BAB II

LANDASAN TEORI

2.1. Teori Graf

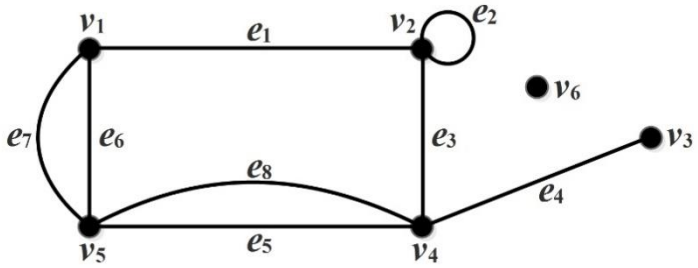
Definisi 2.1.1: Graf (C. Vasudev, 2006:1)

Sebuah graf G terdiri atas sebuah himpunan tak kosong $V(G) = \{v_1, v_2, \cdots\}$ dimana setiap elemen himpunan V disebut sebagai simpul (*vertices*) dan himpunan $E(G) = \{e_1, e_2, \cdots\}$ dimana setiap elemen himpunan E disebut sebagai sisi (*edges*) sedemikian sehingga himpunan $E \subseteq \binom{V(G)}{2}$. Pada umumnya suatu graf dinotasikan sebagai $G = (V, E)$.

Sebuah simpul direpesentasikan sebagai lingkaran kecil atau titik, sedangkan sebuah sisi direpresentasikan dalam bentuk garis lurus ataupun lengkung. Simpul u dan v disebut sebagai simpul ujung atas suatu sisi e pada graf G apabila kedua simpul tersebut terhubung oleh suatu sisi e pada graf G. Pasangan dua buah simpul yang dihubungkan oleh sisi e biasanya dinotasikan dengan $\{u, v\}$ atau $\{v, u\}$ serta boleh juga hanya dituliskan sebagai uv atau vu dimana $u \neq v$. (Jean – Claude Fournier, 2009: 24)

Contoh 2.1:

Graf G_1 yang terdiri dari himpunan simpul $V(G) = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ dan himpunan sisi $E(G) = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$



Gambar 2.1 Graf G_1

Pada Gambar 2.1, v_3 disebut sebagai simpul akhir (*end vertex*) yakni simpul yang hanya memiliki satu buah sisi yakni e_4. Sisi e_5 dan e_8 serta sisi e_6 dan e_7 disebut sebagai sisi paralel (*multiple edges*), yakni dua buah sisi atau lebih yang memiliki simpul – simpul ujung yang sama. Sisi *e* Simpul v_6 disebut sebagai simpul terasing (*isolated vertex*) yakni simpul yang tidak memiliki sisi satupun. Sisi e_2 disebut dengan *loop*. *Loop* sendiri adalah sisi yang menghubungkan sebuah simpul kepada dirinya sendiri. (Gross dan Yellen, 1999:2)

Jika $e = uv$ merupakan suatu sisi di graf *G*, maka *u* dan *v* dikatakan saling *adjacent* di graf *G* (*v* dan *u* saling *adjacent* satu sama lain). Sedangkan simpul *u* dan sisi *e* dikatakan saling *incident* begitupun dengan simpul *v* dan sisi *e*. Jika dua buah sisi yang berbeda katakan *e* dan *f*, keduanya *incident* dengan sebuah simpul yang sama maka kedua sisi tersebut dikatakan saling *adjacent* sisi. (C. Vasudev, 2006:2)

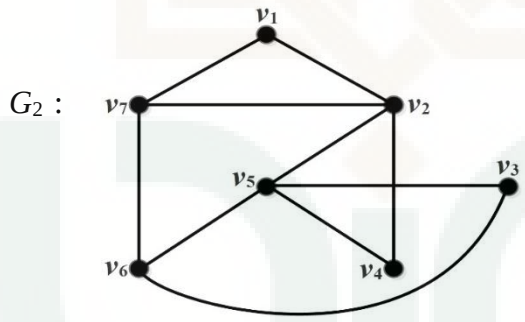
Contoh 2.2:

Pada Gambar 2.1 simpul v_3 *adjacent* dengan simpul v_4. Sisi e_4 *incident* dengan simpul v_3 dan simpul v_4. Sisi e_5 *adjacent* sisi dengan sisi e_3, e_4, e_6, e_7, e_8.

Banyaknya simpul pada suatu graf G disebut dengan *order* yang dinotasikan dengan $|V(G)|$, sedangkan banyaknya sisi pada suatu graf G disebut dengan *size* yang dinotasikan dengan $|E(G)|$. (Harris, Hirst, dan Mossinghoff, 2000:5)

Contoh 2.3:

Pada Gambar 2.2 diberikan sebuah graf $G_2(V, E)$ dengan himpunan simpul $V(G_2) = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$



Gambar 2.2 *Order* dan *Size* graf G_2

Definisi 2.1.2: Derajat (*degree*) (Harris, Hirst, dan Mossinghoff, 2000: 6)

Derajat (*degree*) dari suatu simpul $v \in V(G)$ dinotasikan dengan deg(v) adalah jumlah sisi yang *incident* dengan v. Derajat maksimum (*maximum degree*) pada graf G dinotasikan dengan $\Delta(G)$, didefinisikan sebagai:

$$\Delta(G) = \max\{\deg(v) \mid v \in V(G)\}$$

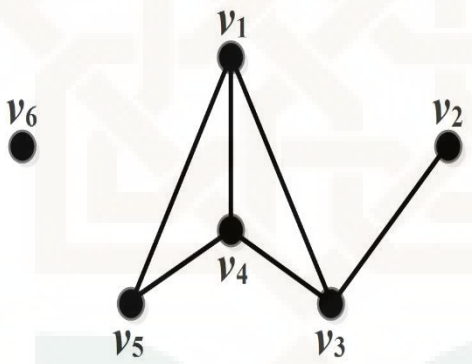
Sedangkan derajat minimum (*minimum degree*) pada graf *G* dinotasikan sebagai $\delta(G)$, didefinisikan sebagai:

$$\delta(G) = \min\{\deg(v) \mid v \in V(G)\}.$$

Contoh 2.4:

Graf G_3 dengan derajat maksimum $\Delta(G_3) = 3$ dan derajat minimum $\delta(G_3) = 0$

G_3 :



$\deg(v_1) = 3$
$\deg(v_2) = 1$
$\deg(v_3) = 3$
$\deg(v_4) = 3$
$\deg(v_5) = 2$
$\deg(v_6) = 0$

Gambar 2.3 Derajat simpul pada graf G_3

Teorema 2.1.1: Handshaking (Aldous dan Wilson, 1988: 37)

Pada suatu graf *G* berlaku:

$$\sum_{v \in V(G)}^{n} \deg(v) = 2 \left| E(G) \right|$$

Bukti:

Pada suatu graf *G*, setiap sisi dengan tepat menghubungkan dua buah simpul maka jika dilakukan penghitungan terhadap seluruh derajat simpul di *G*, setiap sisi dengan tepat terhitung sebanyak 2 kali. ■

Teorema 2.1.2: Teorema Derajat Minimum dan Maksimum (Jean – Claude Fournier, 2009: 34)

Pada suatu graf dengan derajat minimum $\delta(G)$ dan derajat maksimum $\Delta(G)$ maka berlaku:

$$\delta(G) \leq 2\frac{|E(G)|}{|V(G)|} \leq \Delta(G)$$

Bukti:

i. Akan dibuktikan $\delta(G) \leq 2\frac{|E(G)|}{|V(G)|}$.

Berdasarkan Teorema 2.1.1 maka diperoleh:

$$\sum_{v \in V(G)}^{n} \deg(v) = 2\,|E(G)|$$

Karena $\delta(G)$ adalah derajat minimum di *G* maka:

$$\sum^{n} \delta(G) \;\leq \sum_{v \in V(G)}^{n} \deg(v)$$

$$n\delta(G) \;\leq\; 2\,|E(G)|$$

$$\delta(G) \leq 2\,\frac{|E(G)|}{n}$$

karena n merepresentasikan jumlah simpul pada suatu graf maka:

$$\delta(G) \leq 2\,\frac{|E(G)|}{|V(G)|}$$

ii. Akan dibuktikan $2\frac{|E(G)|}{|V(G)|} \leq \Delta(G)$

Berdasarkan Teorema 2.1.1 maka diperoleh:

$$\sum_{v \in V(G)}^{n} \deg(v) = 2\,|E(G)|$$

Karena $\Delta(G)$ adalah derajat minimum di G maka:

$$\sum_{v \in V(G)}^{n} \deg(v) \leq \sum^{n} \Delta(G)$$

$$2\,|E(G)| \leq n\Delta(G)$$

$$2\,\frac{|E(G)|}{n} \leq \Delta(G)$$

karena n merepresentasikan jumlah simpul pada suatu graf maka:

$$2\,\frac{|E(G)|}{|V(G)|} \leq \Delta(G)$$

Berdasarkan i dan ii maka diperoleh :

$$\delta(G) \leq 2\frac{|E(G)|}{|V(G)|} \leq \Delta(G)$$

■

2.2 Jenis – Jenis Graf

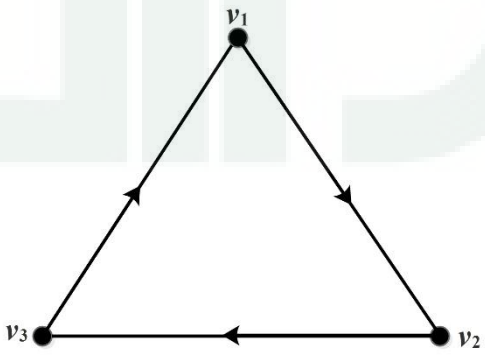
Definisi 2.2.1: Graf Berarah (C. Vasudev, 2006:3)

Suatu graf berarah *G* atau disebut dengan *digraf G* terdiri dari sebuah himpunan $V = \{v_1, v_2, v_3, \cdots\}$ dimana elemen himpunan *V* disebut dengan simpul (*vertices*) dan sebuah himpunan $E = \{e_1, e_2, e_3, \cdots\}$ dimana elemen himpunan *E* disebut dengan sisi berarah (*arcs*). Suatu sisi berarah *e* merupakan sisi yang menghubungkan dua buah simpul *v*, $y \in V$ secara berurutan. Dengan kata lain, jika untuk tiap - tiap sisi pada suatu graf *G* memiliki tepat satu arah maka graf *G* tersebut dinamakan dengan graf berarah (*directed graph*) atau *digraf G*.

Suatu sisi berarah (*arcs*) pada suatu graf berarah *G* direpesentasikan sebagai garis yang memiliki arah (busur panah) dengan simpul asal katakan *u* menuju ke simpul tujuan katakan *v*.

Contoh 2.5:

G_4 :



Gambar 2.4 Graf berarah G_4

Dari Gambar 2.4 di atas sisi $e = v_1v_2$ pada graf berarah G_4 merupakan sisi berarah (*arc*), maka:

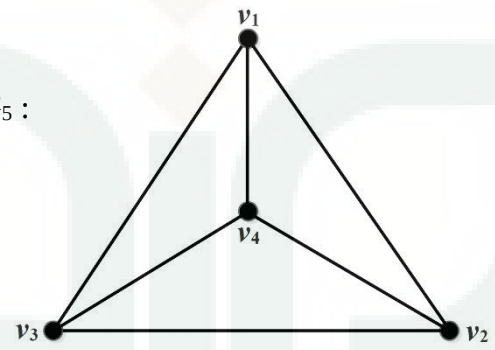
- Simpul v_1 dinamakan simpul asal dan simpul v_2 dinamakan sebagai simpul tujuan.
- Sisi *e* dikatakan *incident* dari simpul v_1 menuju simpul v_2
- Simpul v_1 dikatakan *adjacent* menuju simpul v_2, dan simpul v_2 dikatakan *adjacent* dari simpul v_1

Definisi 2.2.2: Graf tidak berarah (C. Vasudev, 2006:3)

Suatu graf *G* dinamakan graf tak berarah (*undirected graph*) apabila elemen himpunan *E* bukan merupakan sisi berarah.

Contoh 2.6:

G_5 :

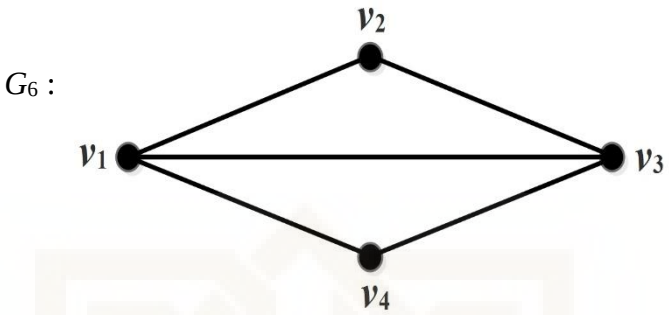


Gambar 2.5 Graf tak berarah G_5

Definisi 2.2.3: Graf Sederhana (Jean – Claude Fournier, 2009: 26)

Suatu graf *G* dikatakan sebagai graf sederhana (*simple graph*) jika tidak memuat *loops* dan sisi paralel (*multiple edges*).

Contoh 2.7:



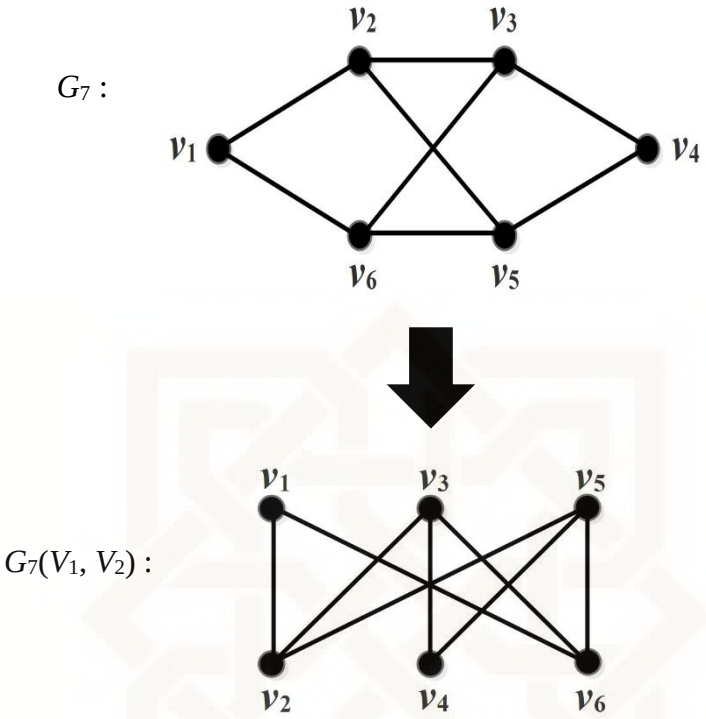
Gambar 2.6 Graf sederhana G_6

Definisi 2.2.4: Graf Bipartit (Ibrahim dan Noor Saif, 2013: 72)

Sebuah graf G disebut bipatit (*bipartite*) jika $V(G)$ dapat dipartisi menjadi dua himpunan tak kosong V_1 dan V_2 sedemikian sehingga setiap sisi dari G menghubungkan sebuah simpul di V_1 dan sebuah simpul di V_2. Dengan kata lain, graf bipartit adalah suatu graf yang himpunan simpulnya dapat dipartisi menjadi dua bagian V_1 dan V_2 sedemikian sehingga setiap sisinya mempunyai simpul ujung di V_1 dan simpul ujung yang lain di V_2. Kita sebut (V_1, V_2) bipartit dari G.

Contoh 2.8:

Gambar 2.7 merupakan contoh graf bipartit. Graf G_7 dipartisi menjadi graf $G_7(V_1, V_2)$ dengan $V_1 = \{v_1, v_3, v_5\}$ dan $V_2 = \{v_2, v_4, v_6\}$.



Gambar 2.7 Graf G_7 yang dipartisi menjadi graf $G_7(V_1, V_2)$

Teorema 2.2.1 : Teorema Graf Bipartit (C. Vasudev, 2006: 169)

Sebuah graf *G* dikatakan bipartit jika dan hanya jika tidak memuat sebuah *cycle* ganjil.

Bukti:

(⇒) Asumsikan bahwa *G* merupakan graf bipartit. Andaikan *G* memuat paling sedikit sebuah *cycle* ganjil katakan *B* dengan $B = (v_1, v_2, \cdots, v_n, v_1)$. Simpul $v_1 \in V_1$, $v_2 \in V_2$, $v_3 \in V_1$ dan seterusnya. Untuk setiap *k* dengan $k \in \{1, 2, \cdots, n\}$, diperoleh:

$$v_k \in \begin{cases} V1 & : k \text{ ganjil} \\ V2 & : k \text{ genap} \end{cases}$$

jelas bahwa $v_n \in V_1$ karna $v_n \in B$ dimana B adalah *cycle* ganjil. Karena $v_1 \in V_1$ dan $v_n \in V_1$ maka hal ini kontradiksi dengan asumsi bahwa G merupakan graf bipartit sehingga pengandaikan harus diingkari.

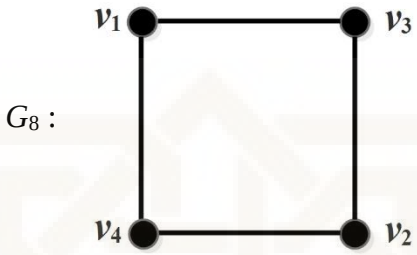
(⇐) Asumsikan bahwa G tidak memuat sebuah *cycle* ganjil. Andaikan bahwa G bukan merupakan graf bipartit maka terdapat dua buah simpul pada himpunan bagian V_1 atau V_2 yang saling *adjacent*. Misal dipilih sebarang simpul $v \in G$. Apabila himpunan simpul $V(G)$ dibagi menjadi dua bagian yakni V_1 merupakan himpunan simpul dimana jarak terpendek untuk setiap simpul di V_1 ke simpul v adalah ganjil serta V_2 merupakan himpunan simpul dimana jarak terpendek untuk setiap simpul di V_2 ke simpul v adalah genap, maka $v \in V_2$ dan $V_1 \cap V_2 = \emptyset$. Misal $a_1, a_2 \in V_1$ dimana keduanya *adjacent* maka paling sedikit terdapat sebuah *cycle* dengan panjang ganjil katakan B dengan $B = (v, \cdots, a_1, a_2, \cdots, v)$. Hal ini kontradiksi dengan asumsi di awal bahwa G tidak memuat sebuah *cycle* ganjil sehingga pengandaian harus diingkari. ■

Definisi 2.2.5: Graf Bipartit Lengkap (Aldous dan Wilson, 1988: 48)

Graf bipartit lengkap adalah sebuah graf bipartit yang mana setiap simpul di V_1 terhubung dengan setiap simpul di V_2 oleh sebuah sisi.

Contoh 2.9:

Diberikan sebuah graf bipartit lengkap dengan himpunan simpul $V_1 = \{v_1, v_2\}$ dan $V_2 = \{v_3, v_4\}$.



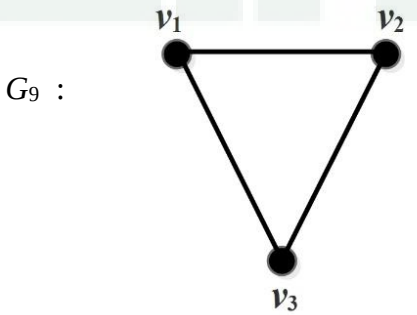
Gambar 2.8 Graf bipartit lengkap G_8

Definisi 2.2.6: Graf bukan Bipartit (Gross dan Yellen, 1990: 17)

Suatu graf G bukan merupakan graf bipartit (*non-bipartite*) apabila himpunan simpul $V(G)$ tidak dapat dipartisi menjadi dua himpunan tak kosong V_1 dan V_2.

Contoh 2.10:

Diberikan sebuah graf G_9 dengan himpunan simpul $V(G_9) = \{v_1, v_2, v_3\}$.



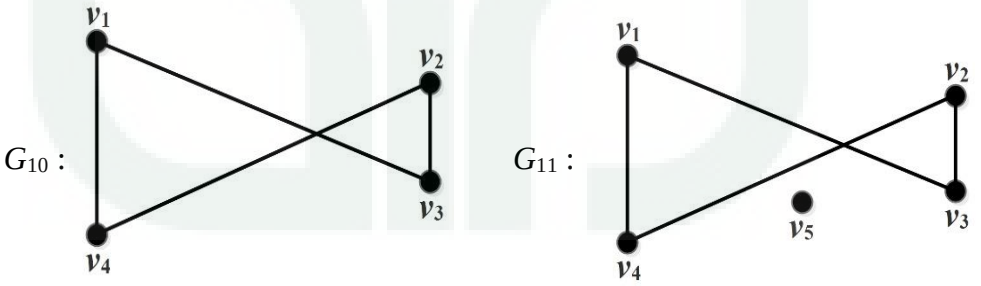
Gambar 2.9 Graf bukan bipartit G_9

Dari Gambar 2.9 himpunan simpul $V(G_9)$ apabila dipartisi menjadi dua himpunan tak kosong V_1 dan V_2 maka akan terdapat minimal dua buah simpul yang saling *adjacent* pada partisi himpunan yang sama sehingga graf G_9 merupakan graf bukan bipartit.

Definisi 2.2.7: Graf Terhubung (Diestel, 2005:10)

Suatu graf G dikatakan terhubung jika untuk setiap dua simpul dalam graf *G* tersebut dapat dihubungkan oleh sebuah lintasan pada graf *G*. Dengan kata lain apabila dapat ditemukan sebuah lintasan untuk setiap dua simpul *G* maka *G* merupakan graf terhubung.

Contoh 2.11:



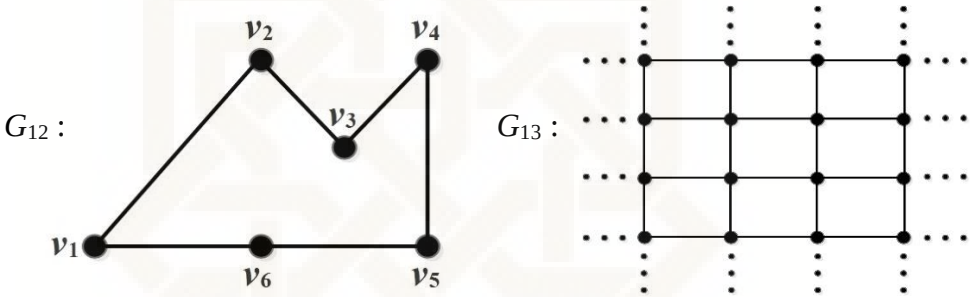
(a) (b)

Gambar 2.10 (a) Graf terhubung G_{10}
(b) Graf tidak terhubung G_{11}

Definisi 2.2.8: Graf Berhingga dan tak Berhingga (C. Vasudev, 2006:4)

Suatu graf *G* dikatakan berhingga jika banyaknya simpul maupun sisi pada *G* jumlahnya berhingga dan dikatakan tak berhingga apabila sebaliknya.

Contoh 2.12:



(a) (b)

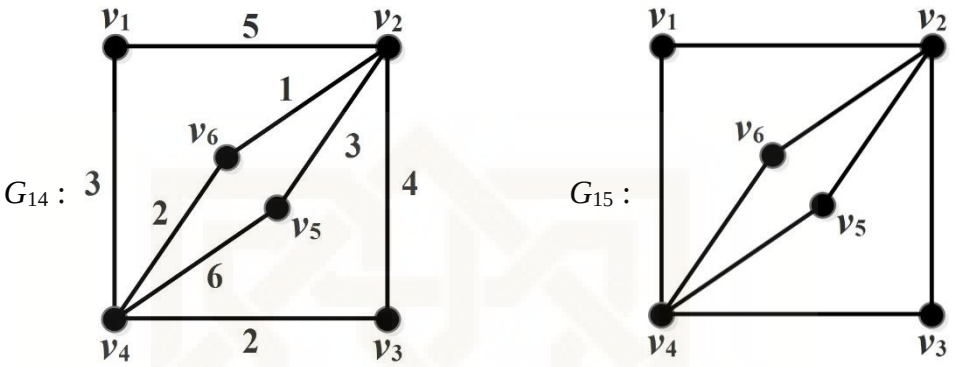
Gambar 2.11 (a) Graf berhingga G_{12}

(b) Graf tak berhingga G_{13}

Definisi 2.2.9: Graf Berbobot dan Graf tidak Berbobot (Bondy dan Murty, 1976: 15)

Graf berbobot (*weighted graph*) merupakan graf dengan tiap sisinya memuat nilai atau bobot berupa bilangan real. Sedangkan graf dengan tiap sisi nya tidak memuat nilai atau bobot disebut dengan graf tidak berbobot (*unweighted graph*).

Contoh 2.13:



(a) (b)

Gambar 2.12 (a) Graf Berbobot G_{14}
(b) Graf tidak berbobot G_{15}

2.3 Subgraf dan Subgraf Perentang

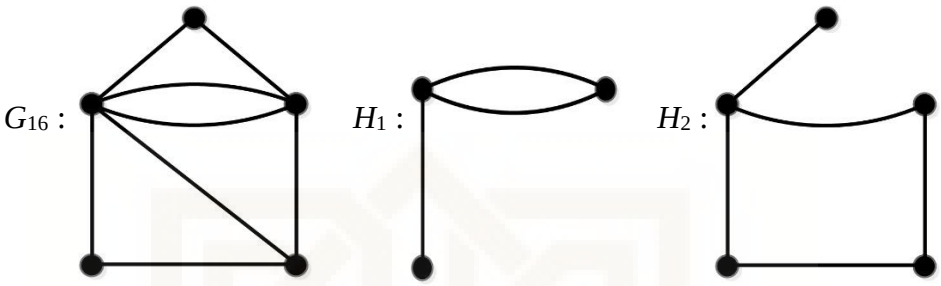
Definisi 2.3.1: Subgraf (Bondy dan Murty, 1976: 8)

Sebuah graf H disebut subgraf dari graf G dinotasikan dengan $H \subseteq G$ jika $V(H) \subseteq V(G)$ dan $E(H) \subseteq E(G)$.

Definisi 2.3.2: Subgraf Perentang (Bondy dan Murty, 1976: 8)

Sebuah graf H disebut subgraf perentang dari graf G jika $V(H) = V(G)$ dan $E(H) \subseteq E(G)$.

Contoh 2.14:



Gambar 2.13 (a) Graf G_{16}
(b) H_1 subgraf dari G_{16}
(c) $H2$ subgraf perentang dari G_{16}

2.4 Jalan, Lintasan, dan *cycle*

Definisi 2.4.1: Jalan (C. Vasudev, 2006:56)

Jalan (*walk*) dalam sebuah graf G merupakan sebuah rangkaian bergantian dari simpul dan sisi dengan sebuah simpul sebagai awal dan akhirnya, sedemikian sehingga:

- Setiap sisinya *incident* dengan simpul sebelum dan simpul sesudahnya
- Memungkinkan terdapat pengulangan simpul maupun sisi. Apabila tidak didapati pengulangan sisi lebih dari sekali maka dinamakan sebuah *trail*.

Berdasarkan simpul awal dan akhir sebuah jalan dibedakan menjadi dua, yakni jalan terbuka dan jalan tertutup. Sebuah jalan disebut sebagai jalan

terbuka apabila simpul awal tidak sama dengan simpul akhir ($v_0 \neq v_n$). Sedangkan sebuah jalan disebut sebagai jalan tertutup apabila simpul awal sama dengan simpul akhir ($v_0 = v_n$). Karena sebuah *trail* merupakan sebuah jalan (tanpa pengulangan sisi) sehingga sebuah *trail* juga dibedakan menjadi dua. *Trail* terbuka jika simpul awal tidak sama dengan simpul akhir sedangkan *trail* tertutup jika simpul awal sama dengan simpul akhir.

Definisi 2.4.2: Lintasan (Aldous dan Wilson, 1988:40)

Suatu lintasan (*path*) pada graf *G* merupakan sebuah jalan dimana setiap simpul dan sisinya tidak berulang.

Sebuah *trail* dan sebuah lintasan merupakan dua hal yang berbeda dimana pada sebuah *trail* tidak terdapat pengulangan sisi sedangkan pada sebuah lintasan tidak terdapat pengulangan sisi dan simpul.

Definisi 2.4.3: *Cycle* (Diestel, 2005:8)

Sebuah *cycle* adalah sebuah lintasan dengan simpul awal sama dengan simpul akhir.

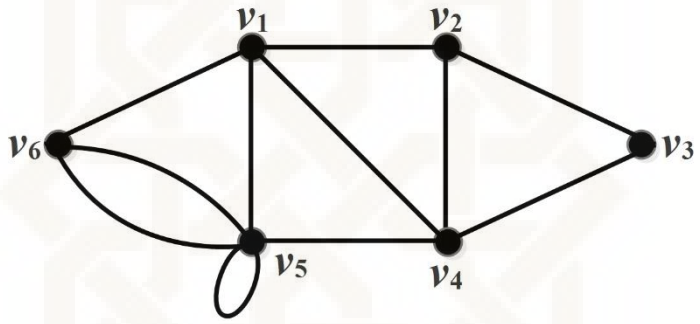
Sebuah *cycle* memiliki jumlah sisi paling sedikit ≥ 3. Berdasarkan jumlah sisinya sebuah *cycle* dibedakan menjadi dua yakni *cycle* genap dan *cycle* ganjil. Sebuah *cycle* disebut sebagai *cycle* genap apabila jumlah sisi

pada *cycle* tersebut berjumlah genap. Sedangkan sebuah *cycle* disebut sebagai *cycle* ganjil apabila jumlah sisi pada *cycle* tersebut berjumlah ganjil.

Contoh 2.15:

Pada Gambar 2.14 akan ditujukkan sebuah jalan, lintasan, dan *cycle* dari suatu graf $G_{17} = (V, E)$ dengan himpunan simpul $V = \{ v_1, v_2, v_3, v_4, v_5, v_6 \}$

G_{17} :



Gambar 2.14 Jalan terbuka, jalan tertutup, lintasan, *cycle* terbuka, dan *cycle* tertutup pada graf G_{15}

Jalan terbuka $v_1 - v_2$: $v_1 - v_4 - v_2 - v_3 - v_4 - v_2$

Jalan tertutup $v_2 - v_2$: $v_2 - v_4 - v_5 - v_1 - v_4 - v_2$

Trail terbuka $v_2 - v_1$: $v_2 - v_4 - v_5 - v_6 - v_5 - v_1$

Trail tertutup $v_1 - v_1$: $v_1 - v_5 - v_6 - v_5 - v_4 - v_1$

Lintasan $v_5 - v_3$: $v_5 - v_6 - v_1 - v_2 - v_4 - v_3$

Cycle genap $v_4 - v_4$: $v_4 - v_2 - v_1 - v_5 - v_4$

Cycle ganjil $v_5 - v_5$: $v_5 - v_1 - v_2 - v_3 - v_4 - v_5$

Semakin besar ukuran (*size*) suatu graf G maka semakin besar kemungkinan didapati sebuah jalan, *trail*, lintasan, *cycle*. (C. Vasudev, 2006:393)

2.5 Operasi pada Graf

Terdapat 4 macam operasi dasar pada graf:

1. Gabungan (*Union*) : Gabungan dua buah graf G_1 dan G_2 dinotasikan dengan $G = G_1 \cup G_2$ sedemikian sehingga

$$V(G_1 \cup G_2) = V(G_1) \cup V(G_2)$$

$$E(G_1 \cup G_2) = E(G_1) \cup E(G_2)$$

2. Irisan (*Intersection*) : Irisan dua buah graf G_1 dan G_2 dinotasikan dengan $G = G_1 \cap G_2$ sedemikian sehingga

$$V(G_1 \cap G_2) = V(G_1) \cap V(G_2)$$

$$E(G_1 \cap G_2) = E(G_1) \cap E(G_2)$$

3. Selisih (*Difference*) : Selisih dua buah graf G_1 dan G_2 dinotasikan dengan $G = G_1 \setminus G_2$ atau $G = G_1 - G_2$ atau $G = G_1 \sim G_2$ sedemikian sehingga

$$V(G) = V(G_1)$$

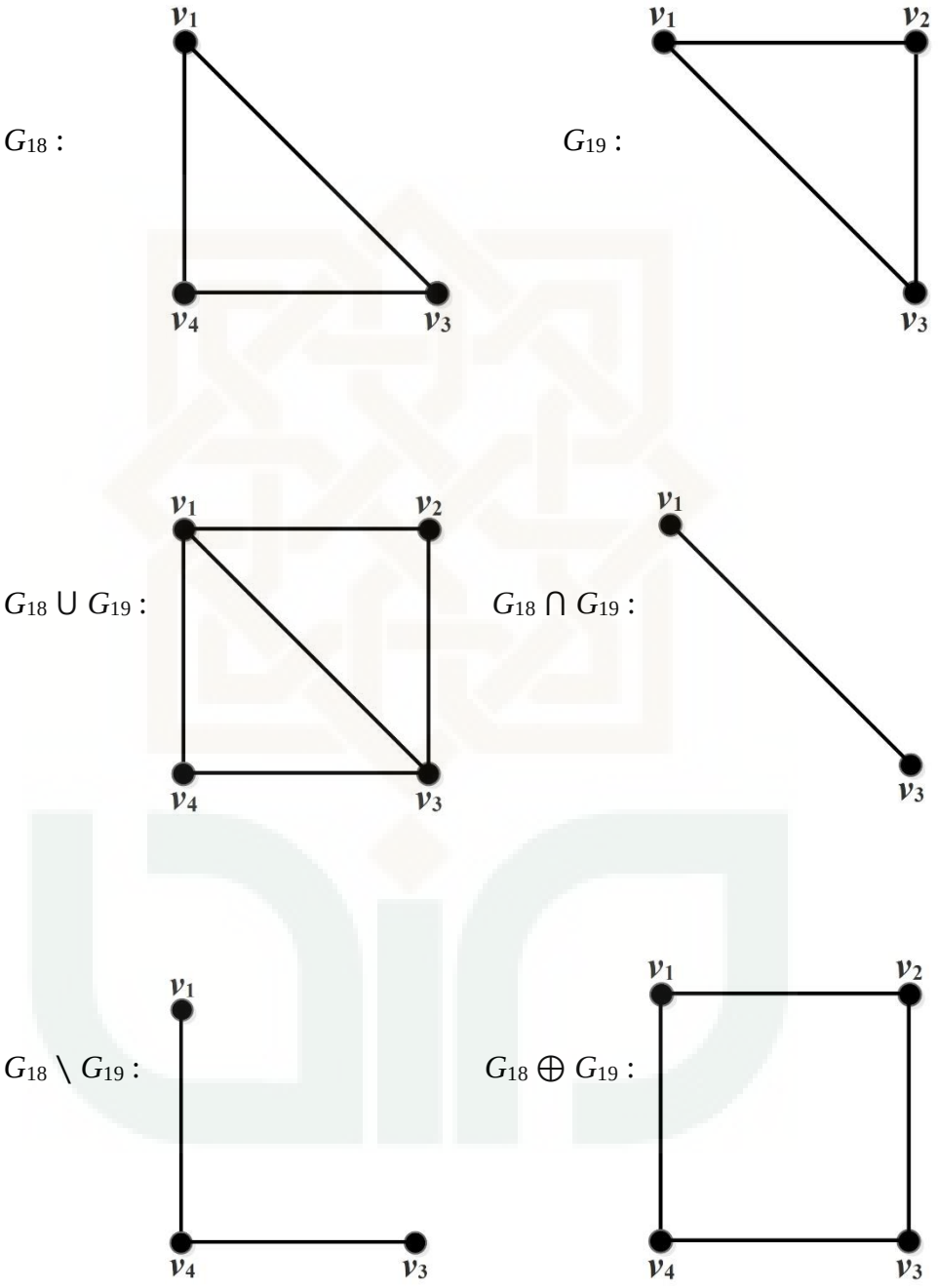
$$E(G) = E(G_1) - E(G_2)$$

4. *Symmetric Difference* (*Ring Sum*) : *Symmetric Difference* dari dua buah graf G_1 dan G_2 dinotasikan dengan $G = G_1 \oplus G_2$ sedemikian sehingga

$$V(G) = V(G_1) \cup V(G_2)$$

$$E(G) = E(G_1) \cup E(G_2) - E(G_1) \cap E(G_2)$$

Contoh 2.16:



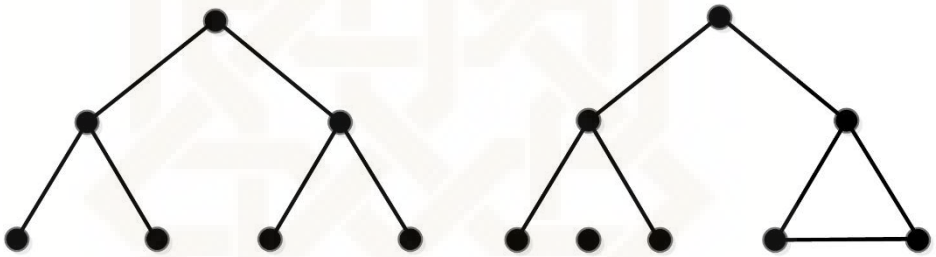
Gambar 2.15 Operasi pada graf G_{18} dan G_{19}

2.6 Pohon

Definisi 2.6.1: Pohon (Jack Edmonds, 1965:454)

Graf *T* disebut sebagai sebuah pohon jika graf *T* merupakan graf terhubung dan tidak memuat *cycle* serta memiliki n buah simpul dan n – 1 buah sisi.

Contoh 2.17:



(a) (b)

Gambar 2.16 (a) Graf pohon
(b) Bukan graf pohon

Pada Gambar 2.16(a) merupakan contoh dari sebuah graf pohon, sedangkan pada Gambar 2.16(b) bukan merupakan graf pohon karena memuat *cycle* dan bukan merupakan graf terhubung

Definisi 2.6.2: Pohon Merentang (Chartrand dan Lesniak, 1996:59)

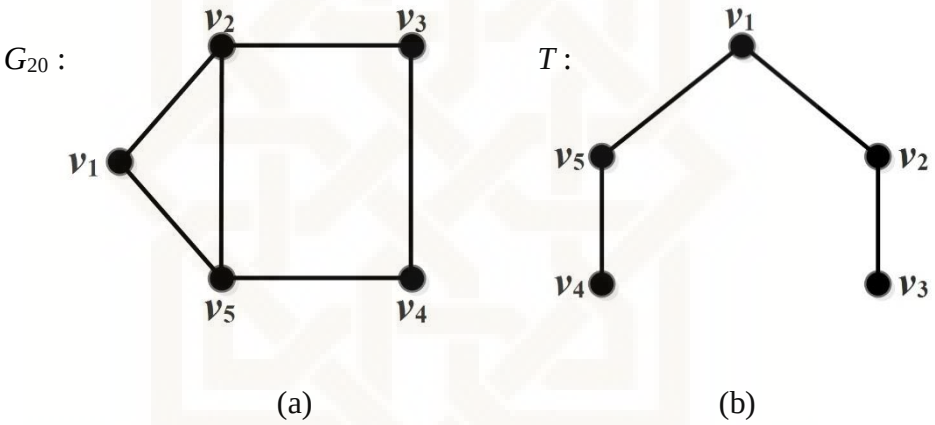
Sebuah pohon merentang *T* atas sebuah graf terhubung *G* merupakan sebuah pohon dengan akar *r* yang mencakup semua simpul pada graf terhubung *G*.

Sebuah akar pada pohon merentang atas suatu graf *G* merupakan sebuah simpul di *G* yang dipilih secara acak sebagai simpul awal penyusunan pohon

merentang atas suatu graf *G* dan dinotasikan sebagai *r*. (C. Vasudev, 2006: 193)

Contoh 2.18:

Diberikan sebuah graf terhubung G_{20} dengan $V = \{v_1, v_2, v_3, v_4, v_5\}$



Gambar 2.17 (a) Graf terhubung G_{20}
(b) Pohon *T* atas graf G_{20}

Pada Gambar 2.17(b) merupakan sebuah contoh pohon merentang *T* atas graf terhubung G_{20}. Simpul v_1 pada graf G_{20} dipilih secara acak sebagai sebuah simpul awal penyusunan pohon merentang *T* yang dinotasikan sebagai $r = v_1$ dengan $v_1 \in V(G_{20})$.

Sebuah pohon merentang atas sebuah graf *G* dapat dikontruksi dengan menggunakan algoritma BFS (*Breadth – First Search*).

2.7 Algoritma *Breadth – First Search* (BFS)

Algoritma *Breadth – First Search* (BFS) merupakan algoritma yang digunakan untuk menyusun pohon merentang atas sebuah graf terhubung. Ide dasar algoritma BFS yakni mengunjungi seluruh simpul pada sebuah graf terhubung dimana untuk simpul yang dikunjungi pertama kali diberi label (*layer*) sebelum kemudian melanjutkan mengunjungi simpul yang lain. Proses dihentikan apabila semua simpul pada sudah memiliki label (*layer*). Berikut prosedur algoritma BFS: (C. Vasudev, 2006:235)

Pilih sebuah simpul (sebarang) kemudian jadikan sebagai akar *r* pada pohon merentang *T* (akar *r* menempati *layer* 0 pada *T*).

(i) Tambahkan semua simpul baru (belum termuat pada *T*) $a_1, a_2, a_3, \cdots, a_n$ yang mana *adjacent* dengan akar *r* sedemikian sehingga penambahan simpul yang dilakukan tidak menghasilkan *cycle*.

(ii) Untuk simpul baru pada langkah (i) akan menempati *layer* 1 pada *T*.

(iii) Untuk *layer* 2 tambahkan semua simpul baru $c_1, c_2, c_3, \cdots, c_n$ pada *T* dimana c_i *adjacent* dengan minimal satu a_i untuk (i = 1,2,3, $\cdots$,n) sedemikian sehingga penambahan simpul yang dilakukan tidak menghasilkan *cycle*.

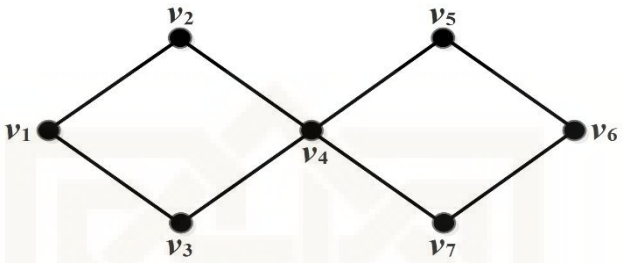
(iv) Ulangi langkah yang sama hingga semua simpul termuat pada pohon merentang *T*.

(v) Proses dihentikan jika semua simpul sudah termuat pada pohon merentang *T*.

Contoh 2.19:

Gunakan algoritma BFS untuk menyusun pohon merentang T atas graf G_{21}

G_{21} :



Gambar 2.18 Graf G_{21}

Solusi: Pilih sebuah simpul sebagai akar, katakan $r = v_1$

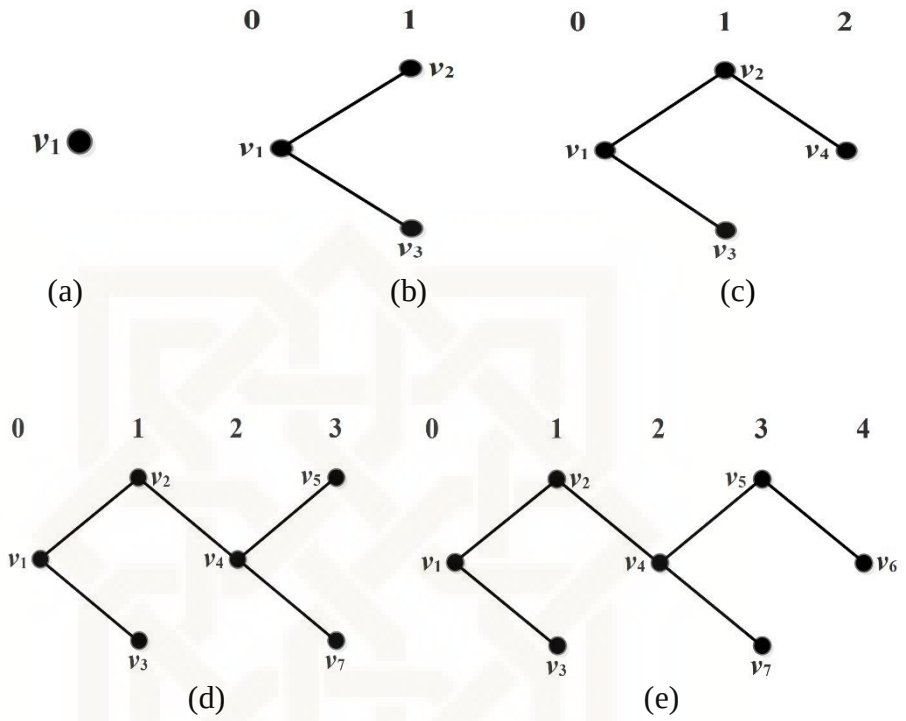
(i) Tambahkan semua simpul baru yang mana *adjacent* dengan simpul v_1 yakni simpul v_2 dan v_3 guna menempati *layer* 1 pada T

(ii) Pada *layer* 2 tambahkan simpul baru pada T yakni simpul v_4 yang mana *adjacent* dengan simpul v_2 dan v_3 sedemikian sehingga tidak menghasilkan *cycle*

(iii) Tambahkan simpul baru yang mana *adjacent* dengan simpul v_4 yakni simpul v_5 dan v_7 guna menempati *layer* 3 pada T.

(iv) Tambahkan simpul v_6 (*adjacent* dengan simpul v_5 dan v_7) pada T guna menempati *layer* 4

(v) Karena semua simpul sudah termuat pada T maka proses dihentikan



Gambar 2.19 Pohon merentang T atas graf G_{21}

2.8 *Matching*

Definisi 2.8.1: *Matching* (Gross dan Yellen, 1999:1103)

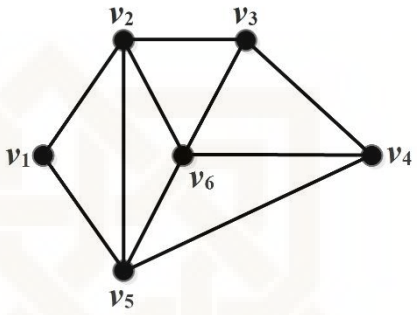
Matching M pada graf $G = (V, E)$ adalah himpunan sisi $M \subseteq E(G)$ sedemikian sehingga tidak terdapat dua sisi pada M yang bertemu pada simpul yang sama.

Sisi *matching* pada graf G merupakan sisi yang termuat ke dalam M serta disimbolkan sebagai garis tebal, sedangkan sisi bukan *matching* (tidak termuat ke dalam M) disimbolkan dengan garis tipis. Sebuah graf G yang

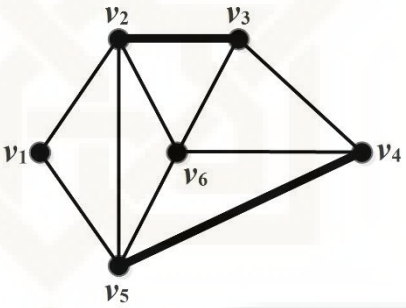
mana berpasangan dengan *matching M* dimana $M \subseteq E(G)$ dinotasikan sebagai (G, M). (Jack Edmonds, 1965:453)

Contoh 2.20:

(a) G_{22} :



(b) (G_{22}, M) :



Gambar 2.20 (a) Graf G_{22}
(b) Graf (G_{22}, M)

Pada Gambar 2.20(b) *matching M* dengan $M = \{v_2v_3, v_4v_5\}$ merupakan *matching* yang dibentuk dari graf G_{22}. Banyaknya jumlah sisi pada M dinamakan kardinalitas *matching* dan dinotasikan sebagai $|M|$.

Contoh 2.21:

Pada Gambar 2.20(b) graf (G_{22}, M) memiliki kardinalitas *matching* sebanyak $|M| = 2$

Simpul pada graf (*G*, *M*) dibedakan menjadi dua yakni:

1. Simpul *Saturated*

   Simpul *saturated* atau simpul *covered* atau simpul *matched* adalah simpul yang *incident* dengan sisi pada *matching M*. (L. Lovasz dan M. D. Plummer, 1986: 357)

   Contoh 2.22:

   Pada Gambar 2.20(b) simpul v_2, v_3, v_4, v_5 merupakan simpul *saturated* karena *incident* dengan sisi pada *matching M*.

2. Simpul *unsaturated*

   Simpul *unsaturated* atau simpul *uncovered* atau simpul *exposed* adalah simpul yang tidak *incident* dengan setiap sisi pada *matching M*. (L. Lovasz dan M. D. Plummer, 1986: 357)

   Contoh 2.23:

   Pada Gambar 2.20(b) simpul v_1 dan simpul v_6 merupakan simpul *unsaturated* karena tidak *incident* dengan setiap sisi pada *matching M*.

Selain simpul suatu lintasan pada graf (*G*, *M*) juga dibedakan menjadi dua yakni:

1. Lintasan *Alternating*

   Lintasan *alternating* atau *M-alternating* merupakan lintasan pada graf (*G*, *M*) dengan sisi – sisinya bergantian antara sisi yang termuat pada *M* dan sisi yang tidak termuat pada *M*. (Dieter Jungnickel, 2008:390)

Contoh 2.24:

Pada Gambar 2.20(b) lintasan $P : v_1 – v_2 – v_3 – v_4 – v_5$ merupakan contoh lintasan *alternating* pada graf (G_{22}, M)

Gambar 2.21 Lintasan *alternating* pada graf (G_{22}, M)

2. Lintasan *Augmenting*

Lintasan *augmenting* atau *M-augmenting* merupakan lintasan pada graf (G, M) dengan simpul awal dan simpul akhirnya merupakan simpul *exposed*. (Dieter Jungnickel, 2008:390)

Contoh 2.25:

Pada Gambar 2.20(b) lintasan $P : v_1 – v_2 – v_3 – v_4 – v_5 – v_6$ merupakan lintasan *augmenting* pada graf (G_{22},M) dimana simpul v_1 dan v_2 merupakan simpul *exposed*

Gambar 2.22 Lintasan *augmenting* pada graf (G_{22}, M)

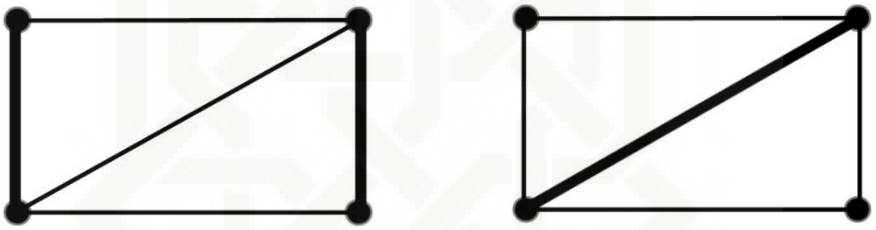
2.9 *Matching* Maksimum

Definisi 2.9.1: *Matching* Maksimum (Bondy dan Murty, 1976: 70)

M disebut *matching* maksimum pada suatu graf G apabila tidak terdapat *matching* M' dengan $|M'| > |M|$.

Matching maksimum pada suatu graf *G* juga dapat dipandang sebagai *matching M* yang mana memiliki kardinalitas terbesar pada suatu graf *G*. (Dieter Jungnickel, 2008: 387)

Contoh 2.26:



(a) (b)

Gambar 2.23 (a) Graf dengan *matching* yang sudah maksimum
(b) Graf dengan *matching* yang belum maksimum

Dari Contoh 2.26 terlihat bahwa pada Gambar 2.23(a) merupakan contoh graf dengan *matching* yang sudah maksimum karena *matching* tersebut memiliki jumlah kardinalitas terbesar yakni 2. Gambar 2.23(b) merupakan contoh graf dengan *matching* yang belum maksimum.

2.10 Algortima Kardinalitas *Matching Edmonds*

Pencarian *matching* maksimum pada graf *non-bipartite* sedikit berbeda dibandingkan dengan pencarian *matching* maksimum pada graf *bipartite*. Dimana pada kasus graf *non-bipartite* setidaknya paling sedikit termuat

sebuah *cycle* dengan panjang ganjil yang mana disebut sebagai "*blossom*". Jack Edmonds (1965) merupakan orang yang pertama kali menggagas sebuah algoritma untuk pencarian *matching* maksimum pada graf *non-bipartite*.(L. Lovasz dan M. D. Plummer, 1986: 357)

Ide utama dalam algoritma kardinalitas *matching Edmonds* adalah proses penyusutan atau "*shrinking*" beberapa *cycle* dengan panjang ganjil yang ditemui pada kasus *non-bipartite* graf menjadi sebuah simpul baru yang disebut sebagai *pseudovertex*. Proses penyusutan suatu *blossom B* pada graf *G* direpresentasikan sebagai pemendekan terhadap graf *G* atas *blossom B* sedemikian sehingga menghasilkan graf baru yang lebih kecil yakni G / B yang mana didefinisikan sebagai berikut: (Dieter Jungnickel, 2008: 400)

- Himpunan simpul di G / B adalah $V / B = (V \setminus B) \cup \{b\}$, dimana b adalah sebuah simpul baru yang disebut sebagai *pseudovertex* ($b \notin V$).
- Himpunan sisi di G / B adalah E / B yang diperoleh dari himpunan sisi E di G serta menghilangkan semua sisi $uv \in E$, $u \in B$ atau $v \in B$, kemudian menambahkan sebuah sisi ub untuk semua $u \in V \setminus B$ di G yang mana *adjacent* dengan paling sedikit satu simpul di B.

Berikut merupakan langkah – langkah algoritma kardinalitas *matching Edmonds* : (R. Evans dan E. Minieka, 2006: 246)

Langkah 1 (*inisialisasi*)

Graf *G* merupakan *non-bipartite* graf dengan *matching* $M_0 = Ø$. Lakukan inisialisasi *matching* pada graf *G* dengan menggunakan algoritma *greedy*.

Langkah 2 (*pemeriksaan atas sebuah exposed vertex*)

2.1 Jika semua simpul di (*G, M*) *incident* dengan sebuah sisi *matching*, maka lanjutkan ke langkah 5.

2.2 Jika hanya tersisa satu simpul *exposed* di (*G, M*), maka lanjutkan ke langkah 5.

2.3 Jika sebaliknya, pilih sembarang simpul *exposed r* di (*G, M*) sebagai akar dan susun pohon *alternating T*:

2.3.1 Jika pohon *alternating T* berakhir dengan sebuah simpul *exposed s* dengan $s \neq r$, maka telah ditemukan sebuah lintasan *augmenting* dari *r* ke *s*. Lanjutkan ke langkah 3.

2.3.2 Jika pada pohon *alternating T* dijumpai sebuah *cycle* dengan panjang ganjil, hentikan pencarian dan lanjutkan ke langkah 4.

Langkah 3 (*Augmenting Path*) *:*

3.1 Jika lintasan *augmenting P* yang diperoleh dari pohon *alternating T* memuat *pseudovertex* b_i (i = 1,2,3...) maka rentangkan *pseudovertex* b_i menjadi *blossom* B_i. Dengan menggunakan lintasan *P* lakukan perunutan dimulai dari simpul *exposed s* menuju simpul akar *r* melewati lintasan *alternating* dengan panjang genap pada *blossom* B_i. Lakukan

langkah serupa hingga diperoleh lintasan *augmenting* di (*G, M*) yang mana tidak memuat satupun *pseudovertex* b_i. Lakukan proses *augmenting* terhadap *matching M* di (*G, M*) dengan $M' = M \oplus P$. Lanjutkan ke langkah 3.2.

3.2 Jika lintasan *augmenting P* yang diperoleh dari pohon *alternating T* sudah tidak memuat *pseudovertex* b_i (i = 1,2,3...) maka lakukan proses *augmenting* terhadap *matching M* di (*G, M*) dengan $M' = M \oplus P$ kemudian kembali ke langkah 2.

Langkah 4 (*Cycle Ganjil*)

Bagian ini dapat dicapai hanya apabila pohon *alternating T* menemukan sebuah *cycle* dengan panjang ganjil atau disebut sebagai *blossom*. Misalkan ditemui sebuah *blossom* B_i (i = 1,2,3, ...) pada proses pembentukan pohon *alternating T* maka lakukan penyusutan pada *blossom* B_i menjadi sebuah *pseudovertex* b_i. Kemudian lanjutkan penyusunan pohon *alternating T* dengan akar *r* sebagaimana pada langkah 2.

Langkah 5

Matching pada graf *G* sudah maksimum. Hentikan proses pencarian.

2.11 *The Battle of Britain* (Peng Koen Ojong, 2003: 130)

Pertempuran di Britania Raya merupakan bagian dari perang dunia ke II dimana pertempuran tersebut melibatkan angkatan udara Jerman yakni

Luftwaffe dengan angkatan udara Inggris yakni *Royal Air Force* pada tahun 1940. Latar belakang pertempuran ini didasari oleh keinginan Hitler untuk menyerang Rusia. Hitler mengira bahwa setelah runtuhnya Perancis oleh Jerman maka Inggris akan bersedia berdamai.

Akan tetapi tawaran dari Hitler ditolak mentah – mentah oleh rakyat Inggris sehingga hal ini membuat Hitler murka. Pada tanggal 16 Juli 1940 Hitler memerintahkan untuk melakukan invasi terhadap Inggris. Laksamana Raeder sebagai pimpinan tertinggi angkatan laut Jerman berkata kepada Hitler bahwa invasi melalui jalur laut atau dikenal sebagai operasi singa laut tidak dapat segera dilakukan dikarenakan masalah cuaca yang buruk serta tidak memiliki cukup kapal dimana dua kapal Jerman yang paling modern yakni *Gneisenau* dan *Scharnhorst* mengalami kerusakan akibat torpedo sehingga untuk sementara tidak dapat digunakan.

Pada hari terakhir bulan Juli 1940 Hitler mengambil keputusan apakah operasi singa laut tetap dilaksanakan atau tidak akan ditentukan setelah *Luftwaffe* menyerang Inggris. Bila serangan *Luftwaffe* dapat melemahkan angakatan udara Inggris *Royal Air Force* maka operasi singa laut tetap dilaksanakan apabila gagal maka sebaliknya.

Jerman menyediakan tak kurang dari tiga armada udara meliputi 963 pesawat tempur dan 1311 yang terdiri dari jenis Heinkel He-111 H (pembom medium), Junkers JU-88 (pembom cepat), Do-17Z (pembom ringan), Messerchmitt Bf-109 (pemburu) dan Junkers JU-87 Stuka (pembom tukik)

sedangkan Inggris hanya menyediakan paling banyak 800 pesawat tempur yang terdiri dari Spitfire, Hawker Hurricane dan Bristol Beufighter.

Awalnya *Luftwaffe* memusatkan serangan pada pelabuhan dan area perkapalan. Akan tetapi lambat laun serangan yang diluncurkan *Luftwaffe* membabi buta sehingga kota london dihujani bom oleh *Luftwaffe* pada waktu itu. Meski *Royal Air Force* kalah jumlah pada peperangan di langit Inggris akan tetapi berkat kehebatan pilot Inggris dalam pertempuran serta didukung oleh radar-radar Inggris yang pada waktu itu lebih unggul dibanding Jerman dan juga kegigihan rakyat Inggris akhirnya dataran Inggris dapat dipertahankan dari serangan Jerman. Jerman menderita banyak kerugian hampir 1733 pesawat jerman hancur, sebagian pesawat pembom. Dengan banyaknya kerugian yang diderita Jerman akhirnya Hitler memutuskan untuk menghentikan operasi singa laut. Pertempuran yang dijuluki sebagai *The Battle of Britain* ini disebut sebagai pertempuran paling heroik dalam sejarah perang duni ke 2 serta banyak pujian diberikan kepada *Royal Air Force* dan juga rakyat Inggris atas keberhasilan mempertahankan wilayahnya.

BAB III

PEMBAHASAN

3.1 Teorema Subgraf Perentang dan Lintasan *Augmenting*

Teorema 3.1 dan Teorema 3.2 merupakan landasan yang digunakan oleh algoritma kardinalitas *matching Edmonds* dalam melakukan pencarian *matching* maksimum pada suatu graf (Dieter Jungnickel, 2008: 393).

Teorema 3.1: Subgraf Perentang (Chartrand dan Lesniak, 1996: 259)

M_1 dan M_2 merupakan *matching* pada suatu graf G. Misal H merupakan suatu subgraf perentang atas graf G maka tiap - tiap komponen pada subgraf perentang H atas graf G dengan $E(H) = (M_1 - M_2) \cup (M_2 - M_1)$ memenuhi salah satu dari tipe berikut:

(i) Sebuah simpul terasing (*isolated vertex*),

(ii) Sebuah *cycle* genap yang mana sisi pada *cycle* tersebut bergantian pada M_1 dan M_2,

(iii) Sebuah lintasan yang mana sisi pada lintasan tersebut bergantian pada M_1 dan M_2 sedemikian sehingga tiap simpul terakhir pada lintasan tersebut *unsaturated* pada M_1 atau M_2 tetapi tidak pada keduanya.

Bukti:

(i) Ingat bahwa H memiliki derajat maksimum $\Delta(H) \leq 2$ karena jika H memuat sebuah simpul v yang mana memiliki $\deg_H v \geq 3$ maka v akan

incident dengan paling sedikit dua sisi pada *matching* yang sama oleh karna itu $\Delta H \leq 2$. Jika simpul $v \notin S\,(M_1 - M_2)$ dan $v \notin S\,(M_2 - M_1)$ dengan $S \subseteq V(H)$, maka v adalah sebuah simpul terasing (*isolated vertex*).

(ii) Karena tidak memungkinkan dua buah sisi pada suatu *matching* saling *adjacent*, oleh karna itu tiap sisi pada *cycle* tersebut saling bergantian pada M_1 dan M_2 sedemikian sehingga hal ini mengakibatkan *cycle* yang terbentuk pada H adalah genap.

(iii) Misal terdapat suatu sisi $e = uv$ pada H dan u adalah simpul akhir pada sebuah lintasan P dimana P merupakan komponen pada H. Akan ditunjukkan bahwa u adalah sebuah simpul *unsaturated* pada M_1 atau pada M_2 tetapi tidak pada keduanya.

Karena $e \in E(H)$, berarti bahwa $e \in M_1 - M_2$ atau $e \in M_2 - M_1$.

Jika $e \in M_1 - M_2$ maka u *saturated* pada M_1. Akan ditunjukkan bahwa u *unsaturated* pada M_2. Misal terdapat sebuah sisi f pada M_2 $(f \neq e)$ sedemikian sehingga f *incident* dengan u. Karena e dan f saling *adjacent*, jelas bahwa $f \notin M_1$ dan $f \in M_2 - M_1 \subseteq E(H)$. Hal ini mengakibatkan bahwa f tidak mungkin *incident* dengan u karena u merupakan simpul akhir pada lintasan P (kontradiksi). Oleh karena itu u *unsaturated* pada M_2.

Jika $e \in M_2 - M_1$ maka u *saturated* pada M_2. Akan ditunjukkan bahwa u *unsaturated* pada M_2. Misal terdapat sebuah sisi f pada M_1 $(f \neq e)$ sedemikian sehingga f *incident* dengan u. Karena e dan f saling

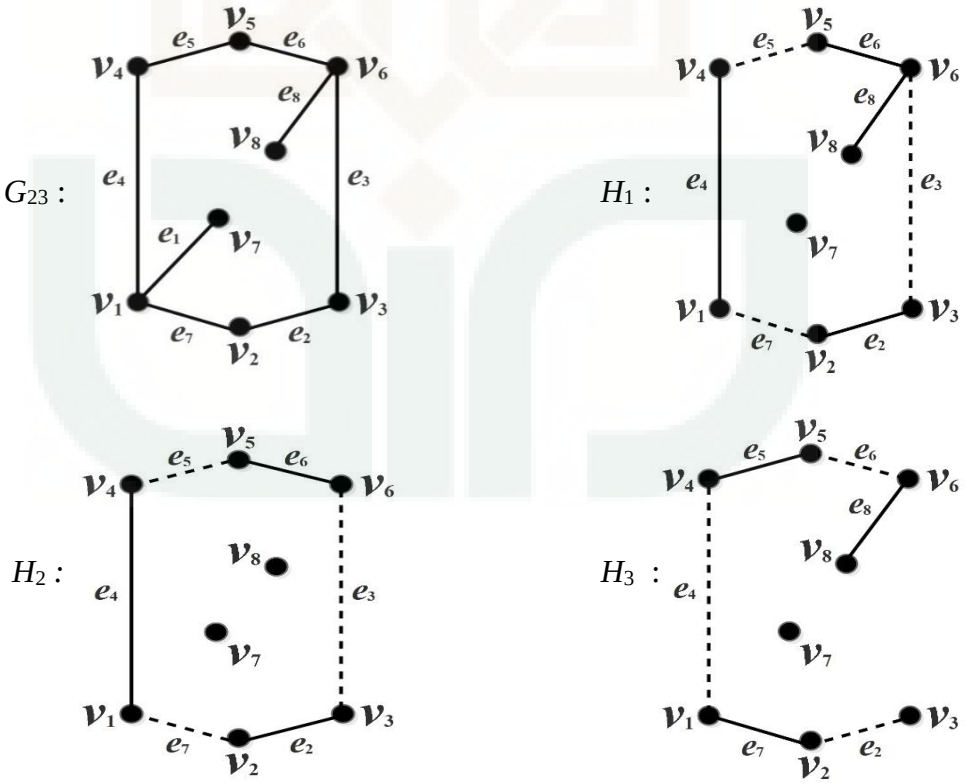
adjacent, jelas bahwa $f \notin M_2$ dan $f \in M_1 - M_2 \subseteq E(H)$. Hal ini mengakibatkan bahwa f tidak mungkin *incident* dengan u karena u merupakan simpul akhir pada lintasan P (kontradiksi). Oleh karna itu u *unsaturated* pada M_1. ■

Diberikan sebuah ilustrasi dari Teorema 3.1. sebagai berikut:

$deg(v_6) \geq 2$, e_6 *adjacent* sisi dengan e_8 pada *matching* $M_2 = \{e_6, e_4, e_2, e_8\}$

$E(H_2) = (M_1 - M_2) \cup (M_2 - M_1)$, $M_1 = \{e_3, e_5, e_7\}$ dan $M_2 = \{e_6, e_4, e_2\}$ di H_2

$E(H_3) = (M_1 - M_2) \cup (M_2 - M_1)$, $M_1 = \{e_8, e_5, e_7\}$ dan $M_2 = \{e_6, e_4, e_2\}$ di H_3



Gambar 3.1 H_1 dan H_2 merupakan subgraf perentang dari G_{23}

Subgraf perentang H_1 merupakan ilustrasi kondisi (i). Subgraf perentang H_2 terdiri dari dua komponen dimana komponen pertama merupakan *cycle* genap sedangkan komponen kedua merupakan simpul terasing yakni simpul v_7 dan simpul v_8. Sedangkan pada subgraf perentang H_3 juga terdiri dari dua komponen dimana komponen pertama merupakan lintasan dengan kedua simpul ujungnya berada pada M_1 atau M_2 tetapi tidak pada keduanya dan komponen keduanya merupakan simpul terasing yakni simpul v_7.

Teorema 3.2: Lintasan *Augmenting* (Dieter Jungnickel, 2008: 391)

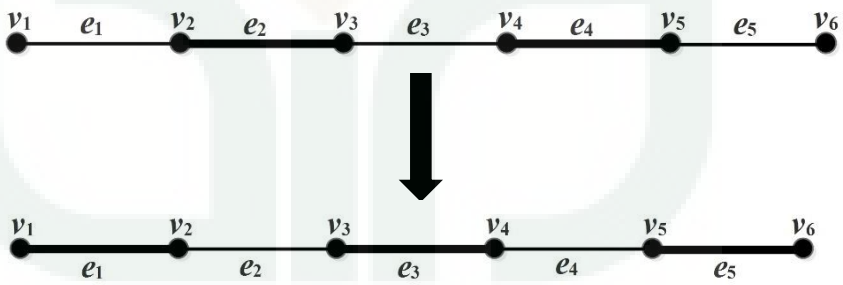
Suatu *matching M* pada graf *G* dikatakan maksimum jika dan hanya jika *G* yang berpasangan dengan *M* tidak memuat lintasan *augmenting.*

Bukti:

($\Longrightarrow$)Misal diasumsikan *M* maksimum pada graf *G*. Andaikan terdapat sebuah lintasan *augmenting P* di graf G, serta dilakukan *augmenting M* dengan $M' = M \oplus P$. Maka *M'* merupakan *matching* maksimum karena memiliki kardinalitas yang lebih besar dibanding *M* yakni $|M'| = |M| + 1$. Hal ini kontradiksi dengan asumsi bahwa *M* adalah *matching* maksimum pada graf *G* sehingga pengandaian harus diingkari.

($\Longleftarrow$)Diasumsikan bahwa graf *G* tidak memuat lintasan *augmenting*. Andaikan *M* bukan merupakan *matching* maksimum pada graf *G* maka *M'* merupakan *matching* maksimum pada graf *G* dengan $|M'| > |M|$.

Misal H subgraf perentang dari graf G dan $E(H) = (M - M') \cup (M' - M)$. Setiap simpul di H memiliki derajat ≤ 2 karena untuk sebuah simpul v di H yang mana memiliki derajat 2 paling banyak hanya dapat *incident* dengan satu sisi pada M dan satu sisi yang lain pada M'. Menurut teorema 3.1 maka setiap komponen H merupakan salah satu dari ketiga tipe pada teorema tersebut. Untuk tipe (i) dan (ii) jumlah sisi atas dua *matching* M' dan M adalah sama yakni $|M'| = |M|$. Pada tipe (iii) jumlah sisi atas dua *matching* M' dan M tidak sama karena $|M'| > |M|$ sehingga paling sedikit terdapat satu lintasan P yang mana sisi awal dan akhirnya termuat pada M' dengan kata lain P *augmenting* terhadap M. Hal ini kontradiksi dengan asumsi bahwa graf G tidak memuat lintasan *augmenting* untuk M sehingga pengandaian harus diingkari. ■



Gambar 3.2 *Augmenting-M* sepanjang P

Gambar 3.2 merupakan ilustrasi dari Teorema 3.2. Misal $M = \{e_2, e_4\}$ merupakan *matching* pada G dengan P : $v_1 — v_2 — v_3 — v_4 — v_5 — v_6$

merupakan lintasan *augmenting* pada (*G, M*) maka $M' = \{e_1, e_3, e_5\}$ adalah *matching* yang diperoleh dari $M' = M \oplus P = \left| M \right| + 1$.

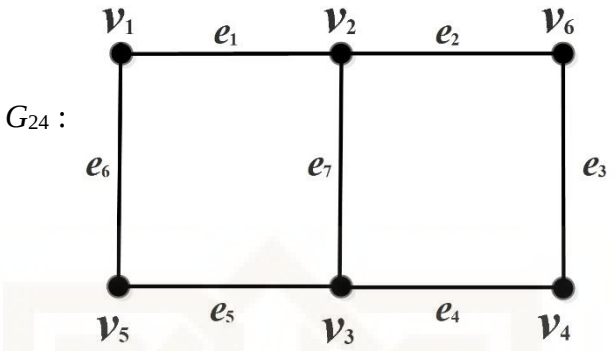
3.2 Inisialisasi *Matching*

Pada umumnya pencarian suatu *matching* maksimum *M* pada suatu graf sukar dilakukan seiring dengan bertambahnya jumlah simpul (*vertices*) maupun sisi (*edges*) pada suatu graf. Untuk itu dapat digunakan sebuah algoritma *greedy* sederhana untuk membantu dalam melakukan inisialisasi *matching* guna memperoleh *matching* dengan jumlah kardinalitas terbesar pada pencarian *matching* maksimum. Algoritma *greedy* merupakan metode sederhana untuk memperoleh solusi optimal lokal dengan harapan menemukan solusi optimum global. (Dieter Jungnickel, 2008: 395)

Berikut merupakan langkah – langkah algoritma *greedy* sederhana dalam proses inisialisasi *matching* pada suatu graf *G* : (Winter, 2005:2)

1. $M = \emptyset$
2. Jika sudah tidak didapati lagi garis *e* yang dapat ditambahkan kedalam *M* maka berhenti
3. Jika sebaliknya maka pilih sebarang garis *e* yang tidak memiliki simpul ujung yang sama dengan garis – garis yang termuat pada *M*
   3.1 $M = (M \cup e)$
4. Kembali ke *M*.

Contoh 3.1:

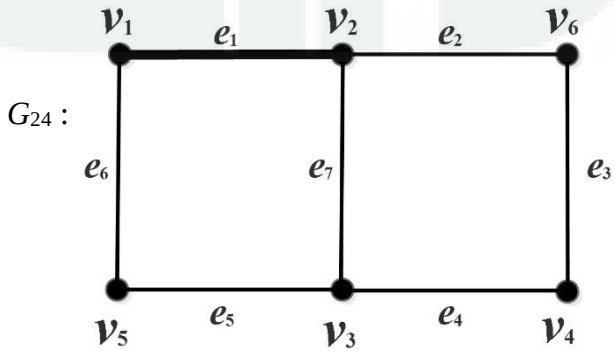


Gambar 3.3 Graf G_{24} dengan 6 simpul dan 7 sisi

Dari Gambar 3.3 diatas proses inisialisasi *matching* menggunakan algoritma *greedy* adalah sebagai berikut:

Iterasi 1:

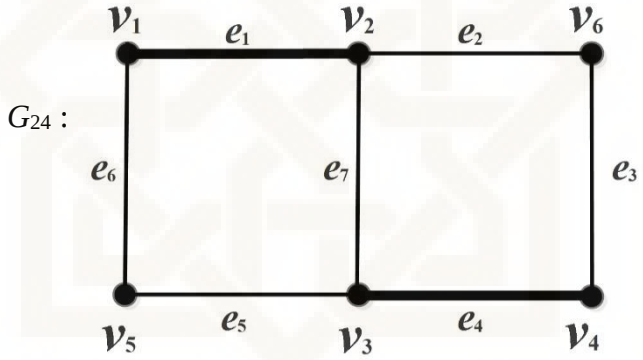
1. $M = \emptyset$
2. Dipilih garis $e_1 = v_1v_2$ untuk ditambahkan kedalam *matching M.*
3. $M = M \cup \{e_1\}$
4. Kembali ke M.



Gambar 3.4 Graf G_{24} dengan *matching* $M = \{e_1\}$

Iterasi 2:

1. $M = \{e_1\}$
2. Dipilih garis $e_4 = v_3v_4$ untuk ditambahkan kedalam *matching M*.
3. $M = \{e_1\} \cup \{e_2\}$
4. Kembali ke *M*.



Gambar 3.5 Graf G_{24} dengan *matching* $M = \{e_1, e_4\}$

Iterasi 3:

1. $M = \{e_1, e_4\}$
2. Hentikan proses inisialisasi *matching*.

Karena graf G_{24} pada Gambar 3.5 sudah tidak terdapat garis *e* yang dapat ditambahkan kedalam *matching M* maka proses inisialisasi *matching* terhadap graf G_{24} dihentikan. Dari proses inisialisasi terhadap graf G_{24} diperoleh *matching* $M = \{e_1, e_4\}$ pada G_{24}.

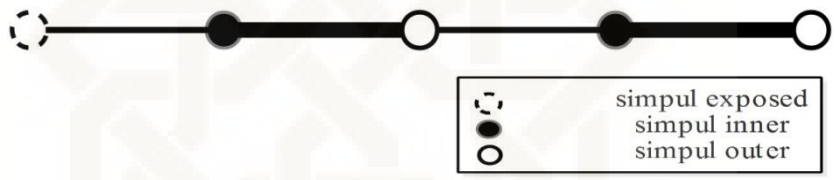
3.3 *Blossom*

3.3.1 Batang, *Blossom*, Bunga pada suatu Graf ber*matching*

Definisi 3.3: Batang, *Blossom*, Bunga (Jack Edmonds, 1965: 455)

Sebuah batang (*stem*) pada (*G*, *M*) adalah sebuah lintasan *alternating* dengan sebuah simpul *exposed* pada salah satu ujungnya dan sebuah sisi *matching* pada ujung yang lain.

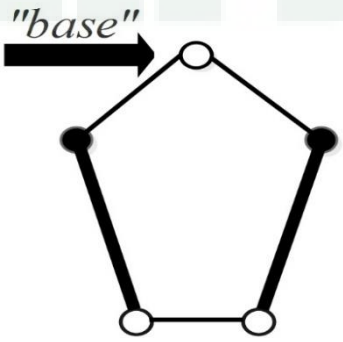
Contoh 3.2:



Gambar 3.6 Batang (*Stem*)

Sebuah *Blossom B* pada (*G*, *M*) adalah sebuah *cycle* ganjil dengan $M \cap B$ merupakan *matching* maksimum di *B* dengan *base* merupakan simpul *exposed* atau simpul *outer* untuk $M \cap B$.

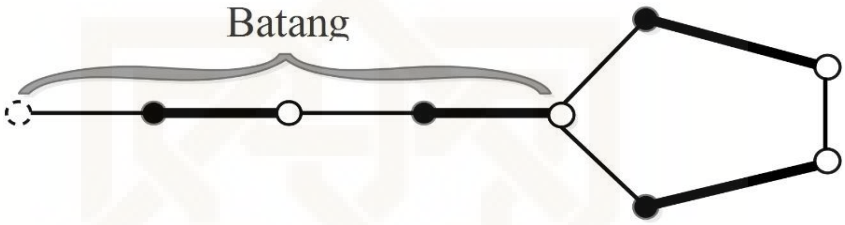
Contoh 3.3:



Gambar 3.7 *Blossom*

Sebuah Bunga (*flower*) terdiri atas sebuah *blossom* dan sebuah batang (*Stem*) dimana keduanya berpotongan hanya pada ujung batang yang mana sisi pada batang tersebut merupakan sisi matching atau pada *base* suatu *blossom*.

Contoh 3.4:

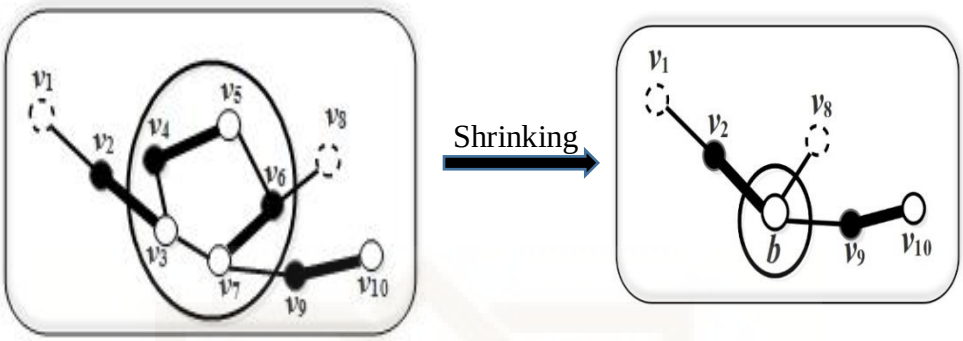


Gambar 3.8 Bunga (*Flower*)

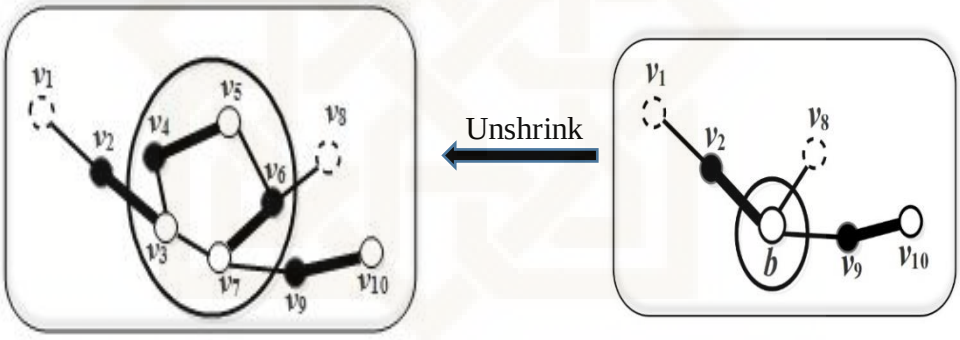
3.3.2 Penyusutan *Blossom* (*Shrinking Blossom*)

Penyusutan sebuah *Blossom* (*Srinking Blossom*) merupakan proses penyusutan terhadap *cycle* ganjil pada (*G, M*) dengan cara menggantikan semua simpul dan semua sisi pada *blossom B* dengan sebuah simpul berlabel katakan *b* dengan *b = B / B*. Simpul *b* disebut sebagai *pseudovertex*. Selama proses perentangan terhadap *b* belum dilakukan (dengan cara menghapus label *b* dan menggantikannya dengan *B* seperti semula), abaikan semua simpul yang termuat dalam *B* dan juga abaikan semua sisi yang menghubungkan simpul satu dengan yang lain dalam *B*. Sedangkan proses perentangan (*Unshrink*) terhadap *b* merupakan proses pengembalian semua simpul dan semua sisi yang termuat pada *blossom B* seperti semula.(Jack Edmonds, 1965: 457)

Contoh 3.5:



Gambar 3.9 Proses Penyusutan *Blossom* (*Shrinking Blossom*)



Gambar 3.10 Proses perentangan *pseudovertex b* (*Unshrink b*)

Teknik penyusutan terhadap sebuah *blossom* (*Shrinking* blossom) merupakan ide yang digagas oleh Jack Edmonds (1965) sebagai sebuah metode yang dapat digunakan dalam proses pencarian *matching* maksimum pada graf sederhana tak berarah, berhingga dan terhubung. Berikut merupakan lemma *cycle shrinking blossom* yang mana menjadi dasar bahwasanya teknik penyusutan sebuah *blossom* dapat digunakan dalam proses pencarian *matching* maksimum pada graf sederhana tak berarah, berhingga dan terhubung.

Lemma 3.3.1: (*Shrinking Cycle*) (Dieter Jungnickel, 2008: 407)

Misal *G* adalah sebuah graf dengan *matching M*, dan misalkan *r* suatu simpul *exposed* pada (*G*, *M*). Anggap bahwa selama proses pembentukan suatu pohon *alternating T* dengan akar *r*, ditemui *blossom B*. Misal simpul *w* merupakan *base* dari *blossom B* dan graf *G' = G / B* adalah graf yang dihasilkan dari penyusutan *blossom B* menjadi *pseudovertex b*. Graf (*G, M*) akan memuat sebuah lintasan *augmenting* yang dimulai dari simpul *r* jika dan hanya jika graf (*G',M'*) dengan *matching M' = M / B* memuat sebuah lintasan *augmenting* yang dimulai dari simpul *r*.

Bukti:

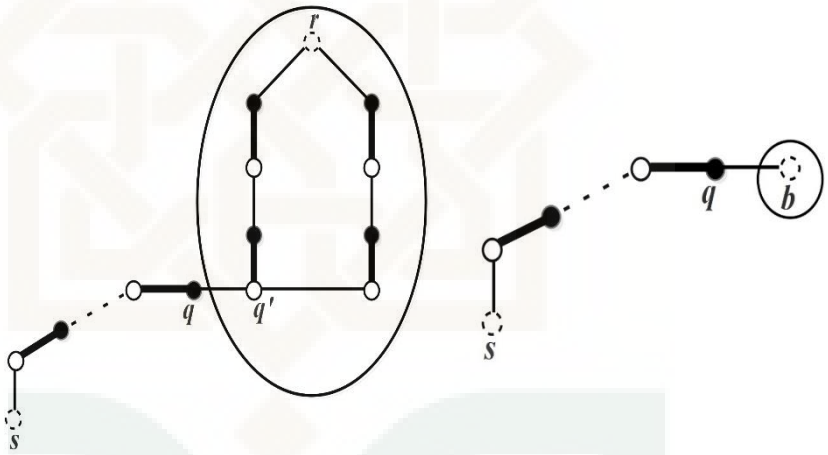
(⟹)Misal *P* sebuah lintasan *augmenting* di (*G, M*) yang dimulai dari simpul *exposed r* dan memiliki simpul akhir katakan *s* (*exposed*), serta asumsikan bahwa *r* dan *s* merupakan simpul *exposed* yang tersisa pada graf (*G, M*). Karna *P* merupakan sebuah lintasan *augmenting* maka semua simpul di *P saturated* kecuali *r* dan *s*,. Andaikan *P* tidak berpotongan dengan *B* maka jelas bahwa *P* juga merupakan lintasan *augmenting* pada (*G', M'*) oleh karna itu kita anggap bahwa *P* berpotongan dengan *B* dimana hal ini memberikan dua kondisi yang berbeda:

a. Akar *r* pada lintasan *augmenting P* termuat di *blossom B*, sehingga *r* merupakan *base* dari *blossom B*. Misalkan *q* simpul pada *P* yang mana menjadi simpul pertama yang tidak termuat di *B* dan *q'* merupakan

simpul pada *P* yang mana merupakan simpul terakhir yang termuat pada *B* (*q'* bisa jadi sama dengan *r*). Maka sisi *e* = *q*'*q* tidak termuat pada *B*. Jelas bahwa apabila penyusutan dilakukan pada *blossom B* yang memuat *r* maka akan menghasilkan sebuah *pseudovertex b* (*exposed*) sedemikian sehingga menghasilkan lintasan *augmenting*

$$P' : b \text{ —— } q \text{ —— } \cdots \text{ —— } s$$

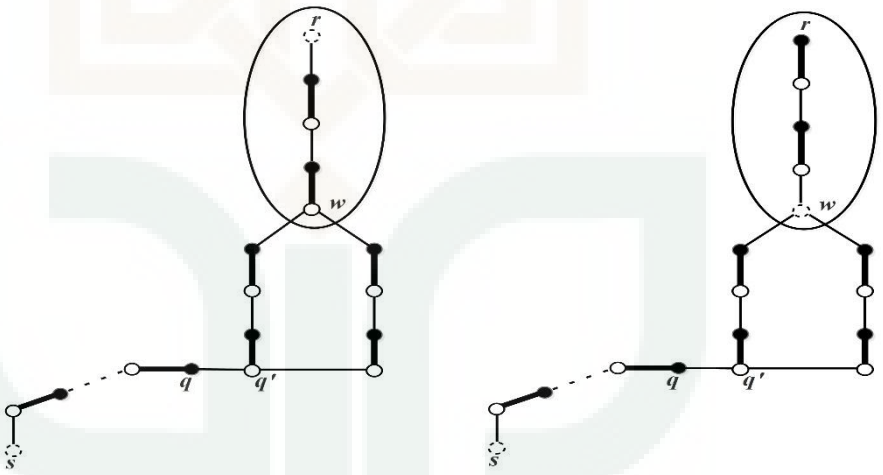
Dimana *P'* merupakan lintasan *augmenting* di (*G', M'*)



Gambar 3.11 Ilustrasi kondisi a

b. Akar *r* pada lintasan *augmenting P* tidak termuat di *blossom B*, sehingga *base* dari *blossom B* adalah sebuah simpul *outer* katakan *w* dengan $w \neq r$. Misal lintasan *alternating* (dengan panjang genap) dari *r* menuju *w* pada *T* dinotasikan sebagai *S* (*S* biasa disebut sebagai *stem*). Pada kondisi (2) ini dapat direduksi kedalam kondisi (1) dengan menggantikan *matching M* dengan *matching* $M_1 = M \oplus S$ (dengan

jumlah kardinalitas yang sama). Akibatnya *w* dan *s* merupakan simpul *exposed* yang tersisa pada (G, M_1). Karna di awal diasumsikan bahwa M memuat lintasan *augmenting* pada G maka M tidak maksimum begitu juga dengan M_1 (karna memiliki jumlah kardinalitas yang sama). Maka berdasarkan teorema *berge* terdapat sebuah lintasan *augmenting* P_1 pada (G, M_1). Berdasarkan kondisi (1) maka terdapat sebuah lintasan *augmenting* P_1*'* di (G', M_1 / B). Jelas bahwa M_1 / B tidak maksimum sehingga M / B juga tidak maksimum (karna memiliki jumlah kardinalitas yang sama) atau dengan kata lain terdapat lintasan *augmenting* dari *r* menuju ke *s* pada (G', M / B).



Gambar 3.12 Ilustrasi kondisi b

(⇐)Misal terdapat sebuah lintasan *augmenting P'* dari *r* menuju ke *s* pada (G', M / B) yang mana tidak memuat *pseudovertex b* maka jelas bahwa *P'* juga merupakan lintasan *augmenting* di (G, M). Oleh karena itu

anggap bahwa *P'* memuat *pseudovertex b* sedemikian sehingga terdapat dua kondisi sebagai berikut:

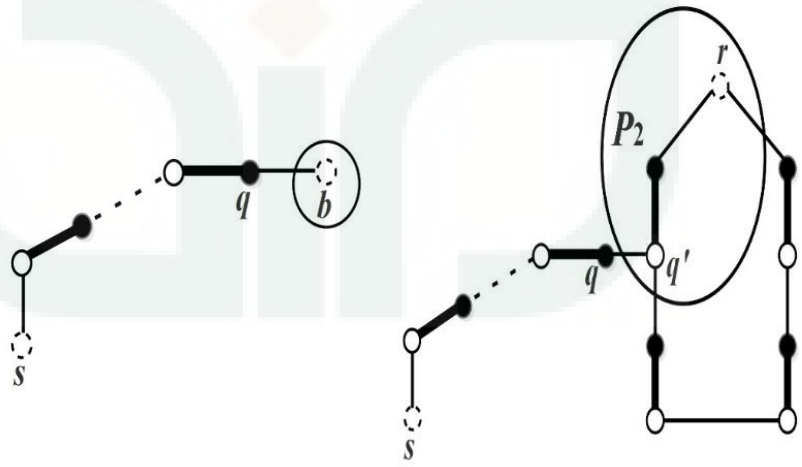
c. Akar *r* pada lintasan *augmenting P'* termuat pada *blossom B*, maka *r* merupakan *base* dari *B* (*r* = *b*) sedemikian sehingga lintasan *augmenting P'* sebagai berikut:

$$P': b \text{ —— } q \text{ —— } \cdots \text{ —— } s$$

dengan *q* simpul pada *P'* yang mana menjadi simpul pertama yang tidak termuat di *B*. Misal lintasan *alternating* dengan panjang genap di *B* dari *base B* menuju *q'* (simpul pada *B* yang *adjacent* dengan simpul *q*) dinotasikan dengan P_2 maka:

$$P : r — P_2 — q' — q — s$$

Merupakan sebuah lintasan *augmenting* di (*G*, *M*).



Gambar 3.13 Ilustrasi kondisi c

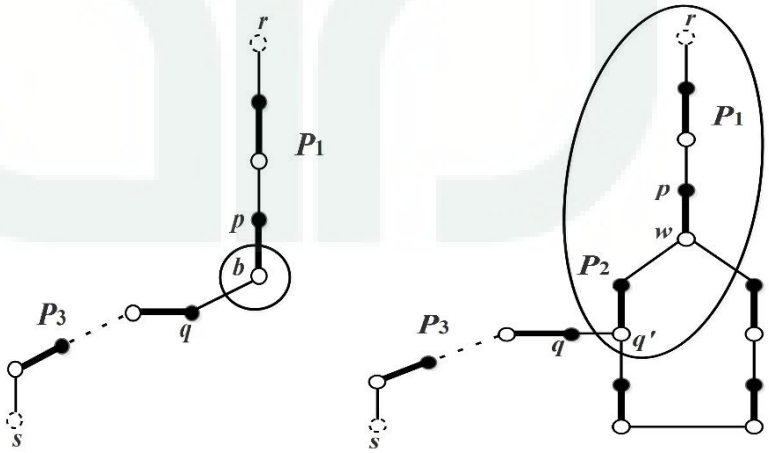
d. Akar *r* pada lintasan *augmenting P'* tidak termuat pada *blossom B*, sehingga lintasan *augmenting P'* adalah sebagai berikut:

$$P': r - P_1 - p - b - q - P_3 - s$$

Dengan P_1 bagian dari lintasan *augmenting P'* dimulai dari *r* menuju $p = mate(b)$ dan P_3 bagian dari lintasan *augmenting P'* dari *q* menuju *s*. Karena *q* merupakan simpul pada *P'* yang mana menjadi simpul pertama yang tidak termuat di *B* maka *q* harus *adjacent* dengan salah satu simpul pada *B* (katakan simpul *q'*). Apabila lintasan *alternating* dengan panjang genap pada *blossom B* berawal dari *w* (*base B*) menuju *q'* dimisalkan sebagai P_2 maka:

$$P : r - P_1 - p - w - P_2 - q' - q - P_3 - s$$

Merupakan lintasan *augmenting P* pada (*G*, *M*). ■

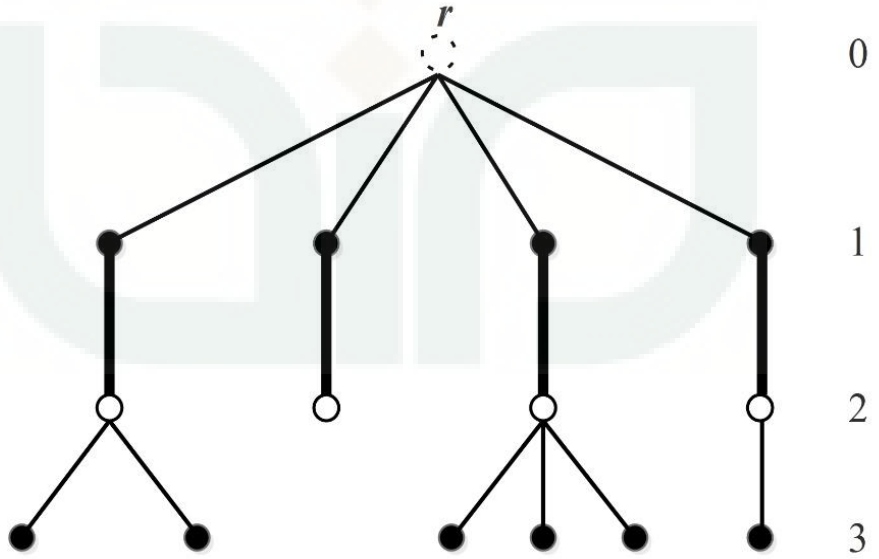


Gambar 3.14 Ilustrasi kondisi d

3.4 Pohon *Alternating*

Sebuah pohon *alternating T* adalah sebuah pohon yang berakar pada sebuah simpul *exposed r* yang mana tiap – tiap sisi pada pohon *T* menghubungkan sebuah simpul *inner* dengan sebuah simpul *outer* sedemikian sehingga setiap simpul *inner* pada *T* dengan pasti *incident* dengan dua sisi pada *T*. Untuk setiap simpul *outer* pada suatu pohon *alternating T* terdapat suatu lintasan *augmenting* di *T* apabila setiap simpul *outer* saling *incident* dengan simpul *inner* (*exposed*). Simpul *inner* merupakan simpul yang menempati layer ganjil (1, 3, 5, ...) pada pohon *T* sedangkan simpul *outer* merupakan simpul yang menempati layer genap (0, 2, 4, ...) pada pohon *T*. **(**Jack Edmonds, 1965:454)

Contoh 3.6:



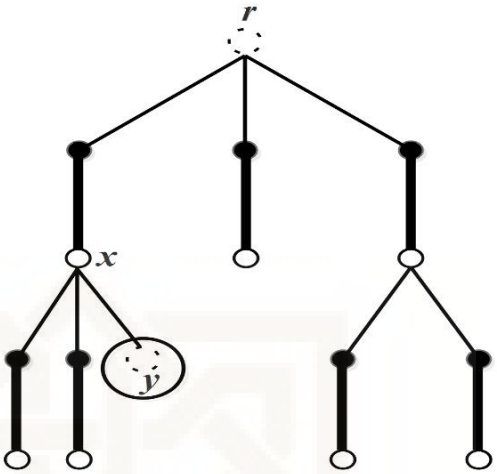
Gambar 3.15 Pohon *Alternating T* dengan akar *r*

Tujuan dibentuknya pohon *alternating T* yakni untuk menemukan sebuah simpul *inner* (*exposed*) katakan *y* sehingga menghasilkan sebuah lintasan *augmenting* dari akar *r* sampai *y*. Langkah – langkah penyusunan pohon *alternating* dalam pencarian lintasan *augmenting* adalah sebagai berikut: (Dieter Jungnickel, 2008: 395)

(Akar) : Pilih sebuah simpul *exposed r* di graf (*G, M*). Jadikan simpul *exposed r* tersebut sebagai akar (layer 0) pada pohon *alternating T*.

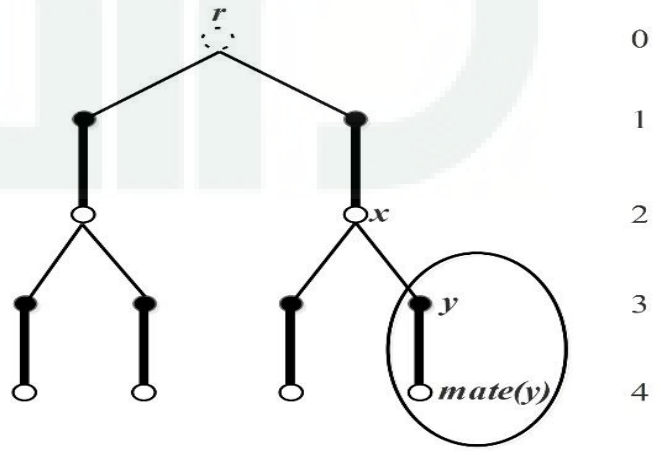
1. Pada layer satu tambahkan semua simpul $a_1, a_2, a_3, \cdots, a_p$ yang mana *adjacent* dengan akar *r*. Ingat bahwa semua simpul yang ditambahkan tersebut hanya simpul yang *saturated*.
2. Tambahkan simpul $b_i = mate(a_i)$ pada layer kedua (genap) di *T*.
3. Untuk layer selanjutnya tambahkan semua simpul $c_1, c_2, c_3, \cdots, c_p$ dimana c_i *adjacent* dengan minimal satu b_i dan sisi yang menguhubungkan c_i dengan b_i tidak termuat pada *M*. Lanjutkan langkah 1, langkah 2 dan langkah 3 hingga didapati salah satu dari dua kemungkinan sebagai berikut: (a) Sebuah lintasan *augmenting*, (b) Sebuah *cycle* ganjil.
4. Misalkan *x* adalah sebuah simpul pada layer 2*i*, dan $y \neq mate(x)$ adalah sebuah simpul yang *adjacent* dengan *x*. Maka terdapat empat kemungkinan:

   Kasus 1: *y* merupakan simpul *exposed* (dan belum termuat di *T*). Maka telah ditemukan sebuah lintasan *augmenting*. Sehingga proses penyusunan pohon *alternating T* dihentikan.



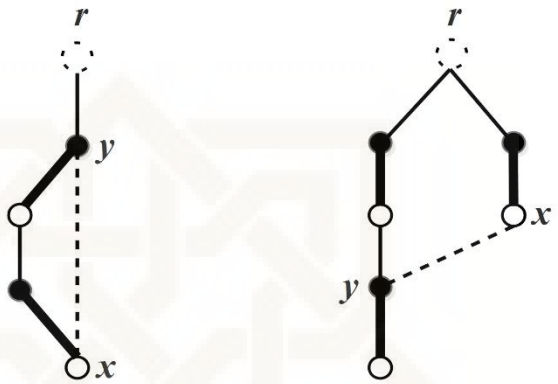
Gambar 3.16 Ilustrasi Kasus 1

Kasus 2: *y* bukan merupakan simpul *exposed*, dan *y* maupun *mate*(*y*) keduanya belum termuat di *T*. Maka tambahkan *y* pada layer 2*i* + 1 dan *mate*(*y*) pada layer 2*i* + 2.



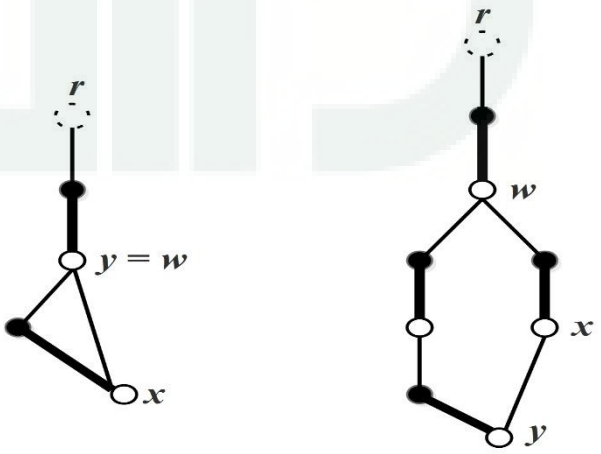
Gambar 3.17 Ilustrasi Kasus 2

Kasus 3: *y* telah termuat di *T* sebagai sebuah simpul *inner*. Maka telah ditemukan *cycle* dengan panjang genap sehingga abaikan jika ditemui kondisi seperti ini.



Gambar 3.18 Ilustrasi Kasus 3

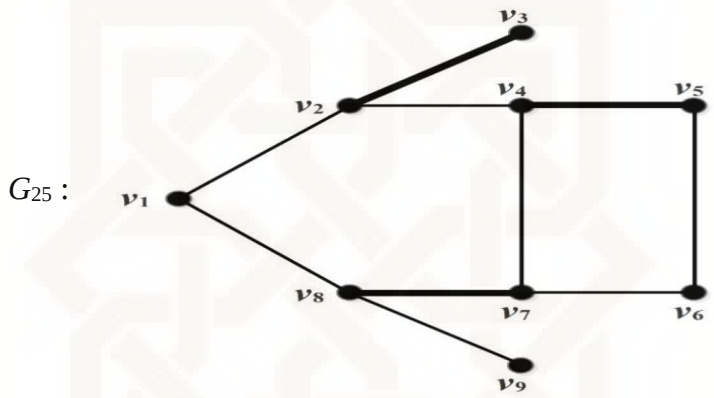
Kasus 4: *y* telah termuat di *T* sebagai sebuah simpul *outer*. Maka telah ditemukan sebuah *Blossom*. Hentikan proses penyusunan pohon *alternating T* dan lakukan proses penyusutan *blossom.*



Gambar 3.19 Ilustrasi Kasus 4

5. Hentikan proses penyusunan pohon *alternating T* apabila didapati: (a) Sebuah lintasan *augmenting*, (b) Sebuah *cycle* ganjil.

Contoh 3.7:



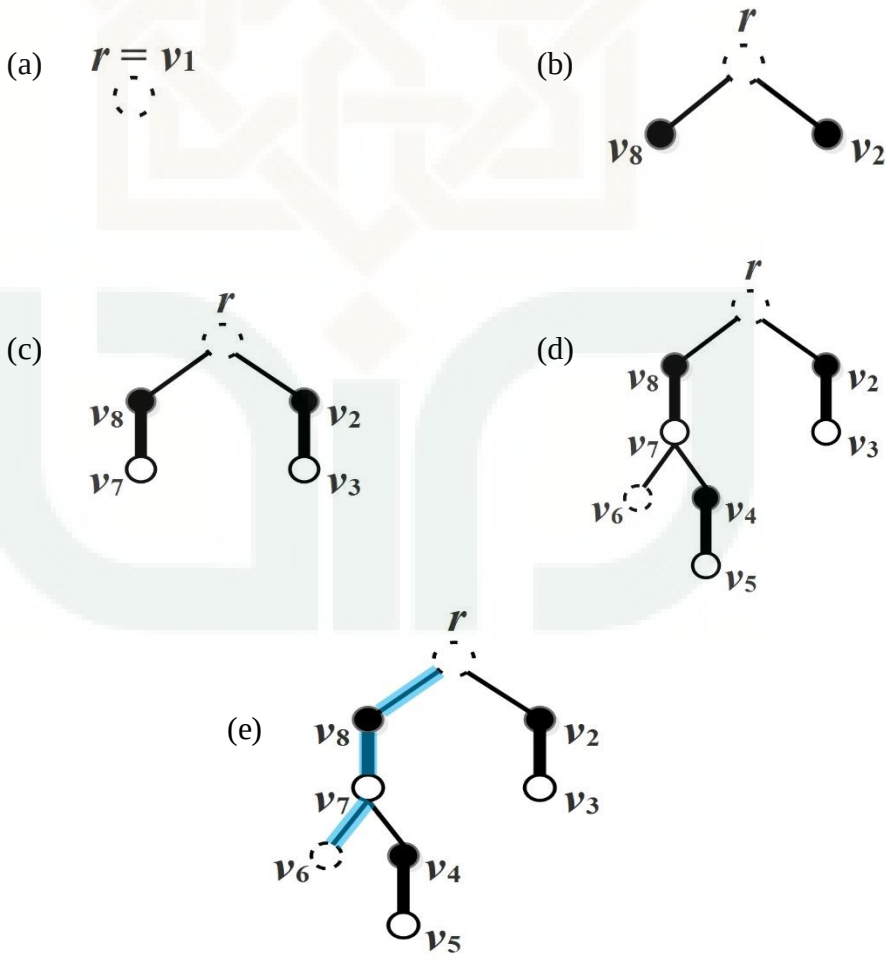
Gambar 3.20 Graf (G_{25}, M) dengan *matching* $M = \{v_2v_3, v_4v_5, v_8v_7\}$

Dari Gambar 3.20 akan disusun sebuah pohon *alternating T* sebagai berikut:

1. Dipilih sebuah simpul *exposed* $r = v_1$ di (G_{25}, M) sebagai akar pada pohon *alternating T*. Pada layer satu tambahkan simpul *saturated* v_2 dan v_8 (*adjacent* dengan simpul $r = v_1$).
2. Tambahkan simpul $v_3 = mate(v_2)$ dan $v_7 = mate(v_8)$ guna menempati layer 2 pada pohon *alternating T*.
3. Untuk mengisi layer selanjutnya (layer 3) dipilih simpul v_6 dan v_4 yang mana keduanya *adjacent* dengan simpul v_7 oleh sisi yang tidak temuat

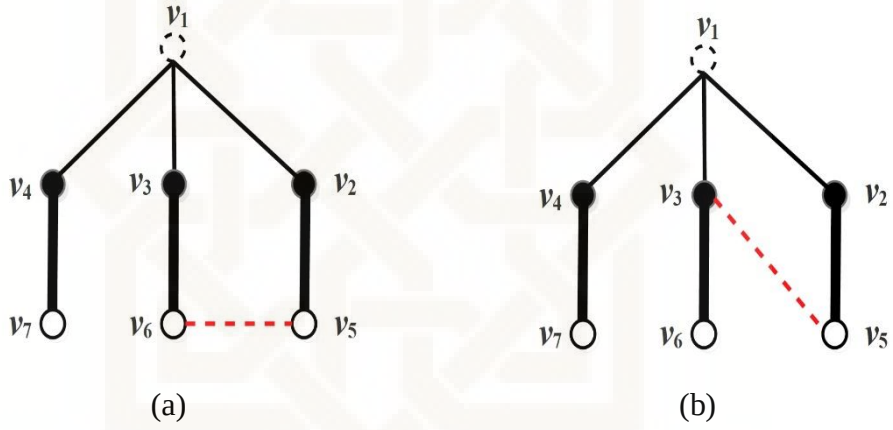
dalam *M*. Tambahkan simpul $v_5 = mate(v_4)$ pada layer selanjutnya yakni layer 4.

4. Pada layer 3 atau layer ganjil didapati sebuah simpul *exposed* v_6, dengan kata lain telah ditemukan sebuah lintasan *augmenting* pada pohon *alternating T* dengan $r = v_1$.
5. Karna telah ditemukan sebuah lintasan *augmenting* maka penyusunan pohon *alternating T* dihentikan.



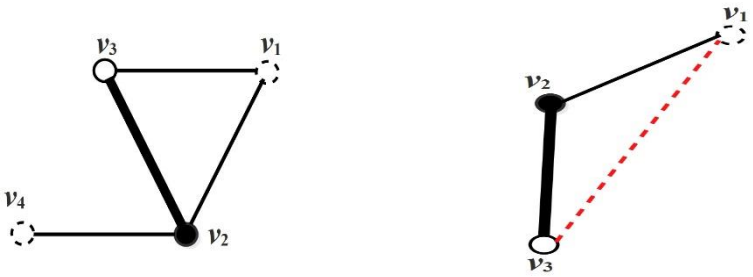
Gambar 3.21 Pohon *alternating T* atas graf (G_{25}, *M*)

Pada kasus graf *bipartite* tidak memungkinkan untuk ditemukan sebuah *blossom* pada saat penyusunan pohon *alternating T* hal ini dikarenakan pada graf *bipartite* tidak terdapat dua buah simpul yang saling *adjacent* pada partisi yang sama sehingga tidak memungkinkan dua buah simpul pada layer yang sama untuk saling terhubung (Uri Zwick, 2009: 4). Berikut diberikan ilustrasi dari graf *bipartite* dan graf *non-bipartite*:



Gambar 3.22 (a) Ilustrasi graf *non-bipartite*
(b) Ilustrasi graf *bipartite*

Sebuah *blossom* pada suatu graf *non-bipartite* dapat memunculkan kemungkinan tidak ditemukannya sebuah lintasan *augmenting* pada saat penyusunan pohon *alternating* (Uri Zwick, 2009: 7).



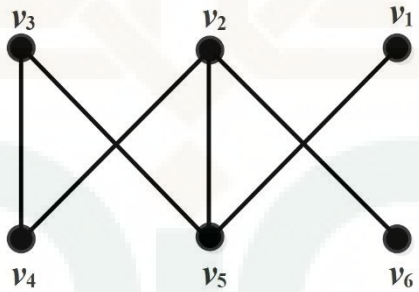
Gambar 3.23 Ilustrasi lintasan *augmenting*

Meski algoritma kardinalitas *matching Edmonds* dapat digunakan untuk melakukan pencarian *matching* maksimum pada suatu graf *bipartite* akan tetapi ide utama dari algoritma kardinalitas *matching Edmonds* yakni penyusutan dan perentangan *blossom* tidak muncul pada saat melakukan pencarian *matching* maksimum pada suatu graf *bipartie*. Hal ini dikarenakan pada graf *bipartite* tidak memuat *blossom* sehingga tidak memungkinkan untuk melakukan proses penyusutan maupun perentangan *blossom*.

Contoh 3.8:

Diberikan sebuah graf *bipartie* G_{24} dengan himpunan partisi $V_1 = \{v_1, v_2, v_3\}$ dan $V_2 = \{v_4, v_5, v_6\}$.

G_{26} :



Gambar 3.24 Graf bipartit G_{26}

Dari Gambar 3.24 akan dilakukan pencarian *matching* maksimum dengan menggunakan algoritma kardinalitas *matching Edmonds* sebagai berikut:

Langkah 1 (*inisialisasi*) :

Iterasi 1 :

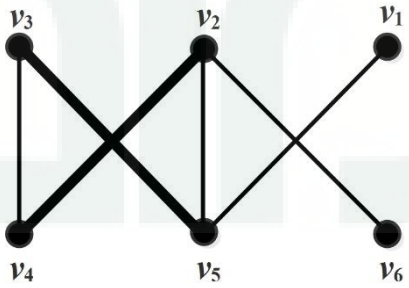
1. $M = Ø$
2. Dipilih sisi v_2v_4 untuk ditambahkan kedalam *matching M*
3. $M = M \cup \{v_2v_4\}$
4. Kembali ke *M*

Iterasi 2 :

1. $M = \{v_2v_4\}$
2. Dipilih sisi v_3v_5 untuk ditambahkan kedalam *matching M*
3. $M = \{v_2v_4) \cup \{v_3v_5\}$
4. Proses dihentikan.

Berikut merupakan hasil inisialisasi *matching* terhadap graf G_{26}:

(G_{26}, M) :

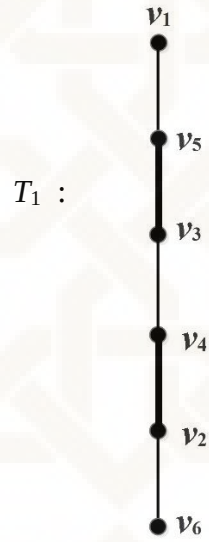


Gambar 3.25 Graf (G_{26}, M)

Karena masih didapati lebih dari satu simpul *exposed* di (G_{24}, M) yakni simpul v_1 dan v_6 sehingga lanjutkan ke langkah 2.3

Langkah 2.3 :

Misal dipilih simpul v_1 sebagai akar, dengan menggunakan BFS maka diperoleh pohon *alternating* T_1 atas graf (G_{26}, M) sebagai berikut:



Gambar 3.26 Pohon T_1 atas graf (G_{26}, M)

Telah ditemukan lintasan *augmenting P* dengan:

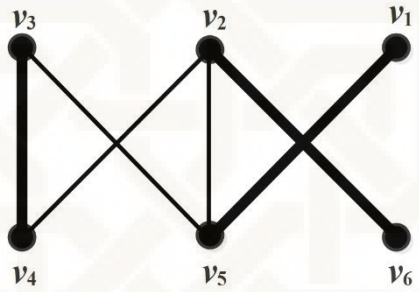
$$P : v_1 — v_5 — v_3 — v_4 — v_2 — v_6$$

Terlihat bahwa lintasan *augmenting P* yang ditemukan pada saat proses penyusunan pohon *alternating* T_1 atas graf *bipartite* (G_{26}, M) sama sekali tidak memuat *pseudovertex b* dengan kata lain graf *bipartite* (G_{26}, M) tidak memuat sebuah *blossom B.* Lanjutkan ke langkah 3.2.

Langkah 3.2:

Dengan melakukan *augmenting* terhadap inisial *matching M* menggunakan lintasan *augmenting P* sehingga diperoleh *matching* baru di (G_{26}, M) yakni $M' = M \oplus P$.

(G_{26}, M') :



Gambar 3.27 Graf (G_{26}, M')

Karena semua simpul pada graf (G_{26}, M') *incident* dengan sebuah sisi *matching* maka lanjutkan ke langkah 5.

Langkah 5:

Matching M' di graf *bipartite* (G_{26}, M') sudah maksimum oleh karena itu hentikan proses pencarian.

Dari langkah – langkah di atas, diperoleh *matching* maksimum *M'* pada graf *bipartite* G_{26} dengan *matching* $M' = \{v_1v_5, v_3v_4, v_2v_6\}$ dengan kardinalitas dari *matching M'* yakni $|M'| = 3$.

Contoh Kasus (*The Battle of Britain*)

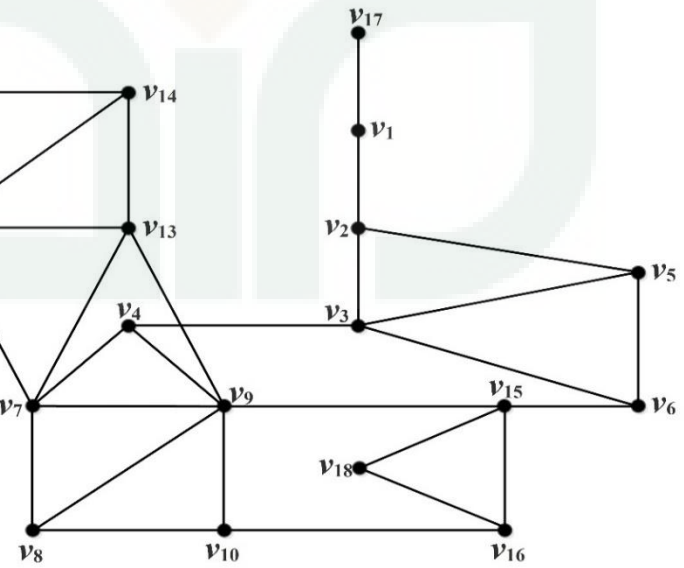
Pada pertempuran Britain tahun 1940, pangkalan udara "*Royal Air Force*" menyediakan beberapa pesawat tempur guna untuk menggagalkan serangan udara yang dilakukan Jerman. Pesawat tempur yang disediakan oleh *Royal Air Force* hanya dapat diterbangkan oleh dua orang sebagai pilot dan co-pilot. Guna mendapatkan hasil terbaik dalam misi menggagalkan serangan Jerman, pangkalan udara *Royal Air Force* mendatangkan beberapa pilot terbaik dari seluruh daerah di Britain. Akan tetapi terdapat sebuah kendala yang dialami oleh pangkalan udara *Royal Air Force* dimana dari seluruh pilot yang didatangkan dari seluruh daerah di Britain, beberapa pilot tidak dapat terbang bersama dalam satu pesawat tempur dikarenakan adanya perbedaan baik dari segi bahasa, teknik terbang yang dikuasai maupun dari segi pengalaman. Misal didapatkan 18 orang pilot terbaik dari seluruh daerah di Britain yang siap untuk menerbangkan pesawat tempur. Berdasarkan kendala diatas pangkalan udara *Royal Air Force* menginginkan untuk memasangkan ke 18 pilot dimana setiap pasangan terdiri dari dua orang pilot sedemikian sehingga jumlah dari seluruh pasangan yang terbentuk merupakan jumlah terbanyak. Guna sebagai langkah awal menyelesaikan masalah diatas pangkalan udara *Royal Air Force* melakukan tes terhadap ke 18 pilot meliputi tes bahasa, teknik terbang, dan *psikologi* untuk mengetahui kecocokan antara mana saja dari ke 18 pilot tersebut yang dapat terbang secara bersama dalam satu pesawat tempur sebagai pilot dan co-pilot. (M. Gondran dan M. Minoux, 1984:279)

Tabel 3.1 Tabel kecocokan antara masing – masing pilot pesawat tempur

Pilot ke *i*, dengan *i* = (1, 2, ⋯ , 18)	**Hasil kecocokan antara masing – masing pilot**
Pilot ke 1	Pilot ke: 2 dan 17
Pilot ke 2	Pilot ke: 1, 3, dan 5
Pilot ke 3	Pilot ke: 2, 4, 5, dan 6
Pilot ke 4	Pilot ke: 3, 7, dan 9
Pilot ke 5	Pilot ke: 2, 3, dan 6
Pilot ke 6	Pilot ke: 3, 5, dan 15
Pilot ke 7	Pilot ke: 4, 8, 9, 11, dan 13
Pilot ke 8	Pilot ke: 7, 9, dan 10
Pilot ke 9	Pilot ke: 4, 7, 8, 10, 13, dan 15
Pilot ke 10	Pilot ke: 8, 9, dan 16
Pilot ke 11	Pilot ke: 7, 12, 13, dan 14
Pilot ke 12	Pilot ke: 11 dan 14
Pilot ke 13	Pilot ke: 7, 9, 11, dan 14
Pilot ke 14	Pilot ke: 11, 12, dan 13
Pilot ke 15	Pilot ke: 6, 9, 16, dan 18
Pilot ke 16	Pilot ke: 10, 15, dan 18
Pilot ke 17	Pilot ke 1
Pilot ke 18	Pilot ke: 15 dan 16

Permasalahan diatas merupakan permasalahan *matching* maksimum. Permasalahan *matching* maksimum merupakan permasalahan pemasangan sedemikian sehingga jumlah pasangan yang terbentuk merupakan jumlah terbanyak atau maksimum (C. Berge, 1973: 122). Permasalahan diatas dapat diselesaikan dengan cara mentransformasi kedalam bentuk graf dimana ke 18 pilot pesawat tempur di representasikan sebagai simpul v_i (i = 1, 2, ⋯ , 18) sedangkan hasil kecocokan antara masing – masing pilot pesawat tempur (pada tabel 1.2) di representasikan sebagai sebuah sisi yang menghubungkan setiap simpul pada graf. Dua buah simpul yang terhubung oleh sebuah sisi pada graf merepresentasikan adanya kecocokan antara dua orang pilot yang dapat berpasangan sebagai pilot dan co-pilot untuk menerbangkan sebuah pesawat tempur. Berikut merupakan graf hasil dari transformasi permasalahan diatas yang diambil dari Dieter Jungnickel (2008: 401):

G_{27} :



Gambar 3.28 Graf G_{27}

Pada graf G_{27} yang terbentuk dari hasil transformasi permasalahan *The Batle of Britain* dapat dengan mudah kita temukan sebuah *cycle* ganjil yang termuat didalamnya. Sebuah *cycle C* dengan $C = (v_2, v_3, v_5, v_2)$ merupakan sebuah *cycle* ganjil yang termuat pada graf G_{24}. Berdasarkan Teorema 2.2.1 maka graf G_{27} bukan merupakan graf bipartit (graf *non-bipartite*).

Dengan mengikuti langkah – langkah pada algoritma kardinalitas *matching edmonds* maka diperoleh *matching* maksimum pada graf G_{27} sebagai berikut:

Langkah 1 (*inisialisasi*) :

Iterasi 1 :

1. $M = \emptyset$
2. Dipilih sisi v_1v_2 untuk ditambahkan kedalam *matching M*
3. $M = M \cup \{v_1v_2\}$
4. Kembali ke *M*

Iterasi 2 :

1. $M = \{v_1v_2\}$
2. Dipilih sisi v_3v_4 untuk ditambahkan kedalam *matching M*
3. $M = \{v_1v_2\} \cup \{v_3v_4\}$
4. Kembali ke *M*

Iterasi 3 :

1. $M = \{v_1v_2, v_3v_4\}$
2. Dipilih sisi v_5v_6 untuk ditambahkan kedalam *matching M*
3. $M = \{v_1v_2, v_3v_4\} \cup \{v_5v_6\}$
4. Kembali ke *M*

Iterasi 4 :

1. $M = \{v_1v_2, v_3v_4, v_5v_6\}$
2. Dipilih sisi v_7v_8 untuk ditambahkan kedalam *matching M*
3. $M = \{v_1v_2, v_3v_4, v_5v_6\} \cup \{v_7v_8\}$
4. Kembali ke *M*

Iterasi 5 :

1. $M = \{v_1v_2, v_3v_4, v_5v_6, v_7v_8\}$
2. Dipilih sisi v_9v_{10} untuk ditambahkan kedalam *matching M*
3. $M = \{v_1v_2, v_3v_4, v_5v_6, v_7v_8\} \cup \{v_9v_{10}\}$
4. Kembali ke *M*

Iterasi 6 :

1. $M = \{v_1v_2, v_3v_4, v_5v_6, v_7v_8, v_9v_{10}\}$
2. Dipilih sisi $v_{11}v_{12}$ untuk ditambahkan kedalam *matching M*
3. $M = \{v_1v_2, v_3v_4, v_5v_6, v_7v_8, v_9v_{10}\} \cup \{v_{11}v_{12}\}$
4. Kembali ke *M*

Iterasi 7 :

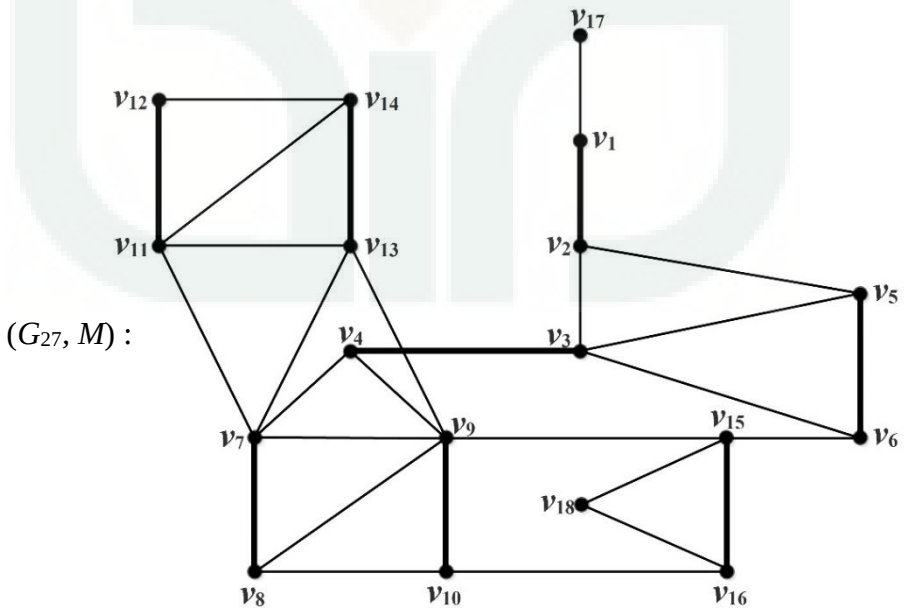
1. $M = \{v_1v_2, v_3v_4, v_5v_6, v_7v_8, v_9v_{10}, v_{11}v_{12}\}$
2. Dipilih sisi $v_{13}v_{14}$ untuk ditambahkan kedalam *matching M*
3. $M = \{v_1v_2, v_3v_4, v_5v_6, v_7v_8, v_9v_{10}, v_{11}v_{12}\} \cup \{v_{13}v_{14}\}$
4. Kembali ke *M*

Iterasi 8 :

1. $M = \{v_1v_2, v_3v_4, v_5v_6, v_7v_8, v_9v_{10}, v_{11}v_{12}, v_{13}v_{14}\}$
2. Dipilih sisi $v_{15}v_{16}$ untuk ditambahkan kedalam *matching M*
3. $M = \{v_1v_2, v_3v_4, v_5v_6, v_7v_8, v_9v_{10}, v_{11}v_{12}, v_{13}v_{14}\} \cup \{v_{15}v_{16}\}$
4. Proses dihentikan.

Berikut merupakan hasil dari inisialisasi *matching* terhadap graf G_{27}:

(G_{27}, M) :



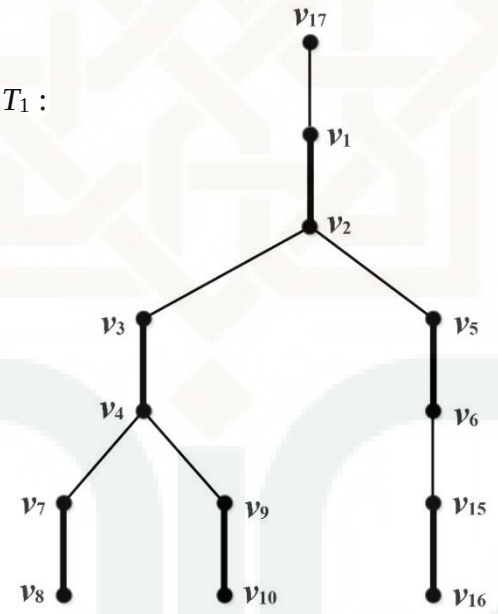
Gambar 3.29 Graf (G_{27}, M)

Dari hasil inisialisasi *matching* terhadap graf G_{27}, masih didapati lebih dari satu simpul *exposed* di (G_{27}, M) yakni simpul v_{17} dan v_{18} sehingga lanjutkan ke langkah 2.3.

Langkah 2.3 :

Misal dipilih simpul v_{17} sebagai akar, dengan menggunakan BFS maka diperoleh pohon *alternating* T_1 atas graf (G_{27}, M) sebagai berikut:

Pohon T_1 :



Gambar 3.30 Pohon *alternating* T_1 atas (G_{27}, M)

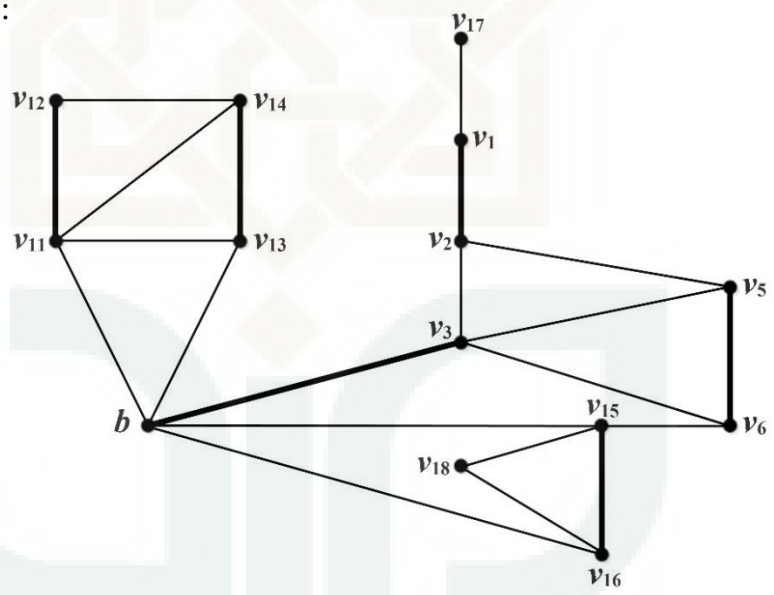
Sisi v_6v_3 dan v_8v_9 di (G_{27}, M) mengakibatkan munculnya *cycle* genap pada T_1, sehingga kedua sisi tersebut diabaikan (pohon *alternating* kasus 3). Sisi v_8v_{10} di (G_{27}, M) mengakibatkan munculnya *blossom* di T_1 (pohon *alternating* kasus 4), misal namakan sebagai *blossom* $B = \{v_4, v_7, v_8, v_9, v_{10}\}$ dengan *base*

blossom B yakni simpul v_4. Berdasarkan langkah 2.3.2 maka hentikan proses penyusunan pohon T_1 dan lanjutkan ke langkah 4.

Langkah 4:

Abaikan semua simpul $B = \{v_4, v_7, v_8, v_9, v_{10}\}$ dan semua sisi yang menghubungkan antar masing – masing simpul B di (G_{27}, M) sedemikian sehingga diperoleh $G_{27}' = G_{27} / B$ dengan *matching* $M_1 = M / B$ sebagai berikut:

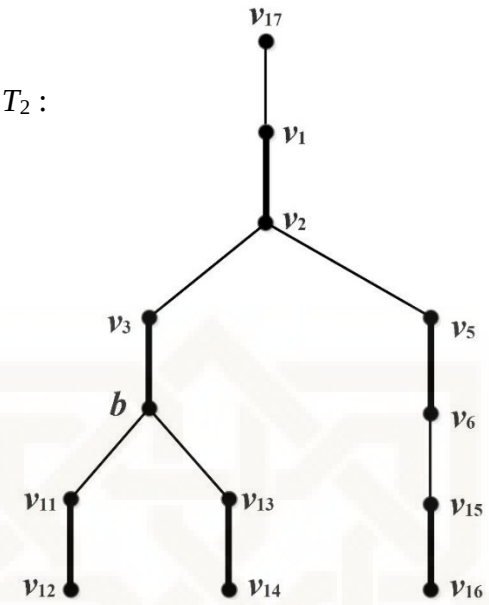
(G_{27}', M_1) :



Gambar 3.31 Graf (G_{27}', M_1)

Lanjutkan kembali penyusunan pohon *alternating* atas graf (G_{27}', M_1) dengan akar simpul v_{17}. Dengan menggunakan BFS diperoleh pohon *alternating* T_2 atas graf (G_{27}', M_1) sebagai berikut:

Pohon T_2 :



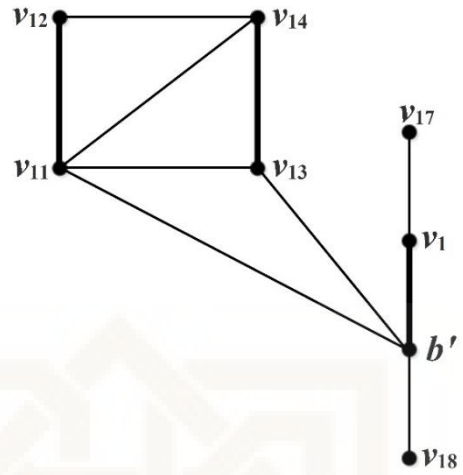
Gambar 3.32 Pohon *alternating* T_2 atas graf (G_{27}', M_1)

Sisi bv_{15} di (G_{27}', M_1) mengakibatkan munculnya *cycle* genap pada T_2, sehingga sisi tersebut diabaikan (pohon *alternating* kasus 3). Sisi bv_{16} di (G_{27}', M_1) mengakibatkan munculnya *blossom* di T_2 (pohon *alternating* kasus 4), misal namakan sebagai *blossom B'* = {b, v_2, v_3, v_5, v_6, v_{15}, v_{16}} dengan *base blossom B'* yakni simpul v_2. Berdasarkan langkah 2.3.2 maka hentikan proses penyusunan pohon T_2 dan lanjutkan ke langkah 4.

Langkah 4:

Abaikan semua simpul *B'* = {b, v_2, v_3, v_5, v_6, v_{15}, v_{16}} dan semua sisi yang menghubungkan antar masing – masing simpul *B'* di (G_{27}', M_1) sedemikian sehingga diperoleh graf $G_{27}'' = G_{27}' / B'$ dengan *matching* $M_2 = M' / B'$ sebagai berikut:

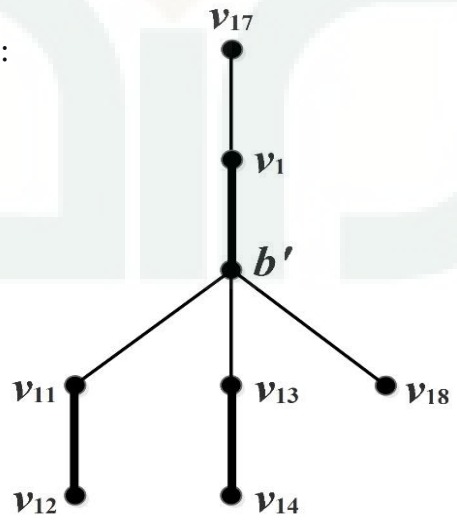
(G_{27}'', M_2) :



Gambar 3.33 Graf (G_{27}'', M_2)

Lanjutkan kembali penyusunan pohon *alternating* atas graf (G_{27}'', M_2) dengan akar simpul v_{17}. Dengan menggunakan BFS diperoleh pohon *alternating* T_3 atas graf (G_{27}'', M_2) sebagai berikut:

Pohon T_3 :



Gambar 3.34 Pohon *alternating* T_3 atas graf (G_{27}'', M_2)

Dari pohon *alternating* T_3 yang telah terbentuk pada Gambar 3.28 didapati sebuah simpul *inner* (*exposed*) yakni simpul v_{18}. Berdasarkan langkah 3.2.1 bahwa telah ditemukan lintasan *augmenting*, misal namakan *P''* dengan:

$$P'' : v_{18} — b' — v_1 — v_{17}$$

Karena pada lintasan *augmenting P''* memuat *pseudovertex b'* maka lanjutkan ke langkah 3.1

Langkah 3.1:

Rentangkan *b'* menjadi *blossom B'* = $\{b, v_2, v_3, v_5, v_6, v_{15}, v_{16}\}$. Karena simpul v_{18} *adjacent* dengan *b'* maka paling sedikit terdapat satu simpul *q''* di (G_{27}', M_1) yang *adjacent* dengan v_{18} dan termuat dalam *B'*. Diperoleh simpul v_{15} dan simpul v_{16}. Misal dipilih simpul *q''* = v_{15}, dengan menggunakan lintasan *P''*maka perunutan dilanjutkan dimulai dari simpul v_{18} melewati simpul v_{15} dan bagian lintasan *alternating* dengan panjang genap di *B'*. Diperoleh lintasan *augmenting* baru, namakan *P''* dengan:

$$P' : v_{18} — v_{15} — v_{16} — b — v_3 — v_2 — v_1 — v_{17}$$

Karena pada lintasan *augmenting P'* masih memuat sebuah *pseudovertex* yakni *pseudovertex b* maka lanjutkan ke langkah 3.1

Langkah 3.1:

Rentangkan *b* menjadi *blossom B* = $\{v_4, v_7, v_8, v_9, v_{10}\}$. Karena simpul v_{16} *adjacent* dengan *b* maka paling sedikit terdapat satu simpul *q'* di (G_{27}, *M*)

yang *adjacent* dengan v_{16} dan termuat dalam *B* yakni simpul v_{10}. Dengan menggunakan lintasan *P'* perunutan dilanjutkan kembali dimulai dari simpul v_{18} melewati simpul v_{15}, v_{16}, v_{10} dan bagian lintasan *alternating* dengan panjang genap di *B'*. Diperoleh lintasan *augmenting* baru, namakan *P'* dengan:

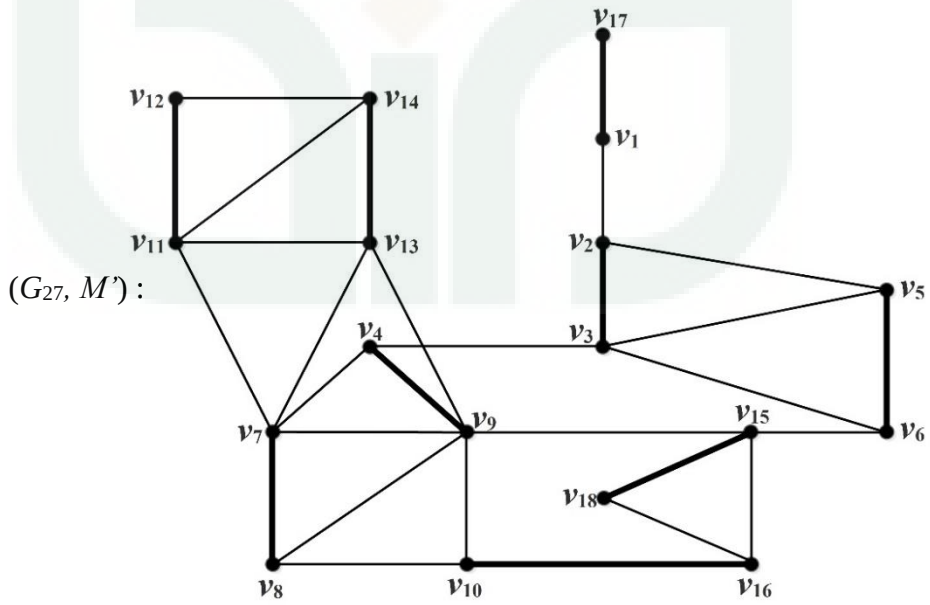
$$P : v_{18} — v_{15} — v_{16} — v_{10} — v_9 — v_4 — v_3 — v_2 — v_1 — v_{17}$$

Karena pada lintasan *augmenting P* tidak memuat satupun *pseudovertex* maka lanjutkan ke langkah 3.2.

Langkah 3.2:

Dilakukan *augmenting* terhadap inisial *matching M* menggunakan lintasan *augmenting P* sehingga diperoleh *matching* baru di (G_{27}, *M*) yakni $M' = M \oplus P$ sebagai berikut:

(G_{27}, *M'*) :



Gambar 3.35 Graf (G_{27}, *M'*)

Karena semua simpul pada graf (G_{27}, M') *incident* dengan sebuah sisi *matching* maka lanjutkan ke langkah 5.

Langkah 5:

*M*atching M' di graf G_{27} sudah maksimum oleh karena itu hentikan proses pencarian.

Dari langkah – langkah di atas, diperoleh *matching* maksimum M' di G_{27} dengan kardinalitas *matching* M' yakni $|M'| = 9$. Dengan kata lain proses pemasangan ke 18 pilot pesawat tempur sudah selesai dilakukan dan menghasilkan pasangan pilot dengan jumlah maksimum sebagai berikut:

1. Pilot nomer 1 berpasangan dengan pilot nomer 17
2. Pilot nomer 2 berpasangan dengan pilot nomer 3
3. Pilot nomer 4 berpasangan dengan pilot nomer 9
4. Pilot nomer 5 berpasangan dengan pilot nomer 6
5. Pilot nomer 7 berpasangan dengan pilot nomer 8
6. Pilot nomer 10 berpasangan dengan pilot nomer 16
7. Pilot nomer 11 berpasangan dengan pilot nomer 12
8. Pilot nomer 13 berpasangan dengan pilot nomer 14
9. Pilot nomer 15 berpasangan dengan pilot nomer 18

Serta jumlah maksimum pesawat tempur yang dapat diterbangkan yakni berjumlah 9 pesawat tempur.