10. Path dan Konektivitas

Oleh : Ade Nurhopipah

Pokok Bahasan :

1. Graph dan Digraph Terhubung
2. Teorema Menger
3. Analog Teorema Manger
4. Studi Kasus

Sumber :

Aldous, Joan M. ,Wilson, Robin J. 2004. *Graph and Applications*. Springer: UK.

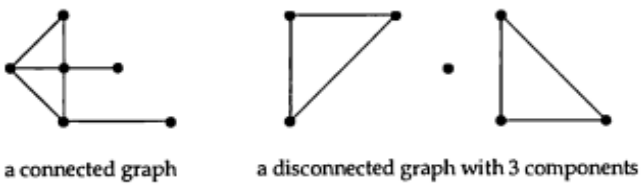
Pada Bab ini, kita akan mempelajari keterhubungan sebuah graph atau digraph secara lebih mendalam. Secara khusus, kita akan mendiskusikan pertanyaan "berapa banyak edge atau vertex yang harus kita hapus dari sebuah graph terhubung sehingga menjadikan graph tersebut tidak terhubung?" Pertanyaan yang berkaitan dengan konektivitas seperti ini, sangat penting untuk dijawab ketika kita mendesain sebuah jaringan telekomunikasi, jaringan jalan raya atau jaringan lainnya. Sebagai contoh dalam jaringan komunikasi, sangat penting untuk memastikan bahwa jaringan tersebut harus tetap dapat dioperasikan, walaupun terdapat beberapa saluran yang rusak, atau diblokir.

Graph dan Digraph Terhubung

Pada bab sebelumnya, kita telah mengenal ide tentang graph terhubung yaitu graph yang menjadi satu kesatuan. Kita mengetahui bahwa pada graph terhubung terdapat sedikitnya sebuah path antara setiap pasang vertex pada graph tersebut. Ingat kembali definisi berikut.

Definisi 10.1
Sebuah graph disebut graph terhubung jika terdapat path antara setiap pasangan vertex.
Setiap graph yang tidak terhubung dapat dibagi menjadi sejumlah graph terhubung yang disebut komponen.

Sebagai contoh perhatikan Gambar 10.1

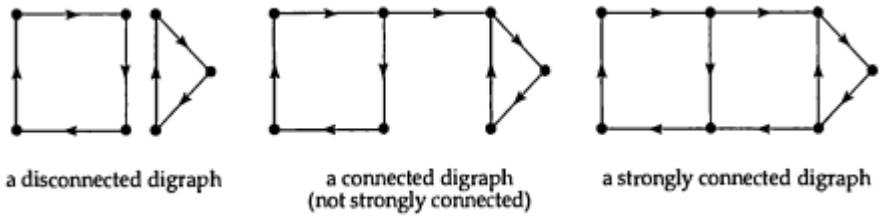


Gambar 10.1 Contoh graph terhubung dan graph tidak terhubung

Pada kasus digraph, terdapat dua buah definisi keterhubungan sebagai berikut.

Definisi 10.2
Sebuah digraph disebut digraph terhubung jika memiliki underlying graph yang terhubung
Sebuah digraph terhubung kuat jika terdapat path di antara setiap pasang vertex pada digraph tersebut.

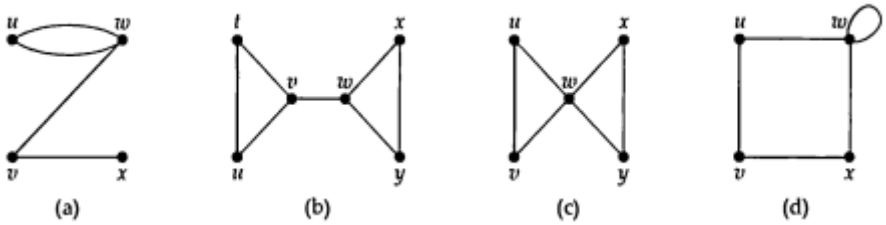
Sebagai contoh perhatikan Gambar 10.2.



Gambar 10.2 Contoh digraph terhubung dan digraph terhubung kuat

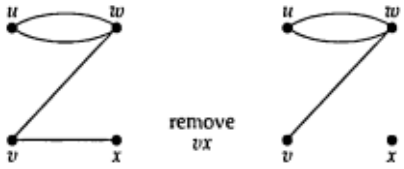
Konektivitas Edge

Untuk banyak aplikasi, kita perlu mengetahui lebih dalam tentang sifat keterhubungan graph dari sekedar apakah graph tersebut terhubung atau tidak. Sebagai contoh, dalam jaringan telekomunikasi biasanya ada beberapa path antara pasangan vertex, dan sangat penting untuk diketahui berapa banyak penghubung (edge) yang dapat rusak tanpa menggangu sebuah panggilan yang dibuat antara dua pelanggan. Untuk menjawab ini, kita akan mempelajari keterhubungan graph secara lebih detail.



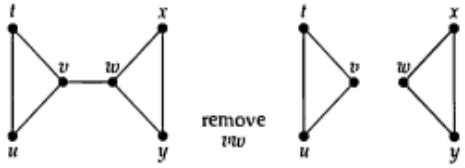
Gambar 10.3 Contoh graph untuk ilustrasi konektivitas edge

Terdapat beberapa graph terhubung pada Gambar 10.3. Graph (a) dapat dibagi menjadi dua komponen dengan menghapus sebuah edge *vw* atau *vx*. Kita menyebut penghapusan kedua edge ini menyebabkan graph menjadi tidak terhubung seperti ditunjukan sebagai berikut.



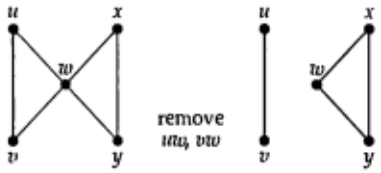
(a)

Graph (b) juga dapat menjadi tidak terhubung dengan menghapus sebuah edge, yaitu *vw*.



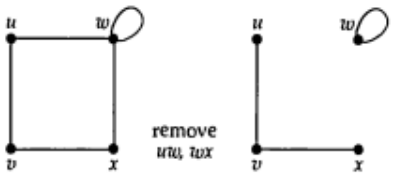
(b)

Graph (c) akan tetap terhubung dengan menghapus sebuah edge, namun ia dapat menjadi tidak terhubung dengan menghapus dua edge, misalnya, *uw* dan *vw*.



(c)

Begitu juga graph (d) dapat menjadi tidak terhubung dengan menghapus dua buah edge, sebagai contoh adalah *uw* dan *wx*.



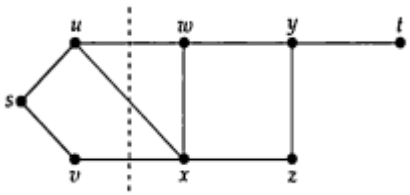
(d)

Dengan contoh tersebut, kita dapat mendefinisikan konektivitas edge pada sebuah graph sebagai berikut.

Definisi 10.3
Konektivitas edge $\lambda(G)$ pada graph terhubung G adalah jumlah edge terkecil yang jika dihapus membuat G tidak terhubung.

Sebagai contoh pada Gambar 10.3, graph (a) dan (b) memiliki konektivitas edge sama dengan 1, sedangkan graph (c) dan (d) memiliki konektivitas edge sama dengan 2.

Jika kita ingin membuat sebuah graph tidak terhubung, kita perlu mengetahui cara membuat sebuah graph tidak terhubung tanpa melibatkan penghapusan *redudant edge*. Perhatikan graph pada Gambar 10.4. Kita dapat membuat *G* tidak terhubung dengan menghapus tiga edge, *uw*, *ux* dan *vx.* Graph tersebut akan tetap terhubung jika kita hanya menghapus dua edge di antara tiga edge tersebut. Selanjutnya , kita juga dapat membuat *G* tidak terhubung dengan menghapus empat edge *uw*,*wx,xz,yz.* Perhatikan bahwa edge *yz* adalah redundant edge (tidak perlu dihapus), karena kita hanya perlu menghapus edge *uw,wx,xz* untuk membuat *G* tak terhubung. Himpunan edge di mana tidak ada edge yang redundant, seperti himpunan {*uw,ux,vx*}, {*wy,xz*} atau {*yt*} disebut sebuah *cutset*.



Gambar 10.4 Contoh graph untuk ilustrasi *redudant edge* dan *cutset*

Definisi 10.4
Sebuah cutset pada graph terhubung *G* adalah sebuah himpunan edge *S* dengan dua sifat berikut.

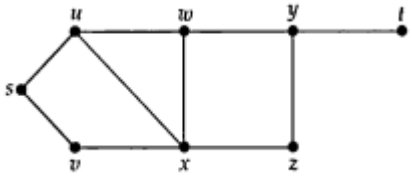
1. Penghapusan semua edge di *S* membuat *G* tidak terhubung
2. Penghapusan beberapa (tidak semua) edge di *S* tidak membuat *G* tidak terhubung.

Catatan :

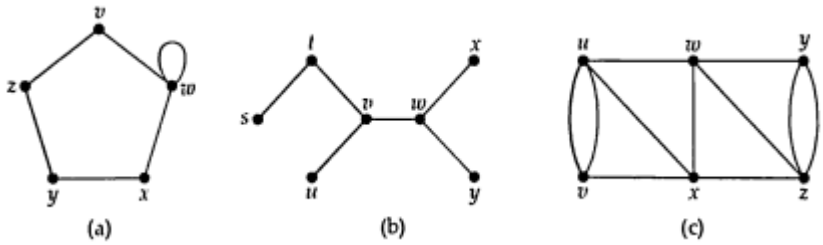
1. Dua cutset dari sebuah graph tidak perlu memiliki jumlah edge yang sama. Sebagai contoh, pada graph di Gambar 10.4, himpunan {*uw,ux,vx*} , {*wy,xz*}, {*yt*} adalah semua cutset yang memiliki jumlah edge berbeda.
2. Konektivitas edge $\lambda(G)$ dari graph *G* adalah nilai dari ukuran cutset terkecil dari *G*. Sebagai contoh pada graph di Gambar 10.4 memiliki $\lambda(G)=1$.

Latihan 10.1

1. Himpunan edge mana yang merupakan cutset dari graph berikut.
   a. {*su,sv*}
   b. {*yt,yz*}
   c. {*ux,wx,yz*}
   d. {*wx,xz,yz*}
   e. {*ux,vx,wx,yz*}
   f. {*uw,wx,wy*}

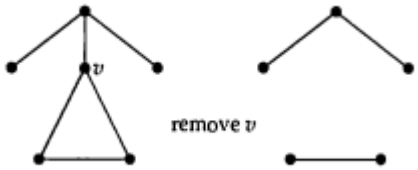


2. Tulislah nilai $\lambda(G)$ pada masing-masing graph G berikut!



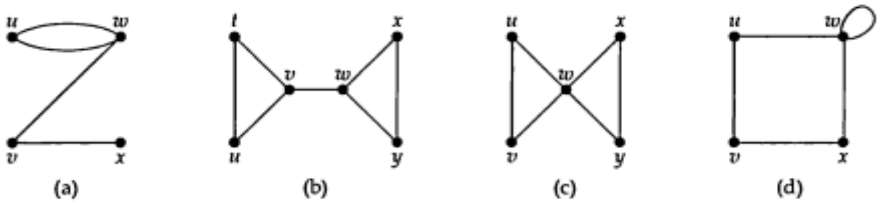
Konektivitas Vertex

Kita juga mempelajari konektivitas dalam konteks jumlah minimal vertex yang jika dihapus membuat suatu graph menjadi tidak terhubung. Ketika kita menghapus sebuah vertex, kita harus menghapus edge yang insiden dengan vertex tersebut seperti ditunjukan pada Gambar 10.5



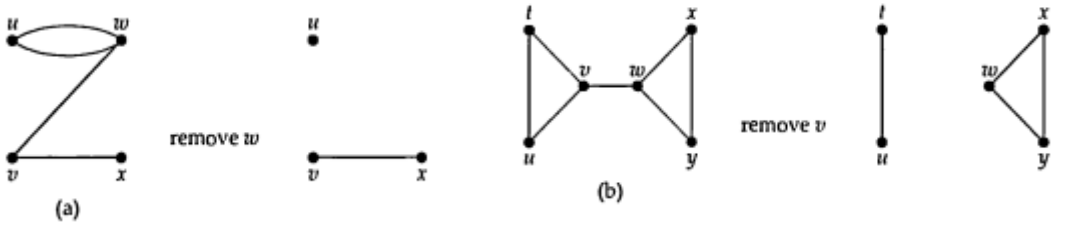
Gambar 10.5 Penghapusan sebuah vertex dan edge yang insiden dengannya

Perhatikan graph pada Gambar 10.6 berikut.

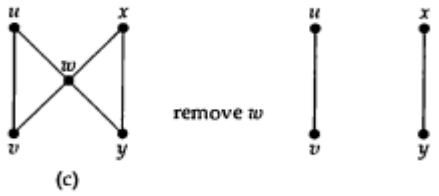


Gambar 10.6 Graph untuk ilustrasi konektivitas vertex

Graph (a) dan (b) dapat menjadi tidak terhubung dengan menghapus sebuah vertex misalnya *v* atau *w* seperti ditunjukan sebagai berikut.

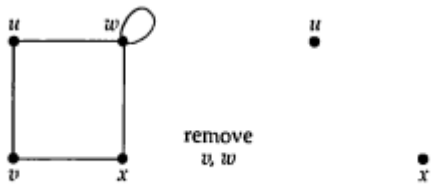


Graph (c) juga dapat menjadi tidak terhubung dengan menghapus sebuah vertex *w* seperti ditunjukan sebagai berikut.



(c)

Pada graph (d), graph tersebut tetap terhubung jika kita hanya menghapus sebuah vertex saja. Pada graph ini kita perlu menghapus dua vertex yang tidak bertetangga, misalnya vertex *v* dan *w*.



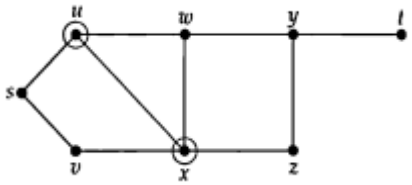
(d)

Definisi 10.5
Konektivitas (atau konektivitas vertex) $\kappa(G)$ pada graph terhubung *G* (selain sebuah graph komplit) adalah jumlah vertex terkecil yang jika dihapus membuat *G* tidak terhubung.

Sebagai contoh, graph (a), (b) dan (c) memiliki konektivitas 1, dan graph (d) memiliki konektivitas 2. Untuk graph komplit kita dapat meninjau definisi berikut.

Definisi 10.6
Konektivitas $\kappa(K_n)$ dari sebuah graph komplit K_n *(n≥3),* adalah *n-1*

Perhatikan graph pada Gambar 10.7. Kita dapat membuat graph *G* tidak terhubung dengan menghapus *u* dan *x*. Tetapi graph tersebut tetap terhubung jika kita hanya menghapus sebuah vertex dari kedua vertex tersebut. Kita juga dapat membuat *G* tidak terhubung dengan menghapus dua vertex *y* dan *z*, namun vertex *z* adalah redundant vertex (tidak perlu dihapus), karena kita hanya perlu menghapus vertex *y* untuk membuat *G* tidak terhubung. Sebuah himpunan vertex di mana tidak ada vertex yang redundant, seperti himpunan {*u,x*} atau {*y*} disebut *cutset vertex*.



Gambar 10.7 Graph untuk ilustrasi redudant vertex

Definisi 10.7
Sebuah cutset vertex pada graph terhubung *G* adalah himpunan vertex *S* dengan dua sifat berikut.

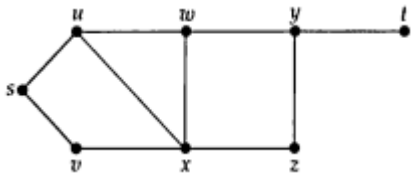
1. Penghapusan semua vertex di *S* membuat *G* tidak terhubung
2. Penghapusan beberapa (tidak semua) vertex di *S* tidak membuat *G* tidak terhubung.

Catatan :

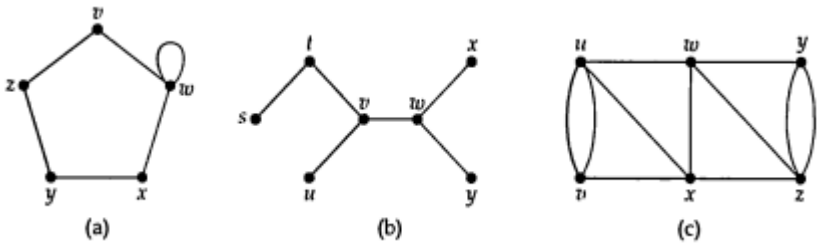
1. Dua cutset vertex pada sebuah graph tidak perlu memiliki jumlah vertex yang sama. Sebagai contoh, pada graph di Gambar 10.7, himpunan $\{u,x\}$, $\{y\}$ keduanya adalah cutset vertex dengan ukuran berbeda.
2. Konektivitas $\kappa(G)$ untuk sebuah graph *G* adalah nilai ukuran cutset vertex terkecil dari *G*. Sebagai contoh untuk graph di Gambar 10.7, memiliki $\kappa(G)=1$.

Latihan 10.2

1. Himpunan vertex mana yang merupakan cutset vertex untuk graph berikut?
   a. $\{u,v\}$
   b. $\{v,w\}$
   c. $\{u,x,y\}$
   d. $\{w,z\}$



2. Tulislah nilai $\kappa(G)$ untuk setiap graph *G* berikut.



Nilai konektivitas vertex $\kappa(G)$, tidak akan melebihi nilai konektivitas edge $\lambda(G)$, juga tidak melebihi derajat vertex terkecil $\delta(G)$. Pertidaksamaan ini berlaku pada semua graph terhubung sebagai berikut.

Teorema 10.1
Misalkan *G* adalah sebuah graph terhubung dengan derajat vertex terkecil $\delta(G)$. Maka berlaku,

$$\kappa(G) \leq \lambda(G) \leq \delta(G)$$

Garis Besar Pembuktian

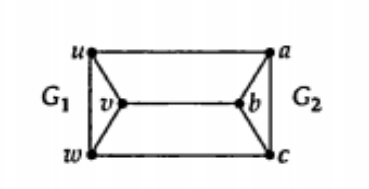
Jika G adalah sebuah graph komplit, K_n, maka $\kappa(G) = \lambda(G) = \delta(G) = n-1$.

Jika G adalah graph terhubung yang bukan K_n, dan jika v adalah vertex dengan derajat $\delta(G)$, maka G dapat menjadi tidak terhubung dengan menghapus sebanyak $\delta(G)$ edge yang insiden dengan v. Oleh karena itu $\lambda(G)$, jumlah minimal edge yang dihapus agar G tidak terhubung tidak melebihi $\delta(G)$. Sehingga,

$$\lambda(G) \leq \delta(G)^*$$

Kita harus membuktikan $\kappa(G) \leq \lambda(G)$ ketika G bukan graph komplit.

Misalkan G adalah graph sederhana terhubung dengan n vertex dan konektivitas edge $\lambda(G)$. Karena G bukan graph komplit K_n, derajat titik terkecil yang paling besar pasti adalah $n-2$, dengan persamaan (*), maka $\lambda(G) \leq n-2$. Ada setidaknya satu himpunan dari $\lambda(G)$ edge yang jika dihapus membuat G tidak terhubung menjadi dua komponen G_1 dan G_2, seperti diilustrasikan pada Gambar 10.8



Gambar 10.8 Graph untuk ilustrasi pembuktian teorema 10.1

Pada Gambar 10.8, $\lambda(G)=3$ dan kita dapat membuat graph tidak terhubung dengan menghapus tiga edge, misalnya ua, vb dan wc. Kita juga dapat menghapus edge ini dengan menghapus paling banyak 3 vertex, karena kita hanya dapat menghapus sebuah vertex yang sesuai, pada pangkal dan ujung edge tersebut. Sebagai contoh pada kasus di atas, kita dapat menghapus vertex u,v dan c.

Misalkan kita menghapus $\lambda(G)$ edge satu pada satu waktu. Karena ada paling banyak n-2 edge yang dihapus, dan pada setiap tahap kita dapat menghapus edge dari G_1 atau dari G_2, kita bisa mendapatkan ini dengan menghapus himpunan vertex yang membuat G_1 ataupun G_2 tidak kosong. Oleh karena itu, jumlah minimal vertex yang dihapus tidak bisa melebihi $\lambda(G)$. Sehingga $\kappa(G) \leq \lambda(G)$.

Misalnya G bukan graph sederhana dan terhubung, dan misalnya G' adalah graph yang diperoleh dengan menghapus loop dan menganti multiple edge dengan satu edge. Jika

$\kappa(G') \leq \lambda(G')$ untuk graph sederhana G', maka $\kappa(G) \leq \lambda(G)$, karena perubahannya tidak merubah $\kappa(G)$, dan hanya mengurangi nilai $\lambda(G)$, maka ,

$$\kappa(G) = \kappa(G') \leq \lambda(G') \leq \lambda(G)$$

Sehingga, untuk setiap graph terhubung G, berlaku,

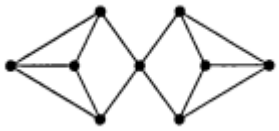
$$\kappa(G) \leq \lambda(G) \leq \delta(G)$$

Catatan :

Adalah mungkin untuk pertidaksamaan pada teorema 10.1 untuk ditulis sebagai,

$$\kappa(G) < \lambda(G) < \delta(G).$$

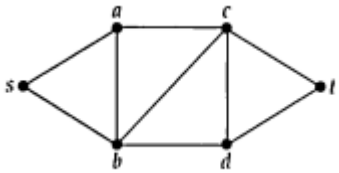
Sebagai contoh untuk graph berikut memiliki $\kappa(G)=1$, $\lambda(G)=2$ dan $\delta(G)=3$.



Teorema Manger

Definisi 10.8
Misalkan *G* adalah graph terhubung dan misalnya *s* dan *t* adalah vertex di *G*.
Sebuah path antara *s* dan *t* disebut path *st*.
Dua atau lebih path *st* adalah edge-disjoint jika kedua path tidak memiliki edge yang sama, dan vertex-disjoint jika keduanya tidak memiliki vertex yang sama, selain *s* dan *t*.

Sebagai contoh, pada Gambar 10.9, path *sact* dan *sbdt* keduanya adalah edge-disjoint dan vertex-disjoint. Path *sact* dan *sbct* keduanya tidak edge-disjoint dan tidak vertex-disjoint, karena mereka memiliki vertex yang sama yaitu *c*, dan edge yang sama yaitu *ct*. Path *sact* dan *sbcdt* adalah edge-disjoint, namun tidak vertex-disjoint, karena mereka memiliki vertex *c* yang sama.

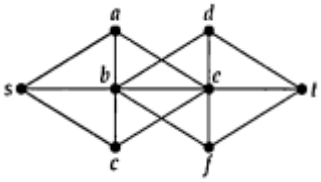


Gambar 10.9 Contoh graph untuk ilustrasi path *st*

Latihan 10.3

1. Tulislah :
   a. Tiga path *st* yang edge-disjoint.
   b. Dua path *st* yang edge disjoint namun tidak vertex-disjoint.
   c. Dua path *st* yang vertex-disjoint.

   Apakah graph tersebut memuat tiga path *st* yang vertex-disjoint?



2. Buktikan bahwa jika dua path *st* pada sebuah graph adalah vertex-disjoint, maka dia juga merupakan edge-disjoint.
3. Berikan sebuah contoh graph yang memiliki dua path *st* yang edge-disjoint dan tidak vertex-disjoint.

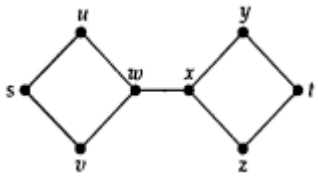
Definisi 10.9
Misalkan *G* adalah graph terhubung, dan misalkan *s* dan *t* adalah vertex pada *G.*
Suatu edge disebut memisahkan *s* dari *t,* jika penghapusan edge tersebut menghancurkan semua path antara *s* dan *t.*
Suatu vertex disebut memisahkan *s* dari *t,* jika penghapusan vertex tersebut menghancurkan semua path antara *s* dan *t.*

Sebagai contoh pada Gambar10.9,

edge *ac,bc, bd* memisahkan *s* dan *t*,
edge *sa,ac,bc,bd,dt* memisahkan *s* dan *t*.
vertex *b* dan *c* memisahkan *s dan t*
vertex *a*, *b* dan *d* memisahkan *s* dan *t*

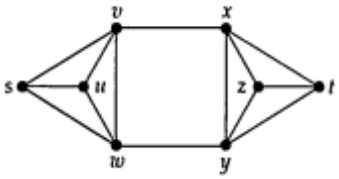
Kita akan menunjukan bagaimana ide ini berhubungan dengan path *st* dan edge-disjoint dengan contoh-contoh berikut.

Contoh 10.1



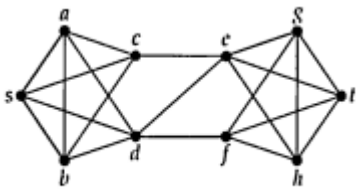
Pada graph ini, sebuah edge *wx* memisahkan *s* dari *t*. Artinya tidak akan ada dua path *st* yang edge-disjoint, karena setiap path harus memuat edge *wx*, maka ada paling banyak sebuah path *st* yang edge-disjoint.

Contoh 10.2



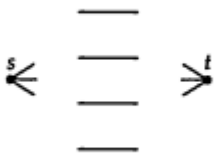
Pada graph ini, dua edge *vx* dan *wy* memisahkan *s* dari *t*. Artinya ada paling banyak dua path *st* yang edge-disjoint, karena setiap path harus memuat satu dari dua edge tersebut.

Contoh 10.3



Pada graph ini, ada tiga edge *ce*, *de*, dan *df* yang memisahkan *s* dari *t*. Artinya ada paling banyak tiga buah path *st* yang edge-disjoint, karena setiap path *st* harus memuat satu dari ketiga edge tersebut.

Lebih umumnya, perhatikan ilustrasi sebuah himpunan edge yang memisahkan *s* dari *t* pada sebarang graph terhubung. Karena penghapusan edge tersebut menghancurkan semua path antara *s* dan *t*, setiap path *st* memuat setidaknya satu edge di antara edge tersebut. Oleh karena itu, jumlah maksimal dari path *st* yang edge-disjoint tidak dapat melebihi jumlah edge pada himpunan tersebut seperti ditunjukan pada Gambar 10.10.



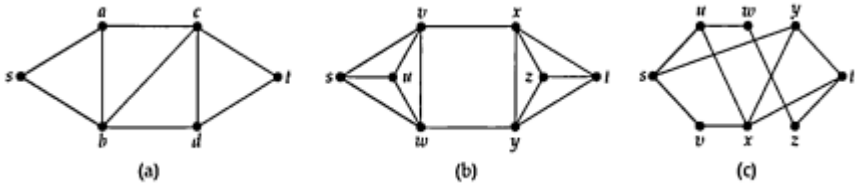
Gambar 10.10. Hubungan edge yang memisahkan *s* dan *t* dengan path *st* yang edge-disjoint

Teorema 10.2
Misalkan *G* adalah graph terhubung, dan misalnya *s* dan *t* adalah vertex dari *G*.
Jumlah maksimal path *st* yang edge-disjoint sama dengan jumlah minimal edge yang memisahkan *s* dari *t*.

Catatan

Kita mengetahui bahwa jika kita dapat menemukan *k* buah path *st* yang edge-disjoint dan *k* edge yang memisahkan *s* dari *t* (dengan nilai *k* yang sama), maka *k* adalah jumlah maksimal dari edge-disjoint path *st* dan jumlah minimum dari edge yang memisahkan *s* dari *t*. *k* edge yang memisahkan *s* dari *t* ini tentunya dari sebuah cutset. Oleh karena itu, ketika kita meninjuanya, kita hanya perlu menyediakan cutset yang penghapusannya membuat G tidak terhubung menjadi dua komponen, satu memuat *s* dan satu memuat *t.*

Latihan 10.4
Dengan menemukan *k* buah path *st* yang edge-disjoint, dan *k* edge yang memisahkan *s* dari *t* (dengan nilai *k* yang sama), dan memakai bentuk edge dari teorema Menger, temukan jumlah maksimal dari path *st* yang edge-disjoint untuk setiap graph berikut.



Kita dapat memakai teorema Manger untuk mendapatkan sebuah hasil tentang konektivitas edge. Ingat kembali bahwa konektivitas edge λ(G) dari graph terhubung G adalah jumlah terkecil dari edge yang penghapusannya membuat G tidak terhubung. Sehingga, dengan teorema Manger, setidaknya ada λ(G) path yang edge-disjoint diantara setiap pasang vertex.

Akibat Teorema Manger untuk Graph (Bentuk Edge)
Sebuah graph terhubung *G* memiliki konektivitas edge *l*, jika hanya jika ada *l* buah path atau lebih yang edge-disjoint antara setiap pasangan vertex di *G*, dan ada tepat *l* path yang edge-disjoint antara setidaknya sepasang vertex di *G.*

Catat bahwa, pada contoh 10.1, 10.2, 10.3, konektivitas edge-nya masing-masing adalah 1,2 dan 3.

Analog Teorema Manger

Kini kita akan mempelajari beberapa analog dari teorema Manger, dimulai dengan teorema Manger untuk digraph, selanjutnya untuk bentuk vertex baik untuk graph ataupun digraph.

Teorema Menger untuk Digraph (Bentuk Arc)

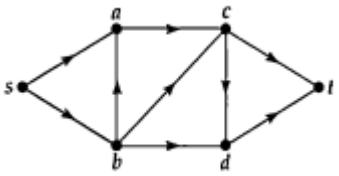
Definisi 10.10
Misalkan *D* adalah digraph terhubung dan misalnya *s* dan *t* adalah vertex di *D*.
Sebuah path antara *s* dan *t* disebut path *st*.
Dua atau lebih path *st* adalah arc -disjoint jika kedua path tidak memiliki arc yang sama, dan vertex-disjoint jika keduanya tidak memiliki vertex yang sama selain *s* dan *t*.

Sebagai contoh, pada Gambar 10.11, ditunjukan bahwa,

Path *sact* dan *sbdt* keduanya adalah arc-disjoint dan vertex-disjoint.

Path *sact* dan *sbct* keduanya tidak arc -disjoint dan tidak vertex-disjoint.

Path *sact* dan *sbcdt* adalah arc-disjoint, namun tidak vertex-disjoint.



Gambar 10.11 Contoh digraph untuk ilustrasi path *st*

Definisi 10.11
Misalkan *D* adalah graph terhubung, dan misalkan *s* dan *t* adalah vertex pada *D.*
Sebuah arc disebut memisahkan *s* dari *t,* jika penghapusan arc tersebut menghancurkan semua path antara *s* dan *t.*
Sebuah vertex disebut memisahkan *s* dari *t,* jika penghapusan vertex tersebut menghancurkan semua path antara *s* dan *t.*

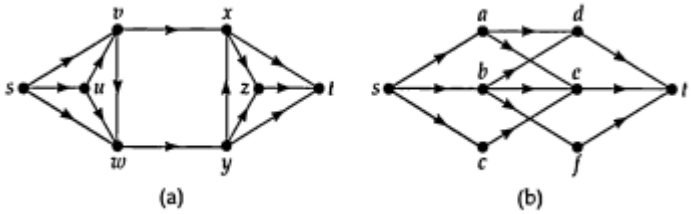
Sebagai contoh, pada digraph pada Gambar 10.11, menunjukan :

Arc *ac, bc,bd* memisahkan *s* dari *t*, seperti juga arc *sa,ac,bc,bd,dt.*

Vertex *b* dan *c* memisahkan *s* dari *t*, seperti juga vertex *a,b,d*.

Teorema 10.3
Misalkan *D* adalah digraph terhubung, dan misalkan *s* dan *t* adalah vertex dari *D.*
Maka jumlah maksimal path *st* yang arc-disjoint sama dengan jumlah minimal arc yang memisahkan *s* dari *t.*

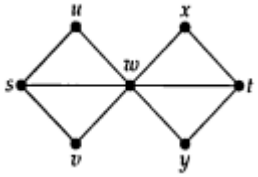
Latihan 10.5
Dengan menemukan *k* arc-disjoint st-path, dan *k* yang memisahkan *s* dari *t* (dengan nilai *k* yang sama) dan menggunakan bentuk arc dari teorema Manger, temukan jumlah maksimum *st*-path yang arc-disjoint untuk setiap digraph berikut.



Teorema Manger untuk Graph (Bentuk Vertex)

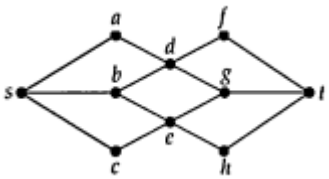
Kita telah melihat bagaimana teorema Manger (bentuk edge) menghubungkan jumlah path *st* yang edge-disjoint pada sebuah graph dengan jumlah terkecil edge yang memisahkan *s* dari *t* , dan bagaimana hasil ini berhubungan dengan konektivitas edge. Kita kini akan mempelajari teorema yang analog untuk path *st* yang vertex-disjoint.
Seperti sebelumnya, kita pelajari konsep ini dengan contoh-contoh.

Contoh 10.4



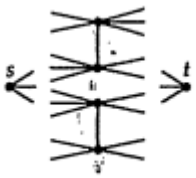
Pada graph ini, vertex *w* memisahkan *s* dari *t*. Oleh karena itu, tidak akan ada dua path *st* yang vertex-disjoint, karena semua path *st* harus memuat vertex *w*, jadi ada paling banyak sebuah path *st* yang vertex-disjoint.

Contoh 10.5



Pada graph ini, vertex *d* dan *e* memisahkan *s* dari *t*. Sehingga paling banyak ada dua path *st* yang vertex-disjoint, karena semua path *st* harus memuat kedua vertex ini.

Secara umum, diberikan sebuah himpunan vertex (tidak terasuk *s* dan *t*) memisahkan vertex tidak bertetangga *s* dan *t* pada sebuah sebarang graph terhubung. Karena penghapusan vertex ini menghancurkan semua path antara *s* dan *t*, setiap *st* path memuat memuat setidaknya satu diantara vertex tersebut. Oleh karena itu, jumlah maksimum vertex-disjoint path *st* tidak dapat melebihi jumlah vertex pada hiimpunan ini.

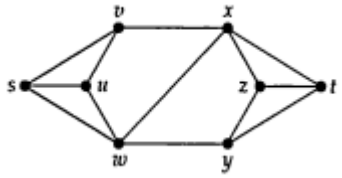


Teorema 10.4
Misalkan *G* adalah graph terhubung, dan misalkan *s* dan *t* adalah vertex yang tidak bertetangga pada *G*.
Maka jumlah maksimum dari path *st* yang vertx-disjoint sama dengan jumlah minimum dari vertex yang memisahkan *s* dan *t.*

Catatan

Kita ketahui bahwa, jika kita dapat menemukan *k* buah path *st* yang vertex-disjoint dan *k* vertex yang memisahkan *s* dari *t* (untuk nilaii k yang sama), maka k adalah jumlah maksimum dari path *st* yang vertex-disjoint dan jumlah minimum vertex yang memisahkan *s* dari *t*. *k* vertex ini memisahkan *s* dari *t* pastilah membentuk sebuah cutset vertex. Oleh karena itu, ketika kita meninjaunya, kita hanya membutuhkan cutset vertex yang jika dihapus menyebabkan *G* tidak terhubung ke dalam dua komponen, satu memuat *s* dan satu memuat *t*.

Latihan 10.6
Dengan menemukan *k* buah path *st* yang vertex-disjoint, dan *k* vertex yang memisahkan *s* dari *t* (dengan nilai *k* yang sama) dan menggunakan bentuk vertex dari teorema Menger, temukan jumlah maksimum path *st* yang vertex-disjoint untuk graph berikut.



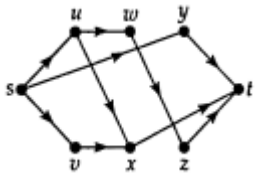
Kita dapat memakai teorema Manger untuk menghasilkan konektivitas vertex. Ingat kembali bahwa konektivitas $\kappa(G)$ pada sebuah graph *G* (selain graph komplit) adalah jumlah minimal vertex yang jika dihapus membuat G tidak terhubung. Jadi, dengan bentuk vertex untuk teorema Menger, ada setidaknya $\kappa(G)$ path yang vertex-disjoint antara setiap pasangan vertex yang diberikan.

Akibat Teorema Menger untuk Graph (Bentuk Vertex)
Sebuah graph terhubung *G* (selain graph komplit) memiliki konektivitas vertex *k* jika dan hanya jika setiap pasangan vertex yang tidak bertetangga di *G* dihubungkan dengan *k* atau lebih path yang vertex-disjoint, dan setidaknya terdapat sepasang vertex tidak bertetangga yang dihubungkan dengan tepat *k* path yang vertex-disjoint.

Catat bahwa contoh 10.4 dan 10.5, masing-masing memiliki konektivitas vertex 1 dan 2.

Teorema 10.5
Misalnya *D* adalah digraph terhubung, dan misalnya *s* dan *t* adalah vertex yang tidak bertetangga pada *D*.
Maka jumlah maksimal path *st* yang vertex-disjoint sama dengan jumlah minimum vertex yang memisahkan *s* dari *t.*

Latihan 10.7
Dengan menemukan *k* buah path *st* yang vertex-disjoint dan *k* vertex yang memisahkan *s* dari *t* (dengan nilai *k* yang sama), dengan menggunakan bentuk vertex teorema Menger, temukan jumlah maksimal path *st* yang vertex-disjoint pada digraph berikut.



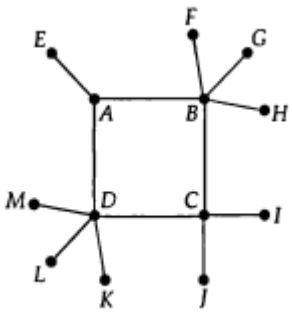
Catatan Sejarah
Bentuk vertex adalah versi teorema Manger yang dibuktikan oleh K.Menger pada tahun 1927. Akibat teorema ini dibuktikan lima tahun kemudian oleh H.Whitney. Bentuk edge dan bentuk arc teorema Menger dibuktikan pada tahun 1955 oleh L.R.Ford dan D.R.Fulkerson.

Studi Kasus

Jaringan Telekomunikasi Handal

Pada bagian ini, kita akan meninjau penggunaan teori graph untuk merepresentasikan jaringan telekomunikasi dan menunjukan bagaimana hal ini dapat berguna dalam desain efisiensi jaringan. Sebuah graph untuk jaringan telekomunikasi mungkin memuat sejumlah besar vertex dan edge. Vertex pada graph tersebut mewakili titik atau pelanggan telepon dan edge mewakili penghubung di antara mereka.

Sangat penting untuk memastikan bahwa jaringan seperti ini dapat diandalkan. Sebuah aspek kehandalan yang dapat diukur adalah bahwa panggilan pelanggan selalu dapat terjadi walaupun terdapat beberapa titik atau penghubung yang rusak. Diberikan graph pada Gambar 10.12 yang merepresentasikan sebuah interkoneksi sistem telekomunikasi.



Gambar 10.12 Contoh graph sistem telekomunikasi

Jika saluran *AB* rusak, maka komunikasi antara *A* dan *B* masih dapat dilakukan, namun jika saluran *EA* rusak, maka *E* tidak dapat berkomunikasi dengan yang lain. Jumlah minimal penghubung yang rusak yang membuat sistem tidak berfungsi secara keseluruhan sama dengan konektivitas edge pada graph yang berkoresponden. Demikina juga, konektivitas vertex menunjukan kita, berapa banyak kantor/pelanggan/titik yang rusak rusak sehingga komunikasi antara pelanggan yang tersisa tidak bisa dilakukan.

Sangat penting bagi kita untuk menyediakan path alternatif antara titik sehingga komununikasi antara pelanggan selalu mungkin dilakukan walaupun sebuah path rusak. Lebih jauh, sebuah penghubung atau titik tertentu pada path antara dua titik mungkin merupakan bagian dari path antara pasangan titik lain. Titik atau penghubung tersebut mungkin telah dipakai untuk panggilan lain ketika sebuah panggilan dilakukan, sehingga mencegah panggilan baru dilakukan. Pada kasus ini, kita menyebut bahwa panggilan tersebut diblok.

Untuk dua buah titik tertentu, jumlah maksimum path alternatif yang tidak ada dua darinya melalui titik antara yang sama adalah jumlah maksimal path yang vertex-disjoint. Dengan menggunakan akibat bentuk vertex teorema Menger, jumlah terkecil path antara dua titik tersebut adalah konektivitas vertex. Demikian juga, jumlah maksimal path alternatiif, yang tidak memakai penghubung yang sama adalah jumlah maksimum path yang edge-disjoint antara vertex yang berkoresponden pada graph. Dengan menggunakan akibat bentuk edge teorema Menger, jumlah terkecil path antara dua titik ini adalah konektivitas edge.

Jika kehandalan adalah satu-satunya pertimbangan, sistem telekomunikasi akan memiliki sebanyak mungkin path alternatif antar titik. Ini menyebabkan sistem akan memiliki vertex yang banyak dan konektivitas edge yang banyak. Setiap titik membutuhkan koneksi ke setiap titik lain, dan graph yang bersesuaian dengan kondisi ini adalah graph komplit. Tentu saja, hal ini adalah sesuatu yang tidak praktis untuk diaplikasikan. Para desainer mencoba untuk membuat kemungkinan terbesar nilai konektivitas vertex $\kappa(G)$ dan konektivitas edge $\lambda(G)$ untuk graph G dengan sejumlah vertex dan egde yang diberikan.

Kita ketahui dari teorema 10.1 bahwa untuk setiap graph terhubung G,

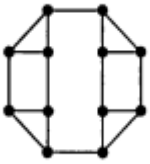
$$\kappa(G) \leq \lambda(G) \leq \delta(G)$$

Dengan $\delta(G)$ adalah derajat titik minimal di G. Misalkan G memiliki n vertex dan m edge. Maka dengan handshaking lemma, jumlah semua derajat vertexnya dalah $2m$. Oleh karena itu rata-rata derajat titik vertex adalah $2m/n$, sehingga derajat minimal vertex tidak akan lebih besar dari nilai tersebut. Dengan mengkombinasikan hasil tersebut, kita mendapatkan pertidaksamaan berikut.

$$\kappa(G) \leq \lambda(G) \leq \delta(G) \leq 2m/n$$

Sebuah graph G yang memiliki konektivitas vertex dan konektivitas edge maksimal yang mungkin untuk setiap graph dengan n vertex dan m edge disebut memiliki konektivitas optimal. Untuk menunjukan bahwa sebuah graph memiliki konektivitas optimal, cukup ditunjukan bahwa $\kappa(G)=2m/n$, karena ini telah menjamin nilai $\kappa(G) = \lambda(G) = \delta(G)$.

Semua graph dengan konektivitas optimal adalah sebuah graph regular (karena derajat vertex terkecil sama dengan rata-rata derajat vertex), tetapi tidak setiap graph regular memiliki konektivitas optimal. Sebagai contoh, graph regular pada Gambar 10.13 memiliki derajat=3, namun $\kappa(G) = \lambda(G) = 2$, jadi G tidak memiliki konektivitas optimal.



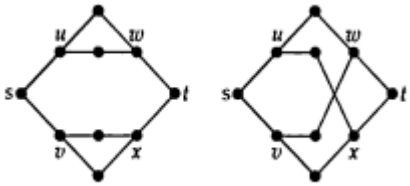
Gambar 10.13 Graph regular yang tidak memiliki konektivitas optimal

Latihan 10.8

1. Tunjukan bahwa graph regular berikut memiliki konektivitas optimal.
   a. Graph cycle C_n ($n \geq 3$)
   b. Graph komplit K_n ($n \geq 3$)
   c. Graph bipartit komplit $K_{r,s}$ ($r \geq 2$)
2. Ada dua graph sederhana yang isomorfik dengan 6 vertex dan 9 edge yang memiliki konektivitas optimal. Gambarlah graph tersebut!
3. Gambarlah sebuah graph regular dengan tujuh vertex dan 14 edge yang tidak memiliki konektivitas optimal.

Kita telah melihat bahwa nilai $\kappa(G)$ dan $\lambda(G)$ untuk sebuah graph G memberikan kita informasi tentang kehandalan sistem telekomunikasi. Namun, nilai ini tidak menunjukan kita keseluruhan cerita. Misalnya dua graph dengan jumlah vertex yang sama, jumlah edge yang sama, dan nilai $\kappa(G)$ dan $\lambda(G)$ yang sama, mungkin saja tidak berkoresponden dengan sistem handal yang sama.

Diberikan graph pada Gambar 10.14. Dua graph tersebut memiliki 10 vertex, 12 edge, dan $\kappa(G)= \lambda(G)=2$. Pada graph pertama, semua path dari s ke t dapat dihancukan dengan menghapus edge pada salah satu cutset dari empat cutset yang memiliki dua edge sebagai berikut : {su,sv}, {wt,xt}, {su,xt} atau {sv,wt}. Namun graph kedua hanya memiliki dua cutset dengan dua edge sebagai berikut ; { su,sv} dan {wt,xt}.



Gambar 10.14 Dua graph dengan konektivitas sama dan kehandalan berbeda

Jika semua edge pada cutset ini memiliki kemungkinan yang sama untuk diblok atau rusak, kita mengharapkan graph kedua lebih handal daripada yang pertama. Oleh karena itu, untuk mendapatkan informasi yang lebih banyak tentang kehandalan sebuah jaringan yang direpresentasikan oleh graph, kita bukan hanya harus mengetahui konektivitas vertex dan konektivitas edge-nya, namun juga semua cutset pada graph tersebut.