BM Công nghệ phần mềm – Khoa CNTT
se.nuce.edu.vn

CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN 2

Chương 1. Tree

Chương 1. Tree

1.1. Các khái niệm cơ bản
1.2. Phép duyệt cây và biểu diễn cây
1.3. Cây nhị phân và cây nhị phân tìm kiếm
1.4. Cây AVL
1.5. Cây AA

Chương 1. Tree

1.1. Các khái niệm cơ bản

1.2. Phép duyệt cây và biểu diễn cây

1.3. Cây nhị phân và cây nhị phân tìm kiếm

1.4. Cây AVL

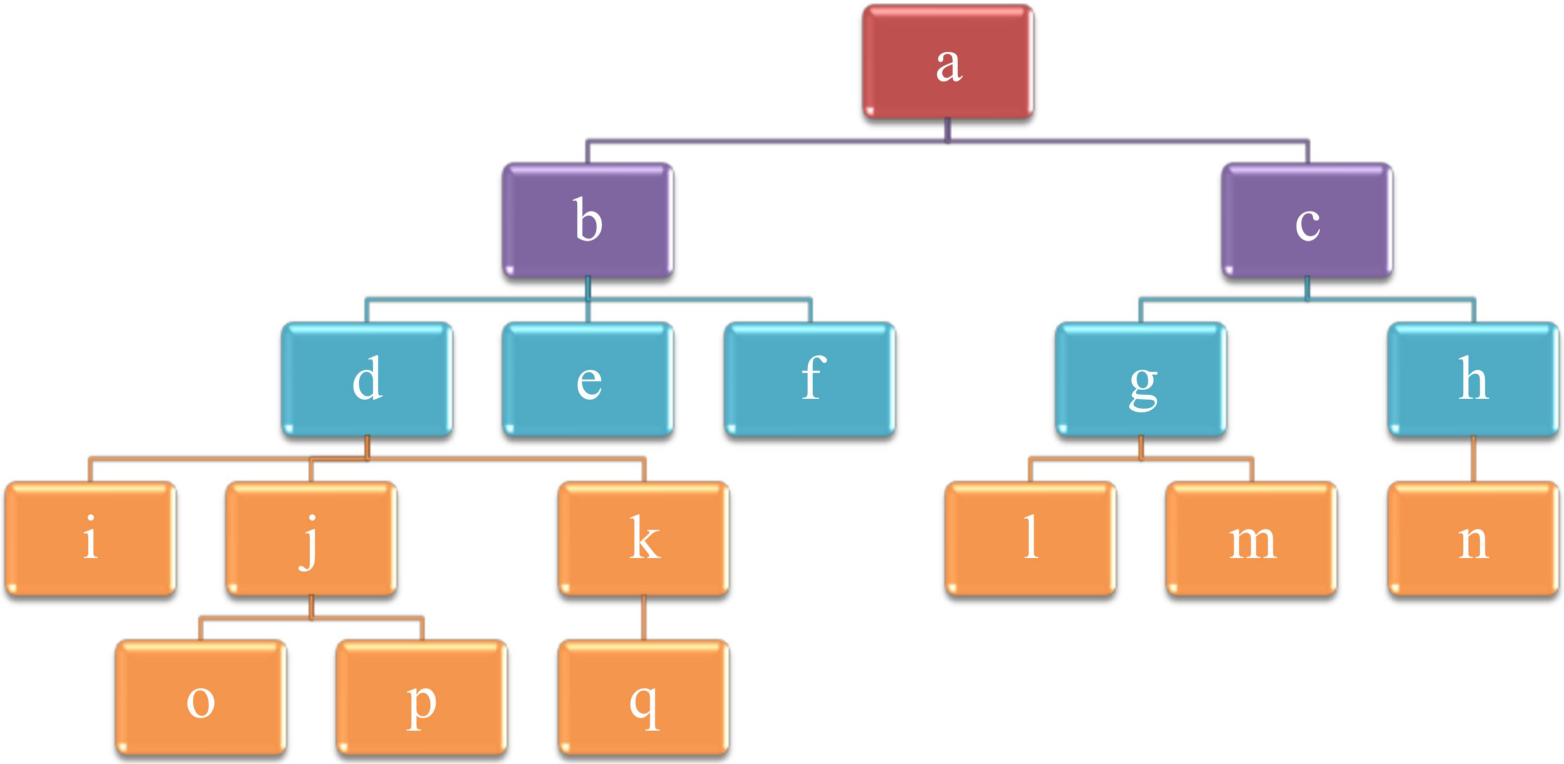
1.5. Cây AA

1.1. Các khái niệm cơ bản

- Một số thuật ngữ
  - Tree
  - Search Tree
  - Binary Search Tree
  - Balanced Tree
  - AVL Tree
  - AA Tree
  - Red-Black Tree
  - ...

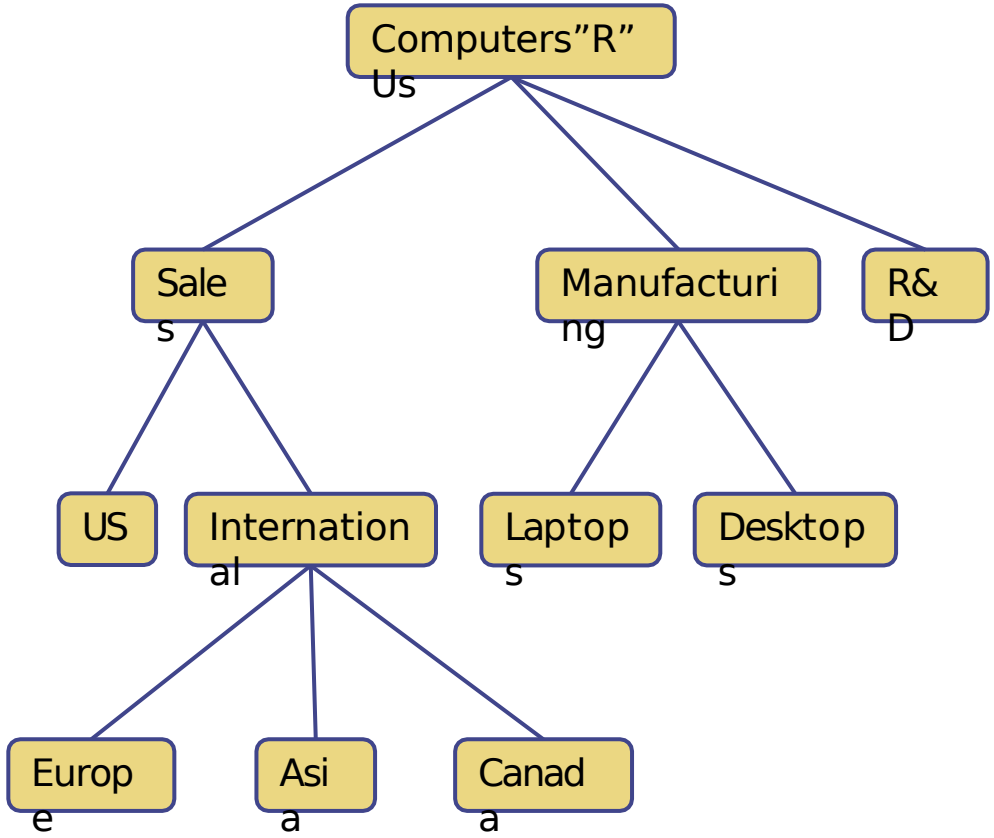
1.1. Các khái niệm cơ bản

- Cây tổng quát

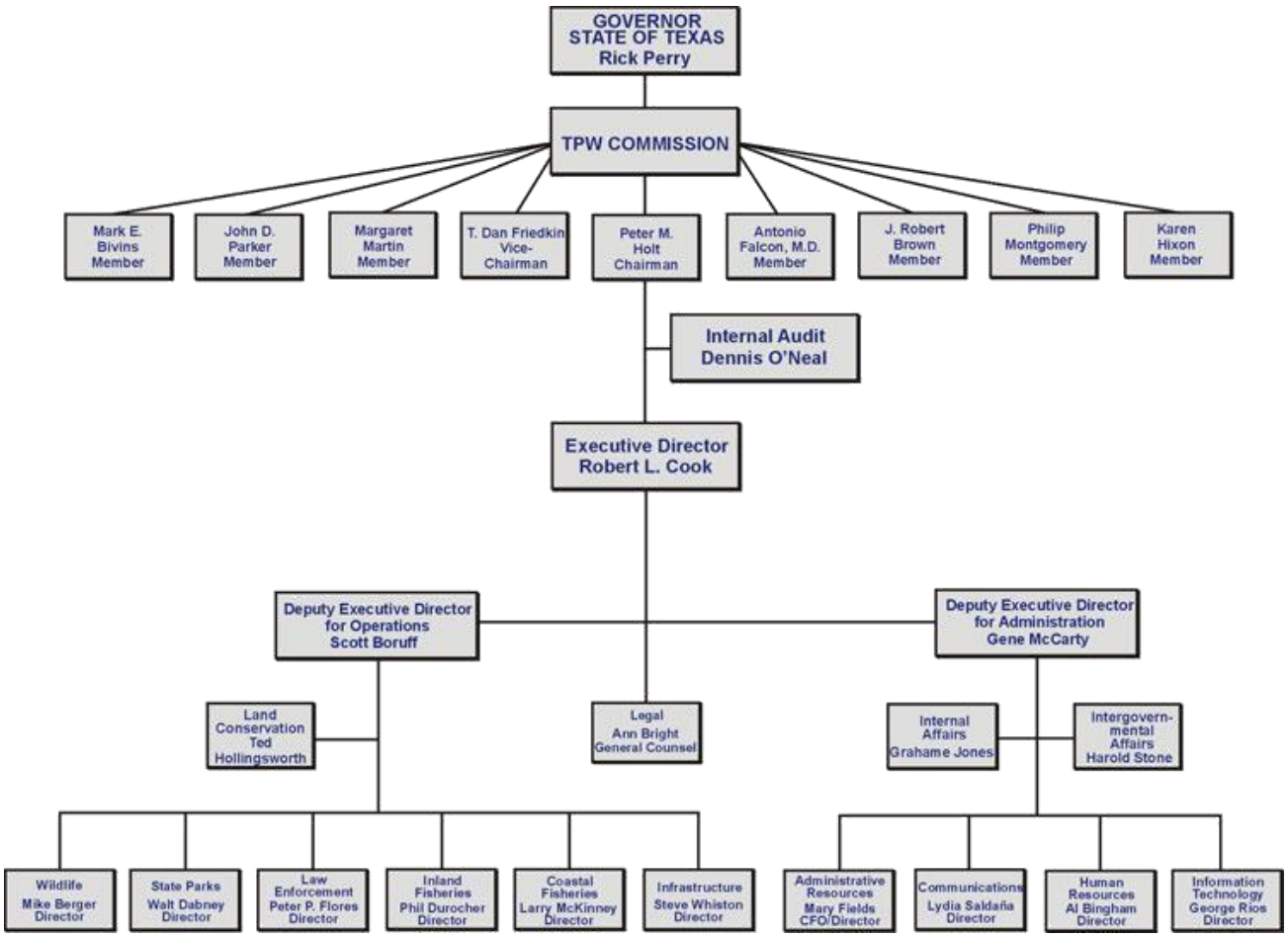


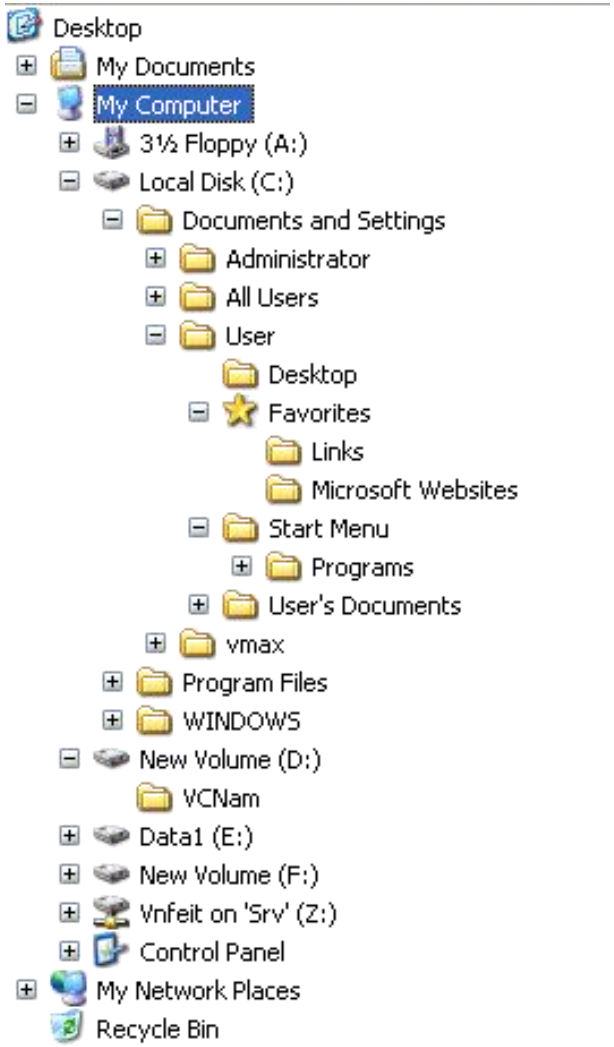
1.1. Các khái niệm cơ bản

- Ví dụ: tập hợp các thành viên trong một dòng họ với quan hệ cha – con
- Trong ngành công nghệ thông tin, cây là mô hình trừu tượng của một cấu trúc phân cấp
- Một cây bao gồm các đỉnh với quan hệ cha – con
- Ứng dụng
  - Sơ đồ tổ chức
  - Hệ thống file
  - Các môi trường lập trình



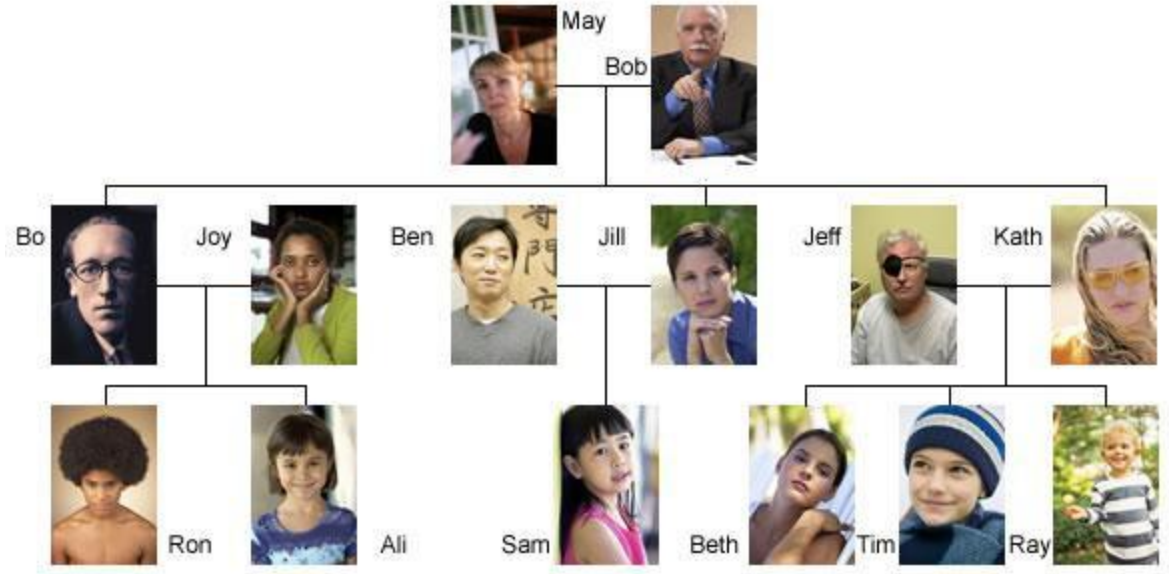
1.1. Các khái niệm cơ bản





1.1. Các khái niệm cơ bản

- Định nghĩa:
  1. Toán học : thông qua đồ thị định hướng
  2. Đệ quy

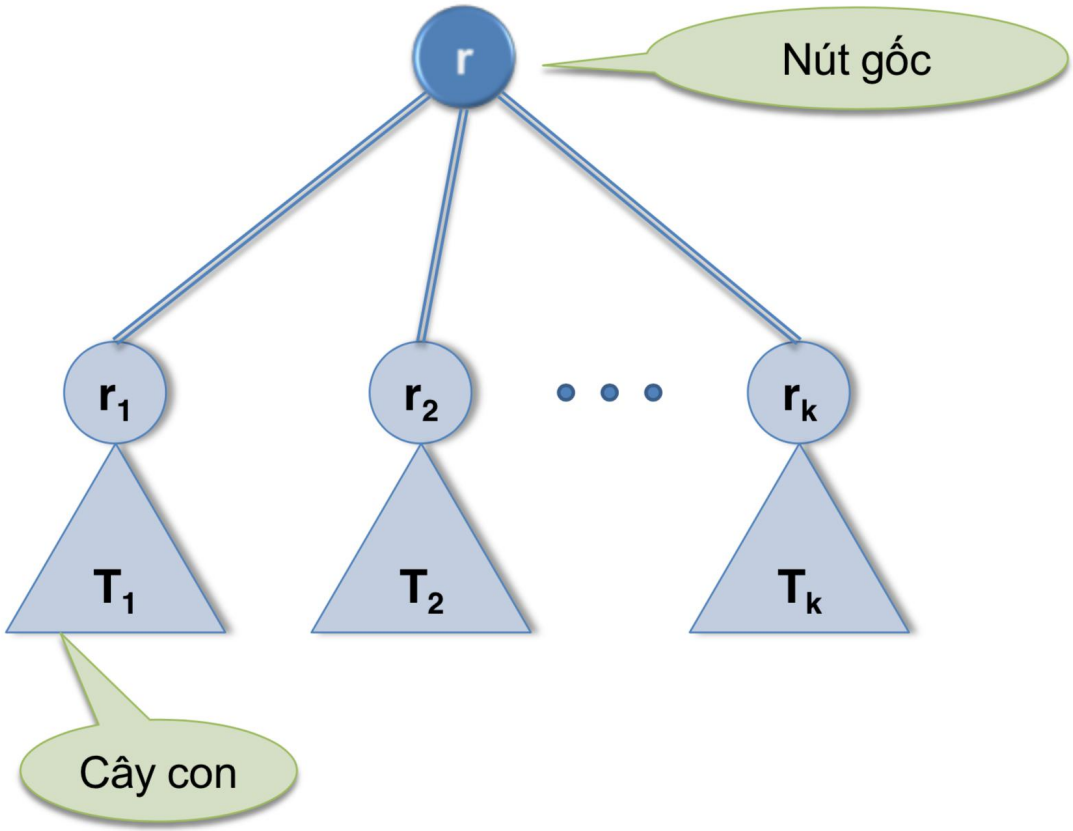


1.1. Các khái niệm cơ bản

- Định nghĩa cây (cây có gốc) được xác định qua đệ quy như sau:

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

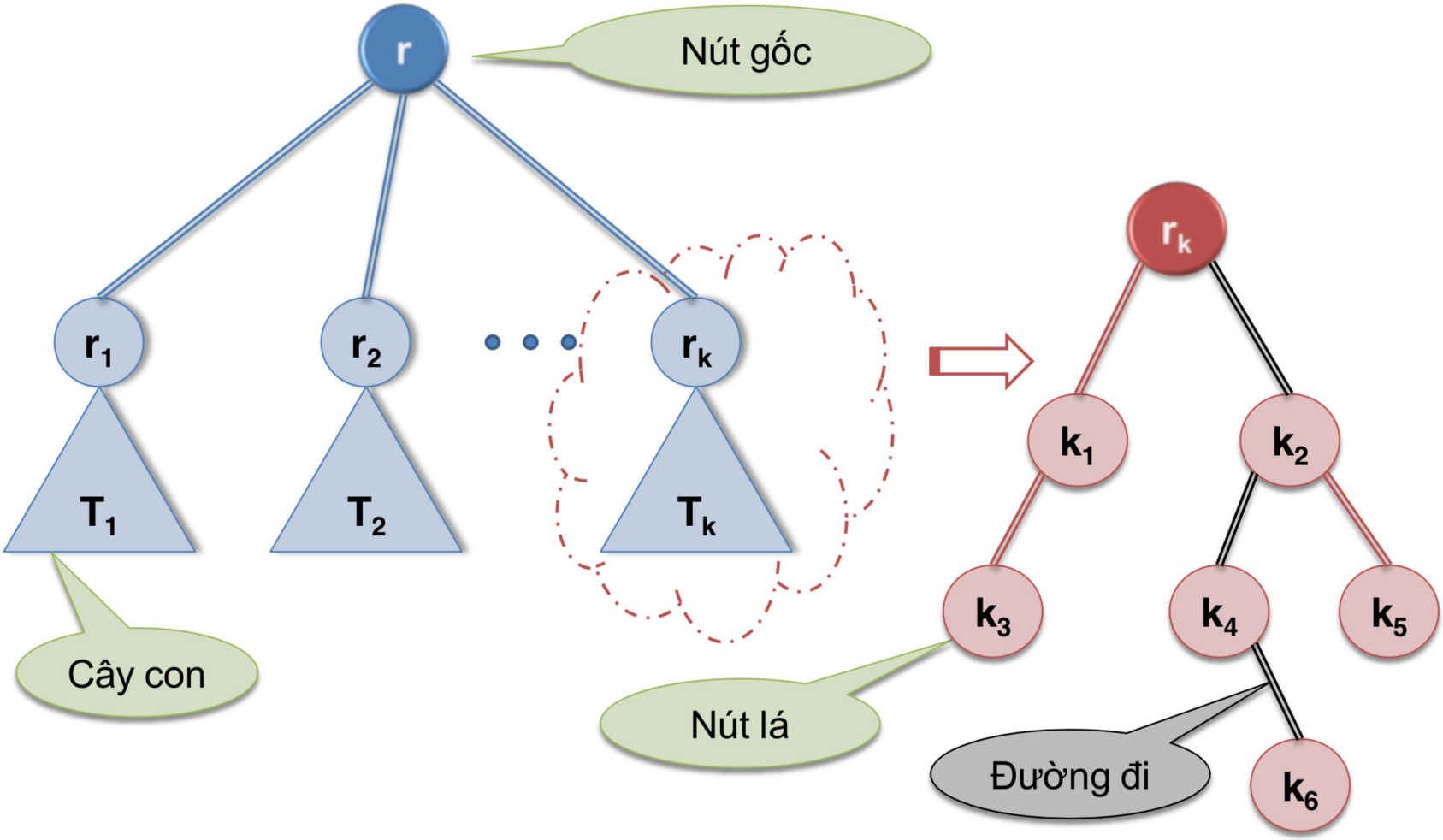
Giả sử r là một đỉnh mới không thuộc các cây T_i. Khi đó, tập hợp T gồm đỉnh r và các cây T_i tạo thành một cây mới với gốc r. Các cây T_1, T_2, ... T_k được gọi là cây con của gốc r.



1.1. Các khái niệm cơ bản

- 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

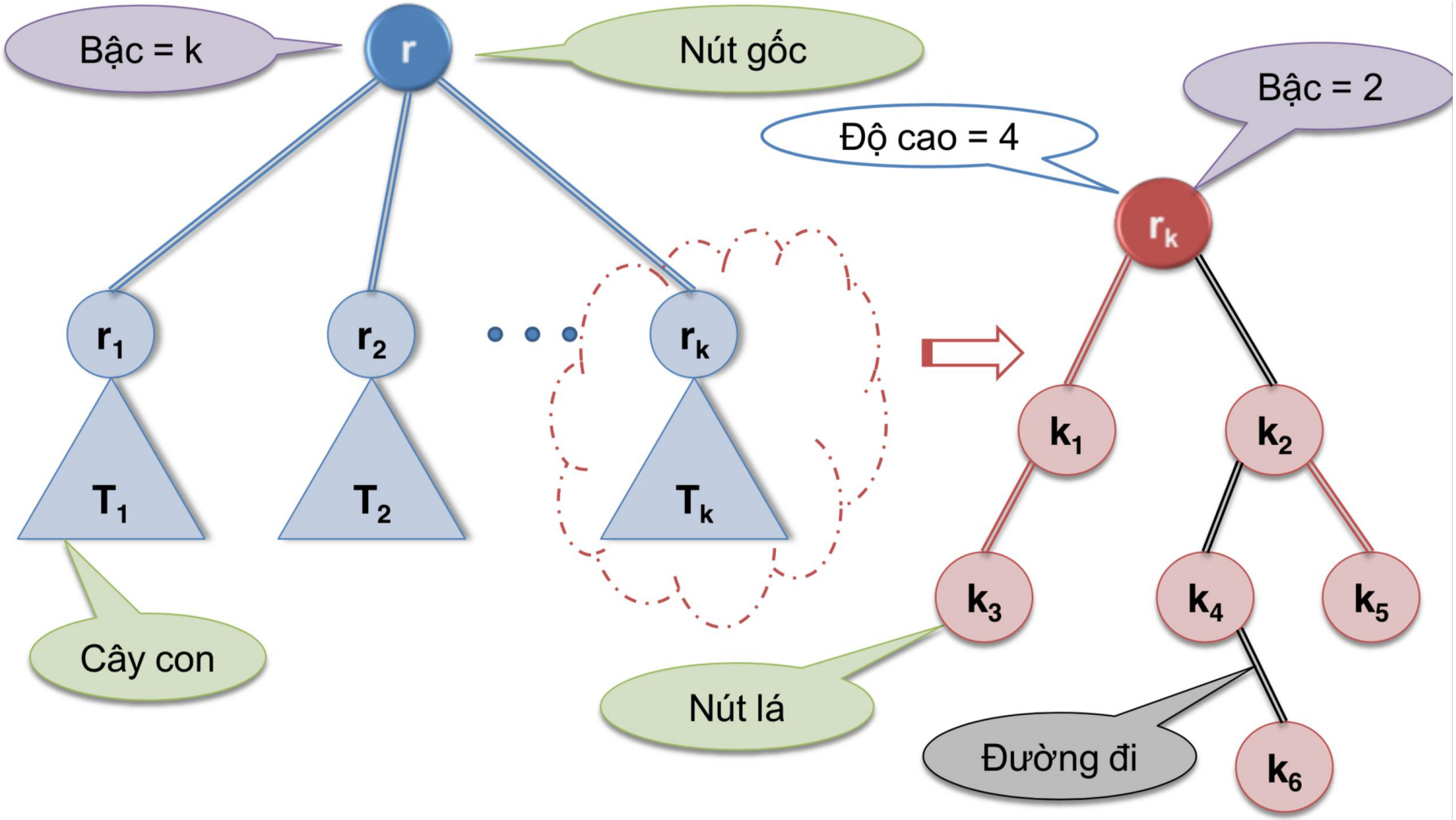
1.1. Các khái niệm cơ bản



1.1. Các khái niệm cơ bản

- Các khái niệm(tiếp)
  - **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

1.1. Các khái niệm cơ bản



Chương 1. Tree

1.1. Các khái niệm cơ bản

1.2. Phép duyệt cây và biểu diễn cây

1.3. Cây nhị phân và cây nhị phân tìm kiếm

1.4. Cây AVL

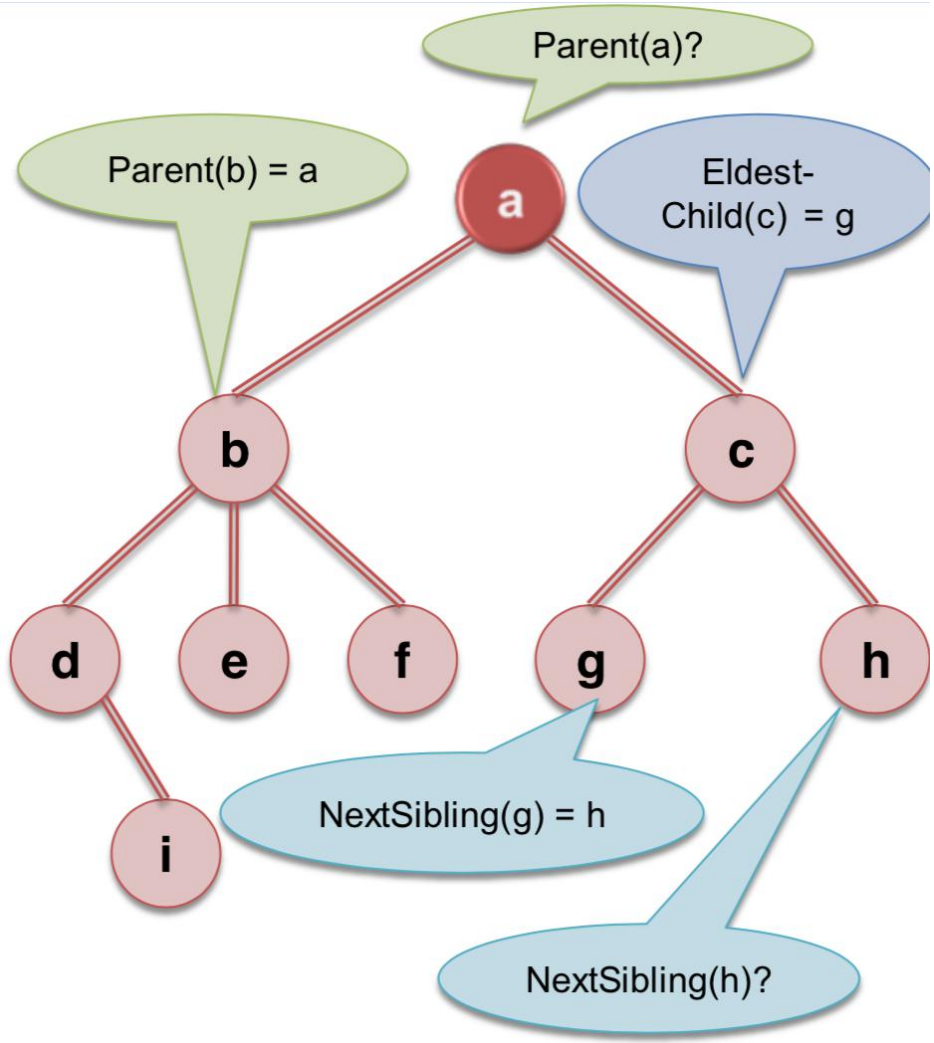
1.5. Cây AA

1.2.1 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 tiền thứ tự (Pre-order)
  - Duyệt trung thứ tự (In-order)
  - Duyệt hậu thứ tự (Post-order)

1.2.1 Phép duyệt cây

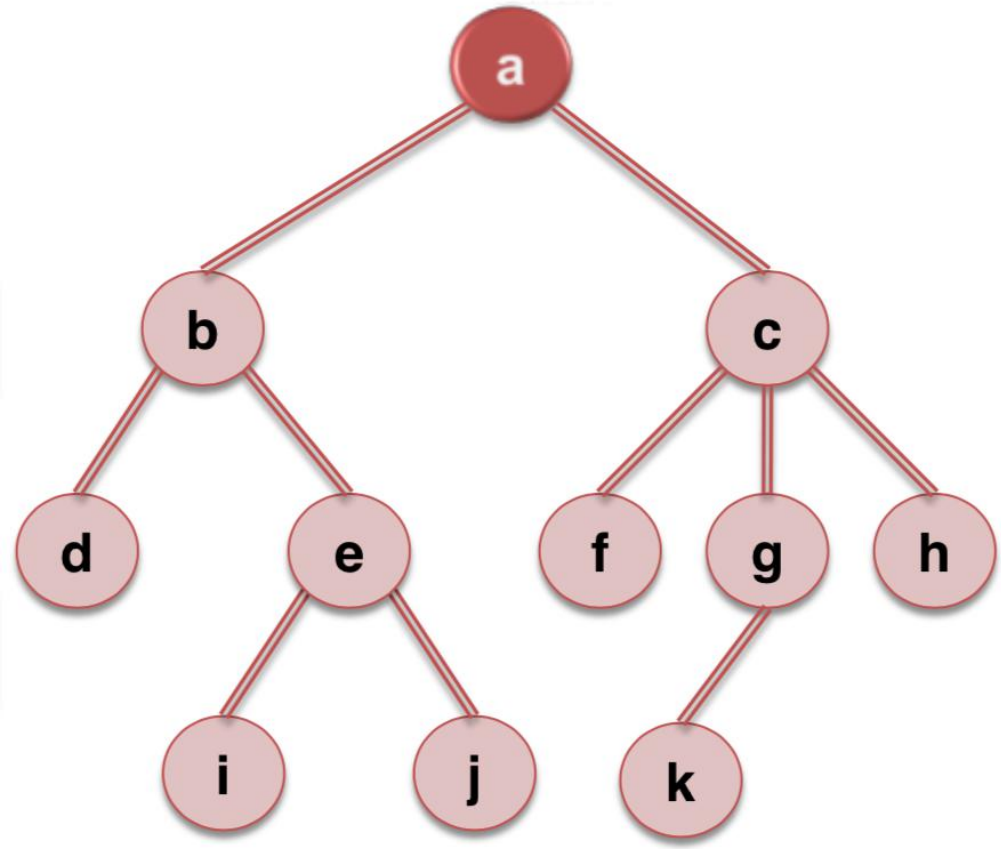
- Tìm cha một đỉnh
  - *Parent(x)*
- Tìm đỉnh con trái nhất
  - *EldestChild(x)*
- Tìm đỉnh kề phải
  - *NextSibling(x)*



1.2.1 Phép duyệt cây

- Duyệt tiền thứ tự
  - *a b d e i j c f g k h*

  => Duyệt theo chiều sâu
- Duyệt trung thứ tự
  - *d b i e j a f c k g h*
- Duyệt hậu thứ tự
  - *d i j e b f k g h c a*



1.2.1 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);
}
```

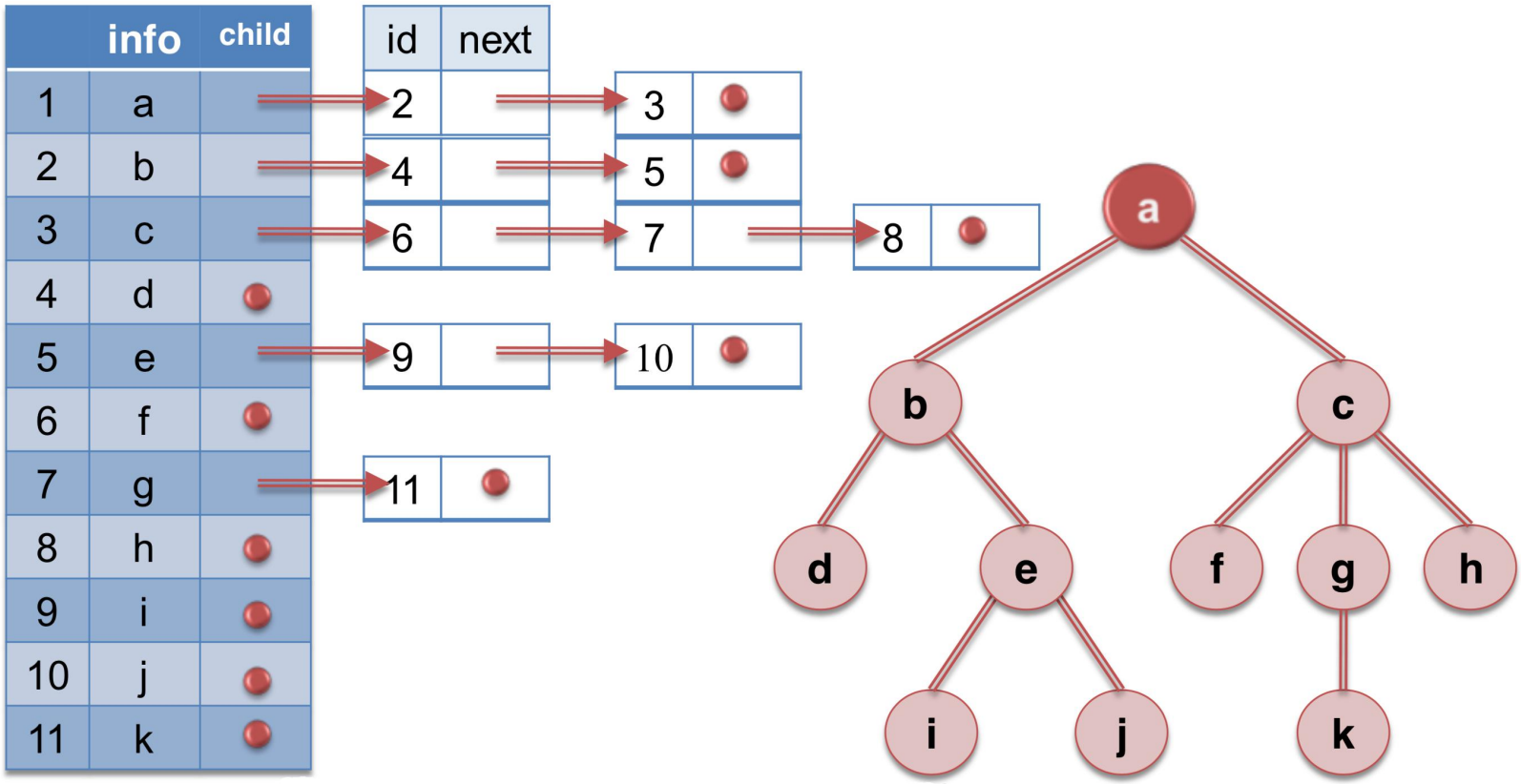
1.2.1 Phép duyệt cây

- In-order

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

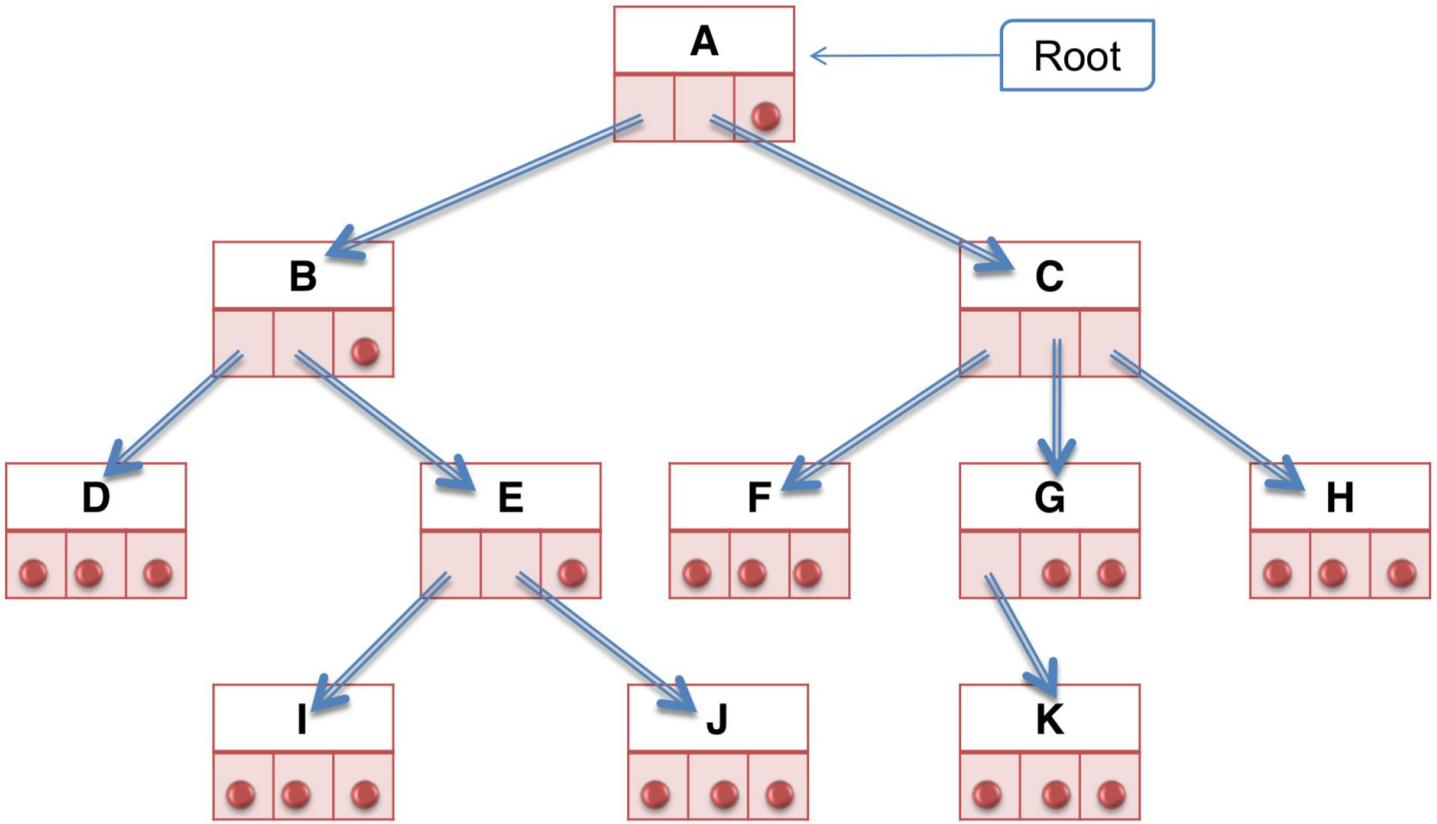
1.2.1 Biểu diễn cây

- Bằng danh sách cây con



1.2.1 Biểu diễn cây

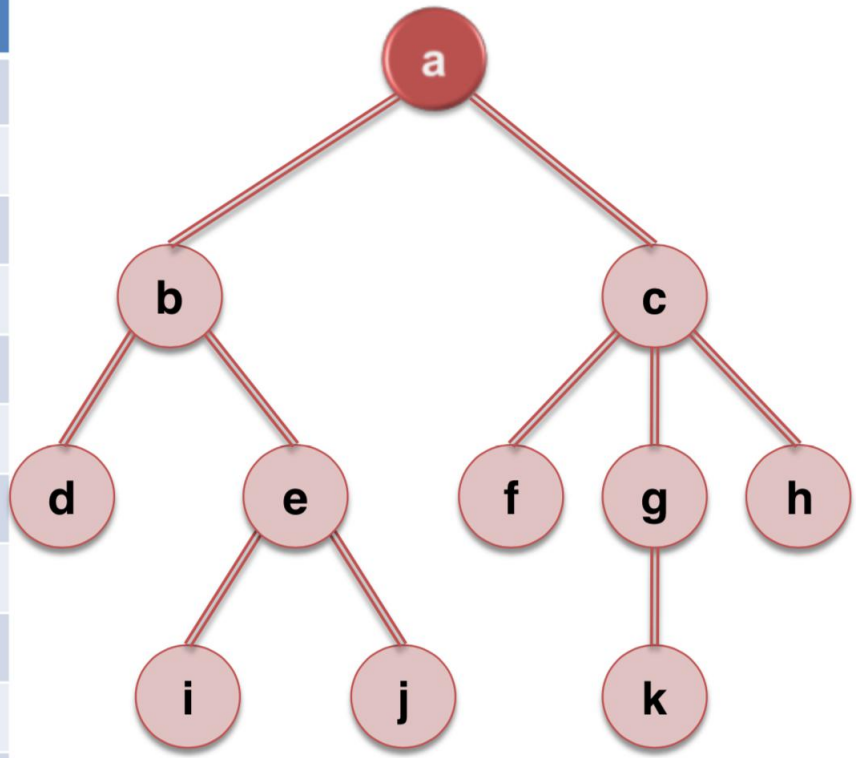
- Bằng danh sách cây con



1.2.1 Biểu diễn cây

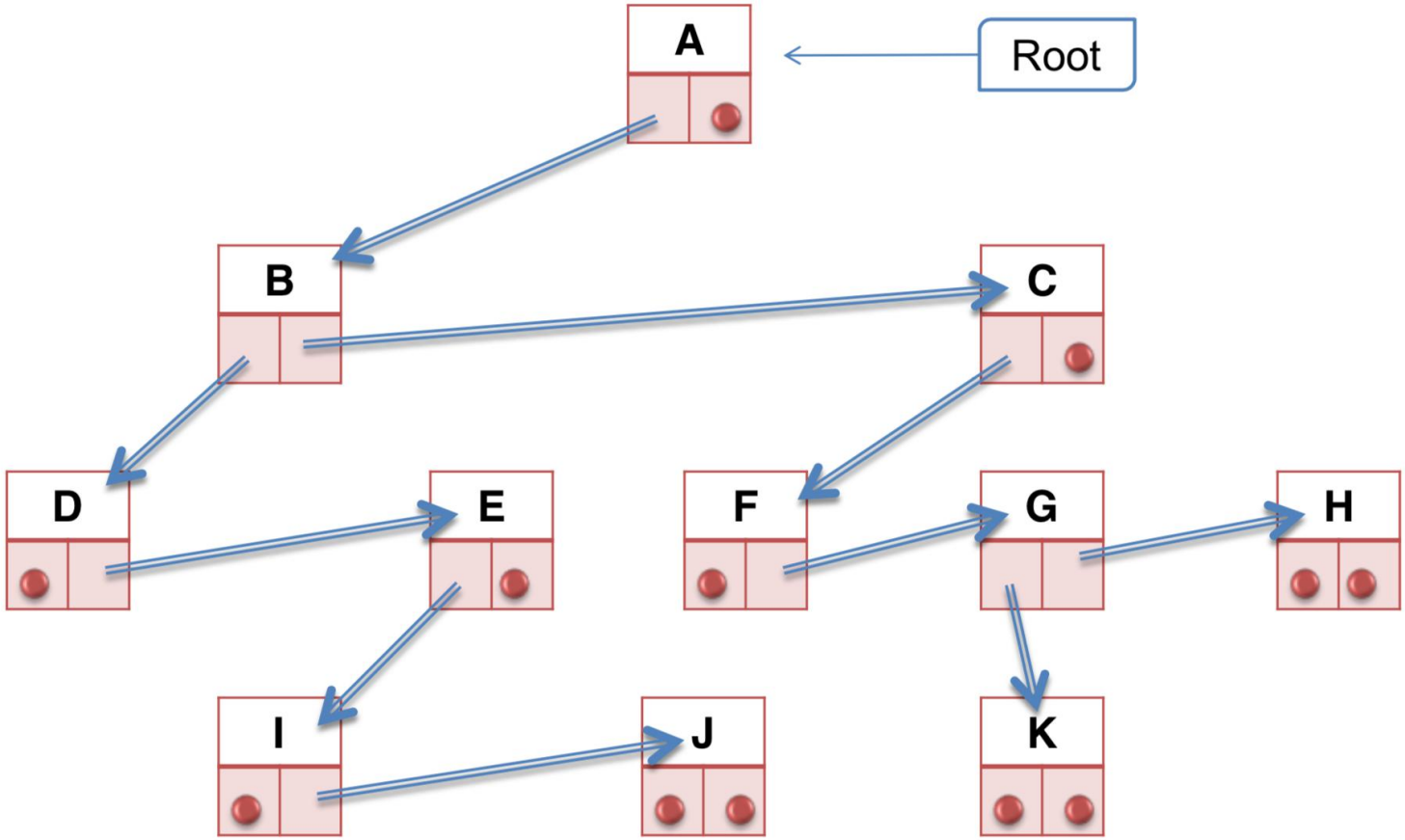
- Bằng đỉnh trái nhất và đỉnh kề phải

	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



1.2.1 Biểu diễn cây

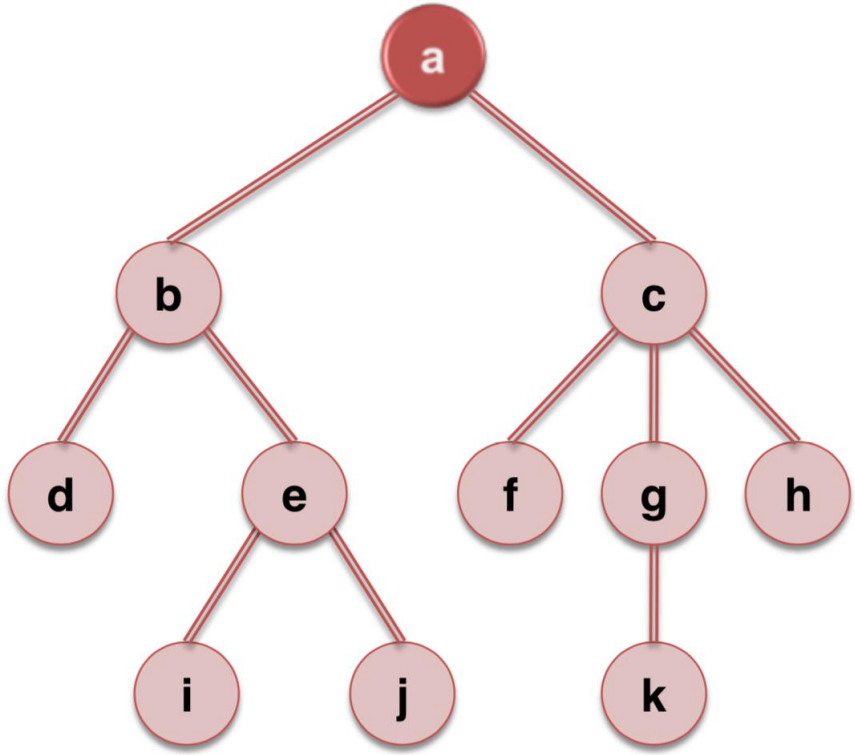
- Bằng đỉnh trái nhất và đỉnh kề phải



1.2.1 Biểu diễn cây

- 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



Chương 1. Tree

1.1. Các khái niệm cơ bản

1.2. Phép duyệt cây và biểu diễn cây

1.3. Cây nhị phân và cây nhị phân tìm kiếm

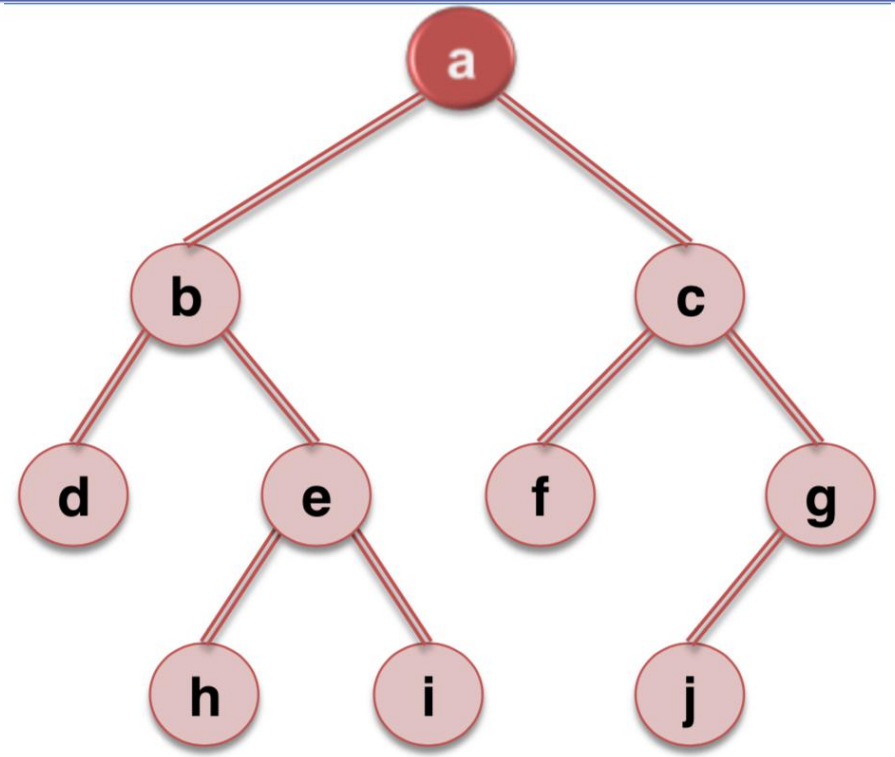
1.4. Cây AVL

1.5. Cây AA

1.3.1 Cây nhị phân

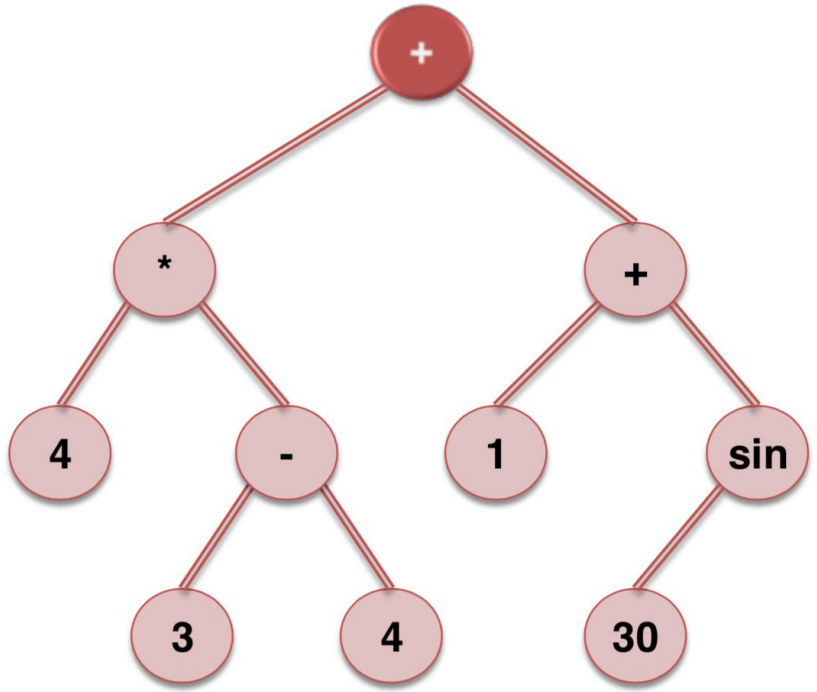
- Là cây mỗi đỉnh có bậc tối đa bằng 2.
- Các cây con được gọi là cây con trái và cây con phải.
- Có toàn bộ các thao tác cơ bản của cây.

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



1.3.1 Cây nhị phân

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

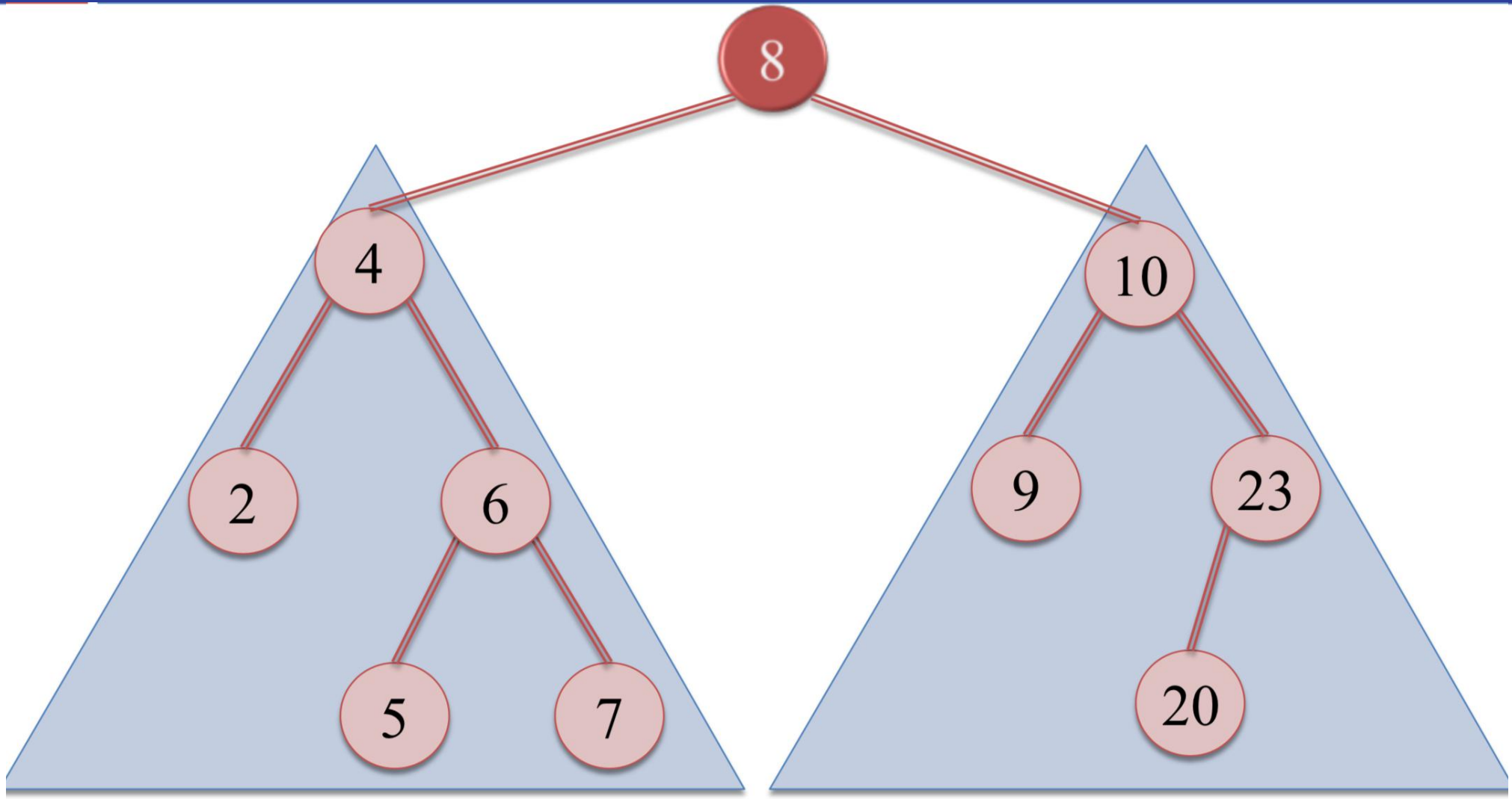


Cây biểu thức:
*4 * (3 – 4) + (1 + sin(30))*

1.3.2 Cây nhị phân tìm kiếm

- Cây nhị phân tìm kiếm là cây nhị phân thoả 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.

1.3.2 Cây nhị phân tìm kiếm



1.3.2 Cây nhị phân tìm kiếm

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

1.3.3 Thao tác trên cây nhị phân tìm kiếm

- Thêm phần tử (khoá)
- Tìm kiếm phần tử (khoá)
- Xoá phần tử (khoá)
- Sắp xếp
- Duyệt cây
- Quay cây

1.3.3 Thao tác trên cây nhị phân tìm kiếm

- Thêm phần tử
  - Bước 1: Bắt đầu từ gốc
  - Bước 2: So sánh dữ liệu (khoá) cần thêm với dữ liệu (khoá) 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 (khoá) cần thêm. Kết thúc

1.3.3 Thao tác trên cây nhị phân tìm kiếm

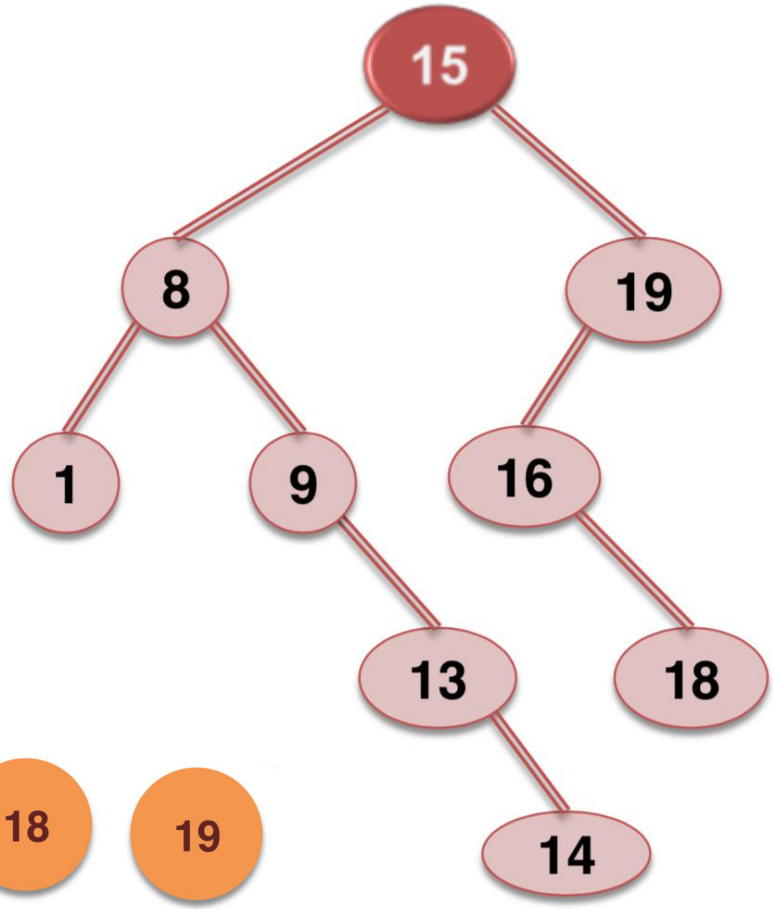
- Tìm kiếm phần tử
  - Bước 1: Bắt đầu từ gốc
  - Bước 2: So sánh dữ liệu (khoá) cần tìm kiếm với dữ liệu (khoá) của node hiện hành
    - Nếu bằng nhau => Tìm thấy. Kết thúc
    - Nếu nhỏ hơn => Đi qua nhánh trái, Tiếp bước 2
    - Nếu lớn hơn => Đi qua nhánh phải, Tiếp bước 2
  - Bước 3: Không thể đi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

1.3.3 Thao tác trên cây nhị phân tìm kiếm

- Xoá phần tử
  - Tìm đến node chứa dữ liệu (khoá) cần xoá
  - Xét các trường hợp
    - Node lá
    - Node chỉ có 1 con
    - Node có 2 con: dùng phần tử thế mạng để xoá thế.

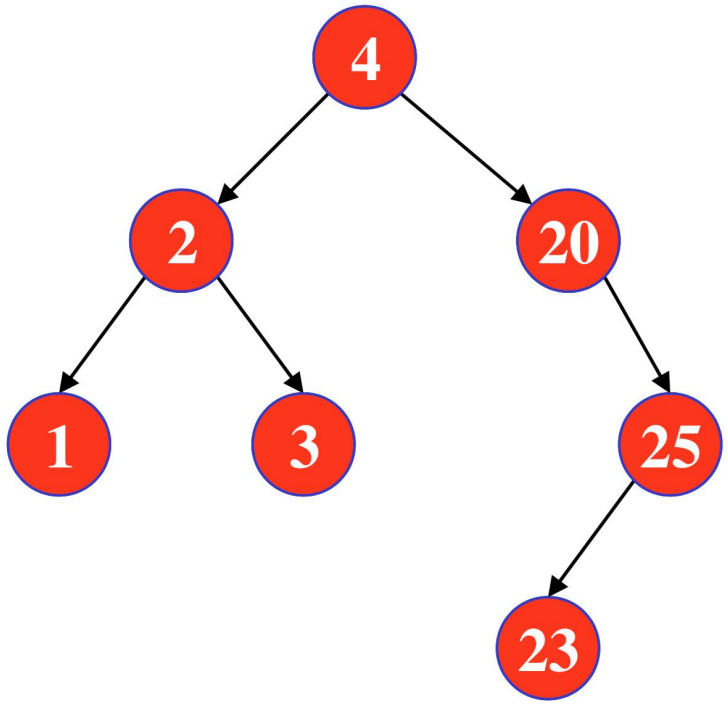
1.3.3 Thao tác trên cây nhị phân tìm kiếm

- Sắp xếp
  - Cho cây nhị phân tìm kiếm
  - Thứ tự duyệt các node nếu sử dụng Duyệt giữa?
  - Nhận xét
    - Có thể dễ dàng tạo dữ liệu sắp xếp nếu dùng phép duyệt giữa



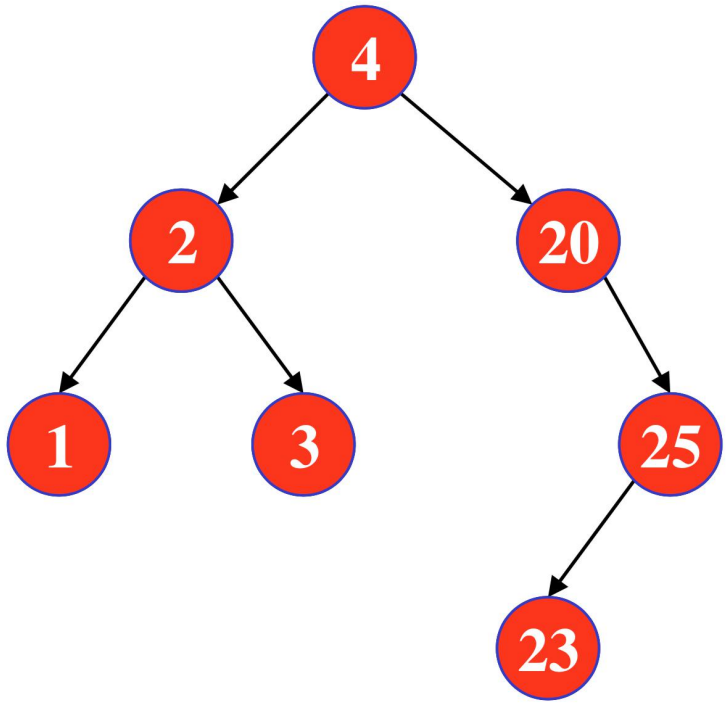
1.3.3 Thao tác trên cây nhị phân tìm kiếm

- Phép duyệt cây
  - Duyệt trước



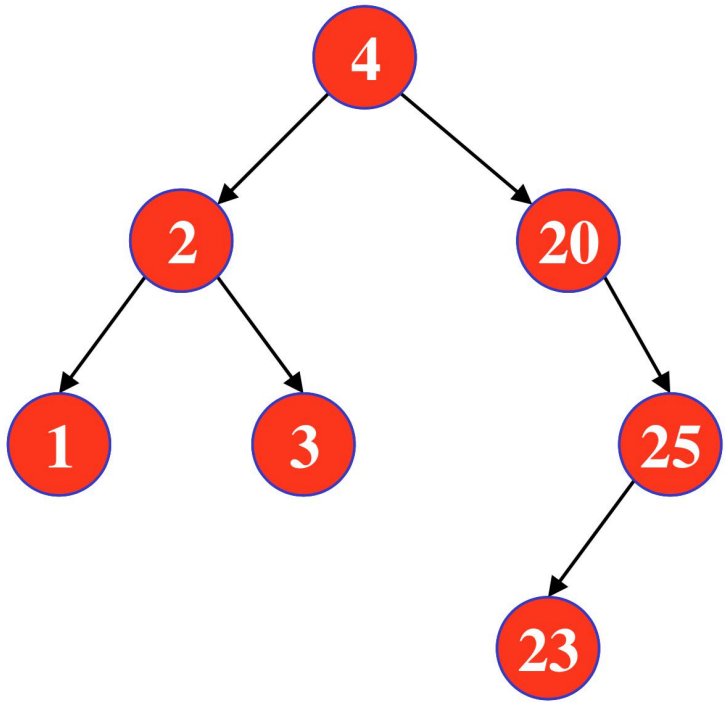
1.3.3 Thao tác trên cây nhị phân tìm kiếm

- Phép duyệt cây
  - Duyệt giữa



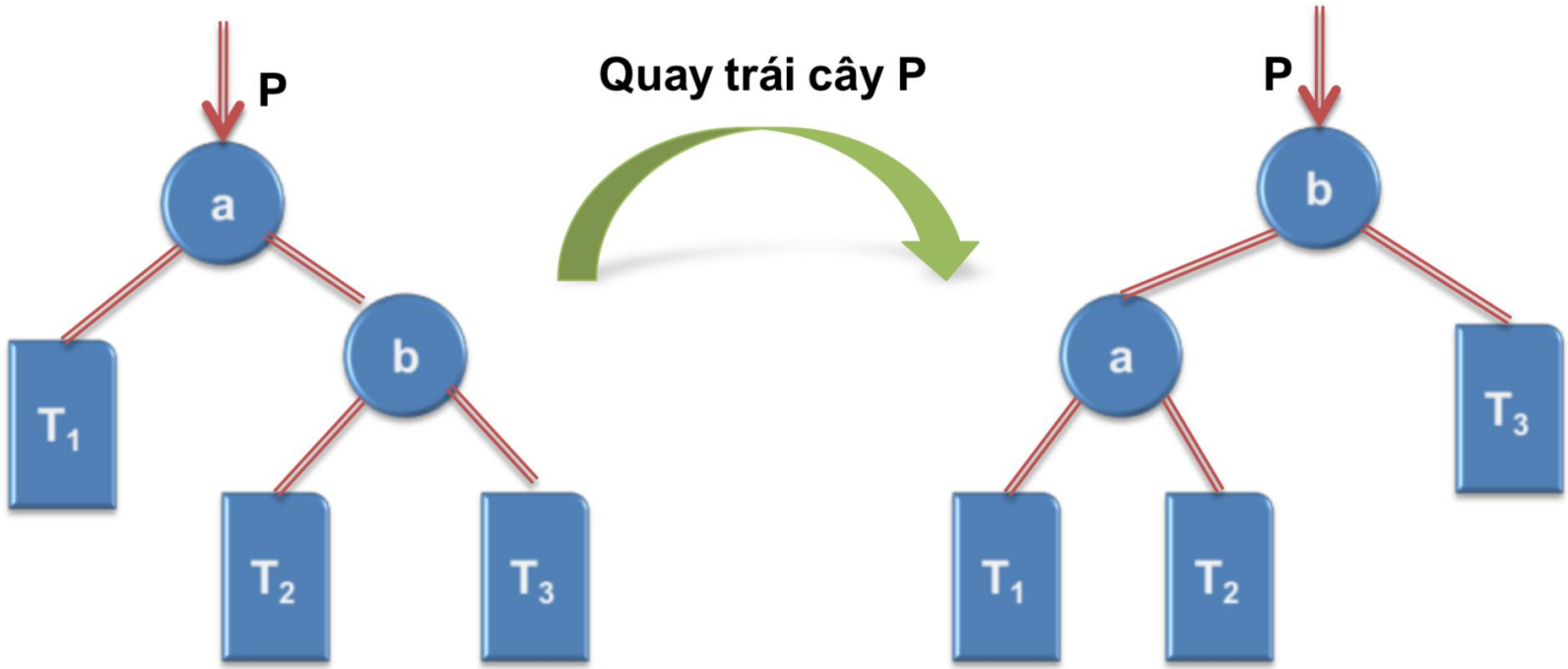
1.3.3 Thao tác trên cây nhị phân tìm kiếm

- Phép duyệt cây
  - Duyệt sau



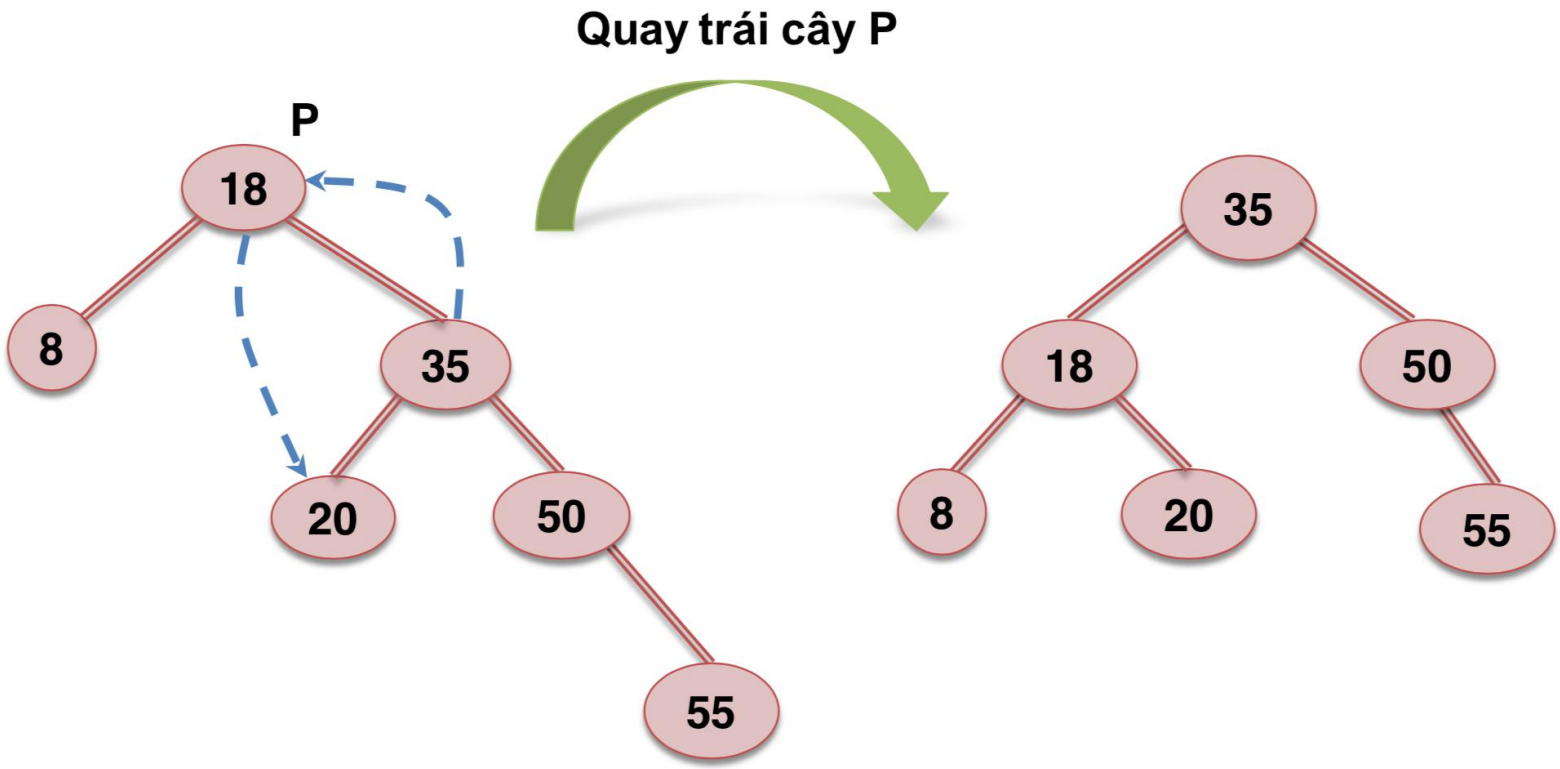
1.3.3 Thao tác trên cây nhị phân tìm kiếm

- Phép quay trái



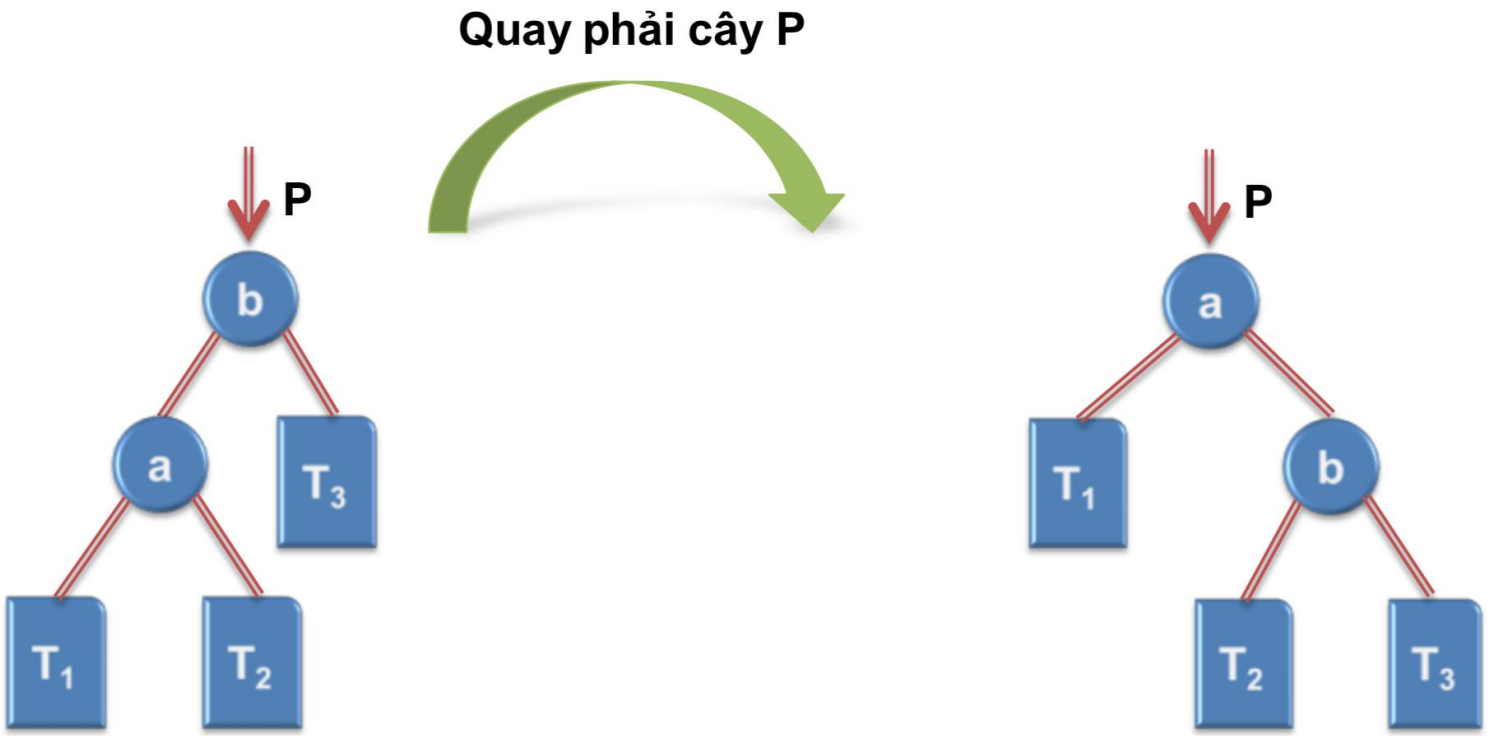
1.3.3 Thao tác trên cây nhị phân tìm kiếm

- Phép quay trái



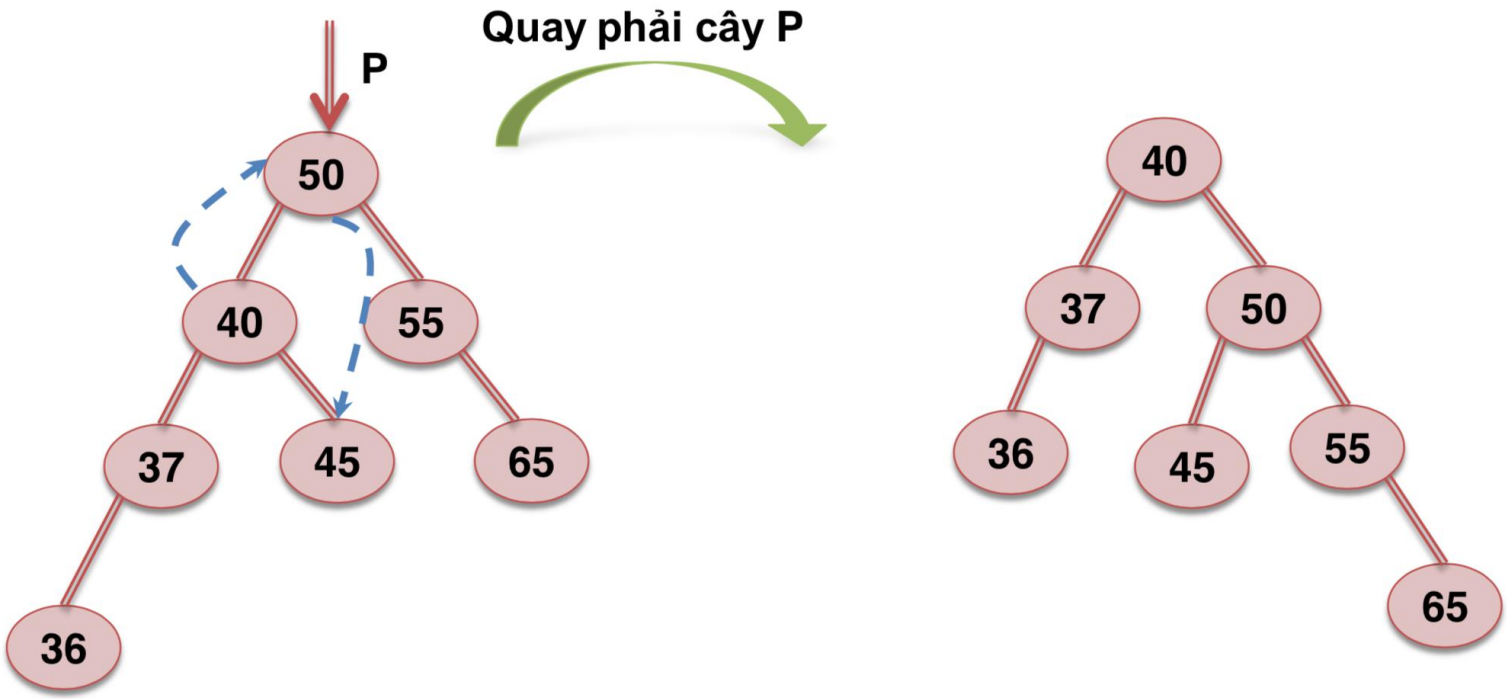
1.3.3 Thao tác trên cây nhị phân tìm kiếm

- Phép quay phải



1.3.3 Thao tác trên cây nhị phân tìm kiếm

- Phép quay trái



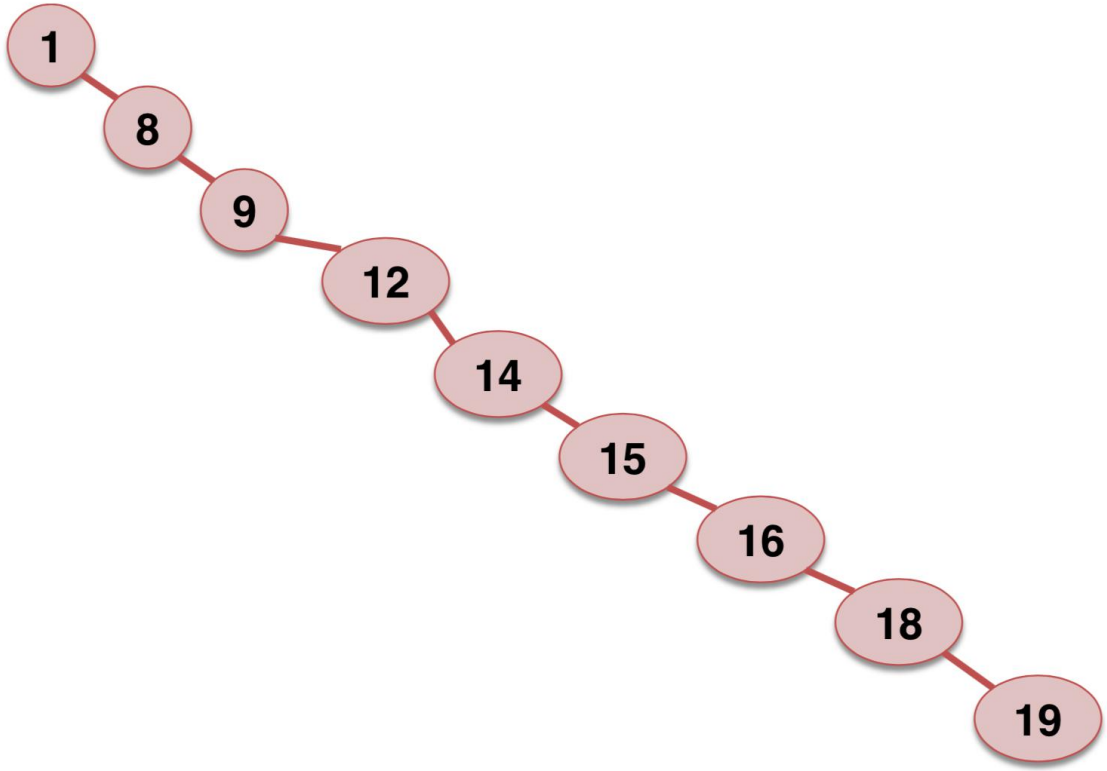
1.3.3 Thao tác trên cây nhị phân tìm kiếm

- 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 ?

      $O(\log_2 n)$

1.3.3 Thao tác trên cây nhị phân tìm kiếm

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



1.3.3 Thao tác trên cây nhị phân tìm kiếm

- Bài tập: Kết quả của phép duyệt tiền thứ tự của một cây nhị phân tìm kiếm là **7 6 4 15 13 9 14 30 31.** Hãy vẽ lại cây nhị phân tìm kiếm đó.

Câu hỏi