APLIKASI ALGORITMA GREEDY

PADA PERSOALAN PEWARNAAN GRAF

Fitria Indrianti (10112246)

Archena Harkafitra (10112264)

Shativa Sonrisa (10112281)

Program Studi Teknik Informatika, Fakultas Teknik dan Ilmu Komputer, Universitas Komputer Indonesia

Jalan Dipati Ukur No 10 Bandung

e-mail: a10112264@yahoo.com

Abstract

Pada paper ini akan membahas mengenai pewarnaan graph dengan menggunakan metode greedy Metode/Algoritma Greedy merupakan algoritma yang membentuk solusi langkah per langkah. Pada setiap langkah tersebut akan dipilih keputusan yang paling optimal. Keputusan tersebut tidak perlu memperhatikan keputusan selanjutnya yang akan diambil, dan keputusan tersebut tidak dapat diubah lagi pada langkah selanjutnya.

Prinsip utama algoritma greedy adalah “take what you can get now!”. Maksud dari prinsip tersebut adalah sebagai berikut: Pada setiap langkah dalam algoritma greedy, kita ambil keputusan yang paling optimal untuk langkah tersebut tanpa memperhatikan konsekuensi pada langkah selanjutnya. Kita namakan solusi tersebut dengan optimum lokal. Kemudian saat pengambilan nilai optimum lokal pada setiap langkah, diharapkan tercapai optimum global, yaitu tercapainya solusi optimum yang melibatkan keseluruhan langkah dari awal sampai akhir.

Jadi, metode greedy akan lebih efektif bila dibandingkan metode brute force dari faktor waktu, karena waktu yang dibutuhkan metode brute force, lebih lama dibandingkan dengan metode greedy.

1.Pendahuluan

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.

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

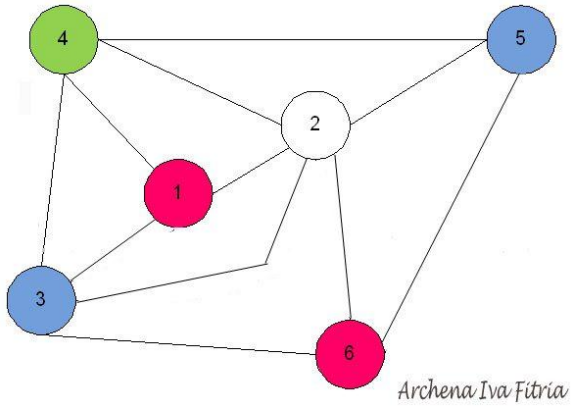
mengurangi pemakaian warna pada pewarnaan graf tidak berarah dan memberikan solusi sebuah optimum lokal yang merupakan solusi yang cukup efektif dibandingkan penggunaan algoritma *brute force* yang memakan waktu lama.

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 Makalah IF3051 Strategi Algoritma – Sem. I Tahun 2010/2011.

2 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.



3. Klasifikasi Pewarnaan Graf

Ada tiga macam pewarnaan graf, yaitu :

1. Pewarnaan garis merupakan pemberian warna pada setiap garis sehingga tidak ada 2 garis bertetangga yang memiliki warna yang sama.

2. Pewarnaan sisi merupakan pemberian warna pada setiap sisi pada graf sehingga sisi-sisi yang berhubungan tidak memiliki warna yang sama.

4.Pembahasan

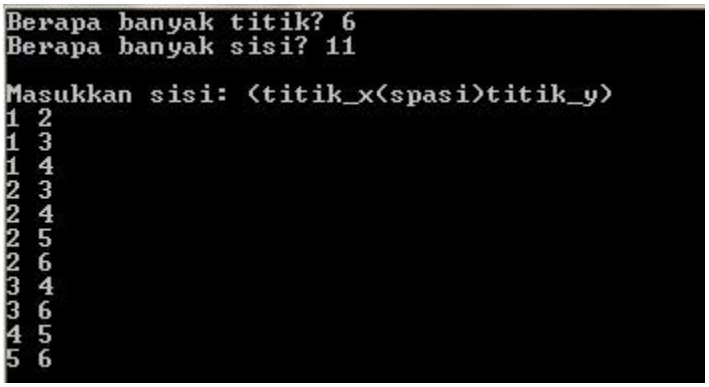
```
1   #include <cstdlib>
2   #include <iostream>
3
4   using namespace std;
5
6   bool **W;
7   int n=0;
8   int m=0;
9   int v=0;
10  int e=0;
11  int x=0;
12  int y=0;
```

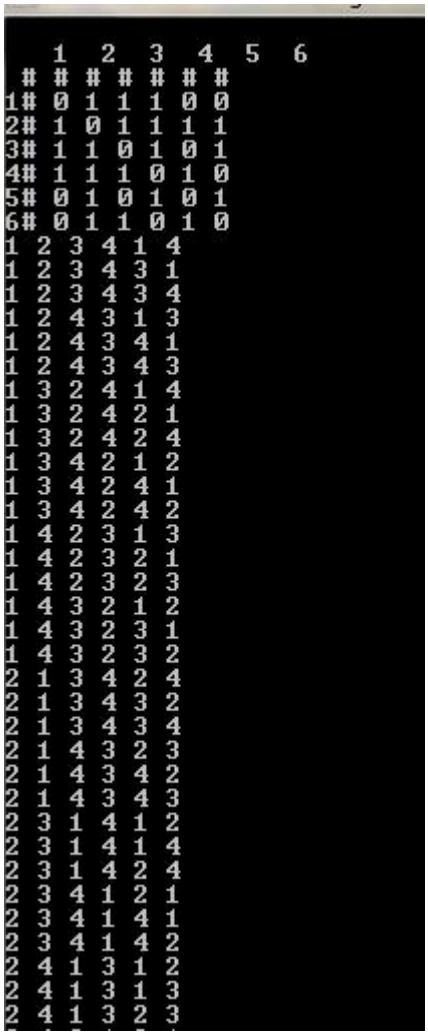
```
13  int *vcolor;
14
15  bool promising(int i)
16  {
17        int j=0;
18        while(++j<i)
19        {
20              if (W[i][j]&&vcolor[i]==vcolor[j])
21              return false;
22        }
23        return true;
24  }
25
26  void m_coloring (int i)
27  {
28        if(promising(i))
29        {
30              if(i+1==n)
31              {
32                    for (i=1;i<n;i++)
33                    cout<<vcolor[i]<<" ";
34                    cout<<endl;
35              }else
36              {
37                    for (int color = 1;color<=m;color++)
38                    {
39                          vcolor[i+1]= color;
40                          m_coloring(i+1);
41                    }
42              }
43        }
44  }
45
46  void initArrays()
47  {
48        for(int i=0;i<n;i++)
49        {
50              W[i]=new bool[n];
51              vcolor[i]=0;
52        }
53  }
54
55  void fillW()
56  {
57        for (int i=0;i<n;i++)
58        {
59              for (int j=0;j<n;j++)
60              {
61                    W[i][j]=false;
62              }
63        }
```

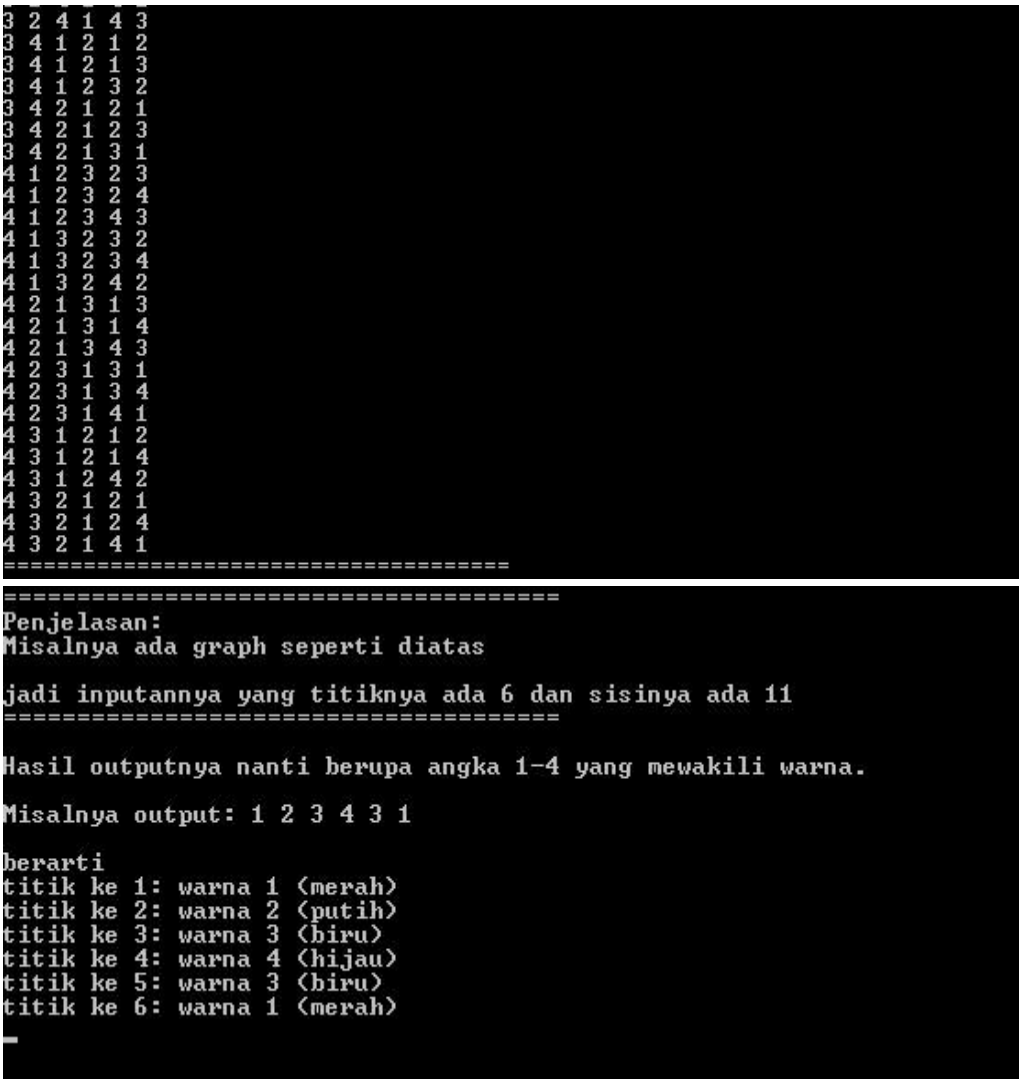
```
64  }
65
66  void askForEdges()
67  {
68        cout<<"Berapa banyak sisi? ";
69        cin>>e;
70        cout<<endl<<"Masukkan sisi: (titik_x(spasi)titik_y)"<<endl;
71
72        for (int i=0;i<e;i++)
73        {
74              cin>>x>>y;
75              W[x][y]=true;
76              W[y][x]=true;
77        }
78  }
79
80  void specialMatrixPrint()
81  {
82        cout<<endl;
83        for(int i=0;i<n;i++)
84        {
85              for(int j=0;j<n;j++)
86              {
87                    cout<<W[i][j]<<" ";
88              }
89              cout<<endl;
90        }
91  }
92
93  void showEdgesMatrix()
94  {
95        int i;
96        cout<<"\n ";for(i=1;i<n;i++)
97        {
98              cout<<"  "<<i;
99        }
100       cout<<"\n ";for(i=1;i<=n;i++)
101       {
102             cout<<"# ";
103       }
104       cout<<endl;
105
106       for(i=1;i<n;i++)
107       {
108             cout<<i<<"# ";
109             for(int j=1;j<n;j++)
110             {
111                   cout<<(W[i][j]?"1 ":"0 ");
112             }
113             cout<<endl;
114       }
```

```
}

void checkFor(int i)
{
      m=i;
      m_coloring(0);
      cout<<"====================================="<<endl;
      cout<<"Penjelasan: "<<endl;
      cout<<"Misalnya ada graph seperti diatas"<<endl<<endl;
      cout<<"jadi inputannya yang titiknya ada 6 dan sisinya ada
11"<<endl;
      cout<<"====================================="<<endl<<endl;
      cout<<"Hasil outputnya nanti berupa angka 1-4 yang mewakili
warna."<<endl<<endl;
      cout<<"Misalnya output: 1 2 3 4 3 1"<<endl<<endl;
      cout<<"berarti"<<endl;
      cout<<"titik ke 1: warna 1 (merah)"<<endl;
      cout<<"titik ke 2: warna 2 (putih)"<<endl;
      cout<<"titik ke 3: warna 3 (biru)"<<endl;
      cout<<"titik ke 4: warna 4 (hijau)"<<endl;
      cout<<"titik ke 5: warna 3 (biru)"<<endl;
      cout<<"titik ke 6: warna 1 (merah)"<<endl;

}

int main(int argc, char *argv[])
{
      cout<<"Berapa banyak titik? ";
      cin>>n;

      n+=1;
      W=new bool *[n];
      vcolor=new int[n];

      initArrays();
      fillW();
      askForEdges();
      showEdgesMatrix();

      checkFor(4);//warna yang digunakan

      cin>>y;
      //return 0;


      system("pause");
      return 0;
}
```

5.Pengujian


Berapa banyak titik? 6
Berapa banyak sisi? 11
Masukkan sisi: (titik_x(spasi)titik_y)
1 2
1 3
1 4
2 3
2 4
2 5
2 6
3 4
3 6
4 5
5 6


1 2 3 4 5 6
# # # # # #
1# 0 1 1 1 0 0
2# 1 0 1 1 1 1
3# 1 1 0 1 0 1
4# 1 1 1 0 1 0
5# 0 1 0 1 0 1
6# 0 1 1 0 1 0
1 2 3 4 1 4
1 2 3 4 3 1
1 2 3 4 3 4
1 2 4 3 1 3
1 2 4 3 4 1
1 2 4 3 4 3
1 3 2 4 1 4
1 3 2 4 2 1
1 3 2 4 2 4
1 3 4 2 1 2
1 3 4 2 4 1
1 3 4 2 4 2
1 4 2 3 1 3
1 4 2 3 2 1
1 4 2 3 2 3
1 4 3 2 1 2
1 4 3 2 3 1
1 4 3 2 3 2
2 1 3 4 2 4
2 1 3 4 3 2
2 1 3 4 3 4
2 1 4 3 2 3
2 1 4 3 4 2
2 1 4 3 4 3
2 3 1 4 1 2
2 3 1 4 1 4
2 3 1 4 2 4
2 3 4 1 2 1
2 3 4 1 4 1
2 3 4 1 4 2
2 4 1 3 1 2
2 4 1 3 1 3
2 4 1 3 2 3



6. Analisis Hasil Pengujian

Berdasarkan contoh kasus di atas, dapat dianalisis bahwa algortima greedy tidak selalu memberikan hasil yang maksimal dalam pewarnaan graf. Itu disebabkan karena keberhasilan algoritma greedy dalam contoh kasus di atas bergantung dari penomoran titik dan jumlah sisi.

7. Kesimpulan

Pewarna greedy adalah mewarnai simpul dari graf yang dibentuk oleh algoritma greedy yang menganggap simpul dari grafik secara berurutan dan memberikan tiap titik warna yang tersedia pertama. Pewarna Greedy tidak digunakan secara umum jumlah minimum warna yang mungkin, namun mereka telah digunakan dalam matematika sebagai suatu teknik untuk membuktikan hasil lainnya tentang pewarna dan dalam ilmu komputer sebagai heuristik untuk menemukan pewarna dengan beberapa warna.