Majalah Ilmiah Matematika dan Statistika Volume 15 Nomor 2 2015, 49 – 60
https://jurnal.unej.ac.id/index.php/MIMS/index ISSN 1411-6669

PENENTUAN SEMUA MINIMUM SPANNING TREE PADA GRAF TERHUBUNG BERBOBOT

(Determination of All Minimum Spanning Tree on Weighted Connected Graph)

Tri Hartati, Firdaus Ubaidillah, Ahmad Kamsyakawuni
Jurusan Matematika FMIPA Universitas Jember
Jl. Kalimantan 37, Jember 68121, Indonesia
Email: firdaus_u@yahoo.com, kamsyakawuni.fmipa@unej.ac.id

Abstract. Minimum Spanning Tree is spanning tree with minimum weigth. There are algorithms was used to get MST, include Kruskal's Algorithm and Prim's Algorithm. But all of that just give one solution, although MST problems have many solution. The ways to get all of MST from a simple weighted connected graph G can do by: the first, decide reference MST, is one of MST from G. Then choose all edges of graph G which use in MST. Futhermore decide equivalen class from every edges in graph G is not contained in reference MST with less weight or equal maximum weight in reference MST and decide choosen subset. Combinating choosen subset so getting all of MST from random simple weighted connected graph. There are two special criteria in decision all MST of simple weighted connected graph, are decision all MST with combine choosen edge is not make cycle and decision all MST with combine choosen edge make cycle. The purposes are counting and getting all MST which formed by a random simple weighted connected graph.

Keywords: Minimum Spanning Tree, Kruskal's Algorithm
MSC 2020: 05C05

1. Pendahuluan

Teori graf pertama kalinya diperkenalkan pada tahun 1736 oleh Leonard Euler, seorang matematikawan Swiss, dalam upaya penyelesaian masalah jembatan Konigsberg. Dari permasalahan itulah akhirnya Leonard Euler mengembangkan beberapa konsep mengenai teori graf. Salah satu diantaranya adalah konsep pohon (*tree*) [1]. Konsep pohon merupakan konsep yang penting dan popular karena konsep ini mampu mendukung pemecahan masalah dalam berbagai terapan graf [3].

Di dalam konsep pohon, terdapat banyak jenis pohon graf. Salah satunya adalah pohon merentang atau lebih dikenal dengan *Spanning Tree*. *Spanning tree* dapat diterapkan pada persoalan yang mengandung unsur pencarian bobot minimum. *Spanning tree* dengan bobot minimum ini dinamakan *Minimum Spanning Tree*, yang digunakan sebagai cara untuk mencari solusi dari masalah dalam dunia terapan graf [3].

Ada beberapa algoritma yang dapat digunakan untuk mendapatkan *minimum spanning tree*, antara lain algoritma *Kruskal* dan algoritma *Prim*. Algoritma *Kruskal* merupakan algoritma yang paling dikenal dari semua algoritma sejenis. Tetapi algoritma *Kruskal* dan *Prim* hanya

memberikan solusi tunggal, meskipun persoalan *minimum spanning tree* yang dihadapi mempunyai solusi tidak tunggal [3]. Dari uraian tersebut menarik untuk dikaji bagaimana cara menentukan semua *minimum spanning tree* pada graf dengan solusi tidak tunggal. Permasalahan yang dibahas dalam skripsi ini adalah menentukan semua *Minimum Spanning Tree* yang dapat dibentuk dari sembarang graf terhubung sederhana berbobot. Tujuan dari penulisan ini adalah menghitung dan mendapatkan semua *Minimum Spanning Tree* yang dapat dibentuk dari sembarang graf terhubung sederhana berbobot.

Sebuah graf *G* didefinisikan sebagai himpunan $(V(G), E(G))$, dengan $V(G)$ adalah himpunan tak kosong yang elemen-elemennya disebut titik (*vertex*) dan $E(G)$ adalah himpunan (boleh kosong) dari pasangan tak terurut (*u*,*v*) dari *vertex* $u, v \in V$ yang disebut sisi (*edge*) [3]. Selanjutnya graf *G* dapat dinyatakan dengan $G = (V, E)$. Jika $V = \{v_1, v_2, v_3, ..., v_n\}$ dan *e* adalah *edge* yang menghubungkan *vertex* v_i dengan v_j maka *e* dapat dinyatakan dengan $e = (v_i, v_j)$. Graf *G* digambarkan dalam bentuk diagram, yaitu setiap *vertex* di *G* digambarkan dengan noktah dan setiap *edge* yang menghubungkan dua *vertex* di *G* digambarkan dengan sebuah garis.

Jalan (*walk*) di graf *G* dinotasikan *W*(*G*) adalah barisan hingga yang diawali dan diakhiri oleh *vertex* yang unsur-unsurnya bergantian antara *vertex* dan *edge*, yaitu $W(G) = v_1, e_1, v_2, e_2, ..., v_n \ (n \geq 1)$ sedemikian hingga $e_i = (v_i, v_{i+1})$ untuk $i = 1,2,3,....,n-1$. *Walk W*(*G*) disebut *walk tertutup* jika $v_1 = v_n$, dan *walk W*(*G*) disebut *walk terbuka* jika $v_1 \neq v_n$. *Walk* dengan *edge* yang tidak diulang disebut jejak (*trail*), sedangkan *walk* dengan *vertex* yang tidak diulang disebut lintasan (*path*). Misal *e* adalah *edge* di graf *G* terhubung, maka didefinisikan $P(e, G)$ sebagai lintasan sederhana di *G* yang menghubungkan kedua *vertex* ujung dari *e*. Sebuah *walk* tertutup tanpa pengulangan *vertex* kecuali *vertex* awal dan *vertex* akhir disebut sikel (*cycle*).

Graf *G* dikatakan terhubung (*connected*) jika untuk setiap *vertex u* dan *v* di *G* terdapat *path* yang menghubungkan kedua *vertex* tersebut, sedangkan jika terdapat dua *vertex u* dan *v* di *G* yang tidak mempunyai *path*, maka graf *G* dikatakan tidak terhubung [4]. *Graf lengkap* adalah graf sederhana yang setiap *vertex*-nya mempunyai *edge* ke *vertex* lainnya. Graf yang setiap *vertex*-nya (e_i) diberi sebuah bobot (bilangan real) $w(e_i)$ disebut *graf berbobot* (*weighted graph*) [3].

Pohon adalah graf terhubung yang tidak memuat *cycle*. Sebuah pohon dapat mempunyai satu *vertex* saja tanpa mempunyai *edge*. Dengan kata lain, jika $T = (V, E)$ adalah pohon, maka *V* tidak boleh berupa himpunan kosong, namun *E* boleh kosong [3]. Misalkan $G = (V, E)$ adalah graf terhubung yang bukan pohon, artinya di *G* terdapat *cycle*. *G* dapat diubah menjadi pohon $T = (V, E)$ dengan cara menghapus *cycle-cycle* yang ada. Caranya yaitu dengan menghapus salah satu *edge* pada *cycle* sehingga tidak ada *cycle* pada *G*. Jika di *G*

tidak lagi ada *cycle* maka pohon T ini disebut dengan *spanning tree* [3]. Misalkan G merupakan graf berbobot, maka bobot *spanning tree* T_1 atau T_2 didefinisikan sebagai jumlah bobot semua *edge* di T_1 atau T_2, dinotasikan $w(T_1)$ atau $w(T_2)$. Diantara *spanning tree* yang ada pada G, yang paling penting adalah *spanning tree* dengan bobot minimum. *Spanning tree* dengan bobot minimum ini disebut *Minimum Spanning Tree* (MST) [3].

Ada beberapa algoritma untuk menemukan suatu MST pada graf berbobot G. Salah satunya Algoritma *Kruskal* yang paling umum diketahui. Berikut akan dijelaskan langkah-langkah dalam algoritma *Kruskal* [3].

1. daftarkan semua *edge* dari graf G berurutan berdasarkan bobotnya dari kecil ke besar.
2. pilih *edge e* dengan bobot yang paling kecil di G, masukkan ke *tree T*.
3. untuk setiap langkah selanjutnya berturut-turut pilih *edge e* yang paling kecil lainnya, tetapi *edge* tersebut tidak membentuk *cycle* di T. Tambahkan *edge e* ke dalam T.
4. ulangi langkah kedua sebanyak n-2 kali.

Langkah-langkah yang diperlukan untuk menentukan semua MST pada sembarang graf terhubung berbobot G :

1. pilih sembarang MST *T,* yaitu salah satu MST dari G. Gunakan T untuk menguji semua *edge* dari G. Misal *edge e* di G. Selidiki apakah e di T atau ada $f \in P(e,T)$ dimana $w(f) = w(e)$ untuk semua *edge* yang akan dipilih. Jika tidak demikian, maka *edge* tersebut dihapus dari G.
2. misal g adalah *edge* yang tidak dihapus di G. Untuk setiap *edge g* yang tidak ada di T, tentukan kelas *equivalen*-nya (dinotasikan $X(g)$) yaitu himpunan yang terdiri dari g dan *edge* dengan bobot yang sama pada $P(g,T)$. Jika terdapat irisan pada kelas *equivalen*, tentukan kelas *transitif*-nya, yaitu gabungan dari beberapa kelas *equivalen*. Tentukan kelas *singleton*-nya yaitu $\{e\}$, jika *edge* $e \in T$ dan $e \notin X(g)$. *Edge e* ini digunakan pada setiap MST dari graf G.
3. tentukan anggota himpunan terpilih dari kelas *equivalen* $X(g)$. Dimulai dengan T keseluruhan dan memindahkan semua anggota himpunan dari $X(g)$ untuk mendapatkan suatu subgraf H. Untuk setiap anggota himpunan dari $X(g)$, bentuk kesatuan dengan H dan uji keterhubungannya. Jika terhubung, maka anggota himpunan ini merupakan anggota himpunan terpilih dari kelas *equivalen* $X(g)$.
4. kombinasikan secara bebas anggota himpunan terpilih dari setiap kelas *equivalen* $X(g)$ dengan mengalikan banyaknya anggota himpunan terpilih dari setiap kelas *equivalen* $X(g)$ untuk mendapatkan semua MST yang ada dalam graf G [5].

2. Hasil dan Pembahasan

Pada bab ini, dibahas langkah-langkah untuk mendapatkan semua MST dari sembarang graf terhubung sederhana berbobot. Pertama-tama tentukan terlebih dahulu suatu MST referensi atau MST acuan (*tree T*) dari graf G yang diberikan menggunakan algoritma *Kruskal*.

Selanjutnya gunakan *tree T* untuk menguji semua *edge* dari graf *G*. Misal *edge e* adalah sembarang *edge* di *G*. Selidiki apakah *edge e* di *tree T* atau ada $f \in P(e,T)$ dimana $w(f) = w(e)$. Jika tidak demikian, maka *edge e* tersebut dihapus dari *G*, karena tidak bisa dipilih dalam pembentukan MST. Selidiki juga untuk *edge* lain di *G*. Kemudian tentukan kelas-kelas *equivalen*-nya dari semua *edge* yang bisa dipilih. Dari setiap kelas *equivalen*, tentukan anggota himpunan terpilihnya. Langkah terakhir adalah mengombinasikan secara bebas anggota himpunan terpilih dari setiap kelas *equivalen* untuk mendapatkan jumlah semua MST dari graf *G*. Pada pembahasan ini akan dikelompokkan ke dalam dua kasus, yaitu untuk beberapa *edge* berbobot tidak sama dan semua *edge* berbobot sama.

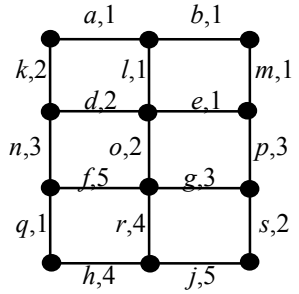
2.1. Graf Terhubung Sederhana dengan Beberapa Edge Berbobot Tidak Sama

Dalam graf terhubung sederhana ini akan ditentukan jumlah semua MST dari suatu graf dengan dua kejadian khusus berbeda, yaitu menentukan jumlah semua MST suatu graf dengan kombinasi *edge* yang dipilih tidak membentuk *cycle* dan kombinasi *edge* yang dipilih membentuk *cycle*.

Menentukan Semua MST Dari Graf Terhubung Sederhana Dengan Kombinasi *Edge* Yang Dipilih Tidak Membentuk *Cycle*

Kombinasi *edge* yang dipilih tidak membentuk *cycle* adalah kombinasi dari *edge* yang dipilih dari anggota himpunan terpilih suatu kelas *equivalen* yang tidak membentuk *cycle*. Kombinasi ini dapat digunakan dalam pembentukkan suatu MST. Misal diberikan suatu graf G_1 seperti Gambar 1.

G_1 :



Gambar 1 *Graf* G_1, *dengan simbol edge a,*1 *masing-masing menyatakan nama edge dan bobot edge.*

Langkah-langkah menentukan jumlah semua MST dari graf G_1 adalah sebagai berikut :

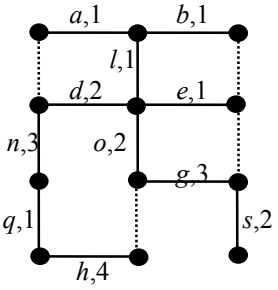
a. Menentukan MST Referensi (MST acuan)

Dalam menentukan MST referensi akan digunakan algoritma *Kruskal.* Pertama-tama daftarkan semua *edge* dari graf G_1 berurutan berdasarkan bobotnya dari kecil ke besar, yaitu :

Edge	(*a,b,e,l,m,q*)	(*d,k,o,s*)	(*g,n,p*)	(*h,r*)	(*f,j*)
Bobot	1	2	3	4	5

Kemudian pilih sembarang *edge* dengan bobot terendah, misalnya *edge a*, lalu masukkan ke *tree* T_1. Selanjutnya pilih *edge b*, masukkan ke dalam *tree* T_1. Setelah itu pilih *edge e,* karena tidak membentuk *cycle* dengan *edge a* dan *edge b*, maka masukkan juga ke dalam *tree* T_1. Lakukan langkah ini berturut-turut terhadap sembarang *edge* yang belum dipilih dengan bobot terendah di G_1 yang tidak membentuk *cycle* dengan *edge* yang telah terpilih sebelumnya sampai tidak ada lagi *edge* yang bisa dipilih. Sehingga didapatkan suatu MST referensi dari graf G_1, yaitu *tree* T_1 seperti pada Gambar 2.

T_1 :



Gambar 2 *Tree referensi* T_1 *dari Grafg* G_1

b. Memilih *Edge* Yang Dipakai Dalam MST G_1

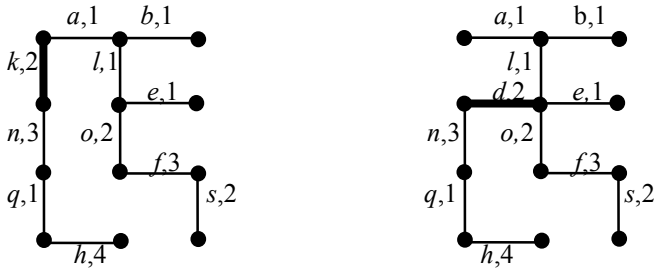
Langkah selanjutnya menguji semua *edge* di G_1. Dari Gambar 2 diketahui *edge* yang tidak termasuk dalam T_1 (*chord* (T_1)) adalah $\{f, j, k, m, p, r\}$. Misal z_1 adalah sembarang *edge* di G_1. Untuk *edge f*, karena $f \in G_1$ dan tidak ada satu pun z_1 yang merupakan anggota dari $P(f,T_1)$ dimana $w(f) = w(z_1)$, maka *edge f* tidak bisa dipilih untuk suatu MST di G_1 (tidak dipergunakan dalam MST dari graf G_1). Begitu juga untuk *edge j*, sedangkan untuk *edge k*, karena $k \in G_1$ dan $\exists\, d \in P(k,T_1)$ dimana $w(k) = w(d) = 2$, maka *edge k* bisa dipilih untuk suatu MST di G_1. Untuk *edge m*, karena $m \in G_1$ dan $\exists\, b \in P(m,T_1)$ dimana $w(m) = w(b) = 1$, maka *edge m* bisa dipilih untuk suatu MST di G_1. Begitu juga untuk *edge p* dan *r*.

c. Menentukan Kelas Equivalen

Misal V_1 adalah himpunan semua *edge* yang masih tersisa di G_1 tetapi tidak ada pada T_1 dengan bobot kurang atau sama dengan bobot terbesar di T_1. Untuk setiap $v \in V_1$, tentukan kelas *equivalen* $X(v)$ yaitu himpunan yang terdiri dari v dan semua *edge* di $P(v,T_1)$ yang mempunyai bobot yang sama dengan v. Dari G_1 dapat ditentukan $X(k) = \{k, d\}$ $X(m) = \{m, b, e, l\}$, $X(p) = \{p, g\}$, $X(r) = \{r, h\}$. Untuk *edge a*, *n, o, q* dan *s* termasuk ke dalam kelas *singleton* karena semuanya termasuk dalam T_1 tetapi tidak ada pada $X(v)$. Jadi, kelas *singleton* untuk graf G_1 adalah $\{a\}$, $\{n\}$, $\{o\}$, $\{q\}$ dan $\{s\}$.

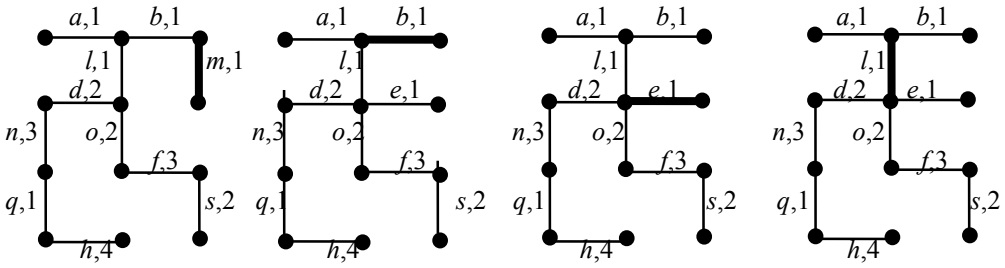
d. Menentukan Anggota Himpunan Terpilih

Untuk mendapatkan anggota himpunan terpilih dari $X(k)$, dilihat dari keterhubungan semua anggota himpunan dari kelas *equivalen* $X(k)$ terhadap T_1. Kedua anggota himpunan dari $X(k)$, yaitu *edge k* dan *edge d* terhubung terhadap T_1 (ditunjukkan dengan garis tebal pada Gambar 3), maka kedua *edge* tersebut merupakan anggota himpunan terpilih dari $X(k)$.



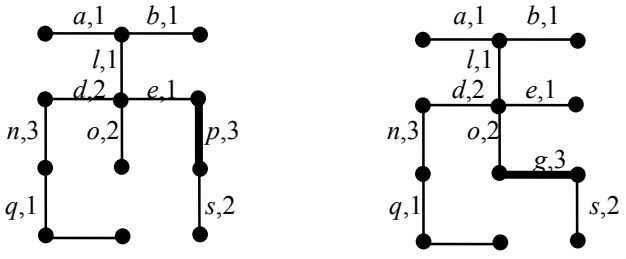
Gambar 3 *Anggota Himpunan Terpilih Dari* $X(k)$ (*cetak tebal*)

Dari Gambar 3 dapat dilihat bahwa jumlah *edge* yang mungkin menjadi *edge* pilihan dari kelas *equivalen* $X(k)$ (tidak menciptakan *cycle*) (dinotasikan $i([k])$) adalah $i([k]) = 1$, setiap satu *edge* membentuk suatu pilihan, yaitu $\{k\}$ atau $\{d\}$. Untuk $X(m) = \{m, b, e, l\}$ keempat anggota himpunan dari $X(m)$ terhubung terhadap T_1, maka keempat *edge* tersebut merupakan anggota himpunan terpilih dari $X(k)$.



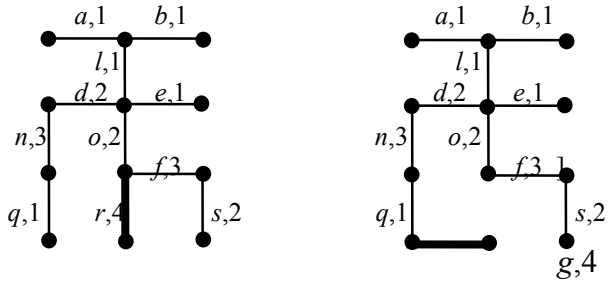
Gambar 4 *Anggota Himpunan Terpilih Dari* $X(m)$ (*cetak tebal*)

Dari Gambar 4 dapat dilihat bahwa jumlah *edge* yang mungkin menjadi *edge* pilihan dari kelas *equivalen* $X(m)$, yaitu $i([m]) = 3$, setiap tiga *edge* membentuk suatu pilihan, yaitu $\{b, l, e\}$, $\{b, m, e\}$, $\{l, e, m\}$ atau $\{l, b, m\}$. Untuk $X(p) = \{p, g\}$ kedua anggota himpunan darinya terhubung terhadap T_1, maka kedua *edge* tersebut merupakan anggota himpunan terpilih dari $X(p)$.



Gambar 5 *Anggota Himpunan Terpilih Dari* $X(p)$ (*cetak tebal*)

Dari Gambar 5 dapat dilihat bahwa jumlah *edge* yang mungkin menjadi *edge* pilihan dari kelas *equivalen* $X(p)$, yaitu $i([p]) = 1$, setiap satu *edge* membentuk suatu pilihan, yaitu $\{p\}$ atau $\{g\}$. Untuk $X(r) = \{r, h\}$ kedua anggota himpunannya terhubung terhadap T_1, maka kedua *edge* tersebut merupakan anggota himpunan terpilih dari $X(r)$.



Gambar 6 *Anggota Himpunan Terpilih Dari* $X(r)$ (*cetak tebal*)

Dari Gambar 6 dapat dilihat bahwa jumlah *edge* yang mungkin menjadi *edge* pilihan dari kelas *equivalen* $X(r)$, yaitu $i([r]) = 1$, setiap satu *edge* membentuk suatu pilihan, yaitu $\{r\}$ atau $\{h\}$. Untuk mendapatkan jumlah semua MST dari graf G_1 dilakukan dengan **kombinasi *edge* yang dipilih tidak membentuk *cycle***, karena dari keempat kelas *equivalen* $X(k)$, $X(m)$, $X(p)$ dan $X(r)$, kombinasi dari masing-masing *i* tidak menciptakan suatu *cycle*.

e. Mengombinasikan Anggota Himpunan Terpilih

Dari langkah (d) didapatkan banyaknya anggota himpunan terpilih dari $X(k)$, $X(m)$, $X(p)$ dan $X(r)$ masing-masing adalah 2, 4, 2 dan 2. Sedangkan jumlah *edge* yang mungkin menjadi pilihan dari $X(k)$, $X(m)$, $X(p)$ dan $X(r)$ masing-masing adalah $i([k]) = 1$, $i([m]) = 3$, $i([p]) = 1$ dan $i([r]) = 1$. Untuk setiap kelas *singleton*, jumlah anggota himpunan terpilih dan pemasukan bilangan *i*-nya adalah 1 ($c([a])$, $c([n])$, $c([o])$, $c([q])$, $c([s]) = \binom{1}{1} = 1$). Dengan cara mengombinasikan semua anggota himpunan terpilih dengan masing-masing *i* dari setiap kelas *equivalen* dan kelas *singleton*-nya maka didapatkan jumlah semua MST dari graf G_1 (dinotasikan $\prod c([G_1])$), yaitu :

$$\prod c([G_1]) = c([a])\, c([n])\, c([o])\, c([q])\, c([s])c([k])\, c([m])\, c([p])\, c([r])$$

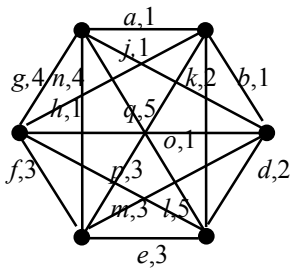
$$= \binom{1}{1}\binom{1}{1}\binom{1}{1}\binom{1}{1}\binom{1}{1}\binom{2}{1}\binom{4}{3}\binom{2}{1}\binom{2}{1}$$

$$= (1)\,(1)\,(1)\,(1)\,(1)(2)\,(4)\,(2)\,(2) = 32$$

Menentukan Semua MST Dari Graf Terhubung Sederhana Dengan Kombinasi *Edge* Yang Dipilih Membentuk *Cycle*

Kombinasi *edge* yang dipilih membentuk *cycle* adalah kombinasi dari *edge* yang dipilih dari anggota himpunan terpilih suatu kelas *equivalen* yang membentuk *cycle*, sehingga tidak semua kombinasi ini dapat digunakan dalam pembentukkan suatu MST.

G_2 :



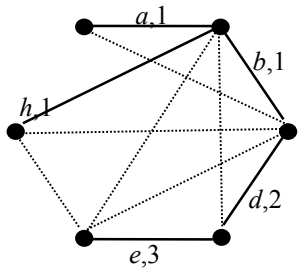
Gambar 7 *Graf* G_2

Langkah-langkah menentukan jumlah semua MST dari graf G_1 adalah sebagai berikut :

a. Menentukan MST Referensi (MST acuan)

Dengan algoritma *Kruskal* didapatkan suatu MST referensi dari graf G_2, yaitu *tree* T_2 dengan bobot 8 seperti pada Gambar 8.

T_2 :



Gambar 8 *Tree Referensi* T_2 *Dari Graf* G_2

b. Memilih *Edge* Pada G_2 Yang Dipakai Dalam MST

Setelah melakukan pengujian pada semua *edge* di G_2, dapat ditentukan *edge* yang bisa dipilih untuk suatu MST G_2 adalah $\{f, j, k, m, o, p\}$, sedangkan *edge* yang tidak bisa dipilih adalah $\{g, l, n, q\}$.

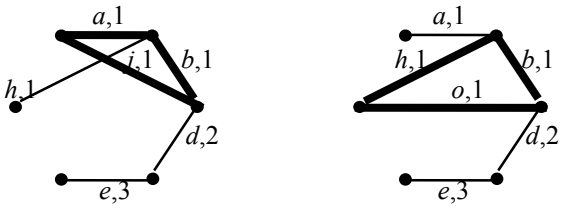
c. Menentukan Kelas Equivalen

Dari *edge* yang bisa dipilih ditentukan kelas *equivalen*-nya yaitu $X(f) = \{f, e\}$, $X(m) = \{m, e\}$, $X(p) = \{p, e\}$, $X(j) = \{j, a, b\}$, $X(o) = \{o, b, h\}$, $X(k) = \{k, d\}$. Karena terdapat irisan pada kelas *equivalen* $X(j)$ dan $X(o)$, maka kedua kelas *equivalen* ini dijadikan ke dalam satu kelas *transitif* $X(y) = \{j, a, b, o, h\}$. Begitu juga untuk kelas *equivalen* $X(f)$, $X(m)$ dan $X(p)$ terdapat irisan sehingga dijadikan ke dalam satu kelas *transitif* $X(s) = \{f, e, m, p\}$. Graf G_2 tidak memiliki kelas *singleton* karena tidak ada *edge* di T_2 yang tidak termasuk $X(v)$.

d. Menentukan Anggota Himpunan Terpilih

Dengan langkah yang sama pada subbab sebelumnya didapatkan anggota himpunan terpilih untuk masing-masing kelas *equivalen* yaitu $X(f) = \{f, e, m, p\}$ dengan $i([f]) = 1$, $X(y) = \{j, a, b, o, h\}$ dengan $i([y]) = 3$, $X(k) = \{k, d\}$ dengan $i([k]) = 1$.

Untuk mendapatkan jumlah semua MST dari graf G_2 dilakukan dengan **kombinasi *edge* yang dipilih membentuk *cycle***, karena ada kombinasi $i([y])$ pada kelas *transitif* $X(y)$ yang menciptakan suatu *cycle*, yaitu $\{a, b, j\}$ dan $\{b, h, o\}$ (ditunjukkan dengan garis tebal pada gambar 9), sehingga dua kombinasi ini tidak digunakan dalam MST.



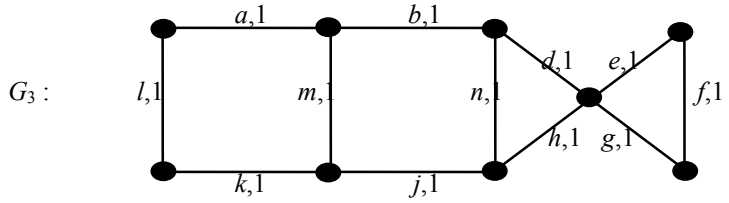
Gambar 9 *Kombinasi* $i([j])$ *Yang Membentuk Cycle*

e. Mengombinasikan Anggota Himpunan Terpilih

Dengan cara mengombinasikan semua anggota himpunan terpilih dengan masing-masing i dari setiap kelas *equivalen* maka akan didapatkan jumlah semua MST dari graf G_2 (dinotasikan $\prod c([G_2])$). Untuk $c([y]) = \binom{5}{3}$ - 2 = 10 – 2 = 8. Sehingga diperoleh :

$$\prod c([G_2]) = c([k])\, c([f])\, c([y])$$
$$= \binom{2}{1}\binom{4}{1}\left(\binom{5}{3} - 2\right) = (2)\,(4)\,(8) = 64$$

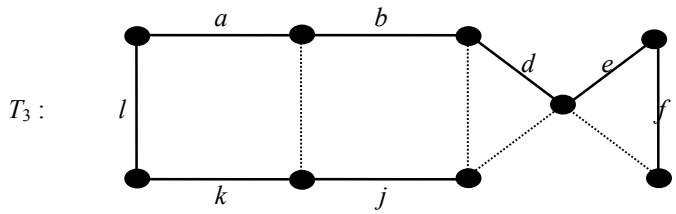
2.2. Graf Terhubung Sederhana Dengan Semua Bobot Sama



Gambar 9 *Graf* G_3

a. Menentukan MST Referensi (MST acuan)

Dengan algoritma *Kruskal* didapatkan suatu MST referensi dari graf G_3, yaitu *tree* T_3 dengan bobot 8 seperti pada Gambar 10.



Gambar 10 *Tree Referensi* T_3 *Dari Graf* G_3

b. Memilih *Edge* Pada G_3 Yang Dipakai Dalam MST

Dari Gambar 10 diketahui bahwa *edge* yang bisa dipilih adalah *g, h, m* dan *n,* dengan kata lain semua *edge* di G_3 dapat digunakan dalam suatu MST di G_3 (ditunjukkan dengan garis putus-putus pada Gambar 10).

c. Menentukan Kelas Equivalen

Dari *edge* yang bisa dipilih dapat ditentukan $X(g) = \{g, e, f\}$, $X(h) = \{h, b, a, l, k, j\}$, $X(m) = \{m, a, l, k\}$, $X(n) = \{n, b, a, l, k, j\}$. Karena terdapat irisan pada kelas *equivalen* $X(h)$, $X(m)$ dan $X(n)$, maka terdapat kelas *transitif* $X(s) = \{a, b, d, l, k, j, h, m, n\}$. Graf G_3 tidak memiliki kelas *singleton* karena tidak ada *edge* di T_3 yang tidak termasuk dalam kelas *equivalen.*

d. Menentukan Anggota Himpunan Terpilih

Dengan langkah yang sama pada subbab sebelumnya didapatkan angggota himpunan terpilih dari $X(g)$ adalah *g, e, f* dengan $i([g]) = 2$, angggota himpunan terpilih dari $X(s)$ dengan $i([s]) = 6$.

Untuk mendapatkan jumlah semua MST dari graf G_3 dilakukan dengan **kombinasi *edge* yang dipilih membentuk *cycle***, karena ada 43 kombinasi $i([s])$ pada kelas *transitif* $X(s)$ yang menciptakan suatu *cycle* sehingga 43 kombinasi ini tidak digunakan dalam MST.

e. Mengombinasikan Himpunan Bagian Terpilih

Dengan cara mengombinasikan semua anggota himpunan terpilih dengan masing-masing *i* dari setiap kelas *equivalen* maka akan didapatkan jumlah semua MST dari graf G_3 (dinotasikan $\prod c([G_3])$). Khusus untuk $c([s]) = \binom{9}{6} - 43 = 84 - 43 = 41$ sehingga diperoleh

$$\prod c([G_3]) = c([g])\, c([s])$$
$$= \binom{3}{2}\left(\binom{9}{6} - 43\right) = (3)\,(41) = 123.$$

3. Kesimpulan

Dari hasil dan pembahasan didapat kesimpulan bahwa sembarang graf terhubung sederhana berbobot dapat ditentukan semua MST-nya. Jumlah MST dari graf terhubung berbobot merupakan perkalian dari semua kombinasi anggota himpunan terpilih dari setiap kelas *equivalen*.

Daftar Pustaka

[1] Chartrand, G and Oellermann, O. R., (1993), Applied and Algorithmic Graph Theory. New York : McGraw_Hill, Inc.

[2] Johnsonbaugh, R., (1998), Discrete Mathematics, 2nd Edition. London: Collier Macmillan Publisher.

[3] Munir, R., (2003), Matematika Diskrit, Bandung: CV. Informatika.

[4] Wilson, R., (1998), Graph an Introductory Approach. New York : John Willey and Sons, Inc.

[5] Wright, P., (1997), Counting and Constructing Minimal Spanning Trees, Bulletin of The Institute of Combinatorics and Its Applications, 21, 65 – 76.

[6] Wright, P., (2000), On Minimum Spanning Trees and Determinants, Mathematics Magazine, 73, 21 – 28.