PENGGUNAAN ALGORITMA RUNUT BALIK DALAM MEMECAHKAN PERMAINAN *RUBIK'S CUBE*

Denny Nugrahadi

Teknik Informatika ITB
Alamat : Jl. Sangkuriang M2
e-mail : if14054@students.if.itb.ac.id

ABSTRAK

Algoritma Runut Balik adalah algoritma yang berbasis DFS yang mencari semua solusi persoalan di antara semua kemungkinan solusi yang ada dan bukan mencoba semua kemungkinan yang belum tentu mengarah pada solusi, seperti pada algoritma *brute-force*. Algoritma ini sering digunakan dalam persoalan-persoalan bertipe *puzzle* seperti labirin, sudoku, *puzzle* geser, serta permainan-permainan lain. Karena itu, penulis mencoba menganalisa seberapa efisien kelihatannya algoritma ini dalam menemukan solusi dari permainan *puzzle* yang dikenal sebagai *Rubik's Cube.*

Kata kunci: *Rubik's Cube,* algoritma runut balik, *backtrack*

1. PENDAHULUAN

Sebelum membahas tentang cara menggunakan algoritma runut balik dalam mencari solusi dari permainan *Rubik's Cube*, pertama-tama harus dijabarkan terlebih dahulu apa yang dimaksud sebagai *Rubik's Cube*. Permainan tersebut adalah semacam *puzzle* 3 dimensi berupa kubus 6 warna berukuran nxnxn yang diciptakan oleh seorang Hungaria bernama Ernő Rubik pada tahun 1974. Pada umumnya, *puzzle* yang terkenal adalah yang berukuran 2x2x2(*Pocket Cube*), 3x3x3(*Rubik's Cube*), 4x4x4(*Rubik's Revenge*), dan 5x5x5(*Professor's Cube*).

Gambar 1. Contoh *Rubik's Cube* (Dari kiri ke kanan: *Rubik's Revenge, Rubik's Cube, Professor's Cube, Pocket Cube*)

Cara permainan adalah sebagai berikut: seperti dilihat pada gambar 1, tiap sisi kubus memiliki warna yang tidak sama pada tiap petaknya. Tujuan permainan adalah bagaimana caranya agar tiap sisi kubus memiliki warna yang sama, seperti yang ditunjukkan pada gambar 2. Hal ini dilakukan dengan cara memutar-mutar sisi-sisi kubus sedemikian rupa sehingga akhirnya diperoleh solusi.

Gambar 2. Contoh *Rubik's Cube* yang telah terselesaikan

Gambar 3. Contoh *Rubik's Cube* yang salah satu sisinya sedang diputar

2. METODE

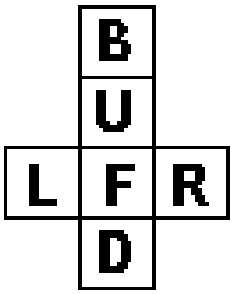
Berikut akan dijabarkan definisi-definisi yang perlu diketahui dan penggunaan algoritma runut balik dalam menemukan solusi.

2.1 Notasi-notasi

Untuk menyederhanakan persoalan, kubus yang akan kita bahas adalah *Pocket Cube.* Sebelum lebih lanjut, kita harus mendefinisikan dulu notasi untuk sisi-sisi kubus.

Untuk sisi-sisi kubus, notasi yang umum dipakai adalah sebagai berikut:

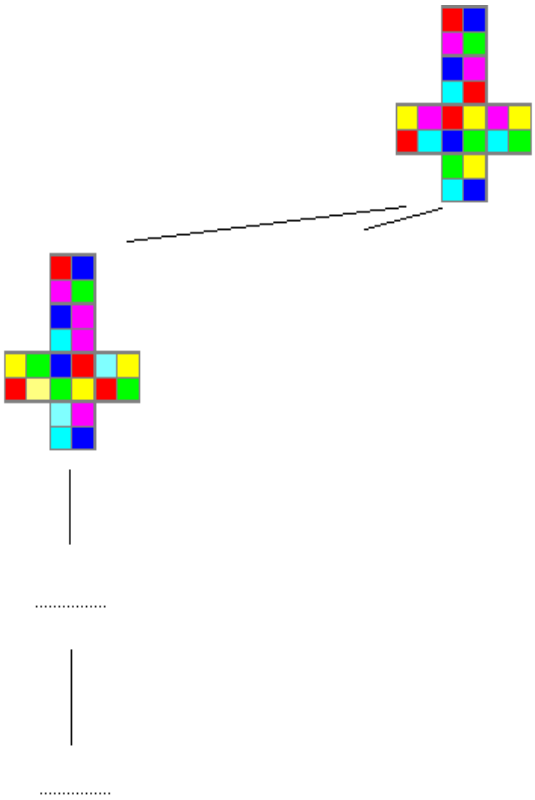
- F : Sisi yang menghadap pemain
- B : Sisi yang bertolak belakang dengan F
- U : Sisi pada sebelah atas dari F
- D : Sisi yang bertolak belakang dengan U, berada pada sebelah bawah dari F
- L : Sisi pada sebelah kiri F
- R : Sisi pada sebelah kanan F, bertolak belakang dengan L



Gambar 4. Notasi tiap-tiap sisi

2.2 Algoritma

Karena *puzzle* ini selalu memiliki solusi, maka pencarian akan dilakukan selama solusi belum ada. Untuk mencegah terciptanya pohon pencarian yang terlalu dalam, maka pencarian dibatasi sampai kedalaman 35 langkah.



Gambar 5. Pohon Pencarian sisi

Program akan melakukan aksi memutar salah satu sisi dari kubus sebesar 90 derajat. Selama belum ditemukan solusi dan hasil putar belum ada di pohon, maka hasil putar akan dimasukkan ke pohon dan program akan memutar sisi itu lagi. Bila hasil putar sudah ada di pohon pencarian, maka program akan melakukan *backtrack* dan kembali ke posisi sebelumnya, kemudian memutar sisi yang lain dengan pola yang sama. Sisi-sisi yang ada(F,B,U,D,L,R) akan diputar secara bergiliran sampai semua putaran pada satu sudah dilakukan atau solusi sudah ditemukan.

IV. KESIMPULAN

Mengingat banyaknya jumlah balok yang harus diperiksa(8 untuk *pocket cube*, 27 untuk *rubik's cube*) setiap kali melakukan putaran, serta banyaknya data yang harus disimpan pada pohon pencarian setalahnya, maka

penggunaan algoritma runut balik akan berat untuk menemukan solusinya. Pendekatan yang mungkin dilakukan bisa dengan memanfaatkan semacam AI untuk memilih sisi mana yang diprioritaskan untuk diputar pada posisi tertentu, dan bukannya mencoba semua kemungkinan selagi masih bisa. dengan mencari mana yang harus diprioritaskan, diharapkan tahap menuju solusi yang benar menjadi lebih dekat. Dengan itu, bisa jadi solusi akan diperoleh lebih cepat sehingga data yang harus diproses tidak sebanyak dengan memanfaatkan Algoritma Runut Balik murni.

REFERENSI

[1] Rinaldi Munir, “Diktat Kuliah IF2251 Strategi Algoritmik”, , 2005.