PENERAPAN ALGORITMA RUNUT-BALIK DALAM MENCARI SOLUSI PERMAINAN AKARI (LIGHT UP)

Ray Suryadiptya

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if15012@students.if.itb.ac.id

ABSTRAK

Algoritma runut-balik (*backtracking*) banyak digunakan dalam pencarian solusi permainan-permainan teka-teki yang membutuhkan logika. Walaupun memakai resource komputer cukup banyak, algoritma ini sudah terbukti cukup mangkus, dan dapat menyelesaikan masalah sama mudahnya dengan algoritma *Brute Force*, dengan kecepatan proses yang jauh lebih cepat. Algoritma ini juga lebih efektif dari DFS karena langsung memangkas kemungkinan solusi yang tidak memenuhi. Permainan teka-teki seperti *Akari* dapat diselesaikan dengan algoritma ini. Peletakkan lampu yang termasuk cukup sulit dapat diselesaikan dengan cepat, walaupun tetap saja tergantung pada kemampuan kinerja komputer.

Kata kunci: *backtracking, runut-balik, akari, light up*

1. PENDAHULUAN

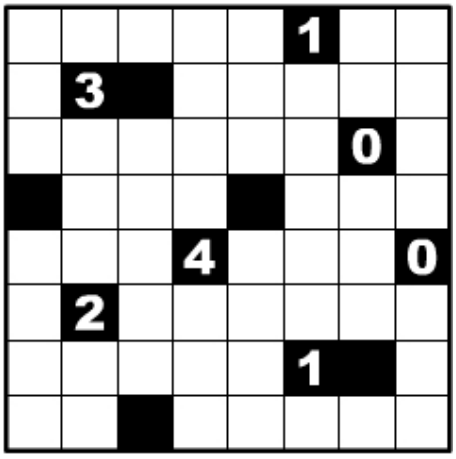
Akhir-akhir ini banyak bermunculan game teka-teki yang berbasis pada logika. Pada makalah ini, yang akan dibahas adalah salah satu jenis permainan teka-teki tersebut, yang bernama *Akari*, atau dalam bahasa Inggris, *Light Up*. *Akari* merupakan permainan yang dimainkan dalam sebuah tabel yang mempunyai jumlah baris dan kolom yang sama. Pada tabel tersebut terdapat sel-sel yang berwarna putih dan hitam. Tujuan dari permainan ini adalah menerangi semua sel putih dengan "lampu". Saat diletakkan, "lampu" tersebut akan langsung menerangi sel-sel yang berada di baris yang sama, ataupun dalam kolom yang sama dengan lampu tersebut, tetapi, peletakkan "lampu" tidak boleh berada di baris ataupun kolom yang sudah diterangi lampu lainnya. Sel hitam merupakan sebuah sel yang tidak tembus cahaya, jadi nyala lampu tidak bisa melewati sel hitam tersebut, dan berfungsi sebagai pembatas.

Untuk menyelesaikan permainan Akari, dapat digunakan algoritma runut-balik (*backtracking*), yang berbasis pada DFS (*Depth First Search*), di mana pengisian lampu langsung pada sel yang mungkin diletakkan lampu, kemudian melakukan runut-balik apabila ternyata solusi tidak ditemukan. Algoritma ini lebih mangkus daripada DFS karena mempunyai batasan-batasan yang langsung mengeliminasi sub-pohon jika ditemukan adanya *dead node* (simpul pohon yang melebihi ketentuan batas) dan langsung melakukan pencarian di sub-pohon yang lain. Backtracking juga lebih mangkus dari *Brute Force* mempunyai terlalu banyak kemugkinan solusi.

2. AKARI

2.1. PENGENALAN AKARI

Akari adalah sebuah permainan teka-teki logika yang diciptakan oleh perusahaan Nikoli pada tahun 2001. Seperti yang dituliskan pada bagian pendahuluan, Akari terdiri atas kotak hitam dan putih, di mana semua kotak putih harus diterangi dengan meletakkan lampu. Peletakkan lampu harus sesuai dengan ketentuan-ketentuan yang berlaku. Permainan akan berakhir apabila semua kotak putih berhasil diterangi, sesuai dengan ketentuan-ketentuan permainan.



Gambar tabel permainan Akari

2.2. PERATURAN DALAM AKARI

Kotak putih merupakan sel-sel yang harus diterangi lampu. Kotak hitam adalah sel yang tak tembus cahaya, dan berfungsi sebagai pemisah. Lampu harus diletakkan pada kotak putih, tidak boleh diletakkan pada kotak hitam. Lampu juga tidak boleh diletakkan pada baris ataupun kolom yang sudah diterangi lampu lain. Pada kotak hitam, ada kemungkinan terdapat sebuah angka. Angka-angka tersebut merupakan jumlah lampu-lampu yang langsung bersebelahan dengan kotak hitam tersebut. Sebagai contoh, jika kotak hitam menunjukkan angka 3, berarti ada lampu yang letaknya di atas, bawah, kiri, ataupun kanan dari kotak tersebut berjumlah 3 buah. Angka 0 berarti tidak ada lampu yang berdekatan dengan kotak hitam tersebut, dengan kata lain, tidak boleh ada lampu yang diletakkan bersebelahan dengan angka 0.

2.3. LANGKAH PENYELESAIAN

Akari dapat diselesaikan dengan proses penandaan, analisa, dan peletakkan. Setelah proses analisa, dilakukan kembali siklus ini terus-menerus hingga tujuan permainan ini, menerangi semua kotak putih, selesai dilakukan.

Pada proses penandaan, sel-sel yang tidak mungkin diletakkan lampu langsung ditandai. Sel-sel yang mungkin diisi lampu juga ditandai, terutama yang berdekatan dengan angka. Di sekitar angka 0 bisa dipastikan tidak ada lampu. Seluruh kotak di dekat angka 4 pasti diletakkan lampu. Untuk memudahkan, tandailah lampu-lampu di dekat angka yang besar terlebih dahulu.

Pada proses analisa, letak lampu-lampu ditentukan seefektif mungkin, sehingga jika ada sel kosong, harus ada kotak kosong pula di baris atau kolom yang sama untuk meletakkan sebuah lampu. Diperiksa kemungkinan-kemungkinan terbaik peletakkan lampu pada sel-sel yang sudah ditandai.

Pada proses peletakkan, lampu diletakkan. Pada baris dan kolom yang sama dengan sel lampu tersebut, sampai terkena sel hitam, ditandai bahwa sel tersebut sudah diterangi. Kemudian, kembali ke proses penandaan, dan melakukan penandaan sesuai dengan daerah-daerah yang sekarang sudah diterangi lampu saat ini.

2.4 KOMPUTASI PADA AKARI

Akari merupakan salah satu bentuk dari masalah *NP-complete.* Permasalahan NP-complete adalah salah satu permasalahan yang sulit diselesaikan, namun sering ditemui, seperti layaknya pengambilan keputusan. Pada dasarnya, Akari dapat diselesaikan dengan algoritma traversal graf dengan membangun seluruh pohon kemungkinan.

3. RUNUT-BALIK

3.1 GAMBARAN SINGKAT ALGORITMA RUNUT-BALIK

Algoritma runut-balik (backtracking) berbasis pada DFS dalam pencarian solusi. Perbedaannya dengan DFS adalah pada penambahan kriteria, yang memungkinkan algoritma ini langsung memangkas kemungkinan-kemungkinan solusi yang pasti tidak memenuhi. Algoritma runut-balik bukanlah algoritma terbaik, melainkan hanya salah satu cara mencari solusi dengan lebih cepat. Karena hanya mencari kemungkinan yang mengarah ke solusi, pemakaian algoritma ini memungkinan pendapatan solusi dengan jauh lebih cepat dibandingkan algoritma brute force. Algoritma runut-balik ini dapat dinyatakan dalam bentuk iteratif maupun rekursif, tetapi yang akan dipakai pada pencarian solusi Akari adalah bentuk iteratif.

3.2. PRINSIP RUNUT-BALIK

Pencarian solusi dengan runut-balik dilakukan secara dinamis, yaitu pohon status dibangun da dimatikan setiap saat terjadi langkah. 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*). Sedangkan, simpul hidup yang sedang diperluas dinamakan simpul-E (*expand node*). Simpul dinomori dari atas ke bawah sesuai dengan urutan kelahirannya.
2. Setiap terjadi perluasan simpul, lintasan yang dibuatnya bertambah panjang. Jika didapatkan sebuah simpul yang di luar kriteria dan tidak menuju pada solusi, simpul tersebut dimatikan menjadi simpul mati (*dead node*). Simpul yang sudah menjadi simpul mati tidak akan diperluas, karena jika diperluas pun, hasilnya tidak akan menuju ke solusi.
3. Jika pembentukan lintasan berakhir dengan simpul mati, maka akan terjadi runut-balik, dan akan meneruskan pencarian dari simpul orang tua yang masih hidup terdekat. Jika ternyata tidak ada simpul orang tua yang masih dapat hidup dan membangkitkan simpul anak, akan terus terjadi runut balik dan meneruskan pencarian ke sub-pohon yang lain. 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.

3.3. BATASAN

Pada pencarian solusi permainan teka-teki Akari, pencarian hanya dilakukan dengan menggunakan algoritma runut-balik yang iteratif.

4. PENERAPAN RUNUT-BALIK PADA PENCARIAN SOLUSI AKARI

Solusi dari Akari sangat sulit ditemukan dengan menggunakan algoritma brute force, karena banyaknya batasan-batasan dalam peletakan lampu. Garis besar penggunaan runut-balik dapat dilihat pada pseudo-code berikut dengan besar tabel Akari adalah nxn :

```
procedure IsTerang( ) {
//sebuah prosedur untuk mengecek apakah sel tersebut //sudah diterangi
   if (ada lampu dalam deretan baris ataupun kolom yang tak terpisah sel hitam) {
     return true;}
    else {return false;}
}
```

```
procedure IsValid( ) {
//sebuah prosedur untuk mengecek apakah sel tersebut //dapat diisi lampu
   if  (IsTerang( )){
     return false;
   } else if (IsBeside0( )){
     return false;
   }  else  if  (jumlah  lampu  sekitar  sel  hitam
         melebihi angka){
     return false;
   }  else {return true;}
}
```

```
procedure Solving( ) {
//prosedur untuk menemukan solusi
//Kamus
    int fase=4;
    boolean backtrack = false;
    boolean finish=false;
    boolean nosolution = false;
//Algoritma
  while (!finish && !nosolution) {
      if(!backtrack){
        if (fase>0){
          while(tidak dekat kotak hitam bernomor
          sesuai fase){next;}
        }
         else{while(belum     menemukan     kotak
            kosong){next;}}
       //ditemukan kotak valid
       if (IsValid( )){putLamp( );}
          else{next;}
       if (IsDone( ){
          //lampu sudah menerangi semua
       finish = true;
       } else if (pointer sudah di ujung){
             if (fase==0){backtracking = true;}
             else{
              setPointer(0,0);
               fase--;
             } //cek fase berikut
             }
       else{
       //backtracking
       if (LampuExist()){
         Searchprev();
         //mundur ke lampu sebelumnya
         DelLamp(); //hapus lampu
         next; //memajukan pointer sekali
         backtracking = false;
       } else if (fase == 4){nosolution = true;}
       else {fase++;
            setPointer(n-1,n-1);
           }
       }
}
//tidak ada solusi, atau selesai
if(finish) {System.out.println("Selesai");}
else {System.out.println("Tak ada solusi");}
```

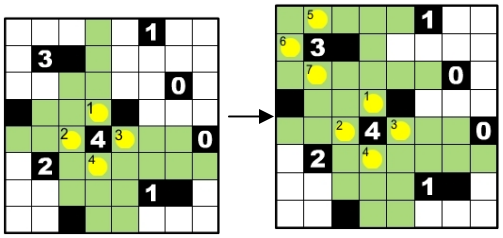
Seperti yang dilihat, aplikasi runut-balik pada Akari cukup rumit jika dibandingkan dengan runut-balik pada Sudoku.

5. CONTOH PENGGUNAAN PSEUDO-CODE

Sekarang akan diperlihatkan langkah-langkah penggunaan *pseudo-code* di atas. Sebagai contoh, digunakan soal yang terdapat pada halaman pertama. Bagian yang sudah diterangi akan berwarna hijau, lampu berupa lingkaran kuning, tempat yang dicoba meletakkan lampu tetapi gagal karena tidak valid akan diberi kotak hitam kecil.

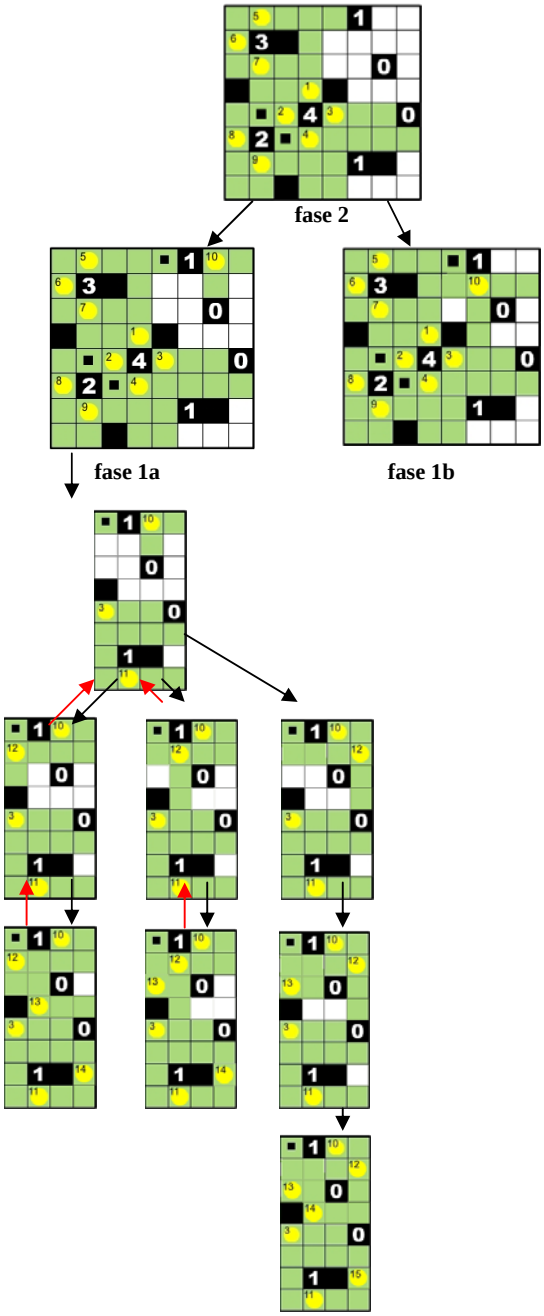
Dimulai dari fase=4, akan diletakkan lampu di sekitar angka 4, sudah pasti semua sel di sekitarnya diletakkan lampu. Urutan peletakkan sesuai dengan angka kecil di dekat lampu. Karena peletakkan sudah jelas, 4 lampu akan langsung diletakkan.

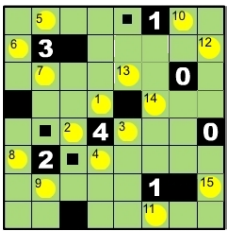
Karena fase 3 juga hanya mempunyai tempat yang pasti, maka peletakkan 3 lampu di dekat angka 3 langsung diletakkan.



peletakkan fase 4 **peletakkan fase 3**

Pada fase 2, terdapat 2 tempat yang sekarang sudah diterangi, tidak boleh diletakkan lampu. Tetapi, karena tempatnya juga jelas, proses validasi peletakkan lampu tidak diperlihatkan. tetapi setelah mencapai fase 1, selain ada 1 sel tak bisa diisi, ada 2 alternatif peletakkan lampu di dekat angka 1. Karena berbasis DFS, peletakkan lampu kanan akan diproses dulu sampai selesai. untuk menghemat tempat, 4 kolom sebelah kiri dianggap sudah selesai, dan tidak akan diperlihatkan saat memasuki fase 0. Panah merah merupakan runut-balik.





solusi

6. ANALISIS PERCOBAAN

Berdasarkan percobaan di atas, dapat dilihat bahwa keefektifan algoritma runut-balik bergantung pada langkah yang diambil terlebih dahulu. Jika solusi lebih dekat ke langkah yang pertama diambil (sub-pohon kiri), maka solusi akan lebih cepat didapat. Jika ternyata solusi berada di sub-pohon kanan, algoritma runut-balik yang berdasarkan DFS ini tidak akan terlalu efektif. Kriteria yang dipakai dalam runut-balik bergantung pada permasalahan yang akan diselesaikan. Semakin banyak kriteria untuk mencegah pengambilan langkah yang menjauhi solusi, akan semakin efektif pula algoritma ini.

7. KESIMPULAN

Penyelesaian permainan Akari dapat dilakukan dengan menggunakan algoritma runut-balik. Kecepatan proses pencarian solusi pun masih dalam tingkat yang dapat diterima. Kemampuan dan efisiensi algoritma runut-balik jauh melebihi Brute Force dan DFS, tetapi, hal ini bukan berarti algoritma runut-balik adalah algoritma terbaik dalam pencarian solusi permainan Akari, sebab masih terdapat banyak algoritma lain yang mungkin bisa diterapkan dalam pencarian solusi.

8. REFERENSI

[1] Munir, Rinaldi. *Diktat Kuliah Strategi Algoritmik IF2251 Strategi Algoritmik*. Bandung. 2007.

[2] Nikoli Co. Ltd. http://www.nikoli.co.jp/en/ diakses pada hari Sabtu, 19 Mei 2007.

[3] Wikipedia. *Light Up*. http://en.wikipedia.org/wiki/Light_Up diakses pada hari Sabtu, 19 Mei 2007.

[4] Wikipedia.*NP-complete problems*. http://en.wikipedia.org/wiki/List_of_np-complete_problems#Games_and_puzzles diakses pada hari Sabtu, 19 Mei 2007.