TGO – PERTEMUAN 1

PENGERTIAN GRAF

Dosen : Arya Yudhi W, S.Kom, M.Kom.
Email : arya@if.its.ac.id

OUTLINE

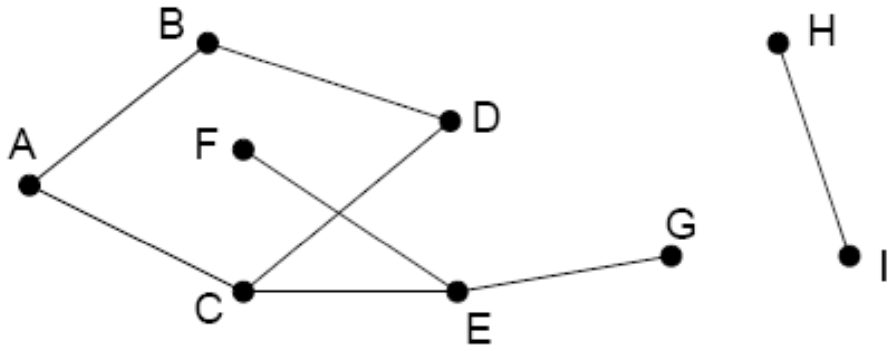
- Pengertian & Definisi Graf
- Simple Graph
- Sub Graph
- Undirected & Directed Graph
- Vertex Degree (derajat)
- Path (lintasan)
- Connection
- Isomorphisme

Definisi Graf

- Secara sederhana **graf didefinisikan sebagai** kumpulan titik yang dihubungkan oleh garis.
- Secara matematis, graf adalah pasangan himpunan (*V*,*E*) dimana *V* adalah himpunan tak kosong yang memiliki elemen disebut *vertices* dan *E* adalah kumpulan dari dua elemen subsets *V* yang disebut *edges*.

Definisi Graf

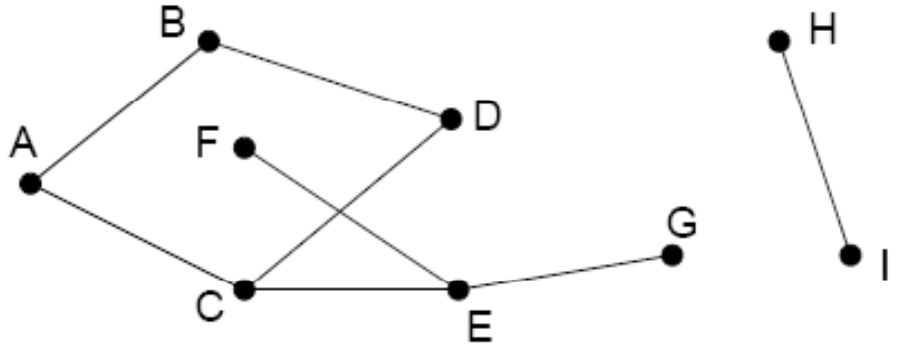
- *Vertices* direpresentasikan dengan titik dan *edges* direpresentasikan dengan garis.



- $V = \{A, B, C, D, E, F, G, H, I\}$, dan
- $E = \{\{A,B\}, \{A,C\}, \{B,D\}, \{C,D\}, \{C,E\}, \{E,F\}, \{E,G\}, \{H,I\}\}$.

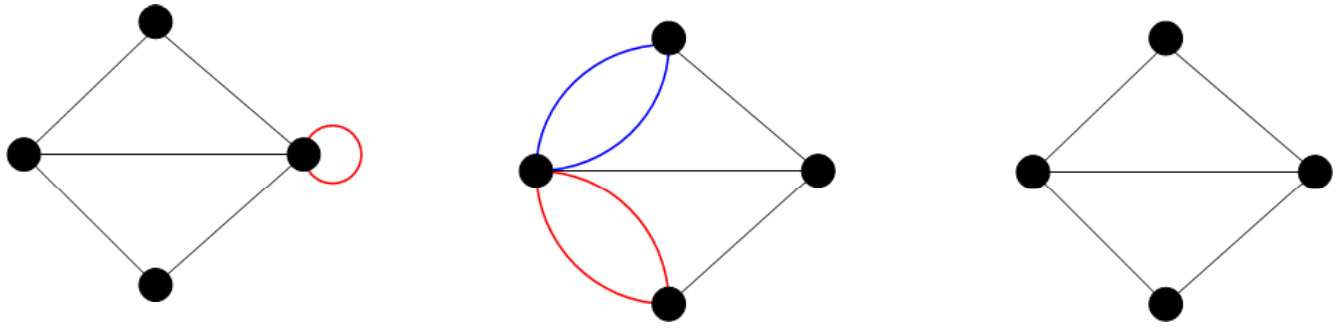
Definisi Graf

- Sebuah *edge* selalu memiliki dua *endpoint*, misalnya *edge* {*H*,*I*} memiliki *endpoint H* dan *I*.
- Graf biasanya digunakan untuk memodelkan objek-objek diskrit dan hubungan antar objek-objek tersebut.



Graf Sederhana (Simple Graph)

- Graf sederhana adalah graf yang tidak mengandung *loops* atau *multiple edges*.
- *Loops* adalah *edge* yang memiliki *endpoint* sama, sedangkan *multiple edges* adalah *edge* yang memiiki pasangan *endpoint* sama.



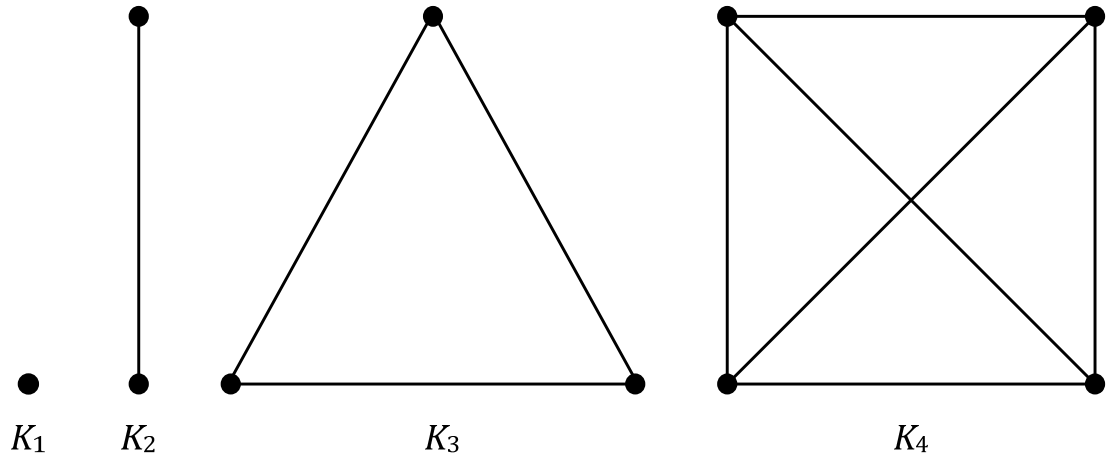
Graf Sederhana (Simple Graph)

- Jenis-jenis Graf Sederhana
  - Graf Komplit (Complete Graph K_n)
  - Cycle (C_n)
  - Wheel (W_n)

Graf Sederhana (Simple Graph)

- Graf Komplit (Complete Graph K_n)

  Adalah graf dimana setiap pasang *vertices* selalu memiliki sebuah *edge*.

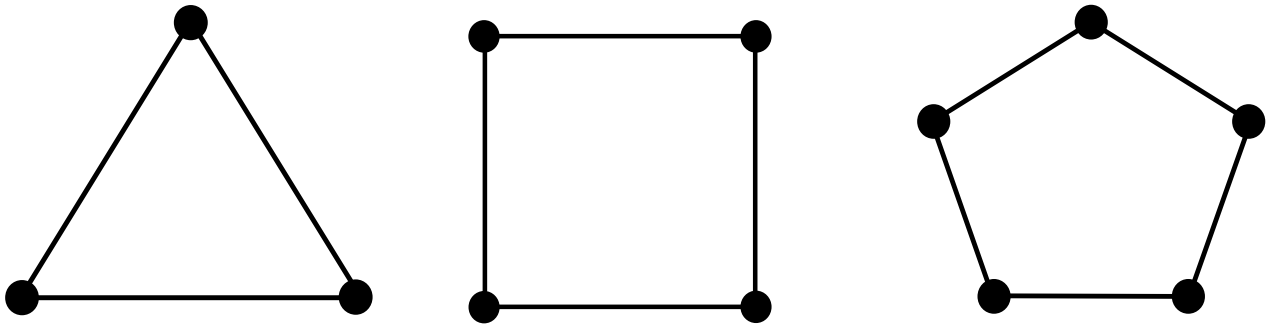


Graf Sederhana (Simple Graph)

- Graf *cycle* (C_n)
  adalah graf $C = (V,E)$ dengan bentuk
  $V = \{v_1, v_2, v_3, v_4, \ldots, v_n\}$ dan
  $E = \{\{ v_1, v_2\}, \{ v_2, v_3\}, \{ v_3, v_4\}, \ldots, \{ v_n, v_1\} \}$,
  dimana $n \geq 3$ dan $v_1, v_2, v_3, v_4, \ldots, v_n$ adalah *vertices* yang berbeda.

Graf Sederhana (Simple Graph)

Graf *cycle* disimbolkan dengan C_n dimana *n* adalah banyanya *vertices*.



C_3 C_4 C_5

Graf Sederhana (Simple Graph)

- Graf *wheel*

  adalah graf *cyle* yang ditambahi sebuah *vertex* baru (v_m) dimana v_m terhubung ke seluruh vertices yang ada. Sehingga graf W = (V,E) dengan bentuk

  $V = \{v_1, v_2, v_3, v_4, \dots, v_n, v_m\}$ dan

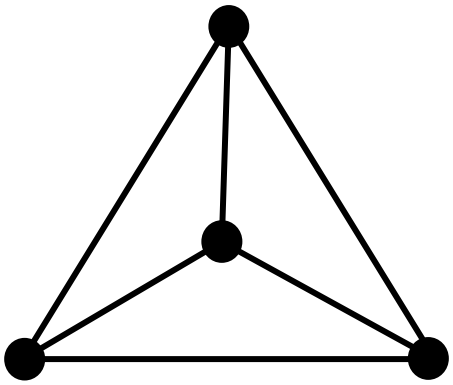
  $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}, \dots, \{v_n, v_1\}, \{v_1, v_m\}, \{v_2, v_m\}, \{v_3, v_m\}, \dots, \{v_n, v_m\}\}$,

  dimana $n \geq 3$ dan $v_1, v_2, v_3, v_4, \dots, v_n, v_m$ adalah *vertices* yang berbeda.

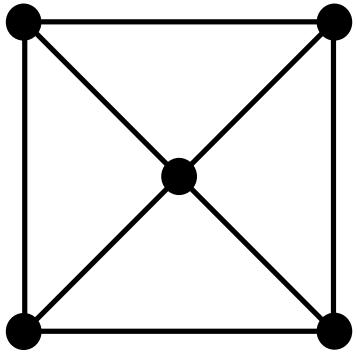
C_4 C_5

Graf Sederhana (Simple Graph)

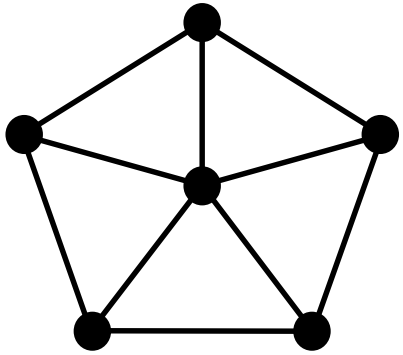
- Graf *wheel* disimbolkan dengan W_n dimana $n+1$ adalah banyanya *vertices*. Gambar 1.5 menunjukkan graf *wheel* W_n untuk n=3, 4, dan 5.



W_3



W_4



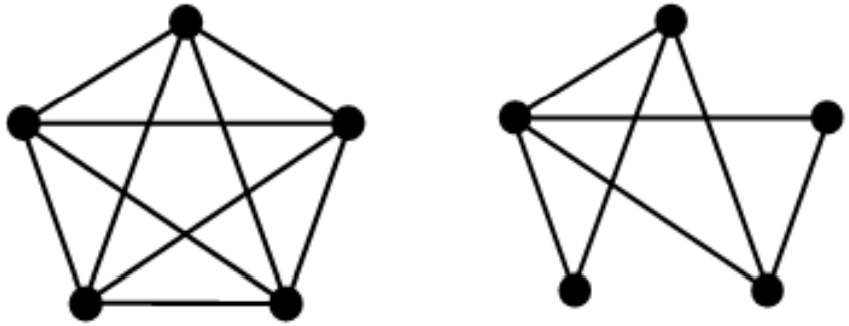
W_5

Graf dan Sub-graf

- Jika $V(G)$ dan $E(G)$ adalah himpunan vertices dan edge pada graf G, serta $V(H)$ dan $E(H)$ adalah himpunan vertices dan edge pada graf H, maka graf H disebut sub-graf dari graf G jika dan hanya jika $V(H) \subseteq V(G)$ dan $E(H) \subseteq E(G)$.

Graf dan Sub-graf

- Jadi, jika *e* adalah edge pada graph *H* yang menghubungkan vertex *v* dan *u* maka *e* juga merupakan edge pada graph *G* yang menghubungkan vertex *v* dan *u* di graf *G*.
- Gambar di bawah menunjukkan graf *H* yang merupakan sub-graf dari graf *G*.

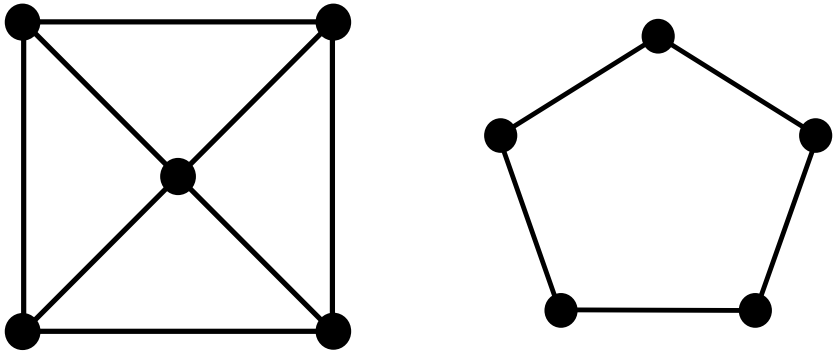


Graf G

Graf H

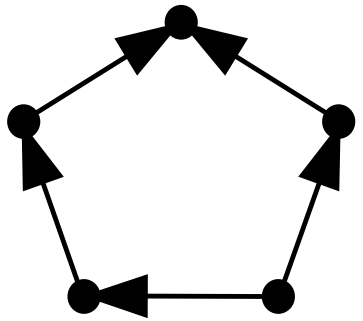
Graf Berarah dan Tak Berarah

- Graph tak-berarah (*undirected graph*) adalah graf yang tidak memiliki orientasi arah pada setiap edge yang dimiliki. Penulisan edge tidak memperhatikan urutan. Penulisan edge $e = (u,v)$, dimana edge e adalah edge yang menghubungakan vertex u dan v sama saja dengan penulisan $e = (v,u)$.



Graf Berarah dan Tak Berarah

- Graf berarah (*directed graph*/ *digraph*) adalah graf yang memiliki orientasi arah pada setiap edge yang dimiliki.
- Sehingga, penulisan edge $e = (u,v)$ untuk edge e yang mengubungkan vertex u dan v berbeda maknanya dengan penulisan edge $e = (v,u)$ yang menghubungkan vertex v dan u.
- Setiap edge pada digraph biasa juga disebut dengan *arc*.



Derajat Vertex (Vertex Degree)

- Untuk menentukan berapa jumlah degree pada setiap vertex harus terlebih dahulu diketahui apakah graf tersebut merupakan graf berarah atau Graf Tak
- Directed Graph

  $d_{in}(v)$: Jumlah Edge yang masuk ke vertex

  $d_{out}(v)$: Jumlah Edge yang keluar dari vertex

  sehingga $d(v) = d_{in}(v) + d_{out}(v)$
- Undirected Graph

  Jumlah Degree dihitung dari jumlah edge yang menyentuh vertex. Jika ada Loop,maka Degree dihitung 2

Derajat Vertex (Vertex Degree)

- Untuk menentukan berapa jumlah degree pada setiap vertex harus terlebih dahulu diketahui apakah graf tersebut merupakan graf berarah atau Graf Tak
- Directed Graph

  $d_{in}(v)$: Jumlah Edge yang masuk ke vertex

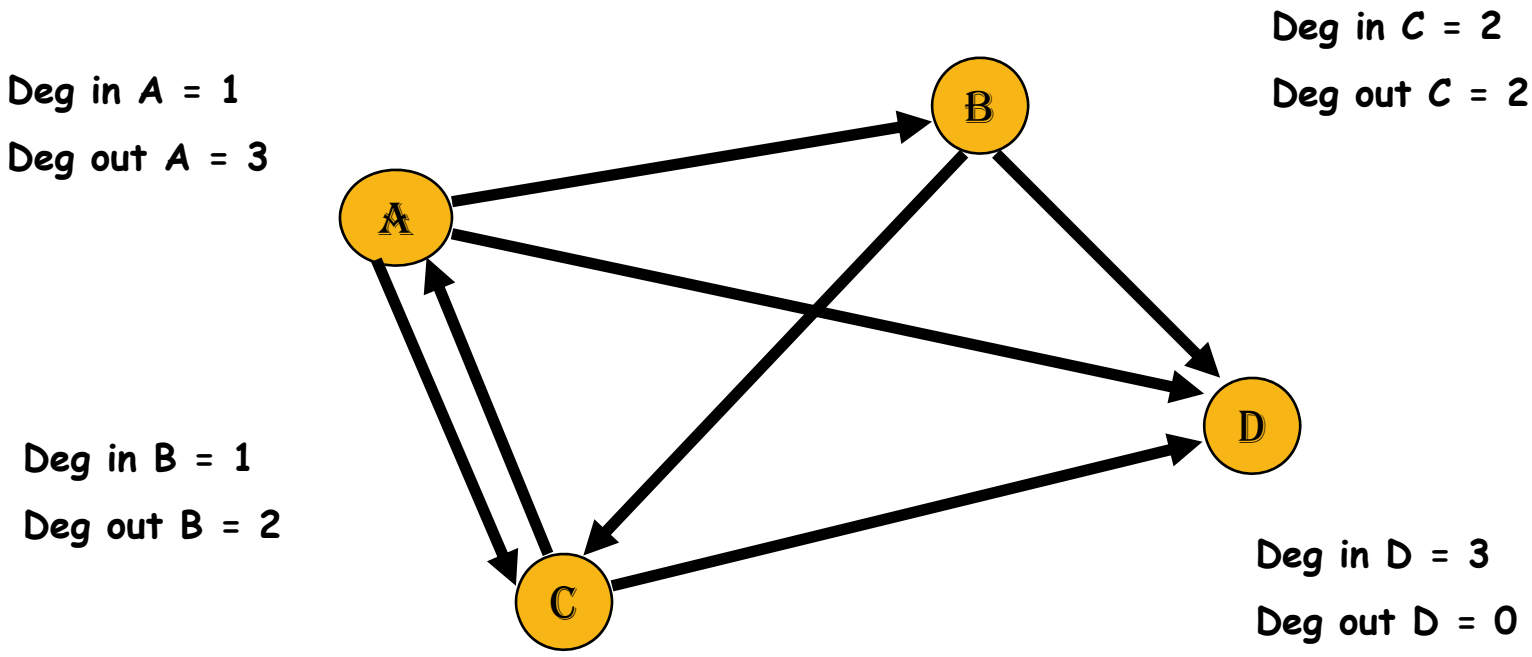
  $d_{out}(v)$: Jumlah Edge yang keluar dari vertex

  sehingga $d(v) = d_{in}(v) + d_{out}(v)$
- Undirected Graph

  Jumlah Degree dihitung dari jumlah edge yang menyentuh vertex. Jika ada Loop,maka Degree dihitung 2

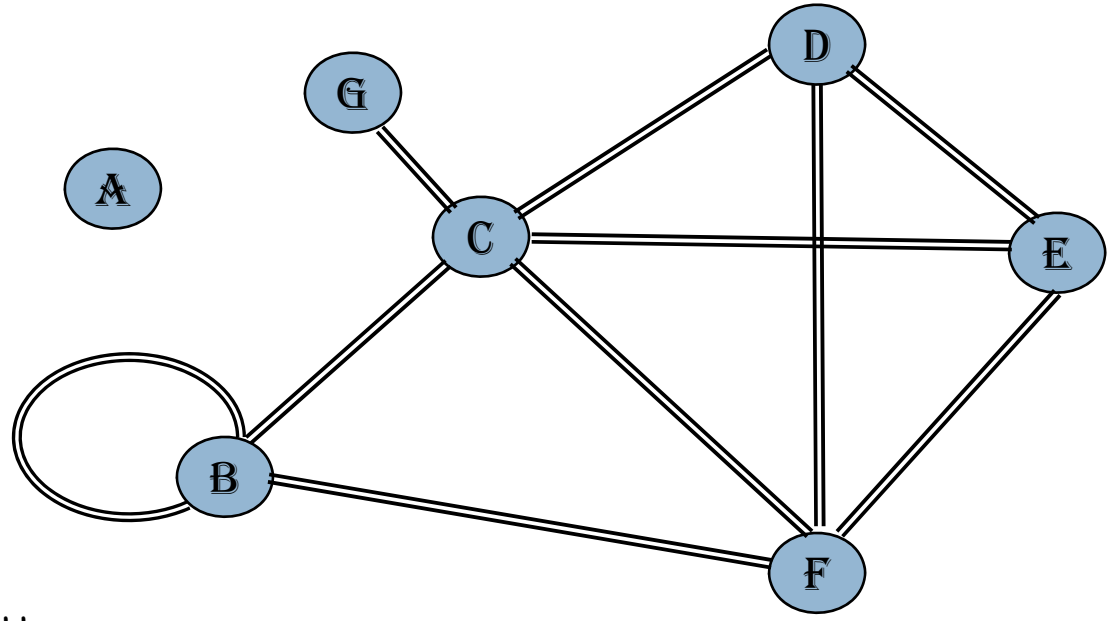
Derajat Vertex (Vertex Degree)

- Contoh Directed Graph



Derajat Vertex (Vertex Degree)

- Contoh Undirected Graph



deg (A) = 0 ;

deg (B) = 4 ;

deg (C) = 4 ;

deg (D) = 3 ;

deg (E) = 3 ;

deg (F) = 4 ;

deg (G) = 1 ;

Catt :

Vertex A dinamakan **Isolated Vertex** (tidak mempunyai Edge),sedangkan

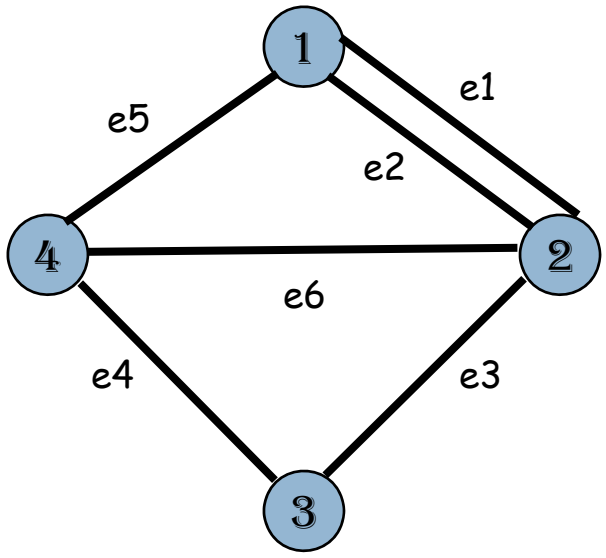
Vertex G dinamakan **Pendant Vertex** (hanya mempunyai satu Edge).

Path (Lintasan)

- Path : Serangkaian vertex dengan edge.
- Simple Path : Path dimana semua Edge dilewati satu kali.
- Closed Walk / Cycle / Circuit : Path yang berawal & berakhir pada vertex yang sama.
- Open Walk : Path yang berawal & berakhir pada vertex yang tidak sama.
- Panjang Path : Jumlah Edge pada Path tersebut.
- Jika ada multiple edge, maka penulisannya adalah penggabungan antara vertex dan edge

Path (Lintasan)

- Contoh Path model :



Example :

➔ 1, e1, 2, 3, 4, 2 adalah simpel Path & Open Walk.
memiliki panjang Path = 4

➔ 1, e2, 2, 3, 4, 1 adalah simpel Path & Closed Walk.
memiliki panjang Path = 4

➔ 1, 4, 3, 2, 4 adalah simpel Path & Open Walk.
memiliki panjang Path = 4

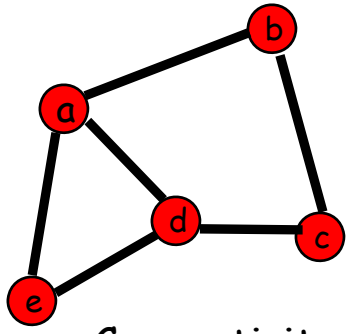
➔ 3 , 2 , 4 , 3 adalah simpel Path & Closed Walk.
memiliki panjang Path = 3

Connectivity

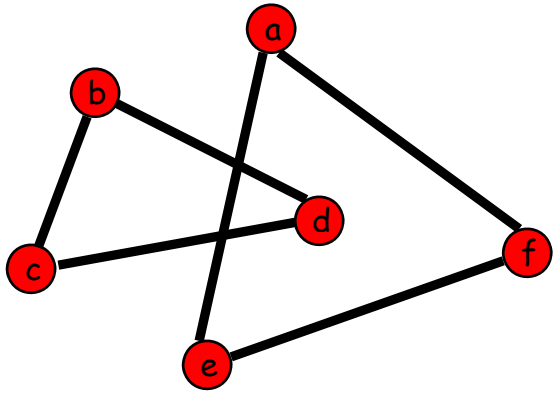
- Undirected Graph

  Pada Undirected Graph dikatakan Connect jika antara setiap pasangan vertex terdapat suatu lintasan (*path*).

Contoh:



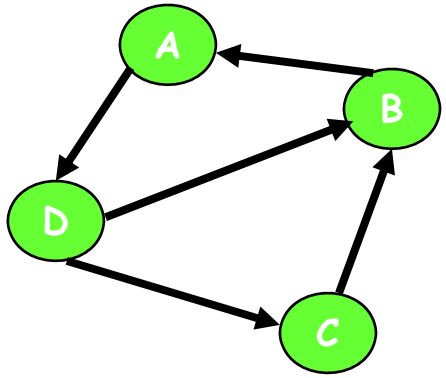
Connectivity



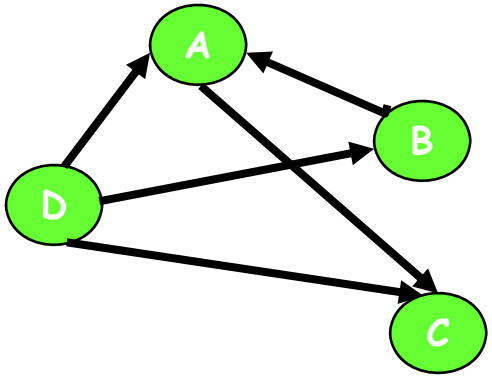
Not Connectivity

Connectivity

- Directed Graph
  - Strong connection : ada lintasan antara vertex i dan vertex j.Sebaliknya ada juga lintasan antara vertex j dan vertex i.
  - Weak connection: bila *underlying* undirected graph-nya merupakan strong connection



Strong connection



Weak connection

Isomorphisme

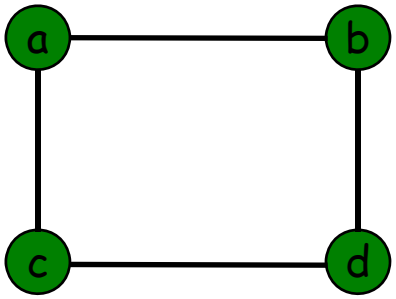
- Syarat Isomorphis :

1. Mempunyai jumlah vertex yang sama.
2. Mempunyai jumlah edge yang sama.
3. Untuk vertex tertentu, jumlah degreenya sama.
4. Dapat merubah dari satu graf ke graf lainnya yang isomorfi secara ilustrasi.
5. Dapat menemukan vertex a1 pada G1 sama dengan vertex a2 pada G2 sehingga matrix adjacent G1 = G2.

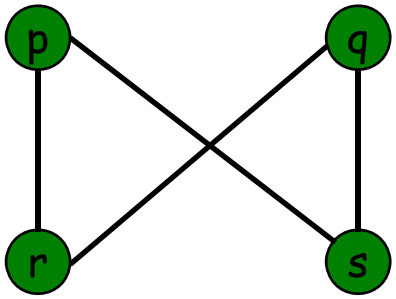
Isomorphisme (Ex1)

- beberapa contoh Isomorphisme :

Graph G = (V1 , E1)



Graph H = (V2 , E2)



p = f (a)

s = f (b)

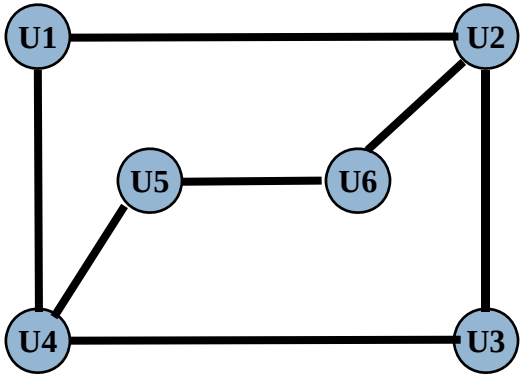
r = f (c)

q = f (d)

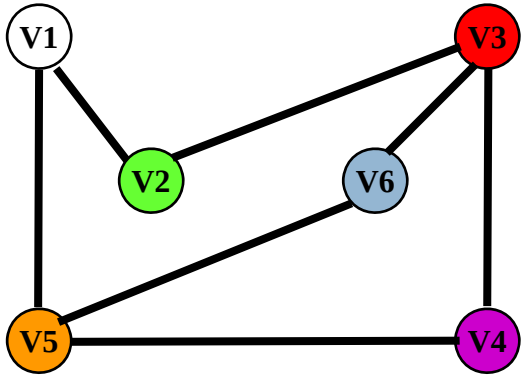
ISOMORPHIS
KAH ??? YA

Isomorphisme (Ex2)

➔



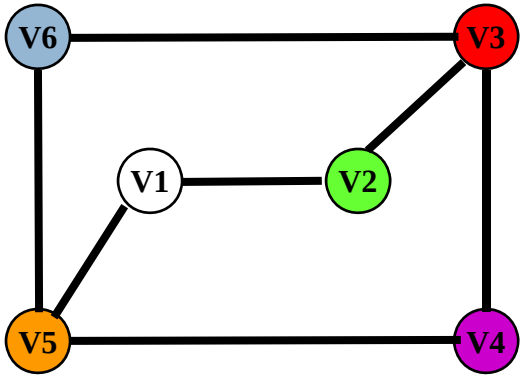
ISOMORPHIS
KAH ???



V 6 = f (U 1)

V 3 = f (U 2)

V 4 = f (U 3)

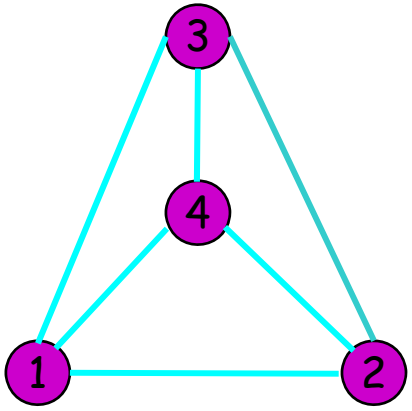


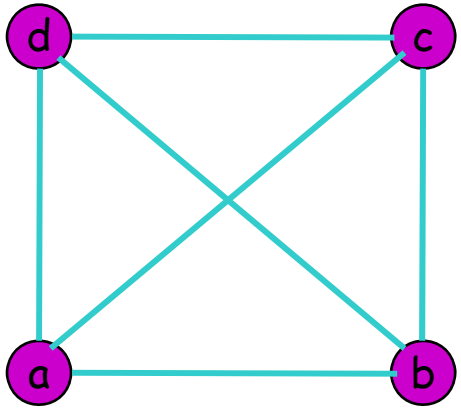
V 5 = f (U 4)

V 1 = f (U 5)

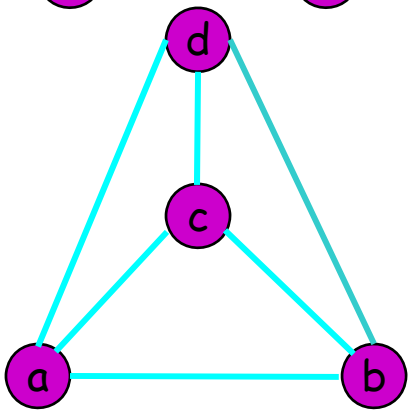
V 2 = f (U 6)

Isomorphisme (Ex3)





ISOMORPHIS
KAH ??? YA



c = f (4)

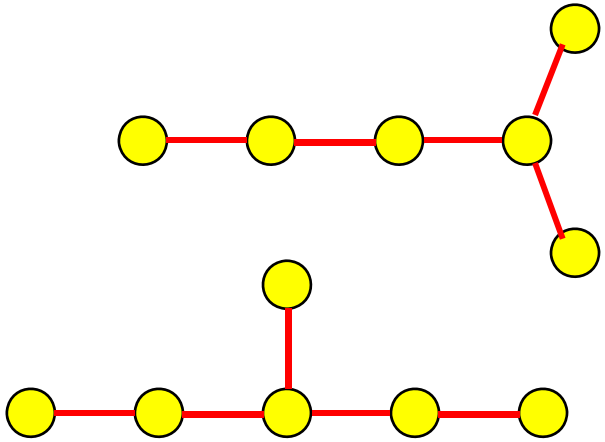
a = f (1)

b = f (2)

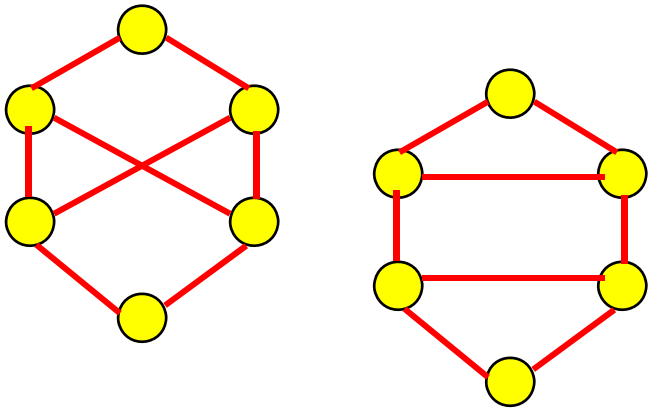
d = f (3)

Isomorphisme (Ex4)

- contoh



Tidak
Isomorphik



Tidak
Isomorphik