Pewarnaan Simpul pada Graf dan Aplikasinya dalam Alokasi Memori Komputer

Andreas Parry Lietara ~ NIM 13506076
Jurusan Teknik Informatika ITB, Bandung, email : andreas-parry@students.itb.ac.id

Abstract *- Makalah ini membahas mengenai salah satu aspek studi dalam graf, yaitu pewarnaan graf. Pewarnaan graf merupakan salah satu aspek penting dalam studi mengenai graf, mengingat banyaknya aplikasi dari graf berdasarkan pewarnaan graf. Dalam makalah ini, akan dibahas salah satu aplikasi dari pewarnaan graf yaitu dalam alokasi memori komputer. Penggunaan ruang memori komputer dapat diperhemat dengan menggunakan metoda pewarnaan graf.*

Kata Kunci *: graf,simpul, sisi, pewarnaan simpul graf, interferensi peubah, graf interferensi, algoritma sequential coloring.*

1. PENDAHULUAN

Dalam mengeksekusi suatu program, *compiler / interpreter* mengubah bahasa pemrograman menjadi bahasa mesin yang akan dieksekusi oleh *CPU*. Untuk mengeksekusi program ini, komputer harus menyimpan instruksi dalam bahasa mesin tersebut beserta seluruh peubah yang dipakai dalam program. Jika program dalam bahasa mesin disimpan dalam satu blok ruang dalam memori, lain halnya dengan peubah - peubah program, peubah – peubah program tidak semuanya digunakan dalam waktu atau rentang waktu yang sama, bahkan beberapa peubah program hanya digunakan sekali lalu tidak digunakan kembali. Penggunaan kembali ruang memori yang telah tidak diperlukan akan memberikan perbedaan yang cukup mencolok dalam hal memori yang dibutuhkan program.

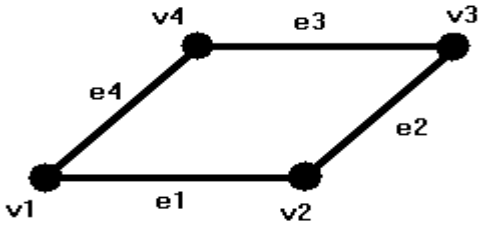
2. TEORI

2.1. Definisi Graf

Sebelum membahas mengenai pewarnaan graf dan aplikasinya dalam efisiensi alokasi memori dalam komputer, hendaknya kita membahas sedikit mengenai graf itu sendiri.

Sebuah graf digambarkan sebagai kumpulan titik-titik simpul yang disebut *vertices* atau *node* beserta sisi-sisi berupa garis yang menghubungkan titik-titik tersebut, garis-garis itu dinamakan *edges* atau *arcs*. Dalam matematika diskrit, graf didefinisikan sebagai pasangan himpunan (V,E) di mana :
V adalah himpunan tidak kosong dari titik-titik simpul = {v1, v2, v3, …, vn}, sedangkan E adalah himpunan sisi dari graf tersebut yang menghubungkan sepasang simpul pada graf = {e1, e2, …, en}. Jika $e1$ menghubungkan vi dan vj, maka dapat ditulis e1 = (vi,vj).

Gambar 1. Contoh Graf G1



Gambar 1 merupakan sebuah contoh graf G = (V, E) di mana V = (v1, v2, v3, v4) dan E = (e1, e2, e3, e4) .[1]

Perhatikan sekali lagi gambar 1, simpul v1 dan v4 dikatakan sebagai simpul-simpul yang saling bertetangga. Dua buah simpul dikatakan bertetangga jika kedua simpul itu dihubungkan langsung dengan sebuah sisi. Dalam kasus di gambar 1 v1 dan v2 dihubungkan langsung oleh sebuah sisi e4. Selain itu, pasangan simpul v1-v2, v2-v3, dan v3-v4 juga dikatakan bertetangga.

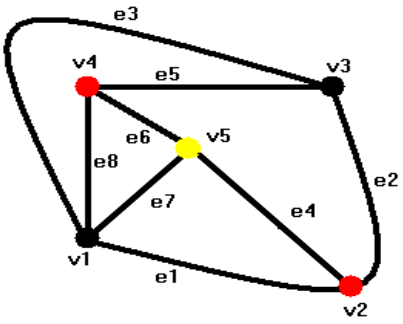
2.2. Pewarnaan Simpul pada Graf

Pewarnaan pada graf dibedakan menjadi tiga, pewarnaan simpul, pewarnaan sisi dan pewarnaan wilayah. Berhubung kita hanya akan membahas masalah yang menyangkut pewarnaan simpul, maka di sini hanya akan dikaji mengenai pewarnaan simpul.

Pewarnaan simpul pada graf adalah proses pemberian warna pada simpul-simpul suatu graf sehingga tidak ada dua buah simpul yang bertetangga pada graf tersebut berwarna sama. [1]

Gambar 2 merupakan sebuah contoh graf yang telah diwarnai masing-masing simpulnya dengan syarat pewarnaan simpul menggunakan tiga buah warna. Perhatikan bahwa tidak ada satupun simpul yang bertetangga yang berwarna sama.

Gambar 2. Graf G2 dengan Tiga Buah Warna



Berdasarkan definisi pewarnaan simpul, kita bisa saja mewarnai graf pada gambar 2 dengan warna yang berbeda-beda untuk setiap simpulnya, sebab hasilnya masih memenuhi persyaratan dalam proses pewarnaan simpul. Hanya saja, dalam aplikasinya pewarnaan graf dengan menggunakan warna yang sesedikit mungkin sangat diperlukan. Jumlah warna minimum yang dibutuhkan oleh sebuah graf G sehingga seluruh simpulnya diwarnai tanpa ada sepasang simpul yang bertetangga berwarna sama disebut bilangan kromatik dari graf G dan dinotasikan dengan lambang $\chi(G)$. Graf pada gambar 2 memiliki bilangan kromatik 3, sehingga dinotasikan dengan $\chi(G2) = 3$.

2.3. Algoritma *Sequential Color*

Algoritma *Sequential Color* adalah sebuah algoritma untuk mewarnai sebuah graf dengan k-warna, di mana k adalah bilangan integer positif. Metoda yang digunakan algoritma ini adalah dengan pewarnaan langsung sebuah graf dengan warna yang sesedikit mungkin.

Berikut ini merupakan langkah-langkah dari algoritma *Sequential Color* :

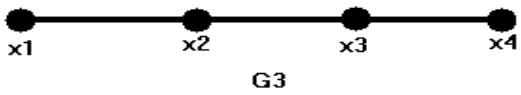
1. G = (V,E) adalah graf dengan jumlah simpul v buah, yaitu x1, x2, x3, …, xv. Kita misalkan warna – warna yang mungkin mewarnai simpul graf adalah : 1, 2, 3, …, v.
2. **For** i = 1 to v **do**
   Buat Li = <1, 2, 3, …, v> dengan Li adalah kumpulan warna yang mungkin menjadi warna dari xi.
3. **For** i = 1 to v **do**
   3.1. Warnai xi dengan Ci, di mana Ci adalah warna pertama pada list Li.
   3.2. **For** j = i to v **do**

        If ((xi,xj) ∈ E(G)) **then**

        Lj = Lj - Ci {Jika Ci anggota Lj, buang Ci dari Lj, sebab xj tidak boleh diwarnai dengan warna Ci (Ci telah menjadi warna xi yang bertetangga dengan xj)**}.**
4. Tiap simpul telah diberi warna dan jumlah warna yang digunakan dihitung.[2]

Berikut ini adalah salah satu cara penyelesaian masalah pewarnaan simpul graf dengan algoritma *sequential color*.

Gambar 3. Graf G3



Kita akan mewarnai graf G3 dengan menggunakan langkah-langkah pada algoritma *sequential color*.

Langkah 1 :
G3 = (E,V)
V = {x1, x2, x3, x4}
E = {(x1,x2), (x2,x3), (x3,x4)}

Langkah 2 & 3 :
Tabel 1. Langkah Pewarnaan Graf G3

Langkah	i	Li	Ci	J	Lj
2	1	<1>			
2	2	<1, 2>			
2	3	<1, 2, 3>			
2	4	<1, 2, 3, 4>			
3.1	1		1		
3.2	1			4	<2, 3, 4>
3.1	2		1		
3.2	2			3	<2,3>
3.1	3		2		
3.2	3			4	<3, 4>
3.1	4		3		

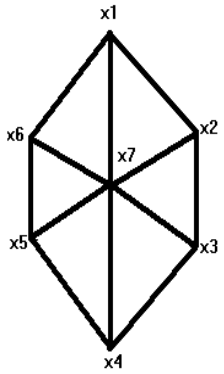
Langkah 4 :
x1 memperoleh warna 1, x2 memperoleh warna 1, x3, memperoleh warna 2, dan x4 memperoleh warna 3. Jumlah warna yang digunakan adalah 3.

Gambar 4. Graf G3 yang telah diberi warna

Pada contoh di atas telah kita lihat bahwa algoritma *Sequential Color* berhasil menetapkan warna-warna yang berbeda bagi simpul-simpul yang bertetangga. Sayangnya keberhasilan algoritma ini masih dipengaruhi oleh urutan pemberian nomor kepada

simpul-simpul. Hal ini bisa di lihat pada dua contoh berikut.

Gambar 5. Graf G4a



Penerapan algoritma *Sequential Color* pada graf G4a menghasilkan :

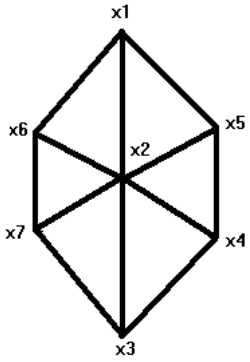
Tabel 2. Hasil *Sequential Color* pada Graf G4a

Langkah	i	Li	Ci	J	Lj
2	1	<1>			
2	2	<1, 2>			
2	3	<1, 2, 3>			
2	4	<1, 2, 3, 4>			
2	5	<1, 2, 3, 4, 5>			
2	6	<1, 2, 3, 4, 5, 6>			
2	7	<1, 2, 3, 4, 5, 6, 7>			
3.1	1		1		
3.2	1			2	<2>
				6	<2, 3, 4, 5, 6>
				7	<2, 3, 4, 5, 6, 7>
3.1	2		2		
3.2	2			3	<1, 3>
				7	<3, 4, 5, 6, 7>
3.1	3		1		
3.2	3			4	<2, 3, 4>
				7	<3, 4, 5, 6, 7>
3.1	4		2		
3.2	4			5	<1, 3, 4, 5>
				7	<3, 4, 5, 6, 7>
3.1	5		1		
3.2	5			6	<2, 3, 4, 5, 6>
				7	<3, 4, 5, 6, 7>
3.1	6		2		
3.2	6			7	<3, 4, 5, 6, 7>
3.1	7		3		

Dari hasil tabel 2 diketahui bahwa Graf G4a dapat diwarnai simpul-simpulnya tanpa ada simpul-simpul bertetangga yang berwarna sama dengan 3 warna.

Sekarang, perhatikanlah hasil dari algoritma *Sequential Color* yang dijalankan pada graf G4b, yang merupakan graf yang sama dengan graf G4a tetapi dengan urutan penomoran berbeda.

Gambar 6. Graf G4b



Tabel 3. Hasil Sequential Color pada Graf G4b

Langkah	i	Li	Ci	j	Lj
2	1	<1>			
2	2	<1, 2>			
2	3	<1, 2, 3>			
2	4	<1, 2, 3, 4>			
2	5	<1, 2, 3, 4, 5>			
2	6	<1, 2, 3, 4, 5, 6>			
2	7	<1, 2, 3, 4, 5, 6, 7>			
3.1	1		1		
3.2	1			2	<2>
				5	<2, 3, 4, 5>
				6	<2, 3, 4, 5, 6>
3.1	2		2		

3.2	2			3	<1, 3>
				4	<1, 3, 4>
				5	<3, 4, 5>
				6	<3, 4, 5, 6>
				7	<1, 3, 4, 5, 6, 7>
3.1	3		1		
3.2	3			4	<3, 4>
				7	<3, 4, 5, 6, 7>
3.1	4		3		
3.2	4			5	<4, 5>
3.1	5		4		
3.1	6		3		
3.2	6			7	<4, 5, 6, 7>
3.1	7		4		

Berdasarkan tabel 3 Graf G4b dapat diwarnai dengan tidak ada simpul bertetangga yang berwarna sama dengan 4 warna. Dari hal ini dapat dikatakan bahwa algoritma *Sequential Color* masih bergantung pada urutan penomoran dari simpul-simpul graf. Namun keuntungan dari algoritma *Sequential Color* adalah efisiensinya. Baik jumlah operasi di langkah 2 maupun langkah 3 dari algoritma ini menunjukkan bahwa *Sequential Color* adalah algoritma $O(V^2)$ dengan V adalah jumlah simpul.

2.4. Alokasi Memori

Marilah kita perhatikan algoritma penghitungan volume suatu balok berikut :

1. Masukkan panjang, lebar dan tinggi.
2. luas = panjang × lebar.
3. volume = luas × tinggi

Untuk memudahkan, kita misalkan memori dalam komputer adalah kumpulan lokasi yang di beri label 0, 1, 2, …, N. Dalam memori nilai dari peubah panjang, lebar, tinggi, luas dan volume dapat dimasukkan ke lokasi 0 sampai 4 secara berurutan. Namun dengan cara alokasi yang lebih efisien, nilai peubah volume dapat disimpan dalam lokasi 0 atau satu mengingat nilai peubah panjang dan lebar tidak digunakan lagi sejak langkah 3. Melihat contoh di atas, dapat dikatakan bahwa dua nilai peubah dapat menggunakan satu lokasi memori untuk menghemat pemakaian memori.

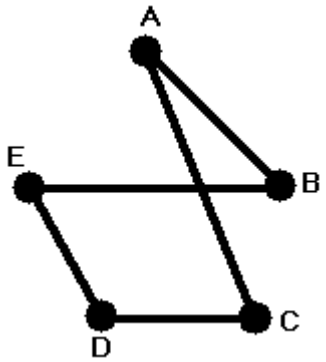
Dua buah peubah dikatakan tidak saling berinterferensi jika kedua peubah itu tidak diperlukan dalam waktu bersamaan saat program dijalankan, jika tidak kedua variabel itu dikatakan saling berinterferensi. Pada contoh di atas, peubah panjang/lebar dikatakan tidak berinterferensi dengan peubah volume.

Permasalahan alokasi memori ini dapat digambarkan sebagai sebuah graf, di mana peubah-peubah dianggap sebagai simpul, sementara dua buah simpul dihubungkan sebuah sisi jika kedua peubah itu berinterferensi. Graf yang dibentuk dengan cara ini disebut dengan graf interferensi.[2] Berikut ini contoh algoritma suatu program beserta graf interferensinya.

Algoritma 1 :

1. Masukkan A dan B
2. A = A × B
3. $C = A^2$
4. $D = C^2$
5. $E = D^2$
6. B = 3
7. E = E × B

Gambar 7. Graf Interferensi dari Algoritma 1



3. HASIL DAN PEMBAHASAN

Berikut ini merupakan contoh sederhana masalah efisiensi alokasi memori yang akan kita selesaikan dengan pewarnaan simpul graf.

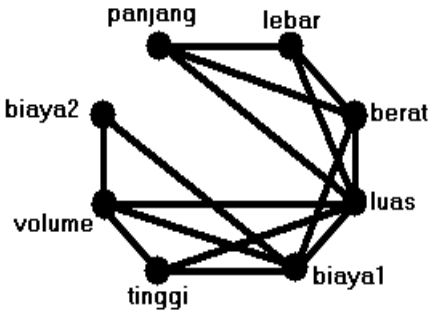
Kasus :

Sebuah kantor pos menawarkan dua pilihan kemungkinan biaya pengiriman bagi sebuah perusahaan. Pilihan pertama, biaya pengiriman adalah hasil kali luas dari amplop pengiriman dengan beratnya, sedangkan pilihan kedua, biaya pengiriman adalah dua kali volume amplop.

Pertama-tama kita tentukan algoritma penyelesaian masalah biaya pos:

1. Masukkan panjang, lebar dan berat amplop.
2. luas = panjang × lebar
3. biaya1 = luas × berat
4. Masukkan tinggi
5. volume = luas × tinggi
6. biaya2 = volume × berat
7. Tampilkan biaya1 dan biaya2

Gambar 8. Graf Interferensi Biaya Pos



Sekarang kita labeli tiap peubah dengan angka-angka :
Panjang = 1 ; lebar = 2 ; berat = 3 ; luas = 4 ; biaya1 = 5 ; tinggi = 6 ; volume = 7 ; biaya2 = 8.
Setelah itu kita jalankan algoritma *sequential color* pada permasalahan ini, hasilnya dapat dilihat di tabel berikut :

Tabel 4. Hasil algoritma *sequential color* pada permasalahan biaya pos

Langkah	i	Li	Ci	j	Lj
2	1	<1>			
2	2	<1, 2>			
2	3	<1, 2, 3>			
2	4	<1, 2, 3, 4>			
2	5	<1, 2, 3, 4, 5>			
2	6	<1, 2, 3, 4, 5, 6>			
2	7	<1, 2, 3, 4, 5, 6, 7>			
2	8	<1, 2, 3, 4, 5, 6, 7, 8>			
3.1	1		1		
3.2	1			2	<2>
				3	<2, 3>
				4	<2, 3, 4>
3.1	2		2		
3.2	2			3	<3>
				4	<3, 4>
3.1	3		3		
3.2	3			4	<4>
				5	<1, 2, 4, 5>
3.1	4		4		
3.2	4			5	<1, 2, 5>
				6	<1, 2, 3, 5, 6>
				7	<1, 2, 3, 5, 6, 7>
3.1	5		1		
3.2	5			6	<2, 3, 5, 6>
				7	<2, 3, 5, 6, 7>
				8	<2, 3, 4, 5, 6, 7, 8>
3.1	6		2		
3.2	6			7	<3, 5, 6, 7>
3.1	7		3		
3.2	7			8	<2, 4, 5, 6, 7, 8>
3.1	8		2		

Mengacu pada tabel di atas, jika kita misalkan memori komputer adalah tempat berlabel 0, 1, 2, …, N, maka persoalan biaya pos hanya memerlukan empat tempat dalam memori untuk seluruh 8 peubah yang digunakan.

4. KESIMPULAN

Pewarnaan simpul pada graf memiliki aplikasi yang penting dalam kehidupan. Walaupun algoritma *sequential search* yang dibahas pada makalah ini masih memiliki kelemahan, yaitu masih bergantung pada urutan penomoran simpul pada graf, tapi algoritma ini masih tergolong efisien.

Salah satu aplikasi pewarnaan simpul pada graf adalah dalam alokasi memori pada komputer, hal ini dapat dilakukan dengan menganggap peubah-peubah sebagai simpul dan sisi menyatakan berinterferensi atau tidaknya dua peubah. Graf interferensi yang telah terbentuk itu lalu diwarnai dengan jumlah warna menandakan ruang memori yang dibutuhkan.

DAFTAR REFERENSI

[1] M. Rinaldi, “Diktat Kuliah IF 2153 Matematika Diskrit”, Program Studi Teknik Informatika, 2006, Bandung, Indonesia.

[2] Joan P. Hutchinson, Professor of Mathematics and Computer Science Macalester College Homepage ~ Discrete Mathematics With Algorithms by Albertson and Hutchinson ~ File Nine : More Graph Theory.

http://www.macalester.edu/~hutchinson/book/book.html
Waktu akses 29 Desember 2007, pukul 21.39