanjang dan dapat menghasilkan skor lebih banyak. Ilustrasi pembangkitan pohon ruang status dapat dilihat pada Grafik 2.1. Pada grafik ini, setiap perhiasan dilambangkan dengan angka 0 sampai 5.

Pada grafik 2.1 dapat dilihat bahwa pada kedalaman 1, simpul yang dipilih adalah simpul yang memiliki biaya bernilai lima. Simpul ini adalah simpul yang memiliki rantai perhiasan terpanjang sehingga dipilih untuk diekspan selanjutnya. Pada saat simpul ini dipilih, pola papan menjadi berubah. Perubahan ini diatur oleh *game engine* dengan meng-*generate* perhiasan secara acak untuk mengisi ruang kosong karena perhiasan yang termasuk kedalam rantai dengan panjang lima satuan tersebut akan hilang. Pola papan baru inilah yang kemudian dijadikan parameter bagi upapohon selanjutnya untuk membangkitkan simpul-simpul. Seperti yang terlihat pada grafik, terdapat perbedaan pada pola papan di kedalaman 1 dan kedalaman 2.

Pada grafik diatas, tidak seluruh simpul dapat digambarkan mempertimbangkan jumlah ruang yang terbatas. Maka dari itu penulis mencantumkan … sebagai tanda bahwa masih terdapat simpul-simpul lain yang tidak digambarkan. Pohon ruang status tersebut akan terus berkembang semakin kebawah sampai skor target dicapai atau tidak ada lagi solusi yang ditemukan (tidak ada lagi rantai perhiasan dengan panjang minimal 3). Pada penyelesaian permainan ini, skor diperoleh setelah sebuah langkah berhasil dilakukan, yang kemudian akan menghasilkan pola papan baru. Perhitungan skor pada penyelesaian ini berdasarkan pada panjang rantai perhiasan yang hilang dari papan. Misalkan suatu langkah diambil dengan panjang rantai perhiasan adalah tiga, maka pemain akan mendapatkan skor tiga



Grafik 2.1 Ilustrasi pohon ruang status pada permainan Bejeweled

C. Algoritma *Greedy*

Algoritma *greedy* adalah algoritma yang juga digunakan untuk mengoptimasi permasalahan. Algoritma *greedy* memiliki beberapa komponen yang digunakan untuk menyelesaikan permasalahan, yakni himpunan kandidat, himpunan solusi, fungsi seleksi, fungsi kelayakan, dan fungsi objektif.

Pada penyelesaian permainan Bejeweled ini, algortima *greedy* memiliki komponen-komponen sebagai berikut:

1) Himpunan kandidat adalah himpunan seluruh kemungkinan gerakan dari seluruh perhiasan yang terletak pada koordinat yang berbeda-beda.
2) Himpunan solusi adalah himpunan gerakan yang menghasilkan rantai perhiasan dengan panjang minimal tiga dari setiap perhiasan yang terletak pada koordinat yang berbeda-beda
3) Fungsi seleksi adalah pemilihan gerakan yang menghasilkan rantai perhiasan dengan panjang minimal tiga
4) Fungsi kelayakan adalah pemilihan gerakan yang menghasilkan rantai perhiasan dengan panjang minimal tig
5) Fungsi objektif adalah pemilihan gerakan yang menghasilkan rantai perhiasan dengan panjang minimal tiga dan paling awal ditemukan

Sesuai dengan fungsi objektif, algoritma *greedy* ini akan memilih simpul paling awal yang dibangkitkan yang memiliki biaya minimal tiga. Artinya, algoritma *greedy* bertujuan untuk mengoptimasi waktu yang diperlukan untuk mengambil gerakan pada setiap langkah.

III. IMPLEMENTASI DAN PENGUJIAN

A. IMPLEMENTASI

Berikut adalah *pseudo-code* penyelesain masalah dengan algoritma *branch and bound*:

```
while (belum mencapai skor target)
    - bangkitkan akar pohon yang merupakan
      representasi pola papan yang
      sekarang.
    - for (setiap perhiasan pada papan
      permainan)
        - bangkitkan simpul yang merupakan
          pola papan setelah perhiasan
          ditukar ke arah kiri,kanan,bawah
          dan atas. Ambil arah yang
          mungkin saja, misalkan rantai di
          koordinat (0,0) hanya bisa
          ditukar ke kanan dan bawah.
        - hitung biaya dari setiap simpul
          yang dibangkitkan
    - dari seluruh simpul yang telah
      dibangkitkan, pilih simpul yang
      memiliki biaya maksimal untuk
      diekspan dan menjadi akar upapohon
      selanjutnya
```

Gambar 4.1 *Pseudo-code* algoritma *branch and bound*

Berikut ini adalah *pseudo-code* penyelesaian masalah dengan algoritma *greedy:*

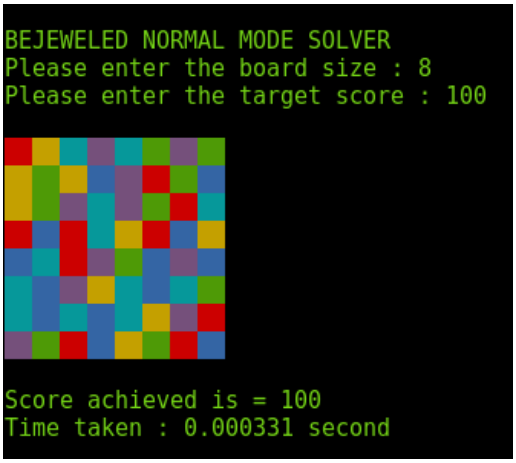
```
while (belum mencapai skor target)
    - for (setiap perhiasan pada papan
      permainan)
        - periksa apakah penukaran dengan
          prioritas kanan,kiri,bawah, dan
          atas menghasilkan sebuah rantai
          dengan panjang minimal tiga
          perhiasan yang sama.
        - jika ya, maka pilih langkah
          tersebut dan keluar dari loop for
```

Gambar 4.2 *Pseudo-code* algoritma *greedy*

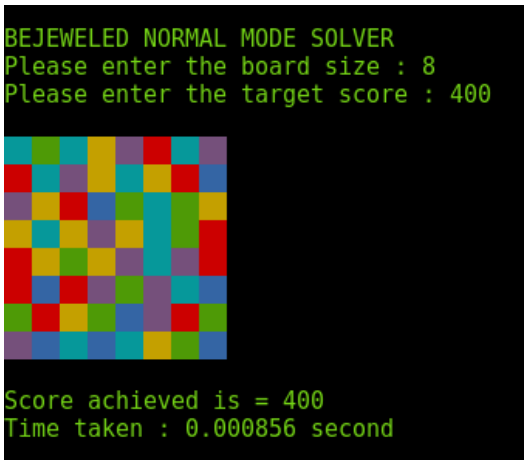
Kedua algoritma diimplementasi dengan menggunakan bahasa C++.

B. PENGUJIAN

Program hasil implementasi diuji pada PC ASUS dengan spesifikasi prosesor Intel Core i3 dan RAM 6GB. Papan Bejeweled yang digunakan berukuran 8 x 8, sesuai dengan papan Bejeweled pada umumnya. Untuk keperluan variansi pengukuran, digunakan pula papan Bejeweled berukuran 16 x 16. Skor target yang harus dicapai untuk menyelesaikan permainan adalah 100 dan 400. *Puzzle* yang ditampilkan adalah pola *puzzle* terakhir yang dicapai. Berikut ditampilkan hasil-hasil eksekusi kedua algoritma untuk masukan-masukan yang berbeda.

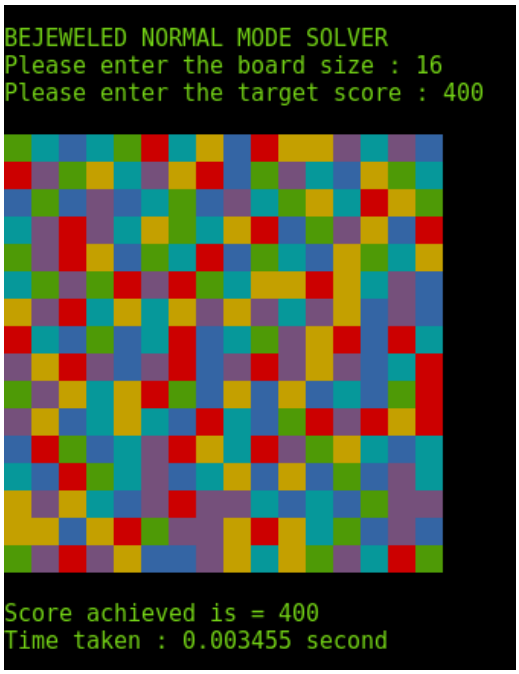


Gambar 4.3 *Branch and bound* – 8 x 8 dengan target skor 100

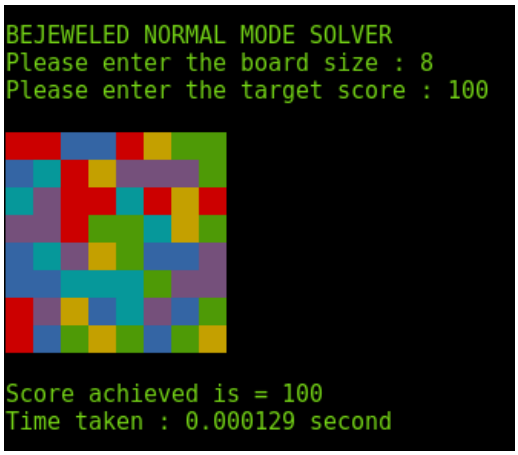


Gambar 4.4 *Branch and bound* – 8 x 8 dengan target skor 400

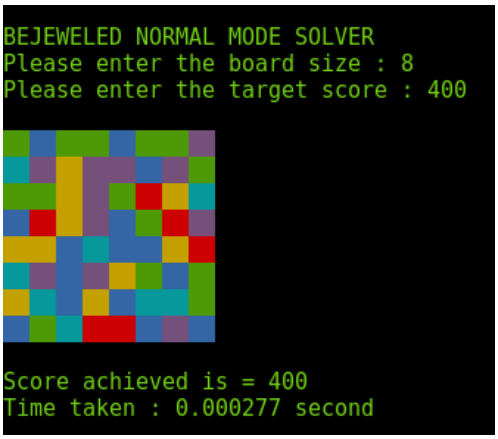
Gambar 4.5 *Branch and bound* – 16 x 16 dengan target skor 100



Gambar 4.6 *Branch and bound* – 16 x 16 dengan target skor 400



Gambar 4.7 *Greedy* – 8 x 8 dengan target skor 100



Gambar 4.8 *Greedy* – 8 x 8 dengan target skor 400

Gambar 4.9 *Greedy* – 16 x 16 dengan target skor 100

Gambar 4.3 *Greedy* – 16 x 16 dengan target skor 400

Berikut adalah ringkasan hasil eksekusi program diatas.

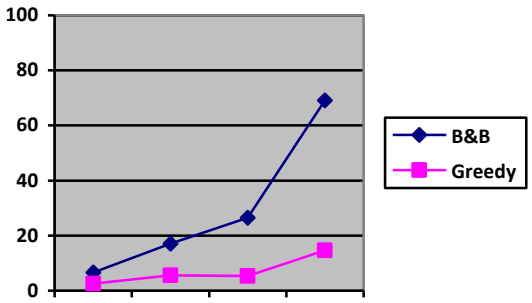
Ukuran Papan	Skor Target	Waktu Eksekusi (s)	
		Branch and bound	Greedy
8 x 8	100	0,000331	0,000129
8 x 8	400	0,000856	0,000277
16 x 16	100	0,001324	0,000265
16 x 16	400	0,003455	0,000729

Tabel 4.1 Ringkasan hasil eksekusi program

Berikut adalah grafik yang menunjukkan perbandingan waktu eksekusi antara kedua algoritma. Sumbu Y menunjukkan skala persentase yang dihitung dengan rumus

$$\frac{waktu}{0,005} \cdot 100$$

Sumbu X menunjukkan jenis-jenis pengujian sesuai dengan urutan pada Tabel 4.1.



Grafik 3.1 Perbandingan *Branch and Bound* dengan *Greedy*

IV. ANALISIS HASIL UJI

Setelah dilakukan pengujian penyelesaian permainan dengan pendekatan *greedy* dan *branch and bound*, ditemukan kesimpulan bahwa pendekatan *greedy* dapat menyelesaikan permainan dengan waktu yang lebih cepat.

Pendekatan *branch and bound* membangkitkan seluruh simpul yang mungkin dibangkitkan. Simpul-simpul ini merupakan pola papan yang dihasilkan dari setiap gerakan perhiasan yang mungkin. Jika papan permainan berukuran 8 x 8, jumlah simpul yang harus dibangkitkan pada setiap langkah adalah

$$4 \cdot (6 \cdot 6) + 3 \cdot (4 \cdot 6) + 2 \cdot (1 \cdot 4) = 224$$

Pembangkitan simpul ini dilakukan dengan mengevaluasi seluruh kemungkinan gerak dari perhiasan mulai dari yang berada pada koordinat (0,0) hingga koordinat (n-1, n-1). Walaupun tujuan dari pembangkitan seluruh simpul ini adalah untuk mengetahui langkah yang memberikan rantai perhiasan terpanjang, namun cara ini memerlukan pemrosesan yang lebih banyak sehingga berakibat pada banyak waktu yang dibutuhkan.

Di sisi lain, algoritma *greedy* mengambil keputusan pada setiap langkah dengan lebih cepat dibanding algoritma *branch and bound*. Algoritma *greedy* tidak perlu mengevaluasi seluruh perhiasan yang ada beserta seluruh kemungkinan geraknya. Alih-alih melakukan hal tersebut, algoritma *greedy* akan memilih langsung langkah pertama yang mungkin dilakukan. Walaupun tidak akan menghasilkan rantai perhiasan terpanjang, namun cara ini berhasil menentukan langkah-langkah selanjutnya dengan cepat.

Pada dasarnya, dalam permainan Bejeweled ini, upaya untuk menemukan rantai perhiasan terpanjang tidaklah terlalu membantu karena selisih antara panjang rantai perhiasan minimal (tiga perhiasan yang sama) dengan rantai perhiasan terpanjang tidak terlalu jauh.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Melalui pengujian yang dilakukan terhadapa algoritma *branch and bound* dan *greedy* untuk menyelesaikan permainan Bejeweled, ditemukan kesimpulan bahwa algoritma *greedy* dapat menyelesaikan permainan dengan lebih cepat, yakni dengan mendapatkan skor target terlebih dahulu.

Algoritma *branch and bound*, meskipun memang mengedepankan optimasi pada setiap langkah yang diambil, namun dalam permainan ini tidak terlalu berdampak karena alih-alih mempercepat untuk mendapatkan skor, pembangkitan pohon ruang status memakan banyak waktu. Di

sisi lain, algoritma *greedy* yang juga mengedepankan prinsip optimasi berhasil menyelesaikan permainan lebih cepat karena setiap langkah yang ditemukan pertama kali langsung dipilih.

Jika kita perhatikan lebih lanjut, terdapat perbedaan pada optimasi antara *branch and bound* dan *greedy* pada permainan ini. Optimasi yang dilakukan *branch and bound* adalah optimasi untuk mendapatkan rantai perhiasan terpanjang dengan harapan dapat mendapatkan skor lebih cepat. Pada algoritma *greedy*, optimasi yang dilakukan adalah optimasi waktu. Setiap langkah yang mungkin (yang setidaknya menghasilkan rantai perhiasan dengan panjang minimal) akan langsung dipilih.

B. Saran

Penelitian lebih lanjut diharapkan dapat mengembangkan penyelesaian permainan Bejeweled yang lebih kompleks. Permainan lebih kompleks yang dimaksud adalah yang melibatkan fitur-fitur lain seperti keberadaan power-up dan sebagainya yang tentunya akan mempengaruhi kecepatan penyelesaian dan jumlah skor yang diperoleh pada setiap langkahnya.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Allah SWT karena telah memberikan rahmat dan berkat sehingga penulis dapat menyelesaikan makalah ini. Penulis juga mengucapkan terima kasih kepada dosen pengajar Mata Kuliah IF2211 Strategi Algoritma yaitu Dr. Masayu Leylia Khodra ST. MT., Dr. Ir. Rinaldi Munir, M.T. dan Dr. Nur Ulfa Maulidevi, S.T., M.Sc atas segala bimbingan selama perkuliahan di semester IV ini.

REFERENSI

[1] https://github.com/SuperPenguin/Bejeweled-AI-Project , diakses pada 18 Mei 2017

[2] https://www.gamedev.net/topic/575282-solving-bejeweled-type-game/ , diakses pada 18 Mei 2017

[3] https://github.com/akleemans/bejeweled-bot , diakses pada 18 Mei 2017

[4] https://www.youtube.com/watch?v=jUvYuqbRO-I , diakses pada 18 Mei 2017

[5] http://www.aslag.net/bejeweled.html , diakses pada 18 Mei 2017

[6] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2016-2017/Algoritma-Greedy-(2017).ppt , diakses pada 18 Mei 2017

[7] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2016-2017/Algoritma-Greedy-(2017).ppt , diakses pada 18 Mei 2017

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, 19 Mei 2017

Rahmad Yesa Surya
13515088