Cấu trúc dữ liệu và giải thuật

CẤU TRÚC CÂY

Văn Chí Nam – Nguyễn Thị Hồng Nhung – Đặng Nguyễn Đức Tiến - Vũ Thanh Hưng

Nội dung trình bày

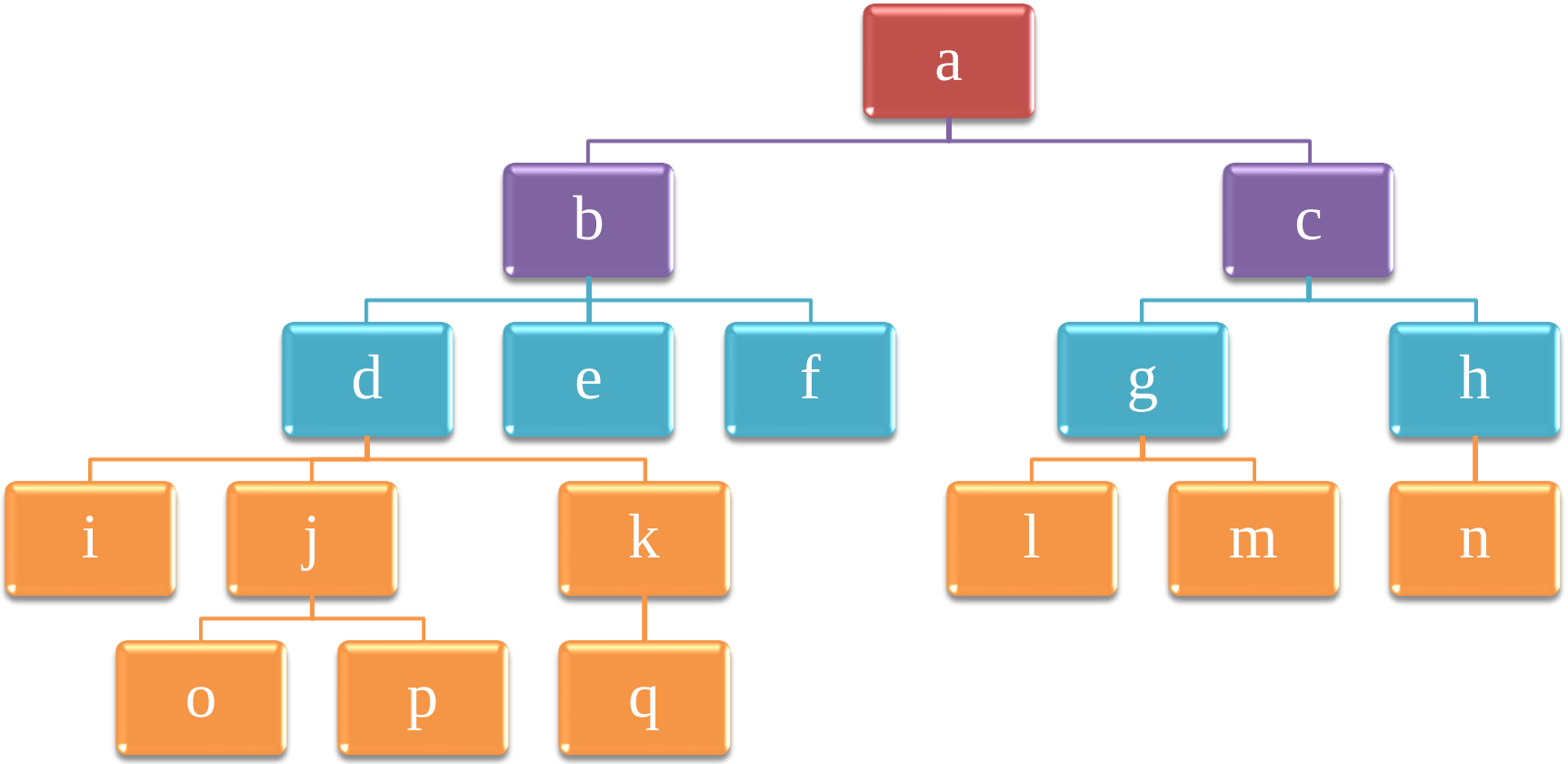
- Khái niệm
- Phép duyệt cây và Biểu diễn cây
- Cây nhị phân và Cây nhị phân tìm kiếm
- Cây AVL
- Cây AA

Khái niệm

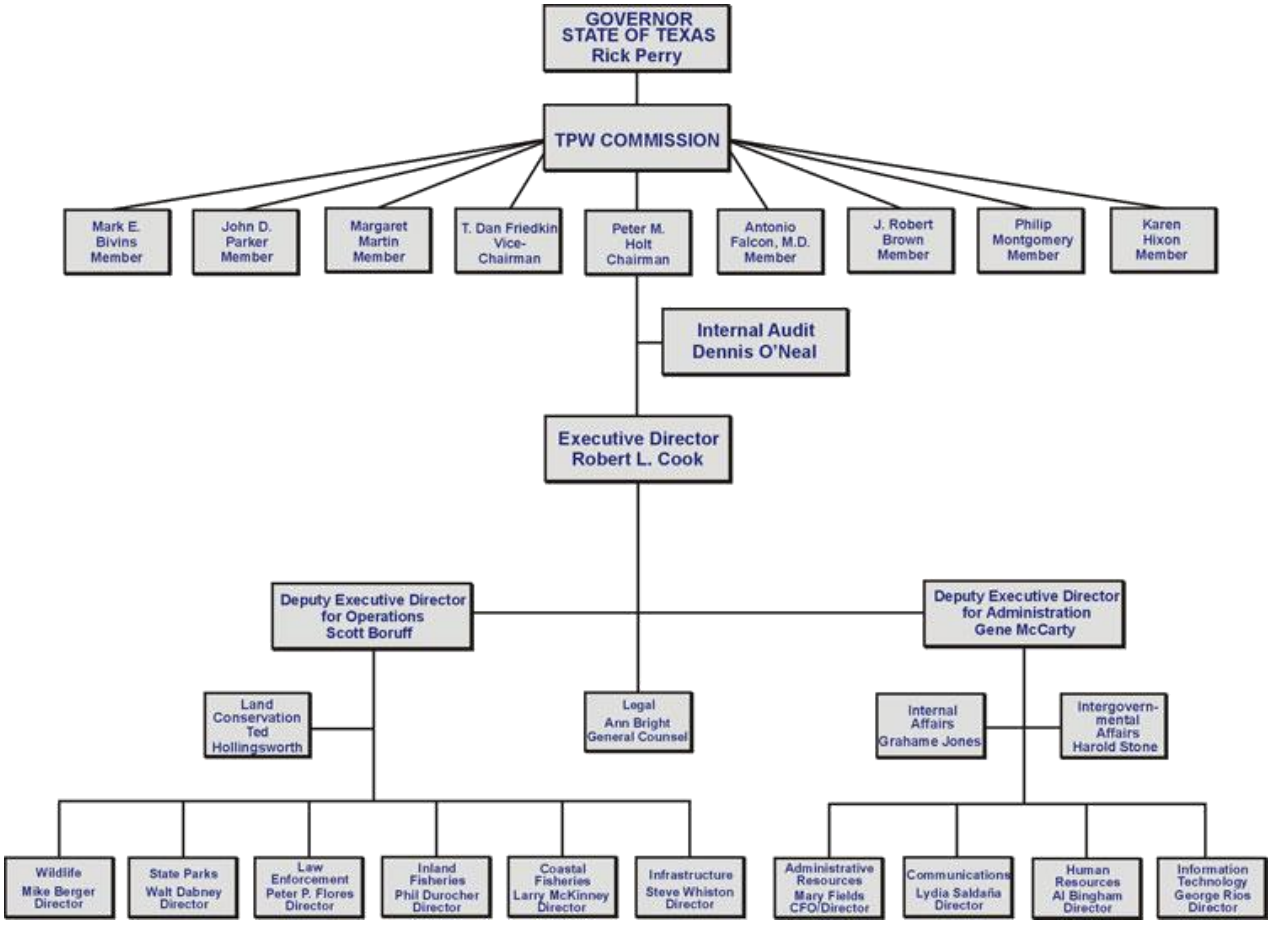
Một số thuật ngữ

- Tree
- Search tree (Cây tiềm kiếm)
- Binary search tree (Cây tìm kiếm nhị phân)
- Balanced tree (Cây cân bằng)
- AVL tree (Cây AVL)
- AA tree (Cây AA)
- Red-Black tree (Cây đỏ đen)
- …

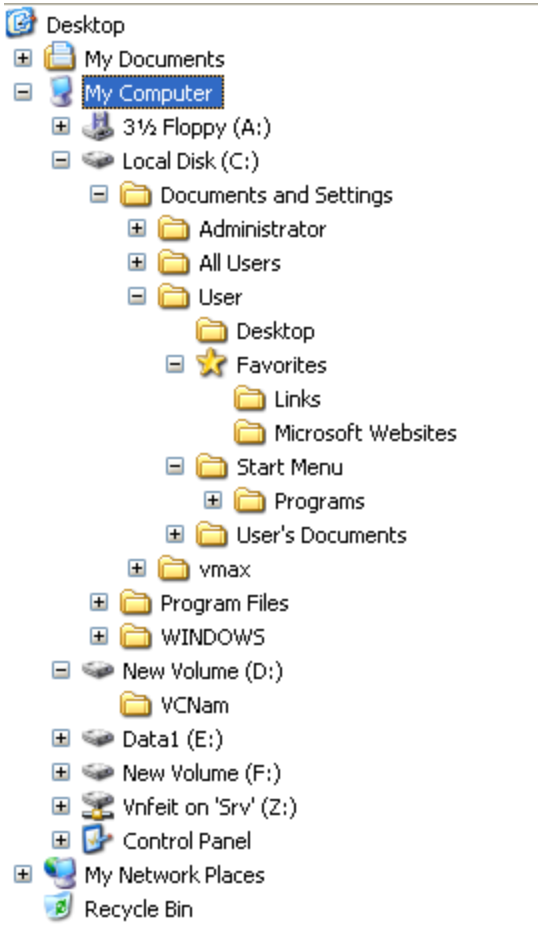
Cây tổng quát



Cây tổng quát



Sơ đồ tổ chức



Cây thư mục

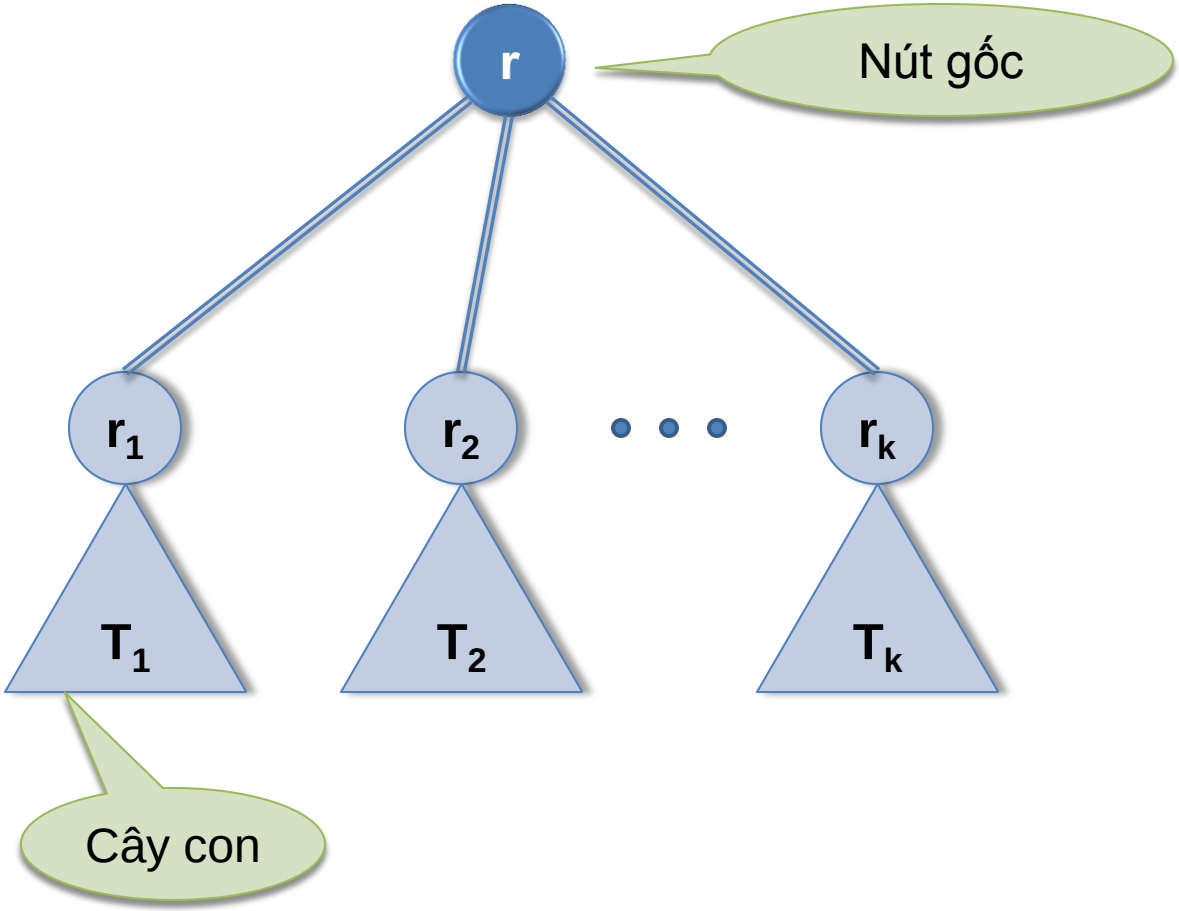
Định nghĩa

Cây (cây có gốc) được xác định đệ quy như sau:

1. Tập hợp gồm 1 **đỉnh** là một cây. Cây này có **gốc** là đỉnh duy nhất của nó.
2. Gọi $T_1, T_2, \ldots T_k$ ($k \geq 1$) là các cây không cắt nhau có gốc tương ứng $r_1, r_2, \ldots r_k$.

Giả sử ***r*** là một đỉnh mới không thuộc các cây T_i. Khi đó, tập hợp T gồm đỉnh ***r*** và các cây T_i tạo thành một cây mới với gốc ***r***. Các cây $T_1, T_2, \ldots T_k$ được gọi là cây con của gốc r.

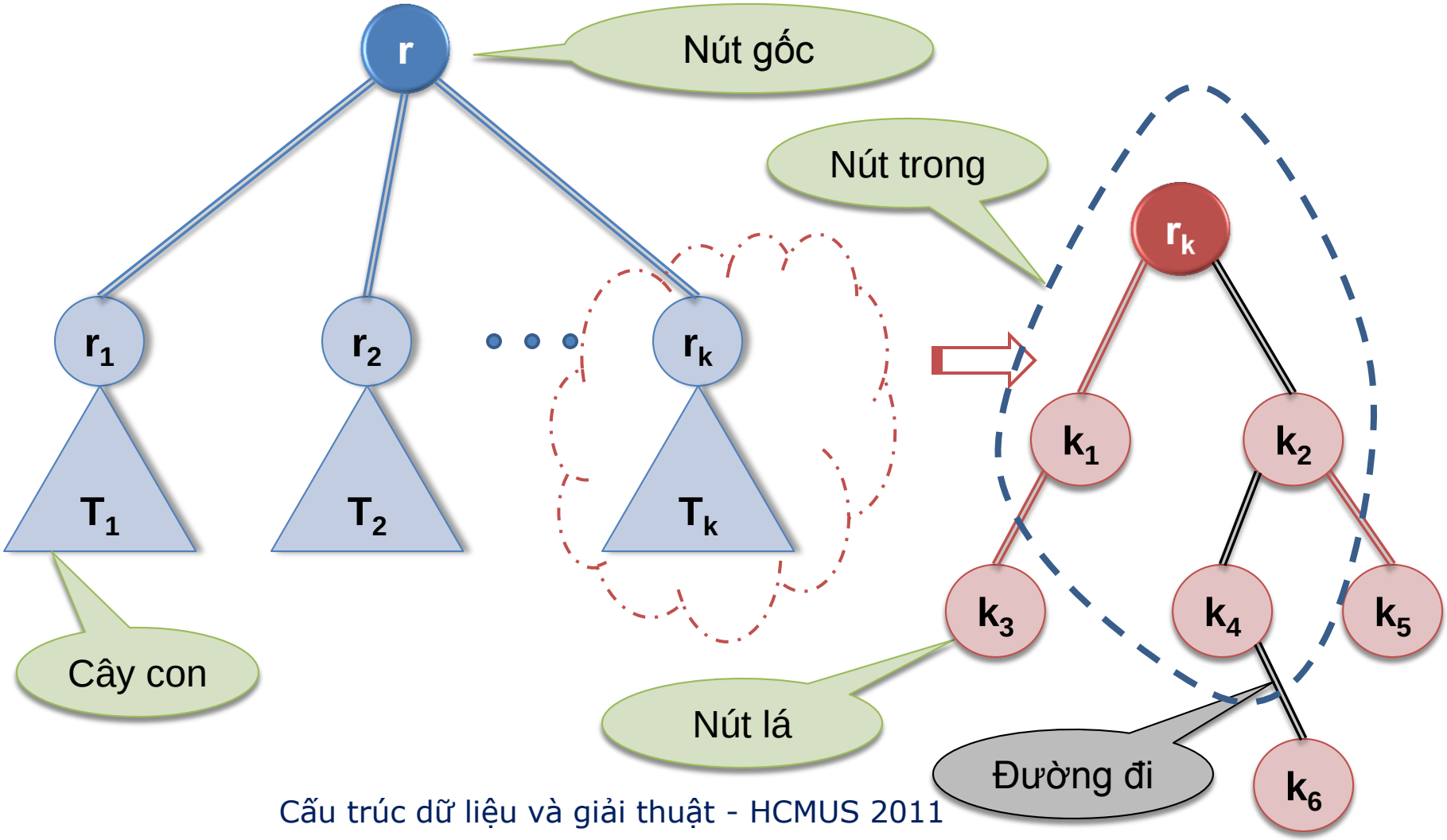
Định nghĩa



Các khái niệm

- node: đỉnh
- root: gốc cây
- leaf: lá
- inner node/internal node: đỉnh trong
- parent: đỉnh cha
- child: đỉnh con
- path: đường đi

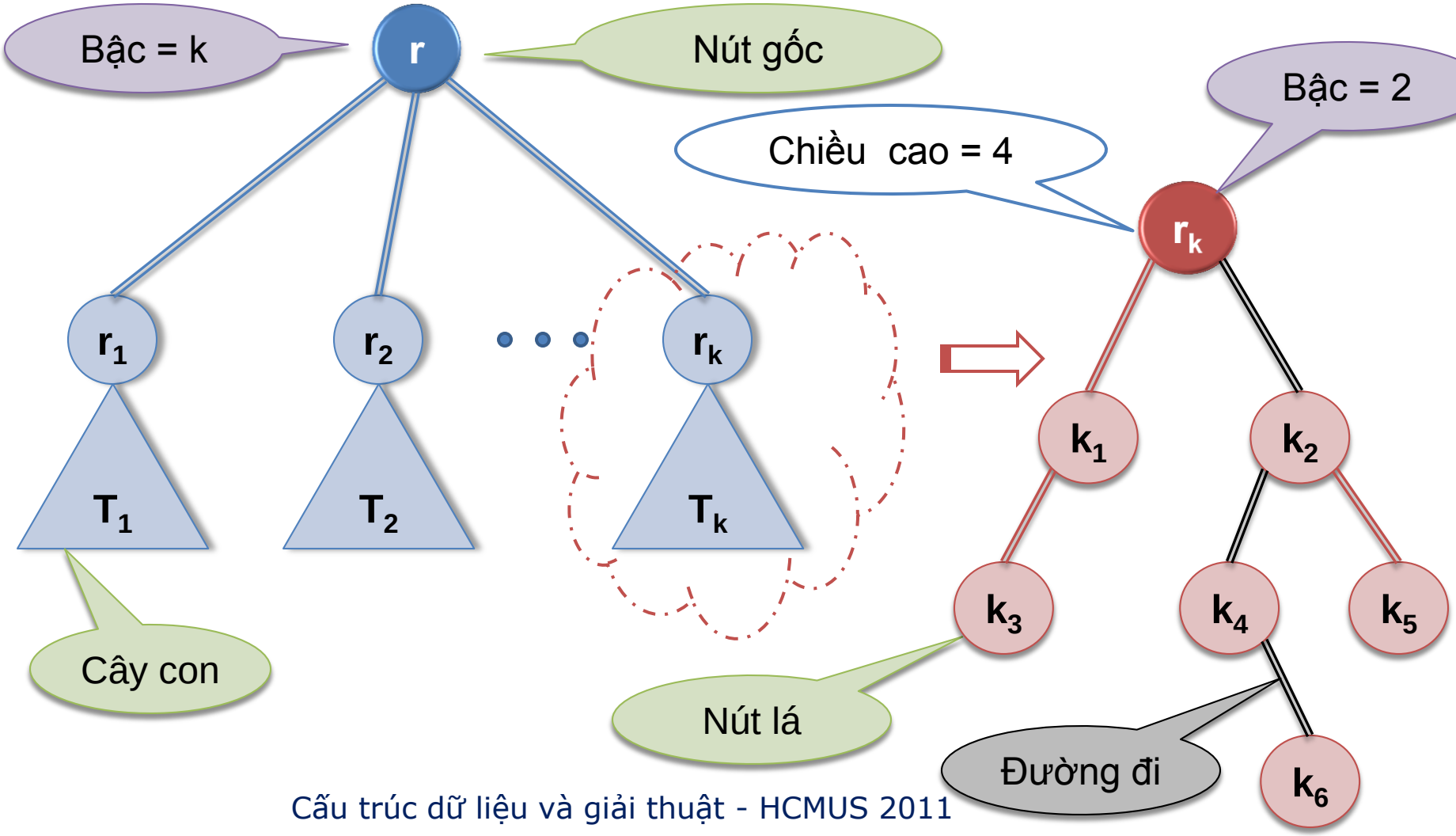
Các khái niệm



Các khái niệm

- degree/order: bậc
  - Bậc của node: Số con của node
  - Bậc của cây: bậc lớn nhất trong số các con
- depth/level: độ sâu/mức
  - Mức (độ sâu)của node: Chiều dài của đường đi từ node gốc đến node đó cộng thêm 1.
- height: chiều cao
  - Chiều cao cây:
    - Cây rỗng: 0
    - Cây khác rỗng: Mức lớn nhất giữa các node của cây

Các khái niệm



Phép duyệt cây

Phép duyệt cây

- Đảm bảo đến mỗi node trên cây **chính xác một lần** một cách **có hệ thống**.
- Nhiều thao tác xử lý trên cây cần phải sử dụng đến phép duyệt cây.
- Các phép cơ bản:
  - Duyệt trước (Pre-order)
  - Duyệt giữa (In-order)
  - Duyệt sau (Post-order)

Phép duyệt cây

Tìm cha một đỉnh.

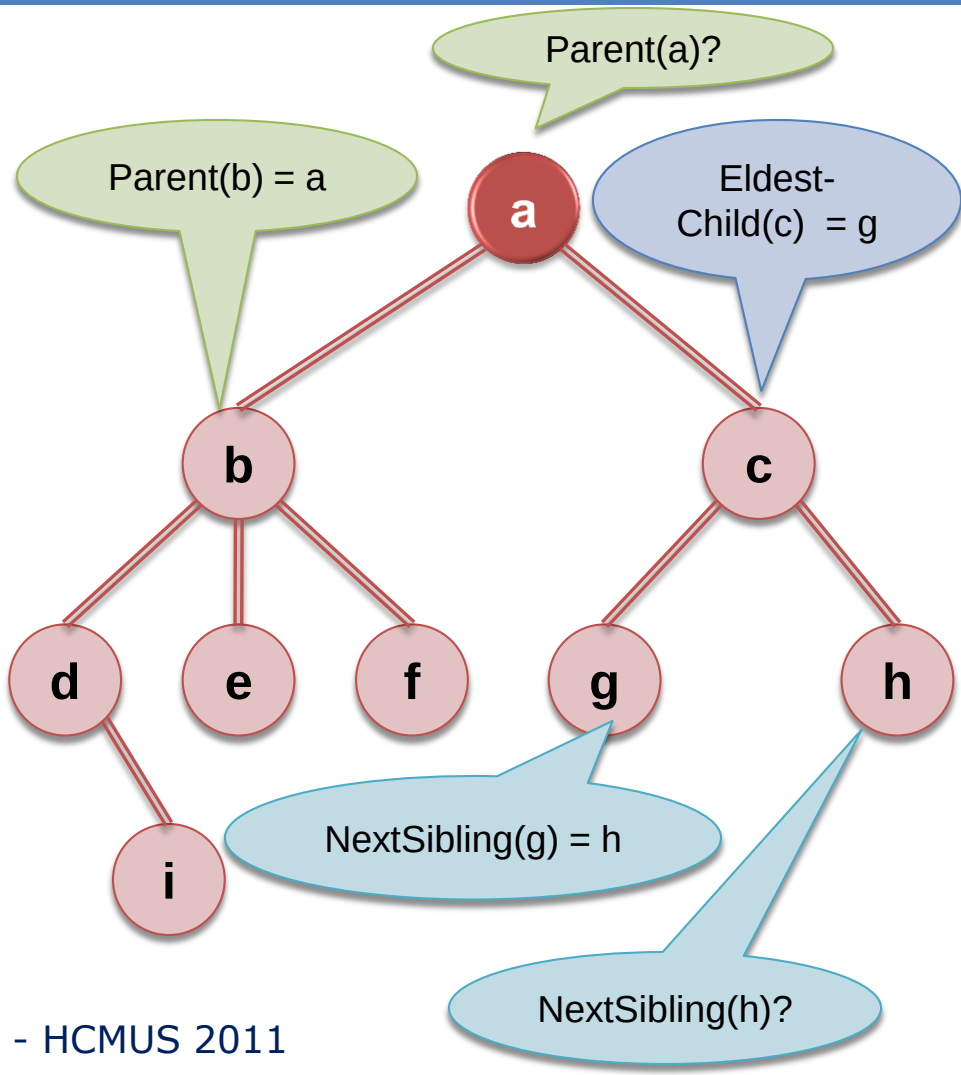
- *Parent(x)*

Tìm đỉnh con trái nhất.

- *EldestChild(x)*

Tìm đỉnh kề phải.

- *NextSibling(x)*



Phép duyệt cây

Duyệt theo chiều sâu

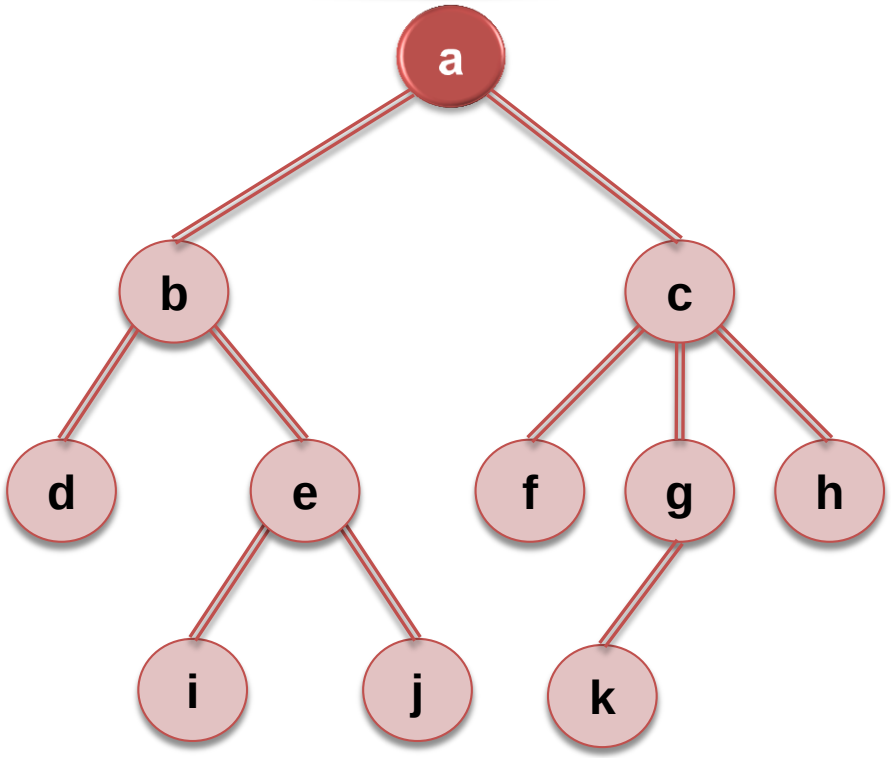
Duyệt trước

- *a b d e i j c f g k h*

Duyệt giữa

- d b i e j a f c k g h

Duyệt sau



Phép duyệt cây

Duyệt theo chiều sâu

Duyệt trước

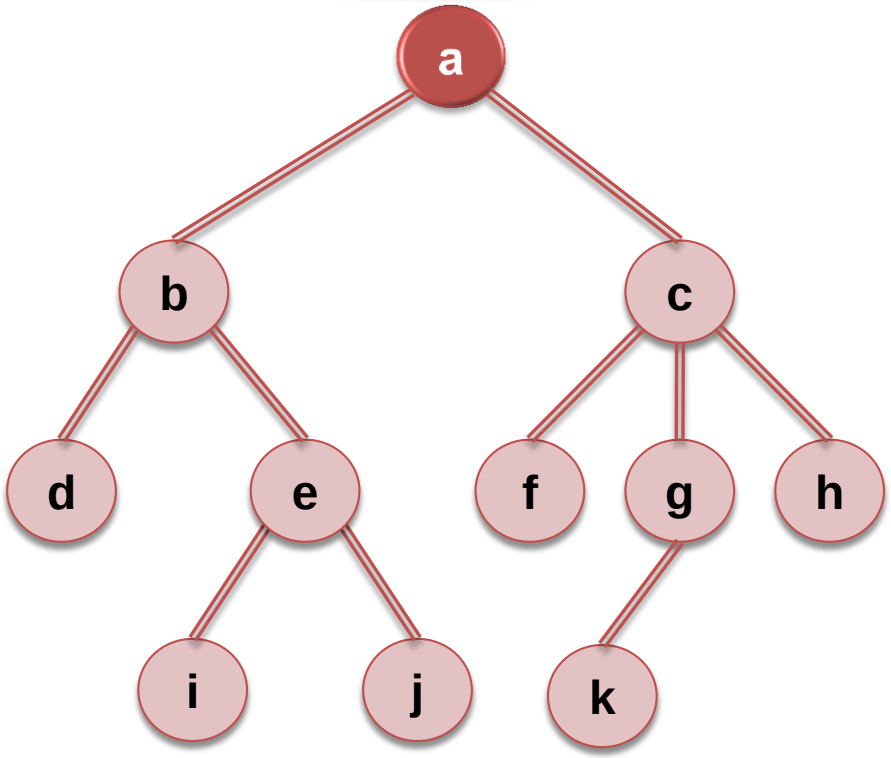
- *a b d e i j c f g k h*

Duyệt giữa

- d b i e j a f c k g h

Duyệt sau

- *d i j e b f k g h c a*



Phép duyệt cây

Pre-order

```
void Preorder(NODE A)
{
   NODE B;
   Visit(A);
   B = EldestChild(A);
   while (B != ∅) {
        Preorder(B);
        B = NextSibling(B);
   }
}
```

Post-order

```
void Postorder(NODE A)
{
   NODE B;

   B = EldestChild(A);
   while (B != ∅) {
        Postorder(B);
        B = NextSibling(B);
   }
   Visit(A);
}
```

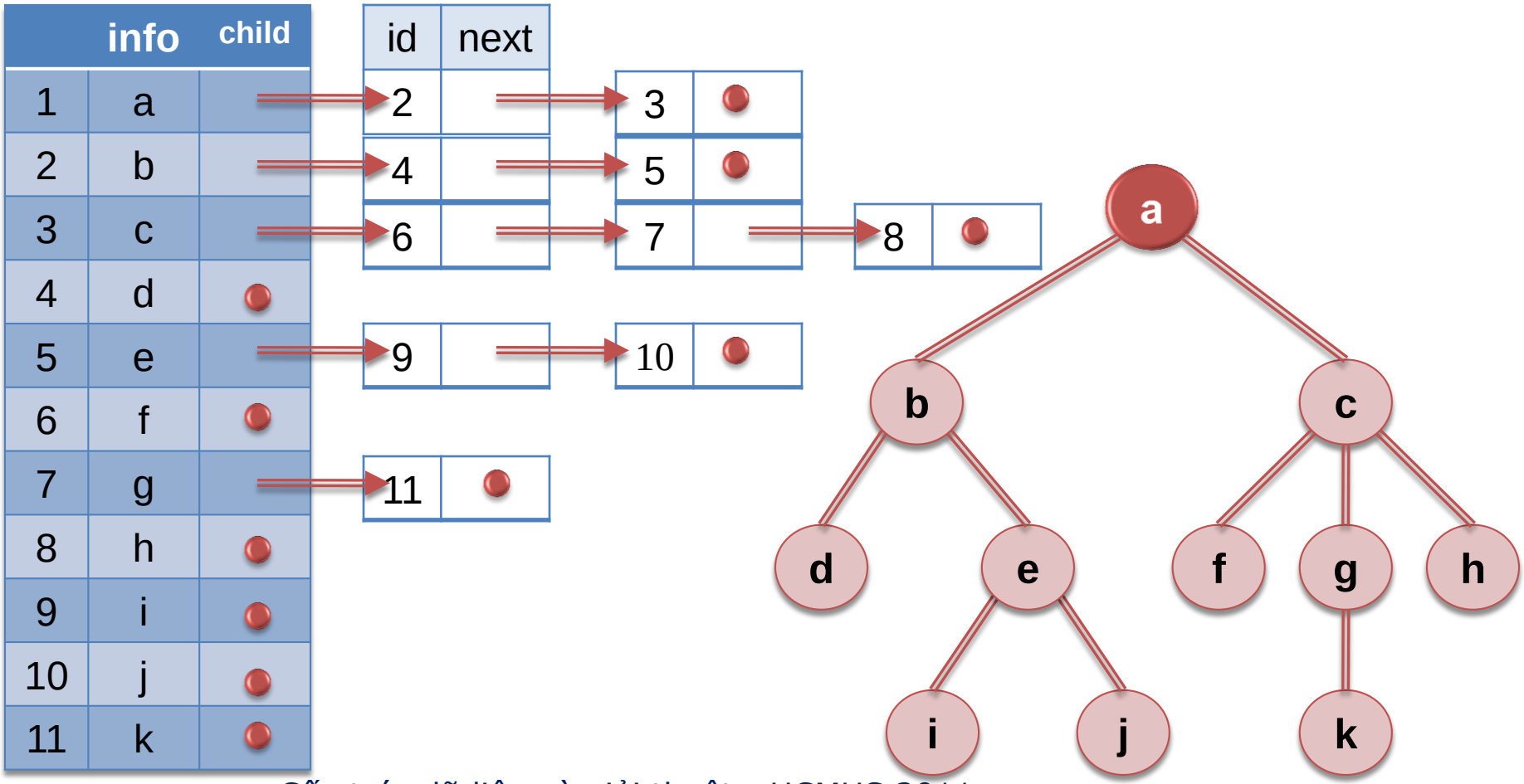
Phép duyệt cây

In-Order

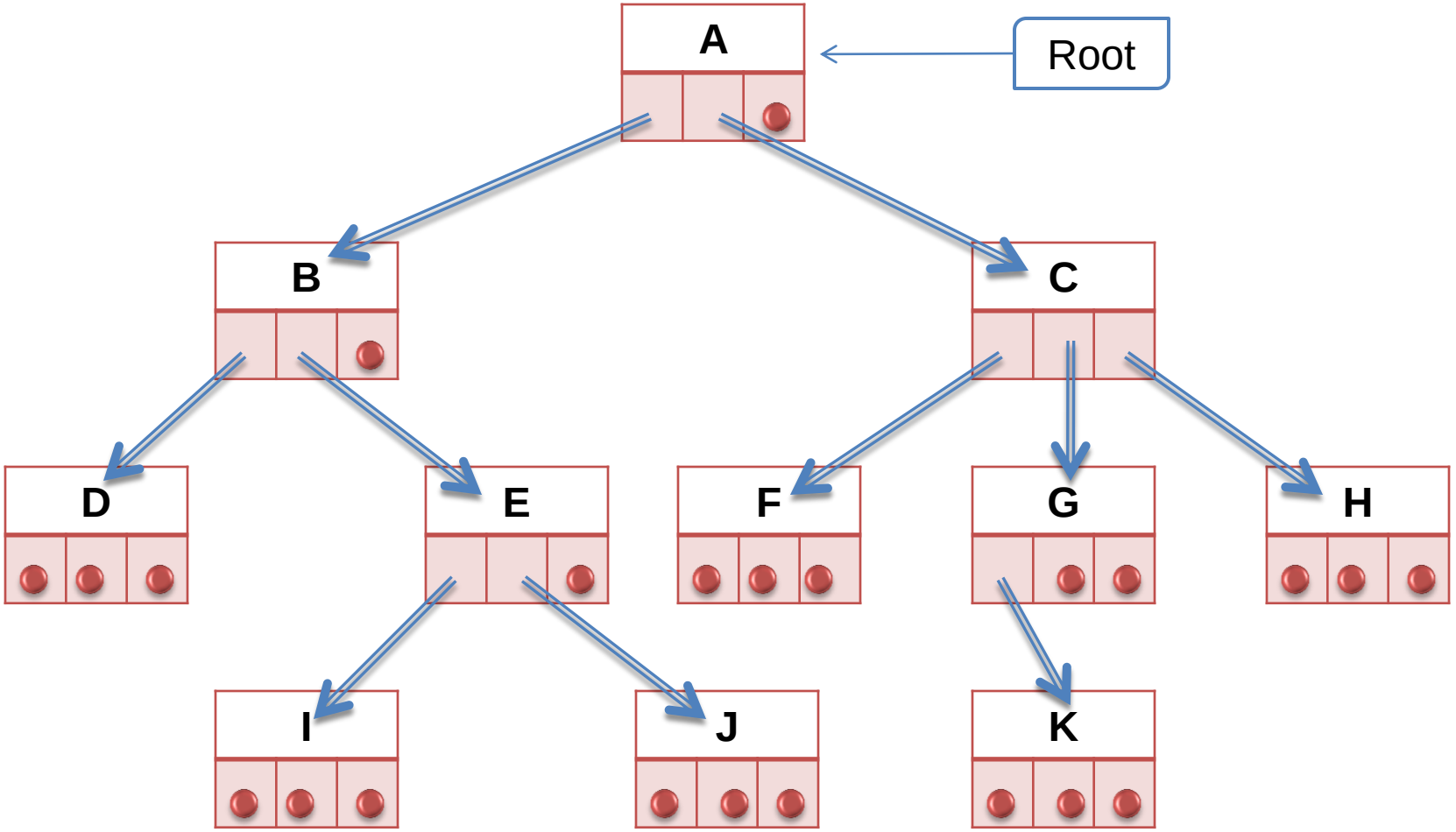
```
void Inorder(NODE A)
{
        NODE B;
        B = EldestChild(A);
        if (B != ∅) {
            Inorder(B);
            B = NextSibling(B);
        }
        Visit(A);
        while (B != ∅) {
            Inorder(B);
            B = NextSibling(B);
        }
}
```

Biểu diễn cây

Bằng danh sách cây con

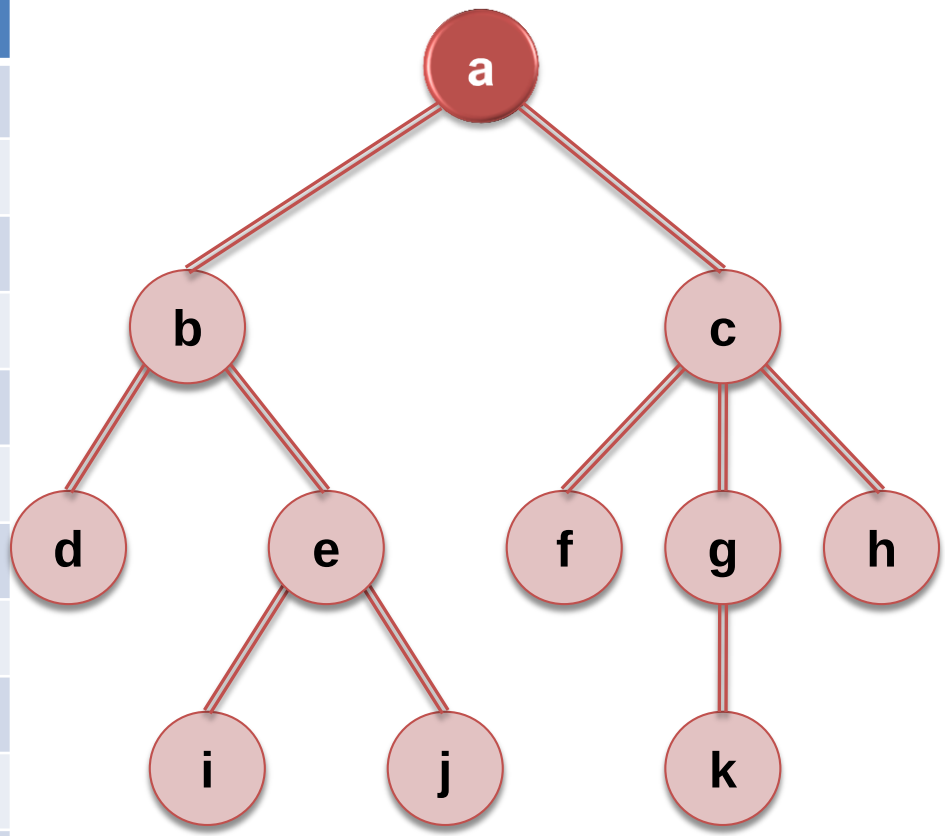


Bằng danh sách cây con

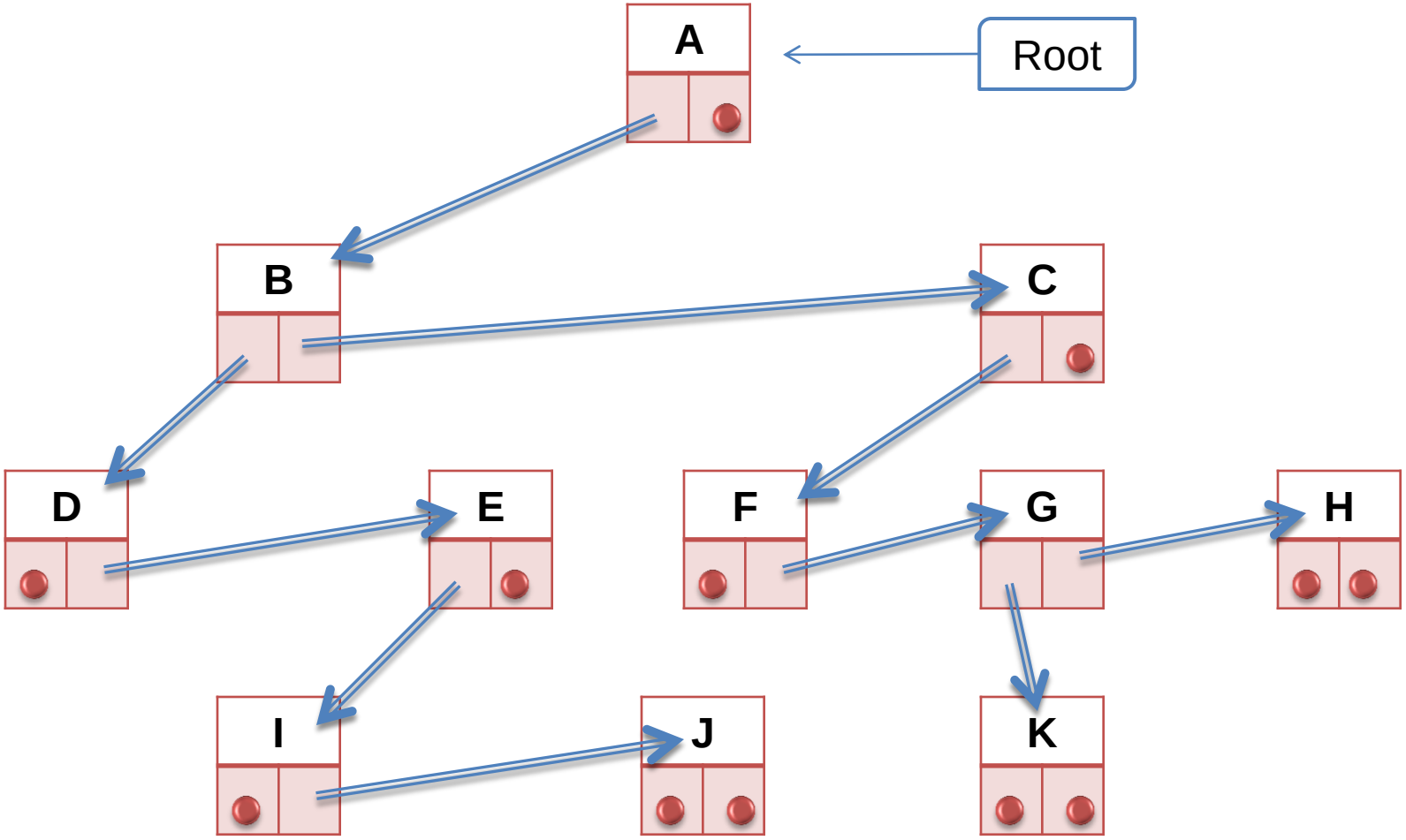


Bằng đỉnh trái nhất và đỉnh kề phải

	Info	Eldest Child	Next Sibling
1	a	2	0
2	b	4	3
3	c	6	0
4	d	0	5
5	e	9	0
6	f	0	7
7	g	11	8
8	h	0	0
9	i	0	10
10	j	0	0
11	k	0	0

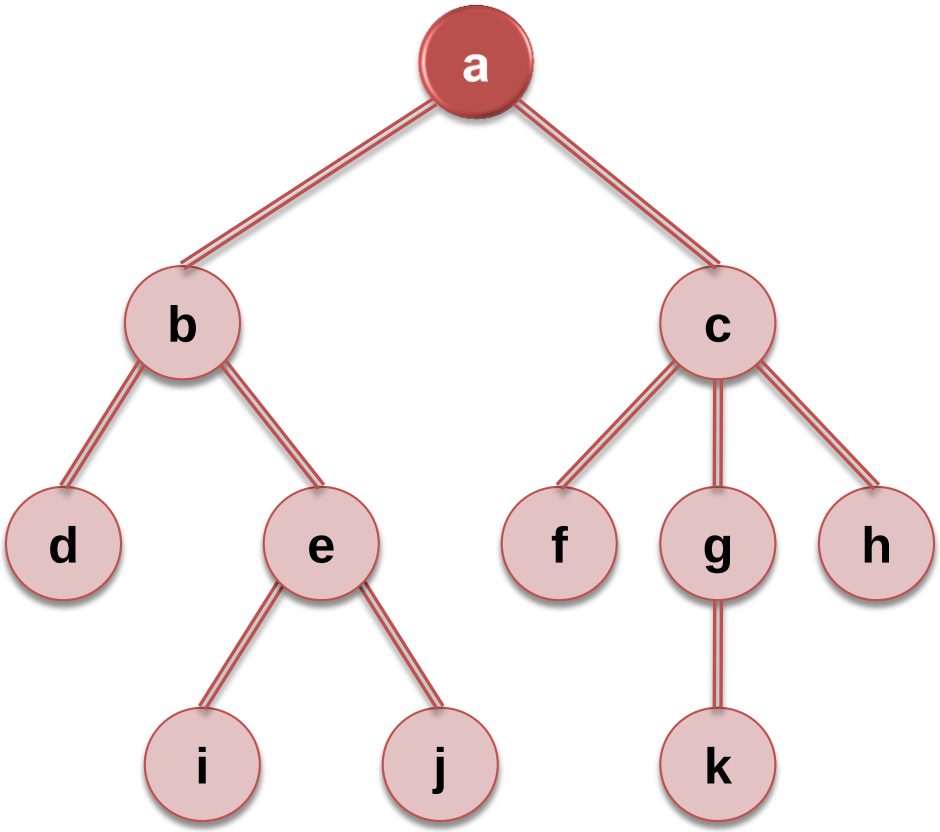


Bằng đỉnh trái nhất và đỉnh kề phải



Bằng cha mỗi đỉnh

	Info	Parent
1	a	0
2	b	1
3	c	1
4	d	2
5	e	2
6	f	3
7	g	3
8	h	3
9	i	5
10	j	5
11	k	7

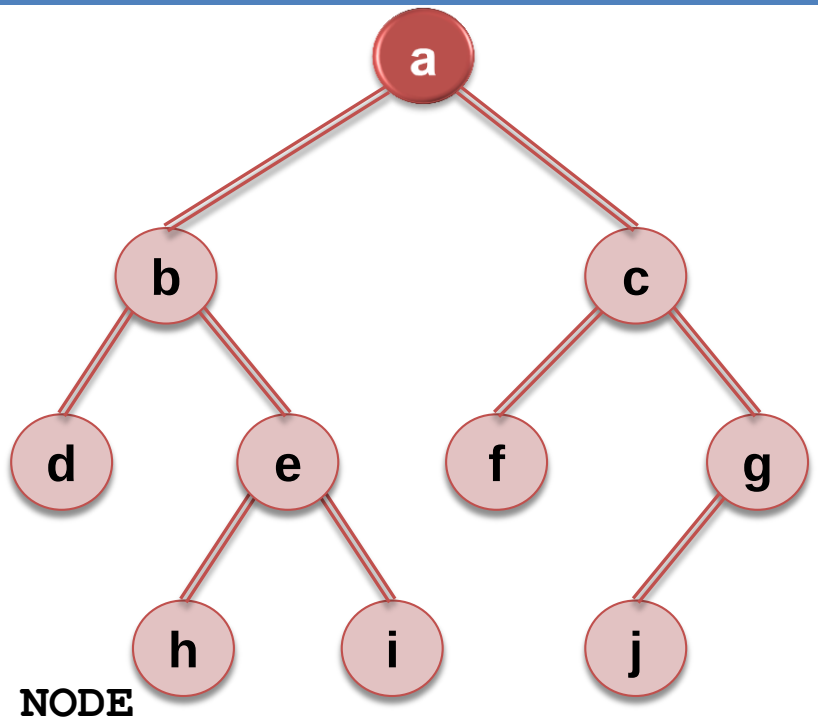


Cây nhị phân

Binary tree

Cây nhị phân

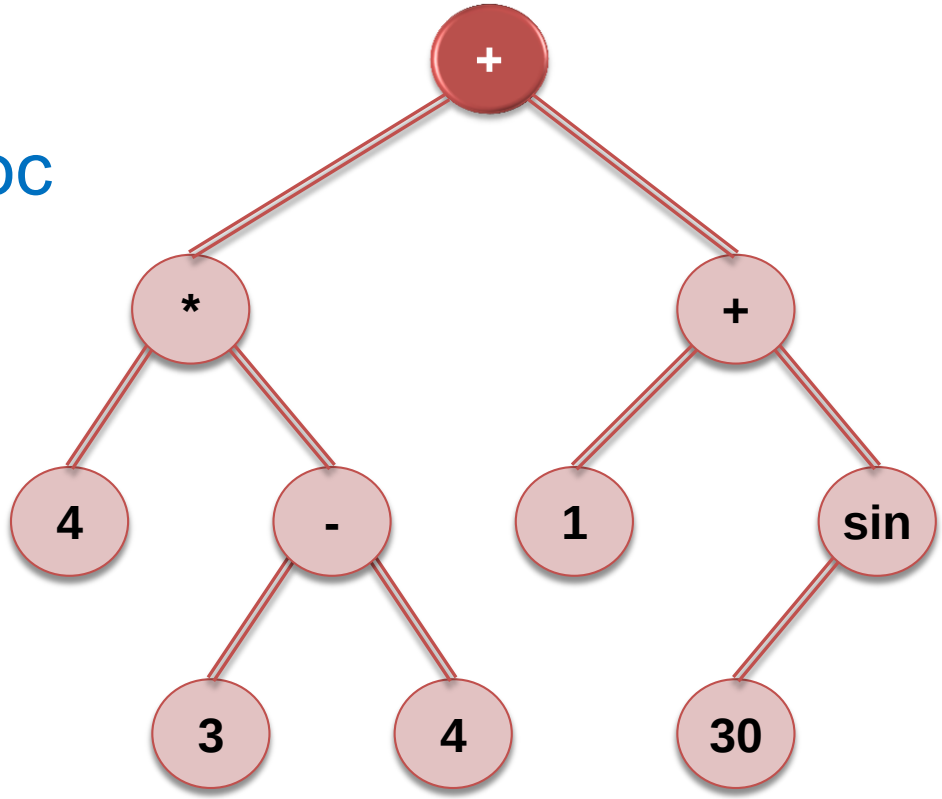
- Là cây mà mỗi đỉnh có bậc tối đa bằng 2.
- Các cây con được gọi là cây con trái và cây con phải.
- Có toàn bộ các thao tác cơ bản của cây.




```
struct NODE
{
        Data key;
        NODE *pLeft;
        NODE *pRight;
};
```

Một số ứng dụng

- Cây tổ chức thi đấu
- Cây biểu thức số học
- Lưu trữ và tìm kiếm thông tin.



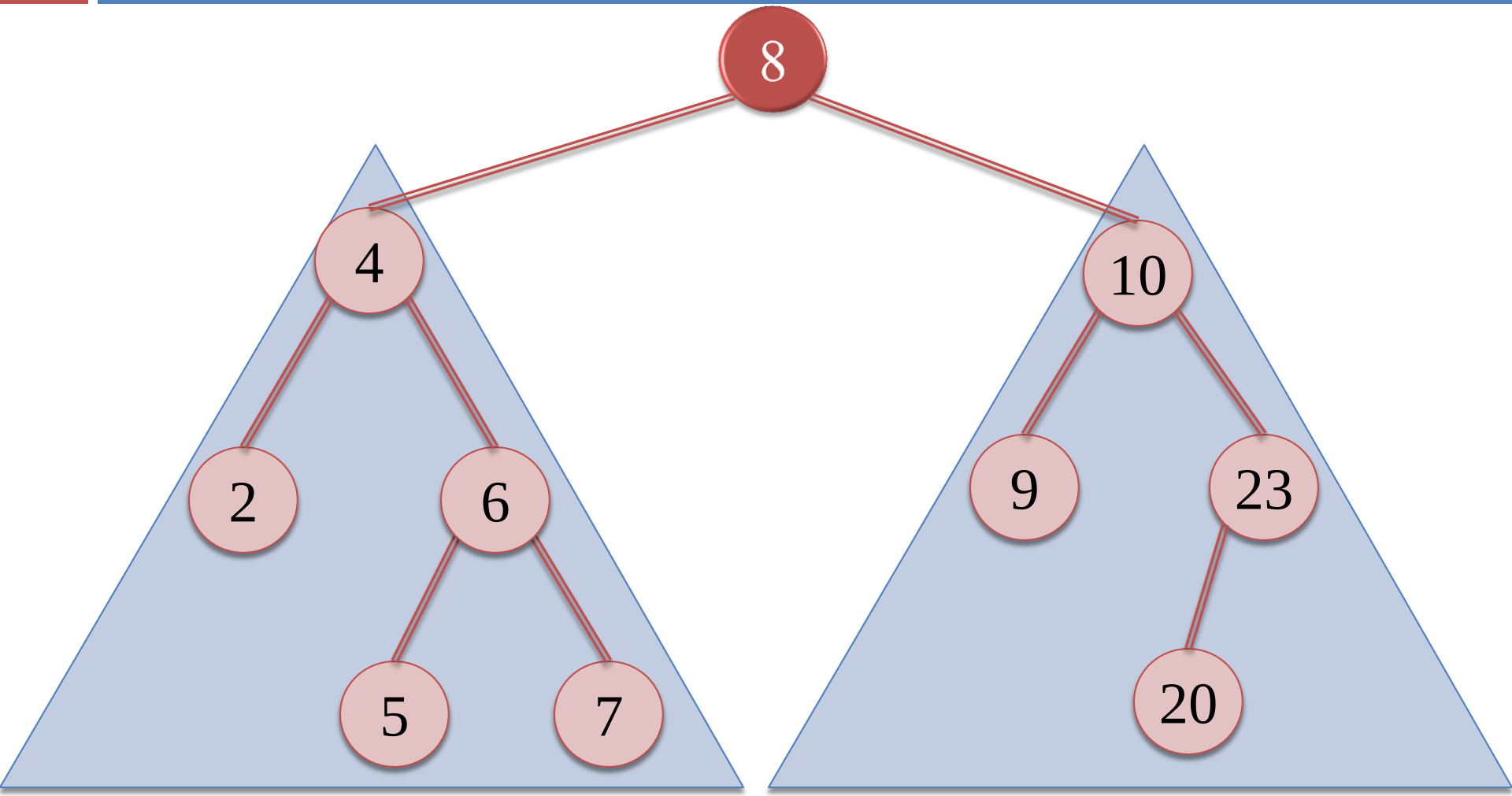
Cây biểu thức:
*4 * (3 – 4) + (1 + sin(30))*

Cây nhị phân tìm kiếm

- Cây nhị phân tìm kiếm là cây nhị phân thỏa mãn các điều kiện sau:

1. Khóa của các đỉnh thuộc cây con trái nhỏ hơn khóa gốc.
2. Khóa của gốc nhỏ hơn khóa các đỉnh thuộc cây con phải.
3. Cây con trái và cây con phải của gốc cũng là cây nhị phân tìm kiếm.

Cây nhị phân tìm kiếm



Cây nhị phân tìm kiếm

- Đặc điểm:
  - Có thứ tự
  - Không có phần tử trùng
  - Dễ dàng tạo dữ liệu sắp xếp, và tìm kiếm

Thao tác trên cây nhị phân tìm kiếm

Các thao tác

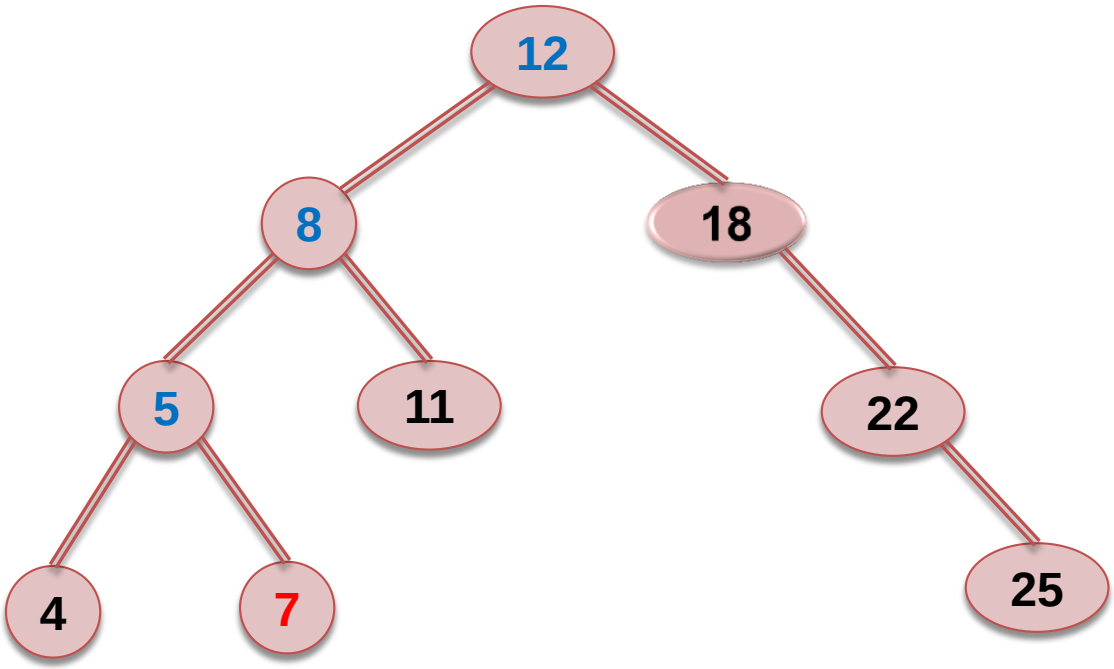
- Thêm phần tử (khóa)
- Tìm kiếm phần tử (khóa)
- Xóa phần tử (khóa)
- Sắp xếp
- Duyệt cây

Tìm kiếm phần tử

- Bước 1: Bắt đầu từ gốc
- Bước 2: So sánh dữ liệu (khóa) cần tìm với dữ liệu (khóa) của node hiện hành.
  - Nếu bằng nhau => Tìm thấy. Kết thúc
  - Nếu nhỏ hơn => Đi qua nhánh trái, Tiếp bước 2.
  - Nếu lớn hơn => Đi qua nhánh phải, Tiếp bước 2.
- Bước 3: Không thể đi tiếp nữa => Không tìm thấy. Kết thúc.

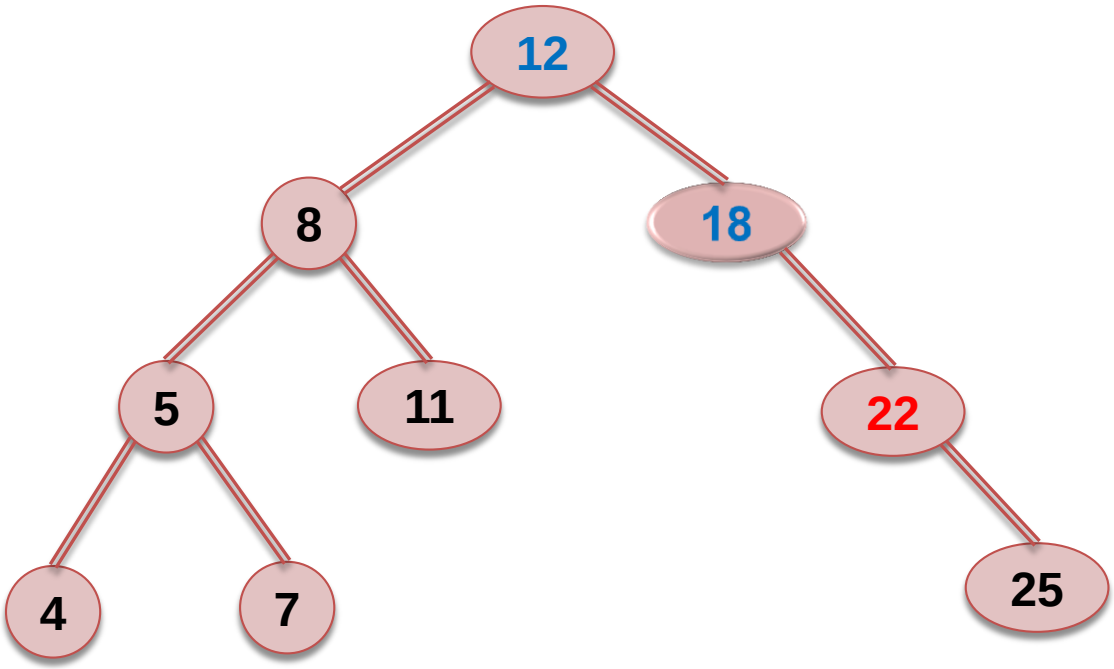
Tìm kiếm

- Tìm kiếm phần tử 7 trong cây:



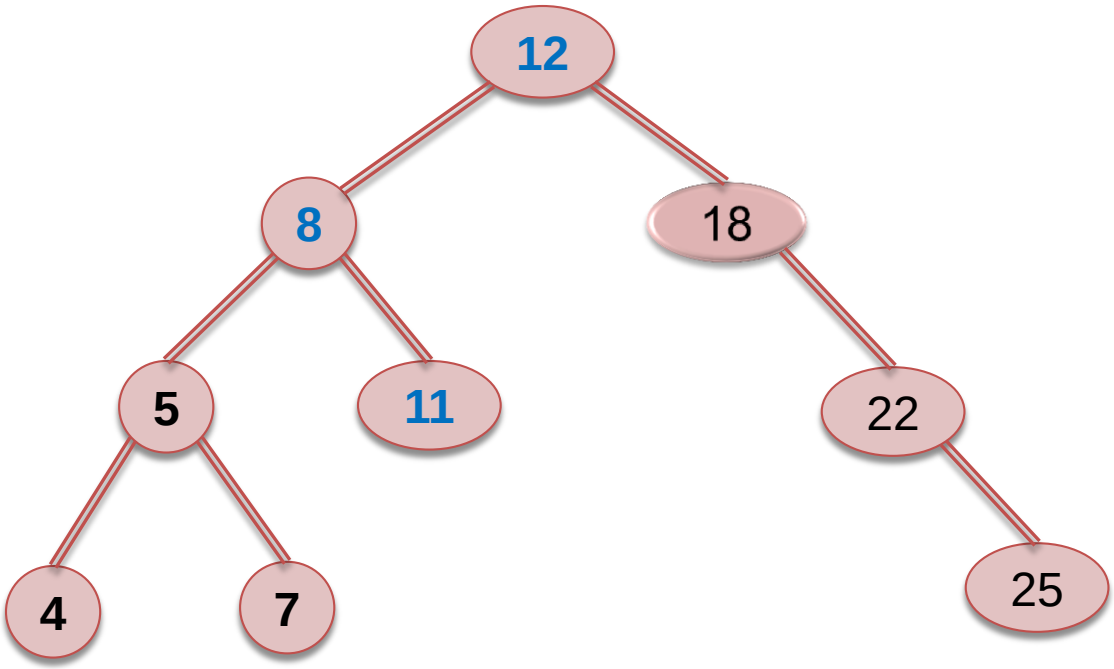
Tìm kiếm

- Tìm kiếm phần tử 22 trong cây:



Tìm kiếm

- Tìm kiếm phần tử 9 trong cây:

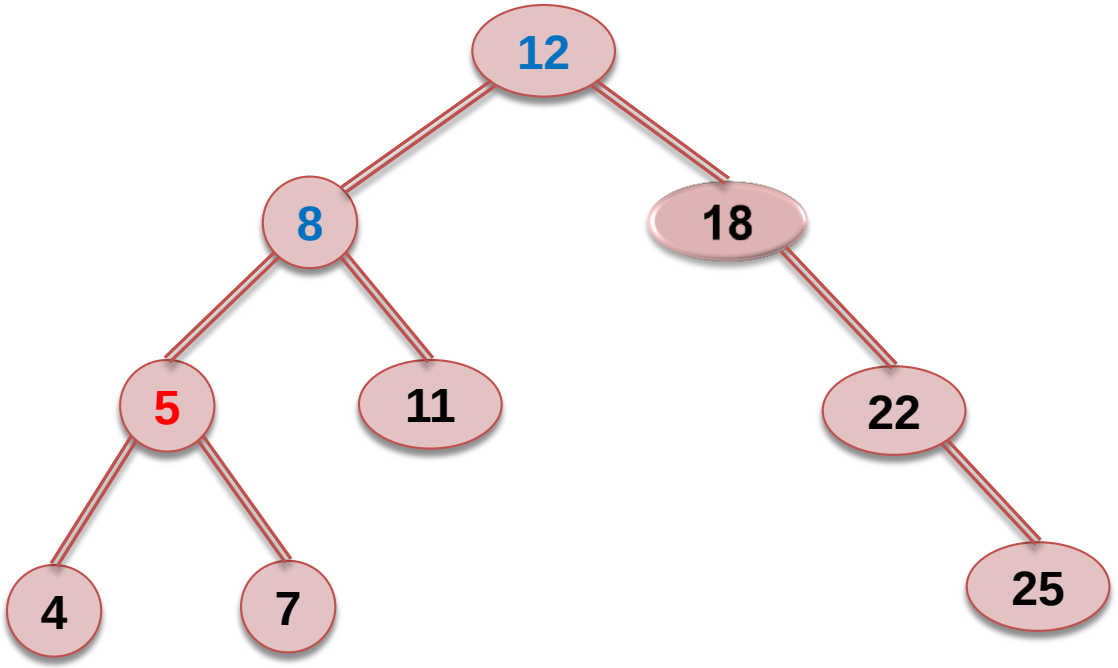


Thêm phần tử

- Bước 1: Bắt đầu từ gốc
- Bước 2: So sánh dữ liệu (khóa) cần thêm với dữ liệu (khóa) của node hiện hành.
  - Nếu bằng nhau => Đã tồn tại. Kết thúc
  - Nếu nhỏ hơn => Đi qua nhánh trái, Tiếp bước 2.
  - Nếu lớn hơn => Đi qua nhánh phải, Tiếp bước 2.
- Bước 3: Không thể đi tiếp nữa => Tạo node mới với dữ liệu (khóa) cần thêm. Kết thúc

Thêm

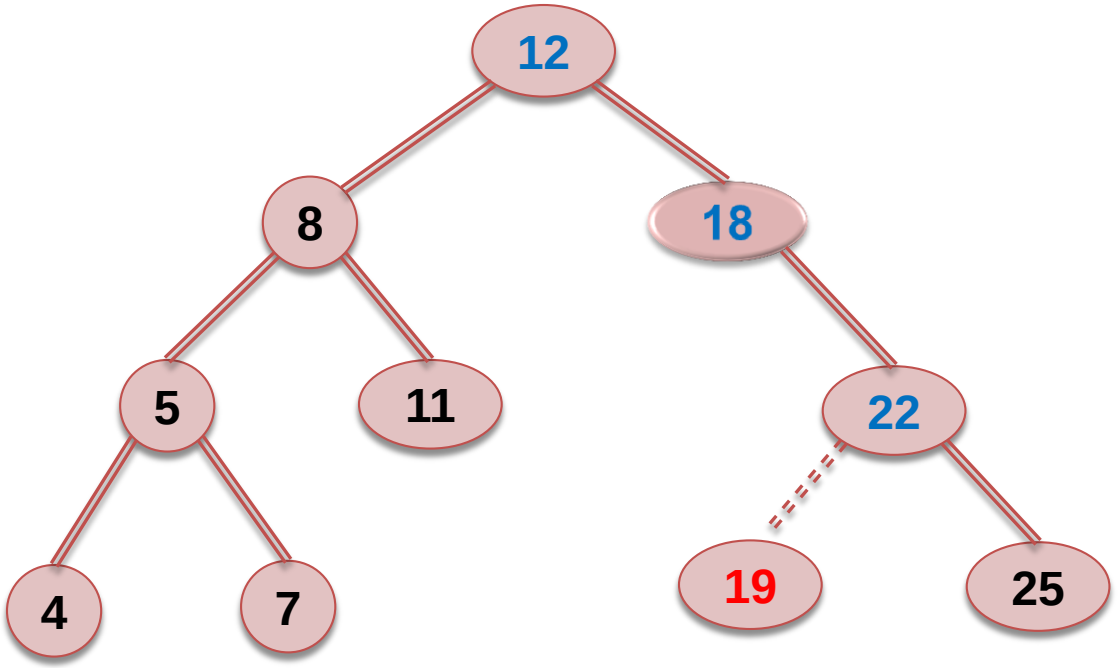
- Thêm phần tử 5 trong cây:



➔ **đã có, không cần thêm vào**

Thêm

- Thêm phần tử 19 trong cây:



Chưa có, thêm vào bên trái node 22

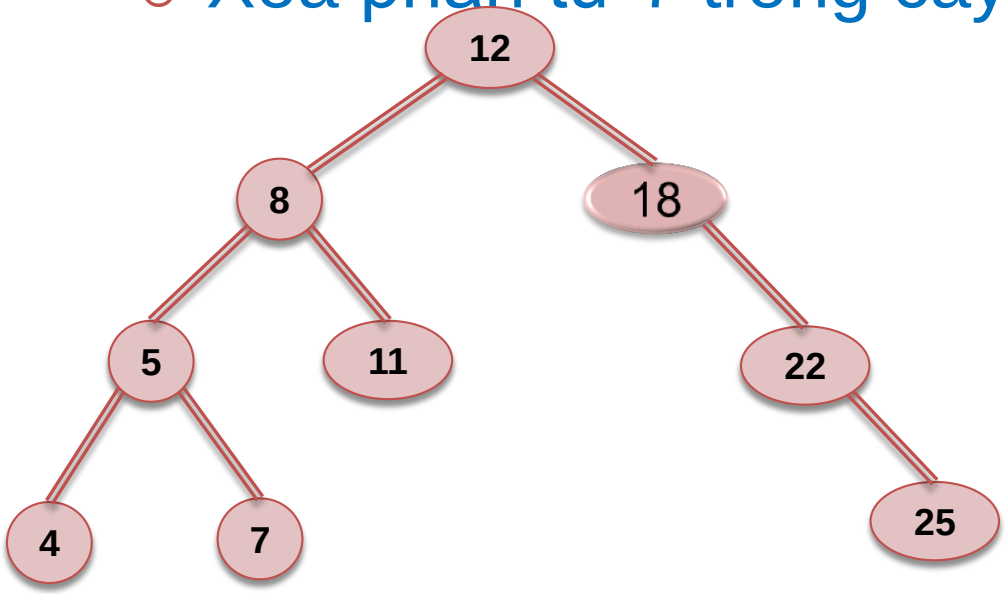
Xóa phần tử

- Tìm đến node chứa dữ liệu (khóa) cần xóa.

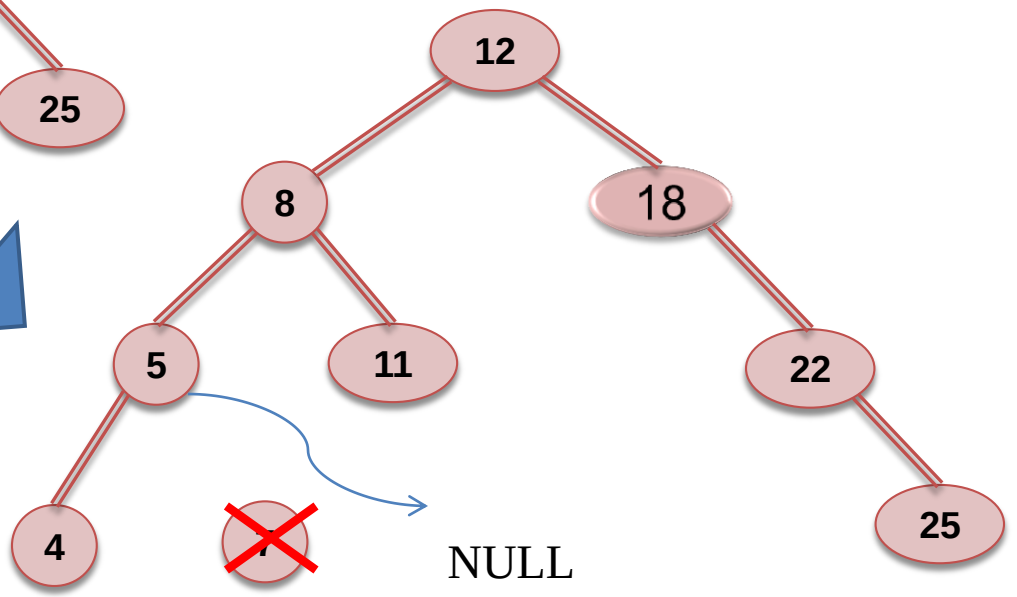
- Xét các trường hợp:
  - Node lá
  - Node chỉ có 1 con
  - Node có 2 con: dùng phần tử thế mạng để xóa thế.

Xóa node lá

- Xóa phần tử 7 trong cây

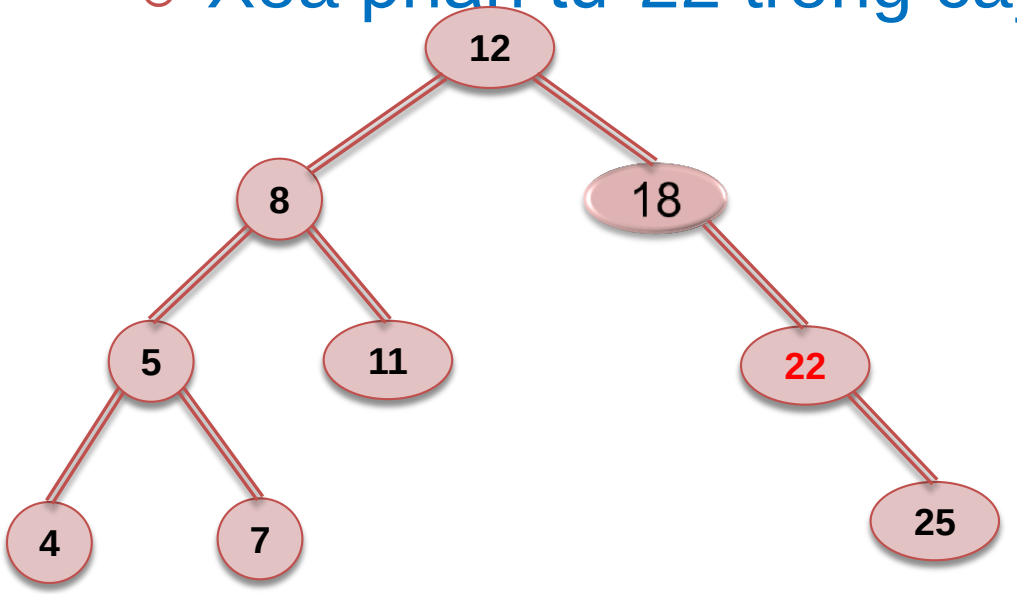


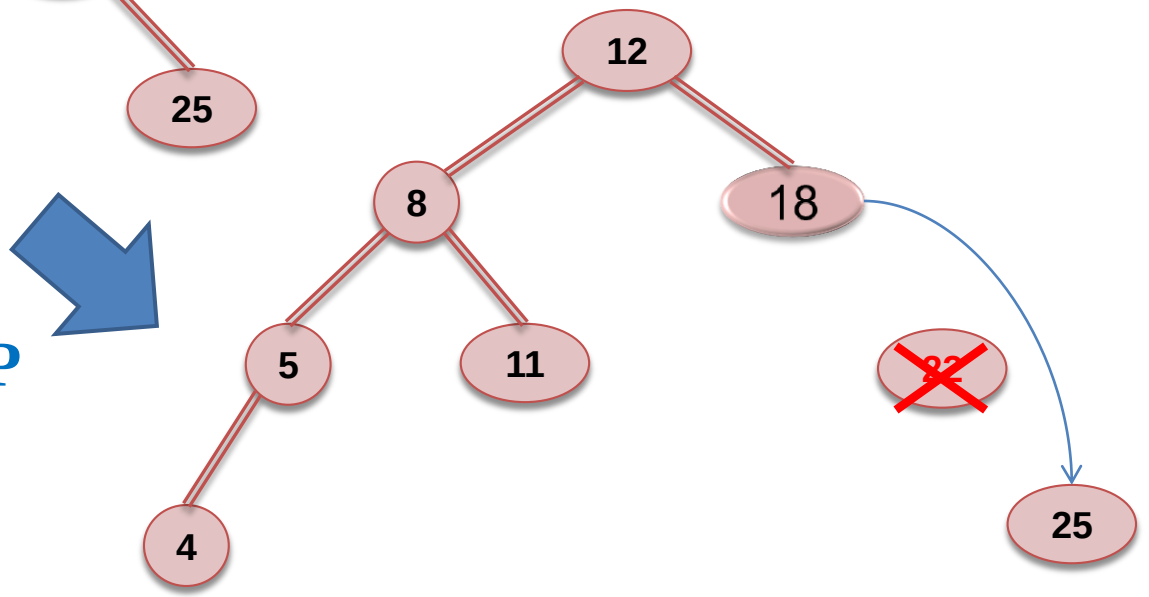
Xóa node lá và đặt con trỏ của node cha là NULL



Xóa node chỉ có 1 node (node con phải)

- Xóa phần tử 22 trong cây

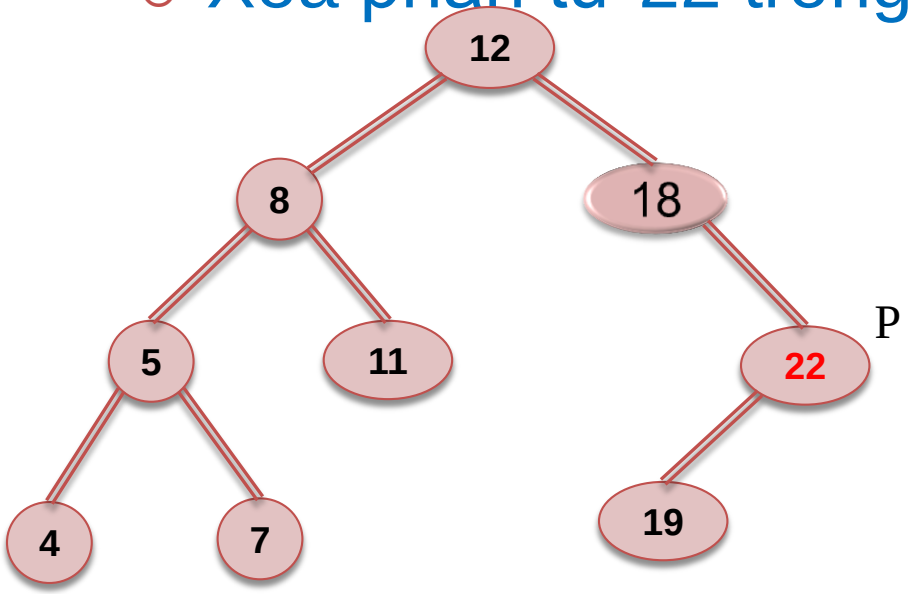


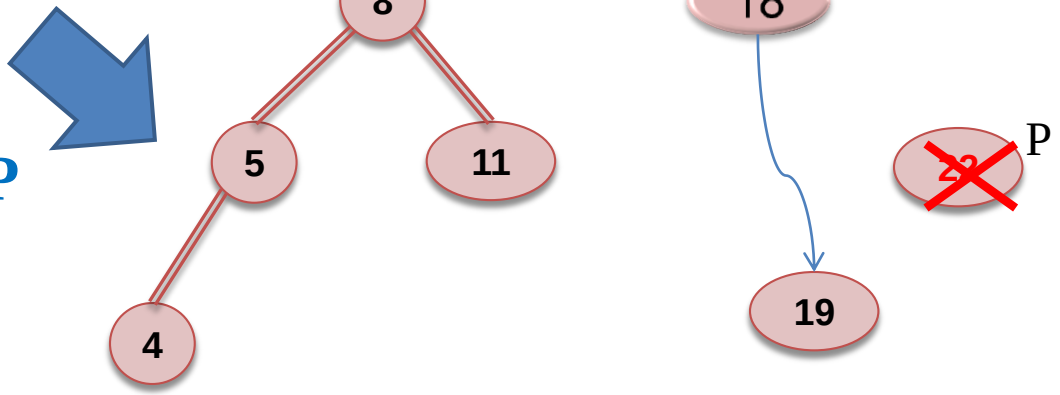


- Nối nhánh còn lại của P lên cha P
- Xóa node P

Xóa node chỉ có 1 node (node con trái)

- Xóa phần tử 22 trong cây

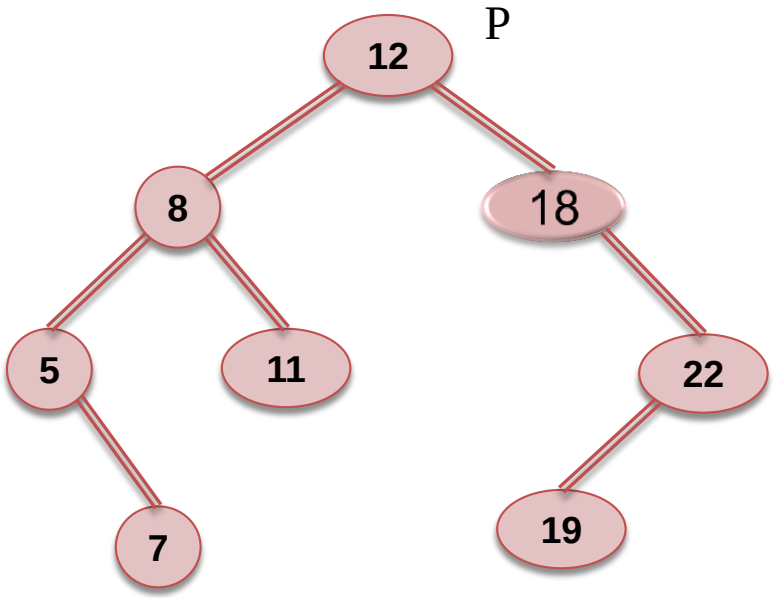


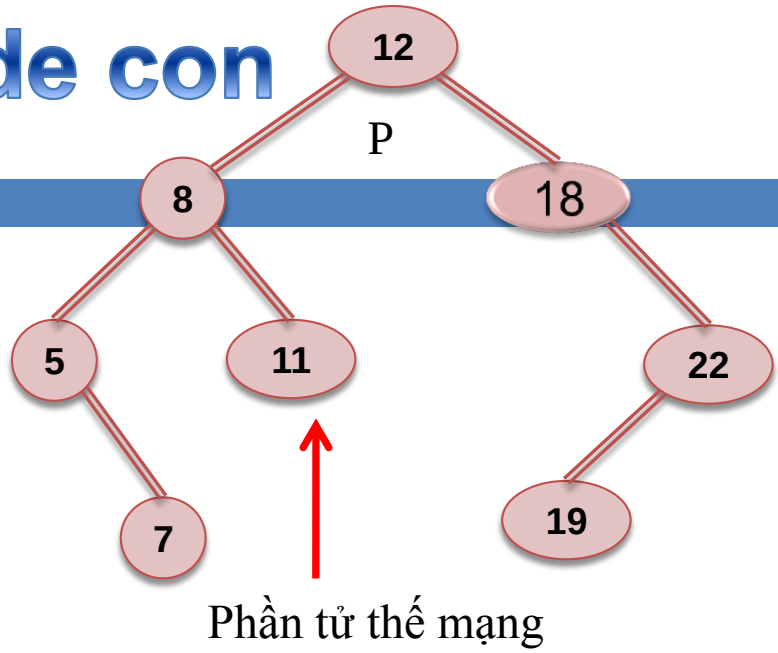


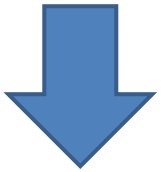
- Nối nhánh còn lại của P lên cha P
- Xóa node P

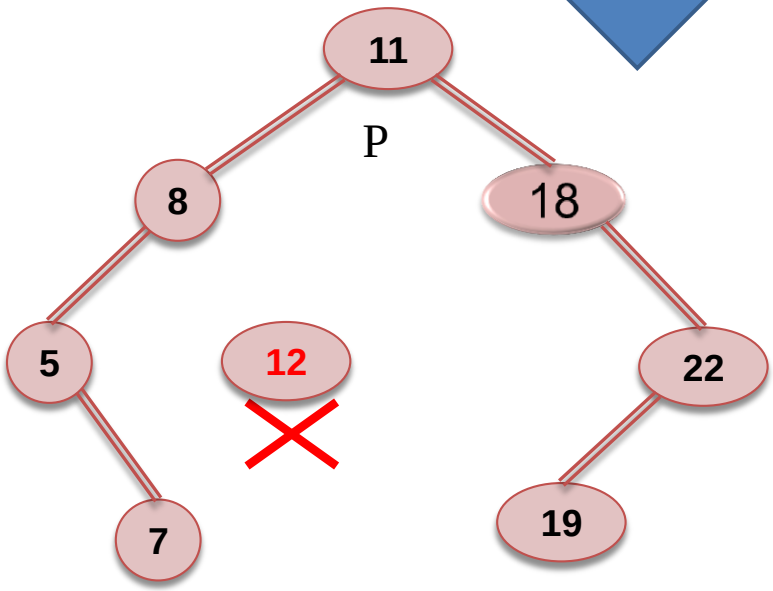
Xóa node chỉ có 2 node con

- Xóa phần tử 12 trong cây





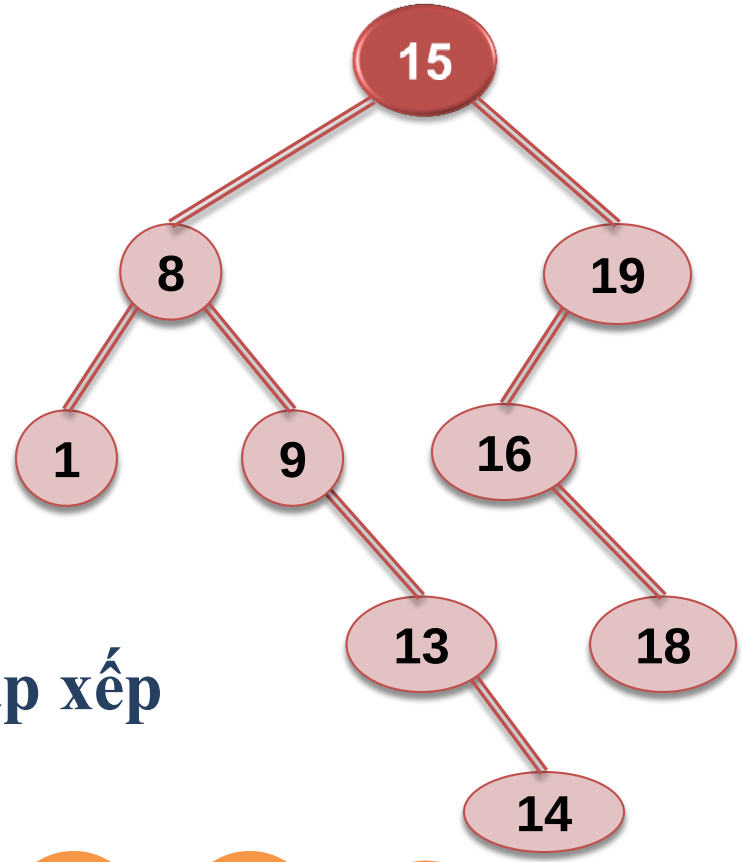




- Tìm phần tử thế mạng
  - phần tử phải nhất của cây con trái hoặc,
  - phần tử trái nhất của cây con phải
- Hoán vị giá trị của phần tử thế mạng và node P
- Xóa phần tử thế mạng

Sắp xếp

- Cho cây nhị phân tìm kiếm

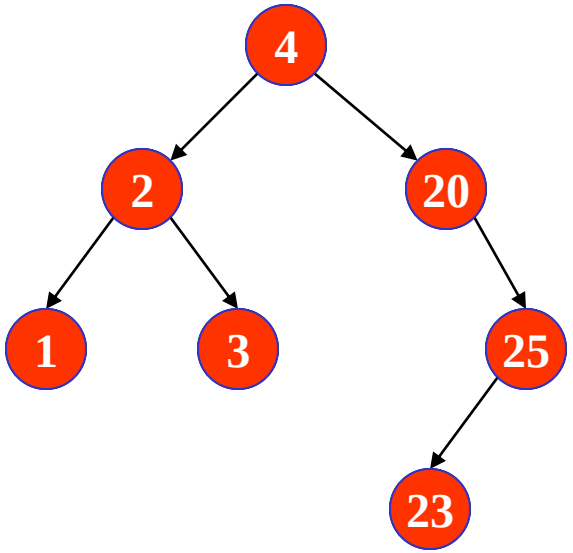


- Thứ tự duyệt các node nếu sử dụng Duyệt giữa?

- Nêu nhận xét
  - **Có thể dễ dàng tạo dữ liệu sắp xếp nếu dùng phép duyệt giữa**

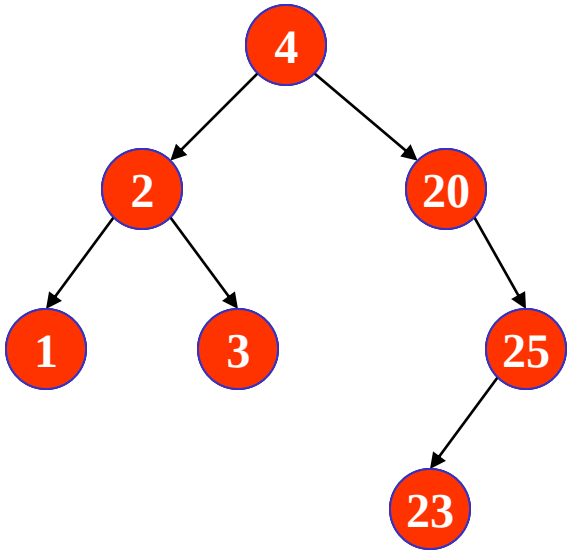
Phép duyệt cây

- Duyệt trước



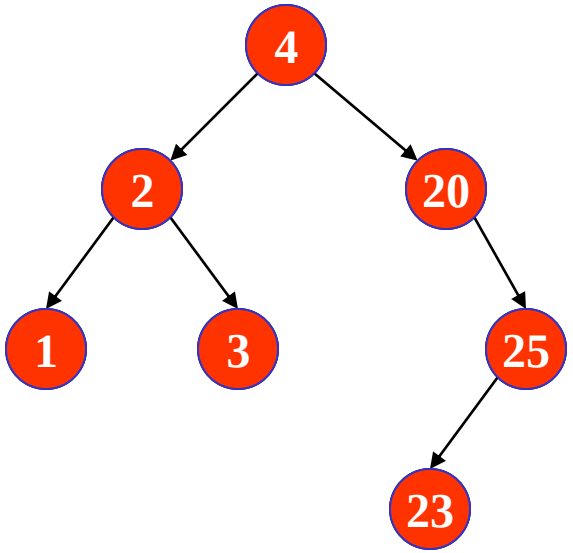
Phép duyệt cây

- Duyệt giữa



Phép duyệt cây

- Duyệt sau



Thời gian thực hiện các phép toán

- Đối với phép tìm kiếm:
  - Trường hợp tốt nhất: mỗi nút (trừ nút lá) đều có 2 con: **$O(\log_2 n)$** (chính là chiều cao của cây).
  - Trường hợp xấu nhất: cây trở thành danh sách liên kết: **$O(n)$**.
  - Trường hợp trung bình là bao nhiêu?

    $O(\log_2 n)$

Ví dụ

- Tạo cây nhị phân tìm kiếm theo thứ tự nhập như sau: 1, 8, 9, 12, 14, 15, 16, 18, 19

Ví dụ

- Tạo cây nhị phân tìm kiếm theo thứ tự nhập như sau: 1, 8, 9, 12, 14, 15, 16, 18, 19