Penerapan Algoritma Backtracking Dalam Pemecahan Permainan Magic Square

Rahadhika Wendar

Laboratorium Ilmu dan Rekayasa Komputasi
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jalan Ganesha No 10 Bandung

Email : if15107@students.if.itb.ac.id

ABSTRAK

Permainan *Magic Square* (persegi ajaib) adalah salah satu jenis permainan *puzzle* yang mempunyai banyak kemungkinan cara dalam menyelesaikan permainan ini. Untuk *Magic Square* berukuran 3x3, jika diselesaikan dengan cara *brute force*, pada kasus terburuk akan memerlukan 9^9 kemungkinan untuk menemukan solusinya. Oleh karena itu diperlukan suatu cara yang lebih baik dalam menyelesaikan permainan ini, salah satunya adalah dengan algoritma *Backtracking*.

Kata kunci : *Magic Square, Backtracking*

1. PENDAHULUAN

Permainan *Magic Square* adalah salah satu jenis permaianan *puzzle*. Gambaran umum tentang permainan ini adalah sebuah persegi berukuran NxN sel, yang harus diselesaikan dengan cara mengisi sel- sel tersebut dengan angka-angka yang unik dan jumlah angka-angka pada setiap baris dan kolom serta diagonal sama besar.

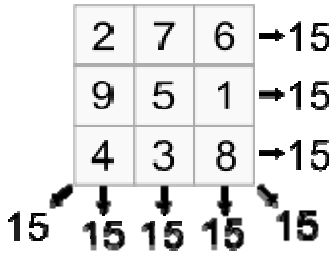
Penyelesaian permainan ini dapat ditempuh dengan beberapa cara, salah satunya adalah dengan menggunakan algoritma *Backtracking* yang berbasis pada algoritma *Deep First Search* (DFS).

2. MAGIC SQUARE

Permainan *Magic Square* adalah permainan *puzzle* yang menurut literature kuno sudah ada di China sejak 2200SM. Selain itu *Magic Square* juga ditemukan di berbagai kebudayaan seperti India kuno dan Mesir kuno. Pada kebudayaan tersebut Magic Square digunakan sebagai pertanda astronomi dan juga pencegahan terhadap wabah penyakit

Permaianan *Magic Square* adalah permaianan dengan bentuk NxN sel. Pada *Magic Square* normal, angka-angka yang bisa diisikan pada sel-sel tersebut adalah bilangan integer dari 1 sampai NxN. Dalam menyelesaikan permaianan ini ada beberapa aturan yang harus dipenuhi, yaitu :

- Angka hanya bisa digunakan sekali, dengan kata lain angka-angka pada sel haruslah unik.
- Jumlah angka-angka pada setiap kolom, baris dan diagonal haruslah sama.



Gambar 1 : Magic Square 3x3 yang sudah terselesaikan

Jumlah angka pada setiap baris, kolom dan diagonal pada *Magic Square* normal dapat diketahui dengan rumus

$$\text{Jumlah} = \frac{N^3 + N}{2}$$

3. ALGORITMA BACKTRACKING

Backtrack (runut balik) merupakan algoritma yang memperbaiki *brute force* search. Algoritma ini berbasis pada algoritma *Depth First Search* yang menggunakan struktur pohon untuk mengorganisasi ruang solusi. Pada algoritma backtrack, kemungkinan-kemungkinan solusi bisa dieliminasi tanpa benar-benar memeriksa seluruhnya dengan cara menggunakan batasan masalah yang ada. Saat ini algoritma *backtracking* sering diterapkan pada program permainan dan juga pada bidang kecerdasan buatan.

Penyelesaian permaianan *Magic Square* juga bisa diselesaiakan dengan algoritma ini.

Algoritma *Backtracking*(runut balik) mempunyai beberapa properti yaitu

1. Solusi persoalan.
Solusi dinyatakan sebagai vektor himpunan dengan n-*tuple* $X = (x_1, x_2, x_3, x_{4,...,} x_k)$, x anggota himpunan berhingga S_i

2. Fungsi Pembangkit nilai x_k
Dinyatakan sebagai T(k)
T(k) membangkitkan nilai untuk x_i yang merupakan komponen vektor solusi

3. Fungsi Pembatas.
Dinyatakan sebagai $B(x_1, x_2, x_3, x_{4,...,} x_k)$
Fungsi pembatas menentukan apakah $(x_1, x_2, x_3, x_{4,...,} x_k)$ mengarah ke solusi. Jika ya, maka pembangkitan untuk nilai x_{k-1} dilanjutkan, tetapi jika tidak maka $(x_1, x_2, x_3, x_{4,...,} x_k)$ tidak dipertimbangkan lagi dalam pencarian solusi.

Pengorganisasian solusi pada algoritma backtrack dengan cara mengorganisasikan ruang solusi dalam struktur pohon. Tiap simpul menyatakan status persoalan. Sedangkan cabang dilabeli dengan nilai x_i. lintasan dari akar ke daun menyatakan solusi yang mungkin. Seluruh lintasan dari akar ke daun membentuk ruang solusi.

Pencarian solusi pada algoritma *backtracking* diterapkan dengan cara membangun pohon ruang status secara dinamis, yaitu pohon ruang status yang dibangun bersamaan dengan pencarian solusi. Prinsip-prinsip pencarian solusinya adalah :

1. solusi dicari dengan membentuk lintasan dari akar ke daun dengan aturan D*epth First Searc*h. Simpul yang dilahirkan disebut simpul hidup. Simpul hidup yang diperluas disebut simpul-E.

2. Tiap kali simpul-E diperluas, maka lintasan yang dibangun bertambah panjang. Jika perluasan tersebut tidak mengarah pada solusi maka simpul tersebut dimatikan. Untuk mematikan simpul-E ini digunakanlah fungsi pembatas. Simpul yang udah dimatikan tidak akan diperluas lagi.

3. Jika pembentukan lintasan berakhir dengan simpul mati maka proses pencarian diteruskan dengan membangkitkan simpul anak yang lainnya. Bila tidak ada lagi simpul anak yang dapat dibangkitkan, maka pencarian solusi dilakukan dengan dengan runut-balik ke simpul orang tua. Selanjutnya simpul ini menjadi simpul-E yang baru. Lintasan dibangun kembali sampai mencapai solusi.
4. Pencarian dihentikan ketika solusi dicapai atau tidak ada lagi simpul hidup yang tersisa untuk runut balik.

4. PENYELESAIAN MAGIC SQUARE DENGAN ALGORITMA BACKTRACKING

Penyelesaian *Magic Square* dengan menggunakan algoritma *Backtracking* adalah dengan langkah-langkah sebagai berikut

1. Menyatakan properti untuk masalah ini
Properti algoritma runut balik pada kasus pengisian *Magic Square* adalah sebagai berikut :

- Solusi Persoalan

Solusi : <koordinat,nilai>
koordinat =$((x_1, y_1), (x_2, y_2),\ldots, (x_i, y_i))$ dimana x_i dan y_i anggota {0,1,2,..,ukuran sel Magic Square}
nilai : $(n_1,n_2,\ldots,n_i)$ dimana n_i adalah anggota {0,1,2,…, jumlah sel *Magic Square*}

- Fungsi Pembatas

Fungsi pembatas pada masalah ini adalah batasan-batasan pada permainan *Magic Square* yaitu :
Tidak boleh ada angka berulang dan jumlah angka pada setiap baris, kolom dan diagonal sama besar

2. Memecahkan masalah pengisian
Pemecahan masalah dimulai dari status awal, yaitu kondisi *Magic Square* yang mungkin sudah terisi beberapa buah angka sebagai problem set untuk dipecahkan. Simpul akar ini akan menjadi simpul hidup sekaligus simpul-E. simpul ini diperluas dengan mencoba mengisi koordinat yang bernilai kosong pertama yang ditemukan dengan nilai 1. jika tidak melanggar fungsi pembatas maka nilai ini di-assign pada koordinat tersebut, jika gagal maka nilai akan di-increment terus hingga ada angka yang sesuai. Ketika nilai yang sesuai maka pencarian solusi berlanjut dengan memperluas pohon dengan simpul ke koordinat selanjutnya dengan mencoba nilai untuk dimasukkan, dan jika tidak melanggar fungsi batas maka nilai di-*assign* ke koordinat tersebut kemudian simpul diperluas lagi kekoordinat selanjutnya. Hal ini berulanng hingga semua sel terisi, kemudian solusi tersebut dites apakah merupakan sebuah *magic square* yang valid. jika tidak, maka akan terjadi *backtrack* ke simpul sebelumnya dan mencoba untuk mencari nilai lain yang sesuai dan akan mengeset ulang nilai yang baru ke koordinat dan mencoba untuk memperluas simpul. Cara ini berlangsung berulang-ulang hingga solusi tercapai yaitu semua sel terisi dengan angka dan sesuai

dengan fungsi pembatas atau tidak ada lagi simpul yang bisa diperluas.

Pseudocode algoritma runut balik untuk permasalahan ini

```
boolean solve(int[][] square) {
  if (sel terisi semua){
    if (Terselesaikan(square)) {
       cetakSolusi
       return true
    }
  } else {// masih ada sel kosong
  //cari angka belum terpakai
     int[] numbers = AngkaBebas(square);
     for (int i=0; i<numbers.length; i++) {
  // isi sel kosong
     put(numbers[i],square);
  // rekursi
     solve(square);
  // hapus angka yang sudah digunakan
     remove(square);
     }
   }
}
```

Dengan algoritma diatas maka *Magic Square* akan divalidasi ketika semua sel terisi.

Dengan program yang dibuat oleh Kees Huizing (www.cs.princeton.edu/introcs/14array/MagicSquare.java), maka solusi pertama dari *Magic Square* kosong berukuran 3x3 diperoleh pada percobaan memasukkan angka ke -187768, Sedangkan pengisian Magic Square dengan menggunakan *brute force* untuk kasus terburuk memerlukan 9^9 (387420489) percobaan memasukkan angka.

5. KESIMPULAN

Algoritma *Backtracking* yang berbasis pada algoritma *Deep First Search* dapat menyelsaikan permainan *Magic Square* dengan efektif sebab pada prinsipnya, dengan Algoritma Backtrack kita tidak perlu memeriksa semua kemungkinan solusi yang ada. Pencarian hanya mengarah pada solusi yang dipertimbangkan saja.

REFERENSI

[1] Huizing, Keis, *Magic Square*, http://www.cs.princeton.edu/introcs/14array/MagicSquare.java, diakses pada tanggal 22 Mei 2007 pukul 13.30

[2] Munir,Rinaldi, *Diktat Kuliah IF2251 Strategi Algoritmik*, Program Studi Teknik Informatika ITB,2005

[3] Wikipedia, *Magic Square*, http://en.wikipedia.org/wiki/Magic_square.htm, diakses tanggal 22 Mei 2007 pukul 13.30