APLIKASI ALGORITMA *GREEDY* PADA PERSOALAN PEWARNAAN GRAF

Fitriana Passa (13508036)

Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jalan Ganeca 10 Bandung
e-mail: if18036@students.if.itb.ac.id

ABSTRAK

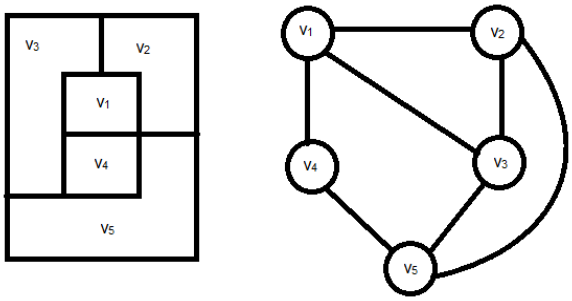
Salah satu permasalahan yang ada di bidang teori graf adalah mengenai pewarnaan graf. Pewarnaan graf M-warna merupakan permasalahan pewarnaan graf tidak berarah menggunakan paling banyak M-warna dengan syarat tidak ada simpul tetangga yang memiliki warna yang sama.
Algoritma *greedy* merupakan salah satu algoritma pemecahan masalah yang sering digunakan pada teori graf. Algoritma *greedy* memberikan solusi per langkah dimana setiap langkah yang diambil merupakan solusi optimal yang bisa didapatkan pada saat tersebut. Dengan pemilihan solusi optimal lokal, diharapkan akan mengarah kepada terbentuknya solusi optimal global.
Dalam kaitannya dengan pewarnaan graf, algoritma *greedy* dapat dijadikan sebagai alternatif solusi dalam mewarnai graf. Tidak semua persoalan pada pewarnaan graf yang diselesaikan menggunakan algoritma *greedy* menuju kepada hasil optimum global. Keberhasilan algoritma *greedy* dalam penanganan kasus pewarnaan graf juga bergantung pada pemilihan pengurutan simpul yang dipakai saat penelusuran terjadi. Akan tetapi, penggunaan algoritma *greedy* tetap membantu dalam menurangi pemakaian warna pada pewarnaan graf tidak berarah dan memberikan solusi sebuah optimum lokal yang merupakan solusi yang cukup efektif dibandingkan penggunaan algoritma *brute force* yang memakan waktu lama.

Kata kunci: *Greedy*, Pewarnaan graf.

1. PENDAHULUAN

Algoritma *greedy* adalah salah satu algoritma yang sering dipakai dalam cabang ilmu matematika dan komputer. Algoritma ini merupakan alternatif lain bagi penggunaan algoritma seperti *brute force*, backtracking, dynamic programming, dan sebagainya dalam pemecahan berbagai persoalan. Salah satu penerapan algoritma *greedy* dapat ditemukan pada topik graf. *Greedy* dapat digunakan sebagai alternatif dalam pencarian rute, penghitungan biaya terpendek pada lintasan graf, maupun dalam pewarnaan graf.

Pewarnaan graf memiliki fungsi yang beragam pada kehidupan nyata. Salah satu diantaranya adalah untuk pewarnaan peta menggunakan warna seminimal mungkin dengan batasan bahwa semua daerah yang bertetangga diberikan warna yang berbeda. Setiap daerah pada peta direpresentasikan dengan sebuah simpul. Jika sebuah daerah terhubung dengan suatu daerah yang lain, daerah tersebut akan dihubungkan dengan sebuah garis.



Gambar 1. Representasi graf dari sebuah peta

Selain itu, pewarnaan graf juga digunakan dalam alokasi register pada kompiler, yang diperkenalkan pada tahun 1981. Pattern matching, Sudoku puzzle, penentuan frekuensi untuk radio, juga dapat dipandang sebagai sebuah permasalahan pewarnaan graf.

2. PEWARNAAN GRAF

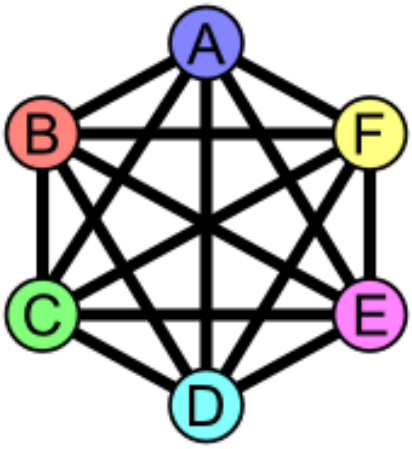
Sejarah pewarnaaan graf berhubungan dengan pewarnaan peta. Ketika itu, muncul sebuah postulat yang menyatakan bahwa empat warna berbeda cukup untuk mewarnai seluruh daerah di Inggris sedemikian sehingga tidak ada daerah yang berbatasan langsung menerima warna yang sama.

Pewarnaan graf telah dipelajari sebagai permasalahan algoritmik sejak tahun 1970. Permasalahan bilangan

kromatik pada pewarnaan graf merupakan salah satu masalah NP komplit.

Pewarnaan graf (*graph coloring*) merupakan permasalahan pewarnaan graf M-warna yang fokus pada pencarian seluruh jalan untuk mewarnai graf tidak berarah menggunakan paling banyak M warna sedemikian hingga tidak ada simpul tetangga yang memiliki warna yang sama (pewarnaan titik) atau tidak ada garis yang saling bertetangga yang memiliki warna yang sama (pewarnaan garis).

Dalam teori graf, pewarnaan graf merupakan kasus khusus pada pelabelan graf. Pelabelan tersebut dikaitkan dengan 'warna' yang menunjuk pada elemen pada graf yang memiliki konstrain tersendiri.



Gambar 2. Contoh pewarnaan simpul pada graf

Pewarnaan graf biasanya dikaitkan dengan pewarnaan pada simpul-simpulnya. Pewarnaan menggunakan paling banyak k warna disebut *k-coloring*. Jumlah paling sedikit warna yang dibutuhkan untuk mewarnai sebuah graf G disebut bilangan kromatik.

2. ALGORITMA *GREEDY*

Algoritma *greedy* merupakan metode yang paling populer untuk memecahkan persoalan mencari solusi optimum (optimasi). Terdapat 2 jenis persoalan optimasi: maksimasi dan minimasi.

Algoritma *greedy* membentu solusi langkah per langkah. Pada setiap langkah, dibuat pilihan optimum lokal untuk kemudian dicari solusi optimum global dari pilihan yang telah diambil. Secara garis besar, algoritma *greedy* terbagi menjadi 2 langkah besar:

1. Pilihan yang diambil merupakan pilihan terbaik yang dapat diperoleh pada saat itu tanpa menganalisis dampak yang akan terjadi dari pemilihan solusi terbaik saat ini.
2. Berharap bahwa dengan memilih pilihan yang terbaik pada saat itu (solusi optimum lokal), solusi terbaik secara global dapat dicapai (solusi optimum global).

Elemen-elemen pada algoritma *greedy*:

1. Himpunan kandidat, C
2. Himpunan solusi, S
3. Fungsi seleksi (*selection function*)
4. Fungsi kelayakan (*feasible*)
5. Fungsi objektif

Algoritma *greedy* melibatkan pencarian sebuah himpunan bagian S dari himpunan kandidat C. S harus memenuhi beberapa kriteria yang ditentukan, yaitu menyatakan suatu solusi dan S dioptimasi oleh fungsi objektif. Dengan cara yang sama, algoritma tersebut digunakan menyelesaikan persoalan pewarnaan graf.

Pewarnaan *greedy* merupakan pewarnaan simpul pada graf yang dibentuk berdasarkan algoritma *greedy*. Algoritma *greedy* mempertimbangkan simpul-simpul pada graf sebagai sebuah urutan dan mengisi setiap simpul dengan warna pertama yang tersedia.

3. PEMBAHASAN

Algoritma *greedy* mengatur simpul-simpul yang ada dengan pengaturan tertentu v_1, ..., v_n dan mengisi v_i dengan warna terkecil yang tersedia yang tidak digunakan oleh seluruh tetangga vi, diantara v1, ..., vi-1. Jika diperlukan, dapat ditambahkan warna baru pada simpul yang sedang diproses.

Pewarnaan *greedy* tidak selalu menghasilkan jumlah warna minimal. Kualitas dari warna-warna yang dipilih tergantung pada pengurutan yang dipilih. Terdapat pengurutan yang menghasilkan pewarnaan *greedy* dengan jumlah warna yang optimal. Di samping itu, algoritma pewarnaan *greedy* juga dapat memberikan hasil yang cukup buruk, contohnya pada crown graph.

Pada makalah ini, strategi pengurutan simpul pada graf didasarkan pada jumlah derajat pada masing-masing simpul. Algoritma *greedy* yang mempertimbangkan jumlah derajat tiap simpul dalam pemilihan solusinya mempunyai warna paling banyak sebesar derajat terbesar simpul + 1.

$$\max_i \min\{d(x_i) + 1, i\}$$

Perhitungan heuristik ini sering disebut sebagai *Welch Powell algorithm*.

Algoritma *greedy* untuk pewarnaan graf:

1. Pada graf G, cari derajat setiap simpul pada G
2. Inisialisasi himpunan simpul takberwarna dengan semua simpul pada graf G dengan urutan derajat tak menaik. Elemen pertama pada himpunan adalah simpul dengan derajat tertinggi.
3. Inisialisasi sebuah himpunan solusi dengan himpunan kosong.

4. Melakukan pemilihan simpul yang akan diisi warnanya dengan fungsi seleksi simpul pada himpunan simpul tak berwarna.
5. Menghapus simpul yang terpilih dari daftar simpul tak berwarna dan mengeset warna simpul terpilih dengan warna yang sekarang aktif.
6. Masukkan simpul dalam himpunan solusi
7. Memeriksa seluruh simpul yang ada di himpunan simpul yang tak berwarna:
   a. Simpul yang layak akan dimasukkan dalam himpunan solusi (tidak bertetangga dengan simpul yang telah ada di himpunan solusi)
   b. Simpul yang telah dinyatakan layak dihapuskan dari himpunan simpul tak berwarna
   c. Simpul yang telah dinyatakan layak diberi warna dengan warna yang sedang aktif.
8. Naikkan indeks warna aktif
9. Jika seluruh simpul sudah diwarnai, proses berakhir. Jika belum semua simpul terwarnai, kembali ke langkah 3.

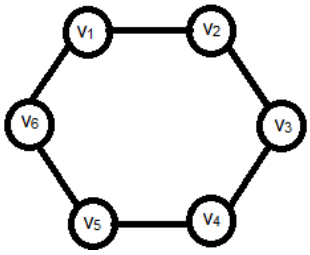
```
given G = (V,E):
ComputeAllDegree(v) for all v in V
Set uncolored = V sorted in decreasing
order of Degree(v)
Set currentColor = 0
setcolorWithCurrent = {}

while there are uncolored nodes:
   set A = first element of uncoloured
   remove A from uncolored
   set Color(A) = currentColor
   set colorWithCurrent = {A}

   for each v in uncolored :
       if v is not adjacent to anything
       in colorWithCurrent:
           set Color(v) = currentColor
           add v to colorWithCurrent
           remove v from uncolored
       end if
   end for
   currentColor = currentColor + 1
end while
```

4. PENGUJIAN

4.1



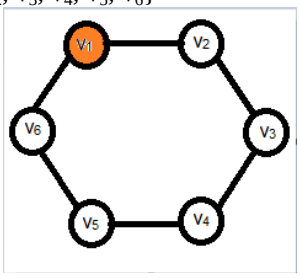
Gambar 3. Contoh kasus 1

Tabel 1. Pengurutan tidak menurun untuk simpul pada contoh soal 1 berdasarkan jumlah derajatnya

No	Simpul	Jumlah derajat
1	v_1	2
2	v_2	2
3	v_3	2
4	v_4	2
5	v_5	2
6	v_6	2

Tahap 0:
Uncolored : {v_1, v_2, v_3, v_4, v_5, v_6}

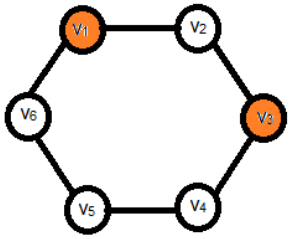
Tahap 1:
currentColor = 0
himpunan solusi tahap 1 : {}
A = v_1
Uncolored : {v_2, v_3, v_4, v_5, v_6}



Gambar 4. Kasus 1 Tahap 1a

himpunan solusi tahap 1 : {v_1}
Color(v_1) = 0

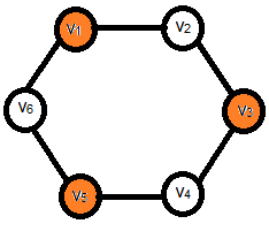
v = v_3
Uncolored : {v_2, v_4, v_5, v_6}



Gambar 5. Kasus 1 Tahap 1b

himpunan solusi tahap 1: {v_1, v_3}
Color(v_3) = 0

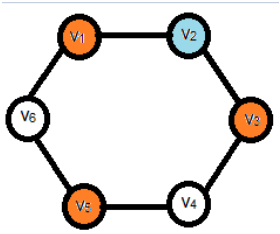
v = v_5
uncolored : {v_2, v_4, v_6}



Gambar 6. Kasus 1 Tahap 1c

himpunan solusi tahap 1: {v_1, v_3, v_5}
Color(v_5) = 0

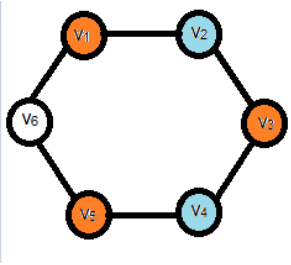
Tahap 2:
currentColor = 1
himpunan solusi tahap 2 : {}
A = v_2
uncolored : {v_4, v_6}



Gambar 7. Kasus 1 Tahap 2a

himpunan solusi tahap 2 : {v_2}
Color(v_2) = 1

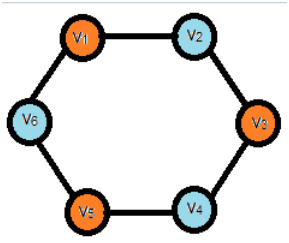
v = v_4
uncolored = {v_6}



Gambar 8. Kasus 1 Tahap 2b

himpunan solusi tahap 2 : {v_2, v_4}
Color(v_4) = 1

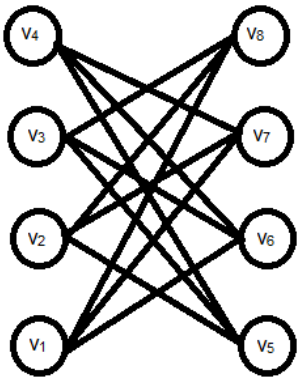
v = v_6
uncolored : {}



Gambar 9. Kasus 1 Tahap 2c

himpunan solusi tahap 2 : {v_2, v_4, v_6}
Color(v_6) = 1

4.2.



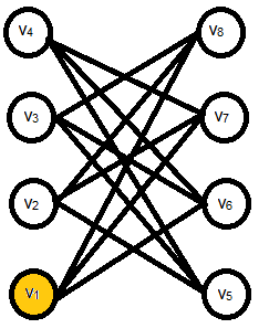
Gambar 10. Contoh Kasus 2

Tabel 2. Pengurutan tidak menurun untuk simpul pada contoh soal 2 berdasarkan jumlah derajatnya

No	Simpul	Jumlah derajat
1	v_1	3
2	v_2	3
3	v_3	3
4	v_4	3
5	v_5	3
6	v_6	3
7	v_7	3
8	v_8	3

Tahap 0:
uncolored : {v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8}

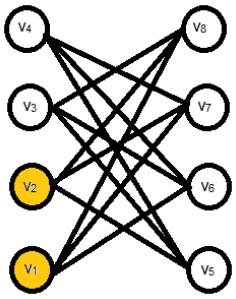
Tahap 1:
currentColor = 0
himpunan solusi tahap 1 : {}
A = v_1
Uncolored : {v_2, v_3, v_4, v_5, v_6, v_7, v_8}



Gambar 11. Kasus 2 Tahap 1a

himpunan solusi tahap 1: {v_1}
Color(v_1) = 0

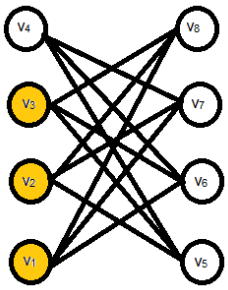
v = v_2
uncolored : {v_3, v_4, v_5, v_6, v_7, v_8}



Gambar 12. Kasus 2 Tahap 1b

himpunan solusi tahap 1 : {v_1, v_2}
Color(v_2) = 0

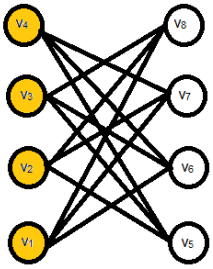
v = v_3
uncolored : { v_4, v_5, v_6, v_7, v_8}



Gambar 13. Kasus 2 Tahap 1c

himpunan solusi tahap 1 : {v_1, v_2, v_3}
Color(v_3) = 0

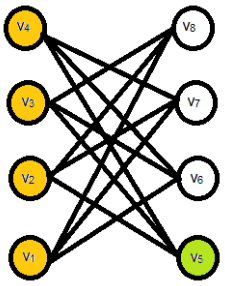
v = v_4
uncolored : {v_5, v_6, v_7, v_8}



Gambar 14. Kasus 2 Tahap 1d

himpunan solusi tahap 1 : {v_1, v_2, v_3, v_4}
Color(v_4) = 0

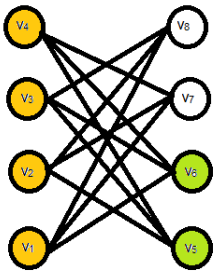
Tahap 2:
currentColor = 0
himpunan solusi tahap 2 : {}
A = v_5
Uncolored : {v_6, v_7, v_8}



Gambar 15. Kasus 2 Tahap 2a

himpunan solusi tahap 2: {v_5}
Color(v_5) = 1

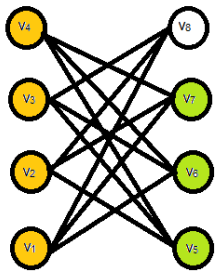
v = v_6
uncolored : {v_7, v_8}



Gambar 16. Kasus 2 Tahap 2b

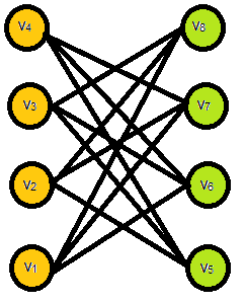
himpunan solusi tahap 2 : {v_5, v_6}
Color(v_6) = 1

v = v_7
uncolored : { v_8}



Gambar 17. Kasus 2 Tahap 2c

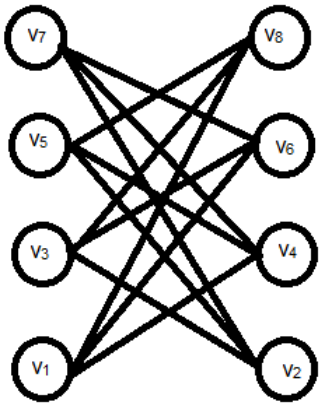
himpunan solusi tahap 2 : { v_5, v_6, v_7}
Color(v_7) = 1
v = v_8
uncolored : {}



Gambar 18. Kasus 2 Tahap 2d

himpunan solusi tahap 2 : { v_5, v_6, v_7, v_8}
Color(v_8) = 1

4.3



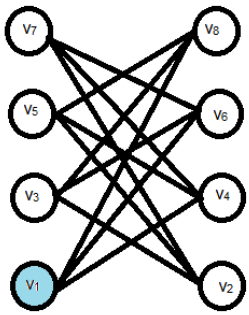
Gambar 19. Contoh Kasus 3

Tabel 3. Pengurutan tidak menurun untuk simpul pada contoh soal 3 berdasarkan jumlah derajatnya

No	Simpul	Jumlah derajat
1	v_1	3
2	v_2	3
3	v_3	3
4	v_4	3
5	v_5	3
6	v_6	3
7	v_7	3
8	v_8	3

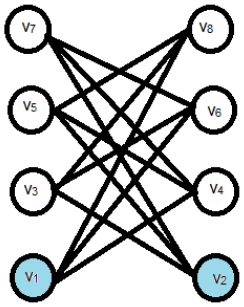
Tahap 0:
uncolored : {v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8}

Tahap 1:
currentColor = 0
himpunan solusi tahap 1 : {}
A = v_1
Uncolored : {v_2, v_3, v_4, v_5, v_6, v_7, v_8}



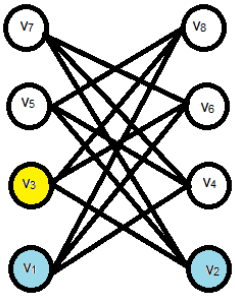
Gambar 20. Kasus 3 Tahap 1a
himpunan solusi tahap 1: {v_1}
Color(v_1) = 0

v = v_2
uncolored : {v_3, v_4, v_5, v_6, v_7, v_8}



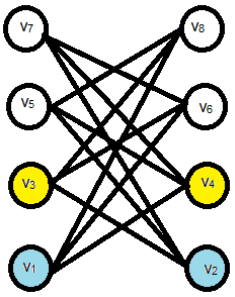
Gambar 21. Kasus 3 Tahap 1b
himpunan solusi tahap 1 : {v_1, v_2}
Color(v_2) = 0

Tahap 2:
currentColor = 0
himpunan solusi tahap 2 : {}
A = v_3
Uncolored : { v_4, v_5, v_6, v_7, v_8}



Gambar 22. Kasus 3 Tahap 2a
himpunan solusi tahap 2: {v_3}
Color(v_3) = 1

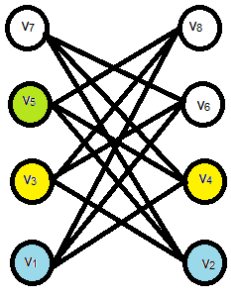
v = v_4
uncolored : { v_5, v_6, v_7, v_8}



Gambar 23. Kasus 3 Tahap 2b
himpunan solusi tahap 2 : {v_3, v_4}
Color(v_4) = 1

Tahap 3:
currentColor = 0
himpunan solusi tahap 3 : {}

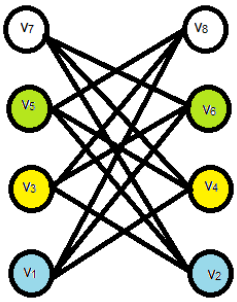
A = v_5
Uncolored : {v_6, v_7, v_8}



Gambar 24. Kasus 3 Tahap 3a

himpunan solusi tahap 3: {v_5}
Color(v_5) = 2

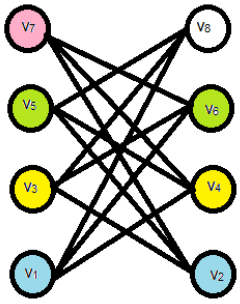
v = v_6
uncolored : {v_7, v_8}



Gambar 25. Kasus 3 Tahap 3b

himpunan solusi tahap 3 : {v_5, v_6}
Color(v_6) = 2

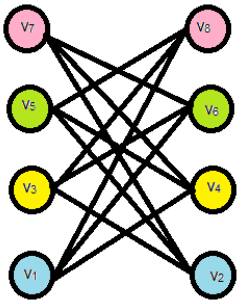
Tahap 4:
currentColor = 0
himpunan solusi tahap 4 : {}
A = v_7
Uncolored : { v_8}



Gambar 26. Kasus 4 Tahap 4a

himpunan solusi tahap 4: {v_7}
Color(v_7) = 3

v = v_8
uncolored : { }



Gambar 27. Kasus 4 Tahap 4b

himpunan solusi tahap 4 : {v_7, v_8}
Color(v_8) = 3

5. ANALISIS HASIL PENGUJIAN

Berdasarkan 3 contoh kasus di atas, dapat dianalisis bahwa pada kasus pertama dan kedua, algoritma *greedy* dapat memberikan hasil optimal. Pada kasus ke 3, algoritma *greedy* yang dipakai tidak dapat memberikan solusi optimal dan memberikan hasil sebanyak 4 warna (kasus terburuk, hasil = jumlah derajat terbanyak + 1). Contoh kasus 2 dan 3 berbeda dalam hal penomoran simpul yang digunakan pada simpul-simpul yang mempunyai derajat sama.

Dengan demikian, dapat dikatakan bahwa algoritma *greedy* pada kasus pewarnaan graf tidak selalu memberikan hasil yang optimal, tetapi jumlah warna yang dihasilkan tidak akan lebih besar dari jumlah derajat simpul maksimal + 1 pada simpul-simpul yang ada pada graf G. Keberhasilan algoritma *greedy* dalam permasalahan pewarnaan graf juga bergantung pada penomoran simpul yang mempunyai derajat yang sama.

6. KESIMPULAN

1. *Greedy coloring* merupakan salah satu algoritma pemecahan masalah pada kasus pewarnaan graf.
2. Tidak semua kasus yang dipecahkan oleh algoritma *greedy* memberikan hasil yang optimum, tetapi hasil yang ada tetap menjadi salah satu solusi optimum lokal. Pemilihan fungsi seleksi merupakan bagian yang sangat penting dalam menghasilan solusi optimal pada algoritma *greedy*
3. Keberhasilan algoritma *greedy* dalam penanganan kasus pewarnaan graf juga bergantung pada pemilihan pengurutan simpul yang dipakai saat penelusuran terjadi

REFERENSI

[1] Neapolitan, Richard E, Naimipour, Kumarss, "Foundation of Algorithms", D.C. Heath and Company, 1996.

[2] Ardiansyah, et al, "Implementasi Algoritma *Greedy* untuk Melakukan Graph *Coloring*: Studi Kasus Peta Propinsi Jawa Timur", Jurnal Informatika, Volume 4, Nomor 1, Tahun 2010, halaman 4.

[3] www.informatika.org/~rinaldi (diakses pada tanggal 7 Desember 2010)

[4] http://wapedia.mobi/en/Graph_*coloring*?t=5. (diakses pada tanggal 7 Desember 2010)

[5] www.worldlingo.com/ma/enwiki/en/Graph_*coloring* (diakses pada tanggal 7 Desember 2010)

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, 8 Desember 2010

Fitriana Passa
NIM 13508036