Penerapan Algoritma *Greedy* dan *Backtrackng* Dalam Penyelesaian Masalah Rubik's Cube

Amir Muntaha NIM: 13505041

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
e-mail: if15041@students.if.itb.ac.id

ABSTRAK

Sekarang ini permainan mekanik mungkin sudah tidak begitu populer lagi, hal ini dikarenakan berkembangnya teknologi elektronik yang membungkus permainan menjadi *video game*, bahkan anak sekarang lebih sering memainkan permainan tersebut ketimbang belajar untuk sekolah. Namun bagi sebagian orang, permainan mekanik adalah teman setia untuk membuang waktu ketika harus menunggu sesuatu, perjalanan panjang, maupun sebagai hiburan sebelum tidur. Bahkan sebagian ilmuwan menganggap bahwa permainan mekanik sangat cocok untuk meningkatkan kemampuan otak, terutama bagi anak kecil. Tidak heran orang tua kita lebih suka memberikan mainan mekanik kepada kita, dibandingkan membelikan mainan elektronik.

Walaupun perkembangan *video game* sangat subur, perkembangan teknologi juga melahirkan banyak jenis permainan mekanik. Contoh dari permainan tersebut adalah *Hanoi Tower*, 2-D *Puzzle*, 3-D *Puzzle*, *Yoyo*, *Gangsing*, *Rubik's Cube* (kadang disebut *Magic Cube*), *dan lain – lain.* Beberapa dari permainan ini memang sengaja dirancang untuk mengasah otak.

Salah satu permainan mekanik tersebut yaitu *Rubik's Cube* di dalam makalah ini akan dibahas cara penyelesaiannya.

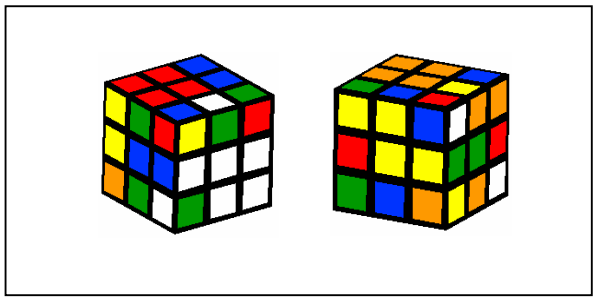
Kata kunci: *Rubik's Cube*, Algoritma *Greedy*, Algoritma *Backtracking*, kotak kecil, kubus kecil, searah jarum jam (SJ), berlawanan arah jarum jam (BJ).

1. PENDAHULUAN

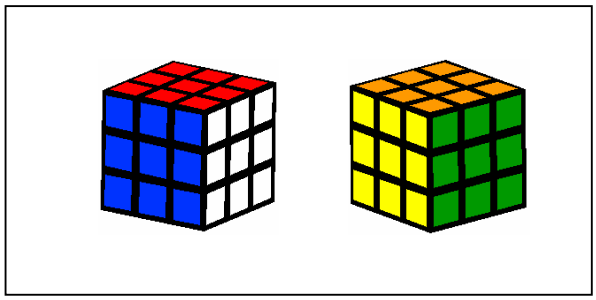
Rubik's Cube merupakan permainan mekanik yang terkenal sangat merumitkan apabila kita tidak mengetahui bagaimana cara menyelesaikannya, kita sering menganggap orang yang menyelesaikannya sebagai seorang jenius. Tapi dalam hal ini yang pantas disebut ssebagai jenius adalah Erno Rubik, seorang profesor desain interior berkewarganegaraan Hungaria yang menemukan kubus ini pada tahun 1974.

Rubik's Cube adalah permainan mekanik berupa kubus yang berukuran 3 x 3 x 3 kubus kecil. Tiap sisi kubus kecil yang mengahadap keluar memiliki warna yang berbeda. *Rubik's Cube* ini terdiri atas dua puluh enam kubus kecil yang disatukan oleh sebuah kubus kecil di pusat *Rubik's Cube*. Setiap sisi *Rubik's Cube* dapat diputar sebanyak 90^o, 180^o dan 270^o searah maupun berlawana arah jarum jam.

Pada awal permainan *Rubik's Cube* diputar secara acak sehingga tiap sisinya akan memiliki kotak – kotak sisi kubus kecil dengan warna yang berbeda (Lihat gambar 1.a). Kemudian mulai diselesaikan oleh pemainnya. Permainan selesai ketika setiap sisi *Rubik's Cube* memiliki kotak – kotak sisi kubus kecil dengan warna yang sama (Lihat gambar 1.b).



Gambar 1.a



Gambar 1.b

Kombinasi warna pada *Rubik's Cube* sangatlah banyak. Pertama (*kemungkinan ujung*), kombinasi setiap kotak kecil yang berada di ujung, terdapat kemungkinan 8 tempat yang berbeda dan pada setiap tempat itu satu kotak kecil memiliki 3 warna yang berbeda. Kedua (*kemungkinan rusuk*), kotak kecil yang berada di rusuk, terdapat 12 tempat yang berbeda dan pada setiap tempat itu terdapat 2 warna yang berbeda. Sedangkan (*kemungkinan tengah*) kotak kecil di tengah tiap – tiap sisi tidak akan berubah tempat, hanya ada satu kemungkinan saja. Jadi, banyaknya kombinasi warna yang ada adalah :

Kombinasi = *kemungkinan ujung* x *kemungkinan rusuk* *x kemungkinan tengah*

$= (8! \times 3^8) \times (12! \times 2^{12}) \times 1$

$= 519.024.039.293.878.272.000$

$\approx 5{,}19 \times 10^{20}$

Karena banyaknya kombinasi warna yang ada, padahal hanya ada satu kombinasi yang benar, maka banyak orang yang menyebut *Rubik's Cube* sebagai *Magic Cube*. Mereka berpikir bahwa kubus ini hanya bisa diselesaikan dengan sulap atau *magic*.

Untuk menyelesaikan *Rubik's Cube* dengan menggunakan algoritma *exhaustive search* sangatlah tidak effisien. Meskipun begitu masih banyak algoritma yang dipakai untuk menyelesaikannya. Dan makalah ini akan membahas cara penyelesaian *Rubik's Cube* dengan menggunakan algoritma *Greedy* pada langkah awal dan algoritma *backtracking* pada langkah seterusnya.

1.2. Dasar Teori

Berikut dasar teori dari tiap – tiap algoritma yang akan dipakai untuk menyelesaikan permasalahan *Rubik's Cube.*

1.2.1. Algoritma *Greedy*

Kita dapat mendefinisikan algoritma *greedy* sebagai algoritma yang memecahkan masalah langkah per langkah, dimana pada setiap langkahnya:

1. Mengambil pilihan terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (prinsipnya "*take what you can get now!*")
2. Berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

Persoalan optimasi dalam algoritma *greedy* disusun oleh elemen – elemen berikut:

1. Himpunan kandidat, C. himpunan ini berisi elemen – elemen pembentuk solusi. Pada setiap langkah, satu buah kandidat diambil dari himpunannya.
2. Himpunan solusi, S. Berisi kandidat terpilih sebagai solusi persoalan.
3. Fungsi seleksi – dinyatakan dengan predikat SELEKSI – yaitu fungsi pada setiap langkah memilih kandidat yang paling memungkinkan mencapai solusi optimal. Biasanya setiap kandidat, x, di-*assign* sebuiah nilai numerik, dan fungsi seleksi memilih x yang mempunyai nilai terbesar atau terkecil
4. Fungsi kelayakan – dinyatakan dengan predikat LAYAK – yang memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama – sama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala yang ada. Kandidat yang layak dimasukkan ke dalam himpunan solusi, sedangkan kandidat yang tidak layak dibuang dan tidak pernah dipertimbangkan lagi.
5. Fungsi obyektif, yaitu fungsi yang memaksimumkan atau meminimumkan nilai solusi.

Jadi, optimasi yang diselesaikan dengan algoritma *greedy* melibatkan pencarian sebuah himpunan bagian, S, dari himpunan kandidat, C; yang dalam hal ini, S harus memenuhi kriteria yang ditentukan, yaitu menyatakan suatu solusi dan S dioptimasi oleh fungsi obyektif.

1.2.2. Algoritma *Backtracking*

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 metode runut-balik, kita tidak perlu memeriksa semua kemungkinan solusi yang ada. Hanya pencarian yang mengarah ke solusi saja yang selalu dipertimbangkan, sehingga waktu pencarian dapat dihemat.

Saat ini algoritma runut-balik banyak diterapkan untuk program *games* (seperti permainan *tic-tac-toe*, menemukan jalan keluar dalam sebuah labirin, catur, sudoku, dll) dan masalah-masalah pada bidang kecerdasan buatan (*artificial intelligence*).

Properti umum yang sering kita jumpai pada aplikasi menggunakan algoritma *backtracking* adalah sebagai berikut :

1. Solusi persoalan.
   - Solusi dinyatakan sebagai vektor dengan *n-tuple*: $X = (x_1, x_2, \ldots, x_n)$, $x_i \in S_i$.
   - Mungkin saja $S_1 = S_2 = \ldots = S_n$.

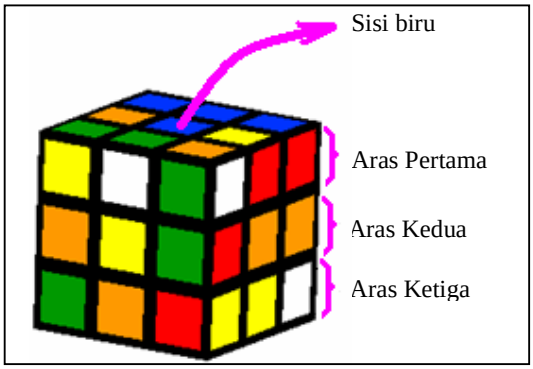
   Contoh: $S_i = \{0, 1\}$, $x_i = 0$ atau 1
2. Fungsi pembangkit nilai x_k
   Dinyatakan sebagai: $T(k)$
   $T(k)$ membangkitkan nilai untuk x_k, yang merupakan komponen vektor solusi.
3. Fungsi pembatas (pada beberapa persoalan fungsi ini dinamakan fungsi kriteria)
   - Dinyatakan sebagai : $B(x_1, x_2, \ldots, x_k)$
   - B bernilai *true* jika $(x_1, x_2, \ldots, x_k)$ mengarah ke solusi. Jika *true*, maka pembangkitan nilai untuk x_{k+1} dilanjutkan, tetapi jika *false*, maka $(x_1, x_2, \ldots, x_k)$ dibuang dan tidak dipertimbangkan lagi dalam pencarian solusi.

2. METODE

Menyelesaikan *Rubik's Cube* bukanlah hal yang mudah, oleh karena itu diperlukan beberapa algoritma untuk menyelesaikan permasalahan ini. Seperti yang telah disebutkan sebelumnya, *Rubik's Cube* akan diselesaikan dengan algoritma *brute force* dan *backtracking*. Berikut adalah langkah – langkah penyelesaian *Rubik's Cube :*

1. Langkah pertama, memilih sisi yang akan diselesaikan pertama, dan menyelesaikan aras pertama atau teratas, yaitu sisi dari *Rubik's Cube* yang dipilih pertama kali dan nantinya akan menjadi acuan untuk langkah – langkah selanjutnya.
2. Langkah kedua, menyelesaikan aras kedua atau tengah, yaitu aras di bawah aras pertama.
3. Langkah ketiga, menyelesaikan aras ketiga atau terbawah, yaitu sisi yang berlawanan dengan aras pertama.

Gambar 2.a di bawah ini keterangan dari aras – aras dari *Rubik's Cube*. Misalkan kita memilih sisi biru sebagai aras pertama.



Gambar 2.a

2.1 Langkah Pertama

Langkah pertama dari penyelesaian dari *Rubik's Cube* adalah menentukan sisi pertama yang akan diselesaikan. Sebenarnya kita bisa memilih sisi manapun, akan tetapi tidak semua sisi bisa diselesaikan dengan cepat, jadi kita akan menggunakan algoritma *brute force* untuk mencari sisi pertama dan menyelesaikannya.

Lingkup yang akan dievaluasi pertama kali adalah setiap sisi dari *Rubik's Cube*. Sisi yang akan dipilih adalah sisi yang diharapkan akan memberikan langkah tersedikit untuk menyelesaikannya.

Pertama, pada setiap sisinya, hitung kubus kecil yang mempunyai warna yang sama dan posisi yang sesuai yang terbesar. Kemudian, ambil sisi dengan hitungan yang terbanyak untuk diselesaikan.

Kedua, jika ada lebih dari satu sisi yang diambil dari hitungan pertama, maka dari sisi – sisi tersebut hitung kubus kecil yang sama warnanya. Kemudian, dari sisi – sisi tersebut, ambil sisi dengan hitungan terendah untuk diselesaikan.

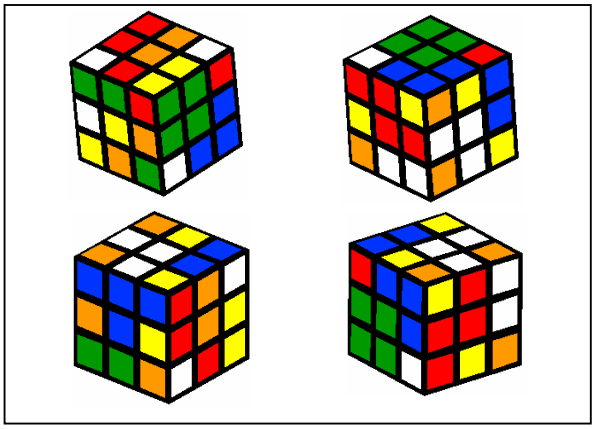
Ketiga, jika masih ada lebih dari satu sisi yang diambil dari hitungan kedua, maka dari sisi – sisi tersebut hitung jumlah hitung kubus kecil yang bewarna sama di sisi yang berlawanan dari sisi tersebut. Kemudian dari sisi – sisi tersebut, ambil sisi dengan hitungan terendah untuk diselesaikan.

Jika masih ada lebih dari satu sisi yang lolos dari hitungan ketiga, maka sisi – sisi tersebut akan dipilih salah satu untuk diselesaikan.

Berikut adalah contoh pemilihan sisi tersebut :

Misalkan H_{n_s} adalah hasil hitungan ke-n pada sisi dengan warna s.

Gambar 2.1.a dibawah adalah contoh *Rubik's Cube* yang akan dicari sisi yang akan diselesaikan pertama kali, dengan empat sudut pandang yang berbeda.



Gambar 2.1.a

Langkah pertama didapat :

$H1_{biru} = 2$ $H1_{hijau} = 3$ $H1_{putih} = 3$

$H1_{merah} = 2$ $H1_{jingga} = 2$ $H1_{kuning} = 2$

Jadi kita akan mengambil sisi hijau dan putih. Kemudian langkah kedua didapat :

$H1_{hijau} = 4$ $H1_{putih} = 3$

jadi kita akan menyelesaikan sisi putih terlebih dahulu.

Setelah dapat sisi yang akan diselesaikan pertama kali kita akan menggunakan algoritma *brute force* untuk menyelesaikan sisi tersebut.

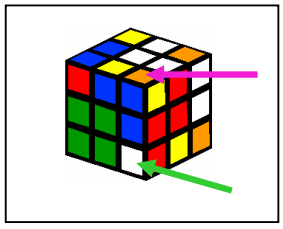
Berikut adalah elemen – elemen nya :

1. Himpunan kandidat (C) : kotak – kotak kecil yang akan ditempatkan di sisi yang akan diselesaikan pertama kali.
2. Himpunan solusi (S) : sisi yang diselesaikan pertama kali.
3. Fungsi seleksi : pertama, fungsi ini akan mencari kotak kecil dari himpunan C yang berada di sisi samping (sisi atas adalah sisi yang akan diselesaikan pertama kali, jika melihat gambar 2.1.a dengan sisi putih sebagai sisi atas, maka sisi biru, jingga, merah dan hijau adalah sisi samping), kemudian pilih yang berada di aras paling bawah terlebih dahulu yang akan diselesaikan jika sudah tidak ada lagi baru pada aras tengah, jika sudah tidak ada lagi baru aras atas. Dan jika semua kotak kecil pada himpunan C pada sisi samping sudah tidak ada, baru selesaikan sisi bagian bawah. Dalam setiap langkah tersebut, kita akan mendahulukan kotak kecil yang berada di sudut kubus.
4. Fungsi layak : tidak ada (karena semua himpunan C harus diselesaikan)
5. Fungsi obyektif : sisi yang dipilih pertama kali terselesaikan.

Berikut adalah contoh penyelesaian sisi tersebut :

Kita akan mengambil contoh yang kasus yang sama dengan gambar 2.1.a. Kita akan menyelesaikan sisi putih terlebih dahulu. Berikut adalah langkah – langkahnya (kotak kecil putih yang ditandai dengan panah berwarna hijau akan ditempatkan ke kotak yang ditandai dengan panah berwarna merah muda) :

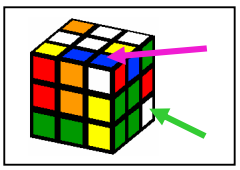
1.



gambar 2.1.1

Pada langkah pertama yang dilakukan dengan posisi seperti itu adalah memutar 90^0 sisi hijau SJ, kemudian memutar 90^0 sisi bawah SJ, kemudian memutar 90^0 sisi hijau BJ. Kemudian akan menjadi seperti gambar 2.1.2, yang akan menunjukkan langkah kedua.

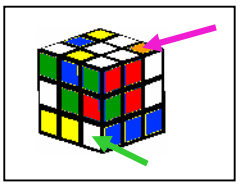
2.



gambar 2.1.2

Pada langkah kedua yang dilakukan adalah memutar 90^0 sisi bawah BJ, kemudian memutar 90^0 sisi jingga SJ, kemudian memutar 90^0 sisi bawah SJ, kemudian memutar 90^0 sisi jingga BJ. Kemudian akan menjadi seperti gambar 2.1.3, yang akan menunjukkan langkah ketiga.

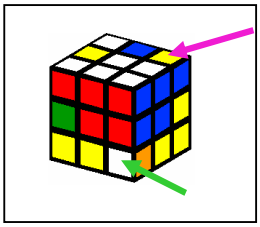
3.



gambar 2.1.3

Pada langkah ketiga yang dilakukan adalah memutar 90^0 sisi yang berlawanan dengan sisi hijau yaitu sisi biru BJ, kemudian memutar 90^0 sisi bawah SJ, kemudian memutar 90^0 sisi biru SJ. Kemudian akan menjadi seperti gambar 2.1.4, yang akan menunjukkan langkah ketiga.

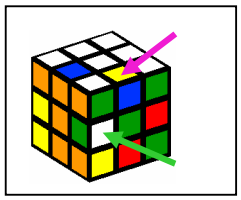
4.



gambar 2.1.4

Pada langkah keempat yang dilakukan adalah memutar 90^0 sisi yang berlawanan dengan sisi merah yaitu sisi jingga BJ, kemudian memutar 90^0 sisi bawah SJ, kemudian memutar 90^0 sisi jingga SJ. Kemudian akan menjadi seperti gambar 2.1.5, yang akan menunjukkan langkah ketiga.

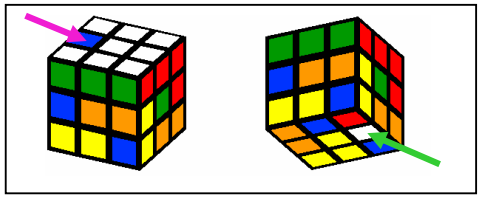
5.



gambar 2.1.5

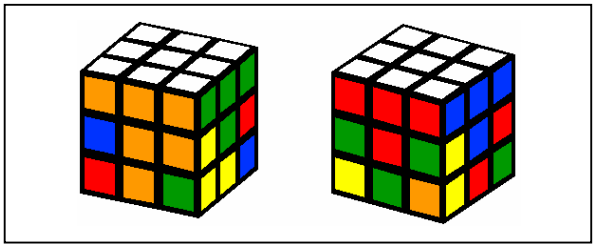
Pada langkah kelima yang dilakukan adalah memutar 180^0 sisi putih SJ, kemudian memutar 90^0 sisi yang berlawanan dengan sisi hijau yaitu sisi biru SJ, kemudian memutar 90^0 sisi bawah SJ, kemudian memutar 90^0 sisi putih BJ, kemudian memutar 90^0 sisi jingga BJ. Kemudian akan menjadi seperti gambar 2.1.6, yang akan menunjukkan langkah ketiga.

6.



gambar 2.1.6

Pada langkah keenam yang dilakukan adalah memutar 90^0 sisi putih BJ, kemudian memutar 90^0 sisi kuning/ bawah BJ, kemudian memutar 90^0 sisi hijau SJ, kemudian memutar 90^0 sisi yang berlawanan dengan sisi hijau yaitu sisi biru BJ, kemudian memutar 180^0 sisi jingga SJ, kemudian memutar 90^0 sisi hijau BJ, kemudian memutar 90^0 sisi biru SJ. Kemudian sisi putih akan menjadi satu kesatuan yang sesuai, seperti gambar 2.1.7. Dan langkah pertama untuk menyelesaikan *Rubik's Cube* selesai.



Gambar 2.1.7

2.2 Langkah Kedua

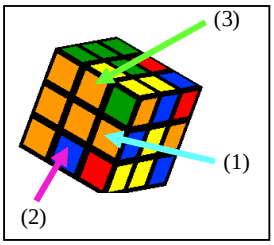
Langkah kedua dari penyelesaian dari *Rubik's Cube* adalah menyelesaikan aras kedua. kita akan menggunakan algoritma *brute force* untuk menyelesaikannya.

Berikut adalah elemen – elemen nya :

1. Himpunan kandidat (C) : kotak – kotak kecil yang akan ditempatkan di aras kedua.
2. Himpunan solusi (S) : aras kedua yang diselesaikan.
3. Fungsi seleksi : pertama, fungsi ini akan mencari kotak kecil dari himpunan C yang berada di pertama pilih yang berada di aras paling bawah terlebih dahulu untuk diselesaikan, jika sudah tidak ada lagi baru selesaikan yang di aras tengah.
4. Fungsi layak : tidak ada (karena semua himpunan C harus diselesaikan)
5. Fungsi obyektif : aras kedua/ tengah terselesaikan.

Cara menyelesaikan pada tiap pengambilan langkah adalah sebagai berikut :

1. Jika posisi kotak kecil di aras bawah maka putar aras bawah sampai kotak kecil tersebut ke sisi yang sama dengan warnanya tapi letaknya pasti belum sesuai. Pada contoh gambar 2.2.1, kotak tersebut ditandai dengan panah warna biru muda.
2. Langkah satu akan menjadikan kubus seperti gambar contoh dibawah ini :



gambar 2.2.1

Misalkan sisi yang diselesaikan pertama kali adalah sisi kiri (sisi putih pada gambar 2.2.1), sisi yang berlawanan adalah sisi kanan, sisi atas (sisi hijau pada gambar 2.2.1), sisi yang berlawanan adalah sisi bawah, sisi depan (sisi jingga pada gambar 2.2.1), dan sisi yang berlawanan adalah sisi belakang.

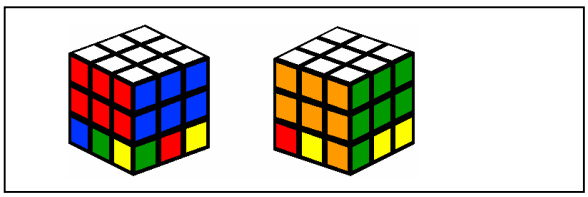
Jika (1) ingin dipindah ke (2) maka, langkah yang dilakukan adalah memutar 90^0 sisi kanan SJ, memutar 90^0 sisi bawah SJ, memutar 90^0 sisi kanan BJ, memutar 90^0 sisi bawah BJ, memutar 90^0 sisi depan SJ, memutar 90^0 sisi bawah BJ, memutar 90^0 sisi depan BJ, dan memutar 90^0 sisi bawah SJ.

Jika (1) ingin dipindah ke (3) maka, langkah yang dilakukan adalah memutar 90^0 sisi kanan BJ, memutar 90^0 sisi atas BJ, memutar 90^0 sisi kanan SJ, memutar 90^0 sisi atas SJ, memutar 90^0 sisi depan BJ, memutar 90^0 sisi atas SJ, memutar 90^0 sisi depan SJ, memutar 90^0 sisi atas BJ,

3. Jika posisi kotak kecil di aras tengah, maka lakukan langkah ke-2 di atas dengan mengambil (1) kotak

yang bukan merupakan anggota himpunan C, dan kotak (2) sebagai kotak yang akan diselesaikan. Dan masukkan (1) ke (2). kemudian kembali ke langkah ke-1.

Kemudian aras kedua akan selesai, seperti gambar 2.2.2.



Gambar 2.2.2

2.3 Langkah Ketiga

Langkah ketiga dari penyelesaian dari *Rubik's Cube* adalah menyelesaikan aras terakhir akan digunakan algoritma *backtracking* untuk menyelesaikannya.

Pertama, adalah *backtracking* untuk membentuk empat tiang. Tiang adalah rusuk dari *Rubik's Cube* yang sudah sesuai yang merentang dari aras pertama sampai terakhir (pada gambar 2.2.2 rusuk kubus warna hijau dan jingga).

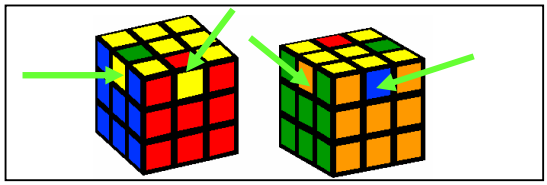
Di dalam pembentukan tiang ini terdapat dua langkah untuk penyelesaiannya, yaitu :

Terlebih dahulu kita harus mempunyai satu tiang sebagai acuan, sisi yang pertama diselesaikan sebagai sisi bawah, dan tiang acuan berada pada pertemuan sisi depan dan kanan.

1. Langkah A : memutar 90^0 sisi kanan SJ, memutar 90^0 sisi atas SJ, memutar 90^0 sisi kanan BJ, memutar 90^0 sisi atas BJ, memutar 90^0 sisi kanan BJ, memutar sisi depan SJ, memutar 90^0 sisi kanan SJ, memutar sisi depan BJ.
2. Langkah B : memutar 90^0 sisi depan BJ, memutar 90^0 sisi atas BJ, memutar 90^0 sisi depan SJ, memutar 90^0 sisi atas SJ, memutar 90^0 sisi depan SJ, memutar 90^0 sisi kanan BJ, memutar 90^0 sisi depan BJ, memutar 90^0 sisi kanan SJ.

Untuk memulai *backtracking* kita harus menentukan tiang acuan terlebih dahulu, caranya adalah memutar sisi atas (aras terakhir) sampai menemukan tiang. Jika setelah satu putaran belum ditemukan tiang, maka lakukan langkah A dengan menggunakan tiang bayangan. Setelah didapat tiang acuan, maka kondisi tersebut akan menjadi akar dari pohon *backtracking* tersebut. Anak dari pohon *backtracking* tersebut dibangkitkan dengan langkah A dan B. Pohon akan melakukan *backtrack* jika simpul yang dibangkitkan sudah pernah dibangkitkan sebelumnya. Proses tersebut akan selesai apabila keempat tiang telah terbentuk.

Setelah keempat tiang terbentuk seperti pada gambar 2.3.1 dibawah, maka akan dilanjutkan *backtracking* untuk kubus kecil yang belum selesai (ditunjuk panah hijau).



Gambar 2.3.1

Di dalam pembentukan empat kubus kecil ini terdapat dua langkah untuk penyelesaiannya, yaitu :

Terlebih dahulu kita harus mempunyai satu sisi lagi yang sudah selesai sebagai acuan, sisi yang pertama diselesaikan sebagai sisi bawah, dan sisi acuan sisi kiri.

1. Langkah α : memutar 90^0 sisi kanan SJ, memutar 90^0 sisi kiri BJ, memutar 90^0 sisi atas SJ, memutar 90^0 sisi kanan BJ, memutar 90^0 sisi atas BJ, memutar 90^0 sisi kiri SJ, memutar 90^0 sisi kanan BJ, memutar sisi depan SJ, memutar 90^0 sisi kanan SJ, memutar sisi depan BJ.
2. Langkah β : memutar 90^0 sisi kanan BJ, memutar 90^0 sisi kiri SJ, memutar 90^0 sisi atas BJ, memutar 90^0 sisi kanan SJ, memutar 90^0 sisi atas SJ, memutar 90^0 sisi kiri BJ, memutar 90^0 sisi kanan SJ, memutar 90^0 sisi belakang BJ, memutar 90^0 sisi kanan BJ, memutar 90^0 sisi belakang SJ.

Untuk memulai *backtracking* kita harus mempunyai sisi acuan, jika belum ada, maka lakukan langkah α tanpa sisi acuan. Setelah didapat sisi acuan, maka kondisi tersebut akan menjadi akar dari pohon *backtracking* tersebut. Anak dari pohon *backtracking* tersebut dibangkitkan dengan langkah A dan B. Pohon akan melakukan *backtrack* jika simpul yang dibangkitkan sudah pernah dibangkitkan sebelumnya. Proses tersebut akan selesai apabila *Rubik's Cube* telah selesai terbentuk (Lihat gambar 1.b).

3. KESIMPULAN

Walaupun terlihat sangat sulit, ternyata *Rubik's Cube* dapat diselesaikan dengan menggunakan langkah – langkah diatas. Tapi masih banyak lagi metode untuk menyelesaikannya, setiap metode memiliki keunggulan tersendiri. Keunggulan metode ini kita bisa menerapkan algoritma ini tanpa menggunakan komputer sekalipun. Jadi selamat mencoba permainan *Rubik's Cube.*

REFERENSI

[1] Getsel, Martin Van, "Cube", versi 1.42, Mach10 Software Production, 2005. (*Freeware program*)

[2] http://informatika.org/~rinaldi/ diakses tanggal 11 Mei 2007, jam 16.30 WIB.

[3] Munir, Rinaldi, "Strategi Algoritmik", edisi 2007, Program Studi Teknik Informatika STEI ITB, 2007.