Strategi Optimized Brute Force Pada Tent Puzzle Solver

Aji Nugraha Santosa Kasmaji - 13510092
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
newtwox@hotmail.com

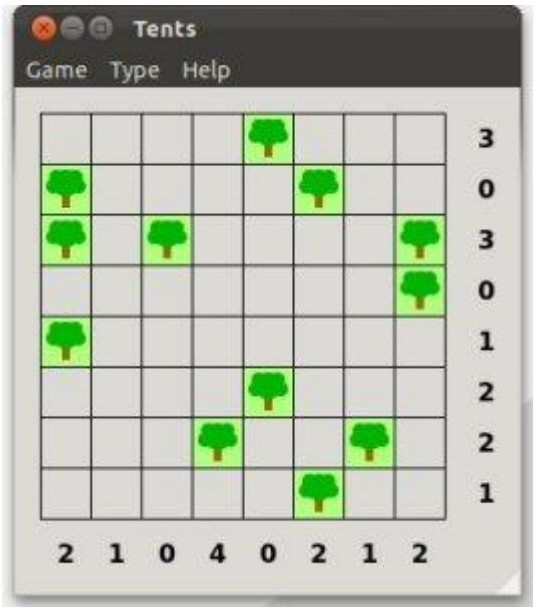
***Abstract*—Makalah ini berisikan pembahasan mengenai penerapan algoritma brute force yang telah dimodifikasi kepada salah satu jenis permainan dengan *genre* puzzle. Permainan puzzle yang dipilih adalah Tent and Tree puzzle. Tent and Tree puzzle merupakan salah satu bentuk puzzle board yang masih dalam satu jenis dengan Sudoku maupun Kakuro. Penggunaan strategi brute force dipilih sebagai alternatif pencarian solusi karena kepastiannya dalam menyelesaikan persoalan dan juga mudah untuk dipahami pergerakannya, sehingga terbilang cukup aman dan mudah untuk diimplementasikan di mesin manapun. Untuk masalah efisiensi, mungkin algoritma ini bisa dibilang cukup baik untuk mengatasi tent puzzle, namun pencarian solusi dapat dioptimisasi dengan melakukan sedikit optimisasi. Hasil optimisasi yang didapatkan ini pun nantinya dapat dibandingkan dan bersaing dengan strategi algoritma lainnya.**

***Index Terms*— brute force, strategi algoritma, board, game, puzzle, tent and tree**

I. PENDAHULUAN

A. Tentang Tent and Tree Puzzle

Permainan Tent Puzzle atau terkadang dikenal juga dengan Tent and Tree Puzzle adalah sebuah game puzzle yang bisa dibilang memiliki tipe permainan sejenis dengan Kakuro dan Sudoku, yaitu puzzle dengan tipe permainan lojik.



Gambar 1. Contoh Tents Puzzle pada Ubuntu

Sampai saat ini masih belum diketahui siapa atau berasal dari manakan puzzle ini, karena puzzle ini lebih menyebar secara elektronik melalui komputer dan sejenisnya lewat bantuan internet. Walaupun begitu, puzzle ini memiliki tingkat ketenaran yang cukup besar. Hal tersebut bisa dilihat dari tingginya situs-situs permainan yang memiliki ataupun menawarkan permainan puzzle ini, juga beberapa versi dari sistem operasi Ubuntu juga menyediakan game ini.

B. Cara Bermain Tent and Tree Puzzle

Tujuan utama dari game ini adalah untuk memasangkan setiap pohon dengan tenda-tenda, yang nantinya akan ditempatkan sesuai dengan keinginan pemain. Sehingga nantinya, pemain harus dapat menentukan mana tile rumput dan mana tile yang harus diisi oleh tenda. Namun ada beberapa kondisi khusus yang harus dipenuhi oleh pemain, yang antara lain adalah :

- Setiap Tenda harus dipasangkan masing-masing dengan sebuah pohon tertentu (artinya, jumlah tenda yang nantinya diassign pasti dengan jumlah pohon yang sebelumnya telah diketahui).
- Untuk memasangkan tenda dengan pohon, tenda harus diletakan secara berdampingan dengan pohon. Tenda boleh berdekatan dengan lebih dari satu pohon. Tenda tapi harus jelas pohon mana yang menjadi pasangannya.
- Tenda hanya dikatakan berpasangan apabila ia berada di sebelah pohon secara vertical dan horizontal. Tenda tidak dianggap berpasangan jika berada dalam posisi diagonal dengan pohon pasangannya.
- Sebuah tenda tidak boleh berdekatan dengan tenda lainnya secara horizontal, vertical, maupun diagonal dengan jarak sejauh 1 kotak dari tempat tenda acuan berada.
- Angka di samping kiri /kanan papan menandakan berapak banyak tenda yang boleh dibangun pada baris yang bersangkutan. Hal yang sama juga berlaku dengan angka di atas/bawah papan, yang membatasi jumlah tenda yang dapat dibangun pada kolom yang bersangkutan.

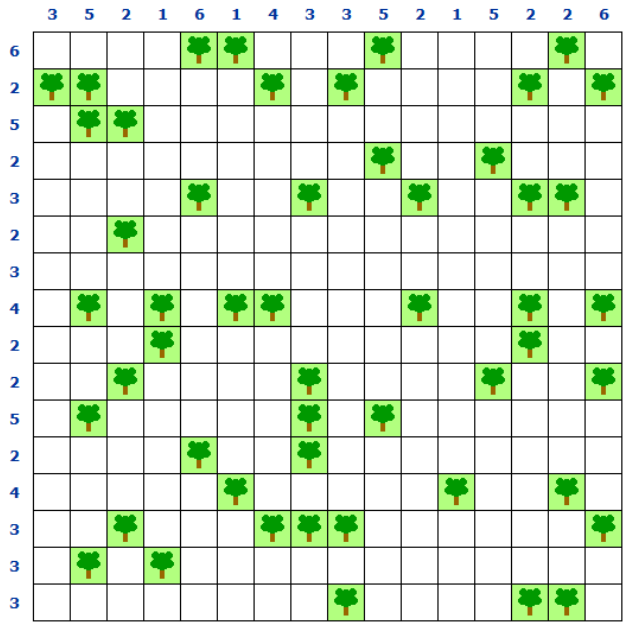
II. DESKRIPSI PERMASALAHAN

A. Kerumitan Puzzle Tent and Tree

Sama halnya dengan permainan lojik lainnya, permainan ini memiliki tingkat kesulitan yang bervariasi, mulai dari yang mudah sampai yang susah. Pada dasarnya ada tiga faktor yang menentukan tingkat kesulitan pada game ini, yaitu :

- Ukuran board yang dipakai, dimana semakin besar board, semakin banyak pula kemungkinan penempatan tenda yang dapat digunakan.
- Pemilihan batasan penempatan panah pada tiap baris dan kolom. Kerumitan yang lebih akan didapat jika hubungan antara kolom dan baris yang diberikan juga lebih rumit.
- Penempatan pohon pada board, karena pada dasarnya pohon disini merupakan acuan untuk menempatkan tenda pada board, sekaligus berfungsi sebagai penghalang bagi pemain dalam menempatkan tendanya.

Sementara berdasarkan data-data dan informasi yang telah berhasil dikumpulkan, sebuah puzzle Tent and Tree yang bagus pasti hanya memiliki satu penempatan saja yang dianggap benar, dengan kata lain hampir tidak dimungkinkan adanya *multiple solution*, kecuali puzzle yang digunakan memiliki kualitas yang cukup buruk.

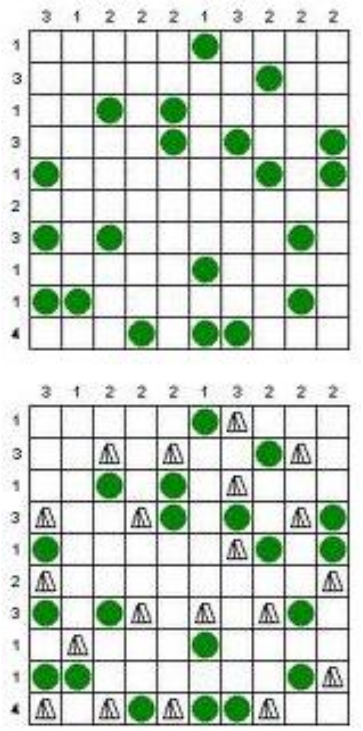


Gambar 2. Puzzle Tents dengan kerumitan menengah

Apabila diselesaikan secara manual, maka puzzle Tents ini memerlukan waktu yang cukup banyak, apalagi bila ternyata ditemukan kesalahan penghitungan pada satu titik, pemain mau tidak mau harus mengubah hampir seluruh penempatan tenda yang sebelumnya dilakukan, atau mungkin malah mengulangi puzzle ini dari awal. Hal ini dikarenakan posisi satu tenda pasti akan mempengaruhi posisi tenda lainnya.

B. Algoritma Solver yang ada dan Keefektifannya

Seperti yang telah sebelumnya dipaparkan, permainan puzzle Tents ini sekarang lebih banyak beredar secara elektronik lewat komputer. Oleh karena itu, saat ini puzzle Tents dapat diselesaikan dengan sangat cepat dengan bantuan program buatan manusia.



Gambar 3, Contoh Penyelesaian Tents Puzzle

Sampai saat ini telah banyak program yang dibuat untuk menyelesaikan puzzle ini, dan dari beberapa situs penyedia permainan puzzle Tents, telah ditemukan bahwa mayoritas problem solver program yang ada menggunakan strategi *Brute Force* dalam pengimplementasiannya.

Brute force yang diimplementasikan pada dasarnya diciptakan mengikuti pola piikir manusia yang sederhana, yaitu melakukan pengecekan untuk tiap-tiap tile satu-persatu dan menempatkannya sesuai ketentuan. Algoritma dari program yang paling sederhana akan menghasilkan sejumlah kemungkinan posisi tenda yang bisa ditempatkan, barulah disortir apakah memenuhi syarat atau tidak.

Hal itu akan menghasilkan kinerja komputasi yang cukup banyak, padahal sebenarnya bisa disingkat dan direkontruksi ulang agar menjadi lebih baik, beberapa situs ternyata juga sudah ada yang mengimplementasikan algoritma yang mungkin akan nanti dibahas pada makalah ini.

Apabila ditelusuri lebih lanjut lagi, sebenarnya pada permainan Tent Puzzle ini sebenarnya memiliki trik-trik khusus yang digunakan untuk menentukan apakah nantinya suatu tile pasti tidak dapat diisi oleh tenda atau tidak. Trik-trik seperti ini juga dapat diimplementasikan untuk memodifikasi algoritma brute force agar lebih efektif dan efisien.

III. ALGORITMA BRUTE FORCE

A. Karakteristik Brute Force

Sesuai namanya, sebuah algoritma Brute Force apabila ia mencoba semua kemungkinan yang ada dan biasanya sangat bergantung kepada kinerja dari komputer. Algoritma ini seringkali dikatakan tidak “cerdas” dan pada umumnya memiliki tingkat kemangkusan yang sangat rendah. Hal tersebut dikarenakan algoritma ini biasanya membutuhkan jumlah langkah pengerjaan yang besar dalam melakukan penyelesaian masalah, terutama jika masalah yang diberikan oleh pengguna tergolong cukup besar. Karena hal-hal tersebutlah algoritma ini juga mendapat julukan algoritma yang naïf.

Algoritma brute force biasanya menjadi pilihan yang kurang disukai, karena terkadang sangat bergantung dengan kemampuan alat komputasi dalam pengerjaannya. Hal ini jugalah yang membuat algoritma brute force digunakan sebagai alat pembanding dalam menguji kompleksitas sebuah algoritma baru.

Namun, meskipun algoritma brute force bukan merupakan teknik pemecahan masalah yang mangkus, hampir semua persoalan yang ada di dunia ini dapat diselesaikan dengan brute force, bahkan ada beberapa operasi yang sampai saat ini hanya bisa diselesaikan menggunakan brute force.

Algoritma brute force juga terkadang lebih mudah diimplentasikan dan lebih mudah dipahami karena kesederhanaannya. Oleh karena itu biasanya dalam proses pengajaran cara pertama yang diajarkan biasanya adalah brute force, sebelum masuk ke cara yang lebih canggih.

Kedua hal di atas disebabkan karena pada dasarnya brute force didasarkan pada pola pikir paling sederhana manusia, namun tidak bisa dikalkulasikan di manusia karena keterbatasan kinerja otaknya.

B. Kelebihan dan Kekurangan Brute Force

Berdasarkan paparan di atas, nerikut dapat disebutkan kelebihan dan kekurangan algoritma brute force, yang antara lain adalah:

Kelebihan :

- wide applicability
- sederhana dan lebih mudah dipahami
- merupakan dasar dan penghasil beberapa algoritma penting lain
- lebih kuat dalam menghadapi berbagai macam kemungkinan masalah

Kekurangan :

- jarang menghasilkan algoritma mangkus
- karena banyak proses yang dijalankan, kadang membutuhkan waktu lama dalam penyelesaian
- terkadang sangat bergantung pada kualitas hardware

IV. IMPLEMENTASI

A. Implementasi Normal Brute Force

Untuk Implementasi brute force normal, maka program akan menggunakan trik untuk melakukan scan dari line-per line dengan bantuan strategi algoritma lain, yaitu backtracking.

Pada saat awal program akan dijalankan, ia akan membaca ukuran tenda, lalu membuat sebuah matriks sebagai alat penyimpanan data berukuran tertentu (panjang map+1 dikalikan lebar map+1). Lalu menyimpan seluruh data jumlah tenda yang bisa didirikan tiap baris dan kolomnya, lalu diisikan ke dalam baris pertama matriks dan kolom pertama matriks. Program juga akan membaca keadaan awal puzzle yang berisikan posisi tiap pohon dan rumput, jadi nantinya tiap cell pada matriks akan memiliki atribut sebagai berikut :

- Value
  Bernilai 0 jika cell masih kosong
  Bernilai 1 jika cell berisikan pohon
  Bernilai 2 jika cell berisikan tenda
- Tipe
  Bernilai 0 jika cell soal (ketentuan jumlah tenda)
  Bernilai 1 jika non-editable
  Bernilai 2 jika editable

Setelah program melakukan inisialisasi matriksnya, maka program akan mulai berjalan dari kolom pertama dan baris pertama, sampai ia menemukan editable cell yang masih kosong.

Setelah menemukan editable cell yang kosong, program akan melakukan pengecekan tiga kali pengecekan, pertama adalah pengecekan jumlah tenda tersisa, yang kedua posisi, apakah ada pohon di bagian horizontal dan vertikalnya, dan terakhir pengecekan keberadaan tenda lain.

Apabila semua pengecekan telah terpenuhi sesuai ketentuan pada Tent Puzzle, maka kolom tersebut diassign tenda, dan program akan memproses kolom berikutnya.

Apabila jumlah tenda yang masih bisa dibangun sudah 0, program akan melakukan jump ke kolom berikutnya / bahkan ke baris berikutnya, untuk mengefektifkan pencarian solusi.

Apabila program telah menelusuri sampai suatu titik dan menemukan kebuntuan, namun masih belum berada dalam posisi yang benar dan solusi dirasa tak mungkin ditemukan, program akan melakukan backtracking ke posisi-posisi tenda sebelumnya dan menghapus tenda sebelumnya yang telah ditempatkan, lalu mencoba alternative penempatan tenda yang lainnya.

Program akan berjalan terus-menerus dan mencoba semua kemungkinan yang ada sampai dirasa jumlah tenda sudah sama dengan jumlah pohon, dan posisi akhir sudah sesuai dengan aturan-aturan Tent Puzzle, baru nantinya sisa editable cell yang masih kosong akan diiterasi dan dinisialisasi dengan tipe rumput.

B. Beberapa Trik Tent and Tree Puzzle

Berikut ini adalah beberapa trik dan teknik yang seringkali digunakan untuk membantu penyelesaian Tent Puzzle secara manual (tanpa komputasi) pada kejadian-kejadian berikut :

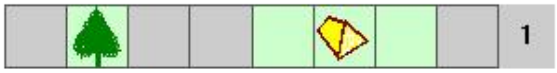
Keterangan, angka di kanan dan bawah adalah jumlah tenda yang boleh dibangun, angka/huruf di kiri dan atas menyatakan koordinat.

1. Ketika semua kotak yang tersisa pada baris sebuah board sudah dipastikan adalah tenda
   Contoh :

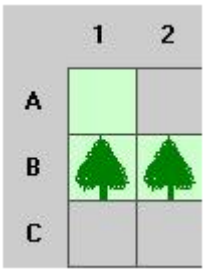
   Maka semua kotak yang tersisa dapat langsung diisikan oleh tenda.

2. Ketika semua kotak yang tersisa pada sebuah baris dapat dipastikan bukan tenda
   Contoh :

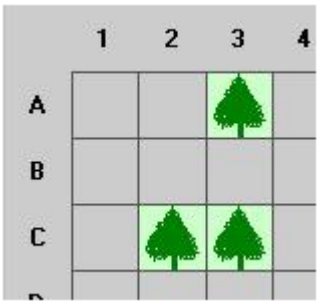
   Maka semua kotak yang tersisa dapat diinisialisasi dengan value rumput (tanah kosong)

3. Ketika hanya ada satu tempat yang dapat ditempati tenda pada suatu pohon
   Contoh :

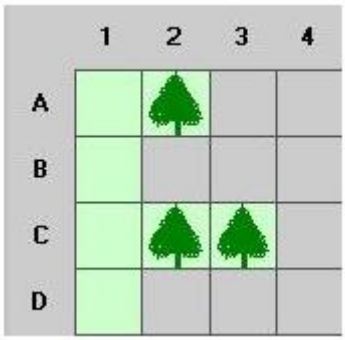
   Dengan kasus di atas, maka pasangan tenda untuk pohon B1 pastilah berada di C1

4. Kotak-kotak tertentu tanpa ada pohon berada di sekitarnya
   Contoh :

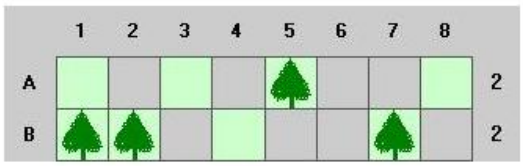
   Maka kotak-kotak A1 dan B1 dapat diisi dengan rumput (tanah kosong), karena tidak ada pohon di kanan, kiri, atas, dan bawahnya

5. Ketika ditemukan kejadian overlapping non-tent, ketika sebuah tenda akan ditempatkan
   Contoh :

   Maka B3 tidak mungkin diisi tenda, karena pasti akan berdampingan dengan tenda pada pohon A2

6. Kemungkinan suatu kotak pasti bertetanggaan dengan tenda pada suatu kolom dengan kombinasi apapun
   Contoh :

   Pada kasus di atas, dapat disimpulkan kemungkinan penempatan tenda pada kolom A adalah :
   - A2 dan A4
   - A2 dan A6
   - A2 dan A7
   - A4 dan A7

   Dan dapat kita lihat, untuk kemungkinan manapun A2 dan A4 selalu terisi, sehingga B3 yang merupakan tetangga A2 dan A4 dapat dirubah menjadi rumput terlebih dahulu.

C. Implementasi Optimized Brute Force

Implementasi optimisasi yang dapat dilakukan pada algoritma brute force ini dapat dilakukan dengan dua cara, yaitu :

- Cara Pertama
  Dengan melakukan pengecekan kemungkinan-kemungkinan 6 kondisi yang sebelumnya telah dipaparkan tersebut sebelum memulai algoritma pencarian brute force.

- Cara Kedua
  Dengan menambahkan pengecekan 6 kemungkinan tersebut ke dalam pengecekan tiap selnya, namun tidak semua dari 6 pengecekan tersebut dapat ditambahkan, yang sudah dicoba hanyalah kemungkinan ke-2 (untuk jump), dan ke-3 (untuk tiap pergantian barisnya)

Tujuan penggunaan cara yang pertama adalah untuk mengurangi / mempersempit daerah pencarian solusi, nantinya setiap cell yang dirasa nilainya sudah pasti diassign dengan value yang sesuai, dan dirubah menjadi non-editable cell, dengan kata lain memodifikasi keadaan awal soal sebelum pencarian dilakukan.

Untuk cara yang kedua digunakan untuk melakukan pengecekan tambahan tiap cellnya, dan berguna untuk melakukan jump (seperti pada istilah jump di *string matching*), sehinnga dapat melewati hal-hal yang dirasa sudah pasti benar / salah.

Cara yang sudah dicoba untuk diimplementasikan masih cara yang pertama, sementara untuk cara yang kedua hanya metode-metode khusus seperti jump dengan pengecekan kemungkinan nomor dua dan tiga saja yang sudah diimplementasikan.

V. ANALISIS

Dengan menggunakan algoritma brute force yang telah dioptimisasi, khususnya dengan cara yang pertama, maka seperti yang telah dijelaskan lingkup pencarian akan berkurang.

Mengingat teorema yang sebelumnya dikatakan bahwa kerumitan dan kompleksitas suatu soal Tent and Tree puzzle dipengaruhi oleh besar map, maka bisa dikatakan bahwa kinerja yang dilakukan program akan berkurang jauh, baik secara kuantitas (jumlah cell yang perlu dicek), maupun kualitas (informasi untuk pengecekan yang diperlukan).

Contohnya saja semisal suatu puzzle Tent and Tree memiliki 10 pohon, di setiap pohon kita tahu bahwa memiliki 4 sisi, sehingga nantinya ada 4^{10} kemungkinan penempatan tenda pada map.

Sedangkan jika salah satu kondisi saja terpenuhi, misalnya keadaan ke-3 yang paling sederhana, maka jumlah pohon misalnya berkurang satu saja, maka kemungkinan penempatan tenda pada map tinggal 4^9, dimana kita menghemat 786432 proses. Andaikan lebih besar map, lebih banyak pohon, dan lebih banyak keadaan yang telah ditemukan maka proses yang dihemat pun juga akan berlipat ganda.

Hal ini paling terlihat lewat waktu eksekusi program yang berbeda antara algoritma brute force biasa dan yang telah dioptimisasi (entah dengan cara pertama maupun kedua), dimana algoritma yang telah dioptimisasi telah menunjukan waktu eksekusi yang lebih cepat.

Namun, hasil tersebut tidak pasti / mutlak terjadi. Terkadang masih ada suatu kejadian / keadaan dimana algoritma brute force yang telah dioptimisasi memiliki waktu eksekusi yang hampir sama, atau malah sedikit lebih lama dibandingkan yang biasa.

Berdasarkan analisa, hal ini terjadi ketika tidak ada keadaan-keadaan yang sesuai dengan 6 keadaan yang sebelumnya telah dipaparkan, sehingga program bekerja secara brute force biasa, atau malah ditambah proses untuk pengecekan 6 kondisi tersebut.

VI. KESIMPULAN

Puzzle Tent and Trees ini dapat diselesaikan dengan algoritma brute force, yang juga membuktikan bahwa brute force memang memiliki lingkup pengaplikasian yang cukup besar.

Namun sisi lemahnya memang kemangkusan brute force tergolong cukup rendah. Akan tetapi, hal ini dapat ditutupi dengan memanfaatkan sifat kesederhanaan dan kenaifan brute force, sehingga algoritma yang sudah ada dapat tergolong mudah dimodifikasi dan ditambahkan ketentuan yang dapat menambah performansi program. Hal inilah yang menyebabkan brute force dapat melahirkan algoritma-algortma baru lainnya yang lebih mangkus dan krusial.

REFERENCES

[1] http://www.brainbashers.com/tents.asp
akses terakhir 12:00, 21 Desember 2012

[2] http://syndicate.yoogi.com/treetent/treetent-solved.html
akses terakhir 14:18, 21 Desember 2012

[3] http://ubuntuincident.wordpress.com/tag/puzzle/
akses terakhir 12:00, 21 Desember 2012

[4] Rinaldi Munir,"Diktat Kuliah IF3051 Strategi Algoritma", 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, 21 Desember 2012

Aji Nugraha S. K. - 13510092