

University of Florida  
Dept. of Computer & Information Science & Engineering

**COT 3100**

**Applications of Discrete Structures**

Dr. Michael P. Frank

Slides for a Course Based on the Text  
*Discrete Mathematics & Its Applications*  
(5<sup>th</sup> Edition)  
by Kenneth H. Rosen

# Bài 22: Lý thuyết đồ thị Graph Theory

Rosen 5<sup>th</sup> ed., chs. 8-9  
~44 slides (more later), ~3 lectures

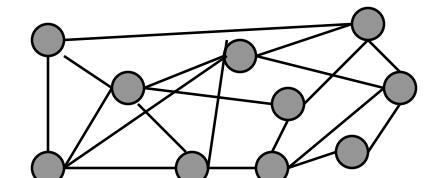
# Đồ thị là gì?

Not



Not Our  
Meaning

- Nghĩa thông thường trong toán học hàng ngày là:  
Vẽ hoặc thể hiện các dữ liệu số sử dụng tọa độ.
- Nghĩa hẹp trong Tóan rời rạc là:  
Một lớp đặc biệt cấu trúc rời rạc (*sẽ định nghĩa*) mà giúp ích thể hiện các quan hệ và có biểu diễn đồ họa rất thuận tiện.



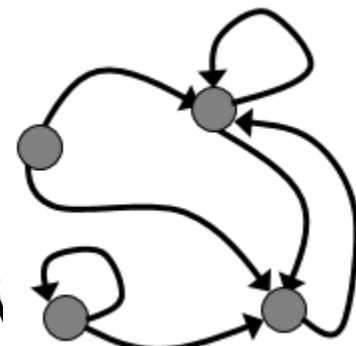
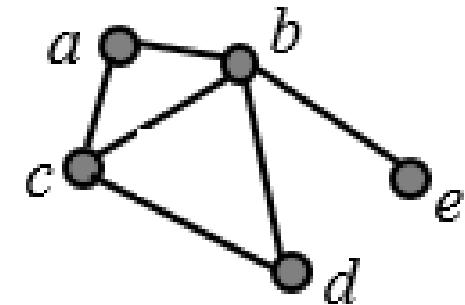
# Ứng dụng của đồ thị

- Hầu như là mọi thứ (đồ thị biểu diễn các quan hệ, quan hệ có thể biểu diễn mở rộng của mọi vị từ).
- Ứng dụng trong mạng, lập lịch, tối ưu luồng, thiết kế mạch, lập đường đi.
- Ứng dụng tiếp theo: phân tích trắc địa, trò chơi điện tử, chương trình dịch, thiết kế hướng đối tượng, ...

# Định nghĩa đồ thị

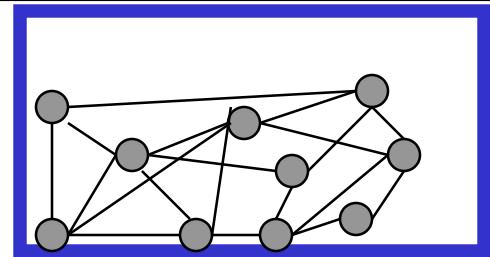
- Đồ thị vô hướng  $G$  là một bộ đôi, ký hiệu  $G=(V,E)$ , trong đó:
  - $V$  là tập các đỉnh (nút)
  - $E$  là tập các cặp không có thứ tự  $\{u_1, u_2\}$ , ở đó  $u_1, u_2 \in V$ .  $E$  được gọi là tập các cạnh.
- Đồ thị có hướng:  $G$  là một bộ đôi, ký hiệu  $G=(V,E)$ , trong đó:
  - $V$  là tập các đỉnh (nút)
  - $E$  là tập các cặp có thứ tự  $(u_1, u_2)$ , ở đó  $u_1, u_2 \in V$  là tập các cung.

Vd:  $E = \{(x,y) \mid x \text{ loves } y\}$



# Đơn đồ thị - Simple Graphs

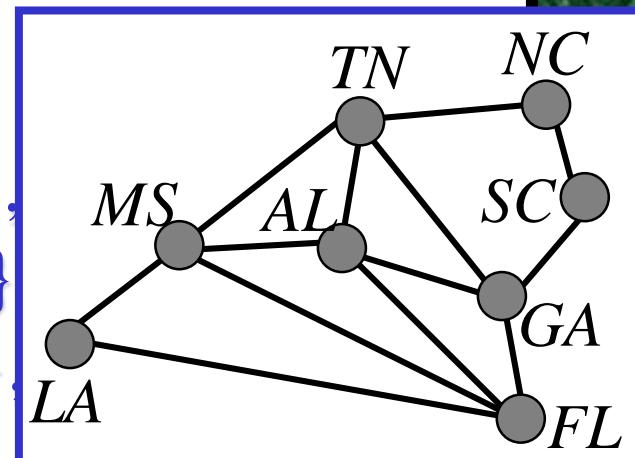
- Đơn đồ thị  $G=(V,E)$  là đồ thị không có khuyên (tức là cạnh nối một đỉnh với chính nó) mà giữa 2 đỉnh bất kỳ có nhiều nhất một cạnh nối



*Visual Representation  
of a Simple Graph*

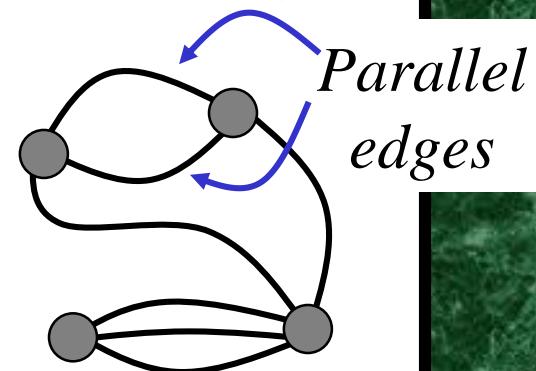
# Example of a *Simple Graph*

- Giả sử  $V$  là tập các bang ở đông nam U.S.:
  - I.e.,  $V=\{\text{FL, GA, AL, MS, LA, SC, TN, NC}\}$
- Giả sử  $E=\{\{u,v\}|u \text{ kề liền với } v\}$ : đường cao tốc
 
$$=\{\{\text{FL,GA}\}, \{\text{FL,AL}\}, \{\text{FL,MS}\}, \{\text{FL,LA}\}, \{\text{GA,AL}\}, \{\text{AL,MS}\}, \{\text{MS,LA}\}, \{\text{GA,SC}\}, \{\text{GA,TN}\}, \{\text{SC,NC}\}, \{\text{NC,TN}\}, \{\text{MS,TN}\}, \{\text{MS,AL}\}\}$$



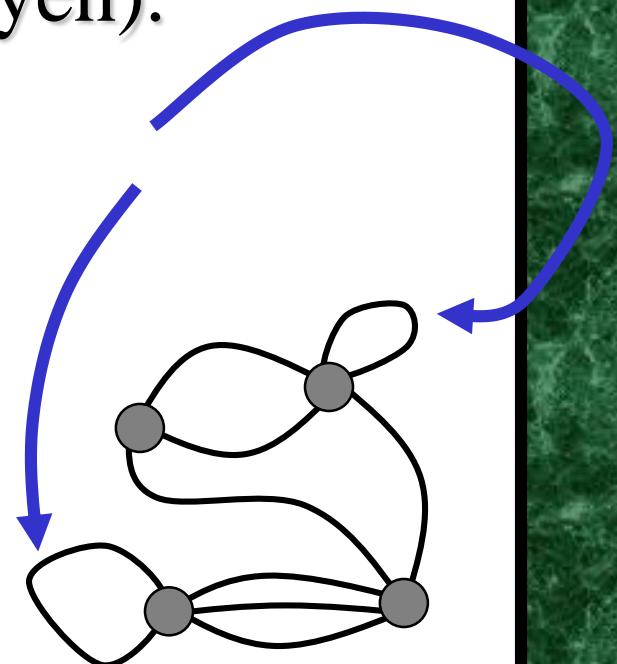
# Đa đồ thị - Multigraphs

- Đa đồ thị vô hướng  $G=(V,E)$  là đồ thị không có khuyên, nhưng có thể có nhiều hơn một cạnh nối hai đỉnh.
- **VD**, đỉnh là các thành phố, cạnh là các đoạn đường quốc lộ.



# Tựa đồ thị - Pseudographs

- Giống như đa đồ thị, nhưng các cạnh có thể nối đỉnh với chính nó (có khuyên).
- VD: đỉnh là các địa điểm cắm trại trong công viên, cạnh là các đường mòn đi bộ.



## §8.2: Các khái niệm về đồ thị

Bạn cần biết các thuật ngữ sau:

- Kè (Adjacent), nối (connects), các đầu mút (endpoints), bậc (degree), điểm đầu (initial), kết thúc (terminal), bậc vào (in-degree), bậc ra (out-degree), đầy đủ (complete), vòng (cycles), bánh xe (wheels),  $n$ -cubes, hai phân (bipartite), đồ thị con (subgraph), hợp (union).

# Kề nhau - Adjacency

G/s  $G$  là đồ thị vô hướng với tập cạnh  $E$ . G/s  $e \in E$  là cặp  $\{u, v\}$ . Khi đó ta nói:

- $u, v$  là kè nhau/ hàng xóm/ nối với nhau.
- Cạnh  $e$  là liên quan với các đỉnh  $u$  và  $v$ .
- Hoặc cạnh  $e$  nối  $u$  và  $v$ .
- Các đỉnh  $u$  và  $v$  là các đầu mút của cạnh  $e$ .

# Bậc của đỉnh - Degree of a Vertex

- G/s  $G$  là đồ thị vô hướng,  $v \in V$  là đỉnh.
- Độ bậc của  $v$ ,  $\deg(v)$ , là số các cạnh liên quan với nó. (Ngoại trừ mỗi khuyên được tính 2 lần)
- Một đỉnh có bậc 0 được gọi là cô lập (*isolated*).
- Một đỉnh có bậc 1 được gọi là móc treo (*pendant*).

# Định lý bắt tay - Handshaking Theorem

- G/s  $G$  là đồ thị vô hướng (simple, multi-, or pseudo-) với tập đỉnh  $V$  và tập cạnh  $E$ . Khi đó

$$\sum_{v \in V} \deg(v) = 2|E|$$

- **Hệ quả:** Mọi đồ thị vô hướng có số chẵn lần các đỉnh bậc lẻ.

# Kè có hướng - Directed Adjacency

- G/s  $G$  là đồ thị có hướng (possibly multi-), và giả sử  $e$  là cạnh của  $G$  mà ánh xạ vào  $(u,v)$ . Khi đó ta nói:
  - $u$  là kè tới  $v$ ,  $v$  là kè từ  $u$
  - $e$  đi từ  $u$ ,  $e$  đi tới  $v$ .
  - $e$  nối  $u$  tới  $v$ ,  $e$  đi từ  $u$  tới  $v$
  - Đỉnh đầu của  $e$  là  $u$
  - Đỉnh đích của  $e$  là  $v$

# Bậc có hướng - Directed Degree

- G/s  $G$  là đồ thị có hướng,  $v$  là đỉnh của  $G$ .
  - Bậc vào của  $v$ ,  $\deg^-(v)$ , là số các cạnh đi đến  $v$ .
  - Bậc ra của  $v$ ,  $\deg^+(v)$ , là số các cạnh đi ra từ  $v$ .
  - Bậc của  $v$ ,  $\deg(v) := \deg^-(v) + \deg^+(v)$ , là tổng của bậc vào và bậc ra của  $v$ .

# Định lý bắt tay có hướng

## Directed Handshaking Theorem

- G/s  $G$  là đồ thị có hướng (possibly multi-) với tập đỉnh  $V$  và tập cạnh  $E$ . Khi đó:

$$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = \frac{1}{2} \sum_{v \in V} \deg(v) = |E|$$

- Lưu ý bậc của một đỉnh không thay đổi khi ta xét cạnh là có hướng hay vô hướng.

# Các cấu trúc đồ thị đặc biệt

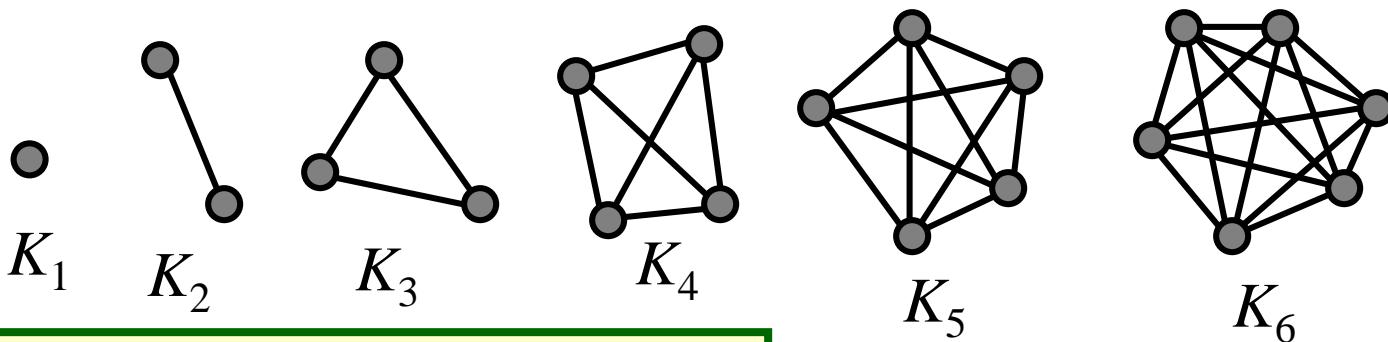
## Special Graph Structures

Các trường hợp đặc biệt của đồ thị vô hướng:

- Đồ thị đầy đủ - Complete graphs  $K_n$
- Vòng - Cycles  $C_n$
- Bánh xe - Wheels  $W_n$
- Siêu khối -  $n$ -Cubes  $Q_n$
- Đồ thị hai phần - Bipartite graphs
- Đồ thị hai phần đầy đủ - Complete bipartite graphs  $K_{m,n}$

# Đồ thị đầy đủ - Complete Graphs

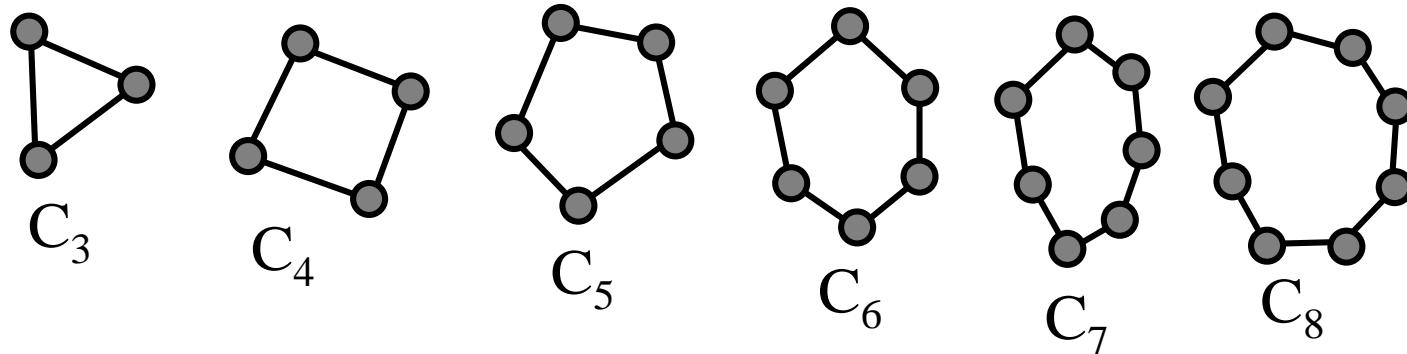
- Với bất kỳ  $n \in \mathbf{N}$ , đồ thị đầy đủ bậc  $n$ ,  $K_n$ , là đơn đồ thị với  $n$  đỉnh trong đó mỗi đỉnh kề với mỗi đỉnh khác:  $\forall u, v \in V:$   
 $u \neq v \Leftrightarrow \{u, v\} \in E.$



Note that  $K_n$  has  $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$  edges.

# Vòng - Cycles

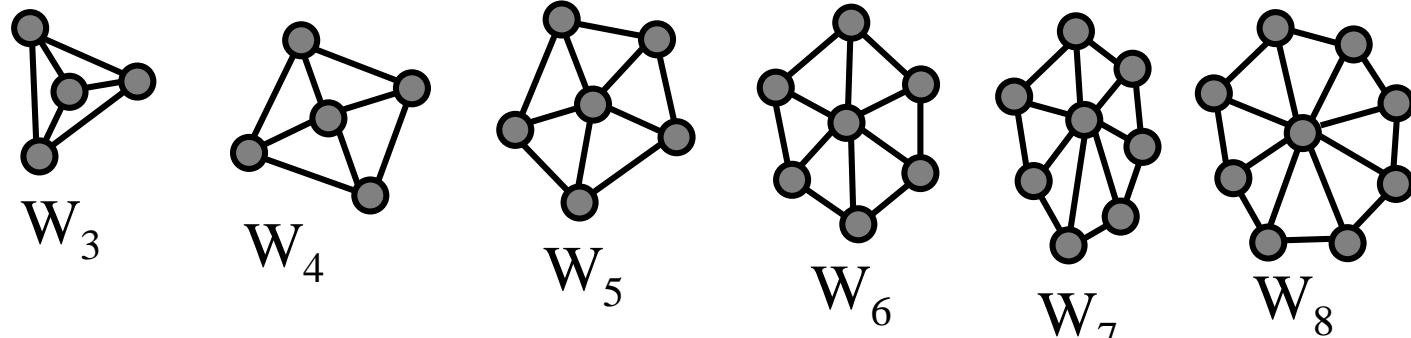
- Với bất kỳ  $n \geq 3$ , vòng trên  $n$  đỉnh,  $C_n$ , là một đơn đồ thị trong đó  $V = \{v_1, v_2, \dots, v_n\}$  và  $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \dots, \{v_{n-1}, v_n\}, \{v_n, v_1\}\}$ .



Có bao nhiêu cạnh trong  $C_n$ ?

# Bánh xe - Wheels

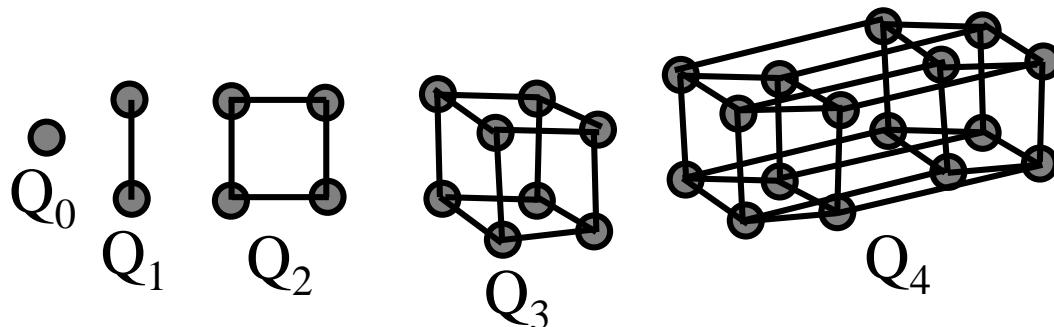
- Với bấy kỵ  $n \geq 3$ , bánh xe  $W_n$ , là đơn đồ thị nhận được từ vòng  $C_n$  bằng cách bổ sung thêm một đỉnh  $v_{\text{hub}}$  và thêm  $n$  cạnh  $\{\{v_{\text{hub}}, v_1\}, \{v_{\text{hub}}, v_2\}, \dots, \{v_{\text{hub}}, v_n\}\}$ .



Có bao nhiêu cạnh trong  $W_n$ ?

# *Siêu khối -- n-cubes (hypercubes)*

- Với bất kỳ  $n \in \mathbb{N}$ , siêu khối  $Q_n$  là đơn đồ thị bao gồm hai bản sao của  $Q_{n-1}$  nối với nhau tại các đỉnh tương ứng.  $Q_0$  có 1 đỉnh.



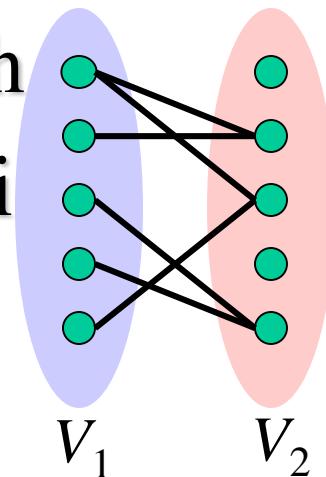
*Số đỉnh:  $2^n$  . Số cạnh:  $n2^{n-1}$*

# Siêu khối -- $n$ -cubes (hypercubes)

- Với bất kỳ  $n \in \mathbf{N}$ , siêu khối  $Q_n$  có thể định nghĩa đê qui như sau:
  - $Q_0 = \{\{v_0\}, \emptyset\}$  (one node and no edges)
  - Với bất kỳ  $n \in \mathbf{N}$ , nếu  $Q_n = (V, E)$ , trong đó  $V = \{v_1, \dots, v_a\}$  và  $E = \{e_1, \dots, e_b\}$ , thì  $Q_{n+1} = (V \cup \{v_1', \dots, v_a'\}, E \cup \{e_1', \dots, e_b'\} \cup \{\{v_1, v_1'\}, \{v_2, v_2'\}, \dots, \{v_a, v_a'\}\})$  trong đó  $v_1', \dots, v_a'$  là các cạnh mới, và trong đó nếu  $e_i = \{v_j, v_k\}$  thì  $e_i' = \{v_j', v_k'\}$ .

# Đồ thị hai phần - Bipartite Graphs

- **Def'n.:** Đồ thị  $G=(V,E)$  là hai phần nếu  $V = V_1 \cup V_2$  trong đó  $V_1 \cap V_2 = \emptyset$  và  $\forall e \in E: \exists v_1 \in V_1, v_2 \in V_2: e = \{v_1, v_2\}$ .
- **Tức là:** Đồ thi có thể chia thành hai phần sao cho mọi cạnh đều đi từ một phần sang phần khác.



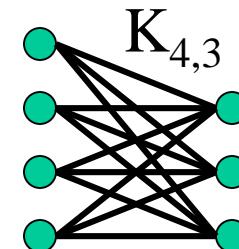
This definition can easily be adapted for the case of multigraphs and directed graphs as well.

Can represent with zero-one matrices.

# Đồ thị hai phần đầy đủ

## Complete Bipartite Graphs

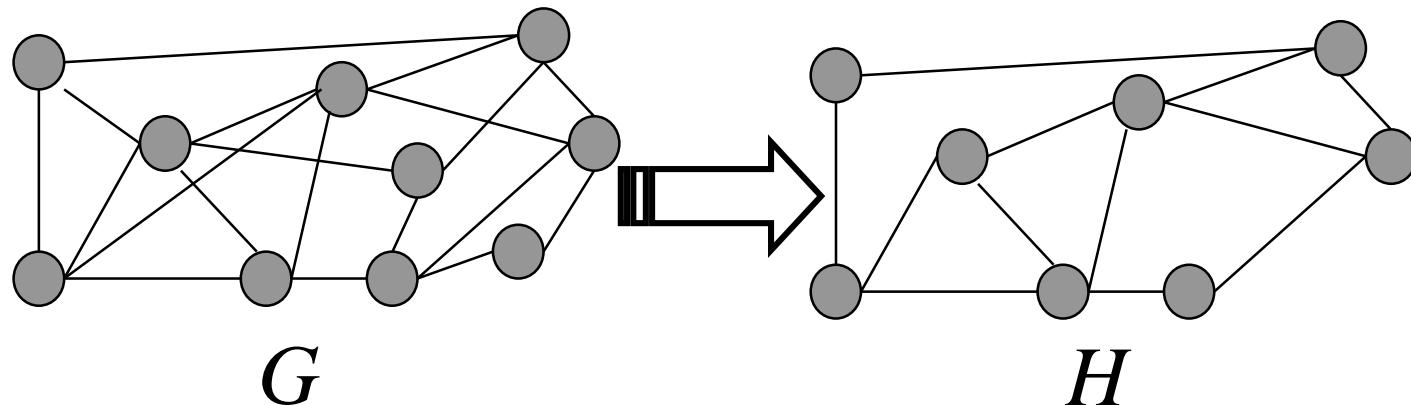
- Với  $m, n \in \mathbb{N}$ , đồ thị hai phần đầy đủ  $K_{m,n}$  là đồ thị trong đó  $|V_1| = m$ ,  $|V_2| = n$ , và  $E = \{\{v_1, v_2\} | v_1 \in V_1 \wedge v_2 \in V_2\}$ .
  - Tức là, có  $m$  đỉnh ở bên trái và  $n$  đỉnh ở bên phải, và mỗi đỉnh ở bên trái được nối với mỗi đỉnh ở bên phải.



$K_{m,n}$  has \_\_\_\_\_ nodes  
and \_\_\_\_\_ edges.

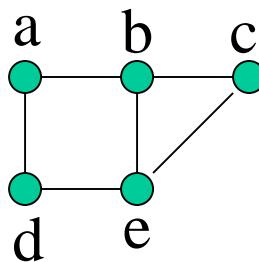
# Đồ thị con - Subgraphs

- Đồ thị con của đồ thị  $G=(V,E)$  là đồ thị  $H=(W,F)$  trong đó  $W \subseteq V$  và  $F \subseteq E$ .

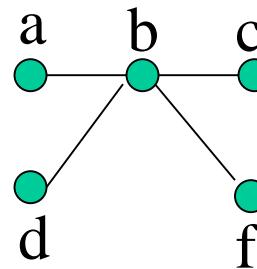


# Hợp đồ thị - Graph Unions

- Hợp của hai đơn đồ thị  $G_1 \cup G_2$  với  $G_1 = (V_1, E_1)$  và  $G_2 = (V_2, E_2)$  là đơn đồ thị  $(V_1 \cup V_2, E_1 \cup E_2)$ .



U

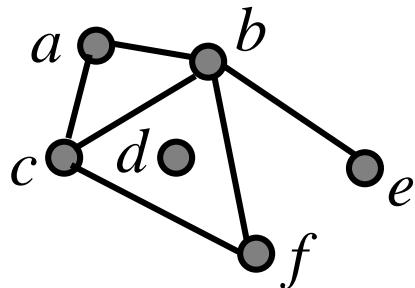


# Biểu diễn đồ thị

- Biểu diễn đồ thị Graph representations:
  - Danh sách liền kề - Adjacency lists.
  - Ma trận liền kề - Adjacency matrices.

# Danh sách liền kề - Adjacency Lists

- Một bảng với mỗi hàng cho một đỉnh và liệt kê các đỉnh kề với nó.



<i>Vertex</i>	<i>Adjacent Vertices</i>
a	b, c
b	a, c, e, f
c	a, b, f
d	
e	b
f	c, b

# Ma trận liền kề - Adjacency Matrices

- Là cách biểu diễn đồ thị đơn
  - Có thể với vòng lặp.
- Ma trận  $\mathbf{A}=[a_{ij}]$ , trong đó  $a_{ij}$  là 1 nếu  $\{v_i, v_j\}$  là cạnh của  $G$ , và 0 nếu ngược lại.
- Có thể mở rộng đến tựa đồ thị (pseudographs) bằng cách cho phép mỗi phần tử của ma trận là số các cạnh (có thể  $>1$ ) giữa các đỉnh.

# Đẳng cấu đồ thị - Graph Isomorphism

- Định nghĩa hình thức:
  - Các đồ thị đơn  $G_1=(V_1, E_1)$  và  $G_2=(V_2, E_2)$  là đẳng cấu nếu  $\exists$  một song ánh  $f:V_1 \rightarrow V_2$  sao cho  $\forall a,b \in V_1$ ,  $a$  và  $b$  là kề nhau trong  $G_1$  khi và chỉ khi  $f(a)$  và  $f(b)$  là kề nhau trong  $G_2$ .
  - $f$  là hàm đặt lại tên giữa hai tập đỉnh để làm cho hai đồ thị là giống nhau hoàn toàn.
  - Định nghĩa này có thể mở rộng ra các kiểu đồ thị khác.

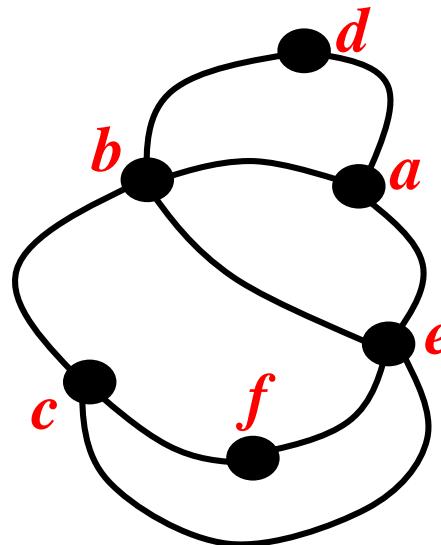
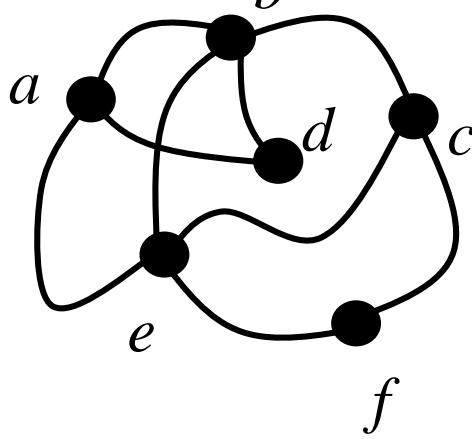
# Các bất biến của đồ thị qua đẳng cấu

*Các điều kiện cần nhưng không đủ để  $G_1=(V_1, E_1)$  là đẳng cấu với  $G_2=(V_2, E_2)$  là:*

- Ta cần phải có  $|V_1|=|V_2|$ , và  $|E_1|=|E_2|$ .
- Số các đỉnh với bậc n nào đó trong hai đồ thị là như nhau.
- Với mỗi đồ thị con thực sự g của một đồ thị này, có một đồ thị con thực sự của đồ thị kia đẳng cấu với g.

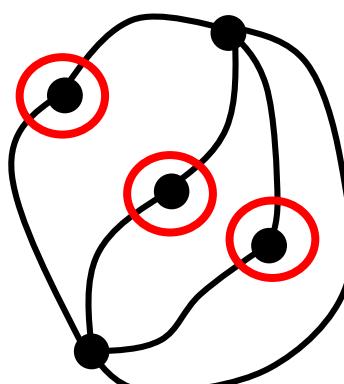
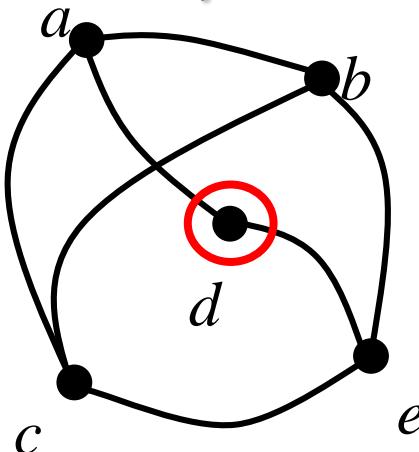
## Ví dụ đẳng cấu – Isomorphism Example

- Nếu đẳng cấu, gán nhãn cho đồ thị thứ hai để chỉ ra đẳng cấu, nếu không chỉ ra sự khác biệt.



# Các đồ thị này có đẳng cấu không? Are These Isomorphic?

- Nếu đẳng cấu, gán nhãn cho đồ thị thứ hai để chỉ ra đẳng cấu, nếu không chỉ ra sự khác biệt.



- *Same # of vertices*
- *Same # of edges*
- *Different # of verts of degree 2!  
(1 vs 3)*

# Đường đi trong đồ thị

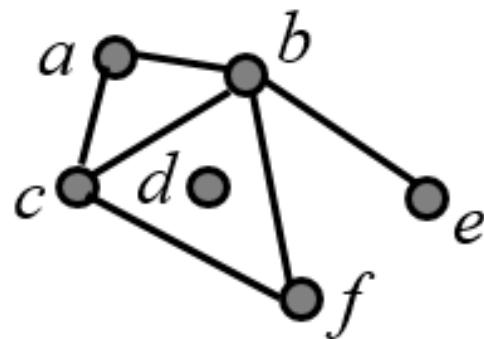
- Trong đồ thị vô hướng, một đường đi độ dài  $n$  từ  $u$  đến  $v$  là một dãy n cạnh kề nhau đi từ đỉnh  $u$  đến đỉnh  $v$ .
  - Đường đi là chu trình nếu  $u=v$ .
  - Đường đi duyệt qua các đỉnh đọc theo nó.
  - Đường đi là *đơn* nếu nó không chứa cạnh nào nhiều hơn một lần.
- Đường đi trong đồ thị có hướng giống như trong đồ thị vô hướng nhưng đường đi phải đi theo hướng của mũi tên.

# Tính liên thông - Connectedness

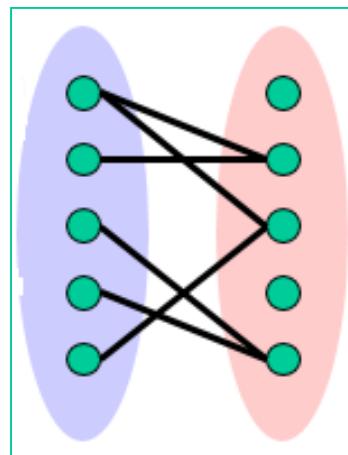
- Đồ thị vô hướng được gọi là liên thông nếu luôn tồn tại đường đi giữa cặp đỉnh khác nhau bất kỳ trong đồ thị.
- **Định lý:** Tồn tại đường đi đơn giữa cặp đỉnh bất kỳ trong đồ thị vô hướng liên thông.
- **Định lý:** Nếu trong một đồ thị vô hướng  $G$  mà tổng bậc hai đỉnh tùy ý đều không nhỏ hơn số đỉnh của đồ thị thì  $G$  liên thông.
- *Đỉnh cắt hoặc cạnh cắt* (cầu) tách một thành phần liên thông thành hai nếu loại bỏ nó ra khỏi đồ thị.

# Ví dụ về đồ thị liên thông

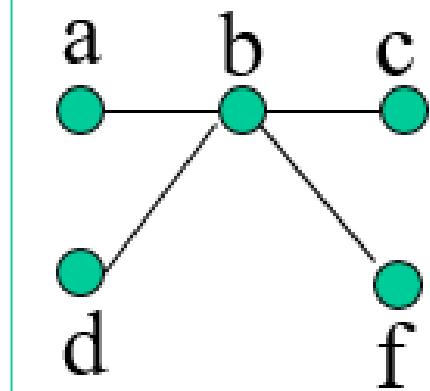
- Đồ thị nào liên thông?



G1



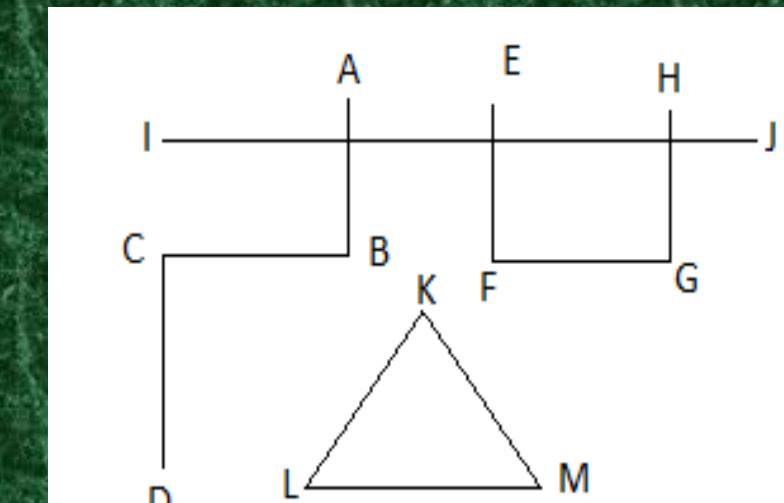
G2



G3

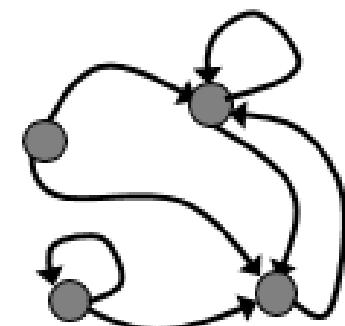
# Ví dụ về thành phần liên thông

- *Một thành phần liên thông (Connected component) của đồ thị  $G$  là một đồ thị con liên thông của  $G$  mà nếu ta thêm bất kỳ một cạnh hoặc một đỉnh vào đồ thị con đó thì nó sẽ không còn liên thông nữa.*
  - *Chú ý: một đồ thị không liên thông có thể có nhiều thành phần liên thông*



# Tính Liên thông trong đồ thị có hướng

- Đồ thị có hướng là *liên thông mạnh* (*strongly connected*) nếu luôn tồn tại đường đi có hướng từ  $a$  đến  $b$  đối với mỗi cặp đỉnh  $a$  và  $b$ .
- Đồ thi là *liên thông yếu* (*weakly connected*) nếu đồ thị vô hướng nền của nó (tức là đồ thị mà bỏ hướng của mỗi cạnh) là liên thông.
- Lưu ý liên thông *mạnh* suy ra liên thông *yếu*, nhưng ngược lại không đúng.

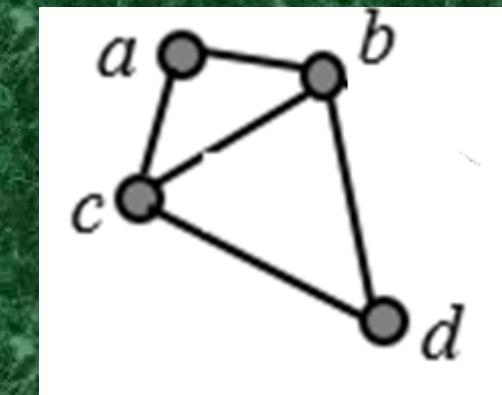


## Đếm đường đi với ma trận liền kề Counting Paths w Adjacency Matrices

- G/s  $\mathbf{A}$  là ma trận liền kề của đồ thị  $G$ .
- Số các đường đi độ dài  $k$  từ  $v_i$  đến  $v_j$  bằng giá trị phần tử tại hàng  $i$  cột  $j$  của ma trận  $\mathbf{A}^k$ .
- *Như vậy để xét tính liên thông của đồ thị có  $n$  đỉnh ta có thể tính đến  $A^{n-1}$ .*

## Ví dụ về số đường đi độ dài k

- Có bao nhiêu đường đi độ dài 3 từ đỉnh a đến đỉnh d; từ đỉnh b đến đỉnh c?

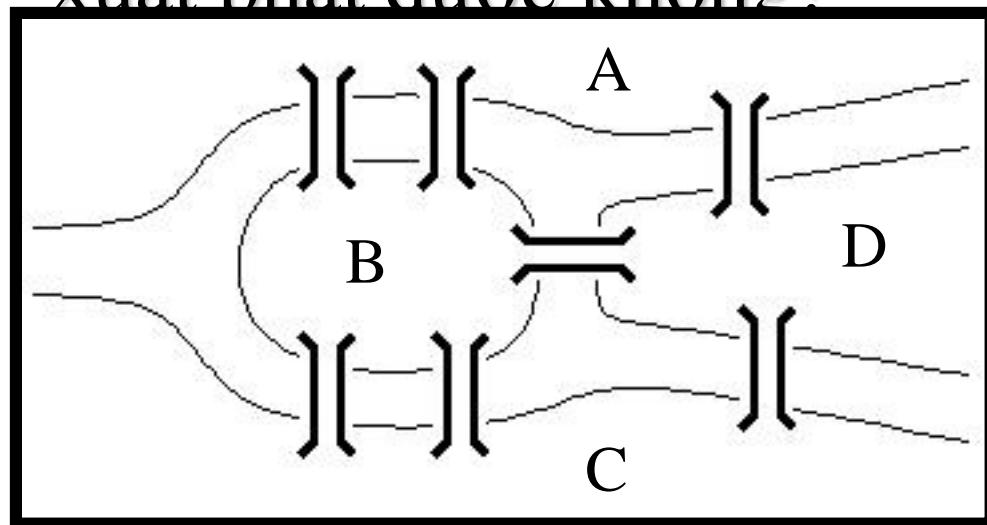


## Đường đi Ole và Hamilton §8.5: Euler & Hamilton Paths

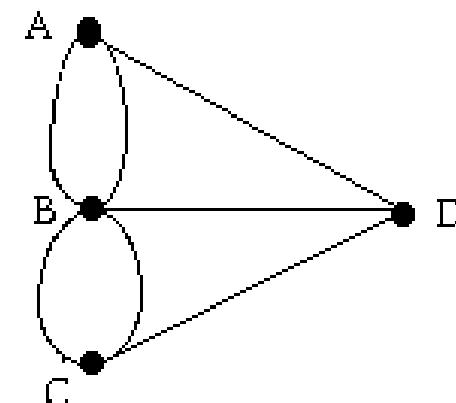
- Chu trình **Euler** trong  $G$  là chu trình đơn chứa mọi cạnh của  $G$ .
- Đường đi **Euler** trong  $G$  là đường đi đơn chứa mọi cạnh của  $G$ .
- Chu trình **Hamilton** là chu trình duyệt qua mỗi đỉnh của  $G$  đúng một lần.
- Đường đi **Hamilton** là đường đi duyệt qua mỗi đỉnh của  $G$  đúng một lần.

# Các cầu trong bài toán Königsberg Bridges of Königsberg Problem

- Chúng ta có thể dạo quanh thành phố qua mỗi cầu đúng một lần rồi lại trở về điểm xuất phát được không?



The original problem



Equivalent multigraph

# Định lý về chu trình và đường đi Euler

- **Theorem:** Đa đồ thị liên thông có chu trình Euler khi và chỉ khi mỗi đỉnh đều có bậc chẵn.
  - **Proof:**
    - ( $\rightarrow$ ) Chu trình bổ sung thêm 2 đến bậc của mỗi đỉnh khi đi qua nó.
    - ( $\leftarrow$ ) Xây dựng theo thuật toán trong sách
- **Theorem:** Đa đồ thị liên thông có đường đi Euler (nhưng không phải chu trình Euler) khi và chỉ khi nó có đúng 2 đỉnh bậc lẻ.
  - Một là đỉnh xuất phát, một là đỉnh kết thúc.

# Thuật toán chu trình Euler

## Euler Circuit Algorithm

- Bắt đầu từ đỉnh bất kỳ.
- Xây dựng đường đi đơn từ nó cho đến khi bạn quay trở lại đỉnh xuất phát.
- Lặp cho đồ thị con còn lại, ghép các kết quả lại cho chu trình gốc.

# Định lý về điều kiện đủ để có chu trình Hamilton

- **Dirac's theorem:** Nếu (nhưng không chỉ nếu)  $G$  là liên thông, đơn, có số đỉnh  $n \geq 3$ , và  $\forall v \deg(v) \geq n/2$ , thì  $G$  có chu trình Hamilton.
  - **Ore's corollary:** Nếu  $G$  là liên thông, đơn, có số đỉnh  $n \geq 3$ , và  $\deg(u) + \deg(v) \geq n$  cho mỗi cặp đỉnh không kề nhau  $u, v$ , thì  $G$  có chu trình Hamilton.

## Bài toán chu trình Hamilton là NP đầy đủ HAM-CIRCUIT is NP-complete

- Bài toán chu trình Hamilton:
  - Cho đồ thị đơn  $G$ , hỏi  $G$  có chứa chu trình Hamilton không?
- Bài toán này được chứng minh là *NP-đầy đủ!*
  - Có nghĩa là, nếu tìm được thuật toán giải bài toán này trong thời gian đa thức, thì nó có thể được sử dụng để giải mọi bài toán NP khác trong thời gian đa thức.

## Đồ thị phẳng

### §8.7: Planar Graphs

- **Định nghĩa 1:** Một đồ thị được gọi là phẳng nếu nó có thể biểu diễn được trên một mặt phẳng mà không có các cạnh nào cắt nhau (tại điểm không phải là điểm đầu mút).

VD: Đồ thị  $K_4$ ,  $Q_3$ ,  $K_5$ ,  $K_{3,3}$  (*bài toán 3 ngôi nhà và 3 thiết bị sinh hoạt*) có là đồ thị phẳng?

**Định nghĩa 2:** Cho  $G$  là một đồ thị phẳng. Mỗi phần mặt phẳng được giới hạn bởi một chu trình đơn không chứa bên trong nó một chu trình đơn khác, gọi là một miền (hữu hạn) của  $G$ . Chu trình giới hạn miền gọi là biên của miền. Mỗi đồ thị phẳng liên thông có một miền vô hạn duy nhất.

# Đồ thị phẳng (tiếp)

- **Định lý Euler:** Cho  $G$  là một đồ thị phẳng liên thông với  $e$  cạnh và  $v$  đỉnh. Gọi  $r$  là số miền trong cách biểu diễn phẳng của  $G$ . Khi đó:

$$r = e - v + 2$$

**Hệ quả 1:** Nếu  $G$  là một đơn đồ thị phẳng liên thông với  $e$  cạnh và  $v$  đỉnh ( $v \geq 3$ ),  $G$  có chu trình đơn độ dài 3, khi đó:  $e \leq 3v - 6$

**Hệ quả 2:** Nếu một đơn đồ thị phẳng liên thông có  $e$  cạnh,  $v$  đỉnh ( $v \geq 3$ ), và không có chu trình đơn độ dài 3, khi đó :  $e \leq 2v - 4$

# Bài toán đường đi ngắn nhất

## §8.6: Shortest-Path Problems

Bài toán: Hãy tìm đường đi ngắn nhất từ đỉnh a đến đỉnh z trong đơn đồ thị vô hướng, liên thông, có trọng số G.

Thuật toán Dijkstra:

```
Procedure Dijkstra(G có các đỉnh a, v1, v2, ..., vn-1, z)
{for i=1 to n do {L(vi)=∞; Truoc(vi)=a}
L(a) = 0; Truoc(a)=a; S = φ;
While z chưa thuộc S do
{u = đỉnh chưa thuộc S có nhãn L(u) nhỏ nhất;
S = S U {u};
for tất cả các đỉnh v chưa thuộc S
  if L(u)+w(u,v)<L(v) then {L(v) = L(u)+w(u,v); Truoc(v)=u}
}
Return L(z);
Return a->Truoc(vx)->...->Truoc(vy)->Truoc(z)->z
}
```

# Giới thiệu về cây

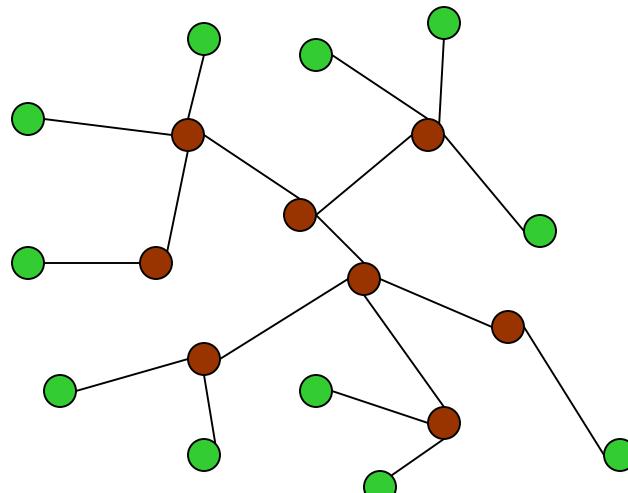
## §9.1: Introduction to Trees

- Cây là đồ thị vô hướng liên thông mà không có chu trình đơn.
  - **Theorem:** Có đường đi đơn duy nhất giữa hai đỉnh bất kỳ.
- Đồ thị vô hướng (không nhất thiết là liên thông) không có chu trình đơn được gọi là *rừng*.
  - Bạn có thể nghĩ như là một tập các cây có các tập đỉnh rời nhau.
- Lá trong cây hoặc rừng là mốc bất kỳ (có bậc 1) hoặc đỉnh cô lập. *Đỉnh trong* là đỉnh không phải là lá (vậy có bậc  $\geq 1$ ).

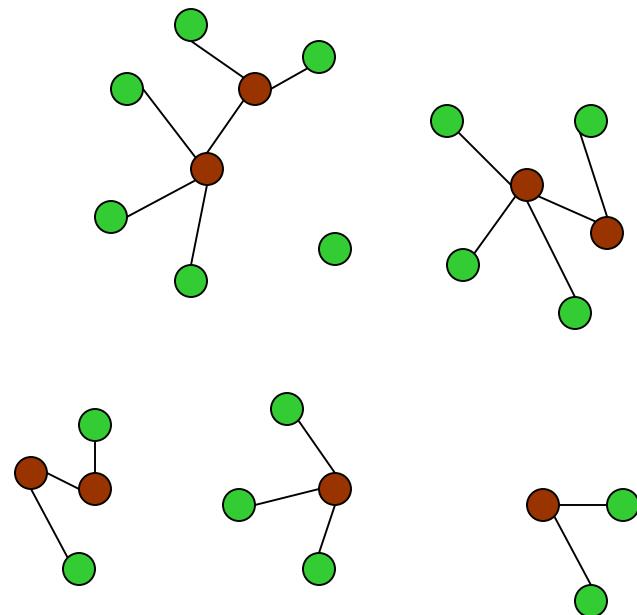
# Tree and Forest Examples

Leaves in green, internal nodes in brown.

- A Tree:



- A Forest:

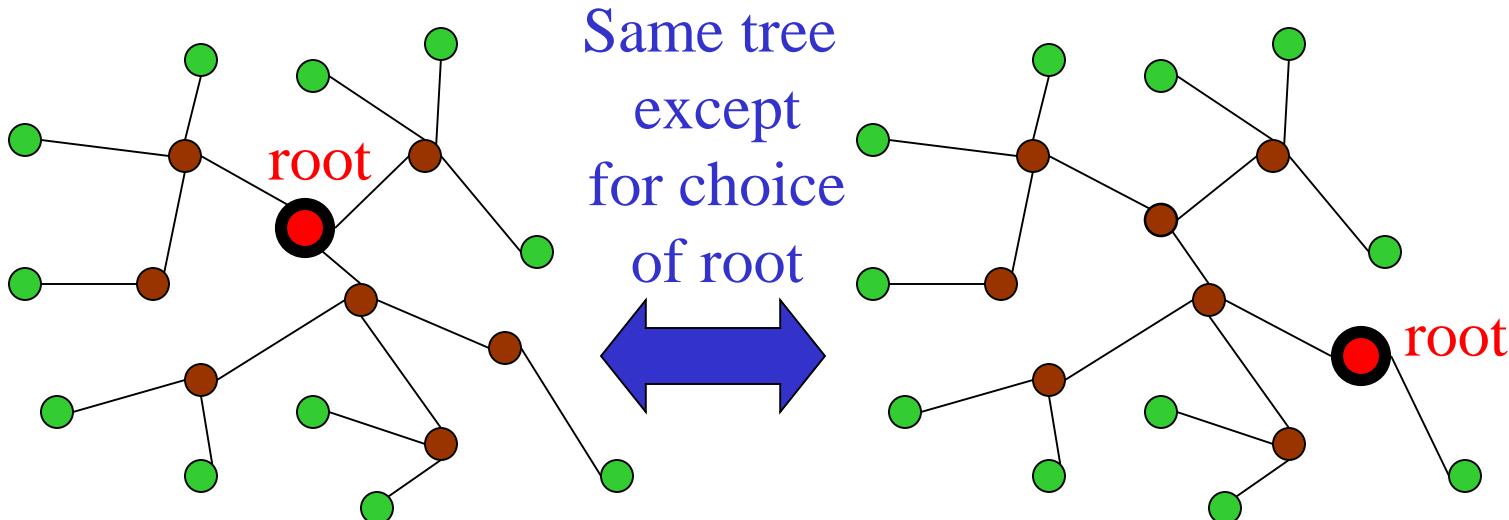


# Cây có gốc - Rooted Trees

- Cây có gốc là cây mà ở đó có một đỉnh được chỉ định là *gốc* (*root*).
  - Mỗi cạnh đi ra (chỉ rõ hoặc ẩn) từ gốc.
- Bạn phải biết các thuật ngữ sau về cây có gốc:
  - Cha (Parent), con (child), anh em (siblings), tổ tiên (ancestors), hậu duệ (descendents), lá (leaf), đỉnh trong (internal node), cây con (subtree).

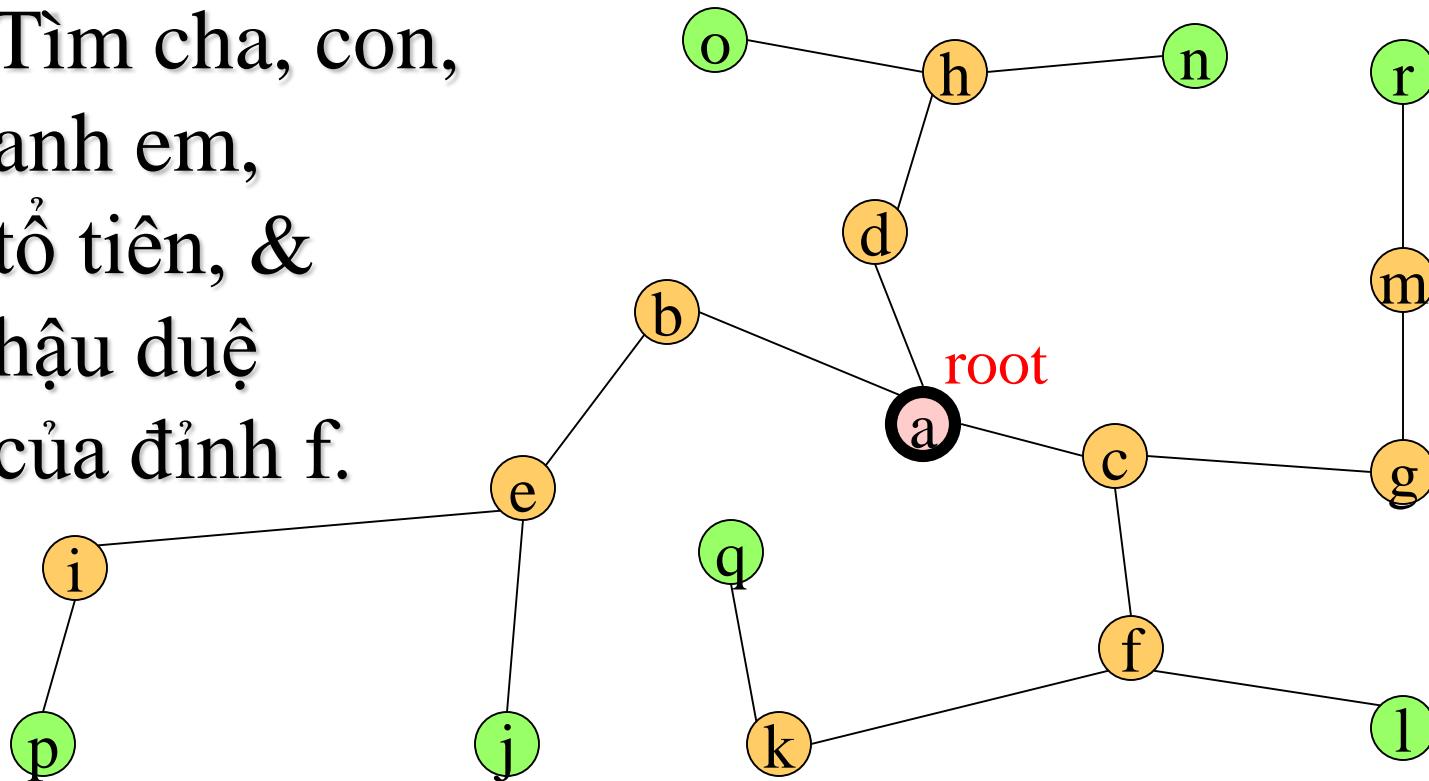
# Rooted Tree Examples

- Lưu ý, cây không gốc với  $n$  đỉnh cho trước tạo nên  $n$  cây con có gốc khác nhau.



# Bài tập về các thuật ngữ của cây có gốc Rooted-Tree Terminology Exercise

- Tìm cha, con, anh em, tổ tiên, & hậu duệ của đỉnh f.

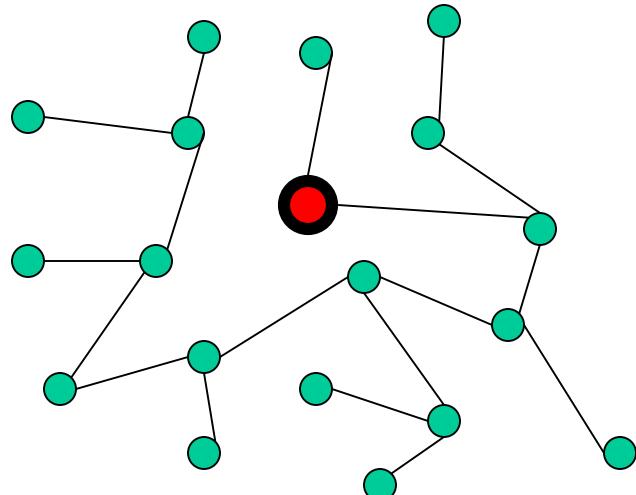
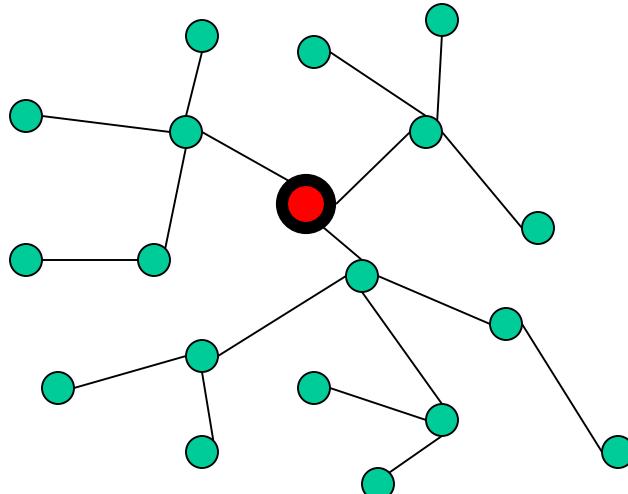


# Cây n-phân -- $n$ -ary trees

- Một cây có gốc được gọi là n-phân nếu mỗi đỉnh của nó có không nhiều hơn  $n$  con.
  - Nó được gọi là đầy đủ nếu mỗi đỉnh trong (không là lá) có đúng  $n$  con.
- Cây 2-phân được gọi là cây nhị phân (*binary tree*).
  - Được sử dụng để mô tả dãy các quyết định có/không.
    - Example: Các so sánh trong thuật toán tìm kiếm nhị phân.

# Cây nào là cây nhị phân Which Tree is Binary?

- **Theorem:** Cây có gốc là nhị phân khi và chỉ khi mỗi đỉnh khác gốc có bậc  $\leq \underline{\hspace{2cm}}$ , và gốc có bậc  $\leq \underline{\hspace{2cm}}$ .



## Cây có gốc có thứ tự Ordered Rooted Tree

- Đây là cây có gốc mà ở đó các con của mỗi đỉnh trong đều được sắp xếp thứ tự.
- Trong cây nhị phân có thứ tự, ta có thể định nghĩa:
  - Con trái, con phải
  - Cây con trái, cây con phải
- Đối với cây  $n$ -phân với  $n > 2$ , có thể sử dụng thuật ngữ “trái nhất”, “phải nhất”.

# Một số Định lý về cây

## Some Tree Theorems

- Mọi cây có  $n$  đỉnh có  $e = n-1$  cạnh.
  - **Proof:** Xét việc tách lá.
- Cây  $m$ -phân đầy đủ với  $i$  đỉnh trong có  $n=mi+1$  đỉnh, và có  $\ell=(m-1)i+1$  lá.
  - **Proof:** có  $mi$  con của đỉnh trong, cộng với gốc. Và có số lá là,  $\ell = n-i = (m-1)i+1$ . □
- Như vậy, khi  $m$  đã biết và cây là đầy đủ, ta có thể tính tất cả 4 giá trị  $e$ ,  $i$ ,  $n$ , và  $\ell$ , nếu cho trước bất kỳ 1 trong số đó.

# Thêm một số Định lý về cây

## Some More Tree Theorems

- **Definition:** Mức (*level*) của đỉnh là độ dài của đường đi đơn từ gốc đến nó.
  - Chiều cao của cây (*The height of a tree*) là mức đỉnh lớn nhất trong cây.
  - Cây  $m$ -phân với chiều cao  $h$  được gọi là cân đối (*balanced*) nếu mọi lá ở mức  $h$  hoặc  $h-1$ .
- **Theorem:** Có nhiều nhất  $m^h$  lá trong cây  $m$ -phân có chiều cao  $h$ .
  - **Corollary:** Cây  $m$ -phân với  $\ell$  lá có chiều cao  $h \geq \lceil \log_m \ell \rceil$ . Nếu là  $m$  đầy đủ và cân đối thì  $h = \lceil \log_m \ell \rceil$ .

## Ứng dụng của cây

### §9.2: Applications of Trees

- Cây tìm kiếm nhị phân (Binary search trees)
  - Cấu trúc dữ liệu đơn giản cho các danh sách sắp xếp
- Cây quyết định (Decision trees)
  - So sánh tối thiểu trong các thuật toán sắp xếp.
- Mã tiền tố - Prefix codes
  - Huffman coding
- Cây trò chơi - Game trees

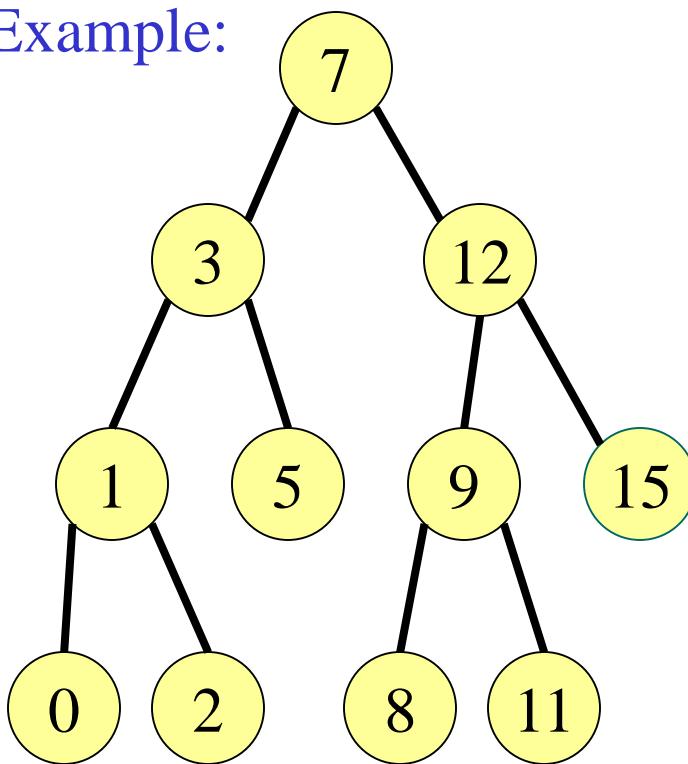
# Cây tìm kiếm nhị phân Binary Search Trees

- Thể hiện tập các đồ vật được sắp xếp.
  - Hỗ trợ các phép toán sau trong thời gian trung bình là  $\Theta(\log n)$ :
    - Tìm kiếm đồ vật đã tồn tại.
    - Chèn đồ vật mới, nếu nó chưa có.
  - Hỗ trợ in mọi đối tượng trong thời gian  $\Theta(n)$ .
- Lưu ý chèn vào danh sách bình thường  $a_i$  mất thời gian xấu nhất là  $\Theta(n)$ .

# Định dạng cây tìm kiếm nhị phân Binary Search Tree Format

- Các đồ vật được cất tại các đỉnh riêng rẽ.
- Ta cần bố trí lại cây để thỏa bất biến sau:
  - Với mỗi đồ vật  $x$ ,
    - Mỗi đỉnh bên cây trái của  $x$  sẽ nhỏ hơn  $x$ .
    - Mỗi đỉnh bên cây phải của  $x$  sẽ lớn hơn  $x$ .

Example:



# Chèn đệ qui vào cây nhị phân

## Recursive Binary Tree Insert

```
procedure insert( $T$ : binary tree,  $x$ : item)
     $v := \text{root}[T]$ 
    if  $v = \text{null}$  then begin
         $\text{root}[T] := x$ ; return “Done” end
    else if  $v = x$  return “Already present”
    else if  $x < v$  then
        return insert(leftSubtree[ $T$ ],  $x$ )
    else {must be  $x > v$ }
        return insert(rightSubtree[ $T$ ],  $x$ )
```

# Cây quyết định Decision Trees (pp. 646-649)

- Cây quyết định thể hiện quá trình ra quyết định.
  - Mỗi điểm quyết định hoặc tình huống có thể được biểu diễn bởi một đỉnh.
  - Mỗi lựa chọn có thể được thực hiện tại các điểm quyết định được thể hiện bằng cạnh đến đỉnh con của nó.
- Trong cây quyết định mở rộng được sử dụng để phân tích quyết định, ta có đưa vào đỉnh mà thể hiện sự kiện ngẫu nhiên và thông tin ra từ chúng.

# Bài toán cân đồng xu Coin-Weighing Problem

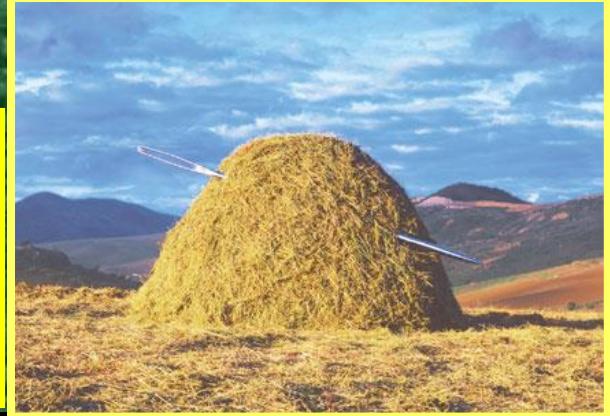
- Tưởng tượng bạn có 8 đồng xu, m trong số đó là đồng giả nhẹ hơn, và một cái cân đối hai đĩa.
  - Không cần quả cân nào cho bài toán này!
- Cân bao nhiêu lần cân để đảm bảo rằng phát hiện ra đồng tiền giả?



## Ứng dụng Định lý về chiều cao của cây Applying the Tree Height Theorem

- Cây quyết định cần có ít nhất 8 đỉnh lá, vì có 8 khả năng kết quả có thể.
  - Để tìm đồng nào trong số đó là giả.
- Dùng Định lý chiều cao của cây,  $h \geq \lceil \log_m \ell \rceil$ .
  - Vậy cây quyết định cần phải có chiều cao  $h \geq \lceil \log_3 8 \rceil = \lceil 1.893\dots \rceil = 2$ .
- Hãy tìm xem ta có thể giải bài toán chỉ cân 2 lần không ...

# Chiến lược giải tổng quát



- Bài toán là ví dụ tìm một đồ vật riêng lẻ duy nhất, trong số danh sách  $n$  đồ vật khác.
  - Tương tự như tìm “cái kim trong đống cỏ khô.”
- Được trang bị cái cân, ta có thể tìm cách giải bài toán sử dụng chiến lược chia để trị, giống như đã làm với tìm kiếm nhị phân.
  - Ta muốn giảm tập các vị trí có thể ở đó đồ vật mong muốn (đồng xu) có thể tìm thấy từ  $n$  xuống 1, trong thời gian logarit.
- Mỗi lần cân có 3 kết quả có thể.
  - Vậy, ta có thể chia không gian tìm kiếm thành 3 phần mà kích thước có thể gần bằng nhau nếu có thể.
- Chiến lược này sẽ dẫn đến số lần cân yêu cầu nhỏ nhất trong t/h xấu nhất.

# Chiến lược cân tổng quát

## General Balance Strategy

- Trong một bước, đặt  $\lceil n/3 \rceil$  của  $n$  đồng xu để tìm ra phần chưa đồng xu giả:
  - Nếu cân nghiêng về trái, thì:
    - Đồng xu giả ở bên phải trong số  $\lceil n/3 \rceil \approx n/3$  đồng xu.
  - Nếu cân nghiêng về phải, thì:
    - Đồng xu giả ở bên trái trong số  $\lceil n/3 \rceil \approx n/3$  đồng xu.
  - Nếu cân cân bằng, thì:
    - Đồng xu giả ở trong tập còn lại ngoài cân gồm  $n - 2\lceil n/3 \rceil \approx n/3$  đồng xu!
- Trừ khi nếu  $n \bmod 3 = 1$  thì ta có thể làm tốt hơn bằng cách cân  $\lfloor n/3 \rfloor$  đồng xu ở mỗi phía.

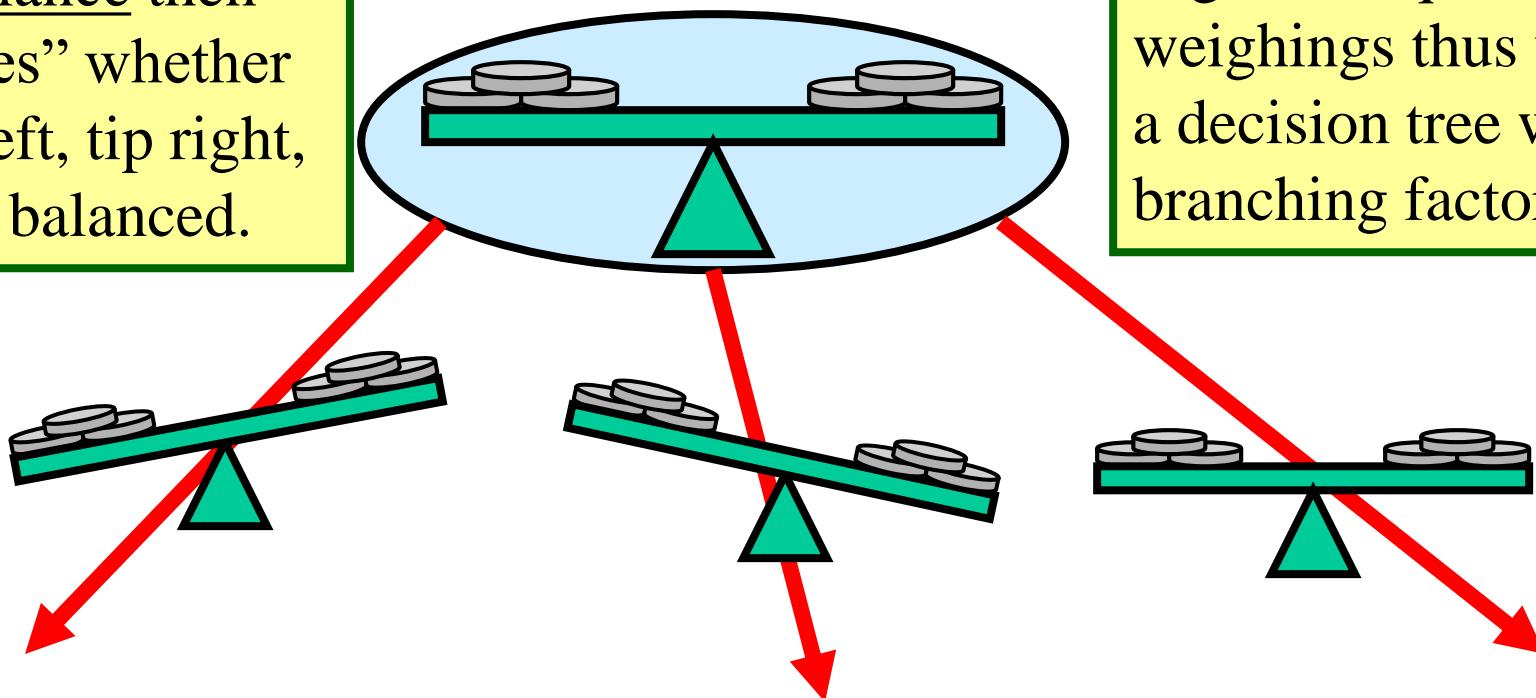
You can prove that this strategy always leads to a balanced 3-ary tree.

# Như bài toán cây quyết định As a Decision-Tree Problem

- Trong mỗi tình huống, ta cần chọn hai tập con đồng xu bằng nhau về kích cỡ và rời nhau để đặt lên cân.

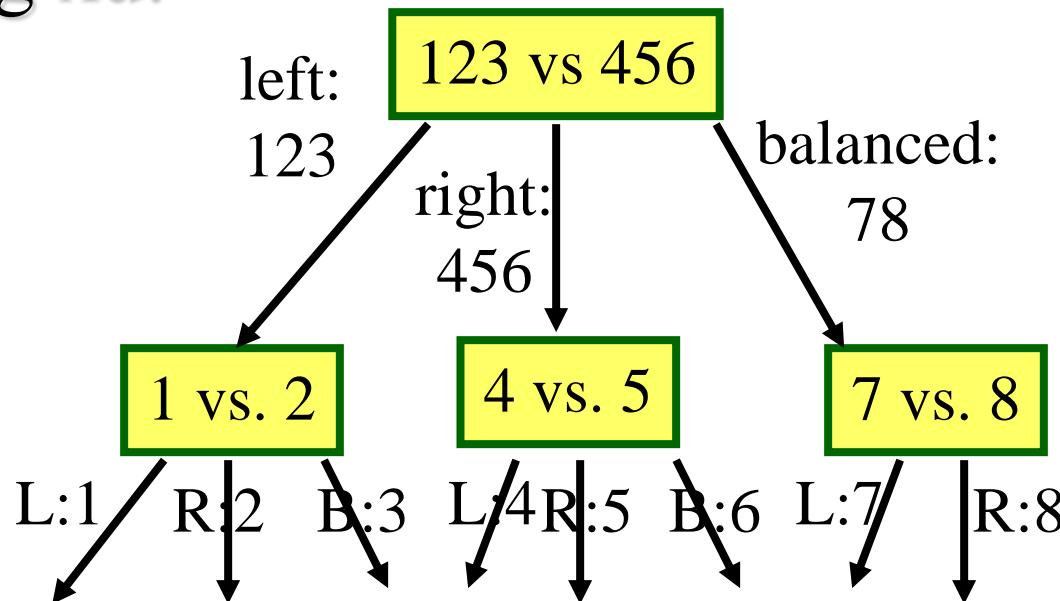
The balance then “decides” whether to tip left, tip right, or stay balanced.

A given sequence of weighings thus yields a decision tree with branching factor 3.



# Cây quyết định cân đồng xu Coin Balancing Decision Tree

- Đây là cây biểu diễn cho t/h bài toán cân đồng xu:



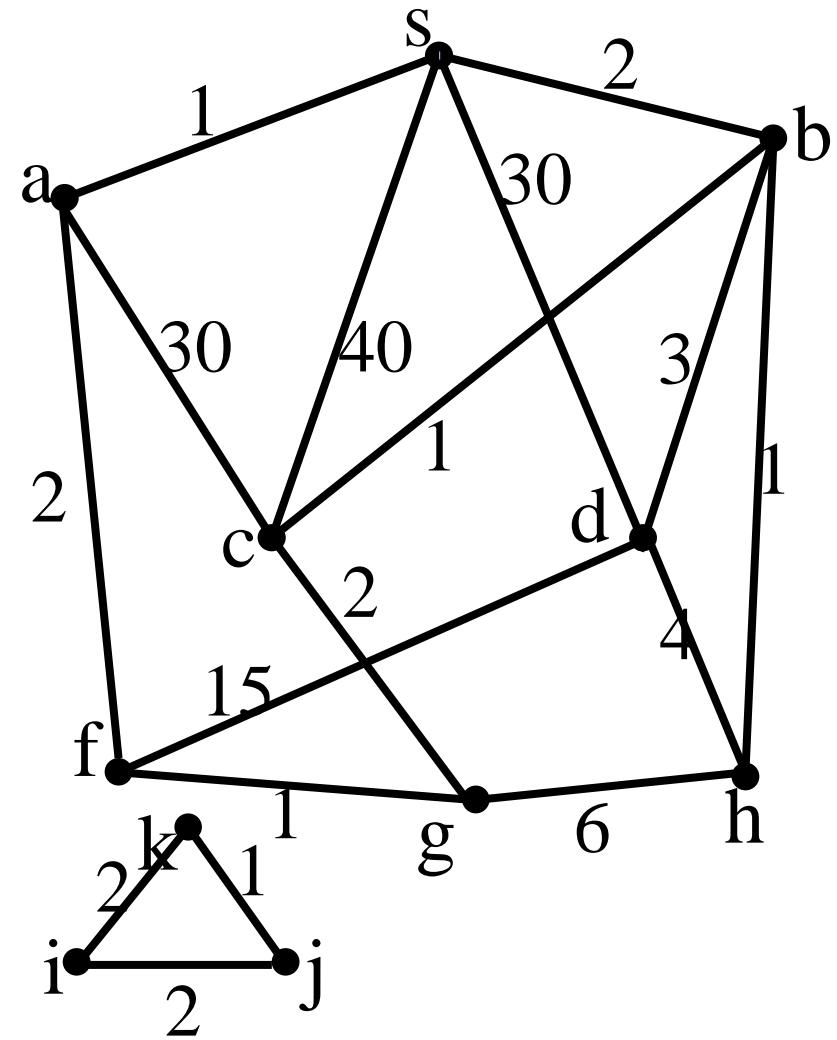
## §9.3: Duyệt cây - Tree Traversal

- Các thuật toán duyệt
  - Duyệt theo chiều sâu - Depth-first traversal:
  - Duyệt theo chiều rộng - Breadth-first traversal
    - Tiền thứ tự - Preorder traversal
    - Trung thứ tự - Inorder traversal
    - Hậu thứ tự - Postorder traversal

## §9.4: Cây bao trùm - Spanning Trees

- Cho một đồ thị vô hướng  $G$   
Cây bao trùm của  $G$  là tập các cạnh mà thỏa mãn:
  - Là Cây (không chu trình, không gốc)
  - Đi qua mọi đỉnh (bao trùm)
- Tìm cây khung của đồ thị: Tìm kiếm theo chiều sâu, Tìm kiếm theo chiều rộng

# Cây bao trùm nhỏ nhất



**Định nghĩa:** Cây bao trùm nhỏ nhất của một đồ thị vô hướng có trọng số là cây bao trùm mà có tổng các trọng số của các cạnh trên cây là nhỏ nhất.

**Input:** Một đồ thị vô hướng có trọng số.

**Output:** Cây bao trùm có tổng trọng số các cạnh trên cây nhỏ nhất

# Thuật toán Kruskal

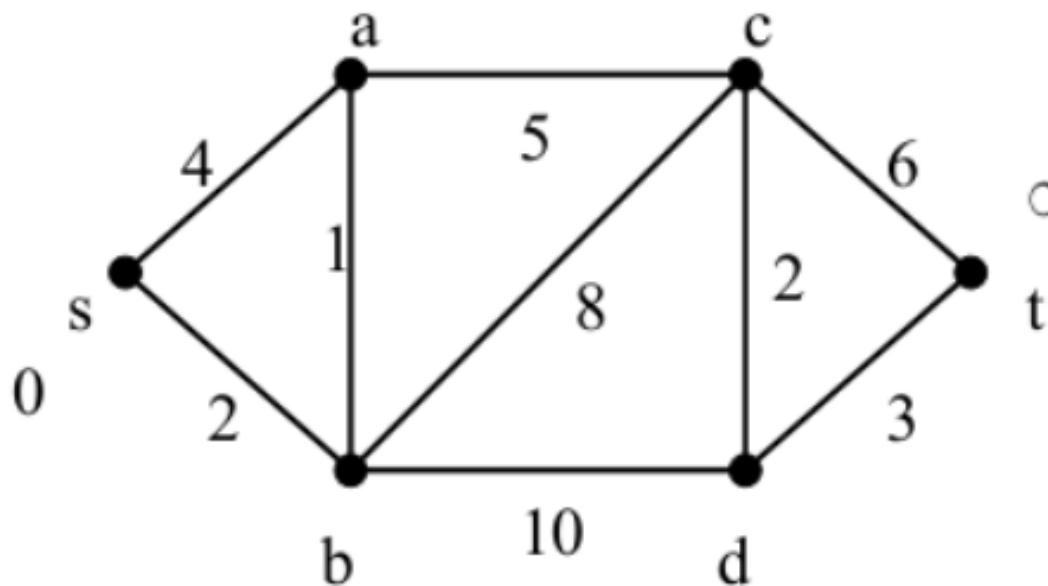
- Sắp xếp các cạnh theo độ dài tăng dần
- Xuất phát từ tập rỗng cạnh T của cây
- Chừng nào chưa chọn đủ số cạnh thì lặp lại việc sau:
  - chọn cạnh có độ dài nhỏ nhất đưa vào tập cạnh T của cây, cập nhật thành phần của các đầu mút.
  - Mỗi bước lặp, chọn cạnh chưa xét có độ dài nhỏ nhất:
    - Nếu 2 đầu mút thuộc 2 thành phần khác nhau, bổ sung cạnh, hợp 2 thành phần.
    - Nếu 2 đầu mút thuộc cùng 1 thành phần, bỏ qua xét cạnh có độ dài nhỏ tiếp theo.

=> Tập cạnh T tìm được là cây khung nhỏ nhất

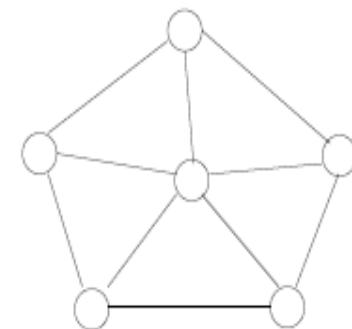
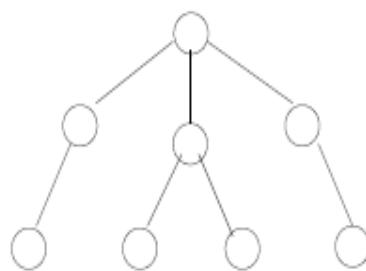
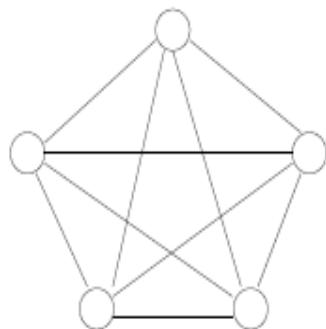
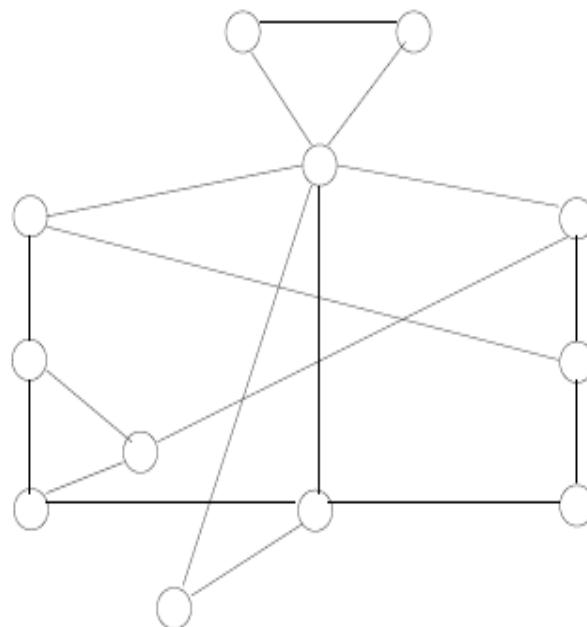
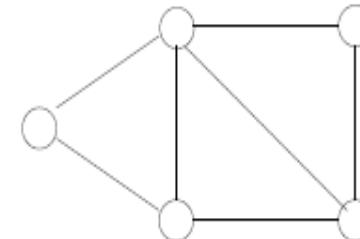
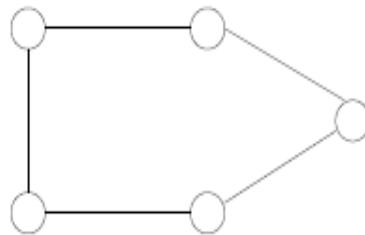
# Thuật toán Prim

- Input: Đồ thị G.
- Output: Cây khung H (khởi tạo  $H = \emptyset$ ).
  - Bước 1: Chọn tùy ý 1 đỉnh của G đặt vào H.
  - Bước 2: Nếu mọi đỉnh của G đều nằm trong H thì dừng.
  - Bước 3: Tìm 1 đỉnh của G mà không nằm trong H và có nối với 1 đỉnh trong H bằng một cạnh có trọng số nhỏ nhất. Thêm đỉnh này và cạnh tương ứng vào H. Quay lại bước 2.

# Ví dụ



**Problem 1.** Color the following graphs with the minimum possible colors, such no two adjacent vertices have the same color.



M

