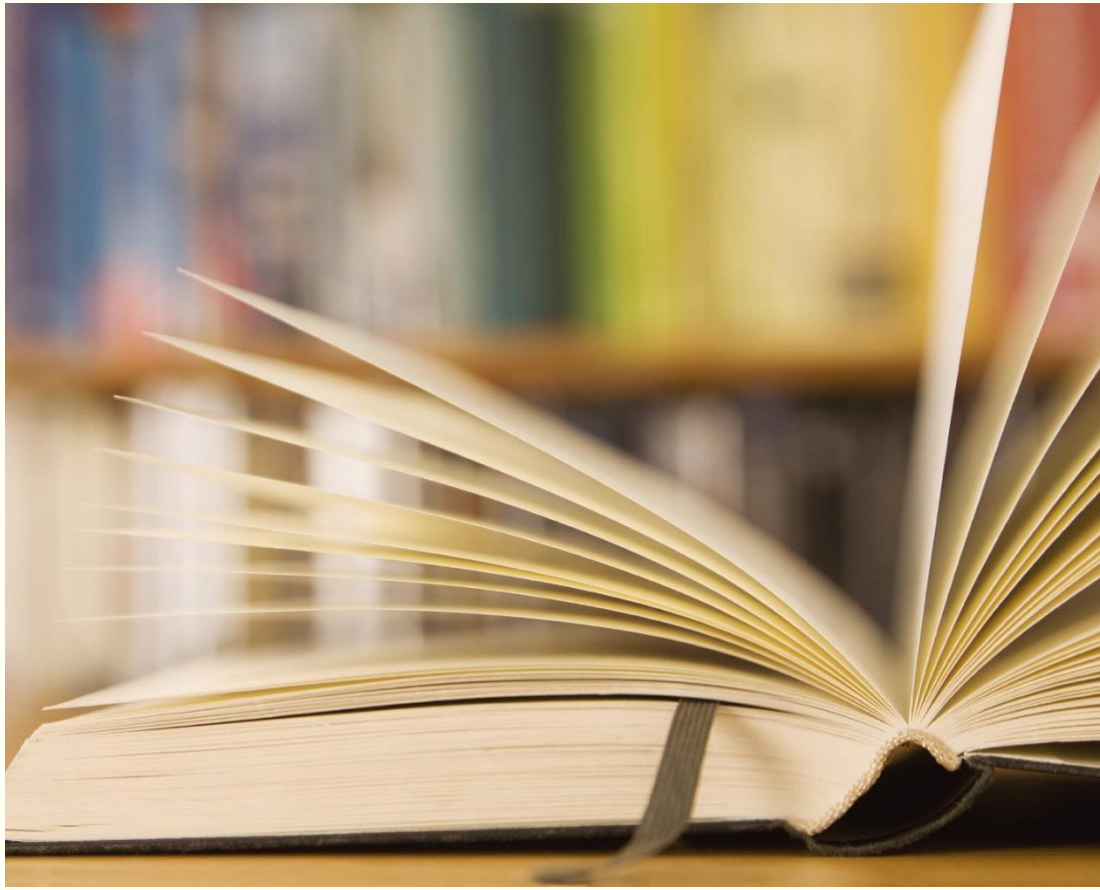
CÂY



Mục tiêu

- Hiểu được cấu trúc dữ liệu cây
- Cài đặt cây, thực hiện các phép toán trên cây

Nội dung

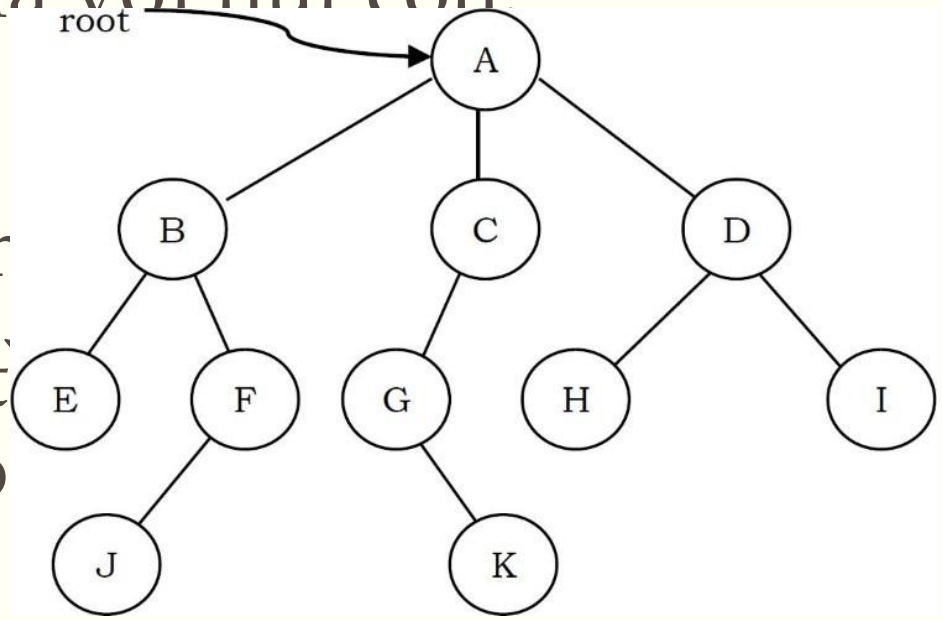
- Các khái niệm
- Cây nhị phân
- Cây tổng quát
- Cây AVL

Khái niệm

- Cây là cấu trúc dữ liệu tương tự như danh sách được liên kết nhưng thay vì mỗi nút trỏ đến nút tiếp theo theo kiểu tuyến tính, mỗi nút trỏ tới một số nút.
- Cây là một ví dụ về cấu trúc dữ liệu phi tuyến.
- Cấu trúc cây là một cách thể hiện bản chất phân cấp của một cấu trúc dưới dạng đồ họa

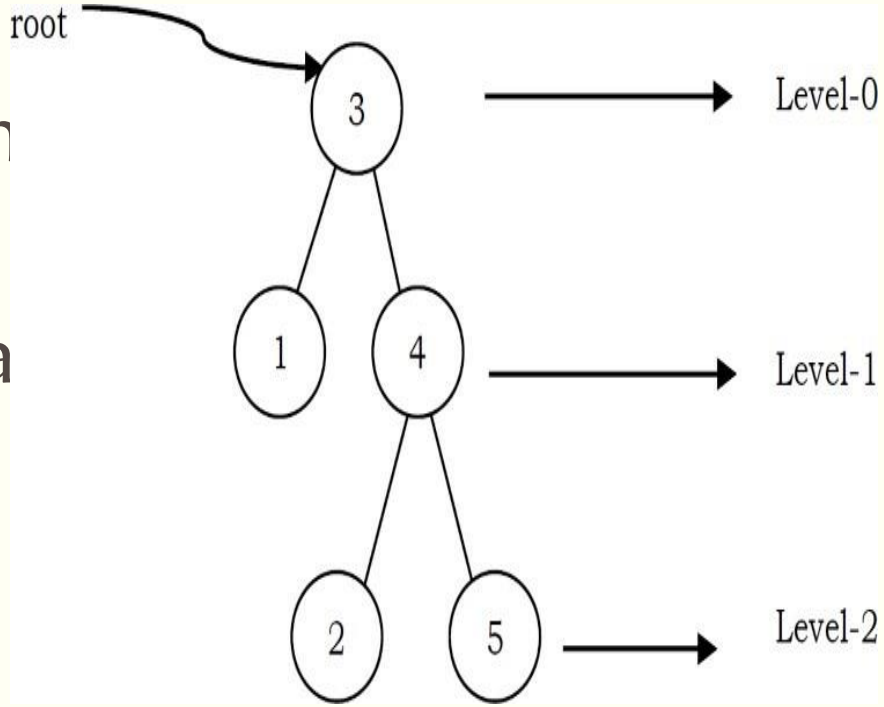
Một số thuật ngữ

- Gốc của cây là nút không có cha.
- Các cạnh (cành cây) kết nút nút cha với nút con
- Lá: Nút không có nút con
- Anh em liền kề: Các nút có cùng m
- Nút p gọi là tổ tiên của nút q nếu t
  từ gốc đến q mà p có mặt trong đó



Một số thuật ngữ

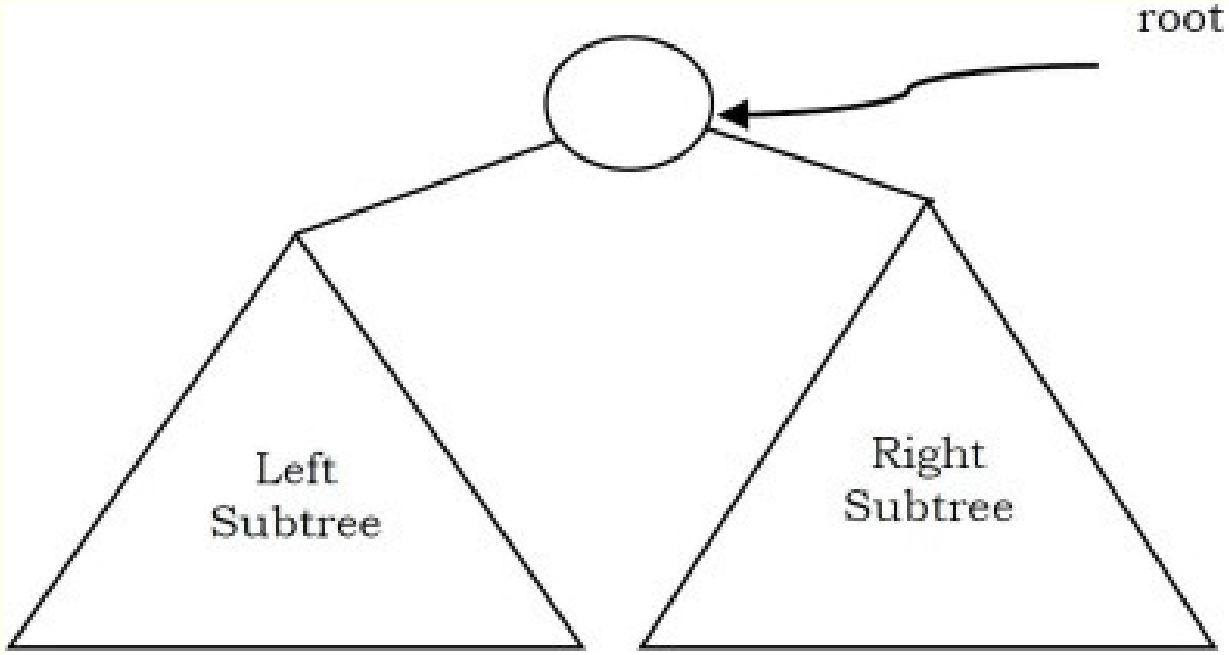
- Mức: Tập hợp tất cả các nút ở độ sâu nhất định được gọi là mức của cây
- Độ sâu của một nút: Độ sâu của m
của đường dẫn từ gốc đến nút đó
- Chiều cao của cây là chiều dài của
đến nút sâu nhất trong cây

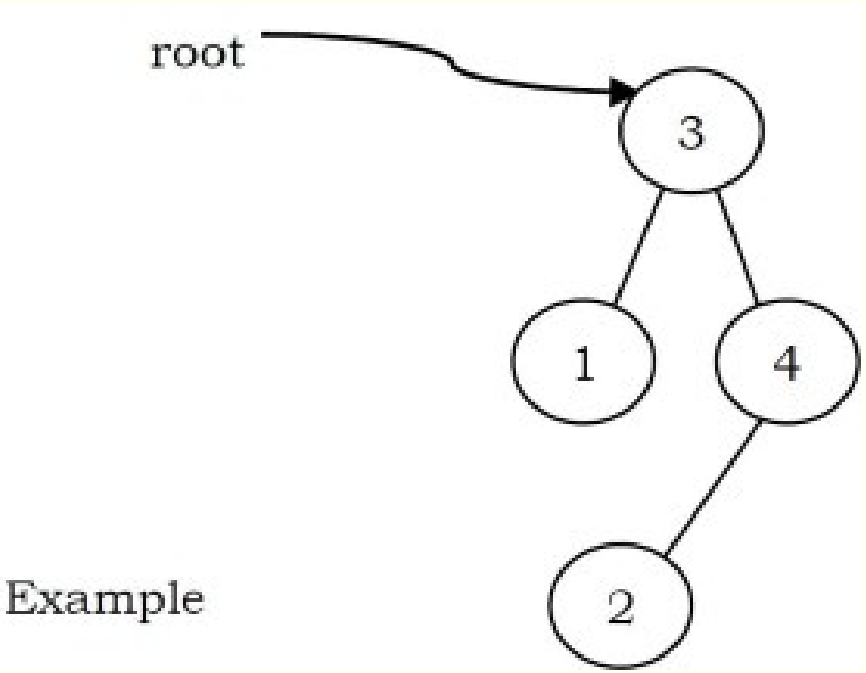


Cây nhị phân

- Cây được gọi là cây nhị phân nếu mỗi nút không có con, một hoặc hai con.
- Cây trống là cũng là một cây nhị phân hợp lệ.
- Chúng ta có thể hình dung một cây nhị phân bao gồm một gốc và hai cây nhị phân tách rời, được gọi là các cây con trái và phải của nút gốc.

Cây nhị phân





Example

Cài đặt cây

- Struct BinaryTreeNode {
  - Int data;
  - Struct BinaryTreeNode * left;
  - Struct BinaryTreeNode * right;
  - }

Các phép toán

- Phép toán cơ bản
  - Chèn một phần tử vào một cái cây
  - Xóa một phần tử khỏi cây
  - Tìm kiếm một phần tử
  - Duyệt cây
- Các phép toán bổ trợ
  - Tìm kích thước của cây
  - Tìm chiều cao của cây
  - Tìm mức có tổng tối đa
  - Tìm kiếm tổ tiên chung ít nhất cho một cặp nút đã cho
  - v..v

Một số ứng dụng của cây nhị phân

- Huffman mã hóa cây được sử dụng trong các thuật toán nén dữ liệu.
- Cây tìm kiếm nhị phân
- Hàng đợi ưu tiên

Các cách duyệt cây

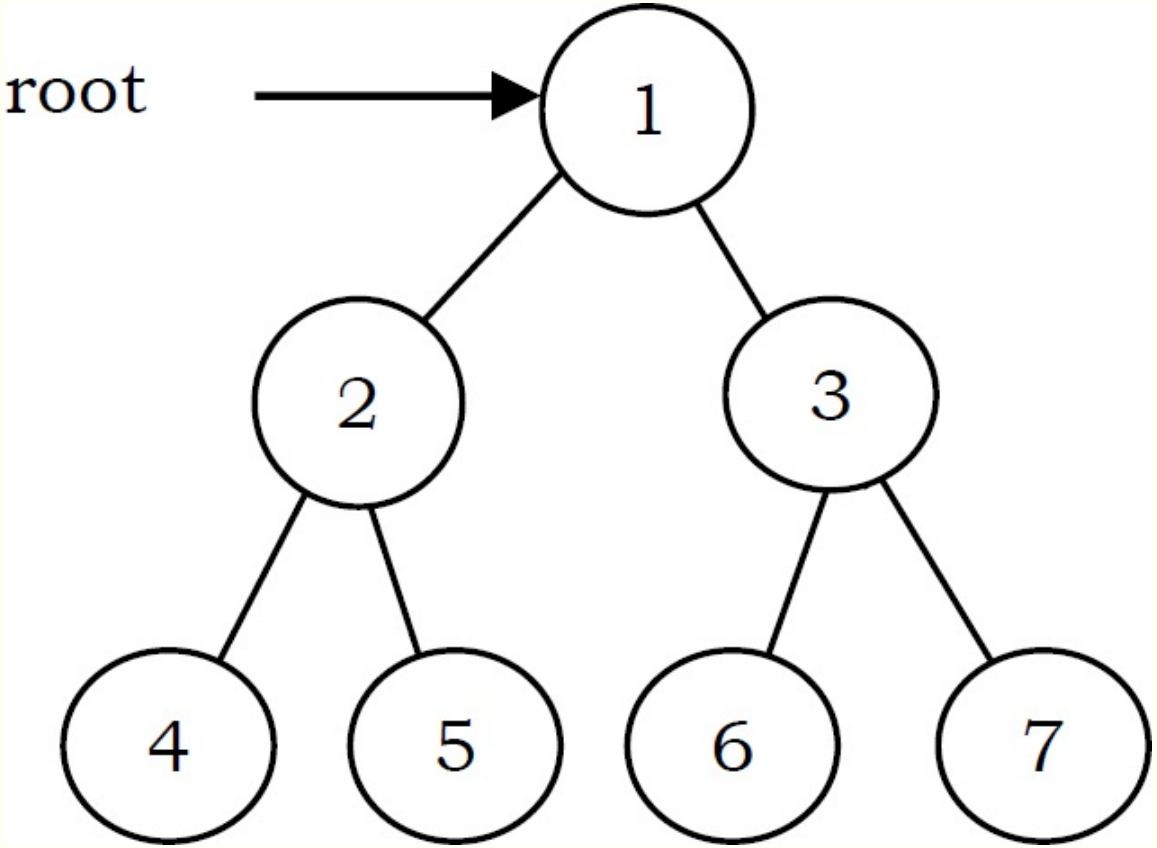
- 1. LDR: Trái, Gốc, Phái
- 2. LRD: Trái, Phái, Gốc
- 3. DLR: Gốc, Trái, Phải
- 4. DRL: Gốc, Phải, Trái
- 5. RDL:
- 6. RLD:

Các cách duyệt cây cổ điển

- 1. Preorder : DLR
- 2. Inorder: LDR
- 3. Postorder: LRD

Các cách duyệt cây - Preorder

1, 2, 4,5,3,6,7

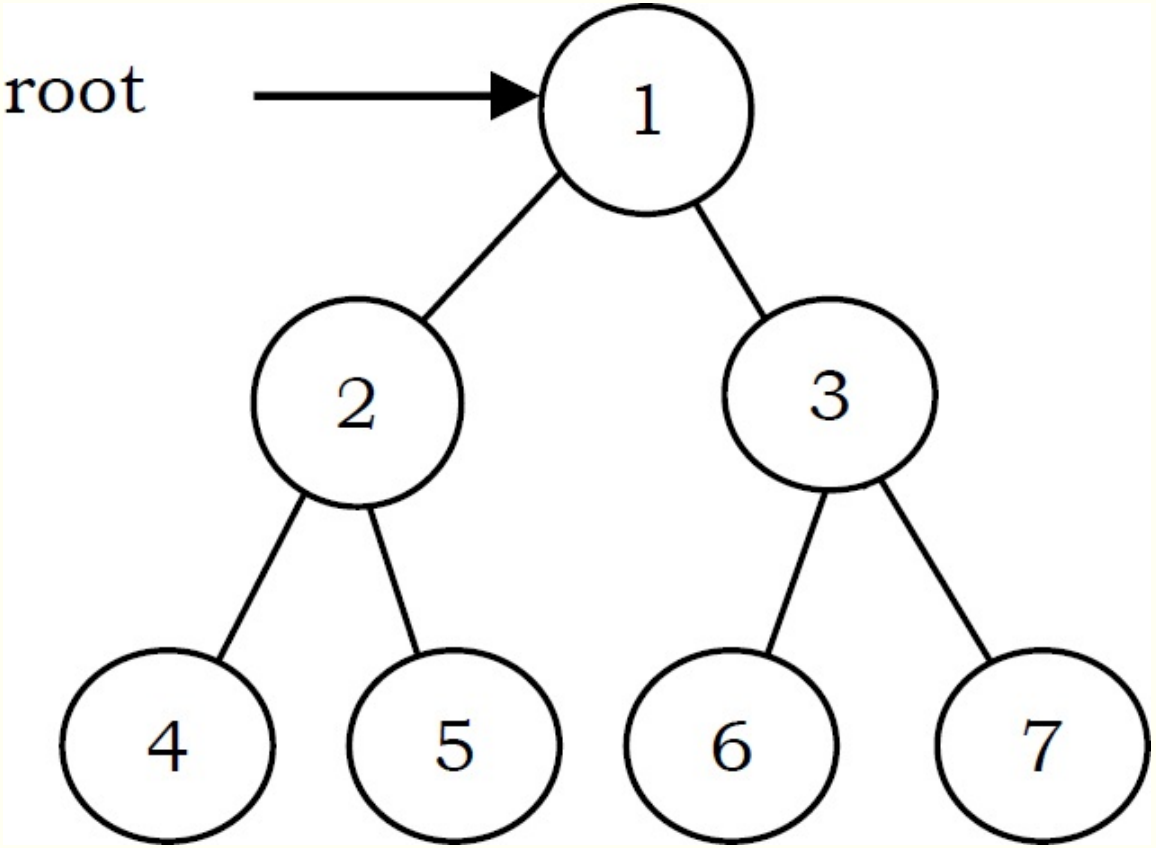


Các cách duyệt cây - Preorder

```
void PreOrder(struct BinaryTreeNode *root){
    if(root) {
        printf("%d",root→data);
        PreOrder(root→left);
        PreOrder (root→right);
    }
}
```

Các cách duyệt cây - Inorder

4 2 5 1 6 3 7

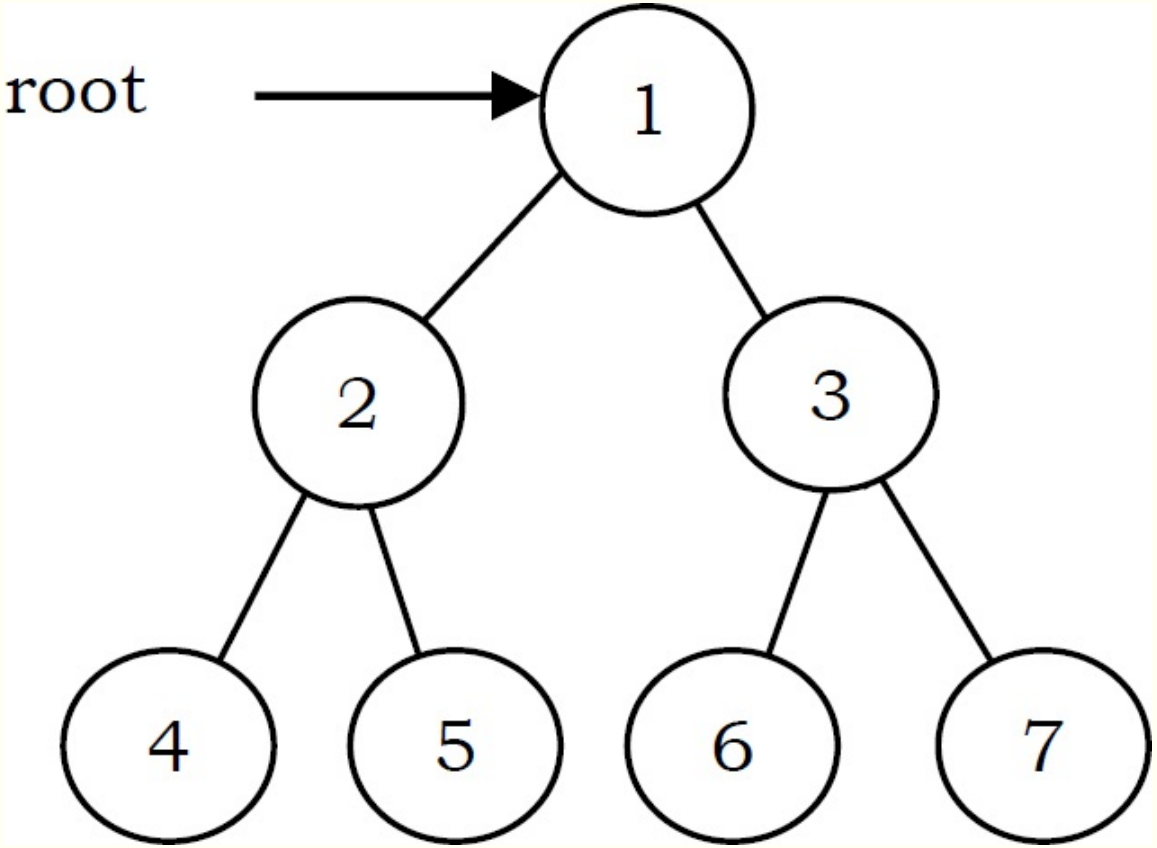


Các cách duyệt cây - Preorder

```
void InOrder(struct BinaryTreeNode *root){
    if(root) {
        InOrder(root→left);
        printf("%d",root→data);
        InOrder(root→right);
    }
}
```

Các cách duyệt cây - Postorder

4 5 2 6 7 3 1

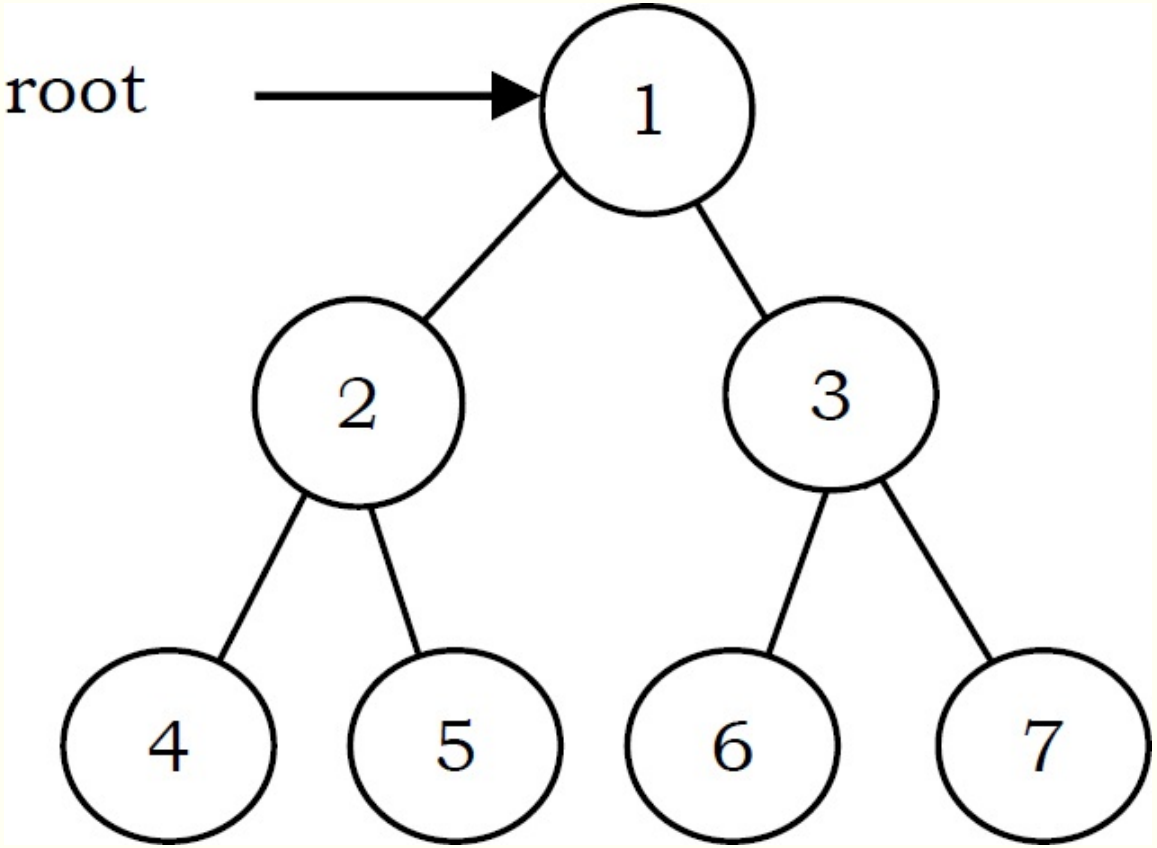


Các cách duyệt cây - Preorder

```
void PostOrder(struct BinaryTreeNode *root){
    if(root)        {
        PostOrder(root→left);
        PostOrder(root→right);
        printf("%d",root→data);
    }
}
```

Các cách duyệt cây – Level Order

1 2 3 4 5 6 7



Các cách duyệt cây – Level Order

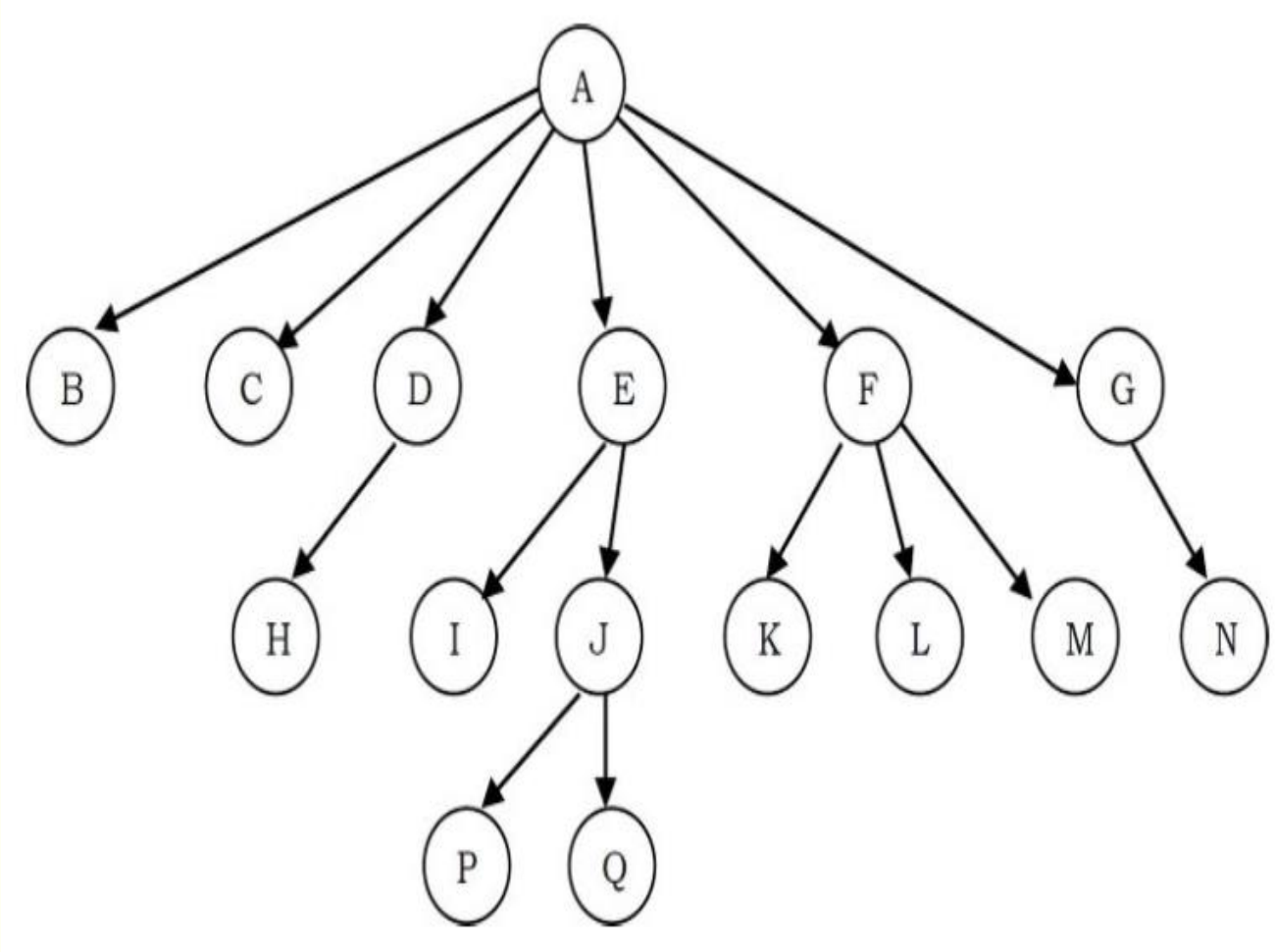
```
void LevelOrder(struct BinaryTreeNode *root){
    struct BinaryTreeNode *temp;
    struct Queue *Q = CreateQueue();
    if(!root)
        return;
    EnQueue(Q,root);
    while(!IsEmptyQueue(Q)) {
        temp = DeQueue(Q);
        //Process current node
        printf("%d", temp→data);
        if(temp→left)
            EnQueue(Q, temp→left);
        if(temp→right)
            EnQueue(Q, temp→right);
    }
    DeleteQueue(Q);
}
```

Cây tìm kiếm nhị phân

- Khóa con trái nhỏ hơn nút gốc
- Khóa nút gốc nhỏ hơn con phải
- Cả hai cây con trái và phải cũng phải là cây tìm kiếm nhị phân.
  - Trình bày chi tiết trong phần các phương pháp tìm kiếm (Tuần tiếp theo)

Cây tổng quát

- Một nút có nhiều con



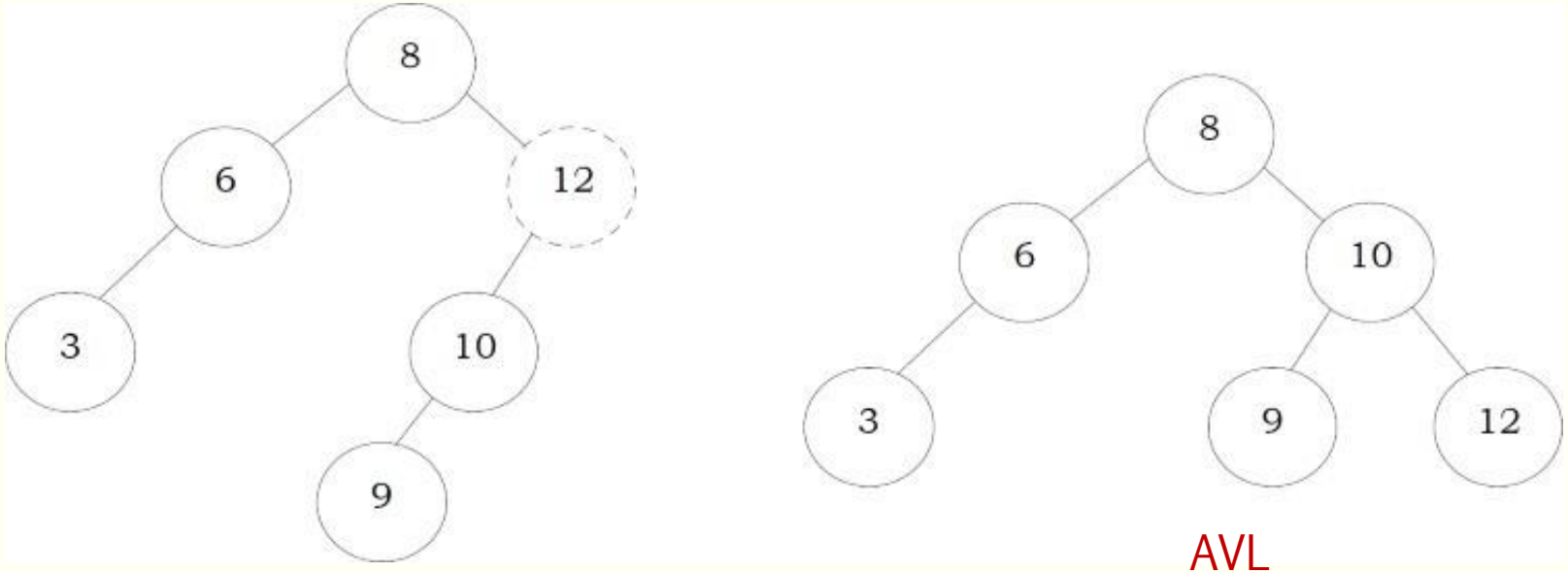
Cây tổng quát

- Biểu diễn cây:
  - Danh sách con của mỗi đỉnh: Tìm bố, tìm con trưởng
  - Danh sách (hoặc Mảng) các nút bố
    - Phép toán cần cài đặt:
      - Tìm con trưởng
      - Tìm em liền kề
  - Danh sách (hoặc mảng) con trưởng và em liền kề
    - Phép toán:
      - Tìm bố

```
struct TreeNode {
    int data;
    struct TreeNode *firstChild;
    struct TreeNode *nextSibling;
};
```

Cây AVL

- Cây nhị phân tìm kiếm cân bằng là cây mà tại mỗi nút của nó độ cao của cây con trái và của cây con phải chênh lệch không quá một.

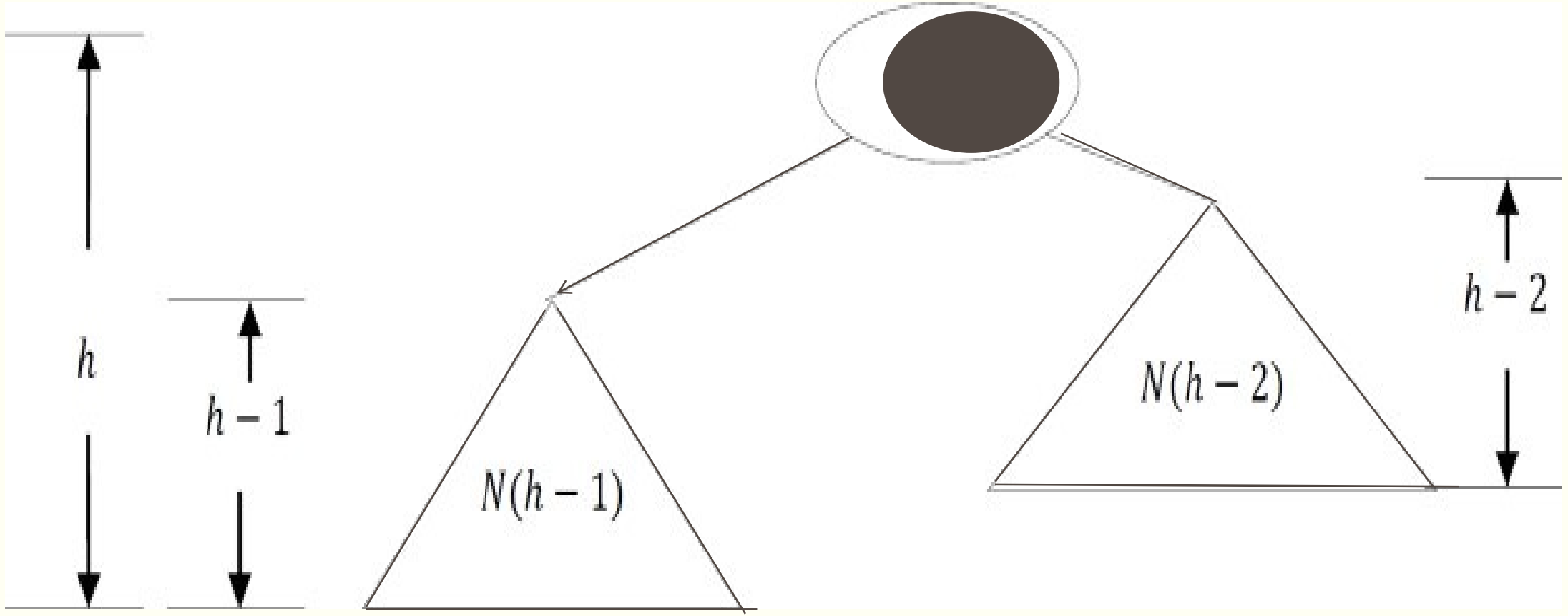


AVL

Chiều cao của cây AVL

- Để có số lượng nút tối thiểu có chiều cao h, chúng ta nên điền vào cây với số lượng nút tối thiểu có thể.
- Điều đó có nghĩa là nếu chúng ta lấp đầy cây con trái với chiều cao h - 1 sau đó chúng ta điền vào các cây con phải với chiều cao h - 2.
- Kết quả là, tối thiểu số nút có chiều cao h là:
  - $N(h) = N(h - 1) + N(h - 2) + 1$

Chiều cao của cây AVL



Cấu trúc cây AVL

```
struct AVLTreeNode{
    struct AVLTreeNode *left;
    int data;
    struct AVLTreeNode *right;
    int height;
};
```

Các phép toán cơ bản trên cây AVL

- Thêm một phần tử vào cây AVL.
- Hủy một phần tử trên cây AVL.
- Cân bằng lại một cây vừa bị mất cân bằng.

Xử lý các trường hợp cây mất cân bằng

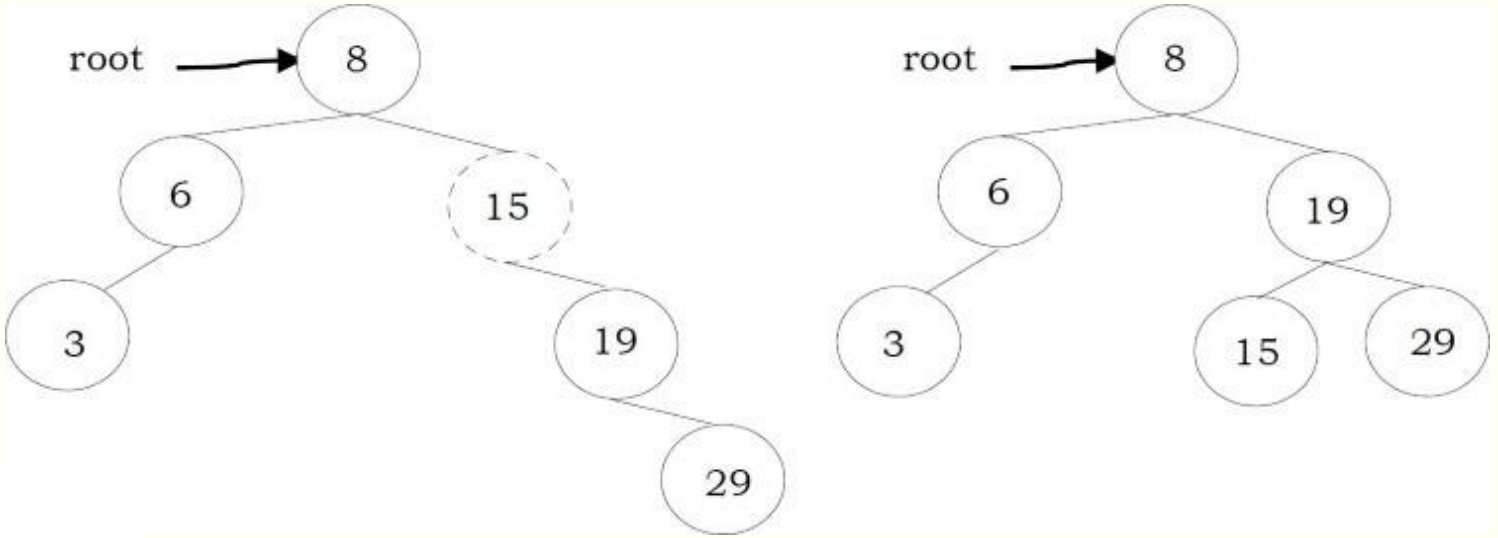
- Các trường hợp xung đột:
  - Chèn vào cây con trái của con trái của X. (1)
  - Chèn vào cây con bên phải của con trái của X. (2)
  - Chèn vào cây con trái của con phải của X.(3)
  - Chèn vào cây con phải của X. (4)

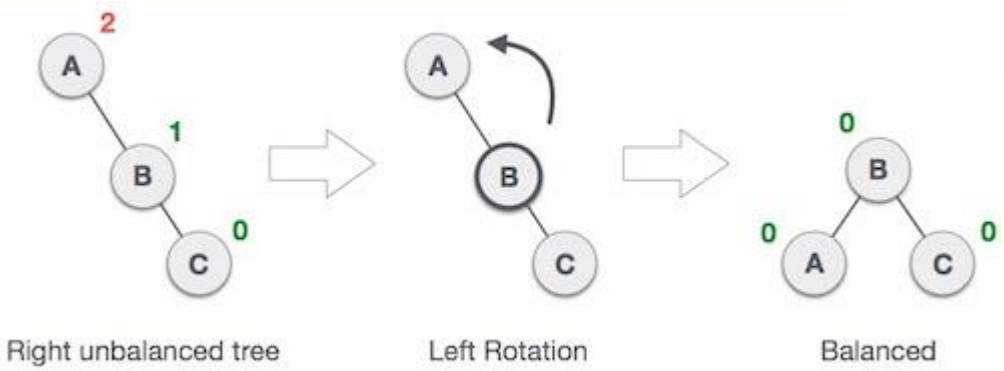
Xử lý các trường hợp cây mất cân bằng

- Lưu ý:
- Trường hợp 1 và 4 là đối xứng và dễ dàng được giải quyết với các phép quay đơn.
- Trường hợp 2 và 3 là cũng đối xứng và có thể được giải quyết với phép quay kép (cần hai phép quay đơn).

Quay đơn (Left Rotation) (1)

6,5,9,3,8,7



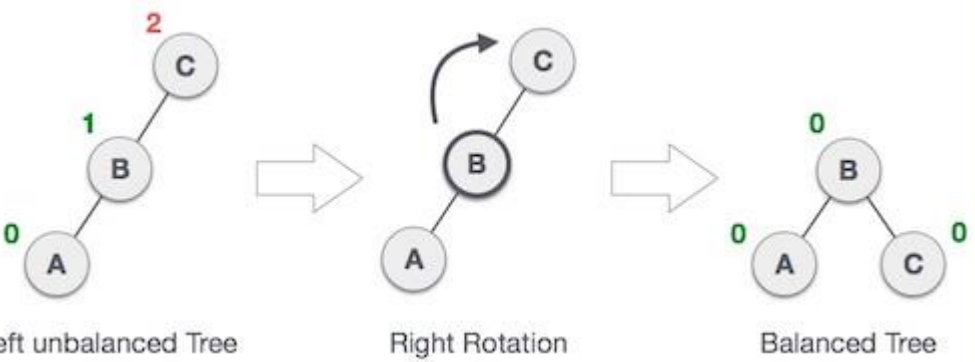


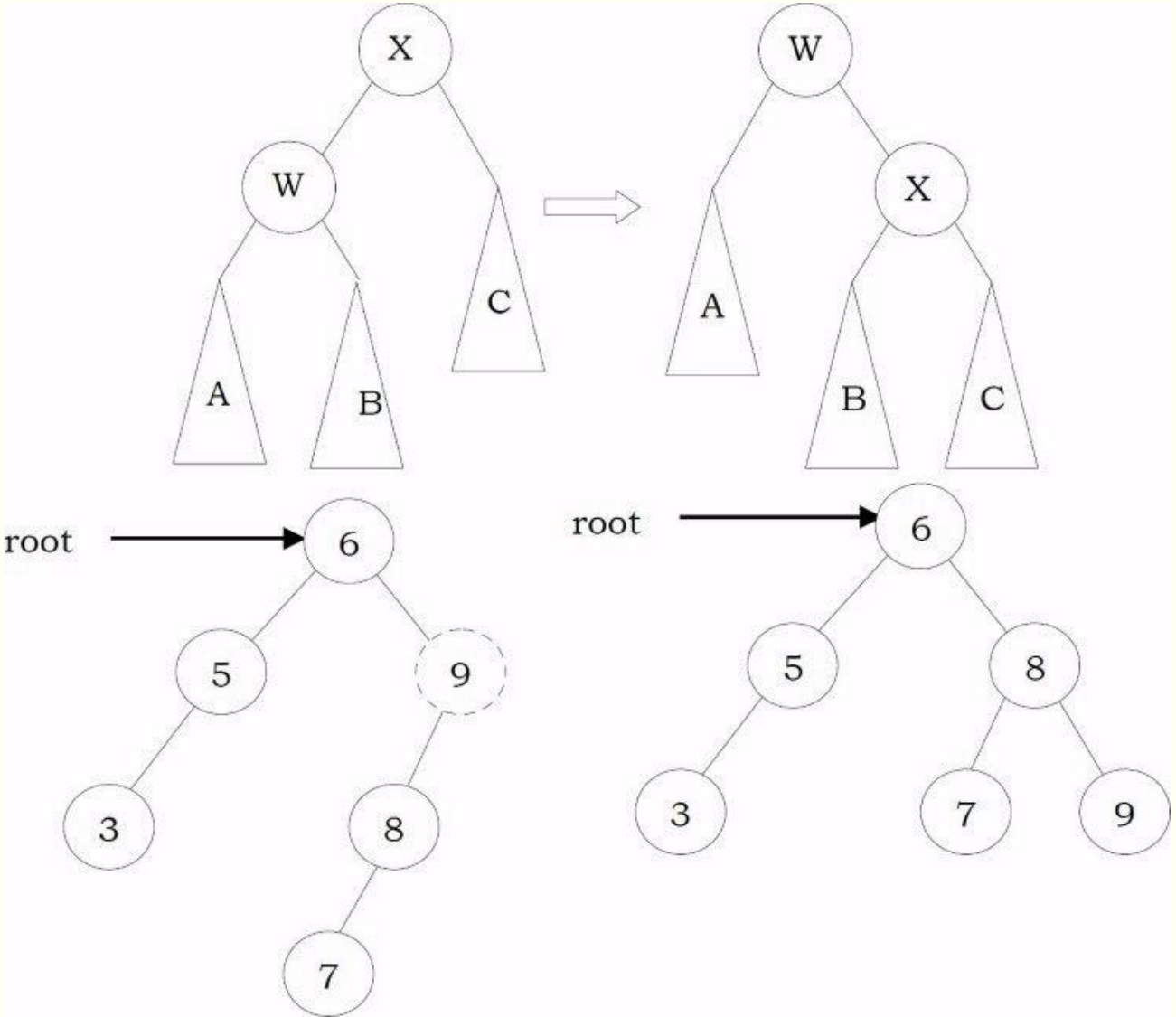
Quay đơn (Left Rotation) (1)

```
struct AVLTreeNode *SingleRotateLeft(struct AVLTreeNode *X ){
    struct AVLTreeNode *W = X→left;
    X→left = W→right;
    W→right = X;
    X→height = max( Height(X→left), Height(X→right) ) + 1;
    W→height = max( Height(W→left), X→height ) + 1;
    return W;  /* New root */
}
```

Quay đơn (Right Rotation) (4)

8,6,3,15,19,29

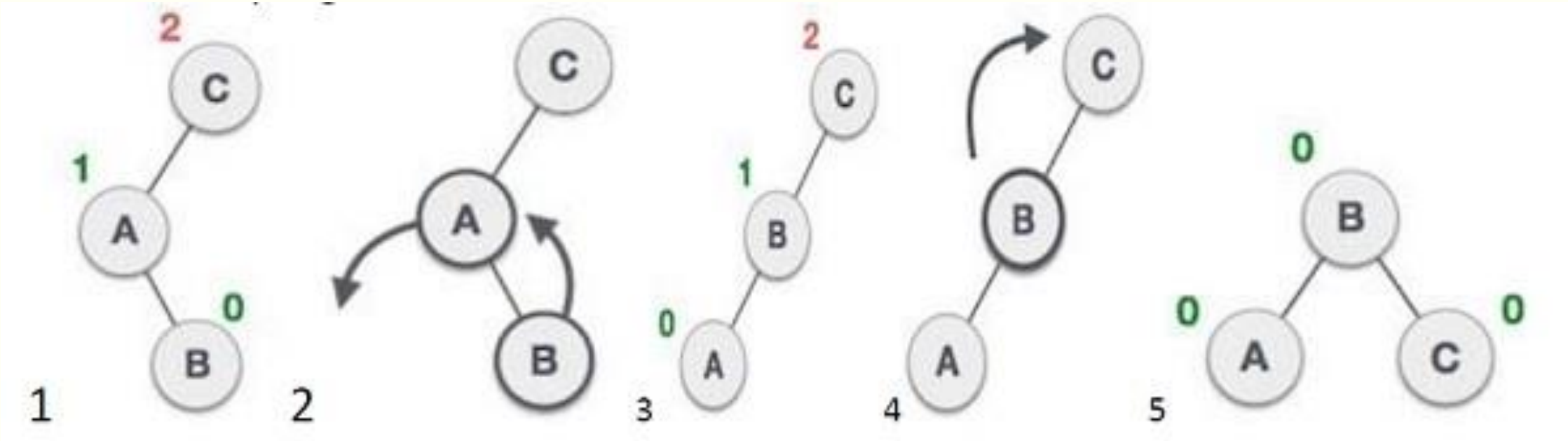




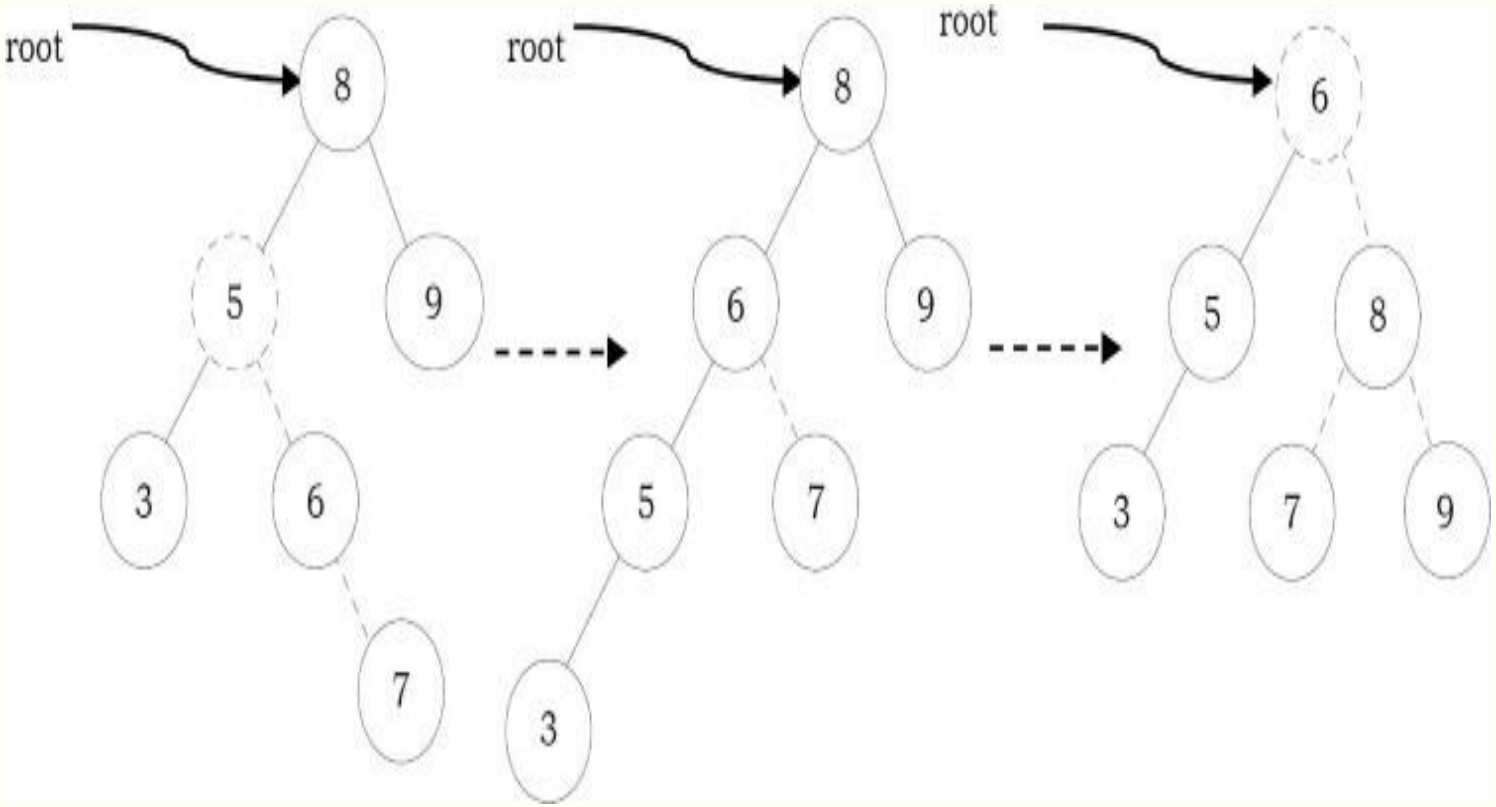
Quay đơn (Right Right Rotation) (1)

```
struct AVLTreeNode *SingleRotateRight(struct AVLTreeNode *W ) {
    struct AVLTreeNode *X  = W→right;
    W→right = X→left;
    X→left = W;
    W→height = max( Height(W→right), Height(W→left) ) + 1;
    X→height = max( Height(X→right), W→height) + 1;
    return X;
}
```

Quay kép (Left Right Rotation) (2)

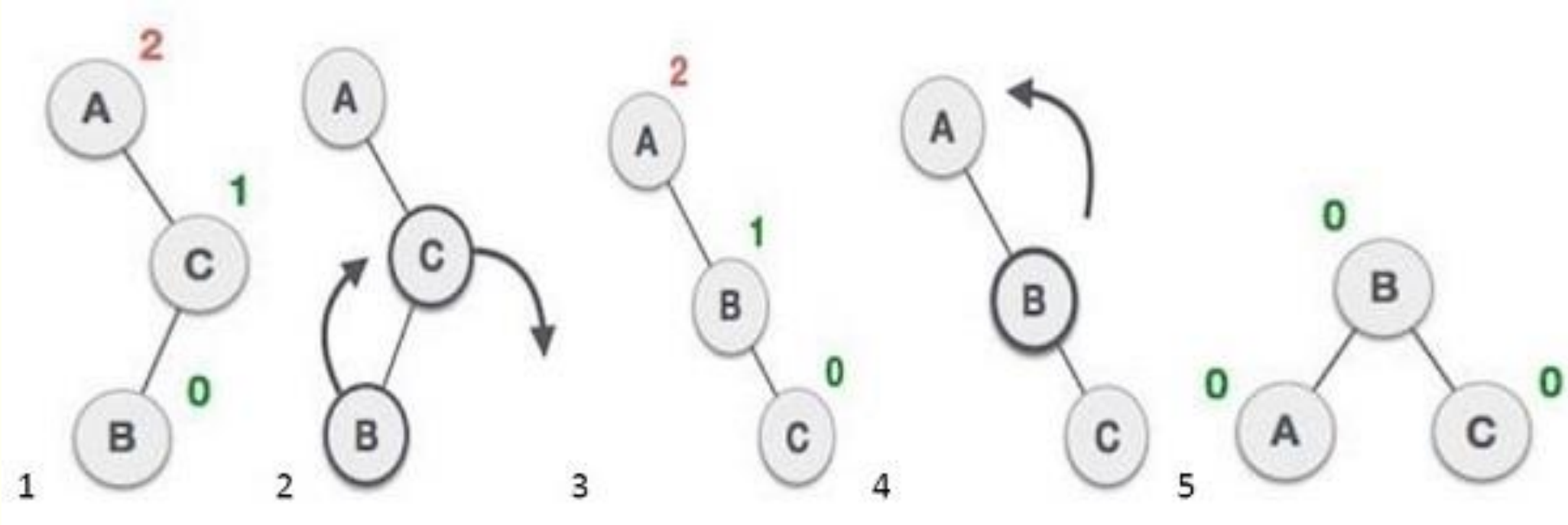


Quay kép (Left Right Rotation) (2)



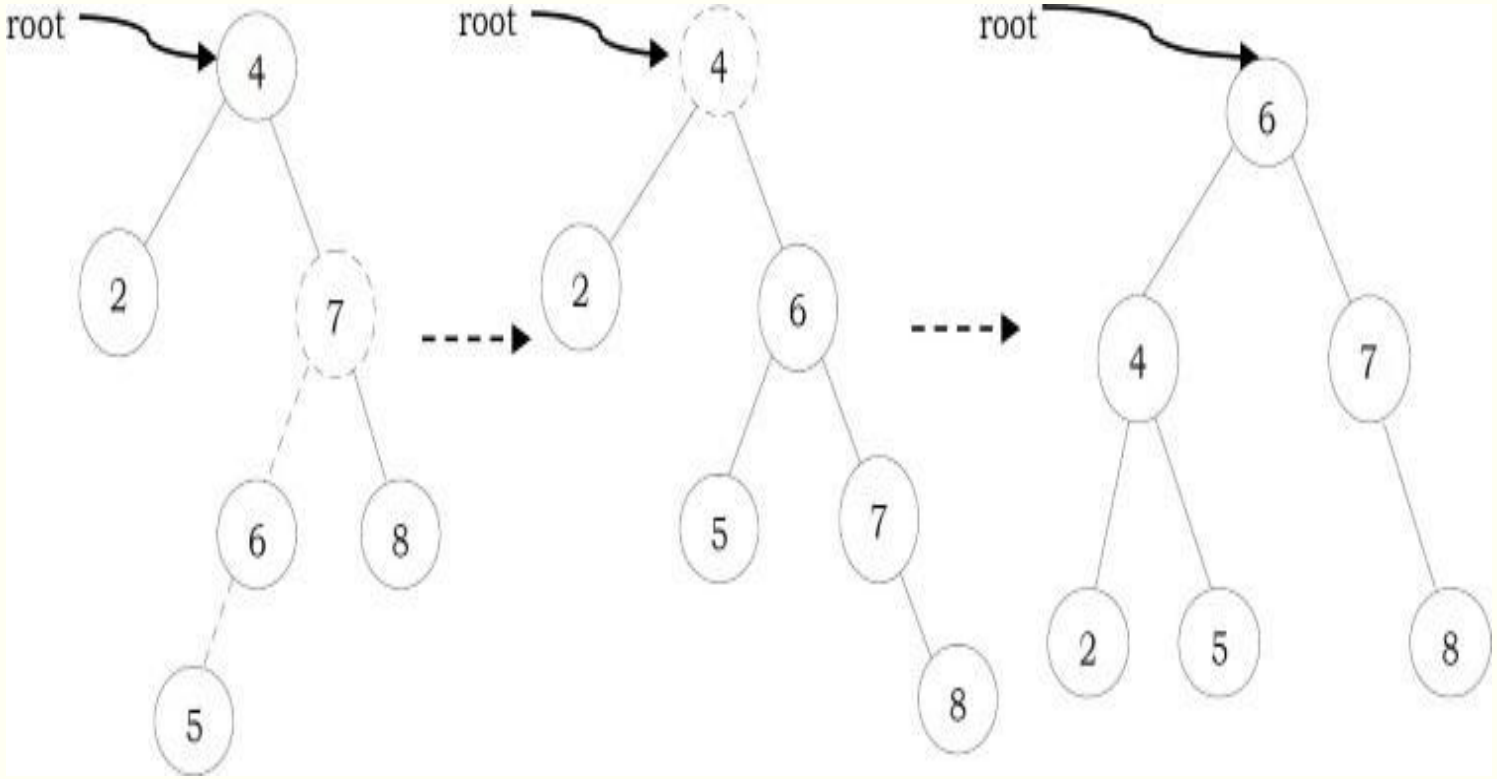
8,5,9,3,6,7

Quay kép (Right Left Rotation) (3)



Quay kép (Right Left Rotation) (3)

4,7,8,6,5,2



Chèn một nút vào trong cây

- Tự nghiên cứu (Bài tập)
- Lưu ý: Việc chèn vào cây AVL tương tự như chèn BST. Sau khi chèn phần tử, chúng ta chỉ cần để kiểm tra xem có bất kỳ sự mất cân bằng chiều cao nào không. Nếu có sự mất cân bằng, thực hiện chức năng xoay phù hợp.

Tổng kết

- Cấu trúc phi tuyến, dữ liệu phân cấp:
  - Cây nhị phân
  - Cây tổng quát
  - Cây cân bằng

Tiếp theo …

- Các chiến lược Tìm kiếm