Pencarian Solusi Permainan "Fig-Jig" Menggunakan Algoritma Runut-Balik

Edward Hendrata (13505111)

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl Ganesha 10, Bandung
E-mail: if15111@students.if.itb.ac.id

Abstract

Salah satu cara untuk menyelesaikan suatu permainan adalah mencoba semua kemungkinan yang ada sampai didapatkan solusi final yang memenuhi batasan dari permainan tersebut (metode *brute force*). Namun, melakukan pencarian solusi dengan mencoba semua kemungkinan memakan waktu dan resource yang sangat banyak. Untuk mengurangi pencobaan kemungkinan solusi, dapat digunakan algoritma runut balik (backtracking). Algoritma ini hanya akan melakukan pencobaan kemungkinan yang mengarah ke solusi saja. Salah satu permainan yang dapat diselesaikan dengan algoritma runut-balik adalah permainan Fig-Jig

Kata Kunci : fig-jig, runut-balik, backtrack

1 Pendahuluan

1.1 Sekilas tentang permainan

Permainan adalah kegiatan yang dibatasi oleh peraturan. Orang-orang melakukan permainan dengan tujuan sebagai rekreasi dan mengembangkan kemampuan fisik ataupun mental.

Permainan dapat dilakukan dengan berbagai macam cara. Dapat dilakukan dengan beberapa orang baik saling berhadapan maupun saling berlawanan. Untuk melakukan permainan, biasanya diperlukan suatu keahlian misalnya seperti pada permainan catur. Dalam bermain catur, kita dituntut untuk memiliki kemampuan analisis dan berstrategi dengan baik agar dapat memenangkan permainan tersebut.

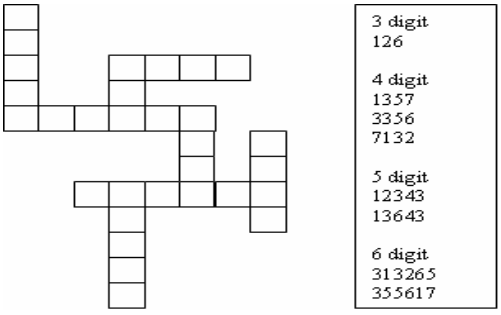
Permainan sudah dilakukan sejak ribuan tahun yang lalu. Orang – orang pada jaman dahulu menggunakan ranting pohon untuk melukis gambar kotak yang digunakan sebagai papan dan menentukan aturan mainnya pada saat itu juga. Jadi, pada saat itu juga, belum ada permainan yang mempunyai peraturan tetap.

Saat ini, telah berkembang beribu-ribu permainan yang telah mempunyai peratutan tetap. Salah satu dari permainan tersebut adalah permainan Fig-Jig.

2 Dasar Teori

2.1 Fig-Jig

Fig-Jig merupakan permainan yang dikenalkan oleh majalah intisari. Fig-Jig dimuat sebagai permainan berhadiah bagi pembacanya yang berhasil menemukan solusi dari permainan tersebut. Pada dasarnya, Fig-Jig adalah permainan yang mirip dengan teka-teki silang. Permainan Fig-Jig mempunyai kotak-kotak yang sama dengan teka-teki silang, namun pada teka-teki silang, diberikan soal-soal yang jawabannya dapat ditulis pada kotak yang diketahui posisinya, sedangkan pada Fig-Jig diberikan jawaban-jawaban yang harus ditulis pada kotak-kotak yang tidak diketahui posisinya. Jawaban yang diberikan berupa angka dengan digit yang berbeda-beda. Tugas dari pemain adalah mengisikan jawaban-jawaban tersebut pada kotak yang tersedia sehingga semua kotak terisi penuh.



Gambar 1. contoh Fig-Jig sederhana

2.2 Algoritma runut-balik

Runut-balik *(backtracking)* adalah algoritma yang berbasis pada *DFS* untuk mencari solusi persoalan secara lebih mangkus. Runut balik, yang merupakan perbaikan dari algoritma brute force, secara sistematis mencari solusi persoalan di antara semua kemungkinan solusi yang ada. Dengan metoda ini, kita tidak perlu memeriksa semua kemungkinan solusi yang ada. Hanya pencarian yang mengarah ke solusi saja yang selalu dipertimbangkan. Akibatnya, waktu pencarian dapat dihemat. Runut-balik lebih alami dinyatakan dalam algoritma rekursif. Kadang-kadang disebutkan pula bahwa algoritma runut balik merupakan bentuk tipikal dari algoritma rekursif.

Di sini, kita hanya akan meninjau pencarian solusi pada pohon ruang status yang dibangun secara dinamis. Langkah-langkah pencarian solusi adalah sebagai berikut :

1. Solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan pembentukan yang dipakai adalah mengikuti metode pencarian mendalam (*DFS*). Simpul-simpul yang sudah dilahirkan dinamakan simpul hidup (*live node*). Simpulsimpul yang sudah dilahirkan dinamakan simpul hidup (*live node*). Simpul hidup yang sedang diperluas dinamakan simpul-E (*expand node*). Simpul dinomori dari atas ke bawah sesuai dengan urutan kelahirannya.
2. Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambah panjang. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi, maka simpul-E tersebut dibunuh sehingga menjadi simpul mati (*dead node*). Fungsi yang digunakan untuk membunuh simpul-E adalah dengan menerapkan fungsi pembatas (bounding function). Simpul yang sudah mati tidak akan pernah 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 dilanjutkan dengan melakukan runut balik ke simpul hidup terdekat (simpul orangtua). Selanjutnya, simpul ini menjadi simpul-E yang baru. Lintasan baru dibangun kembali sampai lintasan tersebut membentuk solusi.
4. Pencarian dihentikan bila kita telah menemukan solusi atau tidak ada lagi simpul hidup untuk runut-balik

3 Algoritma runut balik pada Fig-jig

3.1 Pemindaian soal

Pada tahap ini, program akan memindai soal untuk mendapatkan data-data yang diperlukan.

1. ListJawaban : array of array of string

   Pertama-tama, semua isi data diinisialisasikan " ". Data ini kemudian akan diisikan dengan jawaban-jawaban yang tersedia pada soal. Contoh pada gambar 1, listJawaban akan berisi :

   ListJawaban[3][0] = "126"

   ListJawaban[4][0] = "1357"

   ListJawaban[4][1] = "3356"

   ListJawaban[4][2] = "7132"

   ListJawaban[4][3] = " "

   … dst

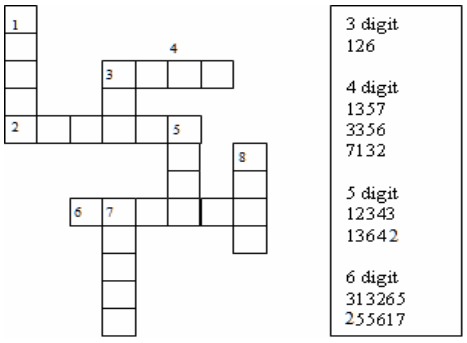
   Index dari array pertama meyatakan jumlah digit, dan index dari array kedua menyatakan data ke-.

2. soal : { isi : array of string

   length : integer }

   ListSoal : array of soal

Data ini merupakan list dari soal yang tersedia. Contoh pada gambar 1 :



Gambar 2. Soal yang telah dibuat listnya

ListKotak terisi sebanyak 6 elemen soal dengan inisisalisasi awal isi = “ “. isi pada ListKotak diisikan apabila pada soal mula-mula sudah ada kotak yang terisi.

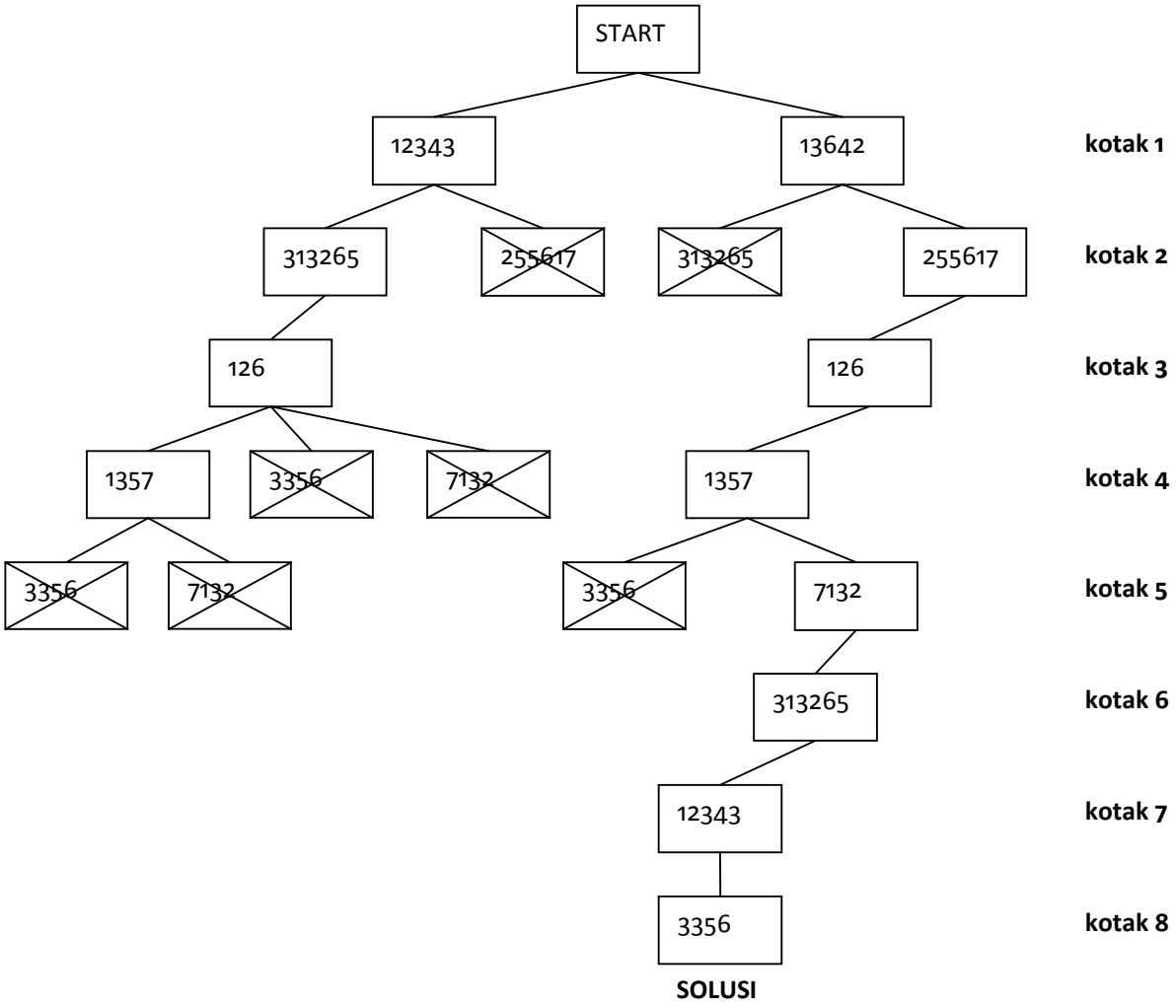
3.2 Pseudo code algoritma runut balik pada Fig-Jig versi rekursif

```
function solusi(idx : integer) : boolean {
Mencari solusi Fig-Jig menggunakan algoritma backtrack
}
Algoritma
if( idx == lastIdx+1 ) then {basis}
      return true
      {seluruh kotak soal sudah terisi}
else // reccurent
if (ListSoal[idx].isi <> “ “) return solusi(idx+1) // skip soal yang
                                                   // terisi
int i = 0; // inisialisasi ListJawaban elemen ke 0
int jmlDigit = ListSoal[idx].length // jmlDigit kotak soal
while (ListJawaban[jmlDigit][i] <> “ “) { // masih ada elemen

      if (isValid(ListJawaban[jmlDigit][i]], ListSoal[idx].isi)) {
      ListSoal[idx].isi = ListJawaban[jmlDigit][i]
      if (solusi(idx+1)) return true
      i = i+1
}
ListSoal[idx].isi = “ “ // kosongkan karena akan backtrack
return false // backtrack

}

------------------------------------------------------------------

Function isValid(jawaban : string, soal : string) {
      valid : boolean
      i : integer
Algoritma
      i = 0
      valid = true
      while (not(endofstring)) {
            if ((soal[i] <> “ “) || (jawaban[i] <> soal[i])) {
                  valid = false // tidak valid
                  break
            }
      }
      return valid
}
```

Pseudo code pencarian solusi Fig-Jig menggunakan algoritma backtrack

3.3 Skema Proses algoritma

Dijalankan pada soal gambar 1 :



= tidak valid

4 Kesimpulan

Permainan Fig-Jig dapat dapat diselesaikan dengan algoritma runut-balik. Algoritma runut-balik sudah cukup mangkus untuk diimplementasikan pada permainan Fig-Jig daripada algoritma brute force karena tidak perlu mencari seluruh kemungkinan solusi. Namun, algoritma runut-balik mempunyai kekurangan yaitu membutuhkan resource memori yang sangat besar karena sifatnya yang rekursif.

5 Referensi

[RI] Munir, Rinaldi, "Strategi Algoritmik", ITB, 2007.

[WI] http://en.wikipedia.org/ Tanggal akses: 20 Mei 2007 pukul: 15.00