Aplikasi Matematika Diskrit dalam Penyelesaian Balok Rubik

Muhammad Fadhriga Bestari - 13516154
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
fadhriga.bestari@gmail.com

***Abstract*—Balok rubik merupakan permainan yang dapat dimainkan oleh masyarakat umum. Balok rubik memiliki premis yang sederhana, dimana pemain diminta untuk mengembalikan balok rubik ke kondisi awalnya, yaitu semua sisi hanya memiliki satu warna saja. Meski demikian, begitu sulitnya untuk seseorang agar dapat melakukan hal tersebut, hingga saat ini bahkan terdapat lomba – lomba yang bertujuan untuk mencari orang yang dapat menyelesaikan balok rubik paling cepat. Pada makalah ini, penulis berharap untuk dapat menyampaikan bagaimana matematika diskrit berpengaruh besar pada proses penyelesaian balok rubik.**

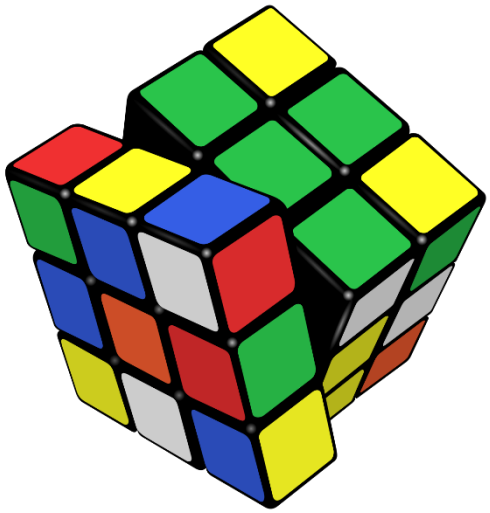
***Keywords*— Rubik, Matematika Diskrit, Balok**

I. PENDAHULUAN

Balok rubik merupakan permainan sederhana yang dapat dimainkan oleh semua orang. Permainan yang memiliki tujuan untuk mengembalikan keadaan balok ke keadaan semula memang terdengar mudah. Tingkat kesederhanaan permainan inilah yang membuat orang – orang mau memainkan balok rubik. Terkadang, bahkan orang tua membujuk anak – anak mereka untuk bermain balok rubik, dengan harapan balok rubik tersebut dapat meningkatkan kecerdasan anak – anak mereka.

Meski memiliki premis yang sederhana, tingkat kesulitan untuk menyelesaikan balok rubik menyebabkan banyaknya situs – situs di internet yang memberikan pengajaran cara menyelesaikan balok rubik tersebut. Kini, sudah banyak langkah pasti untuk menyelesaikan balok rubik, apapun kondisinya. Oleh karena itu, banyak orang yang kini memainkan rubik balok hanya dengan tujuan untuk memamerkan kemampuan mereka mengingat langkah langkah untuk menyelesaikan sebuah balok rubik.

Pada makalah kali ini, penulis ingin mengangkat sebuah topik pada balok rubik yang menarik. Tentunya, kita semua tau bahwa balok rubik merupakan sebuah balok yang kita dapat putar – putar untuk merubah status warna – warna dari balok tersebut. Meski memang benar, terdapat langkah pasti yang telah terbukti dapat menyelesaikan semua balok rubik dalam kondisi apapun, akan lebih mudah jika kita melakukan langkah mundur dari langkah – langkah kita sebelumnya. Dengan pola pikir seperti ini, semua kondisi pada balok rubik hanyalah sebuah keadaan yang diciptakan dari balok rubik yang hanya memiliki satu warna saja. Dengan menggunakan bantuan matematika diskrit, mari kita telusuri bagaimana cara menyelesaikan balok rubik dari segi matematikanya.



Gambar 12.
Sumber : www.wikipedia.com

II. LANDASAN TEORI

2.1. Graf

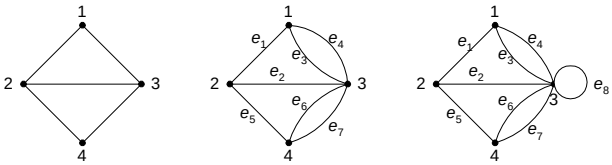
Graf merupakan sebuah pemodelan dari himpunan sebuah simpul yang memiliki suatu hubungan dengan simpul – simpul lainnya yang direpresentasikan dengan sebuah titik yang terhubung dengan titik lain dengan sebuah garis. Hal yang harus diperhatikan, simpul pada sebuah graf tidaklah kosong, dengan hubungan antar simpul yang mungkin kosong. Hal ini dapat disimbolkan dengan

Graf G = (V, E), dimana :

V = himpunan tidak-kosong dari simpul-simpul (vertices)
= { v1 , v2 , ... , vn }

E = himpunan sisi (edges) yang menghubungkan sepasang simpul
= {e1 , e2 , ... , en }

2.1.1. Jenis Graf



(a) (b) (c)

Gambar 1.

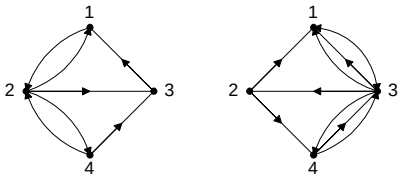
Berdasarkan cara simpul – simpul pada graf saling terhubung dan ada tidaknya gelang atau sisi ganda pada suatu graf, dapat disimpulkan bahwa terdapat dua jenis graf, yaitu :

1. Graf Sederhana
   Graf sederhana merupakan graf yang tidak memiliki sisi ganda maupun gelang. Graf sederhana direpresentasikan pada Gambar 1 (a).

2. Graf Tak Sederhana
   Graf tak sederhana merupakan graf yang memiliki sisi ganda atau gelang. Graf tak sederhana direpresentasikan pada Gambar 1 (b) dan (c).

Jika graf dikelompokkan berdasarkan pemodelan orientasi arahnya, dapat disimpulkan terdapat dua jenis, yaitu :

1. Graf Tak Berarah
   Graf tak berarah adalah graf yang tidak memiliki orientasi arah, maka, graf tak berarah dapat ditelusuri dengan bebas dan pada proses penelusuran dapat dilakukan gerak mundur. Graf tak berarah direpresentasikan pada Gambar 1.
2. Graf Berarah
   Graf berarah adalah graf yang memiliki orientasi arah pada setiap sisinya. Hal ini menyebabkan proses penelusuran pada graf hanya dapat dilakukan sesuai dengan arah yang tersedia. Graf berarah direpresentasikan pada Gambar 2.



Gambar 2.

2.1.2. Terminologi Graf

Beberapa di bawah ini adalah terminologi yang digunakan untuk merepresentasikan sebuah graf, yaitu :

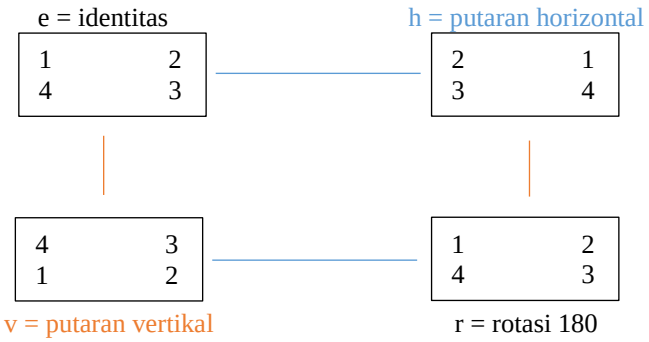
1. Ketetanggaan
   Dua buah simpul dapat dikatakan bertetangga saat kedua simpul tersebut berhubungan langsung dengan sebuah sisi.
2. Bersisian
   Untuk sebuah sisi sembarang, e = (vj, vk) dikatakan bahwa :
   e bersisian dengan simpul vj, atau
   e bersisian dengan simpul vk.
3. Simpul Terkecil
   Simpul terkecil adalah sebuah simpul yang tidak memiliki sisi yang bersisian dengan simpul lainnya.
4. Graf Kosong
   Graf kosong adalah saat dimana graf hanya memiliki simpul yang tidak bersisian dengan siapapun. Antara lain, graf kosong adalah graf yang terdiri atas simpul terkecil, atau himpunan sisinya kosong.
5. Derajat
   Dejarat adalah jumlah sisi yang bersisian dengan simpul tersebut. Dengan menggunakan **Lemma Jabat Tangan**, dapat dibuktikan bahwa untuk semua graf, jumlah derajat yang ada akan selalu genap, yaitu dua kali jumlah sisi pada graf tersbut.
6. Lintasan
   Jalur yang dapat ditempuh untuk menelusuri sebuah graf dari simpul vo ke simpul vn
7. Siklus
   Jalur yang dapat ditempuh untuk menelusuri sebuah graf yang berawal dan berakhir di simpul vo
8. Terhubung
   Saat dua simpul v1 dan v2 memiliki lintasan dari v1 ke v2. Perhatikan bahwa graf terhubung adalah saat dimana semua simpul vi dan vj memiliki lintasan antara vi dan vj.
9. Upagraf
   Upagraf dari G = (V,E) adalah G1 = (V1,E1), jika V1 $\subseteq$ V dan E1 $\subseteq$ E.
10. Upagraf Rentang
    Upagraf rentang dari G = (V,E) adalah G1 = (V1,E1), jika V1 = V, yaitu semua simpul G berada pada G1.

2.1.3. Cayley Graf

Cayley graph adalah sebuah cara untuk memecahkan sebuah informasi dari sebuah bagian dalam sebuah graf. Cayley graf memiliki simpul – simpul terhubung yang berarah, dimana :

1. Setiap panah memiliki warna yang merepresentasikan suatu aksi.
2. Setiap aksi yang terjadi direpresentasikan oleh simpul yang berbeda.
3. Setiap aksi direpresentasikan oleh jalur yang berasal dari *solved state*

Aplikasi sederhana dari penggunaan cayley graf adalah dengan menggunakan cayley graf dalam pemodelan perubahan keadaan. Jika kita bayangkan ada sebuah balok dengan empat kotak yang memiliki angka. Jika pada kotak tersebut kita diperbolehkan hanya dua langkah saja, yaitu memutar balok secara horizontal dan memutar secara vertikal, akan terdapat empat konfigurasi yang dapat tercipta, yaitu sebagai berikut :



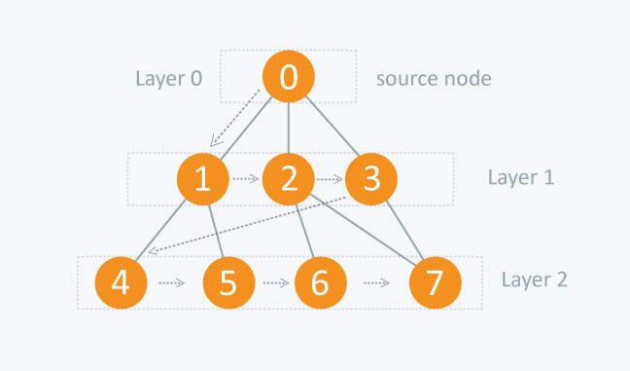
Gambar 3.
Sumber : http://www.math.clemson.edu/

Jika G melambangkan grup diatas, maka dapat disimpulkan :

1. G memiliki empat simpul yang tercipta setelah memberi aksi – aksi tertentu
2. G membutuhkan dua aksi yang berbeda untuk diciptakan, dengan r = hv = vh

2.2. Breadth – First Search

Breadth – first search merupakan salah satu algoritma yang dapat digunakan untuk mencari sebuah keadaan pada sebuah graf atau pohon yang ada. *Breadth – first search* melakukan pencarian secara transversal pada graf, yang seperti namanya, melakukannya dengan cara memeriksa simpul yang bertetanggaan (memiliki tingkat yang sama) lalu menelusuri graf.



Gambar 4.
Sumber : https://www.hackerearth.com

Gambar 5.
Sumber : https://www.hackerearth.com

Dalam sebuah graf, dimungkinkan untuk adanya sebuah simpul ganda ataupun gelang. Hal ini akan menyebabkan simpul memungkinkan untuk ditelusuri lebih dari satu kali. Maka dari itu, untuk menghindari hal ini, diperlukannya simpul yang telah ditelusuri untuk disimpan di suatu tempat. Hal yang dapat memudahkan penelusuran adalah dengan menyimpan data simpul dalam sebuah *queue*, dengan boolean yang menandakan simpul – simpul tersebut telah ditelusuri. Berikut adalah *pseudocode* untuk *Bread – first search* :

```
BFS (G, s)                  //Where G is the graph and s is the source node
      let Q be queue.
      Q.enqueue( s ) //Inserting s in queue until all its neighbour vertices are marked.

      mark s as visited.
      while ( Q is not empty)
           //Removing that vertex from queue,whose neighbour will be visited now
           v  =  Q.dequeue( )

          //processing all the neighbours of v
          for all neighbours w of v in Graph G
               if w is not visited
                         Q.enqueue( w )              //Stores w in Q to further visit its neighbour
                         mark w as visited.
```

Gambar 6.
Sumber : https://www.hackerearth.com

Setelah algoritma pada Gambar 6 diterapkan, pencarian transversal pada Gambar 5 akan berjalan sebagai berikut :

1. Iterasi pertama
   S akan dibuang dari *queue*
   Tetangga dari S, yaitu 1 dan 2 akan ditelusuri berikutnya
   Karena 1 dan 2 belum pernah ditelusuri sebelumnya, maka 1 dan 2 akan dimasukkan ke dalam *queue*, lalu 1 dan 2 ditandai agar tidak ditelusuri lagi.
2. Iterasi kedua
   1 akan dibuang dari *queue*
   Tetangga dari 1, yaitu S dan 3 akan ditelusuri berikutnya
   Meskipun S adalah tetangga dari 1, S diabaikan karena S telah ditelusuri sebelumnya.
   Karena 3 belum pernah ditelusuri sebelumnya, maka 3 akan dimasukkan ke dalam *queue*, lalu 3 ditandai agar tidak ditelusuri lagi.
3. Iterasi ketiga
   2 akan dibuang dari *queue*
   Tetangga dari 2, yaitu S, 3, dan 4 akan ditelusuri berikutnya
   Meskipun S dan 3 adalah tetangga dari 2, S dan 3 diabaikan karena telah ditelusuri sebelumnya.
   Karena 4 belum pernah ditelusuri sebelumnya, maka 4 akan dimasukkan ke dalam *queue*, lalu 4 ditandai agar tidak ditelusuri lagi.
4. Iterasi keempat
   3 akan dibuang dari *queue*
   Tetangga dari 3, yaitu 1, 2 dan 5 akan ditelusuri berikutnya.
   Meskipun 1 dan 2 adalah tetangga dari 3, 1 dan 2 diabaikan karena telah ditelusuri sebelumnya.
   Karena 5 belum pernah ditelusuri sebelumnya, maka 5 akan dimasukkan ke dalam *queue*, lalu 5 akan ditandai agar tidak ditelusuri lagi.
5. Iterasi kelima
   4 akan dibuang dari *queue*
   Semua tetangga 4 telah ditelusuri, maka semuanya diabaikan.
6. Iterasi keenam
   5 akan dibuang dari *queue*

Semua tetangga 5 telah ditelusuri, maka semuanya diabaikan.

2.3. Permutasi

Permutasi adalah jumlah urutan berbeda dari pengaturan objek – objek. Pada dasarnya, permutasi adalah bentuk khusus dari aplikasi kaidah perkalian. Jika dimisalkan ada n buah objek, maka :

1. Urutan pertama dapat dipilih dari n buah objek
2. Urutan kedua dapat dipilih dari n-1 buah objek
3. Urutan ketiga dapat dipilih dari n-2 buah objek
4. Urutan terakhir dapat dipilih dari 1 objek yang tersisa

Menurut kaidah perkalian, permutasi dari n objek adalah :

$$n(n - 1)(n – 2) \ldots (2) (1) = n!$$

2.4. Kompleksitas Algoritma

Kompleksitas algoritma adalah salah satu cara untuk mengukur tingkat keefisienan dari sebuah algoritma yang ada. Kompleksitas algoritma pada umumnya dibagi menjadi 2, yaitu :

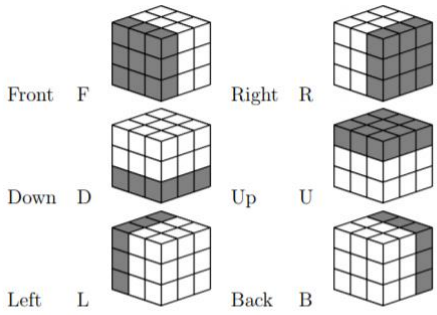
1. Kompleksitas Waktu
   Kompleksitas waktu, T(n) adalah tingkat seberapa cepat algoritma sebagai fungsi dengan ukuran n untuk dikomputasi.
2. Kompleksitas Ruang
   Kompleksitas ruang, S(n), adalah tingkat penggunaan memori yang digunakan oleh struktur data dengan ukuran n.

III. APLIKASI MATEMATIKA DISKRIT DALAM PENYELESAIAN BALOK RUBIK

Permainan balok rubik merupakan salah satu permainan yang sangatlah bergantung terhadap matematika. Meski awalnya, permainan balok rubik mungkin dimainkan dengan asal – asalan, dengan hasil yang mungkin tidak memuaskan. Tetapi, dengan menggunakan bantuan matematika, penulis menemukan cara – cara yang dapat membantu seseorang untuk menyelesaikan permainan balok rubik.

3.1. Notasi Balok Rubik

Perlu diperhatikan bahwa pada permainan balok rubik, terdapat banyak cara untuk merubah keadaan *(state)* dari setiap balok rubik yang ada. Maka dari itu, berikut adalah notasi yang akan penulis gunakan untuk merepresentasikan sebuah aksi perubahan *state* dari sebuah balok rubik :

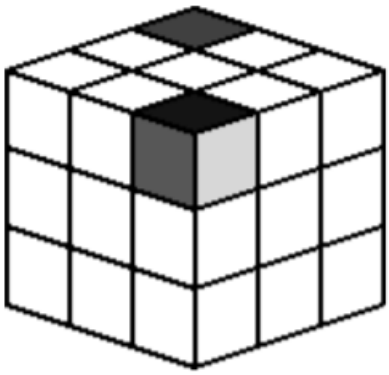


Gambar 7.

Sumber : http://web.mit.edu/

Pemutaran balok rubik akan direpresentasikan sebagai notasi – notasi diatas. Untuk setiap notasi F, perputaran yang dilakukan adalah perputaran 90 derajat searah jarum jam, dimana perputaran dilakukan pada barisan F diatas. Untuk setiap perputaran yang berlawanan dengan arah jarum jam, notasinya adalah F'.

3.2. Batas Penyelesaian Balok Rubik



Gambar 8.
Sumber : http://web.mit.edu/

Jumlah kemungkinan permutasi dari kotak – kotak balok rubik sangatlah besar. Terdapat 8 balok pojok yang dapat diatur sebanyak 8! cara, dengan setiap bagian pojok dari balok rubik memiliki 3 warna yang berbeda, maka terdapat 3^8 kemungkinan untuk permutasi dari bagian pojok balok rubik. Tidak lupa juga, terdapat 12 balok sisi yang setiap balok sisi tersebut memiliki 2 warna konfigurasi yang berbeda – beda, maka terdapat 12! dan 2^{12} permutasi kemungkinan balok sisi dari balok rubik yang ada. Meskipun begitu, dari segala kemungkinan balok rubik yang ada, hanya 1/12 dari kemungkinan tersebut yang dapat direalisasikan dari sebuah rubik balok, hal ini disebabkan setiap kemungkinan yang ada hanya dapat terbentuk jika dan hanya jika kondisi tersebut dapat diciptakan karena pergeseran balok rubik dari *solved state* (semua sisi hanya memiliki 1 warna saja). Maka dari itu, jumlah permutasi dari keadaan balok rubik yang mungkin ada adalah :

$$\frac{8! \cdot 3^8 \cdot 12! \cdot 2^{12}}{3 \cdot 2 \cdot 2} = 4.3252 \cdot 10^{19}$$

3.3. Permutasi dalam Balok Rubik

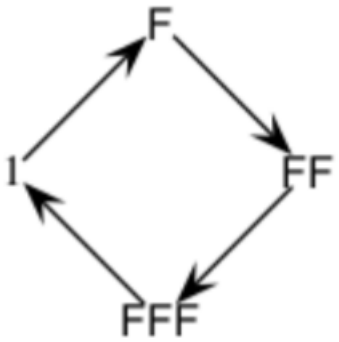
Setiap aksi yang dilakukan pada balok rubik akan menghasilkan *state* yang baru. Perhatikan bahwa setiap pergerakan yang mengembalikan balok ke keadaan asal dapat dikategorikan ke dalam grup yang sama. Karena setiap pengaturan merupakan salah satu kemungkinan dari permutasi yang ada, maka perubahan *state* pada balok rubik dapat direpresentasikan dengan permutasi. Sebagai contoh, pergerakan FFRR merupakan permutasi yang sama dari pergerakan berikut, (DF UF) (DR UR) (BR FR FL) (DBR UFR DFL) (ULF URB DRF).

Pengaplikasian lebih lanjut dari permutasi dalam balok rubik

adalah dengan menggunakan *canonical cycle notation* pada penulisan permutasi diatas. Karena semua pergeseran yang dilakukan merepresentasikan kondisi akhir yang sama, maka urutan pemberian pergeseran tidak diperhatikan.

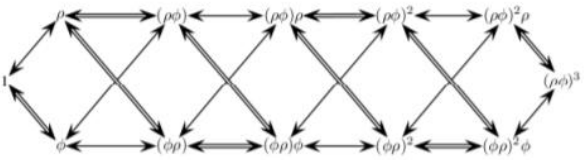
3.4. Cayley Graf dalam Balok Rubik

Balok rubik memiliki begitu banyaknya kemungkinan *state* yang ada. Cayley graf merupakan salah satu cara yang dapat dilakukan untuk merincikan sebuah bagian kecil dari permutasi keadaan balok rubik. Seperti misalnya, daripada menggambarkan $4.3252 \cdot 10^{19}$ kemungkinan balok rubik, cayley graf digunakan untuk memvisualisasikan graf pada bagian terkecilnya. Seperti misalnya, pada Gambar 9 dapat kita hasilkan sebuah cayley graf yang merepresentasikan *state* yang hanya dihasilkan oleh pergeseran F saja.



Gambar 9
Sumber : http://web.mit.edu/

Ada pula cayley graf yang dapat dihasilkan dari pergerakan FF dan RR saja, jika φ = FF dan ρ = RR, maka Gambar 10 adalah hasil dari cayley graf tersebut.



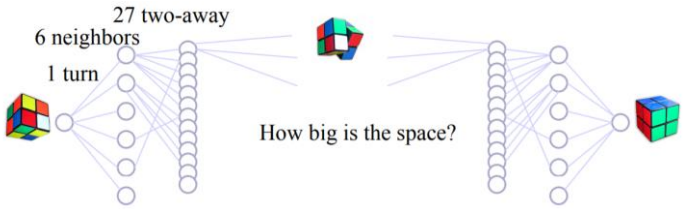
Gambar 10.
Sumber : http://web.mit.edu/

Cayley graf sangatlah berguna pada proses penyelesaian balok rubik karena pemain dapat menganalisis secara rinci hasil dari aksi – aksi yang dilakukan, tanpa membentuk graf yang memiliki miliaran cabang. Hal yang harus diperhatikan, bahwa cayley graf untuk D, dan UULL dapat pula direpresentasikan dengan Gambar 9 dan Gambar 10, hal ini dikarekanakan kedua cayley graf yang akan tercipta nantinya akan isomorfik dengan Gambar 9 dan Gambar 10.

3.5. Menyelesaikan Balok Rubik dengan Graf

Seperti yang telah penulis utarakan sebelumnya, terdapat miliaran kemungkinan keadaan dari balok rubik dengan balok 3 x 3. Meskipun begitu, semua keadaan tersebut hanya akan muncul dari kumpulan pergeseran yang dilakukan terhadap balok rubik yang berada pada *solved state*. Maka dari itu, sebenarnya seluruh balok rubik dapat diselesaikan hanya dengan melakukan semua langkah yang telah dijalansi secara mundur. Hal ini dapat divisualisasikan dengan lebih jelas dengan menggambarkan semua pergerakan yang dilakukan sebagai graf 2 arah, lalu melakukan *backtrack* ke *starting state*.

Melakukan *backtrack* saja memang bukanlah hal yang sulit, dan memang, siapapun dapat melakukan hal yang sangat sederhana tersebut. Di sini penulis hanya ingin mengangkat pemikiran dimana setiap keadaan dari balok rubik, seperti halnya melakukan *backtrack*, dapat direpresentasikan sebagai simpul dalam sebuah graf, dengan tetangga dari simpul tersebut adalah semua kemungkinan yang dapat dihasilkan dari 1 pergerakan bagian dari balok rubik. Miniatur dari penggambaran graf balok rubik tersebut adalah sebagai berikut:



Gambar 11.
Sumber : http://www.facweb.iitkgp.ernet.in/

Pada Gambar 11, dapat diperhatikan bahwa apapun keadaan sebuah balok rubik, pasti terdapat suatu lintasan yang akan mencapai simpul akhir, yaitu simpul *final state*. Perlu diperhatikan pula, bahwa jalan menuju *final state* tidak hanya satu. Hal ini menyebabkan cara penyelesaian yang banyak, dengan *canonical cycle notation* dengan panjang yang bervariasi. Permasalahan yang muncul pada proses penyelesaian balok rubik dengan cara ini adalah bagaimana besarnya graf yang akan tercipta. Karena tidak seperti cayley graf, disini graf yang dibentuk harus merepresentasikan semua kemungkinan yang ada.

Kemampuan manusia untuk memproses hal yang sebesar ini sangatlah terbatas. Waktu yang diperlukan untuk menggambar setiap kemungkinan yang ada pun tidaklah sedikit. Maka dari itu, digunakanlah bantuan komputer untuk menganalisa dan menelusuri graf balok rubik ini. *Breadth – first search* merupakan salah satu algoritma pencarian pada graf. Tidak hanya itu, BFS merupakan algoritma yang melakukan pencarian transversal dan mengeluarkan jawaban dengan melalui lintasan tercepat pada graf.

Meski dengan bantuan komputer, akan tetap diperlukan tingkat komputasi yang tidak rendah untuk menganalisis semua simpul yang ada (melalui proses pencarian transversal BFS). Penggunaan BFS pada proses pencarian menyebabkan setiap simpul dan sisi yang ada untuk diperiksa satu kali, dan jika n = jumlah simpul dan m = jumlah sisi, maka tingkat kompleksitas waktunya adalah O(n + m). Perhatikan bahwa jumlah simpul dan sisi dari graf balok rubik tidaklah sedikit, pencarian transversal yang sedikit rumit.

V. KONKLUSI

Matematika diskrit memiliki peran besar dalam proses penyelesaian balok rubik. Pengaplikasian materi graf, permutasi dan kompleksitas algoritma yang dibahas dalam makalah ini tentunya memiliki banyak ruang untuk perkembangan. Pada tahun 2010, Thomas Rokicki, Herbert Kociemba, Morley Davidson, dan John Dethridge, membuktikan bahwa *God's Number* (langkah terbesar yang dibutuhkan untuk menyelesaikan semua keadaan balok rubik) adalah 20. Penulis harap dengan makalah ini pembaca akan lebih mengerti akan bagaimana pola pikir dari penyelesaian masalah balok rubik ini.

VI. UCAPAN TERIMA KASIH

Penulis berterima kasih kepada Dr. Ir. Rinaldi Munir yang telah memberikan pembelajaran dan bimbingan dalam kuliah Matematika Diskrit. Penulis juga mengucapkan terima kasih kepada seluruh pihak yang telah memberikan dukungan baik secara fisik maupun moral sehingga penyusunan makalah ini dapat selesai tepat waktu.

REFERENCES

[1] Munir, R. Matematika Diskrit. Bandung: Informatika Bandung, 2005
[2] https://www.hackerearth.com/ diakses pada tanggal 02/12/2017
[3] https://courses.csail.mit.edu diakses pada tanggal 02/12/2017
[4] http://mathworld.wolfram.com/ diakses pada tanggal 02/12/2017
[5] http://web.mit.edu/ diakses pada tanggal 02/12/2017
[6] http://www.cube20.org/ diakses pada tanggal 02/12/2017
[7] http://www.math.clemson.edu/ diakses pada tanggal 02/12/2017

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 3 Desember 2017

Muhammad Fadhriga Bestari - 13516154