Penerapan Algoritma Greedy untuk Memecahkan Masalah Pohon Merentang Minimum

Bramianha Adiwazsha - NIM: 13507106
Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganeca 10 Bandung
E-mail : if17106@students.if.itb.ac.id

Abstract – *Makalah ini membahas penerapan prinsip algoritma greedy menemukan solusi optimum dalam pemecahan masalah pembentukan pohonmerentang minimum pada graf. Walaupun pada beberapa persoalan algoritma greedy gagal menemukan solusi optimum, namun metode ini tetap yang paling populer dalam persoalan optimasi dengan membentuk solusi langkah per langkah (step by step). Dan Algoritma yang termasuk dalam algoritma greedy yang dapat memecahkan masalah minimum spanning tree adalah algoritma Prim dan Kruskal. Maka akan dilihat pemecahan masalah ini dengan kedua metode yang ada.*

Kata Kunci*: graf, greedy, pohon merentang minimum, Prim ,Kruskal .*

1. PENDAHULUAN

Agoritma greedy adalah algoritma yang memecahkan masalah langkah per langkah, pada setiap langkah membuat pilihan optimum (local optimum) pada setiap langkah dengan harapan bahwa langkah berikutnya mengarah ke solusi optimum global (*global optimum*).

Algoritma greedy membuat keputusan berdasarkan pilihan yang ada sekarang, tidak melihat pilihan yang akan muncul kemudian. Karena itulah algoritma greedy dikategorikan dalam algoritma yang 'berpandangan pendek' dan tidak dapat diulang karena keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya. Padahal dalam permasalahan optimasi terdapat banyak pilihan yang perlu dieksplorasi pada setiap langkah solusi. Terkadang algoritma greedy mengambil keputusan yang diambil terlalu dini tanpa melihat yang akan ditemui berikutnya sehingga menimbulkan apa yang disebut "*good next move, bad overall move*".

Melihat kelemahan yang dimiliki, solusi optimum global yang didapatkan belum tentu merupakan solusi optimum (terbaik), tetapi *sub-optimum* atau *pseudo-optimum*. Karena algoritma *greedy* tidak beroperasi secara menyeluruh terhadap semua alternative solusi yang ada.

Namun begitu algoritma ini tetap adalah pilihan utama untuk permasalahan yang sederhana karena ini adalah metode yang paling cepat dibanding metode yang lain dan dapat memberikan solusi hampiran atau aproksimasi terhadap nilai optimum yang diinginkan serta hasil yang diberikan masih merupakan solusi yang layak (feasible solution).

Algoritma yang termasuk ke dalam tipe algoritma greedy antara lain kode Huffman,algoritma Dijkstra, algoritma Prim, dan algoritma Kruskal yang ketiganya digunakan dalam menyelesaikan permasalahan optimasi pada graf.

Kali ini pembahasan akan ditik beratkan pada pemecahan masalah pembentukan pohon merentang minimum. Pohon merentang minimum sangat banyak aplikasinya dalam kehidupan sehari-hari.

Dan dari algoritma greedy yang ada yang dapat memecahkan masalah ini adalah Agoritma Prim dan Algoritma Kruskal. Kedua algoritma ini memiliki karakter tersendiri dan akan kita lihat dalam pembahasan berikut ini.

2. PEMBAHASAN

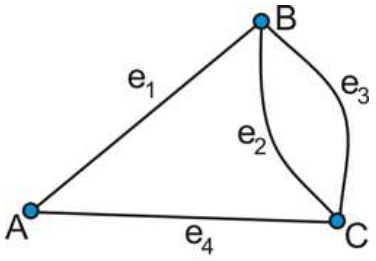
2.1. Teori Graf dan Pohon

2.1.1 Graf

Graf (*Graph*) didefinisikan sebagai:

$G = \{V, E\}$

Dalam hal ini, V merupakan himpunan tidak kosong dari simpul-simpul (vertices / node) digambarkan dalam titik-titik, dan E adalah himpunan sisi-sisi (edges / arcs) digambarkan dalam garis-garis yang menghubungkan sepasang simpul.
Dapat dikatakan graf adalah kumpulan dari simpul-simpul yang dihubungkan oleh sisi-sisi.



Gambar 1: Graf G

Pada G diatas, graf terdiri dari himpunan V dan E yaitu:

$V = \{A, B, C\}$

$E = \{e_1, e_2, e_3, e_4\}$

$= \{(A,B),(B,C),(B,C),(A,C)\}$

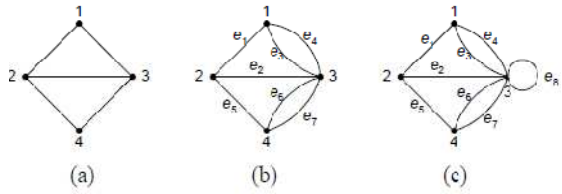
Pengelompokan graf dapat dipandang berdasarkan ada tidaknya sisi ganda atau sisi kalang, berdasarkan jumlah simpul, atau berdasarkan orientasi arah pada sisi.

1. Graf sederhana

   Graf yang tidak mengandung gelang maupun sisi-ganda dinamakan graf sederhana.

2. Graf tak sederhana

   Graf yang mengandung sisi ganda atau gelang dinamakan graf tak-sederhana (*unsimple graph*). Ada dua macam graf tak-sederhana, yaitu graf ganda (*multigraph*) dan graf semu (*pseudograph*). Graf ganda adalah graf yang mengandung sisi ganda. Graf semu adalah graf yang mengandung gelang



Gambar 2 : Graf berdasarkan ada tidaknya sisi gelang atau sisi ganda, (a). Graf sederhana, (b). Graf ganda, (c). Graf semu

3. Graf berhingga

   Graf berhingga adalah graf yang jumlah simpulnya, *n*, berhingga.

4. Graf tak berhingga

   Graf yang jumlah simpulnya n, tidak berhingga banyaknya disebut graf takberhingga.

5. Graf berarah

   Graf yang setiap sisinya diberikan orientasi arah disebut sebagai graf berarah. Kita lebih suka menyebutsisi berarah dengan sebutan busur (*arc*).

6. Graf tak berarah

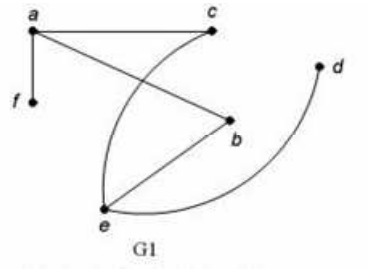
   Graf yang sisinya tidak mempunyai orientasi arah disebut graf tak-berarah. Pada graf tak-berarah, urutan pasangan simpul yang dihubungkan oleh sisi tidak diperhatikan.

7. Graf Berbobot (*Weight Graf*)

   Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga.

2.1.2 Pohon

Pohon adalah graf tak-berarah terhubung dan tidak mengandung sirkuit.



Gambar 3: Pohon G1

Sifat yang terpenting pada pohon adalah terhubung dan tidak mengandung sirkuit. Pohon dinotasikan sama dengan T = (V,E)

Keterangan :

T : Tree

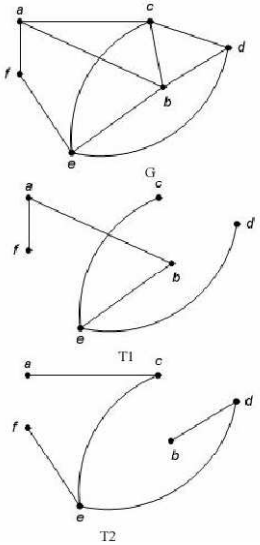
V : Vertices atau node atau vertex atau simpul, V merupakan himpunan tidak kosong. V = {v1,v2,…,vn}

E : Edges atau Arcs atau sisi yang menghubungkan simpul E = {e1,e2,..,en}

2.1.3Pohon merentang

- Pohon merentang dari graf terhubung adalah upagraf merentang yang berupa pohon.
- Pohon merentang diperoleh dengan memutus sirkuit di dalam graf.
- Setiap graf terhubung mempunyai paling sedikit Satu buah pohon merentang.
- Graf tak-terhubung dengan *k* komponen mempunyai *k* buah hutan merentang yang disebut hutan merentang (*spanning forest*).

Contoh pembentukan pohon merentang:



Gambar 4: Pembentukan pohon merentang

Keterangan :
T1 dan T2 merupakan pohon merentang dari graf G
Pohon merentang T1 dibentuk dengan cara menghapus sisi {(a,c), (b,c), (b,d), (c,d), (e,f)} dari graf G.
Pohon merentang T1 dibentuk dengan cara menghapus sisi {(a,f), (a,b), (b,c) (b,e), (c,d)} dari grafG.

2.14. Pohon Merentang Minimun

Jika G pada merupakan graf berbobot, maka bobot pohon merentang T1 atau T2 didefinisikan sebagai jumlah bobot semua sisi di T1atau T2. Diantara pohon merentang yang ada padaG, yang paling penting adalah pohon merentang dengan bobot minimum.

Pohon merentang dengan bobot minimum ini disebut dengan pohon merentang minimum atau *Minimum Spanning Tree (MST).*

Contoh aplikasi MST yang sering digunakan adalahpemodelan proyek pembangunan jalan rayamenggunakan graf.

MST digunakan untuk memilihn jalur dengan bobot terkecil yang akan meminimalkan biaya pembangunan jalan.

2.2.Algoritma Greedy untuk Pohon merentang minimum

Terdapat dua macam algoritma tipe greedy yang dapat memenuhi pemecahan masalah pohon merentang ini. Yaitu algoritma Prim dan algoritma kruskal.

2.2.1.Algoritma Prim

Algoritma Prim membentuk pohon merentang minimum dengan langkah per langkah. Pada setiap langkah kita mengambil sisi graf G yang memiliki bobot minimum namun yang terhubung dengan pohon merentang T yang telah terbentuk.

Algoritma Prim

1. Ambil sisi dari graf G yang berbobot minimum, masukan ke dalam T.
2. Pilih sisi (u, v) yang memiliki bobot minimum dan bersisian dengan dengan simpul di T. Tetapi (u, v) tidak membentuk sirkuit di T. Tambahkan (u, v) ke dalam T.
3. Ulangi langkah 2 sebanyak (n-2) kali.

Pseudo-code algoritma Prim adalah sebagai berikut:

```
procedure Prim (input G : graf, output T :
pohon)
{
1.Membentuk pohon merentang minimum T dari
  graf terhubung G.
2.Masukan: graf-berbobot terhubung G =(V, E),
  yang mana |V|= n
3.Keluaran: pohon rentang minimum T = (V, E')
}
Deklarasi
       i, p, q, u, v : integer
Algoritma
Cari sisi (p,q) dari E yang berbobot terkecil
T ← {(p,q)}
for i←1 to n-2 do
       Pilih sisi (u,v) dari E yang bobotnya
       terkecil namun bersisian dengan suatu
       simpul di dalam T.
       T←T u {(u,v)}
endfor
```

2.2.2Algoritma Kruskal

Algoritma Kruskal juga termasuk algoritma yang dapat digunakan untuk mencari pohon merentang minimum. Pada Algoritma Kruskal sisi graf diurut terlebih dahulu berdasarkan bobotnya dari kecil membesar. Dan cara pemasukan sisi dalam T tidak perlu bersisian todak seperti pada algoritma Prim.

Algoritmanya:

1. Himpunan sisi dari G diurutkan membesar sesuai bobot sisi tersebut.
2. Buat T dengan memasukkan 1 sisi terpendek dari G tersebut.
3. Ulang (banyak sisi T = (banyak simpul G) -1)
   a. Ambil sisi selanjutnya dari G.
   b. Jika sisi itu tidak membuat sirkuit di T

- Masukkan sisi itu ke T.
- Masukkan simpul-simpul sisi itu ke T.

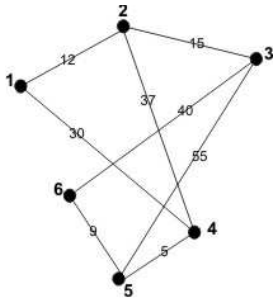
```
procedure Kruskal (input G: graf, output T:
pohon)
{Membentuk pohon merentang minimum T dari
graf terhubung G}
Deklarasi:
i, p, q, u, v: integer
Algoritma:
{Asumsi: sisi-sisi graf sudah diurut menaik
berdasarkan bobotnya}
T <- {}
while jumlah sisi T < n-1 do
   Pilih sisi dari E yang bobotnya terkecil
   if (u, v) tidak membentuk siklus di T then
      T <- T U {(u, v)}
   endif
endwhile
```

2.3.Aplikasi Pembentukan Pohon Merentang

Sekarang akan kita membentuk sebuah pohon merentang menggunakan algoritma greedy. Karena ada dua algoritma greedy yang bisa digunakan untuk membentuk pohon merentang minimum yaitu algoritma prim dan kruskal, maka akan kita lihat perbedaannya pada kasus berikut.

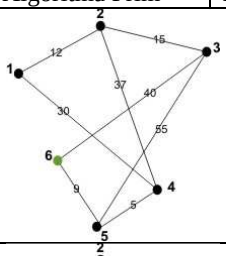
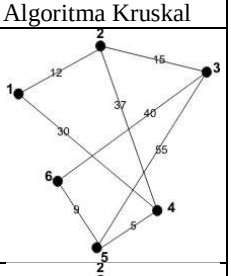
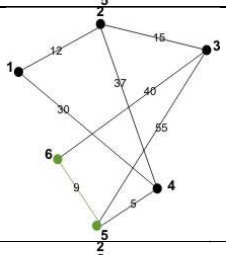
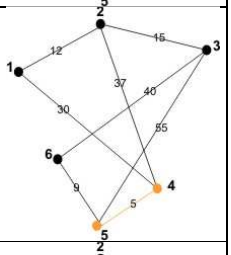
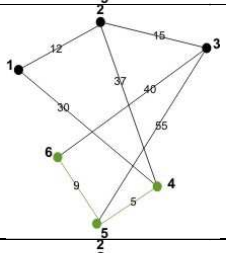
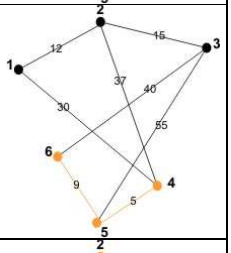
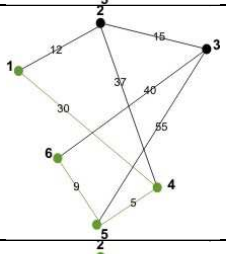
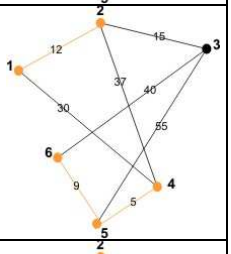
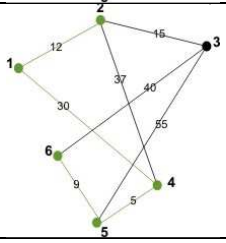
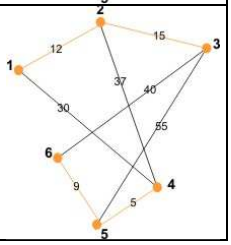
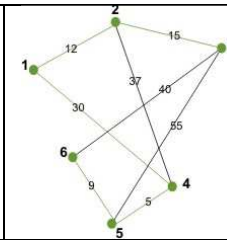
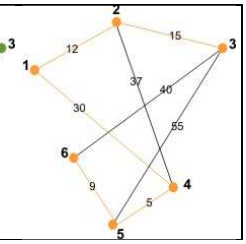
Sebenarnya pada kasus yang berbeda hasil yang ditunjukkan oleh kedua algoritma yang ada juga akan memberikan jumlah langkah yang berbeda walaupun

hasil akhir pohon merentangnya akan tetap sama.



Gambar 5: graf X yang akan dicari pohon merentangnya

Tabel 1. Langkah Pembentukan MST (gambar)

langkah	Algoritma Prim	Algoritma Kruskal
0		
1		
2		
3		
4		
5		

Kemudian dapat kita lihat urutan kerja pada tiap langkah untuk masing-masing algoritma pembentukan pohon merentang minimum ini.

Tabel 2. Langkah Pembentukan MST (urutan)

Langkah	Prim		Kruskal	
	Simpul	Sisi terbentuk	Sisi	Simpul masuk
0	6	-	-	-
1	5	{(6,5)}	(5,4)	{4,5}
2	4	{(6,5), (5,4)}	(5,6)	{4,5,6}
3	1	{(6,5), (5,4), (4,1), }	(1,2)	{1,2,4,5,6}
4	2	{(6,5), (5,4), (4,1), (1,2)}	(2,3)	{1,2,3,4,5,6}
5	3	{(6,5), (5,4), (4,1), (1,2), (2,3)}	(1,4)	{1,2,3,4,5,6}

3. HASIL DAN PEMBAHASAN

Dari tabel terlihat bahwa secara umum, algoritma Kruskal bisa berjalan hampir sama dibanding algoritma Prim.
Namun jumlah simpul dan jumlah sisi dalam graf juga menentukan algoritma apa yang sebaiknya dipakai karena tiap algoritma memiliki kekuatan tersendiri sesuai dengan spesifikasinya.

Saat graf memiliki sisi berjumlah sedikit namun memiliki sangat banyak simpul, Kruskal akan lebih cocok diterapkan karena orientasi kerja algoritma ini adalah berdasarkan pada urutan bobot sisi, tidak berdasarkan simpul.

Sebaliknya untuk graf lengkap atau yang mendekati lengkap, dimana setiap simpul terhubungkan dengan semua simpul yang lain Prim menunjukkan perbedaan yang cukup besar. Dibandingkan dengan algoritma Kruskal.

Sebenarnya hal ini sudah terlihat dari tahap-tahap

algoritma itu sendiri. Prim lebih berorientasi kepada pencarian simpul sedangkan Kruskal pada pencariam sisi, dimana sisi-sisi tersebut harus diurutkan dan ini memakan waktu yang tidak sedikit. Apalagi bila algoritma pengurutan yang digunakan juga tidak mangkus.

Sehingga pada pengujian kali ini masih belum bias terlihat mana algoritma yang lebih unggul untuk kasus graf X diatas. Karena jumlah simpul dan sisi dalam graf tersebut masih relative sedikit. Dimana kita tahu algoritma dengan kompleksitas yang berbeda baru akan terlihat bedanya pada jumlah input yang besar.

4. KESIMPULAN

Permasalahan membentuk pohon merentang minimum dari sebuah graf dapat diselesaikan dengan menggunakan algoritma greedy (algoritma Prim dan algoritma kruskal) yang mengupayakan pengambilan pilihan optimum pada setiap langkah dengan harapan akan mendapatkan hasil optimum global pada akhirnya.

Algoritma Prim dan Kruskal memiliki perbedaan dalam jumlah langkah yang harus diambil, cara penentuan langkah dan aturan dalam pembentukan pohon merentang minimum.

Algoritma Kruskal menitikberatkan pemilihan sisi berdasarkan urutan bobotnya. Dan karena terlebih dahulu harus mengurutkan sisi berdasarkan bobotnya, algoritma ini lebih cocok untuk pohon dengan jumlah simpul sedikit kurang disarankan untuk pohon dengan jumlah simpul yang banyak.
Jumlah langkah yang harus diambil adalah sebanyak n-1 kali, (n=jumlah simpul).

Algoritma Prim menitikberatkan pada pemilihan bobot minimum berdasarkan simpul yang diambil. Dan karena tidak perlu mengurutkan terlebih dahulu Algoritma prim cocok untuk pohon dengan jumlah simpul banyak.
Jumlah langkah yang harus diambil adalah sebanyak n kali, (n=jumlah simpul).

DAFTAR REFERENSI

[1] Munir, Rinaldi. (2008). Diktat Kuliah IF2091 Struktur Diskrit Edisi Keempat. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung. \

[2] Wikipedia. (2008)
http://en.wikipedia.org/wiki/Greedy_algorithm
Tanggal akses: 3 Januari 2008 pukul 20:30 WIB

[3] Wikipedia. (2008)
http://en.wikipedia.org/wiki/Greedy_algorithm
Tanggal akses: 3 Januari 2008 pukul 21:10 WIB

[4] Wikipedia. (2008)
http://id.wikipedia.org/wiki/Algoritma_Prim
Tanggal akses: 4 Januari 2008 pukul 20.:15 WIB

[5] NIST. (2008)
http://www.nist.gov/dads/HTML/primJarnik
Tanggal akses: 5 Januari 2008 pukul 19.00 WIB