Pencarian Solusi Permainan Flow Free Menggunakan Brute Force dan Pruning

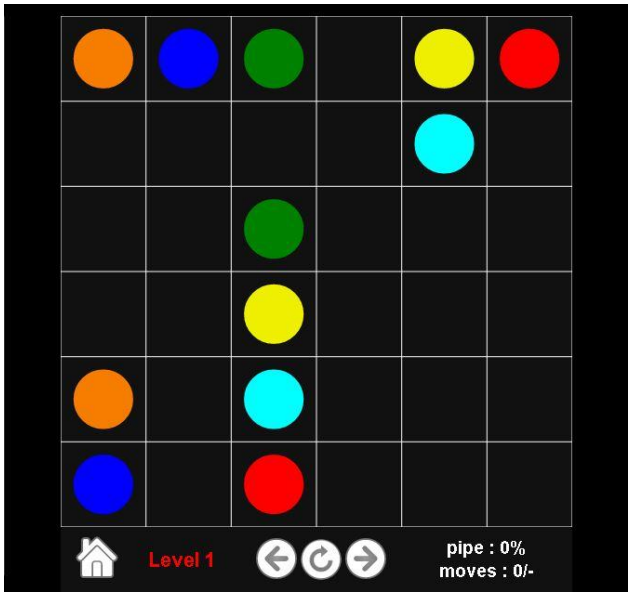
Mamat Rahmat / 13512007
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
mamat.rahmat@students.itb.ac.id

***Abstrak*—*Flow Free* adalah sebuah permainan bertema *puzzle* yang cukup populer. Solusi dari game *Flow Free* ini tidak unik. Makalah ini akan membahas metode pencarian semua solusi dari game tersebut menggunakan *brute force* dan *pruning*.**

***Kata Kunci*—*Flow Free, Puzzle, Brute Force, Pruning, Depth First Search*.**

I. PENDAHULUAN

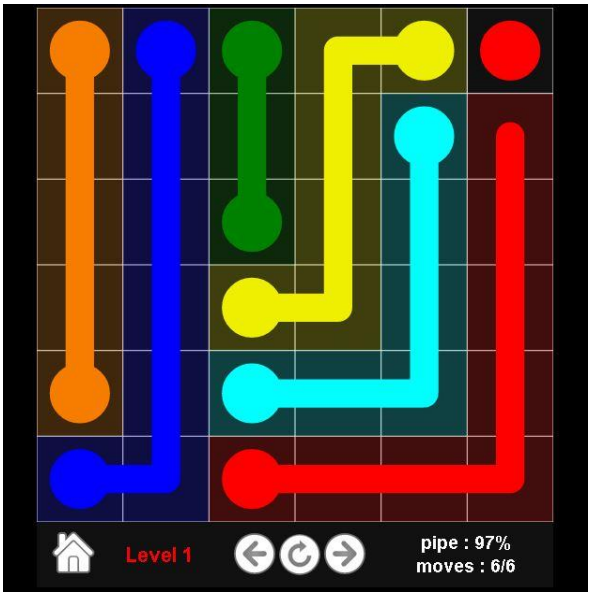
Flow Free adalah sebuah permainan bertema puzzle. Game ini tersedia dalam berbagai macam platform, dari web hingga mobila Game ini terdiri dari sebuah matriks n x n berisi beberapa pasangan lingkaran berwarna yang tersebar di dalam grid. Untuk selanjutnya dalam makalah ini sel berisi lingkaran berwarna dituliskan sebagai *colored cell*.



Contoh Tampilan Game Flow Free [2]

Tujuan dari game ini adalah membangun lintasan dengan menelusuri sel dari setiap sel berisi lingkaran berwarna ke sel pasangannya sedemikian sehingga tidak ada lintasan yang berpotongan.

Untuk memenangkan permainan, pemain harus menghubungkan setiap pasangan lingkaran berwarna dengan lintasan.



Contoh Solusi Game Flow Free [2]

Untuk membentuk lintasan, pemain harus melakukan klik atau touch pada salah satu *colored cell* lalu dengan melakukan drag, pemain menelusuri sel ke kiri, ke kanan, ke atas atau ke bawah sedemikian sehingga lintasan berakhir di *colored cell* pasangannya. Lintasan dianggap tidak valid jika tidak berakhir di *colored cell* pasangannya atau jika lintasan memotong lintasan lain yang telah dibuat sebelumnya. Pemain juga dapat menghapus lintasan yang telah dibuat tetapi dengan konsekuensi penambahan jumlah *move* yang dicatat, sehingga menyebabkan pengurangan nilai.

Tingkat kesulitan dari game ini berdasarkan kepada besar matriks yang digunakan serta sebaran pasangan dari *colored cell*.

II. LANDASAN TEORI

Untuk mendapatkan semua solusi dari permainan flow free ini dibutuhkan algoritma yang dapat mencakup semua kemungkinan kandidat solusi.

Salah satu metode yang dapat digunakan adalah metode brute force, yaitu dengan membangkitkan semua kemungkinan lintasan dari setiap pasangan dan metode pruning, yaitu memangkas pembangkitan lintasan dengan mengabaikan yang tidak mungkin menghasilkan solusi.

A. Metode Brute Force

Sesuai namanya, sebuah algoritma Brute Force merupakan algoritma yang 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.

Berikut adalah kelebihan dan kekurangan dari algoritma *brute force*:

Kelebihan

1. Dapat diaplikasikan secara luar pada berbagai persoalan
2. Sederhana dan lebih mudah dipahami
3. Merupakan dasar dan penghasil beberapa algoritma penting lain
4. Kebih kuat dalam menghadapi berbagai macam kemungkinan masalah

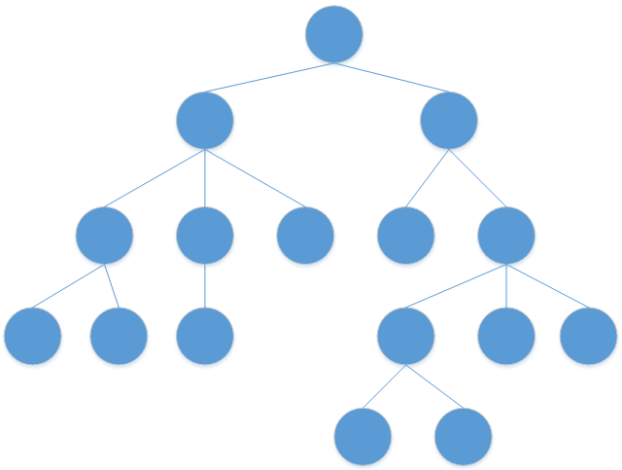
Kekurangan :

1. Jarang menghasilkan algoritma mangkus
2. Karena banyak proses yang dijalankan, kadang membutuhkan waktu lama dalam penyelesaian
3. Kerkadang sangat bergantung pada kualitas hardware

B. Metode Pruning

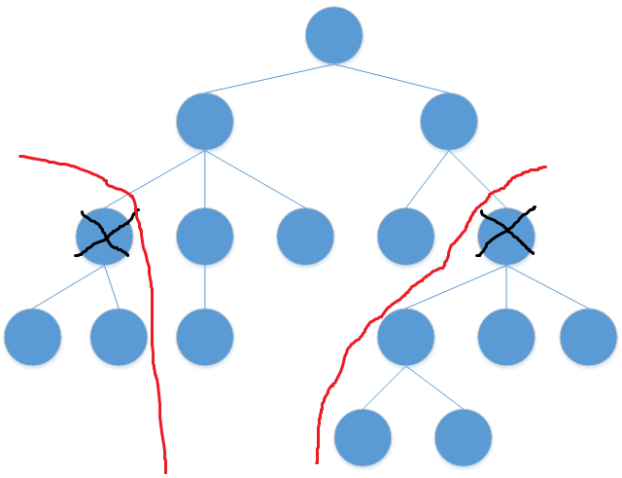
Metode pruning, yaitu memangkas pembangkitan state pencarian dengan mengabaikan state yang tidak mungkin menghasilkan solusi.

Misalkan tree di bawah merepresentasikan ruang pencarian yang harus ditelusuri untuk menemukan solusi



Contoh pohon ruang pencarian

Dengan lemma yang dapat dibuktikan secara matematis, kita dapat memangkas pencarian dengan tidak menelusuri state pencarian yang tidak perlu.



Contoh pruning pada pohon ruang pencarian

Karekteristik dari metode pruning ini adalah :

1. Semakin tinggi state atau semakin kecil kedalaman state yang kita pruning maka semakin banyak state pencarian yang kita pangkas
2. Dengan pruning yang tepat, keuntungan dari pemangkasan ini dapat mengurangi waktu pencarian secara eksponensial.

Dengan metode ini, waktu penelusuran dapat dikurangi.

Salah satu aplikasi dari metode pruning yang terkenal adalah penggunaan *Alpha-Beta pruning* pada komputer catur [4]. *Alpha-beta pruning* mengeliminasi kemungkinan langkah ke depan dari himpunan hasil analisis jika mereka terbukti tidak lebih baik dari langkah yang telah ditemukan. Pengembang dari komputer catur Deep Blue menggunakan algoritma ini dan menghasilkan inovatif teori yang disebut sebagai "*singular extensions*" [5].

Permainan catur pertama dari komputer Deep Blue melawan Kasparov

C. Algoritma Pencarian Depth-First-Search

Algoritma Depth-First Search (DFS) atau algoritma pencarian mendalam merupakan salah satu algoritma untuk melakukan traversal pada graf. Algoritma ini menelusuri simpul tetangga pertama dari simpul yang dikunjungi saat ini, kemudian masuk ke simpul anak dan melakukan hal yang sama sampai simpul tujuan ditemukan atau tidak ada lagi anak dari simpul tersebut. Jika hal itu terjadi, maka akan dilakukan runut balik (backtrack) ke simpul level atasnya.

Algoritma DFS secara alami menggunakan rekursif dalam implementasinya. Berikut ini adalah langkah-langkah traversal graf dengan memanfaatkan algoritma DFS :

1. Kunjungi simpul akar.
2. Untuk sebuah tetangga dari simpul ini yang belum pernah dikunjungi sebelumnya, anggap simpul ini sebagai akar untuk melakukan DFS pada langkah pertama.
3. Jika sudah tidak terdapat tetangga yang belum pernah dikunjungi lagi, lakukan runut balik ke simpul yang telah dikunjungi sebelumnya.
4. Penelusuran berhenti ketika simpul awal dari DFS ini sudah tidak mempunyai tetangga yang belum pernah dikunjungi.

Berikut ini adalah pseudocode penelusuran graf dengan algoritma DFS. [1]

```
procedure DFS(input v : integer)
{ Mengunjungi seluruh simpul graf dengan algoritma pencarian DFS
        Masukan: v adalah simpul awal kunjungan
        Keluaran: semua simpul yang dikunjungi ditulis ke layar
}

Deklarasi
        w : integer
Algoritma
        write(v)
        dikunjungi[v] <- true
        for tiap simpul w yang bertetangga dengan simpul v do
                if not dikunjungi[w] then
                        DFS(w)
                endif
        endfor
```

III. METODE PEMECAHAN MASALAH

A. Lemma

"Jika suatu sel berada pada semua kemungkinan lintasan suatu pasangan *colored cell*, maka lintasan solusi dari pasangan *colored cell* itu juga mengandung sel tersebut"

Hal ini benar karena himpunan lintasan solusi suatu pasangan *colored cell* merupakan himpunan bagian semua lintasan suatu pasangan *colored cell.* Sehingga karakteristik yang terdapat pada himpunan semua lintasan juga merupakan karakteriskik dari lintasan solusi.

Akibatnya, pada pembangunan lintasan untuk pasangan *colored cell* selanjutnya, sel yang pasti 'milik' *colored cell* sebelumnya tidak akan ditelusuri. Prinsip inilah yang akan dipakai pada metode pruning di pembahasan selanjutnya.

B. Model Game Flow Free

Game Flow Free dimodelkan dalam matriks nxn berisi dua buah bilangan 1, 2, 3, hingga banyaknya *colored cell* yang merepresentasikan pasangan *colored cell* pada game sebenarnya. Sementara cell yang bukan colored cell berisi bilangan nol.

C. Algoritma Penelusuran Lintasan

.Algoritma penelusuran lintasan diimplementasikan menggunakan algoritma DFS. Akar dari algortma DFS ini adalah koordinat dari *colored cell* pertama dari yang dicari. reserved[w] dimana w adalah sebuah koordinat, artinya w pasti merupakan bagian dari *colored cell* nomer reserved[w], kecuali untuk reserved[w] = 0. Simpul solusi adalah koordinat dari *colored cell* kedua.

```
procedure DFSFlowFree(input v : point, num: interger,  L
: list, z : point)
{ Menelusuri grid dengan memperhatikan variable
reserved sehingga diperoleh semua lintasan dari akar ke
simpul solusi.
  Masukan: v adalah koordinat dari sel, num adalah nomer
colored cell yang sedang dicari lintasannya, L adalah list
dari lintasan dari koordinat colored cell pertama hingga v,
z adalah kooridinat colored cell kedua.
  Keluaran: lintasan yang menghubungkan colored cell
}

Deklarasi
   w : point
Algoritma
   L.push_back(v)
   if v = z then
      masukkan L ke Lintasan[num]
   else
      for tiap point w yang bersisian dengan v do
          if w.x>=1 and w.x<=NRow and w.y>=1 and
w.x<=NCol and grid[w]=0 and not dikunjungi[w] and
reserved[w]=0 then
             dikunjungi[v] <- true
             DFS(w)
             dikunjungi[v] <-false
             endif
     endfor
```

D. Algoritma Utama

Untuk setiap *colored cell*, cari semua lintasan menggunakan algoritma DFSFlowFree, lalu iterasi untuk setiap koordinat sel, jika ia terdapat pada semua kemungkinan lintasan, tandai reserved di koordinat tersebut dengan nilai nomer dari *colored cell*. Lakukan hal ini dua kali. Jika iterasi selesai, reserved akan berisi semua cell yang merupakan milik dari *colored cell* dengan nomor reserved[koordinal sel].

```
Deklarasi
   w : point
Algoritma
   for a in [1..2]
   for k in[1..NColoredCell]
      L.empty()
      DFSFlowFree(CoordColoredCell1[k], L, I,
CoordColoredCell2[k])
      for i in [1..NRow]
        for j in [1..NCol]
          if point(i,j) is in every L[k] then
          reserved(point(i,j)) <- k
```

IV. KESIMPULAN DAN SARAN

Algoritma *brute force* dapat diterapkan sebagai solusi pemecahan masalah pencarian semua solusi permainan Flow Free yang juga membuktikan bahwa *brute force* memang memiliki lingkup penyelesaian yang cukup besar. Metode pruning menggunakan lemma dari karakteristik persoalan dapat memangkas waktu pencarian.

Algoritma *brute force* ini dapat juga ditingkatkan dengan penambahan algoritma *greedy* di dalamnya, misalnya dengan membangkitkan lintasan mulai dari colored cell dengan jarak antar pasangannya adalah paling kecil.

V. PENGANTAR

Penulis mengucapkan syukur kepada Tuhan Y.M.E dan terima kasih kepada rekan-rekan yang telah membantu penulis sehingga pengerjaan makalah ini dapat dilakukan dan diselesaikan.

1. Rinaldi M. dan Masayu L.K selaku dosen Strategi Algoritma
2. Dea Monik yang tak kenal lelah memberi semangat
3. dan rekan-rekan lain yang tidak disebutkan namanya

REFERENSI

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

[2] http://moh97.us/flow/
Diakses tanggal 12/5/2014 07.30

[3] http://blog.bigduckgames.com/2012/07/13/flow-free-faq/
Diakses tanggal 12/5/2014 08:03

[4] http://illumin.usc.edu/printer/188/deep-blue-the-history-and-engineering-behind-computer-chess/
Diakses tanggal 12/5/2014 10.00

[5] T. Anantharaman et al. "Singular extensions:Adding selectivity to brute-force searching." Artificial Intelligence, vol. 43, pp. 99-109, 1990

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, 12 Mei 2014

ttd

Mamat Rahmat / 13512007