BÀI 7

CÂY

Vũ Thương Huyền
huyenvt@tlu.edu.vn

NỘI DUNG

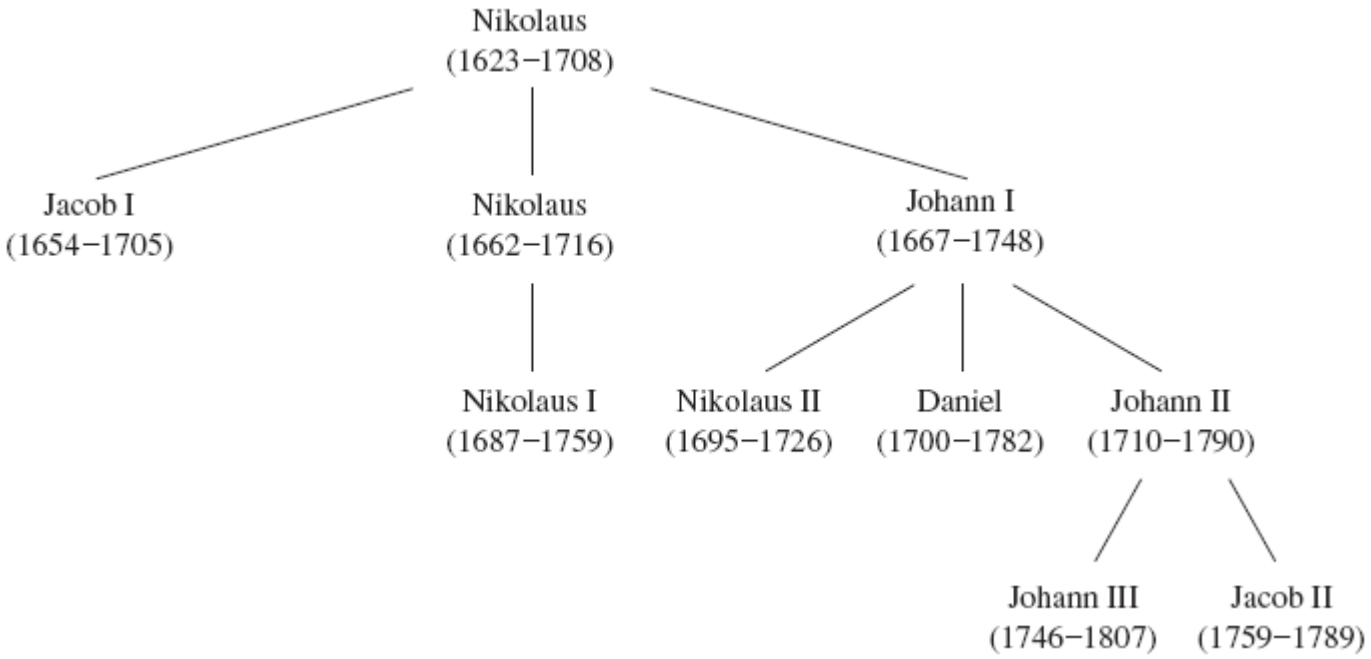
- **Các định nghĩa và tính chất**
- **Các ứng dụng của cây**
- **Cây khung**
- **Cây khung nhỏ nhất**

9.1 CÁC ĐỊNH NGHĨA VÀ TÍNH CHẤT CÂY

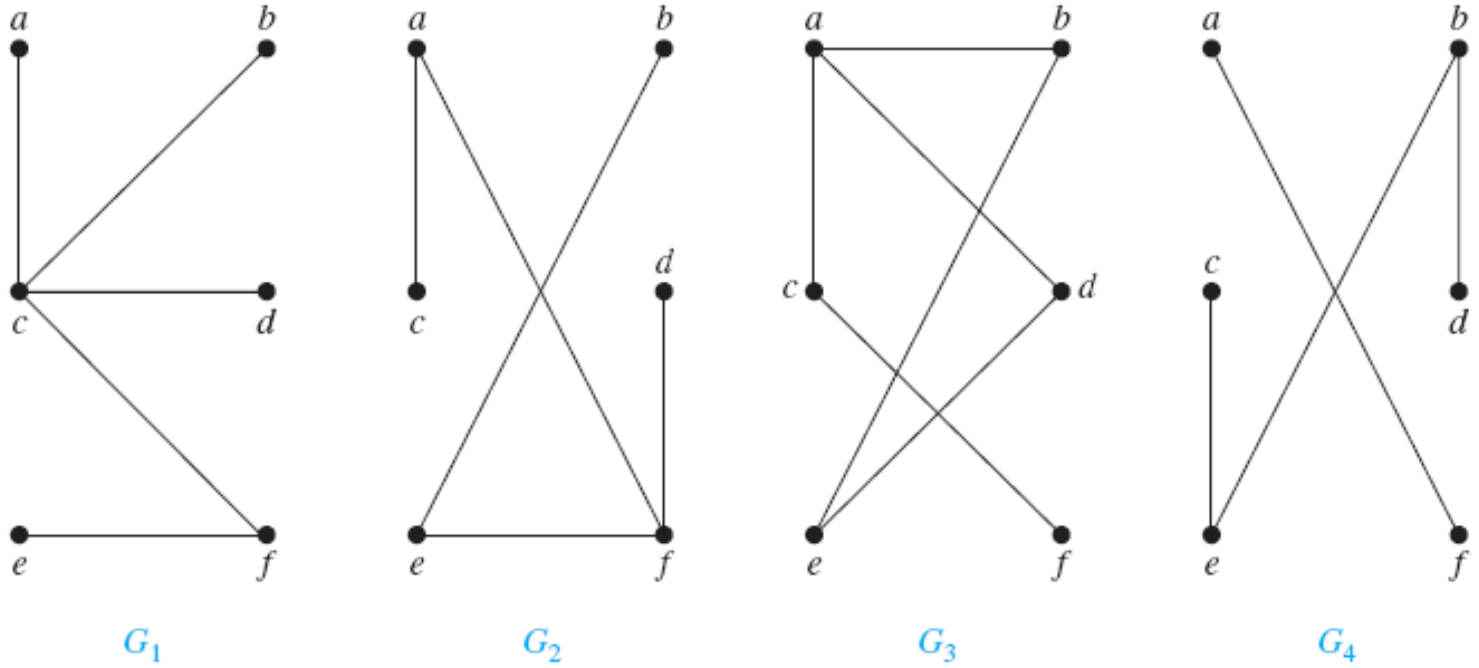
Định nghĩa 1:

Cây là một đồ thị vô hướng, liên thông và không có chu trình đơn

Ví dụ:



Ví dụ: Đồ thị nào sau đây là cây?



Đồ thị không có chu trình đơn và không liên thông gọi là **rừng.**

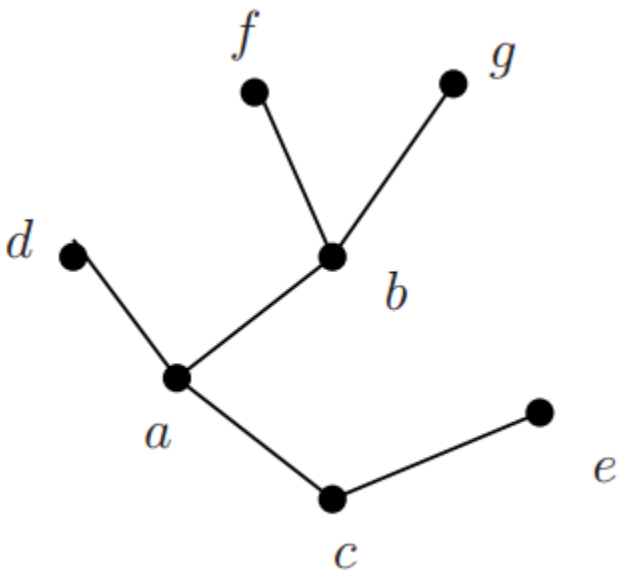
Định lí 1:

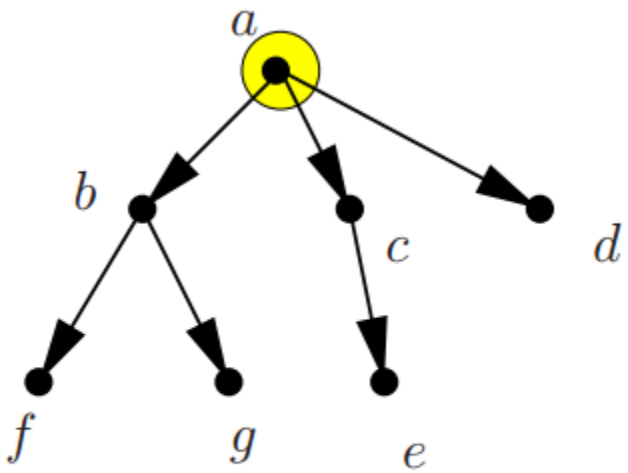
Một đồ thị vô hướng **là một cây** nếu giữa mọi cặp đỉnh của nó luôn tồn tại **đường đi đơn duy nhất**.

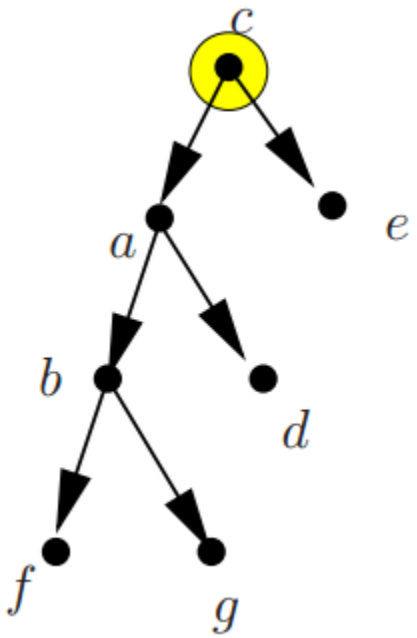
Định nghĩa 2:

Cây có gốc là cây có một đỉnh được gọi là gốc và mọi cạnh có hướng từ gốc đi ra.

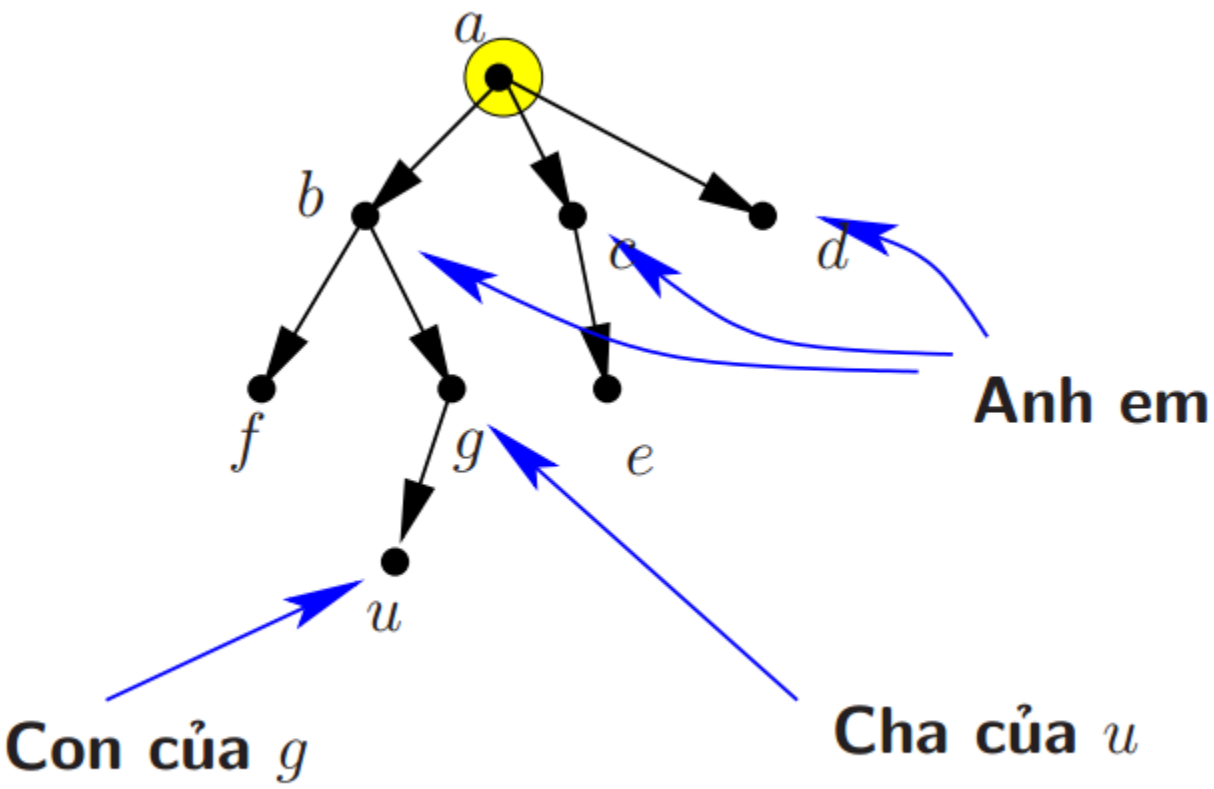
Ví dụ:



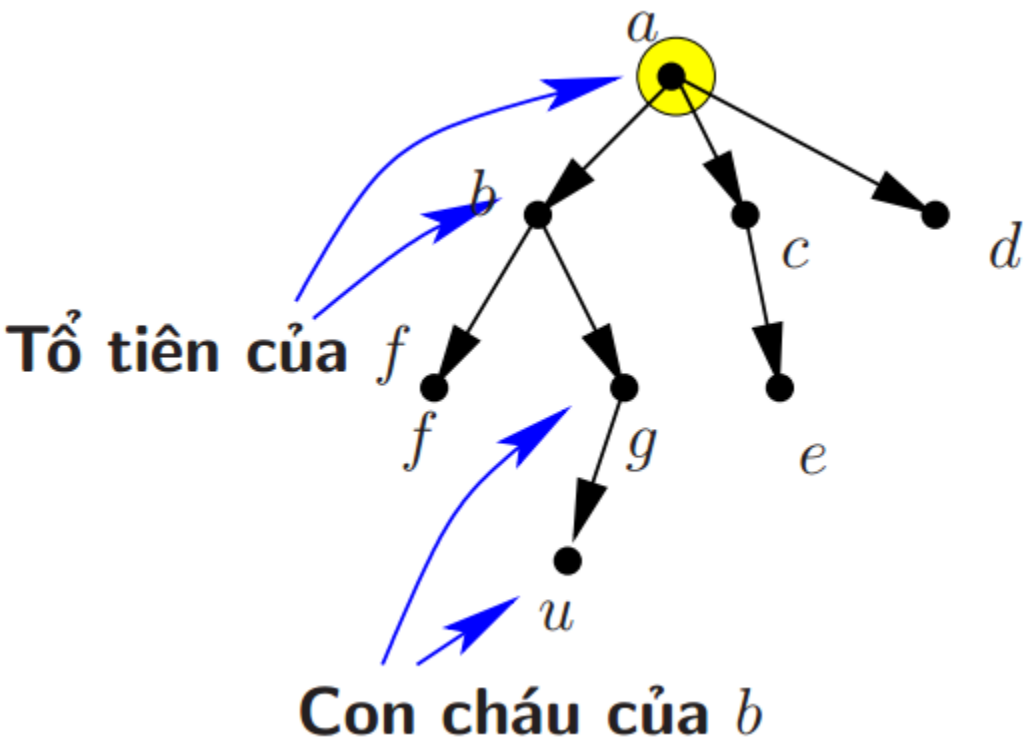




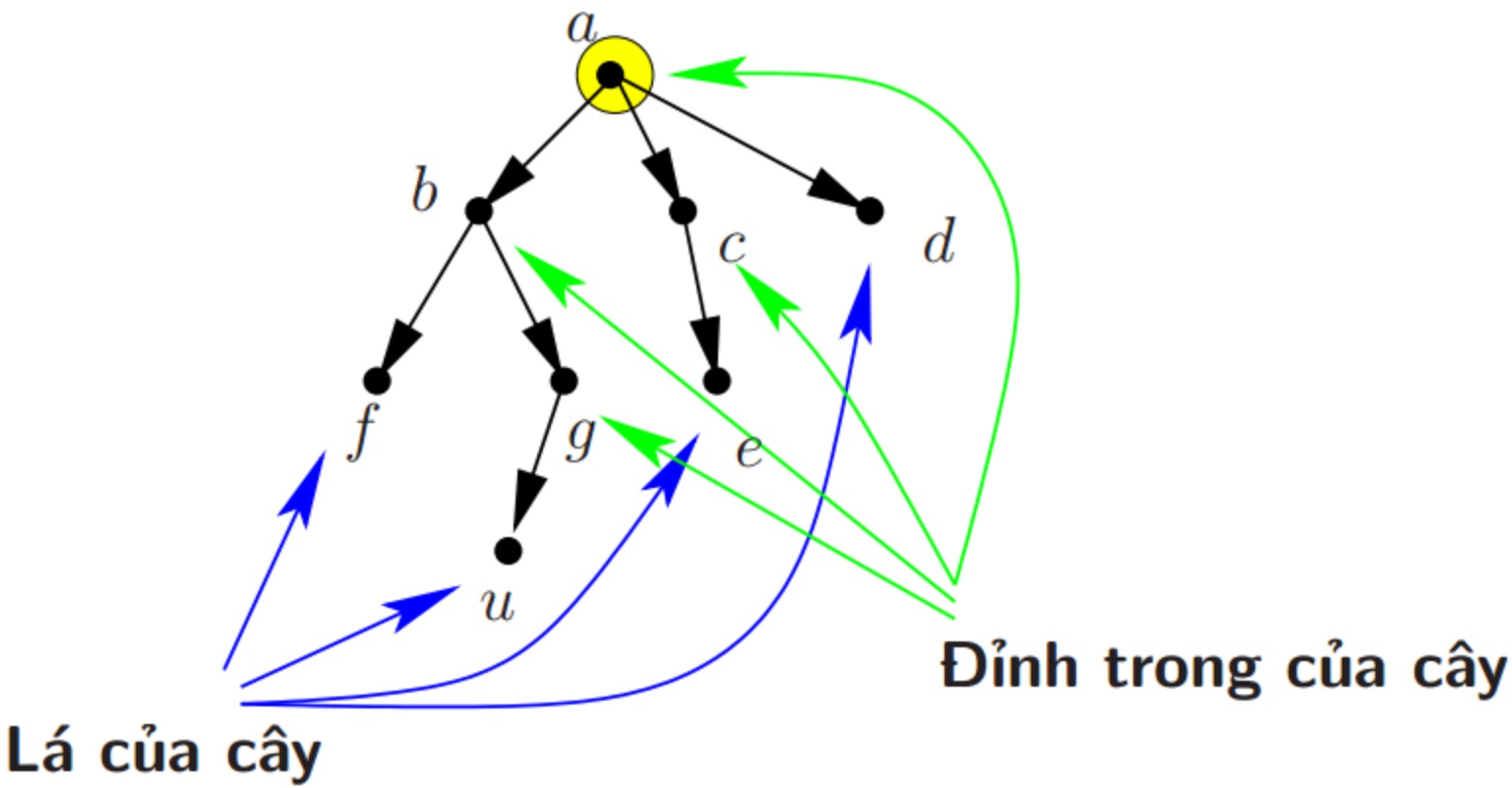
MỘT SỐ THUẬT NGỮ



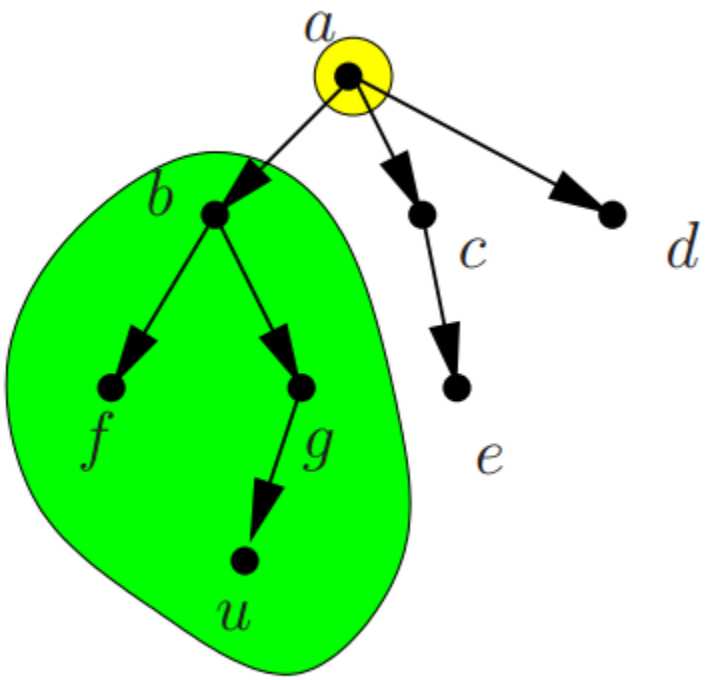
MỘT SỐ THUẬT NGỮ



MỘT SỐ THUẬT NGỮ



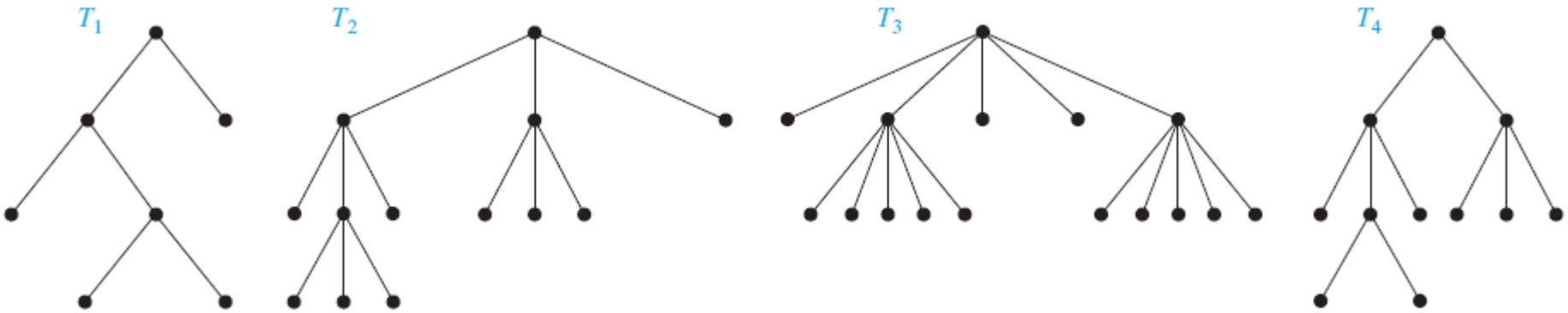
MỘT SỐ THUẬT NGỮ



Cây con gốc là b

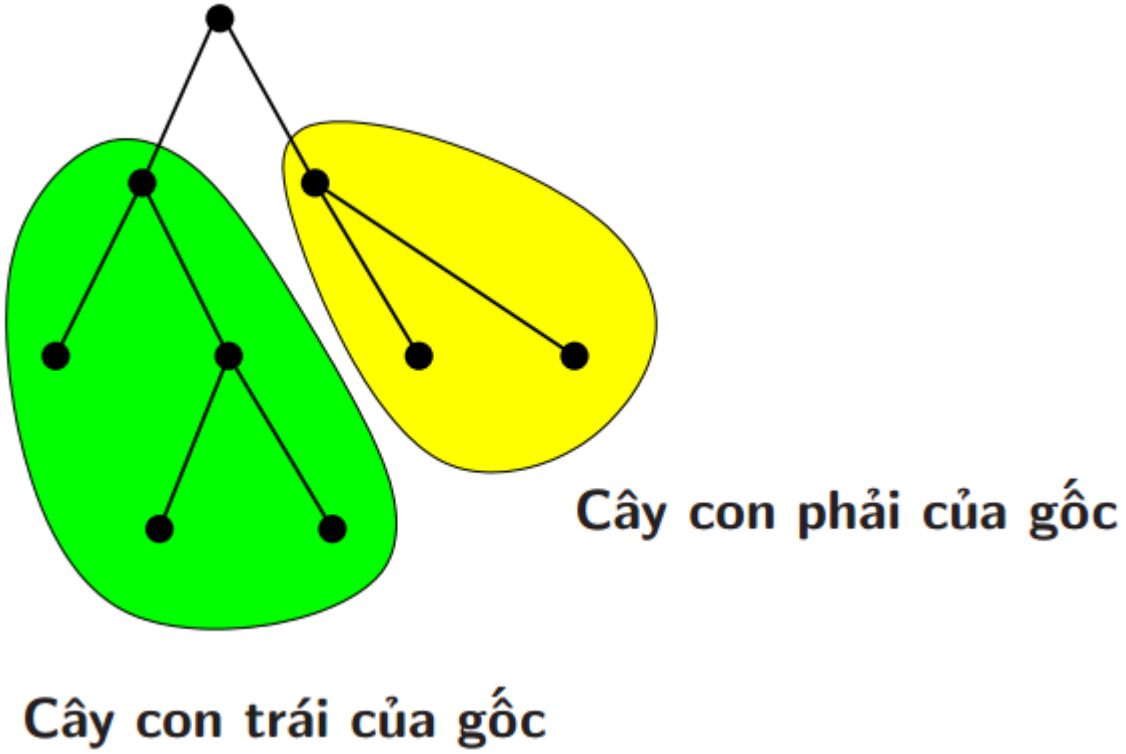
Định nghĩa 3:

- Cây có gốc được gọi là ***cây m-phân*** nếu tất cả các đỉnh trong của nó không có hơn ***m*** con.
- Cây được gọi là ***m-phân đầy đủ*** nếu mọi đỉnh trong có đúng ***m*** con.
- Cây *m*-phân với $m = 2$ gọi là ***cây nhị phân***

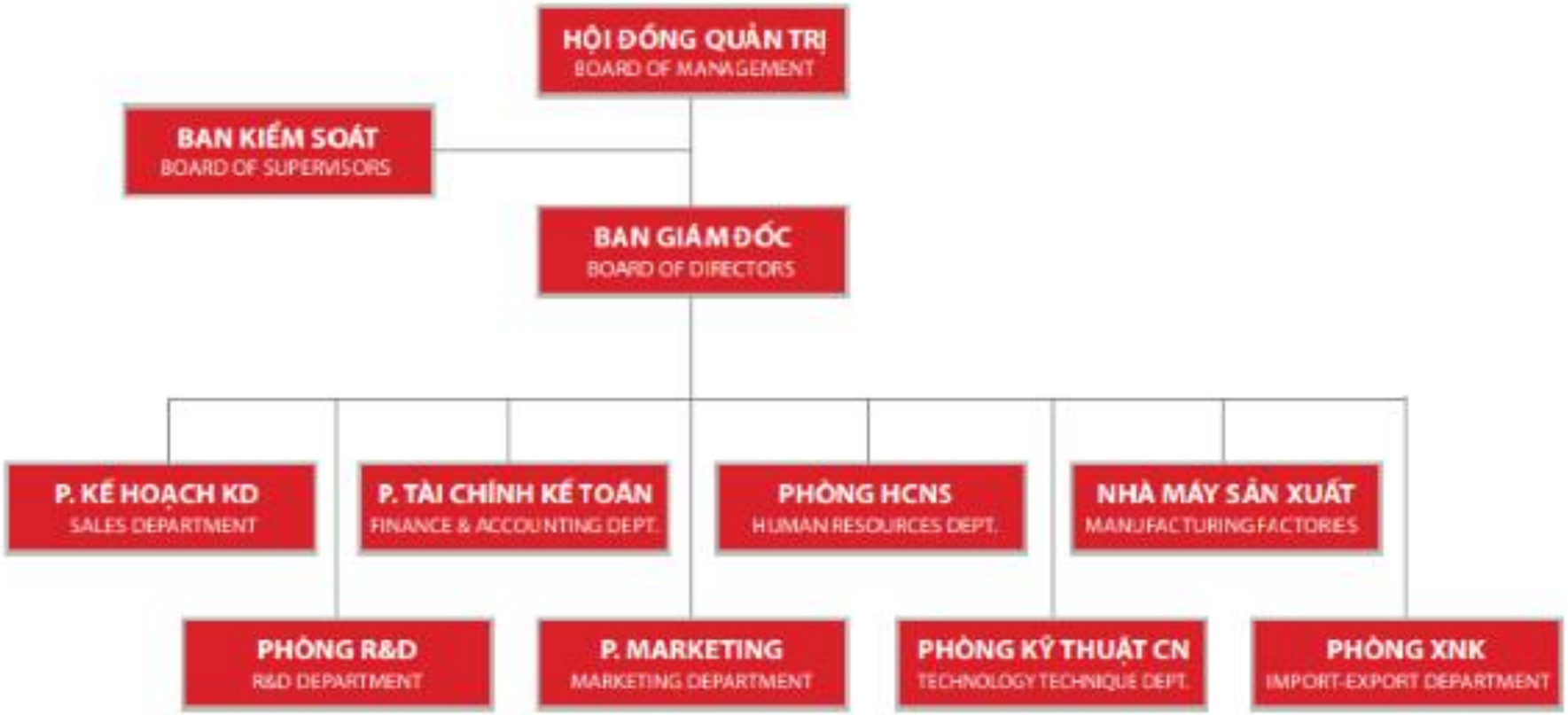


Cây có gốc được sắp:

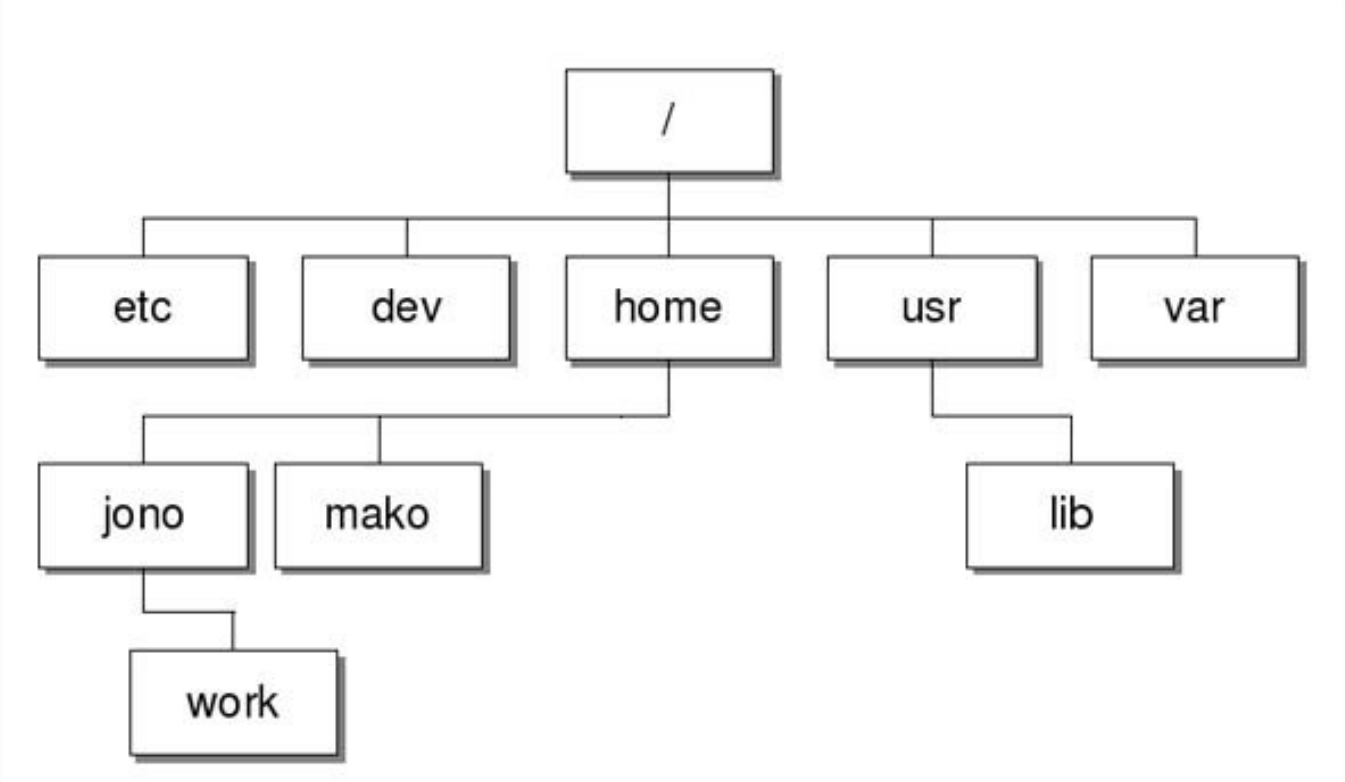
Cây có gốc được sắp (có thứ tự) là cây có gốc trong đó các con của mỗi đỉnh trong được sắp xếp theo một thứ tự nhất định.



VÍ DỤ VỀ CÂY



Tổ chức trong công ty



Cấu trúc thư mục

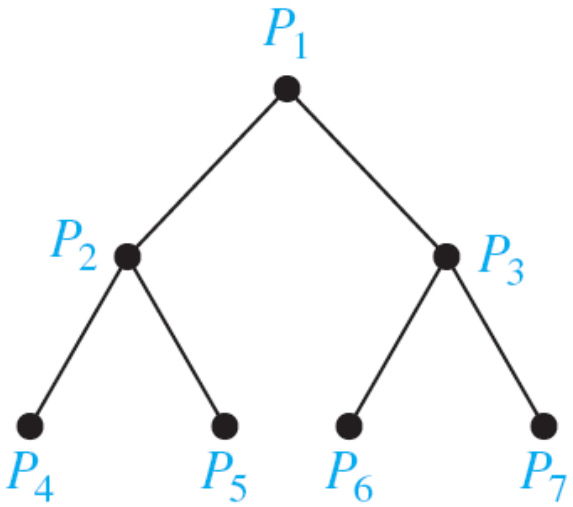
CÁC TÍNH CHẤT CỦA CÂY

Định lí 2:

Cây với ***n*** đỉnh có đúng **(*n*-1)** cạnh

Định lí 3:

Cây m-phân đầy đủ với ***i*** đỉnh trong sẽ có tất cả ***n = m.i*** **+ 1** đỉnh.



Định lí 4:

Cây ***m-phân đầy đủ*** với

(i) **n đỉnh** có $i = \frac{n-1}{m}$ đỉnh trong và $l = \frac{(m-1)n+1}{m}$ lá

(*ii*) **i đỉnh trong**, có $n = m.i + 1$ đỉnh và $l = (m-1).i + 1$ lá

(iii) **l lá**, có $n = \frac{ml-1}{m-1}$ đỉnh và $i = \frac{l-1}{m-1}$ đỉnh trong

Ví dụ:

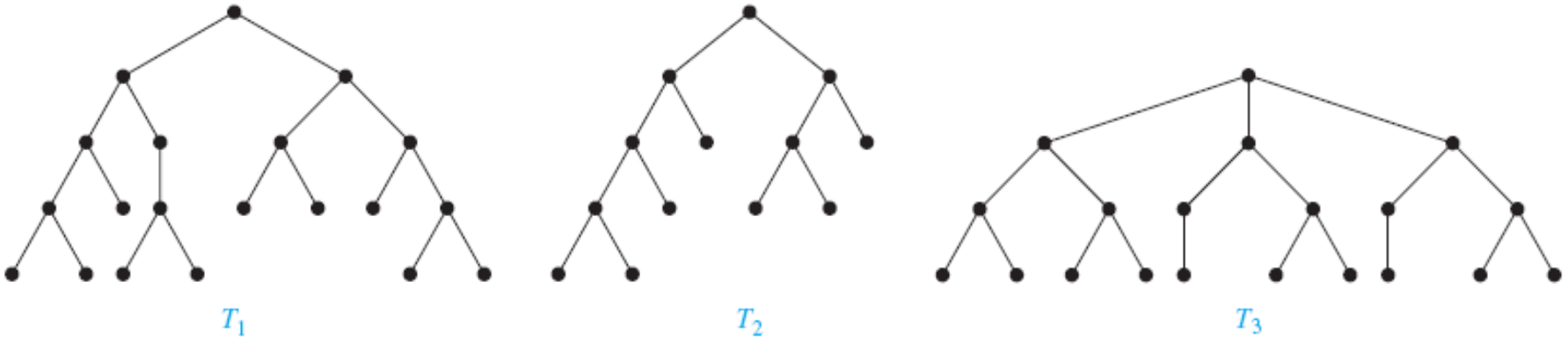
Trò chơi viết thư dây chuyền. Ban đầu có một người nhận được một bức thư và giả sử rằng khi nhận được một bức thư hoặc sẽ viết thư cho bốn người khác hoặc không viết cho ai. Hỏi **có bao nhiêu người nhận được thư** kể cả người đầu tiên nếu không có ai nhận được nhiều hơn một bức và trò chơi kết thúc khi có 100 người nhận thư mà ko viết cho ai?

Giải:

- Trò chơi biểu diễn bằng cây tứ phân.
- Có 100 không viết thư nên số lá của cây là $l = 100$
- Số người nhận thư là $n = (4.100 - 1)/(4-1) = 133$
- Số các đỉnh trong là $i = (100-1)/(4-1) = 33$ đỉnh, tức 33 người viết thư

CÁC TÍNH CHẤT CỦA CÂY

- **Mức** của đỉnh *v* trong cây là độ dài của đường đi duy nhất tới nó
- **Độ cao** của cây là mức cao nhất của tất cả các đỉnh
- **Cây *m*-phân** có gốc và độ cao *h* **được gọi là cân đối** nếu tất cả các lá đều ở mức *h* và (*h*-1)



Định lí 5:

Có nhiều nhất m^h **lá** trong cây ***m*-phân** với độ cao ***h***

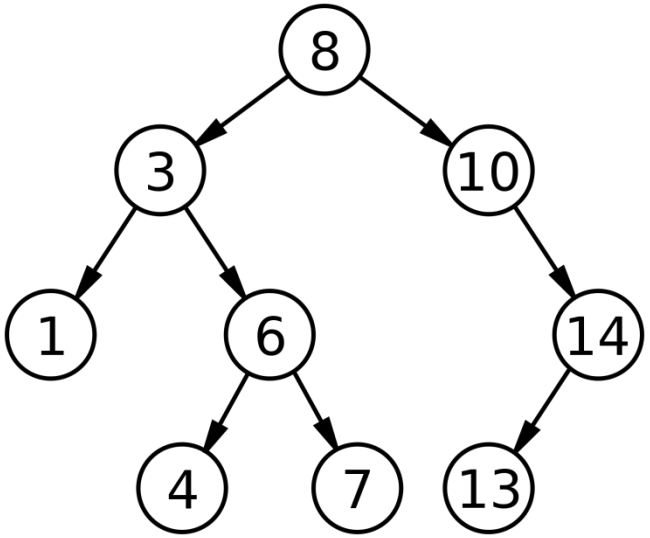
9.2 CÁC ỨNG DỤNG CỦA CÂY

CÁC ỨNG DỤNG CỦA CÂY

- **Cây tìm kiếm nhị phân**
- **Cây quyết định**
- **Các mã tiền tố**

Cây nhị phân:

- Cây có một **con trái** và một **con phải**
- Đỉnh được gán một khóa sao cho:
  - Lớn hơn khóa của tất cả các đỉnh thuộc cây con trái
  - Nhỏ hơn khóa của tất cả các đỉnh thuộc cây con bên phải

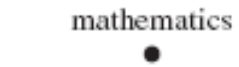
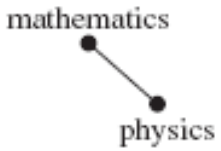


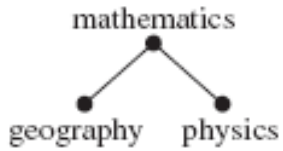
Xây dựng cây tìm kiếm nhị phân:

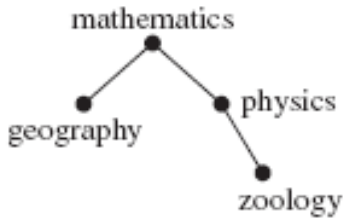
- Bắt đầu cây có đúng 1 đỉnh (gốc)
- Thêm một phần tử mới:
  - ✓ So sánh với khóa của đỉnh, bắt đầu từ gốc
  - ✓ Đi sang trái nếu nó nhỏ hơn
  - ✓ Đi sang phải nếu nó lớn hơn
  - ✓ **Tạo đỉnh mới là con bên trái** nếu phần tử nhỏ hơn khóa của đỉnh và đỉnh không có con trái
  - ✓ **Tạo đỉnh mới là con bên phải** nếu phần tử lớn hơn khóa của đỉnh và đỉnh không có con phải

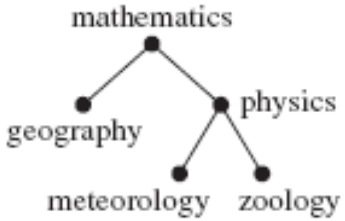
TÌM KIẾM NHỊ PHÂN

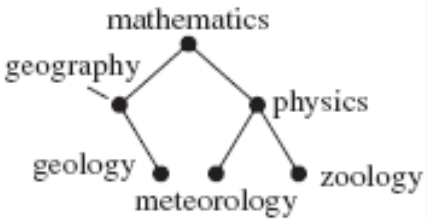
Ví dụ: Tạo cây tìm kiếm nhị phân cho các từ sau: *mathematics, physics,geography, zoology, meteology, geology, psychology, chemistry*

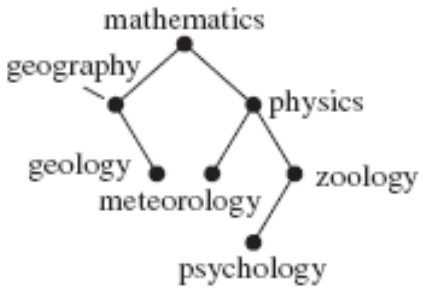
<table>
<tr>
<td> 
</td>
<td> 

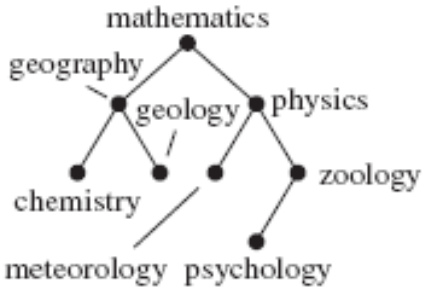
Physics > Mathemetics</td>
<td> 

Geography < Mathemetics</td>
<td> 

Zoology > Mathemetics
Zoology > Physics</td>
</tr>
<tr>
<td> 

Meteorology > Mathemetics
Meteorology < Physics</td>
<td> 

Geology < Mathemetics
Geology > Geography</td>
<td> 

Psychology > Mathemetics
Psychology > Physics
Psychology < Zoology</td>
<td> 

Chemistry < Mathemetics
Chemistry < Geography</td>
</tr>
</table>

Thuật toán tìm kiếm nhị phân:

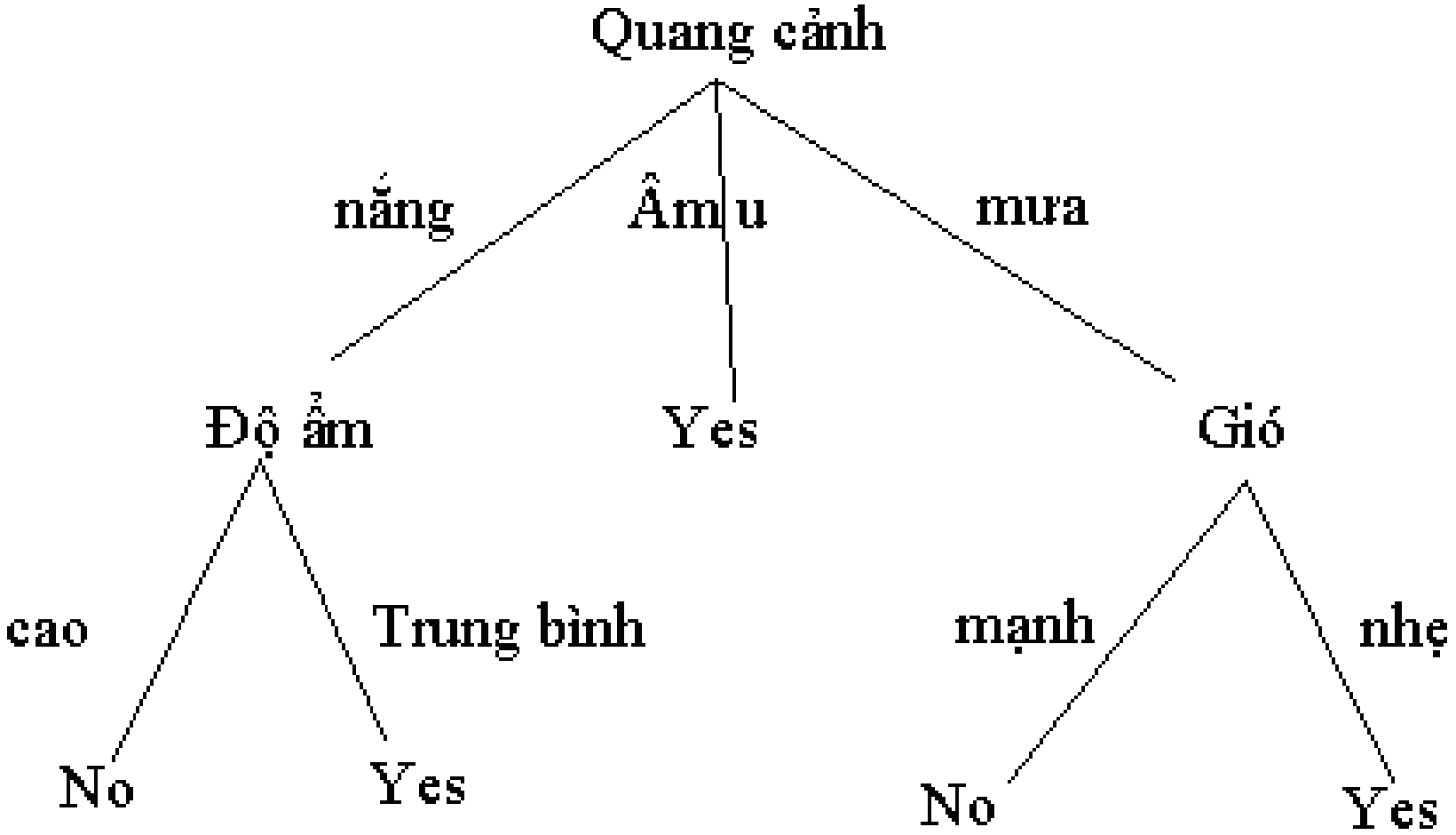
- Định vị phần tử *x* trong cây tìm kiếm nhị phân nếu nó là khóa của một đỉnh
- Nếu *x* không là khóa của đỉnh nào, thêm mới *x* vào cây

TÌM KIẾM NHỊ PHÂN

THUẬT TOÁN : Thuật toán tìm kiếm nhị phân

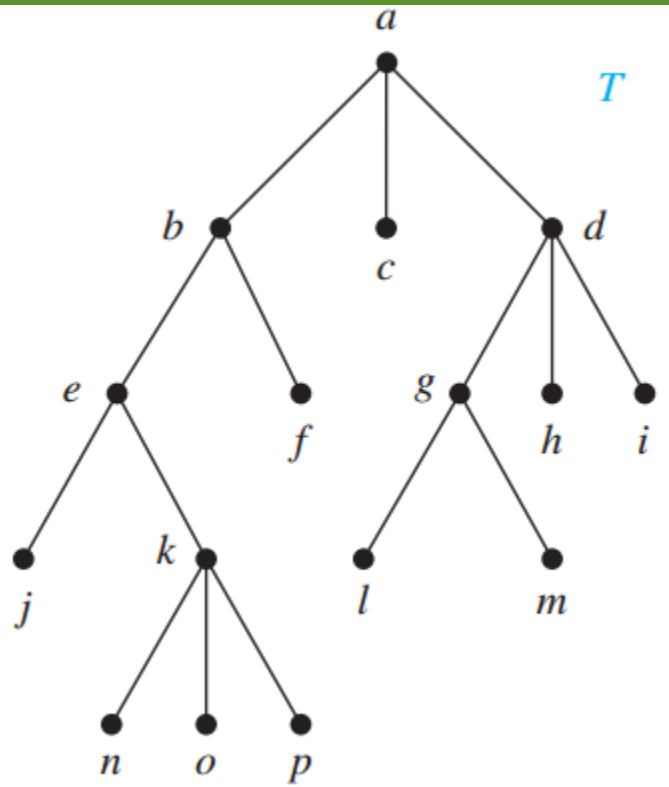
Procedure *insertion*(*T:* cây tìm kiếm nhị phân, *x*: phần tử)
v := gốc của *T* { đỉnh không có trong *T* sẽ có giá trị bằng *null* }
while *v* ≠ *null* và *label(v)* ≠ *x*
begin
    if *x* < *label(v)* **then**
        if con bên trái của *v* ≠ null **then** *v* := con bên trái của *v*
        else thêm *đỉnh mới* là con trái của *v* và đặt *v* := *null*
    else
        if con bên phải của *v* ≠ null **then** *v* := con bên phải của *v*
        else thêm *đỉnh mới* là con phải của *v* và đặt *v* := *null*
end
if gốc của *T* = *null* **then** thêm đỉnh *v* vào cây và gán cho nó nhãn là *x*
else if *v* = *null* hoặc *label(v)* ≠ *x* **then** gán nhãn cho *đỉnh mới* là *x* và đặt *v* là đỉnh mới này
{ v = vị trí của *x* }

CÂY QUYẾT ĐỊNH



9.3 CÁC PHƯƠNG PHÁP DUYỆT CÂY

CÁC PHƯƠNG PHÁP DUYỆT CÂY



- *Duyệt **tiền** thứ tự*
- *Duyệt **trung** thứ tự*
- *Duyệt **hậu** thứ tự*

DUYỆT TIỀN THỨ TỰ

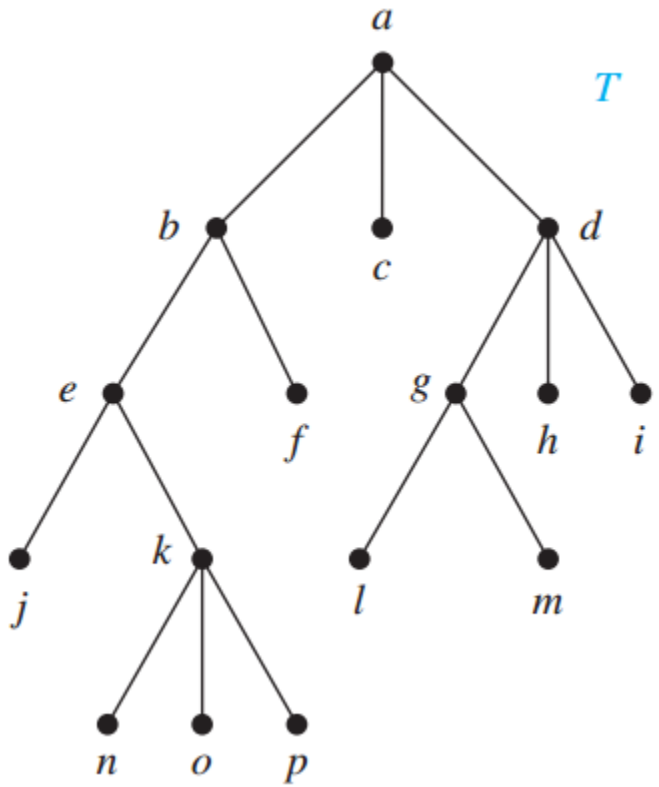
Định nghĩa 1:

Giả sử T là một cây có gốc được sắp thứ tự với gốc r. Nếu T chỉ có r thì r là ***duyệt tiền thứ tự*** của T. Nếu không thì gọi T_1 , T_2 …, T_n là các cây con tại r từ trái qua phải của T. **Duyệt tiền thứ tự:**

- *Thăm* **r** *đầu tiên*
- *Duyệt* $\mathbf{T_1}$ *theo kiểu tiền thứ tự*
- *Duyệt* $\mathbf{T_2}$ *theo kiểu tiền thứ tự*
- .....
- *Duyệt* $\mathbf{T_n}$ *theo kiểu tiền thứ tự*

DUYỆT TIỀN THỨ TỰ

Ví dụ: Cách duyệt **tiền** thứ tự sẽ viếng thăm các đỉnh của cây theo thứ tự nào?



DUYỆT TIỀN THỨ TỰ

THUẬT TOÁN : Duyệt kiểu tiền thứ tự

Procedure *Preorder (T:* cây có gốc được sắp)
r := gốc của *T*
Liệt kê ***r***
for mỗi cây con ***c*** của ***r*** từ trái sang phải
begin
    T(*c*) := cây con với gốc *c*
    Preorder(*T*(*c*))
end

DUYỆT TRUNG THỨ TỰ

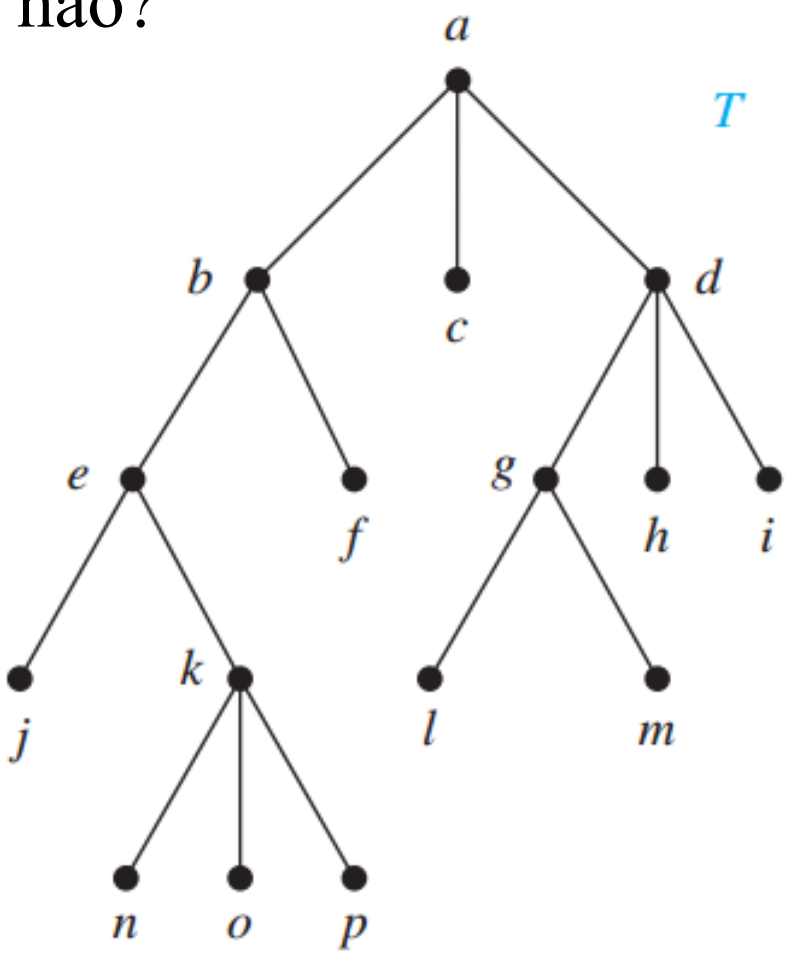
Định nghĩa 2:

Giả sử T là một cây có gốc được sắp thứ tự với gốc r. Nếu T chỉ có r thì r là ***duyệt trung thứ tự*** của T. Nếu không thì gọi T_1 , T_2 …, T_n là các cây con tại r từ trái qua phải của T. **Duyệt trung thứ tự:**

- *Duyệt* $\boldsymbol{T_1}$ *theo kiểu trung thứ tự*
- *Thăm* **r**
- *Duyệt* $\boldsymbol{T_2}$ *theo kiểu trung thứ tự*
- .....
- *Duyệt* $\boldsymbol{T_n}$ *theo kiểu trung thứ tự*

DUYỆT TRUNG THỨ TỰ

Ví dụ: Cách duyệt **trung** thứ tự sẽ viếng thăm các đỉnh của cây theo thứ tự nào?



DUYỆT TRUNG THỨ TỰ

THUẬT TOÁN : Duyệt kiểu trung thứ tự

Procedure *Inorder (T:* cây có gốc được sắp)
r := gốc của *T*
if *r* là lá **then** liệt kê *r*
else
begin
l := con đầu tiên từ trái sang phải của r
T(l) := cây con với gốc *l*
Inorder(*T*(*l*))
Liệt kê ***r***
for mỗi cây con ***c*** của ***r*** từ trái sang phải trừ *l*
    T(*c*) := cây con với gốc *c*
    Inorder(*T*(*c*))
end

DUYỆT HẬU THỨ TỰ

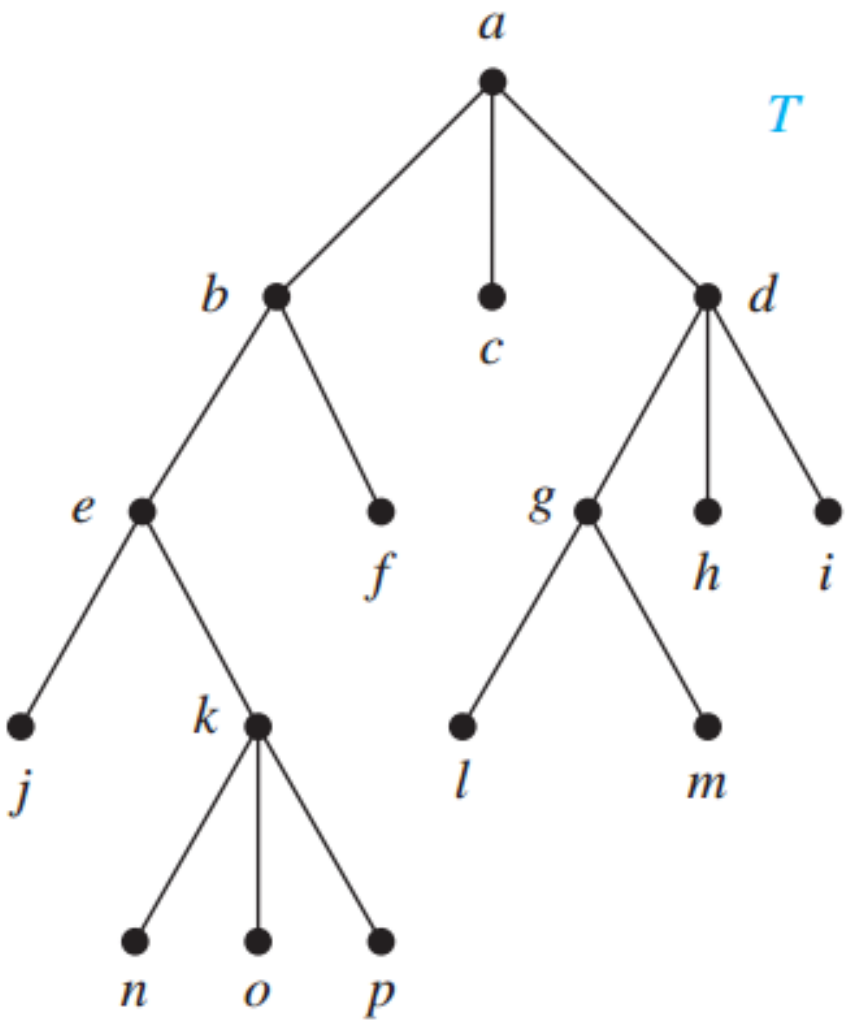
Định nghĩa 3:

Giả sử T là một cây có gốc được sắp thứ tự với gốc r. Nếu T chỉ có r thì r là ***duyệt hậu thứ tự*** của T. Nếu không thì gọi T_1 , T_2 …, T_n là các cây con tại r từ trái qua phải của T. **Duyệt hậu thứ tự:**

- *Duyệt* $\boldsymbol{T_1}$ *theo kiểu hậu thứ tự*
- *Duyệt* $\boldsymbol{T_2}$ *theo kiểu hậu thứ tự*
- .....
- *Duyệt* $\boldsymbol{T_n}$ *theo kiểu hậu thứ tự*
- *Thăm* **r**

DUYỆT TRUNG THỨ TỰ

Ví dụ: Cách duyệt **hậu** thứ tự sẽ viếng thăm các đỉnh của cây theo thứ tự nào?



DUYỆT HẬU THỨ TỰ

THUẬT TOÁN : Duyệt kiểu hậu thứ tự

Procedure *Postorder (T:* cây có gốc được sắp)
r := gốc của *T*
for mỗi cây con ***c*** của ***r*** từ trái sang phải
begin
    T(*c*) := cây con với gốc *c*
    Postorder(*T*(*c*))
end
Liệt kê ***r***

CÁC KÍ PHÁP TRUNG TỐ, TIỀN TỐ VÀ HẬU TỐ

- Có thể biểu diễn biểu thức phức tạp
  - **Mệnh đề phức hợp**
  - **Tập hợp**
  - **Biểu thức số học**

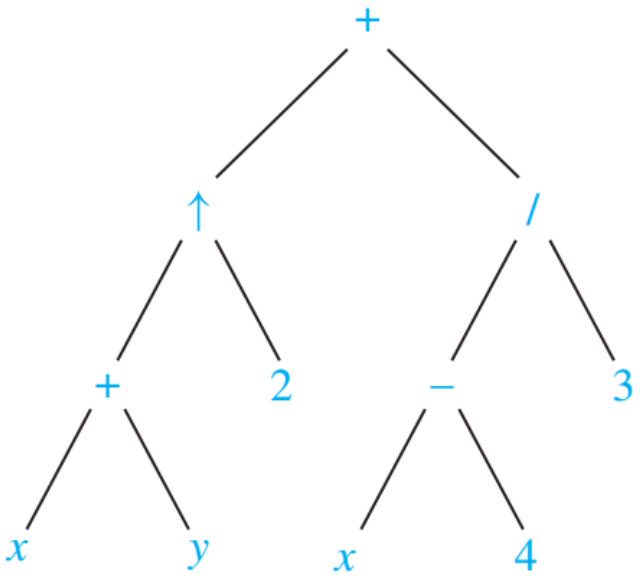
Bằng cây có gốc và được sắp, trong đó:

- **đỉnh trong** biểu thị các **phép toán**
- **lá** biểu thị các **số** hay các **biến**

CÁC KÍ PHÁP TRUNG TỐ, TIỀN TỐ VÀ HẬU TỐ

Ví dụ: Tìm cây có gốc biểu diễn biểu thức

$$\left((x+y)\uparrow 2\right)+\left(\frac{x-4}{3}\right)$$



Biểu thức có đầy đủ dấu ngoặc đơn gọi là **dạng trung tố**

CÁC KÍ PHÁP TRUNG TỐ, TIỀN TỐ VÀ HẬU TỐ

- Khi duyệt cây có gốc theo kiểu tiền thứ tự có *dạng tiền tố* của biểu thức
- Biểu thức được viết dưới dạng tiền tố gọi là **kí pháp Ba Lan**
- **Đánh giá** một biểu thức ở dạng tiền tố:
  - Đi từ phải sang trái
  - Gặp một toán tử thực hiện phép toán tương ứng với hai toán hạng đi liền bên phải của toán tử

Ví dụ: Tính giá trị biểu thức tiền tố

+ - * 2 3 5 / ↑ 2 3 4

CÁC KÍ PHÁP TRUNG TỐ, TIỀN TỐ VÀ HẬU TỐ

- Khi duyệt cây có gốc theo kiểu hậu thứ tự có *dạng hậu tố* của biểu thức
- Biểu thức được viết dưới dạng hậu tố gọi là **kí pháp Ba Lan ngược**
- **Đánh giá** một biểu thức ở dạng tiền tố:
  - Đi từ trái sang phải
  - Thực hiện phép toán khi có một toán tử đi sau hai toán hạng

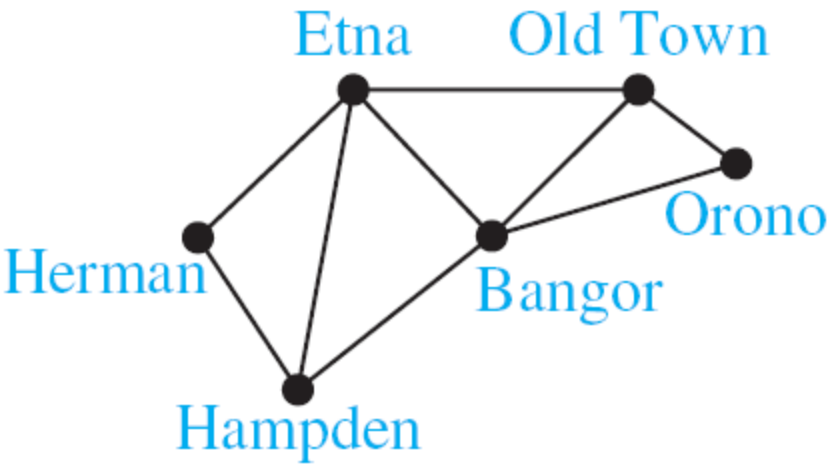
Ví dụ: Tính giá trị biểu thức hậu tố

7 8 + 2 ↑ 7 4 – 3 / +

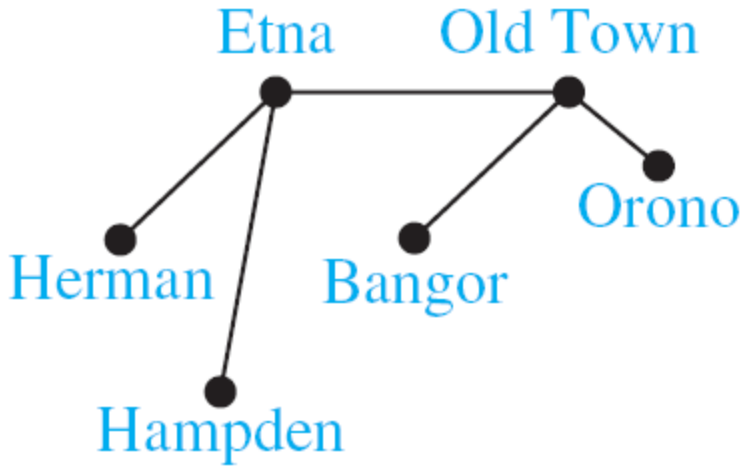
9.4 CÂY KHUNG

Bài toán giao thông ở Maine:

- Chính quyền địa phương muốn cào tuyết một số ít nhất các con đường sao cho luôn luôn có đường thông suốt nối hai thành phố bất kì



(a)



(b)

Định nghĩa 1:

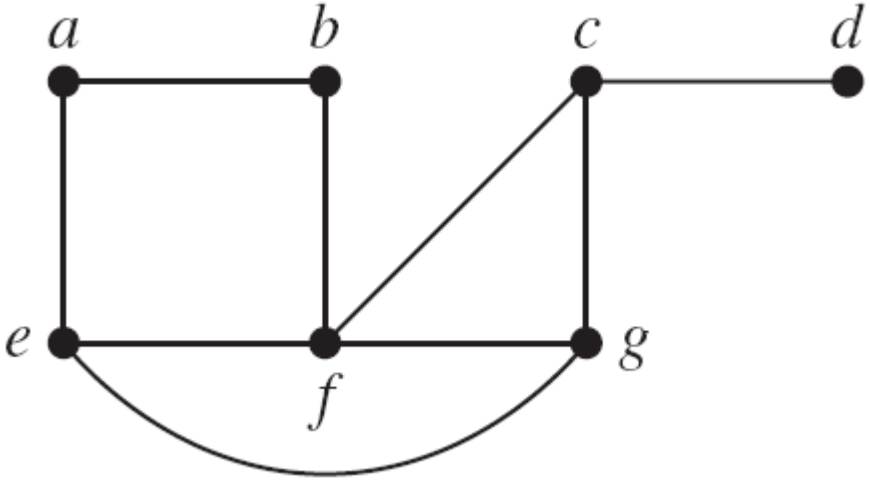
Cho G là một đơn đồ thị. Một cây được gọi là ***cây khung*** của G nếu nó là một đồ thị con của G và chứa tất cả các đỉnh của G.

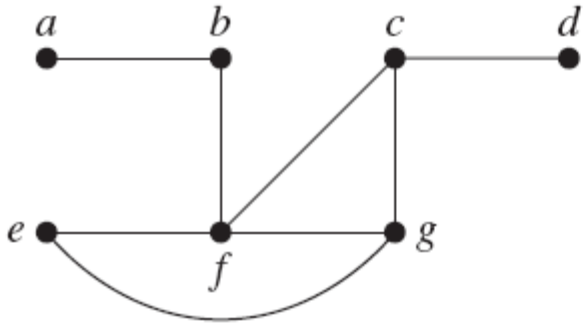
Tìm cây khung của đồ thị:

- ✓ *Xóa đi các cạnh tạo ra chu trình*
- ✓ *Bằng phương pháp tìm kiếm ưu tiên chiều sâu*
- ✓ *Bằng phương pháp tìm kiếm ưu tiên chiều rộng*

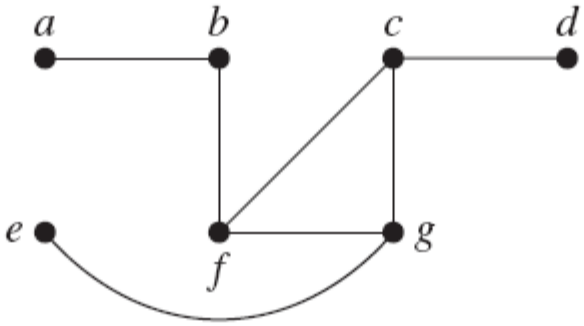
TÌM CÂY KHUNG CỦA ĐỒ THỊ

Xóa đi các cạnh tạo ra chu trình

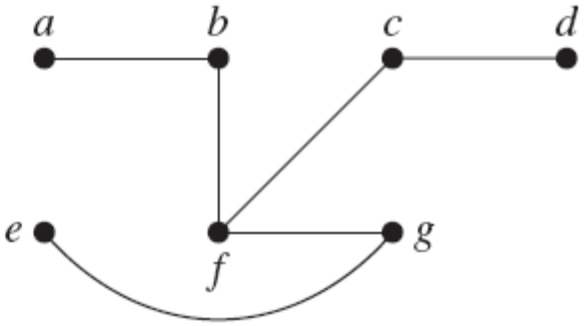




$\{a, e\}$



$\{e, f\}$



$\{c, g\}$

Định lí 1:

Một đơn đồ thị là ***liên thông*** nếu và chỉ nếu nó **có cây khung**

TÌM KIẾM ƯU TIÊN CHIỀU SÂU

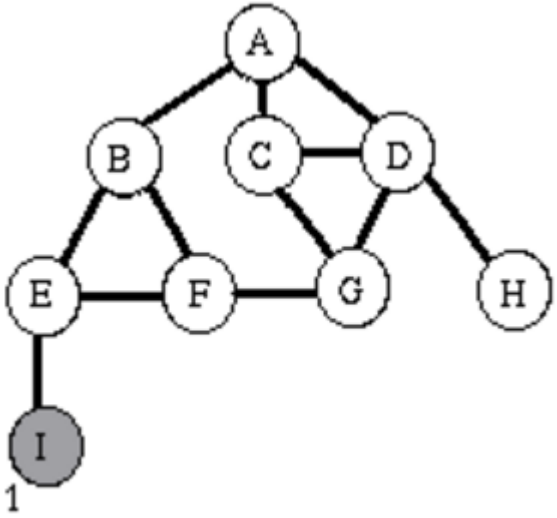
- Chọn một đỉnh của đồ thị làm gốc
- Xây dựng đường đi từ đỉnh gốc bằng cách ghép các cạnh vào đường đi cho đến khi không thể ghép
- Nếu đường đi không qua tất cả các cạnh thì lùi lại một bước và xây dựng đường đi mới qua các đỉnh chưa thuộc đường đi.

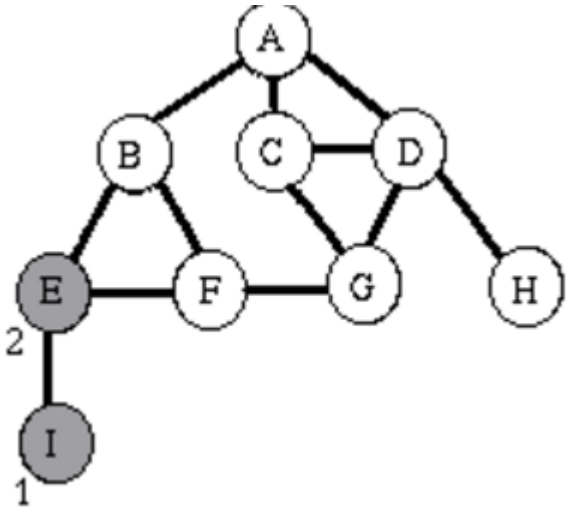
- Tiếp tục lùi lại cho đến khi không thực hiện được nữa

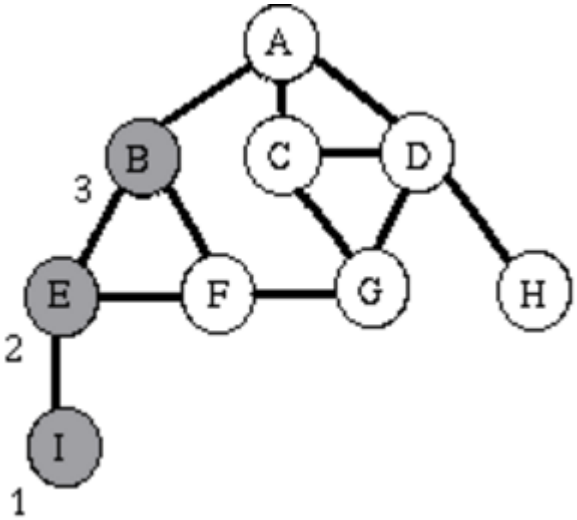
- *Tìm kiếm ưu tiên chiều sâu cũng được gọi là thủ tục* ***quay lui***

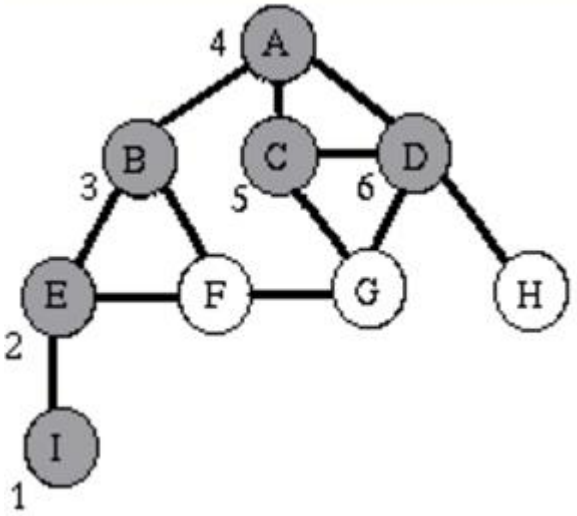
TÌM KIẾM ƯU TIÊN CHIỀU SÂU

Ví dụ:

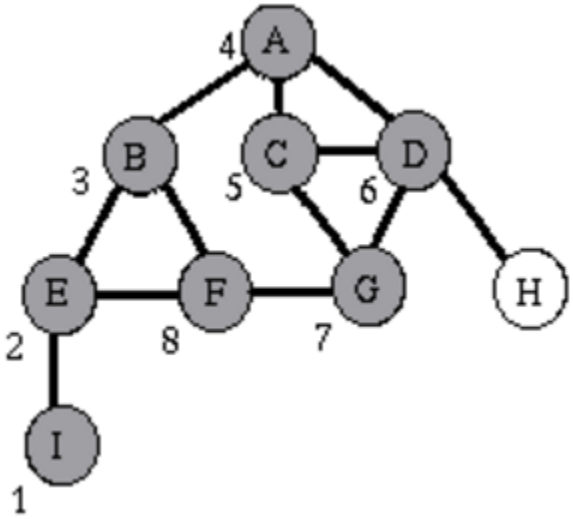


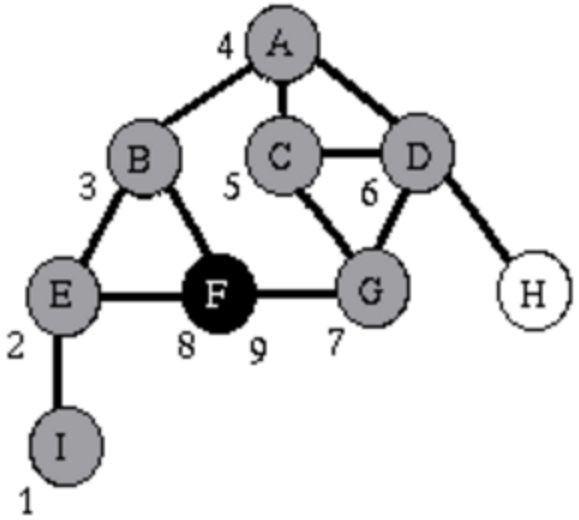


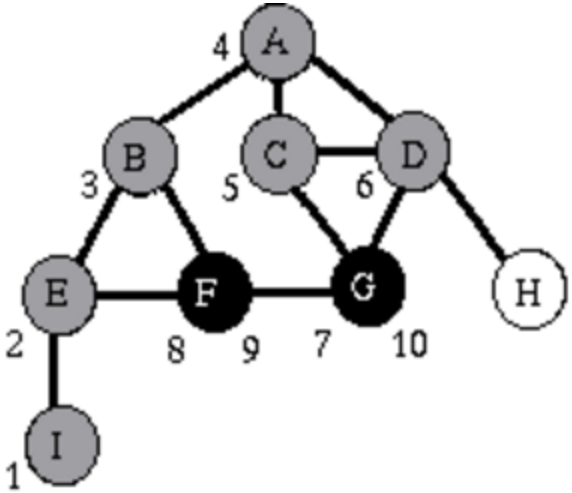


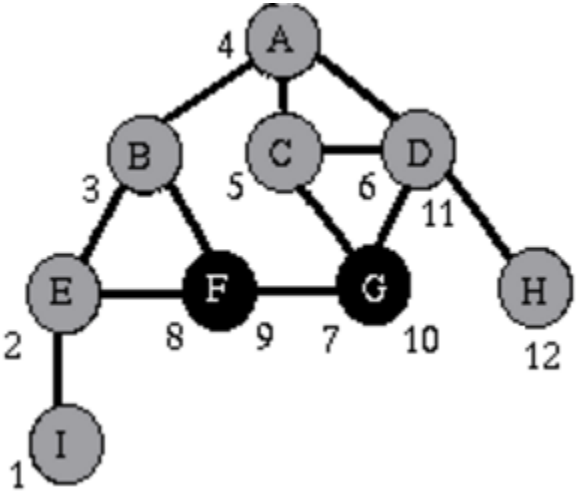


TÌM KIẾM ƯU TIÊN CHIỀU SÂU









TÌM KIẾM ƯU TIÊN CHIỀU SÂU

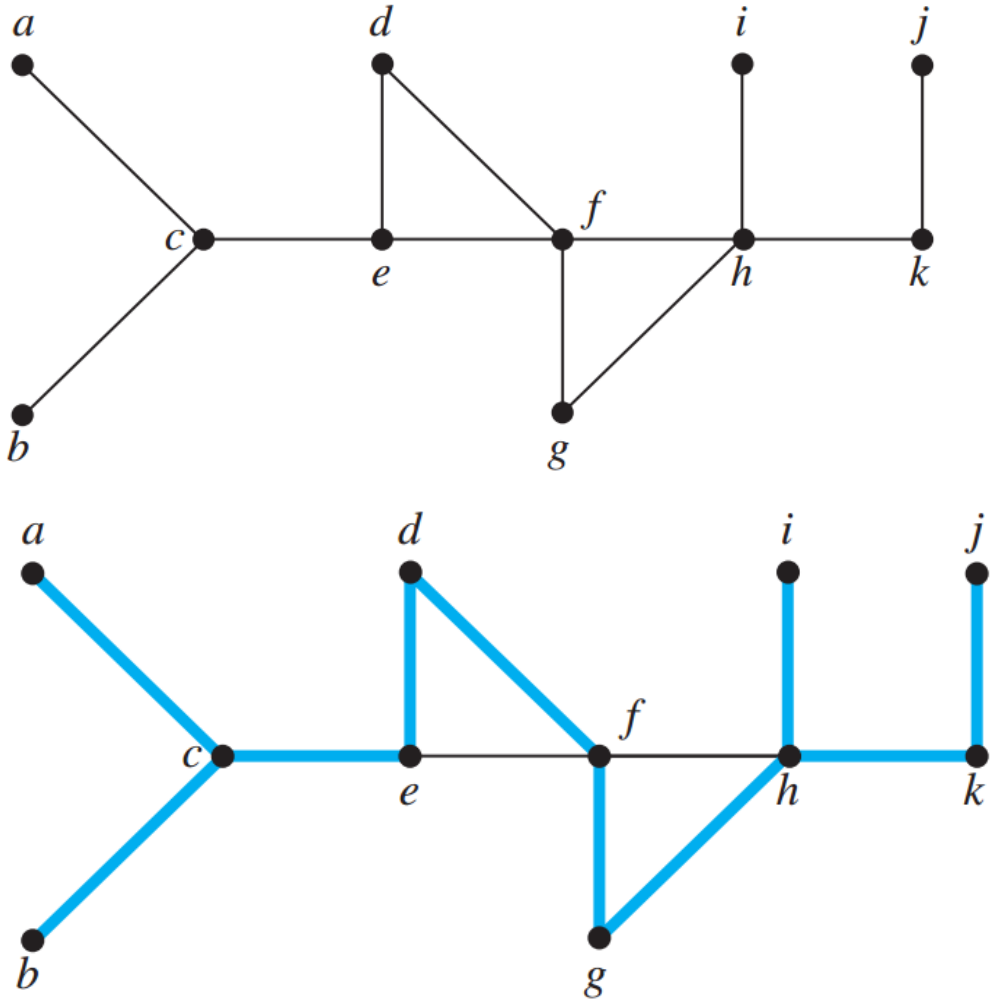
THUẬT TOÁN : Tìm kiếm ưu tiên chiều sâu

Procedure *DFS(G:* đồ thị liên thông với các đỉnh v_1 , v_2... v_n)
T := cây chỉ chứa một đỉnh v_1
Visit(v_1)

Procedure *Visit(v:* đỉnh của G)
for mỗi đỉnh ***w*** liền kề với ***v*** và chưa có trong *T*
begin
thêm đỉnh *w* và cạnh (*v, w)* vào *T*
Visit(w)
end

TÌM KIẾM ƯU TIÊN CHIỀU SÂU

Ví dụ:



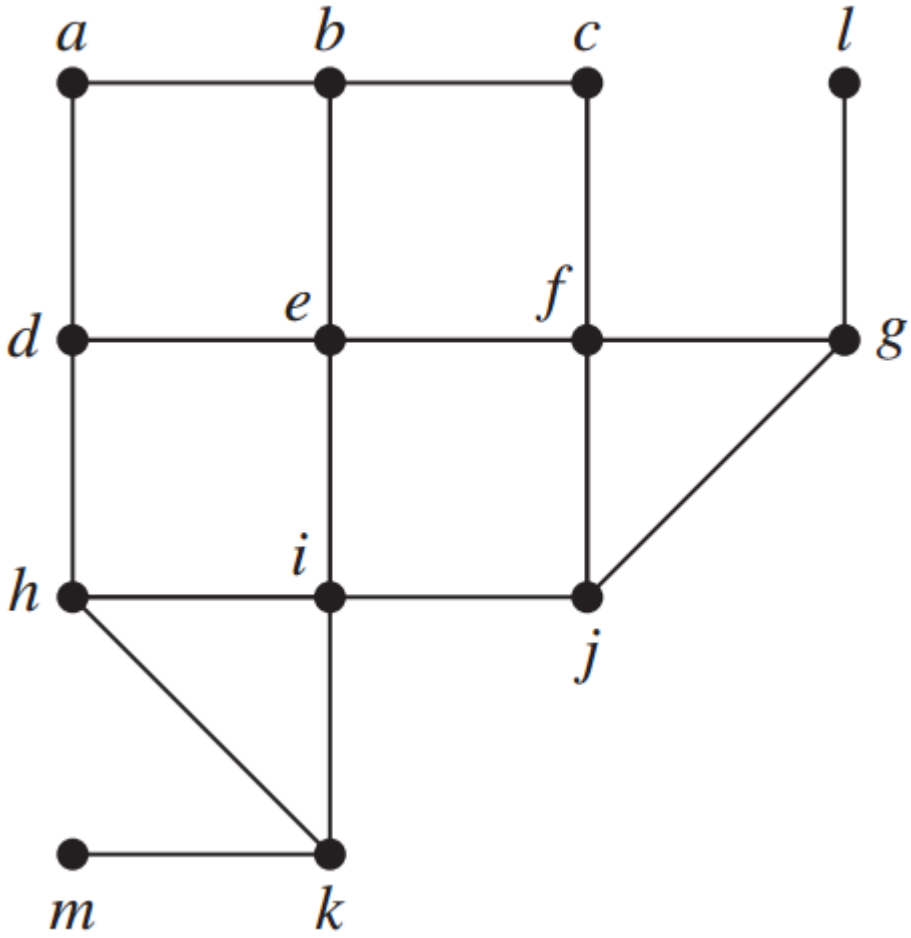
TÌM KIẾM ƯU TIÊN CHIỀU RỘNG

- Chọn một đỉnh của đồ thị làm gốc
- Ghép vào tất cả các cạnh liên thuộc với đỉnh này. Đỉnh ghép vào là đỉnh mức 1 của cây khung
- Với mỗi đỉnh mức 1, ghép tất cả các cạnh liên thuộc với nó vào cây mà không tạo ra chu trình.

- Tiếp tục cho đến khi tất cả các cạnh được ghép vào cây

TÌM KIẾM ƯU TIÊN CHIỀU RỘNG

Ví dụ: Dùng thuật toán ưu tiên chiều rộng, tìm cây khung của đồ thị

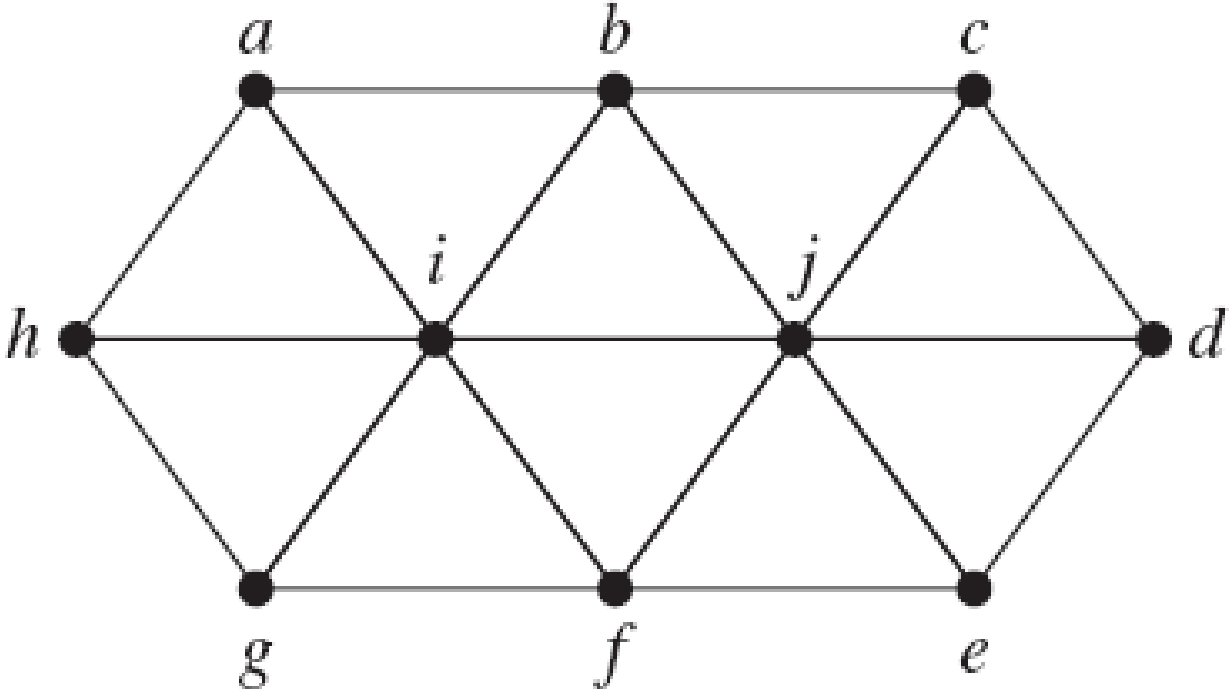


THUẬT TOÁN : Tìm kiếm ưu tiên chiều rộng

Procedure *BFS(G:* đồ thị liên thông với các đỉnh v_1 , v_2... v_n)
T := cây chỉ chứa một đỉnh v_1
L := danh sách rỗng
Đặt v_1 vào danh sách *L* gồm các đỉnh không xử lí
while *L khác rỗng*
begin
Xóa đỉnh đầu tiên, *v*, khỏi *L*
for mỗi đỉnh liền kề ***w*** của ***v***
    if *w* chưa nằm trong *L* và không thuộc *T* **then**
    begin
      thêm đỉnh *w* vào cuối danh sách *L*
      thêm *w* và cạnh {*v*, *w*} vào *T*
    end
end

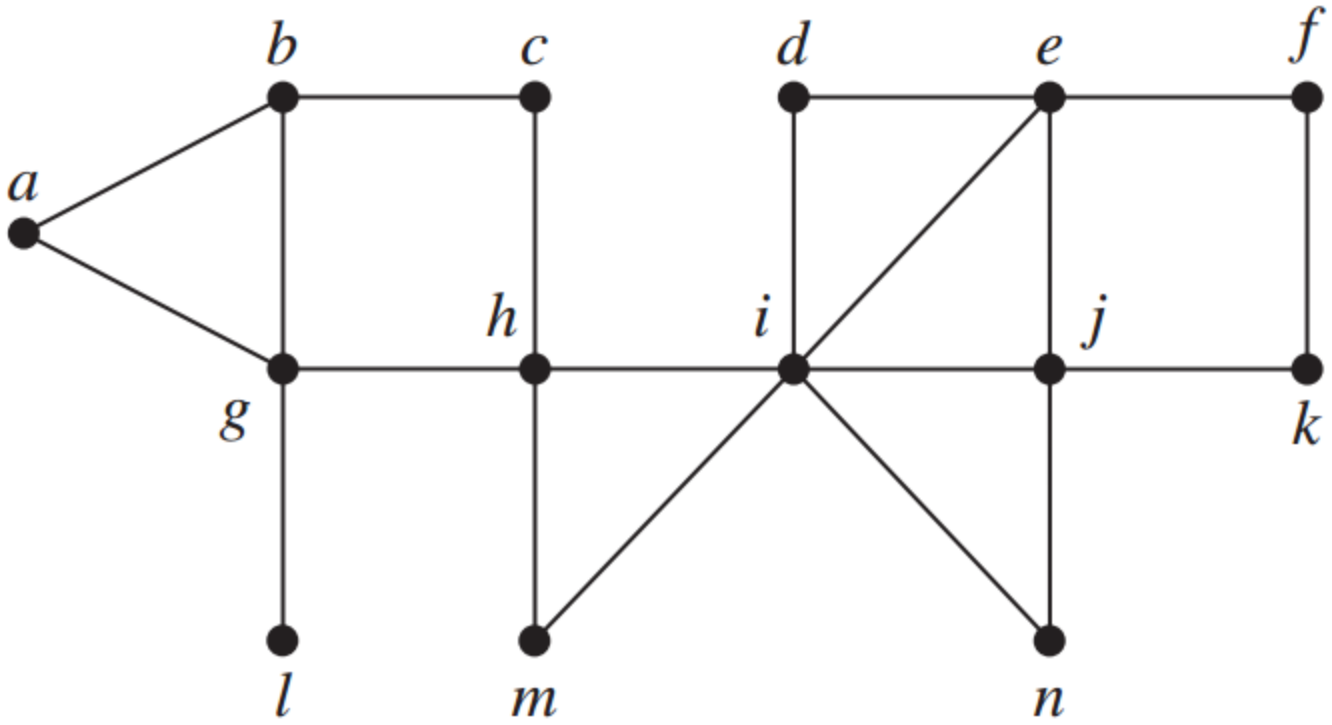
BÀI TẬP

- **Bài 1**: Tìm cây khung của đồ thị bằng cách xóa đi các cạnh



BÀI TẬP

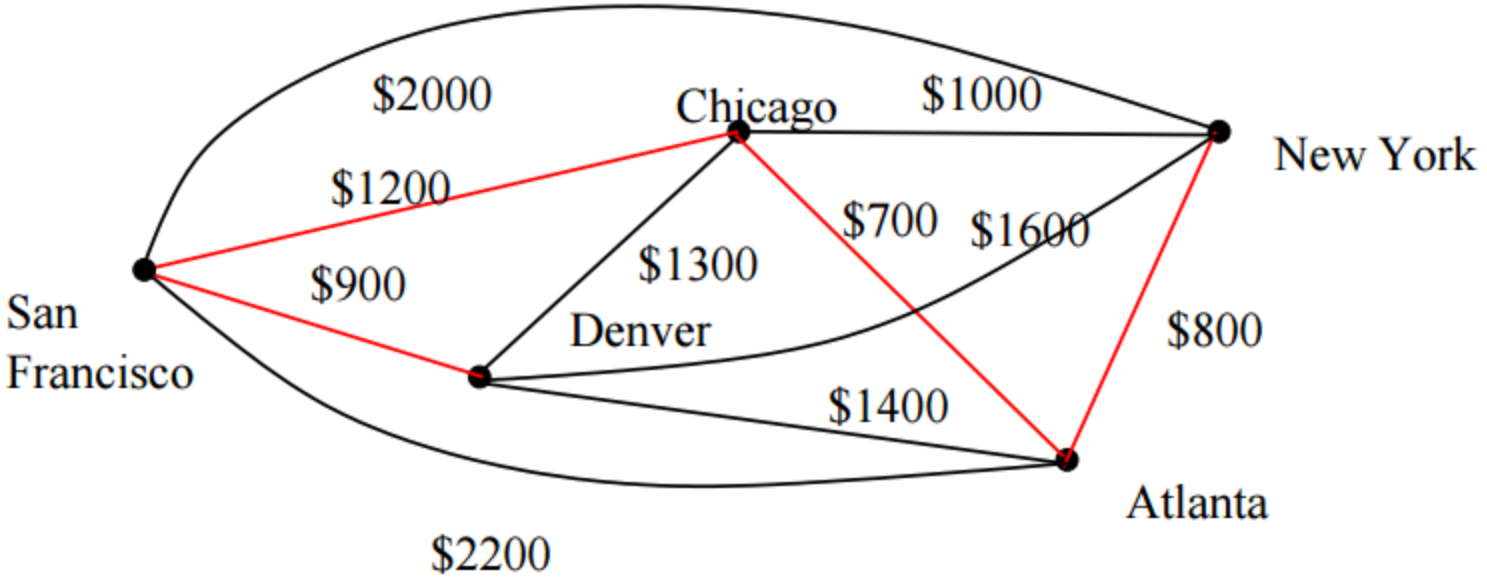
- **Bài 2**: Tìm cây khung của đồ thị bằng kĩ thuật tìm kiếm ưu tiên chiều sâu. Chọn *a* làm gốc của cây.



9.5 CÂY KHUNG NHỎ NHẤT

Định nghĩa 1:

Cây khung nhỏ nhất trong một đồ thị liên thông có trọng số là một cây khung có **tổng trọng số** trên các cạnh của nó là **nhỏ nhất**



Thuật toán tìm cây khung nhỏ nhất:

- Ghép các cạnh có trọng số nhỏ nhất vào cây

- Là ví dụ về thuật toán tham lam (*thủ tục thực hiện một lựa chọn tối ưu ở mỗi giai đoạn, không đảm bảo tối ưu toàn cục*)

- **2 Thuật toán**:
  - ✓ Thuật toán Prim
  - ✓ Thuật toán Kruskal

THUẬT TOÁN PRIM

- Do Robert Prim đưa ra năm 1957

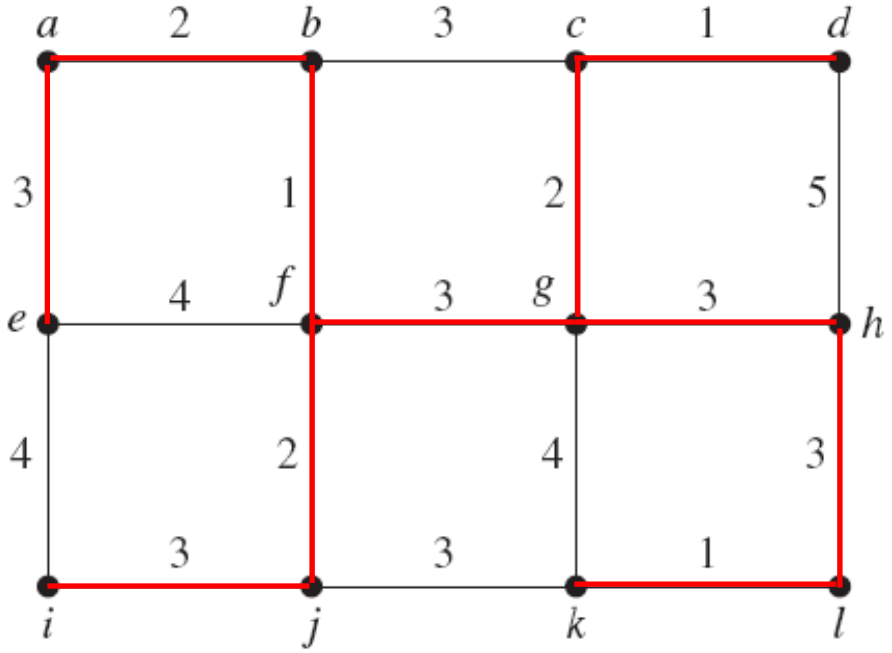
- **Thuật toán**:
  - ✓ ***Chọn một cạnh bất kì*** có trọng số nhỏ nhất, đặt vào khung
  - ✓ ***Ghép vào cây*** các cạnh có trọng số tối thiểu **liên thuộc với đỉnh** của cây, không tạo ra chu trình
  - ✓ ***Dừng*** khi có (n-1) cạnh được ghép vào cây

THUẬT TOÁN PRIM

THUẬT TOÁN : Thuật toán Prim

Procedure *Prim*(*G:* đồ thị liên thông có trọng số với *n* đỉnh)
T := cạnh có trọng số nhỏ nhất
for *i*:= 1 **to** *n*-2
begin
    e := cạnh có trọng số tối thiểu liên thuộc với một đỉnh trong *T* và không tạo ra chu trình trong *T* nếu ghép nó vào *T*
    T := *T* với *e* được ghép vào
end
{ T là cây khung nhỏ nhất của G }

Ví dụ: Tìm cây khung nhỏ nhất

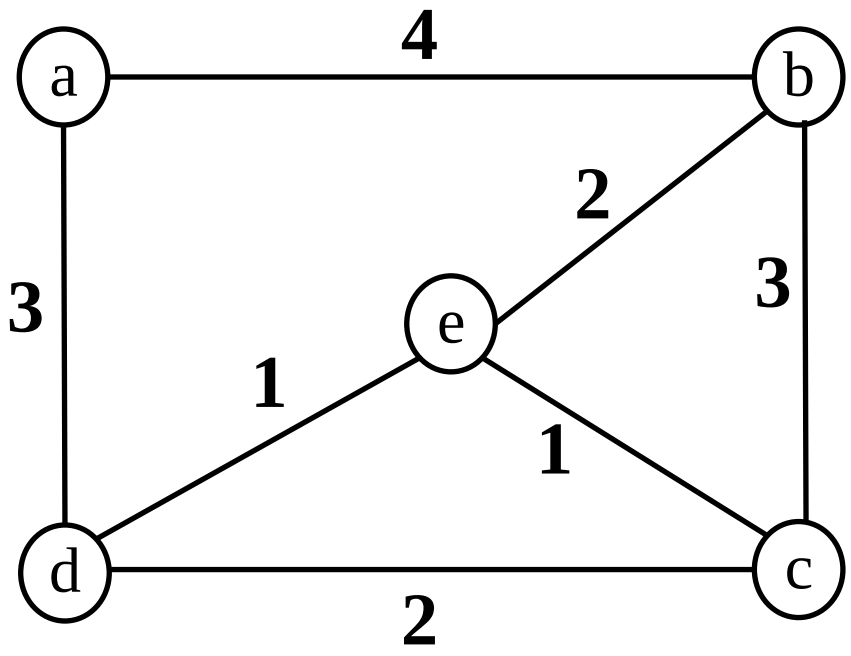


Lần chọn	Cạnh	Trọng số
1	{b, f}	1
2	{a, b}	2
3	{f, j}	2
4	{a, e}	3
5	{i, j}	3
6	{f, g}	3
7	{c, g}	2
8	{c, d}	1
9	{g, h}	3
10	{h, l}	3
11	{k, l}	1

Tổng cộng : 24

THUẬT TOÁN PRIM

Ví dụ: Tìm cây khung nhỏ nhất

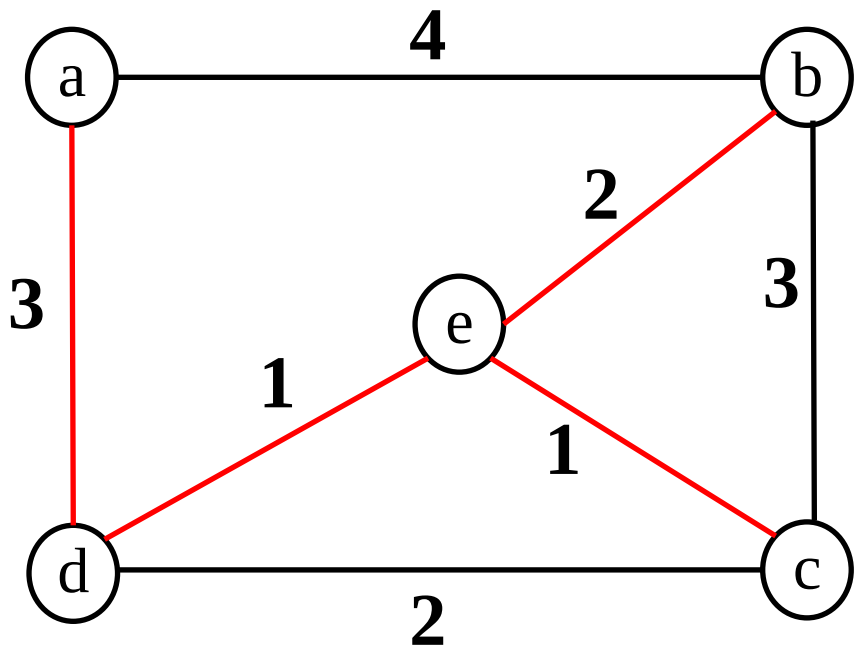


THUẬT TOÁN PRIM

u	S	*T*	A	B	C	D	E
E	E		∞	[2, E]	[1, E]*	[1, E]	-
C	E, C	{E, C}	∞	[2, E]	-	[1, E]*	-
D	E, C, D	{E, C}, {E, D}	[3, D]	[2, E]*	-	-	-
B	E, C, D, B	{E, C} ; {E, D}; {E, B}	[3, D]*	-	-	-	-
A	E, C, D, B, A	{E, C} ; {E, D}; {E, B} ; {D, A}	-	-	-	-	-

THUẬT TOÁN PRIM

- Cây khung nhỏ nhất gồm các cạnh: {E, C} ; {E, D}; {E, B} ; {D, A}
- Tổng trọng số nhỏ nhất của cây khung là: **7**



THUẬT TOÁN KRUSKAL

- Do Joseph Kruskal đưa ra năm 1956

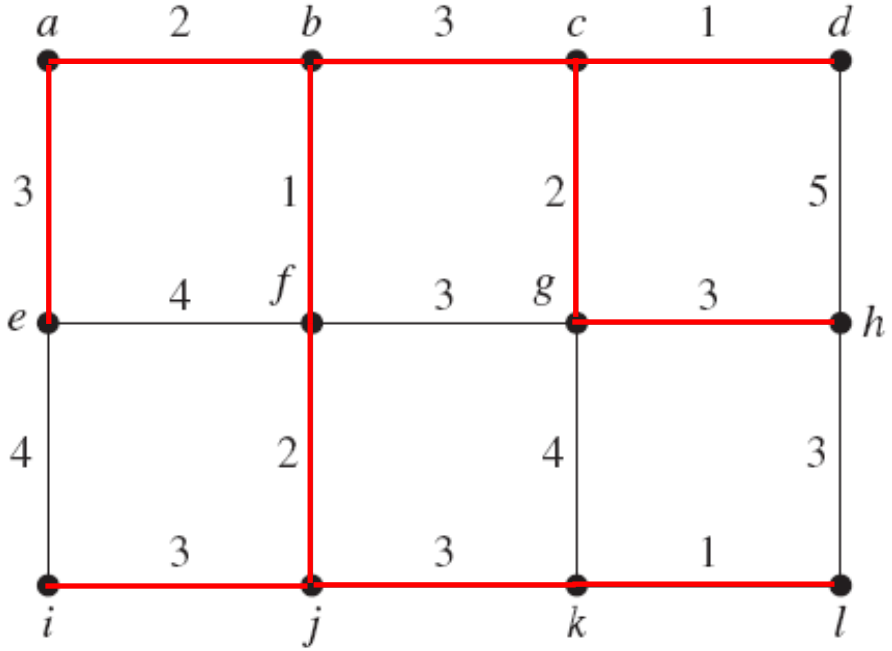
- **Thuật toán**:
  - ✓ ***Chọn một cạnh*** có trọng số nhỏ nhất, đặt vào khung
  - ✓ ***Ghép vào cây*** các cạnh có trọng số tối thiểu, không tạo ra chu trình
  - ✓ ***Dừng*** khi có (n-1) cạnh được ghép vào cây

THUẬT TOÁN KRUSKAL

THUẬT TOÁN : Thuật toán Kruskal

Procedure *Kruskal*(*G:* đồ thị *n* đỉnh, liên thông, có trọng số)
T := đồ thị rỗng
for *i*:= 1 **to** *n*-1
begin
    e := cạnh bất kì của G với trọng số nhỏ nhất và không tạo ra chu trình trong *T* khi ghép vào *T*
    T := *T* với *e* được ghép vào
end
{ T là cây khung nhỏ nhất của G }

Ví dụ: Tìm cây khung nhỏ nhất

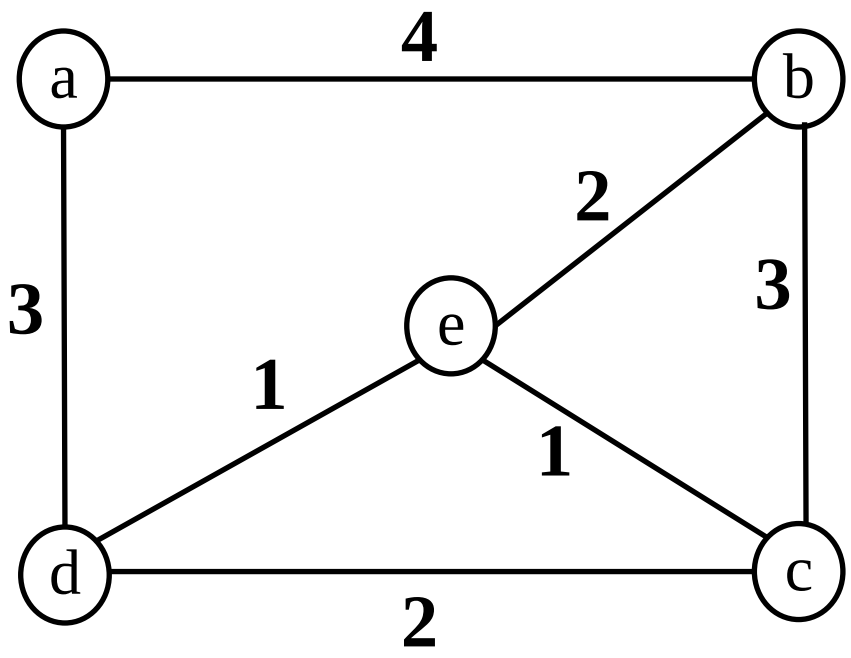


Lần chọn	Cạnh	Trọng số
1	{c, d}	1
2	{k, l}	1
3	{b, f}	1
4	{c, g}	2
5	{a, b}	2
6	{f, j}	2
7	{b, c}	3
8	{j, k}	3
9	{g, h}	3
10	{i, j}	3
11	{a, e}	3

Tổng cộng : 24

THUẬT TOÁN PRIM

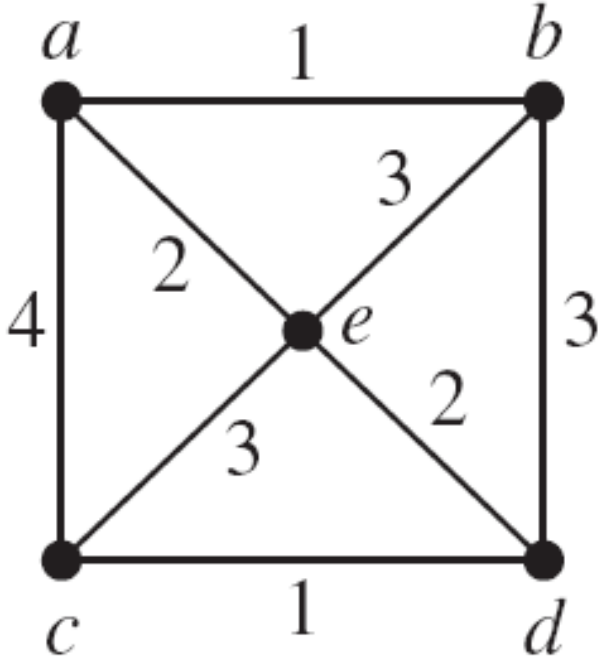
Ví dụ: Tìm cây khung nhỏ nhất



BÀI TẬP

- **Bài 2**: Tìm cây khung nhỏ nhất dùng thuật toán **Prim** và **Kruskal**

2.



BÀI TẬP

- **Bài 2**: Tìm cây khung nhỏ nhất dùng thuật toán **Prim** và **Kruskal**

3.