Solusi Terbaik Permainan *Rocket Mania Deluxe* dengan Pendekatan Algoritma *BFS* dan Algoritma *Greedy*

Putri Amanda Bahraini

Laboratorium Ilmu Rekayasa dan Komputasi
Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Kampus ITB Jl. Ganesha no.10 Bandung
if14041@students.if.itb.ac.id

ABSTRAK

Dalam kehidupan kita sekarang ini, game menjadi suatu bagian penting bagi kita. Kita memerlukan game, bukan hanya sekedar sebagai pengisi waktu senggang di sela-sela kesibukan, tapi juga sebagai model dari permasalahan kehidupan kita sehari-hari. Salah satu contoh permainan yang selain dapat menjadi penghilang stress juga sekaligus dapat melatih kreatifitas pemain adalah game *Rocket Mania Deluxe*. Permainan ini dikembangkan oleh perusahaan *Nuclide Games* dan dipopulerkan oleh *PopCap Games* pada tahun 2003. Selama ini, sebagian besar pemain memainkan game ini dengan mengandalkan intuisi dan perasaan saja. Dalam makalah ini, saya ingin menjabarkan sebuah algoritma baru, yang merupakan hasil pendekatann 2 algoritma dasar, *Greedy* dan BFS, guna menemukan solusi paling optimum dalam permainan *Rocket Mania Deluxe.*

Kata kunci: Rocket Mania Deluxe, Greedy, BFS

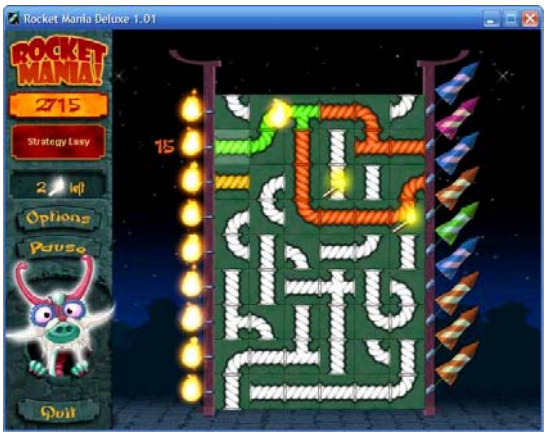
1. PENDAHULUAN

Permainan Rocket Mania Deluxe merupakan permainan yang mengasah kreatifitas pemainnya. Permainan ini mengharuskan pemainnya untuk menggunakan daya pikir dan daya imajinasi untuk menciptakan rangkaian solusi untuk mendapatkan score paling tinggi. Sampai saat ini, permainan ini masih dimainkan dengan mengandalkan intuisi dan perasaan saja. Dalam makalah saya kali ini, saya mencoba untuk menemukan algoritma guna mendapatkan solusi paling optimal dalam permainan ini. Algoritma yang akan saya kemukakan, menggunakan pendekatan algoritma *BFS* (Breadth First Search) sebagai algoritma untuk menemukan alternatif solusi dan algoritma *Greedy* untuk memilih solusi paling optimal.

2. ROCKET MANIA DELUXE

Sampai saat ini, *Rocket Mania Deluxe* merupakan sebuah game yang cukup digemari. *Rocket Mania Deluxe* merupakan permainan yang hampir mirip dengan permainan *Plumber Pipes*, permainan menyusun dan menyambung pipa air guna menyalurkan air antara sumber air dan keran air. Dalam permainan *Plumber Pipes* ini, semakin panjang pipa yang disusun, semakin tinggi nilai score yang bisa diraih. Permainan *Plumber Pipes* ini sampai saat inipun masih populer dimainkan.

Sama halnya dengan game *Plumber Pipes*, *Rocket Mania Deluxe*-pun mengharuskan pemainnya untuk menyusun dan menyambung benda tertentu, hanya saja bedanya, bukan menyambungkan rangkaian pipa antara sumber air dan keran air, tetapi menyambungkan rangkaian sumbu antara sumber api dan kembang api. Dan satu lagi perbedaannya, bukan panjangnya pipa yang dinilai dalam permainan ini, tapi berapa banyak kembang api yang bisa anda luncurkan dengan susunan pipa yang anda ciptakan.



Gambar 1. Tampilan game *Rocket Mania Deluxe*

Untuk lebih jelasnya, bisa dilihat pada Gambar 1. Permainan ini menyediakan 9 korek api dan 9 kembang api. Semakin banyak kembang api yang bisa anda luncurkan, maka akan semakin besar *score* yang anda dapatkan.

Pemain diharuskan menyusun rangkaian sumbu dengan menggunakan bentuk-bentuk sumbu yang disediakan. Pemain dapat memutar sumbu-sumbu tersebut sebanyak

90, 180 atau 360 derajat untuk menciptakan rangkaian sumbu yang terbaik.

Sebenarnya, game ini memiliki 3 mode permainan, ke-3 mode ini memiliki aturan permainan yang sama seperti saya deskripsikan sebelumnya. Hanya saja perbedaanya adalah, pada mode *Classic*, pemain diberi batas waktu, pada *Arcade*, pemain diharuskan menyusun sumbu dengan persediaan sumbu yang terbatas dan pada mode *Strategy*, pemain diharuskan menyusun rangkaian sumbu terbaik dengan persediaan korek api yang terbatas dan diharuskan untuk mengambil bonus korek api dalam salah satu bagian rangkaian sumbu tersebut. Untuk lebih jelasnya, dapat dilihat pada Gambar 2.

Gambar 2. Tampilan game *Rocket Mania Deluxe* dengan menggunakan mode *Strategy*

Pada makalah kali ini, saya hanya akan membahas permainan *Rocket Mania Deluxe* dengan mode *Strategy*, dengan asumsi, tidak ada pembatasan pada persediaan korek api. Algoritma yang akan dibahas kali ini, murni hanya bertujuan untuk menemukan solusi paling optimal, dalam artian mencari solusi rangkaian sumbu yang dapat meluncurkan kembang api paling banyak dalam permainan Rocket Mania Deluxe.

3. ALGORITMA

Makalah ini melakukan pendekatan terhadap 2 jenis algoritma, yaitu algoritma *BFS* (Breadth First Search) dan algoritma *Greedy*.

3.1 Pengertian Algoritma Greedy

Algoritma Greedy merupakan algoritma yang banyak digunakan untuk persoalan optimasi, yaitu persoalan yang terdiri dari 3 elemen, yaitu :

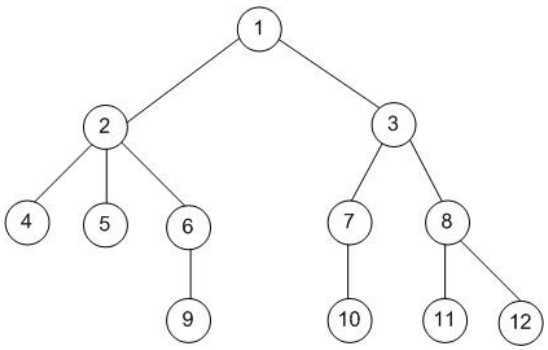
1. Himpunan Kandidat, C. himpunan yang berisi elemen-elemen pembentuk solusi.
2. Himpunan Solusi, S. Berisi elemen dari himpunan kandidat yang terpilih sebagai solusi persoalan.
3. Fungsi Seleksi, fungsi yang pada setiap langkah memilih kandidat yang paling memungkinkan mencapai solusi optimal.
4. Fungsi Kelayakan, yang memeriksa apakah kandidat yang dipilih dapat memberikan himpunan solusi yang tidak melanggar *constraint.*
5. Fungsi obyektif, yaitu fungsi yang memaksimalkan atau meminimumkan nilai solusi.

Algoritma *Greedy* adalah algoritma yang memecahkan masalah langkah per langkah dengan mengambil pilihan terbaik yang dapat diperoleh saat itu tanpa memperhatikan solusi ke depannya dan berharap bahwa dengan memilih optimum local pada setiap langkah akan berakhir dengan optimum global.

3.2 Pengertian Algoritma BFS (Breadth First Search)

Algoritma *BFS* (Breadth First Search) atau yang biasa disebut Algoritma Pencarian Melebar adalah algoritma yang melakukan pencarian solusi dengan memaparkan terlebih dahulu semua pilihan solusi yang ada pada tiap tahap pencarian.

Bayangkanlah sebuah struktur data Pohon dengan simpul yang bertindak sebagai bapak dan simpul yang bertindak sebagai anak. Pada pencarian solusi menggunakan BFS, solusi dicari dengan menghidupkan tiap anak terlebih dahulu pada tiap tahapan, baru setelah itu, jika ada, menghidupkan anak dari simpul anak sebelumnya yang sekarang bertindak sebagai parent.



Gambar 3. Pohon pencarian solusi dengan menggunakan algoritma BFS

Gambar 4 merupakan gambaran urutan penghidupan anak sebagai solusi. Urutan penghidupannya adalah {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

3.2 Algoritma Rocket Mania Deluxe

Algoritma yang digunakan dalam pencarian solusi *Rocket Mania* Deluxe ini, bisa dibilang merupakan algoritma gabungan antara algoritma *BFS* dan algoritma *Greedy*. Algoritma BFS digunakan untuk memberikan alternatif pilihan solusi dari tiap tahapan, dalam permainan Rocket Mania deluxe ini berarti pilihan jenis sumbu yang akan digunakan, beserta pilihan rotasinya.

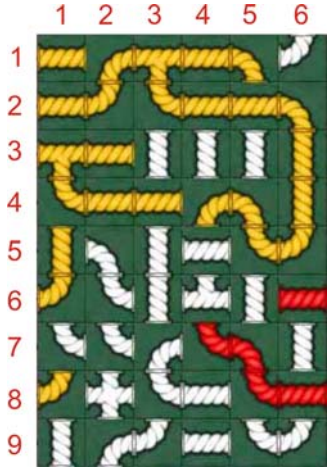
Untuk lebih jelasnya, permainan Rocket Mania Deluxe memiliki 4 jenis sumbu :

Gambar 4. Jenis Sumbu dalam Rocket Mania Deluxe

Dan tiap jenis sumbu tersebut, dapat dirotasi sebanyak 4 arah, contohnya pada sumbu dengan 3 cabang :

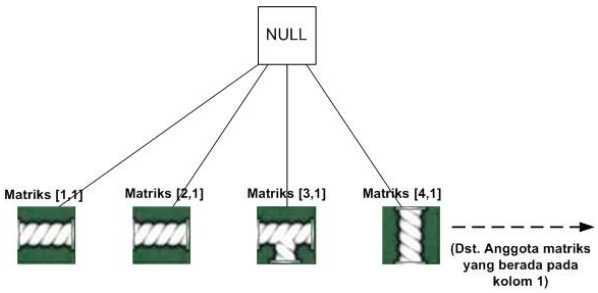
Gambar 5. Sumbu 3 cabang yang dirotasi sebanyak 4 arah

Kita misalkan gambar-gambar sumbu pada persoalan rangkaian yang diberikan pada permainan *Rocket Mania Deluxe* tersebut dipetakan ke dalam sebuah matriks menjadi seperti ini :



Gambar 6. Rangkaian sumbu yang diberi indeks matriks 9x6

Rangkaian sumbu telah dipetakan ke dalam matriks yang berukuran 9 x 6 untuk membantu pencarian solusi, maka dengan ini kita siap menghidupkan elemen pertama sebagai akar untuk pohon solusi BFS kita. Pada algoritma kita ini, kita terpaksa menjadikan akar sebagai parent dengan value NULL, dikarenakan 'akar' itu sendiri berupa sumbu yang memiliki lebih dari 1 pilihan. Berbeda dengan permainan Sudoku yang semua kotak di dalam permainan itu harus diisi dengan value tertentu dan pencarian solusi bisa dimulai dari matriks [1,1]. Pada *Rocket Mania Deluxe*, akar pencarian solusi bisa dimulai dari mana saja selama masih berada dalam kolom 1, jadi ada 9 alternatif pilihan solusi untuk tahap ini.



Gambar 7. Pohon Solusi tahap 1

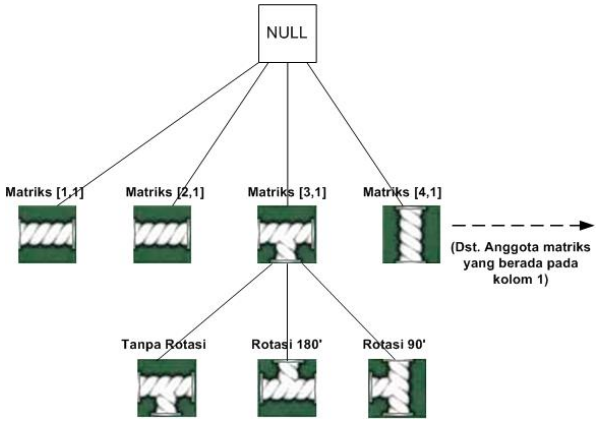
Dalam proses penentuan sumbu sebagai anggota solusi inilah kita menggunakan algoritma *Greedy*, dikarenakan syarat optimasi algoritma kita kali ini adalah menciptakan rangkaian solusi yang dapat meluncurkan roket paling banyak, diasumsikan kita akan mencoba sumbu dengan cabang paling banyak terlebih dahulu. Bisa dilihat pada gambar 7, alternatif akar hanya dapat ditemukan pada kolom ke-1, ada 9 alternatif akar disana, dan pilihan sumbu dengan cabang paling banyak terdapat pada matriks [3,1] yaitu sumbu dengan 3 cabang. Jika sekiranya pencarian dengan sumbu 3 cabang ini sudah selesai, maka pencarian akan dilanjutkan dengan menggunakan sumbu 2 cabang berikutnya, yaitu dimulai dari matriks [1,1] dan seterusnya.

Matriks [3,1] adalah sumbu dengan 3 cabang, dan sebenarnya memiliki 4 alternatif rotasi, namun dalam pencarian solusi kita ini, algoritma *Greedy* mengambil peran dan langsung membuang kemungkinan rotasi sumbu tersebut yang tidak layak dalam solusi.

Gambar 8. Gambaran Alternatif Solusi dengan 4 jenis rotasi yang berbeda

Jelas terlihat pada gambar 8, alternatif ke-4 merupakan sumbu dengan pilihan rotasi yang tidak layak untuk dimasukkan ke dalam himpunan solusi, dikarenakan alternative sumbu tersebut tidak menghubungkan sumbu dengan korek api.

Oleh karena itu, pilihan solusi-pun hanya tersisa 3 pilihan, yang jika digambarkan kedalam pohon solusi :

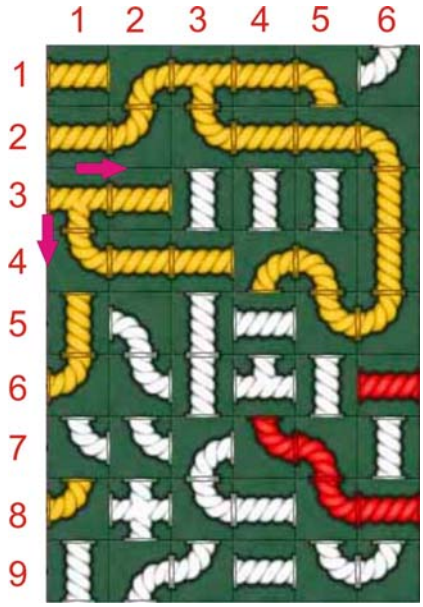


Gambar 9. Pohon Solusi tahap 2

Penentuan sumbu berdasar rotasi ini juga dilakukan dengan menggunakan algoritma *Greedy*, berdasarkan pada berapa banyak cabang yang bisa diciptakan. Memang, sudah jelas dengan posisi matriks [3,1] ini, ke-3 pilihan rotasi di atas akan menciptakan masing – masing sebanyak 2 cabang, namun dalam kasus tertentu, semisal dalam posisi matriks yang berada pada baris 1 dan 9, kemungkinan jumlah cabang akan berkurang, dikarenakan tidak bisa disambungkan ke sumbu manapun, untuk lebih jelasnya :

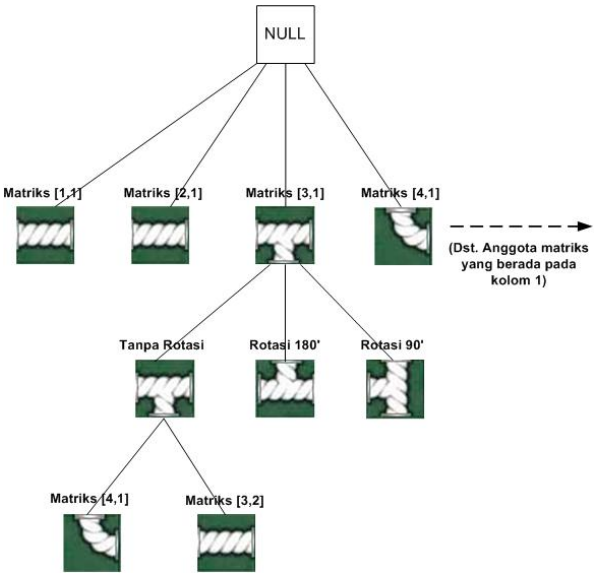
Gambar 10. Contoh posisi yang menyebabkan pengurangan kemungkinan cabang

Jika semua syarat layak sudah dipenuhi, maka pencarian akan berlanjut ke tahap selanjutnya, yaitu menentukan sumbu kedua. Penentuan sumbu kedua ini, dilakukan dengan mengikuti arah cabang dari sumbu pertama. Misalkan yang menjadi sumbu pertama dalam pencarian kali ini adalah Matriks [3,1] Tanpa Rotasi, maka kita akan mendapat 2 kemungkinan indeks matriks solusi, yaitu matriks di kanan dan di bawah Matriks [3,1] : Matriks [3,2] dan Matriks [4,1], seperti yang diperlihatkan pada Gambar 11.



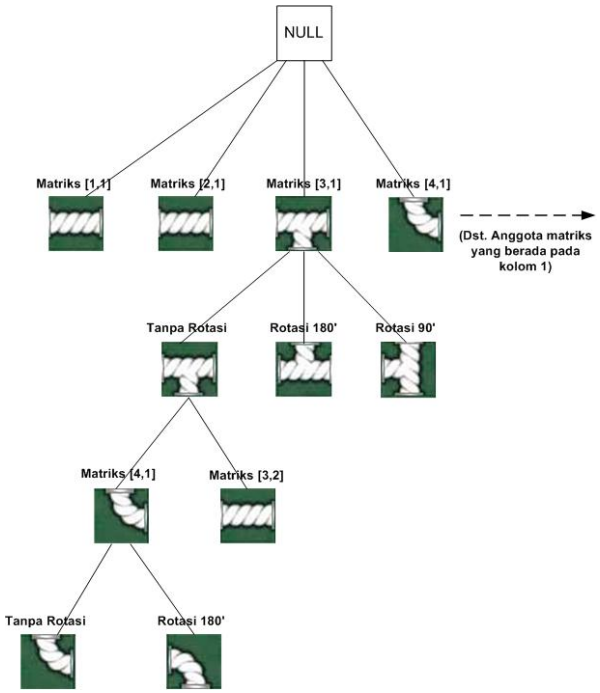
Gambar 11. Kemungkinan Solusi berdasar arah cabang

Yang mana, jika digambarkan ke dalam pohon solusi, akan menjadi seperti ini :



Gambar 12. Pohon Solusi tahap 3

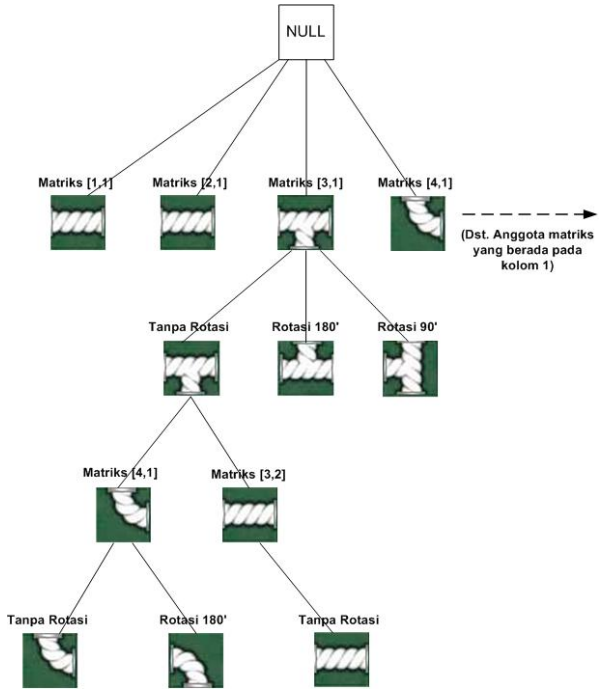
Lalu, setelah melewati proses Algoritma *Greedy* di tahap 3, yaitu memilih cabang terbanyak, yang mana kedua pilihan yang ada sama-sama memilik 1 cabang, maka, proses berlanjut ke tahap selanjutnya yaitu tahap 4. Semisal, Matriks [4,1] terpilih sebagai solusi tahap ke-3, maka pilihan rotasinya :



Gambar 13. Pohon Solusi tahap 4

Dan begitu seterusnya, sampai ditemukan solusi 1, dengan jumlah kembang api yang dapat diluncurkan tercatat dengan baik, dan kesemua data tersebut di masukkan ke dalam array solusi. Setelah solusi satu ditemukan, maka proses akan berlanjut ke solusi 2, solusi 3, dan seterusnya.

Dalam pencarian, mungkin saja solusi mengambil lebih dari satu pilihan, sebagai contoh, solusi tahap 4 bisa saja mengambil solusi sebagai berikut :



Gambar 14. Pohon Solusi tahap 4, dengan 2 pilihan

Mungkin saja solusi tahap 4 memiliki 2 jawaban, yaitu Matriks [4,1] Tanpa Rotasi dan Matriks[3,2] Tanpa Rotasi seperti pada Gambar 14. Selama tidak menyalahi aturan kelayakan solusi, algoritma ini disusun untuk mencari cabang terbanyak.

Setelah semua kemungkinan selesai dicari, maka di akhir proses, algoritma Greedy akan kembali berperan pada pemilihan solusi terbaik, yaitu solusi dengan jumlah kembang api terbanyak.

4. KESIMPULAN

Algoritma *BFS* dan *Greedy* dapat diterapkan pada permainan *Rocket Mania Deluxe*. Pendekatan dari kedua algoritma ini dapat menemukan solusi paling optimal dari sekian banyak solusi yang ada. Hanya saja, algoritma hasil terapan kedua algoritma tersebut ini, masih dinilai terlalu boros, dan memakan jatah memori yang cukup besar, bahkan meskipun sudah diberlakukan fungsi kelayakan dengan algoritma *Greedy*, dengan tidak meneruskan solusi yang tidak mungkin dilakukan (hampir mirip dengan algoritma *backtrack*) namun tetap saja, pada kenyataannya algoritma ini memakan memori yang cukup besar.

Selain itu, algoritma ini tidak memilik batasan kedalaman pohon solusi yang jelas. Pencarian 1 solusi baru akan berhenti jika sudah tidak ada kemungkinan lagi / cabang sumbu yang bisa disambung kembali.

Pada kenyataannya permainan *Rocket Mania Deluxe* ini memiliki banyak syarat dalam pemilihan solusinya, oleh karena itu, saya menemukan algoritma yang saya

paparkan di makalah ini, sebagai algoritma paling efektif yang bisa dipaparkan. Namun mungkin saja masih banyak algoritma lain yang bisa digunakan untuk menemukan solusi *Rocket Mania* Deluxe dengan lebih cepa, lebih baik dan tidak menghabiskan memori.

REFERENSI

[1] Rinaldi Munir, *Diktat Kuliah Strategi Algoritmik*, Departemen Teknik INformatika Institut Teknologi Bandung 2005.

[2] Horrowitz, Ellis. *Fundamental of Computer Algorithmic*. McGraww.

[3] http://www.nearlygood.com/game/plumbertwo.html. Diakses pada tanggal 19 Mei 2007 pukul 08:00 WIB.

[4] http://www.popcap.com. Diakses pada tanggal 19 Mei pukul 08:30 WIB.

[5] http://en.wikipedia.org/wiki/Greedy_algorithm. Diakses pada tanggal 21 Mei 14:00 WIB.

[6] http://en.wikipedia.org/wiki/Breadth-first_search. Diakses pada tanggal 21 Mei 14:10 WIB.