Penerapan Graf pada Permainan Kubus Rubik (*Rubik's Cube*)

Ahmad Naufal Hakim - 13517055
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517055@std.stei.itb.ac.id

***Abstrak*—*Rubik's Cube* adalah permainan *puzzle* berbentuk sebuah kubus dengan 6 sisi dan masing-masing sisi memiliki warna yang berbeda. Tujuan dari *puzzle* tersebut adalah untuk menyusun kembali warna-warna pada *Rubik's Cube* menjadi seperti semula. Hingga saat ini sudah banyak beredar petunjuk-petunjuk atau tutorial tentang menyelesaikannya, namun menurut penulis masih banyak orang yang tidak mengetahui alur berpikir yang tepat untuk menyelesaikan permainan ini. Maka dari itu, penulis ingin menunjukkan representasi graf pada permainan *Rubik's Cube* agar pembaca menjadi lebih paham terkait tahap-tahap penyelesaiannya.**

***Kata kunci*—Graf, *Rubik's Cube*, Fridrich, Metode CFOP.**

I. PENDAHULUAN

Rubik's Cube adalah sebuah permainan *puzzle* berbentuk kubus dengan 6 sisi. Tujuan utama dari permainan tersebut adalah untuk mengembalikan kubus pada kondisi akhirnya, dengan menyamakan warna pada setiap sisi kubus. Permainan ini sangat digemari oleh banyak orang, mulai dari anak kecil hingga orang dewasa, bahkan hingga saat ini sudah banyak perlombaan resmi yang diselenggarakan di seluruh dunia. Nama permainan tersebut berasal dari nama penciptanya, yakni Erno Rubik, seorang profesor arsitek dari Hungaria.

Setiap *piece* pada *Rubik's Cube* bergerak bebas di setiap sisinya, dan permainan ini sangat sulit untuk diselesaikan karena memiliki sebanyak 4.3×10^{19} kemungkinan dan hanya memiliki 1 solusi diantara kemungkinan tersebut. Serta ada banyak algoritma yang bisa digunakan untuk menyelesaikannya.

II. LANDASAN TEORI

Dalam bab ini akan dijelaskan tentang teori-teori yang berkaitan.

Graf

1. Definisi Graf

Graf G didefinisikan sebagai pasangan himpunan (V,E), ditulis dengan notasi G=(V,E), yang dalam hal ini V adalah himpunan tidak-kosong dari simpul-simpul (*vertices* atau *node*) dan E adalah himpunan sisi (*edges* atau *arcs*) yang menghubungkan sepasang simpul.

Simpul pada graf dapat dinomori dengan huruf, seperti a, b, c, …, v, w, …, dengan bilangan asli 1, 2, 3, …, atau gabungan keduanya. Sedangkan sisi yang menghubungkan simpul v_1 dengan simpul v_2 dinyatakan dengan pasangan (v_1, v_2) atau dinyatakan dengan lambang e_1, e_2, …. Dengan kata lain, jika e adalah sisi yang menghubungkan simpul v_1 dengan simpul v_2, maka e dapat ditulis sebagai

$$e = (v_1, v_2)$$

Secara geometri graf digambarkan sebagai sekumpulan noktah (simpul) di dalam bidang dwimatra yang dihubungkan dengan sekumpulan garis (sisi).

2. Jenis Graf

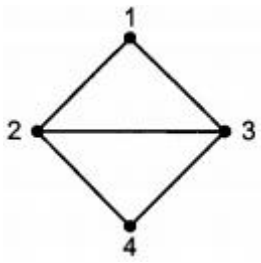
Graf dapat dikelompokkan menjadi beberapa kategori (jenis) bergantung pada sudut pandang pengelompokannya.

Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, maka secara umum graf dapat digolongkan menjadi dua jenis:

i. Graf sederhana (*simple graph*)

Graf yang tidak mengandung gelang maupun sisi-ganda dinamakan graf sederhana.

Pada graf sederhana, sisi adalah pasangan tak-terurut (*unordered pairs*). Jadi, menuliskan sisi (v_1, v_2) sama saja dengan (v_2, v_1).



Gambar 1. Graf sederhana (sumber: Munir, Rinaldi. 2005. Matematika Diskrit, Edisi 3.)

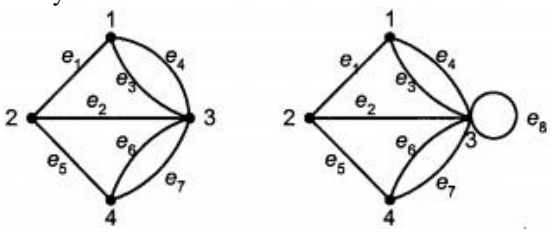
ii. Graf tak-sederhana (*unsimple-graph*)

Graf yang mengandung sisi ganda atau gelang dinamakan graf tak-sederhana (*unsimple graph*). Ada dua macam graf tak-sederhana, yaitu **graf ganda** (*multigraph*) dan **graf semu** (*pseudograph*).

Graf ganda adalah graf yang mengandung sisi ganda. Sisi ganda yang menghubungkan sepasang

simpul bisa lebih dari dua buah.

Graf semu adalah graf yang mengandung gelang (*loop*). Graf semu lebih umum daripada graf ganda, karena sisi pada graf semu dapat terhubung ke dirinya sendiri.



Gambar 2. Graf ganda (kiri) dan graf semu (kanan) (sumber: Munir, Rinaldi. 2005. Matematika Diskrit, Edisi 3.)

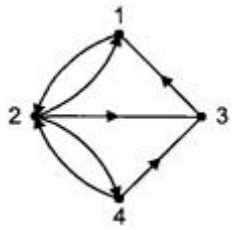
Sisi pada graf dapat mempunyai orientasi arah. Berdasarkan orientasi arah pada sisi, maka secara umum graf dibedakan atas 2 jenis:

i. Graf tak-berarah (*undirected graph*)

Graf yang sisinya tidak mempunyai orientasi arah disebut graf tak-berarah. Pada graf tak-berarah, urutan pasangan simpul yang dihubungkan oleh sisi tidak diperhatikan. Jadi, sisi $(v_1, v_2)=(v_2, v_1)$ adalah sama.

ii. Graf berarah (*directed graph* atau *digraph*)

Graf yang setiap sisinya diberikan orientasi arah disebut sebagai graf berarah.



Gambar 3. Graf berarah (sumber: Munir, Rinaldi. 2005. Matematika Diskrit, Edisi 3.)

3. Terminologi Dasar

Ada beberapa terminologi yang berhubungan dengan graf.

Berikut ini adalah beberapa contohnya:

a. Bertetangga (*Adjacent*)

Dua buah simpul pada graf tak-berarah G dikatakan bertetangga bila keduanya terhubung langsung dengan sebuah sisi. Dengan kata lain, u bertetangga dengan v jika (u,v) adalah sebuah sisi pada graf G.

b. Bersisian (*Incident*)

Untuk sembarang sisi e=(u,v), sisi e dikatakan bersisian dengan simpul u dan simpul v.

c. Simpul Terpencil (*Isolated Vertex*)

Simpul terpencil adalah simpul yang tidak mempunyai sisi yang bersisian dengannya. Atau, dapat juga dinyatakan bahwa simpul terpencil adalah simpul yang satupun bertetangga dengan simpul-simpul lainnya.

d. Graf Kosong (*Null Graph* atau *Empty Graph*)

Graf yang himpunan sisinya merupakan himpunan kosong disebut sebagai graf kosong dan ditulis sebagai N_n, yang dalam hal ini adalah jumlah simpul.

e. Derajat (*Degree*)

Derajat suatu simpul pada graf tak-berarah adalah jumlah sisi yang bersisian dengan simpul tersebut.

Pada graf berarah, derajat simpul v dinyatakan dengan $d_{in}(v)$ dan $d_{out}(v)$, yang dalam hal ini adalah:

$d_{in}(v)$ = derajat masuk = jumlah busur yang masuk ke simpul v;

$d_{out}(v)$ = derajat keluar = jumlah busur yang keluar dari simpul v;

dan $d(v) = d_{in}(v) + d_{out}(v)$.

f. Lintasan (*Path*)

Lintasan yang panjangnya n dari simpul awal v_0 ke simpul tujuan v_n di dalam graf G ialah barisan berselang-seling simpul-simpul dan sisi-sisi yang berbentuk $v_0, e_1, v_1, e_2, v_2, \ldots, v_{n-1}, e_n, v_n$ sedemikian sehingga $e_1 = (v_0, v_1)$, $e_2 = (v_1, v_2)$, …, $e_n = (v_{n-1}, v_n)$ adalah sisi-sisi dari graf G.

Sebuah lintasan dikatakan **lintasan sederhana** (*simple path*) jika semua simpulnya berbeda (setiap sisi yang dilalui hanya satu kali).

Lintasan yang berawal dan berakhir pada simpul yang sama disebut **lintasan tertutup** (*closed path*), sedangkan lintasan yang tidak berawal dan berakhir pada simpul yang sama disebut **lintasan terbuka** (*open path*).

g. Siklus (*Cycle*) atau Sirkuit (*Circuit*)

Lintasan yang berawal dan berakhir pada simpul yang sama disebut sirkuit atau siklus.

Sebuah sirkuit dikatakan **sirkuit sederhana** (*simple circuit*) jika setiap sisi yang dilalui berbeda.

h. Terhubung (*Connected*)

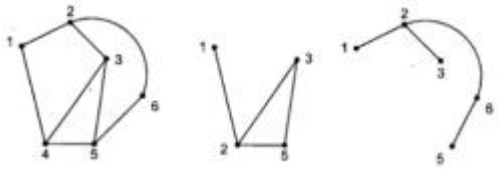
Grak tak-berarah G disebut **graf terhubung** (*connected graph*) jika untuk setiap pasang simpul u dan v di dalam himpunan V terdapat lintasan dari u ke v (yang juga harus berarti ada lintasan dari u ke v). Jika tidak, maka G disebut **graf tak-terhubung** (*disconnected graph*).

Graf berarah G disebut **graf terhubung kuat** (*strongly connected graph*) apabila untuk setiap pasang simpul sembarang a dan b terdapat lintasan berarah dari a ke b dan juga dari b ke a. Graf dikatakan **graf terhubung lemah** jika lintasan hanya terdapat satu arah, yaitu a ke b atau b ke a.

i. Upagraf dan Komplemen Upagraf

Sebuah upagraf dari graf G = (V, E) adalah sebuah graf $G_1 = (V_1, E_1)$ dengan $V_1 \subseteq V$ dan $E_1 \subseteq E$.

Komplemen dari upagraf G_1 adalah $G_2 = (V_2, E_2)$ dengan $E_2 = E - E_1$ dan E_2 bersisian simpul pada himpunan simpul V_2.



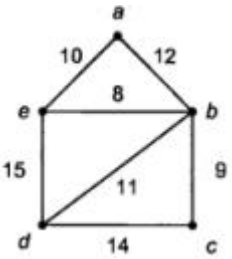
Gambar 4. Sebuah graf (kiri), upagraf (tengah), dan komplemennya (kanan) (sumber: Munir, Rinaldi. 2005. Matematika Diskrit, Edisi 3.)

j. Upagraf Merentang

Sebuah upagraf G_1 dikatakan upagraf merentang dari graf G bila upagraf G_1 mengandung seluruh simpul dari graf G.

k. Graf Berbobot

Sebuah graf berbobot adalah graf yang pada setiap sisinya diberi sebuah bobot. Arti dari bobot tersebut dapat berbeda-beda bergantung pada aplikasi dari graf. Contoh masalah yang dapat dimodelkan graf berbobot adalah jarak antar kota, waktu tempuh, dan lain-lain.



Gambar 5. Graf berbobot (sumber: Munir, Rinaldi. 2005. Matematika Diskrit, Edisi 3.)

III. ANATOMI RUBIK'S CUBE

Ada 3 macam *piece* yang menyusun *Rubik's Cube*:

1. *Corner piece*

Terdapat 8 buah *corner piece*. Setiap *corner piece*-nya memiliki 3 warna dan juga 3 orientasi yang berbeda.

Gambar 6. Corner piece pada Rubik's Cube (sumber: https://ryanstutorials.net/rubiks-cube-tutorial)

2. *Edge piece*

Terdapat 12 buah *edge piece*. Setiap *edge piece*-nya memiliki 2 warna dan juga 2 orientasi yang berbeda.

Gambar 7. Edge piece pada Rubik's Cube (sumber: https://ryanstutorials.net/rubiks-cube-tutorial)

3. *Center piece*

Terdapat 6 buah *center piece*. Setiap *center piece*-nya hanya memiliki 1 warna dan 1 orientasi. *Piece* ini yang akan menentukan warna apa yang tepat untuk setiap sisi kubusnya.

Gambar 8. Center piece pada Rubik's Cube (sumber: https://ryanstutorials.net/rubiks-cube-tutorial)

IV. METODE MENYELESAIKAN RUBIK'S CUBE

Ada banyak sekali metode dasar yang bisa digunakan untuk menyelesaikan *Rubik's Cube*, seperti contohnya metode *corner-first* (menyelesaikan *corner piece* terlebih dahulu) dan *edges-first* (menyelesaikan *edge piece* terlebih dahulu). Metode-metode ini sangat intuitif dan membutuhkan jumlah algoritma yang relatif sedikit.

Terdapat beberapa metode untuk menyelesaikan *Rubik's Cube* yang dikenal luas di dunia *speedcubing* (menyelesaikan rubik secepat mungkin), diantaranya:

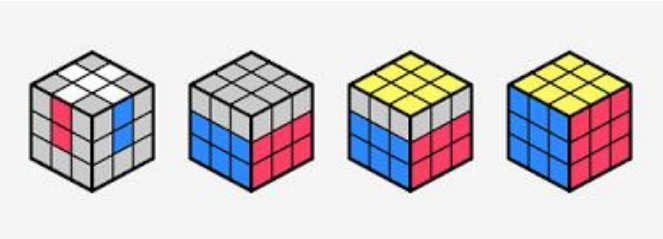
1. Metode Fridrich (CFOP)

Metode ini diberi nama sesuai dengan nama penemunya, yaitu Jessica Fridrich. Fridrich adalah seorang profesor di bidang elektro dan komputer di Binghamton University. Dia berhasil meraih peringkat 1 pada Chech National Championship pada tahun 1982 dengan metodenya sendiri. Metode ini biasa disingkat dan dikenal sebagai metode CFOP (Cross, F2L, OLL, PLL).

Sekitar tahun 1990, demam *Rubik's Cube* sempat memudar, namun minat Jessica dalam *speedcubing* muncul kembali saat temannya berhasil mengembangkan algoritma komputer yang dapat mencari gerakan paling efisien untuk menyelesaikan kubus rubik.

Seiring berjalannya waktu, beberapa variasi metode dan algoritma mulai dikembangkan dari metode dasar Fridrich. Algoritma-algoritma pada metode tersebut telah melewati banyak proses optimalisasi, dan hingga saat ini menjadi metode yang paling banyak digunakan bagi para peminat *speedcubing*.

Langkah awal dari metode ini yaitu membuat *Cross* (tanda plus) pada sisi putih, lalu menyelesaikan *First 2 Layers* (2 lapis terbawah, lalu *Orient Last Layer* (mengorientasikan lapis terakhir), dan terakhir *Permute Last Layer* (mempermutasikan lapis terakhir).



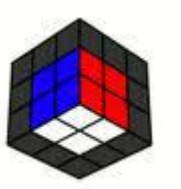
Gambar 6. Langkah-langkah metode Fridrich (sumber: http://jamiebicknell.com)

2. Metode Petrus

Metode ini dirancang oleh Lars Petrus, *speedcuber* seangkatan Jessica Fridrich. Berbeda dengan metode Fridrich yang sistematis, metode Petrus jauh lebih intuitif.

Metode ini mengharuskan untuk menggunakan logika dan kekuatan pikiran untuk membuat apa yang biasanya disebut "*block building*", yaitu menyelesaikan kubus rubik dengan membuat blok-blok kecil lalu memperluas blok tersebut. Pertama yang diselesaikan adalah blok 2x2x2, lalu diperluas menjadi 2x2x3. Selanjutnya seluruh *edge piece* diorientasikan dengan benar, kemudian kedua sisi rubik yang tersisa diselesaikan dengan algoritma Sune, Allan, dan Niklas.

Metode ini banyak digunakan dalam cabang perlombaan *Fewest Moves Competition* (menyelesaikan kubus rubik dengan gerakan paling sedikit). *Speedcuber* Indonesia yang terkenal cakap dalam menggunakan metode ini adalah Ardianto Satriawan ST, MT, *speedcuber* asal Bandung, lulusan Institut Teknologi Bandung.



Gambar 7. Gambaran metode Petrus (sumber: https://antomuhammad.wordpress.com/2013/09/22/metode-metode-speedcubing)

3. Metode Waterman

Metode Waterman dikembangkan dari metode *corner-first*. Orang yang mengembangkan metode ini adalah Marc Waterman yang memperoleh rekor waktu rata-rata 16 detik pada tahun 1980-an.

Gambar 8. Gambaran metode Waterman (sumber: https://antomuhammad.wordpress.com/2013/09/22/metode-metode-speedcubing)

4. Metode Roux

Metode ini dikembangkan oleh Gilles Roux. Langkah dalam metode ini diawali dengan membangun blok 3x2x1 yang terletak dibagian bawah pada lapis kiri kubus rubik. Tahap kedua adalah dengan menyusun blok 3x2x1 lainnya pada lapis yang berlawanan. Setelah keempat *corner piece* diselesaikan, yang tersisa adalah enam *edge piece* dan empat *center piece* yang diselesaikan pada tahap terakhir.

Metode ini tidak memerlukan banyak rotasi seperti pada metode Fridrich, dengan demikian gerakan yang dilakukan saat menyelesaikan kubus rubik lebih efisien.

Gambar 9. Gambaran metode Roux (sumber: https://antomuhammad.wordpress.com/2013/09/22/metode-metode-speedcubing)

5. Metode Heise

Metode yang memiliki tingkat kerumitan sangat tinggi ini dikembangkan oleh Ryan Heise. Tahap pertama pada metode ini adalah menyusun empat blok 1x2x2 yang saling menempel. Hal yang menarik, blok-blok ini tidak perlu memiliki warna yang sama sehingga kita leluasa untuk mengambil keuntungan terhadap blok-blok yang sudah tersusun di awal.

Tahap selanjutnya, *edge pieces* akan diorientasikan dan secara bersamaan blok yang telah ada akan disusun sesuai pasangannya, lalu *edge* yang masih tersisa diselesaikan. Bila

telah selesai, barulah *corner* diselesaikan dalam 2 tahap.

Gambar 10. Gambaran metode Heise (sumber: https://antomuhammad.wordpress.com/2013/09/22/metode-metode-speedcubing)

6. Metode Zborowski-Bruchem

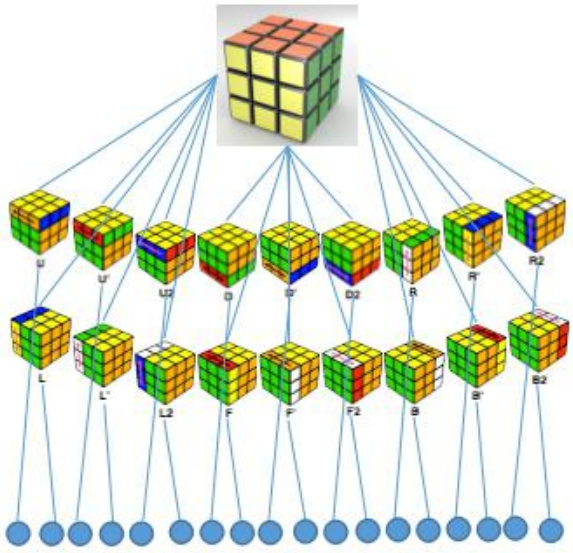
Metode ini dikembangkan oleh Zbigniew Zborowski dari Polandia dan Ron van Bruchem dari Belanda. Keuntungan besar metode ini terletak pada eksekusi *last layer* yang sangat cepat. Metode ini diyakini mampu membuat orang yang menguasainya dapat menyelesaikan kubus rubik dalam waktu

Gambar 11. Gambaran metode Zborowski-Bruchem (sumber: https://antomuhammad.wordpress.com/2013/09/22/metode-metode-speedcubing)

Masih banyak metode lain yang umumnya adalah hasil penyesuaian dari metode-metode di atas. Namun pada jurnal ini, penulis akan membahas tentang representasi graf pada metode Fridrich

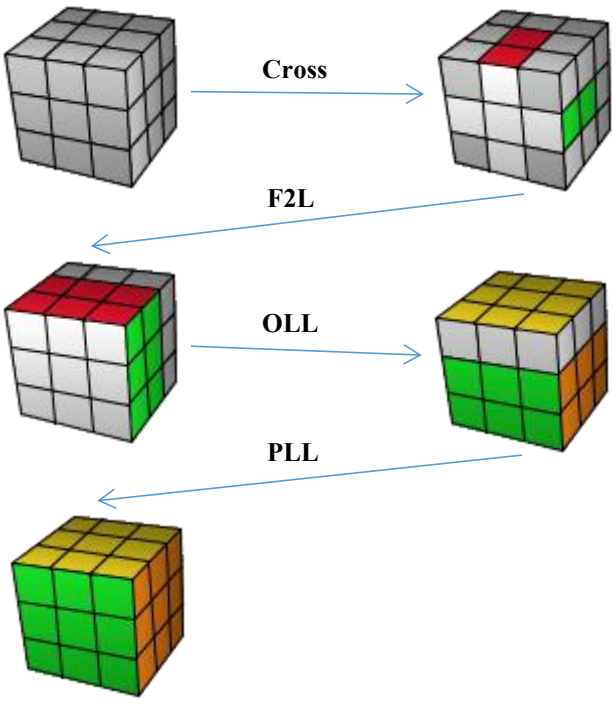
Pada dasarnya, setiap kemungkinan yang ada pada kubus rubik hanya dipengaruhi oleh gerakan pada 6 sisinya, di mana pada setiap sisinya akan direpresentasikan dengan lambang U (up), D (down), F (front), B (back), R (right), L(left), dan pada setiap sisinya terdapat 3 gerakan yang dapat dilakukan yaitu 90^o (X), 180^o (X2), dan 270^o (X') searah jarum jam. Sehingga total gerakan yang mungkin dilakukan adalah sebanyak 18 gerakan.

Dari kemungkinan gerakan-gerakan tersebut akan menghasilkan kombinasi kubus rubik berikutnya. Hal ini berlanjut sampai tercapai sebanyak 4,3 x 10^{19} kondisi. Akan kita anggap kondisi-kondisi kubus rubik tersebut sebagai simpul dari graf, dan gerakan-gerakan semua sisinya sebagai sisi dari graf. Namun hal ini tentu saja akan menjadi sangat rumit oleh karena banyaknya kondisi pada kubus rubik.



Graf 1. Semua kondisi pada Rubik's Cube

Jika kita menggunakan graf tersebut untuk menjadi acuan dalam tahapan menyelesaikan kubus rubik, maka kita tidak akan bisa mengetahui detil dari setiap langkah yang kita lakukan, dan apabila setiap kondisi dapat ditelusuri dalam waktu 1 detik, maka akan membutuhkan waktu sekitar 3 abad untuk bisa menyelesaikan kubus rubik tersebut, dikarenakan banyaknya kondisi pada kubus rubik tersebut. Oleh karena itu, dengan bantuan metode Fridrich, graf tersebut bisa disimpulkan menjadi graf sebagai berikut:



Graf 2. Langkah-langkah metode Fridrich (CFOP)

Dengan demikian, tahap-tahap untuk menyelesaikan *Rubik's Cube* menjadi lebih detil, serta jumlah gerakan yang diperlukan untuk menyelesaikannya menjadi lebih optimal, karena untuk

membuat *Cross* rata-rata memerlukan 6-8 gerakan, *F2L* rata-rata memerlukan 7-9 gerakan x 4 *pair F2L*, *OLL* rata-rata memerlukan 7-11 gerakan, dan *PLL* rata-rata memerlukan 7-17 gerakan. Sehingga total gerakan yang dibutuhkan untuk menyelesaikan kubus rubik dengan metode ini adalah sekitar 48-72 gerakan.

V. KESIMPULAN

Berdasarkan data di atas, dapat disimpulkan bahwa penerapan materi Matematika Diskrit terkait Graf dapat menyelesaikan suatu permasalahan pada pengoptimalan langkah dalam menyelesaikan permainan *puzzle Rubik's Cube.* Dengan menggunakan representasi graf, kita juga bisa melakukan penyederhanaan suatu algoritma yang rumit.

REFERENCES

[1] Munir, Rinaldi. 2005. Matematika Diskrit, Edisi 3. Bandung: Informatika Bandung.

[2] http://www.math.ubc.ca/~cass/courses/m308/projects/rtran/rtran.pdf Diakses pada tanggal 09 Desember 2018

[3] https://magnusonlineblog.wordpress.com/2017/10/27/rubiks-cube-solution-with-advanced-fridrich-cfop-method/ Diakses pada tanggal 09 Desember 2018

[4] https://antomuhammad.wordpress.com/2013/09/22/metode-metode-speedcubing/ Diakses pada tanggal 09 Desember 2018

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, 10 Desember 2018

Ahmad Naufal Hakim
13517055