Penarapan Teori Graf dan Kompleksitas Algoritma dalam Penyelesaian *Rubik's Cube* menggunakan strategi *Meet in the Middle*

Melvin Kent Jonathan - 13521052[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13521052@std.stei.itb.ac.id

***Abstrak* —*Rubik's Cube* atau Rubik merupakan permainan *puzzle* terpopuler di dunia dengan total kombinasi konfigurasi dari sebuah Rubik berukuran 3 x 3 x 3 sebanyak 43.252.003.274.489.856.000 kombinasi. Kini, Rubik juga dijadikan sebagai kejuaraan dunia dengan setiap pemain berusaha menyelesaiakan Rubik dengan waktu paling singkat. Selain kecepatan tangan, strategi penyelesaian yang efisien juga dibutuhkan untuk menyelesaikan sebuah kasus Rubik dengan waktu sesingkat mungkin. Makalah ini membahas mengenai penyelesaian Rubik menggunakan teori graf dan strategi *Meet in the Middle* yang akan memberikan langkah paling efisien untuk menyelesaikan sebuah kasus Rubik.**

***Kata Kunci*—*Rubik's Cube*, *Meet in the Middle*, graf, kompleksitas, algoritma.**

I. PENDAHULUAN

Rubik's Cube merupakan permainan *puzzle* berbentuk sebuah kubus yang setiap sisinya memiliki warna unik dan dipecah ke dalam *n x n* bagian untuk Rubik *n x n x n*. Permainan ini dicipatkan oleh seorang profesor arsitektur asal Hungaria bernama Ernő Rubik pada tahun 1974. Sejak dilisensikan dan dijual pada tahun 1978 kepada perusahaan Pentagle Puzzles di Inggris, permainan *Rubik's cube* menjadi permainan terlaris sepanjang masa, dengan estimasi total penjualan lebih dari 450 juta unit.

Tujuan dari permainan *Rubik's Cube* adalah membuat setiap sisi dari kubus menjadi warna yang sama sehingga tidak ada warna berbeda yang dapat ditemukan pada setiap sisinya. Penyelesaian dari kasus *Rubik's Cube* membutuhkan ketelitian dan kemahiran dalam berpikir secara dimensional karena permainan ini melibatkan perpindahan 3 dimensi. Adapun kini telah banyak teknik-teknik yang dikembangkan untuk menyelesaikan *Rubik's cube* sehingga yang perlu dilakukan pemain adalah menganalisis pola warna yang timbul pada kubus.

Variasi format bermain *Rubik's cube* cukup beragam. Salah satu yang terpopuler adalah *speedcubing*, yakni menyelesaikan sebuah *Rubik's Cube* dengan waktu sesingkat mungkin. Format bermain seperti ini sudah dikategorikan sebagai olahraga layaknya permainan catur dan telah dilombakan. Kejuaraan dunia dari perlombaan ini dinaungi oleh World Cube Association yang memiliki basis di Los Angeles, California, Amerika Serikat. Rekor dunia kejuaraan *speedcubing* dipegang oleh Yusheng Du untuk Rubik 3 x 3 x 3 dengan waktu 3,47 detik. Selain itu, rekor dunia untuk waktu penyelesaian rata-rata tersingkat dipegang oleh Mark Park dan Tymon Kolasinski dengan waktu 4,86 detik.

Untuk melakukan *speedcubing*, pemain tidak hanya membutuhkan kecepatan tangan, tetapi juga kecepatan berpikir dan pemilihan strategi penyelesaian yang efisien. Strategi penyelesaian yang efisien dapat ditentukan dari jumlah putaran sisi yang paling sedikit. Umumnya, pemain *Rubik's Cube*, terutama *speedcuber*, akan menghafalkan serangkaian gerakan putaran untuk kasus-kasus pola warna tertentu sehingga dapat menyelesaikan kasus yang disajikan tanpa perlu memvisualisasikan *outcome* dari gerakan putaran terlalu lama.

Jika dihitung berdasarkan kemungkinan konfigurasi yang ada, sebuah *Rubik's Cube* ukuran 3 x 3 x 3 memiliki kemungkinan konfigurasi sebanyak 43.252.003.274.489.856.000. Hal ini tentunya membuat pemain tidak mungkin menghafalkan konfigurasi dari *Rubik's Cube* satu per satu sehingga pada umumnya pemain tidak memikirkan pentingnya strategi penyelesaian dengan jumlah putaran paling sedikit. Namun, dengan tenaga komputasi yang dimiliki komputer zaman sekarang, pemain dapat mencari langkah terpendek yang dibutuhkan untuk menyelesaikan *Rubik's Cube* sehingga dapat mencapai efisiensi yang maksimal.

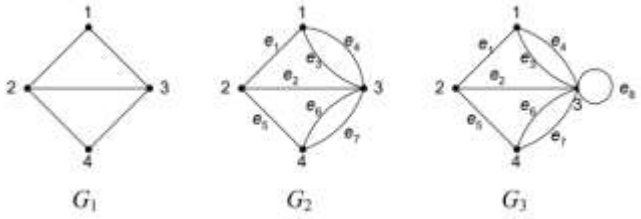
Representasi konfigurasi dan relasi antarkonfigurasi dari *Rubik's Cube* dapat menggunakan teori graf yang dipelajari di mata kuliah IF2120 Matematika Diskrit. Makalah ini akan membahas penggambaran konfigurasi *Rubik's Cube* menggunakan graf beserta dengan teknik pencarian langkah penyelesaian paling sedikit dari suatu konfigurasi menggunakan strategi *Meet in the Middle*.

II. TEORI DASAR

A. Graf

Graf merupakan konsep yang digunakan untuk merepresentasikan objek-objek diskrit serta relasinya satu sama lain. Secara formal, graf *G* didefinisikan sebagai pasangan himpinan (*V, E*). *V* merupakan himpunan tidak-kosong dari simpul-simpul (verteks), sedangkan *E* merupakan himpunan sisi (edge) yang menjadi penghubung sepasang simpul. Sebuah graf minimal memiliki satu buah simpul/verteks tanpa sebuah sisi pun. Graf yang seperti ini dinamakan graf trivial.

Graf seringkali direpresentasikan secara visual dengan 2 komponen utama, yakni verteks yang menggambarkan objek diskrit dan *edge*/sisi yang menggambarkan relasi antarobjek pada graf. Representasi visual seperti ini bukan merupakan satu-satunya, tetapi menjadi salah satu yang paling sering ditemukan karena sifatnya yang intuitif serta memberikan keuntungan dalam beberapa peninjauan terkait sifat-sifat graf.



Gambar 1. Representasi Visual Graf

Terdapat beberapa terminilogi terkait representasi sisi pada graf, yakni:

a. sisi ganda atau *multiple edges* atau *parallel edges*
   Sisi ganda merupakan sepasang sisi yang menghubungkan dua buah simpul yang sama. Contoh dari sisi ganda dapat dilihat pada Gambar 1 graf G_2 pada sisi e_3 dan sisi e_4.
b. gelang atau kalang atau *loop*
   Gelang merupakan sebuah sisi yang memiliki simpul awal dan simpul akhir yang sama. Contoh dari gelang dapat dilihat pada Gambar 1 graf G_3 pada sisi e_8.

B. Jenis-Jenis Graf

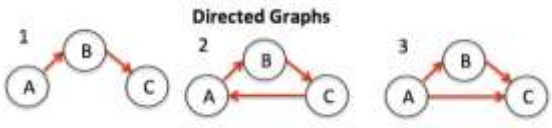
Pengkategorian Graf dapat ditinjau dari 2 aspek, yakni berdasarkan ada tidaknya gelang atau sisi ganda, maupun berdasarkan orientasi arah pada sisi graf.

Berdasarkan ada tidaknya gelang atau sisi ganda, graf dapat digolongkan menjadi 2 jenis:

1. graf sederhana (*simple graph*)
   Graf sederhana tidak mengandung gelang maupun sisi ganda.
2. graf tak-sederhana (*unsimple graph*)
   Graf tak-sederhana mengandung gelang atau sisi ganda. Graf tak-sederhana diberdakan lagi menjadi dua, yakni:
   1. graf ganda (*multi-graph*)
      Graf ganda merupakan graf yang mengandung sisi ganda.
   2. graf semu (*pseudo-graph*)
      Graf semu merupakan graf yang mengandung sisi gelang.

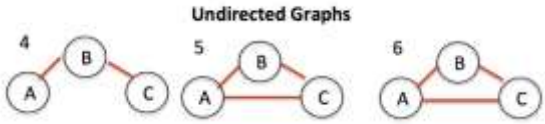
Berdasarkan orientasi arah pada sisi graf, graf dapat digolongkan menjadi 2 jenis:

1. graf berarah (*directed graph* atau *digraph*)
   Graf jenis ini memiliki orientasi arah pada sisinya. Contohnya dapat dilihat pada gambar di bawah ini.



Gambar 2. Graf Berarah

2. graf tak-berarah (*undirected graph*)
   Graf jenis ini tidak mempunyai orientasi arah pada sisinya. Contohnya dapat dilihat pada gambar di bawah ini.



Gambar 3. Graf Tak-Berarah

B. Terminologi Graf

Adapun beberapa terminologi terkait teori Graf dan sifat-sifatnya adalah sebagai berikut:

1. Ketetanggaan (*Adjacent*)
   Dua buah simpul disebut bertetangga apabila keduanya terhubung langsung oleh sebuah *edge*.
2. Berisisan (*Incidency*)
   Untuk sembarang sisi/*edge* $e = (v_j, v_k)$ dikatakan
   e bersisian dengan simpul v_j, atau
   e bersisian dengan simpul v_k
3. Simpul Terpencil (*Isolated Vertex*)
   Sebuah simpul/verteks v disebut sebagai simpul terpencil jika tidak terdapat sisi yang bersisian dengannya atau dengan kata lain v memiliki derajat 0 (tidak bertetangga).
4. Graf Kosong (*Null Graph* atau *Empty Graph*)
   Suatu graf G dikatakan sebagai graf kosong apabila himpunan sisinya merupakan himpunan kosong. Dengan kata lain, setiap verteks pada graf G memiliki derajat 0.
5. Derajat (*Degree*)
   Derajat suatu simpul adalah jumlah sisi yang bersisian dengan simpul tersebut. Derajat dinotasikan sebagai: $d(v)$. Lemma Jabat Tangan menyatakan bahwa jumlah derajat semua simpul selalu genap, yaitu dua kali jumlah sisi pada graf tersebut. Dengan mengetahui lemma tersebut, dapat disimpulkan untuk sembarang graf G, banyaknya simpul yang berderajat ganjil selalu genap.
6. Lintasan (*Path*)
   Lintasan merupakan barisan selang-seling simpul-simpul dan sisi-sisi pada graf G yang harus dilewati untuk mencapai suatu verteks dari verteks lainnya yang terhubung pada graf G. Panjang lintasan merupakan jumlah sisi yang berada dalam lintasan tersebut.
7. Siklus (*Cycle*) atau Sirkuit (*Circuit*)
   Siklus atau sirkuit merupakan sebutan untuk sebuah lintasan yang berawal dan berakhir pada simpul/verteks yang sama. Panjang sirkuit adalah jumlah sisi yang berada dalam sirkuit tersebut.
8. Keterhubungan (*Connected*)
   Dua buah simpul v_1 dan v_2 dikatakan terhubung bila terdapat lintasan dari v_1 ke v_2 pada suatu graf G. Graf G disebut sebagai graf terhubung (*connected graph*) jika untuk setiap pasang simpul v_i ke v_j dalam himpunan V terdapat lintasan dari v_i ke v_j. Jika tidak, maka G disebut sebagai graf tak-terhubung (*disconnected graph*).
   Dua simpul, u dan v, pada graf berarah G disebut terhubung kuat (*strongly connected*) jika terdapat lintasan berarah dari u ke v dan juga lintasan berarah dari v ke u. Jika u dan v tidak terhubung kuat tetapi terhubung pada graf tidak berarahnya, maka u dan v dikatakan terhubung lemah (*weakly connected*).

Graf berarah G disebut graf terhubung kuat (*strongly connected graph*) apabila untuk setiap pasang simpul sembarang u dan v di G, terhubung kuat. Kalau tidak, G disebut graf terhubung lemah.

9. Upagraf (*Subgraph*) dan Komplemen Upagraf
   Misalkan $G = (V, E)$ adalah sebuah graf. $G_1 = (V_1, E_1)$ adalah upagraf (subgraph) dari G jika $V_1 \subseteq V$ dan $E_1 \subseteq E$. Komplemen dari upagraf G_1 terhadap graf G adalah graf $G_2 = (V_2, E_2)$ sedemikian sehingga $E_2 = E - E_1$ dan V_2 adalah himpunan simpul yang anggota-anggota E_2 bersisian dengannya.
10. Graf Berbobot (*Weighted Graph*)
    Graf berbobot merupaka graf yang setiap sisinya diberi sebuah harga atau bobot.
11. Graf Lengkap (*Complete Graph*)
    Sebuah graf sederhana dapat dikatakan graf lengkap apabila setiap simpulnya mempunyai sisi ke semua simpul lainnya. Graf lengkap dengan n buah simpul dilambangkan dengan K_n. Jumlah sisi pada graf lengkap yang terdiri dari n buah simpul adalah $n(n-1)/2$.
12. Graf Lingkaran
    Graf lingkaran ialah graf sederhana yang setiap simpulnya berderajat dua dan dilambangkan dengan C_n untuk banyak simpul n.
13. Graf Teratur (*Regular Graph*)
    Graf teratur merupakan graf yang setiap simpulnya memiliki derajat yang sama. Apabila derajat setiap simpul adalah r, maka graf tersebut disebut sebagai graf teratur derajat r. Jumlah sisi pada graf teratur adalah $nr/2$.
14. Graf Bipartite (*Bipartite Graph*)
    Graf G yang himpunan simpulnya dapat dipisah menjadi dua himpunan bagian V_1 dan V_2, sedemikian sehingga setiap sisi pada G menghubungkan sebuah simpul di V_1 ke sebuah simpul di V_2 disebut graf bipartit dan dinyatakan sebagai $G(V_1, V_2)$.

C. Lintasan dan Sirkuit Euler

Lintasan Euler ialah lintasan yang melalui masing-masing sisi di dalam graf sebanyak tepat satu kali sedangkan sirkuit Euler merupakan sirkuit yang melewati masing-masing sisi tepat satu kali. Graf yang mempunyai sirkuit Euler disebut sebagai Graf Euler (*Eulerian Graph*) sedangkan graf yang mempunyai lintasan Euler disebut sebagai graf semi-Euler (*semi-Eulerian Graph*).

D. Lintasan dan Sirkuit Hamilton

Lintasan Hamilton adalah lintasan yang melalui tiap simpul di dalam graf sebanyak tepat satu kali sedangkan sirkuit Hamilton adalah sirkuit yang melalui tiap simpul di dalam graf tepat satu kali, kecuali simpul asal (sekaligus simpul akhir) yang dilalui dua kali. Graf yang mempunyai sirkuit Hamilton disebut sebagai graf Hamilton sedangkan graf yang hanya mempunyai lintasan Hamilton disebut sebagai graf semi-Hamilton.

E. Kompleksitas Algoritma

Algoritma yang efisien adalah algoritma yang menggunakan waktu dan memori seminimimum mungkin. Kebutuhan waktu dan ruang memori suatu algoritma bergantung pada ukuran masukan (n), yang menyatakan ukuran data yang diproses oleh algoritma.

Sebuah persoalan dapat diselesaikan dengan menggunakan berbagai jenis algoritma. Tentunya algoritma yang diinginkan adalah algoritma yang paling efisien secara waktu maupun penggunaan memori. Oleh karena itu, efisiensi dari algoritma dapat digunakan untuk menilai algoritma yang cocok untuk menyelesaikan sebuah persoalan secara optimal. Besaran yang dipakai untuk menerangka model abstrak pengukuran waktu/ruang ini adalah kompleksitas algoritma.

Kompleksitas algoritma dibagi ke dalam dua macam, yakni kompleksitas waktu (*time complexity*) dan kompleksitas ruang (*space complexity*). Kompleksitas waktu, $T(n)$, diukur dari jumlah tahapan komputasi yang dilakukan di dalam algoritma sebagai fungsi dari ukuran masukan n. Kompleksitas ruang, $S(n)$, diukur dari memori yang digunakan oleh struktur data yang terdapat dalam algoritma sebagai fungsi dari ukuran masukan n. Dengan besaran kompleksitas algoritma ini, kita dapat menentukan laju peningkatan waktu/ruang yang diperlukan algoritma seiring meningkatnya ukuran masukan n.

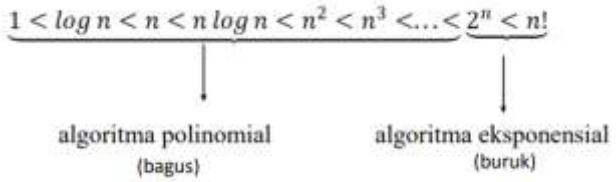
F. Notasi Big-O

Notasi "Big O" merupakan eskpresi formal untuk menyatakan kategori efisiensi dari sebuah algoritma dengan menggambarkan pertambahan langkah komputasional seiring bertambahnya data/jumlah elemen (n).

Tabel 1 Pengelompokkan Algoritma Berdasarkan Notasi Big-*O*

Kelompok Algoritma	Nama
$O(1)$	Konstan
$O(\log n)$	Logaritmik
$O(n)$	Linier
$O(n \log n)$	Linier Logaritmik
$O(n^2)$	Kuadratik
$O(n^3)$	Kubik
$O(2^n)$	Eksponensial
$O(n!)$	Faktorial

Urutan spektrum kompleksitas algoritma yakni:



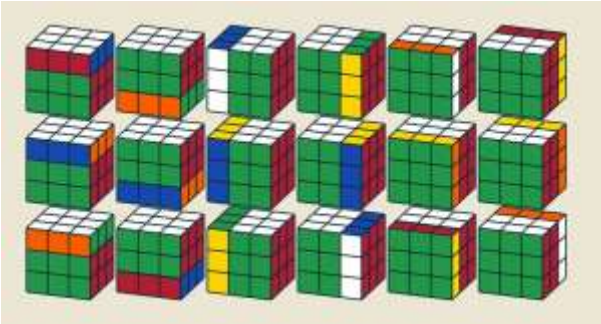
Gambar 4 Urutan Spektrum Kompleksitas Algoritma

III. Representasi Konfigurasi Rubik's Cube dengan Graf

A. Graf Ketetanggaan Konfigurasi

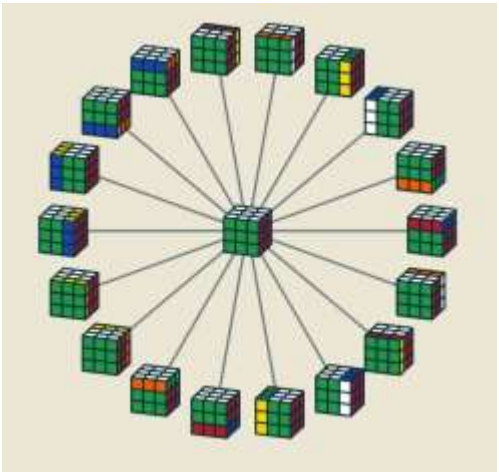
Konfigurasi warna dari *Rubik's Cube* dapat dimanipulasi dengan melakukan rotasi pada sisi-sisi dari kubus sehingga untuk setiap saat, langkah yang dapat dilakukan adalah rotasi pada sisi depan, belakang, kanan, kiri, atas, dan bawah. Setiap sisi dapat dirotasikan sejauh 270^o dalam kelipatan 90^o sehingga menghasilkan 3 variasi konfigurasi berbeda. Oleh karena itu, untuk setiap konfigurasi *Rubik's Cube* terdapat konfigurasi-konfigurasi lain yang dapat diperoleh dengan melakukan satu gerakan, yakni sejumlah :

$$6\ sisi \times 3\ konfigurasi/sisi = 18\ konfigurasi$$



Gambar 5 18 Konfigurasi Berbeda *Rubik's Cube*

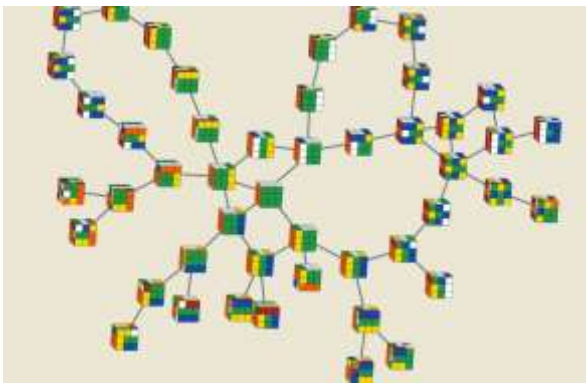
Bila dikaitkan dengan konsep graf, setiap konfigurasi pada 18 konfigurasi tersebut merupakan tetangga dari konfigurasi awal. Konfigurasi awal dengan tetangganya dihubungkan oleh sebuah sisi atau *edge* yang merepresentasikan 1 langkah valid dengan merotasikan salah satu sisi kubus. Representasi visual graf relasi antara konfigurasi awal dengan 18 konfigurasi tetangganya dapat dilihat pada gambar berikut.



Gambar 6 Representasi Visual Relasi Ketetangaan

B. Representasi Pencarian Solusi Rubik's Cube dengan Graf

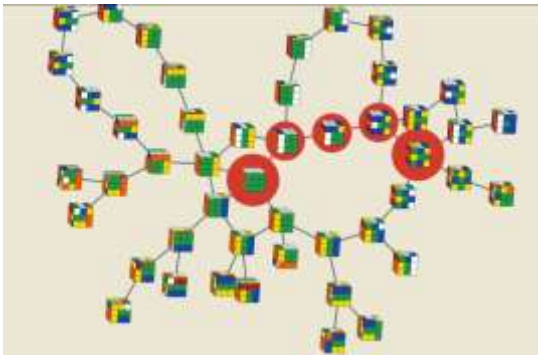
Untuk memahami pencarian solusi dari sebuah konfigurasi pada *Rubik's Cube*, kita dapat mengilustrasikan kasus-kasus pada *Rubik's Cube* menggunakan graf seperti gambar berikut.



Gambar 7 Representasi Visual Upagraf *Rubik's Cube*

Setiap konfigurasi unik dari *Rubik's Cube* digambarkan sebagai sebuah verteks. Dua buah konfigurasi yang dapat diperoleh dengan melakukan satu gerakan dihubungkan dengan sebuah garis/*edge*. Perlu dicatat bahwa ilustrasi di atas merupakan upagraf dari keseluruhan graf sehingga tidak menggambarkan seluruh kemungkinan konfigurasi tetangga dari masing-masing verteks. Dengan mengenal sifat *Rubik's Cube*, dapat diketahui bahwa seluruh kemungkinan dari konfigurasi pada *Rubik's Cube* dapat dicapai dari konfigurasi sembarang. Jika dihubungkan dengan teori graf, dapat disimpulkan bahwa graf keseluruhan dari konfigurasi *Rubik's Cube* merupakan graf terhubung (*connected graph*).

Pada graf terhubung ini, mencari sebuah solusi dari *Rubik's Cube* berarti mencari sebuah lintasan dari sebuah konfigurasi ke konfigurasi awal. Pada suatu kasus non-ideal, lintasan yang dibentuk dapat mengandung sisi ganda atau *multiple edges*, lintasan Euler, maupun lintasan Hamilton. Lintasan tersebut dapat diilustrasikan seperti pada gambar berikut.

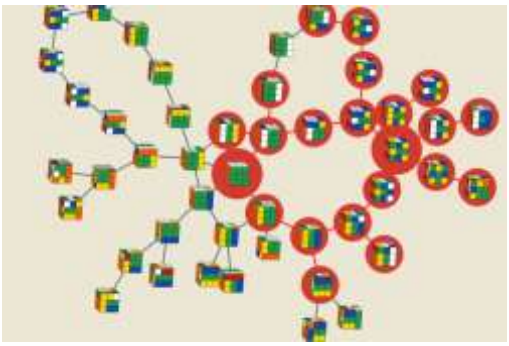


Gambar 8 Representasi Visual Lintasan

IV. Representasi Algoritma Pencarian Solusi Paling Efisien dari Konfigurasi Unik Rubik's Cube

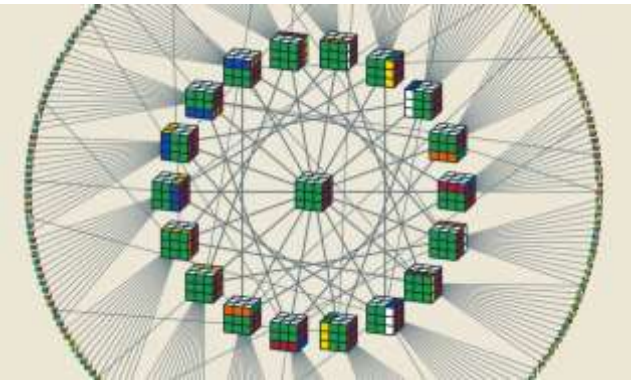
A. Strategi Breadth-First Search dan Kendalanya

Dalam mencari langkah penyelesaian paling efisien, bentuk lintasan yang ditargetkan adalah lintasan dengan panjang paling minimal (*shortest path*). Bila ditelaah lebih lanjut s*hortest path* juga akan membentuk lintasan Euler dan lintasan Hamilton karena setiap sisi dan setiap verteks dalam lintasan tersebut hanya akan dilewati sebanyak satu kali. Untuk mencari lintasan serupa pada sebuah graf, umumnya dilakukan *Breadth-First Search*. Algoritma *Breadth-First Search* (BFS) pada graf akan mengunjungi setiap verteks tetangga dari konfigurasi kasus, lalu mengunjungi setiap tetangga dari setiap verteks tetangganya, dan seterusnya hingga ditemukan verteks target. Berikut ini adalah ilustrasi BFS pada upagraf graf *Rubik's Cube*.



Gambar 9 Representasi Visual Alogritma *Breadth-First Search*

Namun, pengimplementasian algoritma *Breadth-First Search* pada graf *Rubik's Cube* tidaklah efisien karena untuk setiap verteks atau konfigurasi, terdapat 18 konfigurasi tetangga yang harus dikunjungi. Sebagai contoh, untuk menemukan lintasan dengan panjang 2, mesin harus menelusuri paling banyak 243 konfigurasi (beberapa konfigurasi dapat dicapai dari lintasan yang berbeda). Ilustrasi penelusuran yang harus dilakukan algoritma BFS pada panjang lintasan sebesar 2 dapat dilihat pada gambar berikut.



Gambar 10 Representasi Visual Algoritma BFS pada Lintasan dengan Panjang 2

Berdasarakan penelitian yang dilakukan oleh Tomas Rokicki dan dipublikasi pada Juli 2010 dengan judul "Towards God's Number for Rubik's Cube in the Quarter-Turn Metric", jumlah langkah paling banyak yang perlu dilakukan untuk menyelesaikan kasus *Rubik's Cube* adalah sebanyak 20 langkah. Dengan mengacu pada penelitian tersebut, panjang lintasan yang dibutuhkan untuk *worst-case scenario* adalah 20. Ilustrasi berikut menggambarkan *worst-case scenario* apabila pencarian dilakukan dari konfigurasi tertentu sebanyak 20 langkah.

Gambar 11 Representasi Visual *Worst-Case Scenario* Algoritma BFS

Hal ini akan menjadi masalah besar apabila pencarian *shortest path* dilakukan dengan menggunakan algoritma BFS karena setiap verteks pada setiap tingkatan di bawah 20 haruslah dikunjungi terlebih dahulu. Aproksimasi jumlah verteks yang harus dikunjungi BFS untuk panjang lintasan 20 dapat dilihat pada tabel berikut.

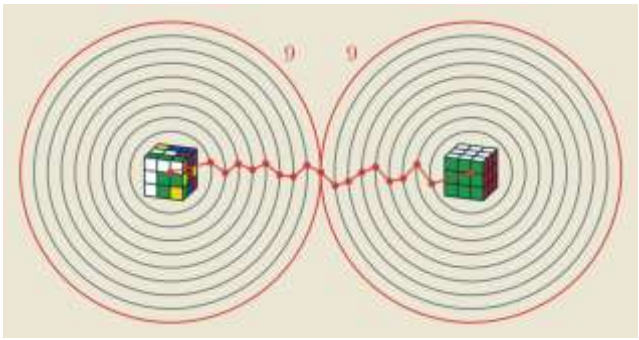
Tabel 2 Progresi Jumlah Konfigurasi yang Harus Dikunjungi Seiring Bertambahnya Panjang Lintasan

Langkah (Panjang Lintasan)	Konfigurasi yang Sudah Dikunjungi	Aproksimasi
0	1	$= 10^0$
1	19	$\sim 10^1$
2	262	$\sim 10^2$
3	3.502	~ 103
4	46.741	$\sim 10^4$
5	621.649	$\sim 10^5$
6	8.240.087	$\sim 10^6$
7	109.043.123	$\sim 10^7$
8	1.441.386.411	$\sim 10^8$
9	19.037.866.206	$\sim 10^9$
10	251.285.929.522	$\sim 10^{10}$
11	3.314.574.738.534	$\sim 10^{11}$
12	43.689.000.394.782	$\sim 10^{12}$
13	575.342.418.679.410	$\sim 10^{13}$
14	7.564.662.997.504.768	$\sim 10^{14}$
15	98.929.809.184.629.081	$\sim 10^{15}$
16	…	$\sim 10^{16}$
17	…	$\sim 10^{17}$
18	…	$\sim 10^{18}$
19	…	$\sim 10^{19}$
20	43.252.003.274.489.856.000	$\sim 10^{20}$

Dengan mengunjungi $\sim 10^n$ verteks/konfigurasi untuk panjang lintasan *n* pada graf, dibutuhkan waktu yang sangat panjang dan memori yang sangat besar. Untuk *worst-case scenario*, yakni $\sim 10^{20}$, apabila pencarian solusi dilakukan menggunakan komputer pada umumnya yang memiliki kecepatan eksplorasi sebesar 10^6 konfigurasi per detik, akan dibutuhkan waktu selama 10^{14} detik untuk mengeksplorasi verteks-verteks pada graf hingga ditemukan *shortest path* atau langkah penyelesaian minimal. Waktu 10^{14} detik ekuivalen dengan ~ 3 juta tahun. Maka dari itu, pencarian solusi paling efisien menggunakan algoritma BFS tidak dimungkinkan.

B. Strategi Meet in the Middle sebagai Alternatif Solusi

Ketimbang melakukan pencarian dari satu sisi saja seperti yang dilakukan pada algoritma BFS, strategi *Meet in the Middle* menawarkan algoritma yang lebih efisien, yakni dengan melakukan ekspansi dari kedua belah konfigurasi (konfigurasi awal dan konfigurasi akhir/solusi). Sering berjalannya ekspansi dari kedua belah konfigurasi, suatu saat akan ditemukan konfigurasi yang sama (beririsan) dan terbentuklah lintasan yang menghubungkan konfigurasi awal dan konfigurasi akhir.



Gambar 12 Representasi Visual *Shortest Path* dengan Algoritma *Meet in the Middle*

Dengan fakta pendukung bahwa *worst-case scenario* penyelesaian sebuah konfigurasi adalah sebanyak 20 langkah,

dapat dipastikan bahwa ekspansi dari masing-masing konfigurasi awal dan konfigurasi akhir tidak memakan lebih dari 10 langkah. Hal ini seolah menawarkan efisiensi yang sama dengan efisiensi yang ditawarkan algoritma BFS. Namun, apabila dihitung, verteks yang perlu dikunjungi oleh algoritma ini maksimal sebanyak ~ 2 x 10^{10} verteks, relatif kecil dibandingkan 10^{20} verteks oleh algoritma BFS.

Waktu yang dibutuhkan algoritma *Meet in the Middle* apabila pencarian solusi dilakukan menggunakan komputer pada umumnya yang memiliki kecepatan eksplorasi sebesar 10^6 konfigurasi per detik adalah ~ 2 x 10^4 detik untuk mengeksplorasi verteks-verteks pada graf hingga ditemukan *shortest path* atau langkah penyelesaian minimal. Waktu ini ekuivalen dengan sekitar 5 jam, yang berarti sangat mungkin dilakukan oleh komputer dengan tenaga komputasi zaman sekarang.

C. Perbandingan Kompleksitas Waktu dan Memori antara Algoritma Breadth-First Search dengan Algoritma Meet in the Middle

Apabila dituliskan dalam bentuk Notasi Big-O seperti yang dipelajari dalam mata kuliah IF2120 Matematika Diskirt, perbandingan kompleksitas waktu dan memori antara algoritma *Breadth-First Search* dan *Meet in the Middle* dapat dilihat pada tabel di bawah ini.

Tabel 3 Perbandingan Kompleksitas Waktu dan Memori

Kompleksitas	BFS	Meet in the Middle
Waktu	$O(10^n)$	$O(10^{n/2})$
Memori	$O(10^n)$	$O(10^{n/2})$

*n : jumlah langkah (*shortest path*)

V. KESIMPULAN

Graf memiliki aplikasi yang luas dalam kehidupan, salah satunya yakni pemodelan konfigurasi *Rubik's Cube* yang menjadi fokus bahasan pada makalah ini. Adapun pemahaman tentang sifat-sifat graf akan membantu dalam memahami penyelesaian dari sebuah kasus konfigurasi *Rubik's Cube*. Pemahaman tersebut jika diiringi dengan pemilihan strategi dapat membawa keuntungan perihal pemilihan langkah yang paling efisien dan menyelesaikan kasus yang disajikan. Dalam pembahasan makalah ini, algoritma *Meet in the Middle* dapat lebih mudah dipahami jika divisualisasikan menggunakan konsep graf serta dapat menawarkan kelebihan kompleksitas waktu dan memori yang signifikan ketimbang algoritma *Breadth-First Search* yang umumnya dipakai sebagai strategi penyelesaian kasus *searching* pada graf.

VI. UCAPAN TERIMA KASIH

Penulis berterima kasih kepada:

1. Tuhan YME atas rahmat anugerah-Nya sehingga makalah ini dapat selesai dengan baik.
2. Bapak Dr. Ir. Rinaldi Munir, M. T., Ibu Dra. Harlili S., M.Sc., Ibu Fariska Zakhralativa Ruskanda, S.T., M.T., dan Ibu Dr. Nur Ulfa Maulidevi, S.T., atas bimbingan dan pengajarannya dalam mata kuliah IF2120 Matematika Diskrit, khususnya kepada Ibu Ulfa sebagai dosen pengajar di K-01Teknik Informatika ITB.
3. Teman-teman penulis yang telah mendukung penulisan makalah ini.

REFERENCES

[1] Munir,Rinaldi. 2022. Graf (Bag. 1): Bahan Kuliah IF2120 Matematika Diskrit. Merupakan slide bahan ajar perkuliahan. Diakses pada 11 Desember 2022.

[2] Munir,Rinaldi. 2022. Graf (Bag. 2): Bahan Kuliah IF2120 Matematika Diskrit. Merupakan slide bahan ajar perkuliahan. Diakses pada 11 Desember 2022.

[3] Munir,Rinaldi. 2022. Graf (Bag. 3): Bahan Kuliah IF2120 Matematika Diskrit. Merupakan slide bahan ajar perkuliahan. Diakses pada 11 Desember 2022.

[4] Munir,Rinaldi. 2022. Kompleksitas Algoritma (Bag. 1): Bahan Kuliah IF2120 Matematika Diskrit. Merupakan slide bahan ajar perkuliahan. Diakses pada 11 Desember 2022.

[5] Munir,Rinaldi. 2022. Kompleksitas Algoritma (Bag. 2): Bahan Kuliah IF2120 Matematika Diskrit. Merupakan slide bahan ajar perkuliahan. Diakses pada 11 Desember 2022.

[6] T. Rokicki, "Towards god's number for Rubik's Cube in the quarter-turn metric," *The College Mathematics Journal*, vol. 45, no. 4, pp. 242–253, 2014.

[7] "God's algorithm out to 15F*," *God's Algorithm out to 15f* | Domain of the Cube Forum*. [Online]. Available: http://cubezzz.dyndns.org/drupal/?q=node%2Fview%2F201. [Accessed: 12-Dec-2022].

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, 12 Desember 2020

Melvin Kent Jonathan
13521052