Penggunaan Algoritma *Backtracking* Untuk Menentukan Keisomorfikan Graf

Neni Adiningsih[1], Dewi Pramudi Ismi[2], Ratih[3]

Laboratorium Ilmu dan Rekayasa Komputasi
Departemen Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if13007@students.if.itb.ac.id[1] , if13014@students.if.itb.ac.id[2],
if13016@students.if.itb.ac.id[3]

Abstrak

Graf merupakan pokok bahasan yang banyak terapannya hingga saat ini. Representasi visual dari graf adalah menyatakan objek sebagai titik (*vertices*) dan hubungan antar objek dinyatakan dengan garis (*edges*). Salah satu permasalahan dalam topik graf adalah menentukan apakah dua buah graf isomorfik atau tidak. Dua buah graf dikatakan isomorfik jika terdapat korespondensi satu-satu antara simpul-simpul keduanya. Jika dua buah graf yang isomorfik maka keduanya mempunyai jumlah simpul yang sama, mempunyai jumlah sisi yang sama, dan mempunyai jumlah simpul yang sama berderajat tertentu. Namun ternyata ketiga syarat tersebut belum cukup menjamin keisomorfikan. Dalam makalah ini, akan dibahas pemanfaatan algoritma *bactracking* untuk mencari korespondensi satu-satu antara simpul-simpul pada suatu graf dengan simpul-simpul pada graf lain.

***Kata kunci**: graf, graf isomorfik, algoritma backtracking*

1. Pendahuluan

Graf sangat banyak terapannya dalam teknologi informasi. Dalam banyak hal terkadang diperlukan suatu pembuktian apakah dua graf isomorfik atau tidak. Biasanya hal ini cenderung dilakukan dengan metode *trial and error*. Untuk mendapatkan cara yang lebih valid, diperlukan suatu algoritma yang dapat membuktikan keisomorfikan dengan pasti .

2. Graf

2.1 Definisi Graf

Graf merupakan suartu diagram yang memuat informasi tertentu jika diinterpresentasikan secara tepat. Dalam kehidupan sehari – hari, graf digunakan untuk menggambarkan berbagai macam struktur yang ada. Tujuannya adalah sebagai visualisasi objek – objek agar lebih mudah dimengerti. Misalnya : graf organisasi, bagan alir pengambilan mata kuliah, peta, rangkaian listrik, dan lain – lain.
Tiap – tiap diagram memuat sekumpulan objek (kotak, titik, dan lain – lain) beserta garis – garis yang menghubungkan objek – objek tersebut. Garis bisa berarah atau tidak. Garis yang berarah biasanya digunakan untuk menyatakan hubungan yang mementingkan urutan antar objek – objek (arah).

Graf G didefenisikan sebagai pasangan himpunan (V,E), yang dalam hal ini :
V = himpunan tidak kosong dari simpul – simpul (*vertices* atau *node*)

$$= \{v_1, v_2, v_3, \ldots, v_n\}$$

dan

E = himpunan sisi (*edges* atau *arcs*) yang menghubungkan sepasang simpul

$$= \{e_1, e_2, e_3, \ldots, e_n\}$$

Suatu graf G terdiri dari 2 himpunan yang berhingga, yaitu himpunan titik – titik tidak kosong (simbol V (G)) dan himpunan garis – garis (simbol (E (G))). Dua titik dikatakan **berhubungan (*adjacent*)** jika ada garis yang menghubungkan keduanya. Titik yang tidak mempunyai garis yang berhubungan disebut **titik terasing**. Graf yang tidak memiliki titik (sehingga tidak mempunyai garis) disebut graf kosong.

Jika semua garisnya berarah maka graf tersebut disebut **Graf Berarah (*Directed Graph* / *Digraph*)**. Jika semua garis tidak berarah, maka graf tersebut disebut **Graf Tak Berarah (*UnDirected Graph*).**

2.2 Isomorfisme pada Graf

Dalam Geometri, dua gambar disebut kongruen jika keduanya mempunyai sifat – sifat geometri yang sama. Dengan cara yang sama, dua graf dikatakan isomorfik jika keduanya menunjukkan bentuk yang sama. Kedua graf hanya berbeda dalam hal pemberian label titik dan garisnya saja.

Jika G adalah suatu graf dengan himpunan titik V (G) dan himpunan garis E (G). G' adalah graf dengan himpunan titik V (G') dan himpunan garis E (G'). G isomorfik dengan G' jika dan hanya jika ada korespodensi satu - satu

$$g : V(G) \rightarrow V(G') \text{ dan}$$

h : E (G) → V (G')

Sedemikian hingga

$(\forall v, w \in V(G)$ dan $e \in E(G))$

V, w adalah titik – titik ujung e $\Leftrightarrow$ g(v) dan g(w) adalah titik – titik ujung h(e).

Hingga saat ini belum ada teori yang dapat dipakai untuk menentukan apakah dua graf G dan G' isomorfik. Tetapi, jika graf G dan G' isomorfik, maka kedua graf tersebut selalu memenuhi 3 syarat sebagai berikut :

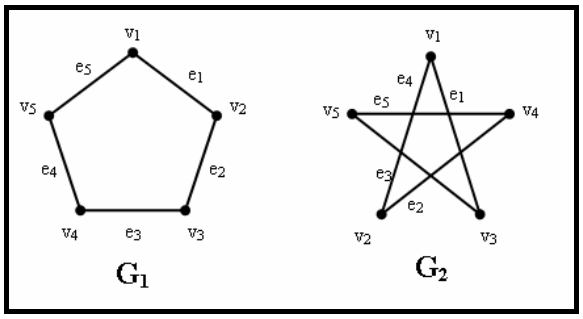
1. jumlah titik G = jumlah titik G' (jumlah simpul yang sama)
2. jumlah garis G = jumlah garis G' (jumlah sisi yang sama)
3. jumlah garis yang mempunyai derajat tertentu dalam graf G dan G' sama (mempunyai jumlah simpul yang sama berderajat tertentu)

Ketiga syarat tersebut belum cukup menjamin bahwa kedua graf isomorfik. Untuk menunjukkan bahwa kedua graf G dan G' isomorfik, maka dapat dilihat dari matriks ketetanggaan kedua graf tersebut sama.

Isomorfime pada graf berarah sama dengan isomorfisme pada graf tak berarah. Hanya saja pada isomorfisme graf berarah, korespondensi dibuat dengan memperhatikan arah garis.

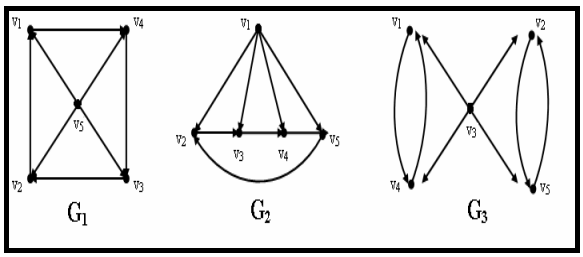
2.3 Gambar Graf Isomorfik

2.3.1 Graf Tak Berarah



Gambar 3.1 G_1 Isomorfik dengan G_2

2.3.2 Graf Berarah



Gambar 3.2 G_1, G_2, G_3 saling Isomorfik

3. Algoritma *Backtracking*

3.1 Pengertian Algoritma *Backtracking*

Istilah *Backtracking* pertama kali diperkenalkan oleh D.H. Lehmer tahun 1950. Selanjutnya R.J. Walker, Golomb, dan Baumert menyajikan uraian umum tentang algoritma ini serta penerapannya pada berbagai persoalan.

Algoritma *Backtracking* merupakan algoritma yang berbasis pada DFS (*Depth First Search*). Algoritma ini merupakan perbaikan dari algoritma *Brute Force*. Algoritma *Backtracking* ini tidak memeriksa semua kemungkinan solusi yang ada, hanya pencarian yang mengarah ke solusi saja yang selalu dipertimbangkan. Sehingga waktu pencarian dapar dihemat. Ada beberapa metode yang digunakan dalam algoritma *Backtracking* yaitu skema iteratif dan skema rekursif.

3.2 Properti Algoritma *Backtracking*

Untuk menerapkan algoritma ini, perlu didefinisikan properti-properti berikut

- Solusi persoalan
  $X = (x_1,x_2,\ldots x_n)$
  Solusi persoalan dinyatakan dengan vektor n – *tuple*.
- Fungsi pembangkit
  Fungsi ini digunakan untuk membangkitkan komponen vektor solusi.
  Fungsi ini dinyatakan sebagai T(k).
  T(k) membangkitkan nilai k.
- Fungsi pembatas / fungsi kriteria
  Fungsi pembatas menentukan apakah $(x_1,x_2,..x_k)$ mengarah kepada solusi.
  Fungsi batas ini dinyatakan B(k).

Algoritma *backtracking* mengorganisasi ruang solusi dengan struktur pohon. Tiap simpul pohon menyatakan status persoalan, sedangkan sisi pohon menyatakan nilai-nilai x_i.

3.3 Prinsip Umum Algoritma *Backtracking*

Solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan yang dipakai adalah mengikuti pencarian secara DFS. Simpul yang sudah dilahirkan dinamakan **simpul hidup**. Simpul hidup yang sedang diperluas dinamakan **simpul-E** (*Expand-node*). Simpul dinomori sesuai dengan urutan pembangkitannya.

Setiap kali simpul-E diperluas, lintasan yang dibangun bertambah panjang. Jika lintasan yang dibentuk tidak mengarah kepada solusi maka simpul-E dihapus. Fungsi yang digunakan untuk menghapus simpul-E adalah fungsi pembatas/ fungsi kriteria. Simpul yang sudah mati tidak akan diperluas lagi.

Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian diteruskan dengan membangkitkan simpul anak yang lainnya. Bila tidak ada simpul anak yang lainnya maka solusi dilanjutkan dengan melakukan runut balik ke simpul hidup yang terdekat. Selanjutnya simpul ini menjadi simpul hidup yang baru. Lintasan baru dibangun kembali hingga lintasan tersebut membentuk solusi.

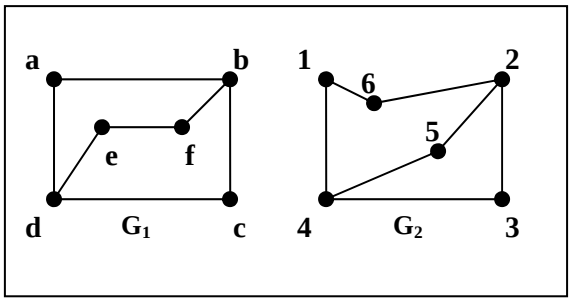
Pencarian dihentikan bila tidak menemukan solusi atau tidak ada lagi simpul hidup untuk *backtracking.*

4. Penentuan Keisomorfikan Graf dengan Algoritma *Backtracking*

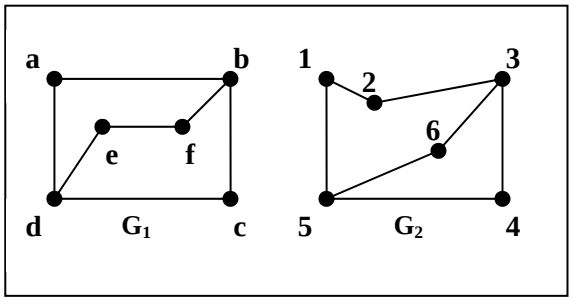
Dua buah graf yang memiliki jumlah simpul yang sama (misalkan n simpul) G1 dan G2 akan dicari keisomorfikannya. Kedua graf tersebut harus dipilih salah satu untuk digunakan sebagai patokan awal. Dalam hal ini misalkan G1 digunakan sebagai patokan kemudian semua simpul pada graf G1 disimpan dalam suatu vektor n-*tuple* misalkan V1.

Algoritma *backtracking* digunakan untuk menghasillan vektor n-*tuple* yang lain yaitu V2 yang komponennya merupakan simpul-simpul G2 yang berkorespondensi satu-satu dengan komponen pada vektor V1. G1 dan G2 dikatakan isomorfik jika V2 mengandung semua simpul pada graf G2.

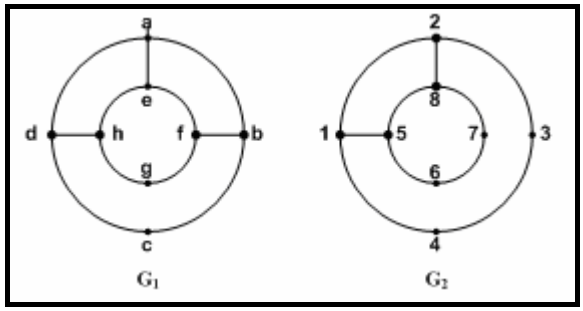
Misalkan algoritma *backtracking* digunakan untuk meunjukkan keisomorfikan kedua graf berikut ini.



Gambar 4.1 Graf Isomorfik



Gambar 4.2 Graf Isomorfik



Gambar 4.3 Graf Tidak Isomorfik

Semua simpul dalam graf G1 disimpan dalam V1 = (a,b,c,d,e,f). Masing-masing simpul yang tersimpan dalam vektor V1 merupakan simpul yang terhubung secara berurutan.

a terhubung dengan b

b terhubung dengan c

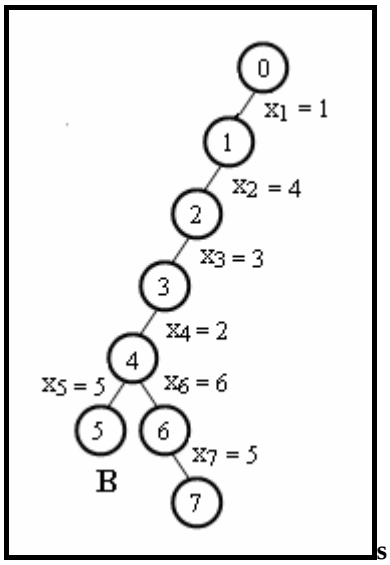
c terhubung dengan d

d terhubung dengan e

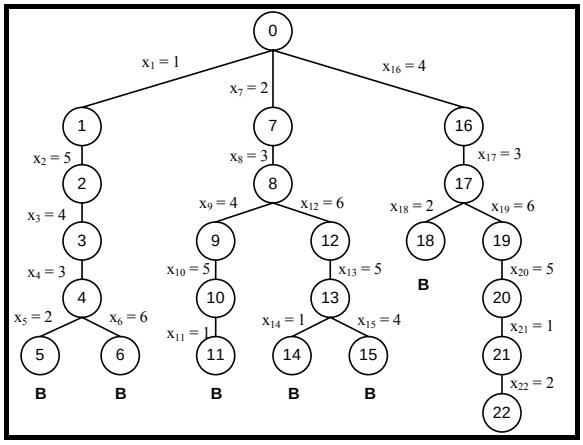
e terhubung dengan f

Kemudian akan dibangun pohon solusi untuk memperoleh vektor V2 yang merupakan vektor solusi. Komponen vektor V2 merupakan simpul-simpul yang berkorespondensi dengan komponen vektor V1.

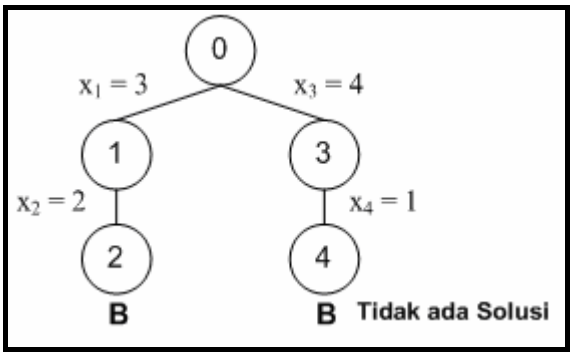
4.1 Pembangkitan Pohon Ruang Solusi



Gambar 4.1.1 Pohon Ruang Solusi untuk gambar 4.1



Gambar 4.1.2 Pohon Ruang Solusi untuk gambar 4.2



Gambar 4.1.3 Pohon Ruang Solusi untuk gambar 4.3

4.2 Pseudocode Algoritma *Backtracking*

```
procedure BacktrackingForIsomorphism ()
begin
    Susuan V1 dari semua simpul pada G1
    repeat
     pilih simpul yang sesuai komponen V1
     if simpul pada G2 cocok dengan simpul G1
     then
      begin
        masukkan simpul pada V2
        if  V2 belum mencakup semua simpul
        pada G2  then
        begin
          BacktrackingForIsomorphism()
          If simpul pada G2 tidak cocok dengan
          simpul pada G1 then batalkan
        end
      end
    until semua simpul pada G2 sudah
    dibangkitkan dalam V2
end
```

4.3 Analisis

Pada subbab 2.2 telah dijelaskan bahwa cukup sulit untuk menentukan bahwa dua buah graf isomorfik. Bahkan belum ada cara yang pasti untuk menentukannya, namun berdasarkan beberapa percobaan yang kelompok kami lakukan terbukti bahwa untuk graf yang isomorfik akan selalu menemui solusi, dan untuk dua graf yang tidak isomorfik tidak akan pernah menemui solusi.

Untuk graf yang tidak memiliki solusi(tidak isomorfis), pohon ruang solusi yang dibentuk tidak akan menghasilkan jalur yang memuat semua simpul graf kedua. Pohon ruang solusi akan selalu melakukan *backtrack* sampai akhirnya tidak ada lagi simpul yang dapat dibangkitkan.

Untuk dua buah graf yang memang isomorfik akan terdapat daun akhir yang jalur atau pathnya memuat semua simpul yang ada pada graf kedua. Jalan untuk mendapatkan path itu bisa berbeda-beda. Karena itu kemangkusan dari algoritma ini berbeda-beda untuk setiap kasus. Kompleksitas akan bergantung pada penempatan atau pemilihan urutan simpul masing-masing graf. hal ini dapat dilihat dengan urutan penempatan simpul pertama (lihat pohon ruang solusi untuk gambar 4.1.1) langkah yang ditempuh hanya tujuh langkah dengan satu kali backtracking. Berbeda jauh dengan kompleksitas dari urutan penempatan simpul kedua(lihat pohon ruang solusi untuk gambar 4.1.2) langkah yang ditempuh sebanyak 22 langkah dengan 6 kali backtracking.

Sejauh ini kami belum dapat menemukan langkah yang pasti untuk memastikan bahwa urutan penempatan simpul yang digunakan adalah yang terbaik. namun kami beranggapan bahwa algoritma ini cukup mangkus untuk digunakan dalam menentukan keisomorfisan dari dua buah graf karena pembuktian dengan cara manual (memenuhi tiga syarat dan memeriksa secara visual) akan membutuhkan waktu yang lebih lama, dan tentu saja jika direalisasikan dalam bentuk algoritma juga cukup sulit atau bahkan mustahil dibuat, selain itu belum tentu memiliki hasil yang selalu tepat untuk setiap kasus.

5. Kesimpulan

Algoritma *Backtracking* dapat digunakan untuk menentukan keisomorfikan dua buah graf secara pasti. Artinya dengan algoritma ini dapat disimpulkan apakah dua buah graf tersebut isomorfik atau tidak. Jika solusi persoalan yang dihasilkan (vektor n-*tuple*) mencakup semua simpul dari graf dengan n buah simpul maka kedua graf tersebut isomorfik. Sebaliknya jika vektor n-*tuple* tidak bisa dibangkitkan maka kedua graf tersebut tidak isomorfik.

Daftar Pustaka

1. Ellis Horowitz and Sartaj Sahni, *Fundamentals of Computer Algorithms,* Pitman Publishing, 1978.
2. Rinaldi Munir, *Diktat Kuliah IF2151 Matematika Diskit,* Lab Ilmu Rekayasa Komputasi, 2001
3. Kenneth Rosen, *Discrete Mathematics and Its Application* , Mc Graw Hill.
4. Jong Jeng Siang, Matematika Diskrit dan Aplikasinya pada Ilmu Komputer, Andi Yogyakarta, Yogyakarta : 2002.
5. Niklaus Wirth, *Algorithms + Data Structures = Programs*, Prentice-Hall, Inc., 1976.