Data Structure & Algorithm

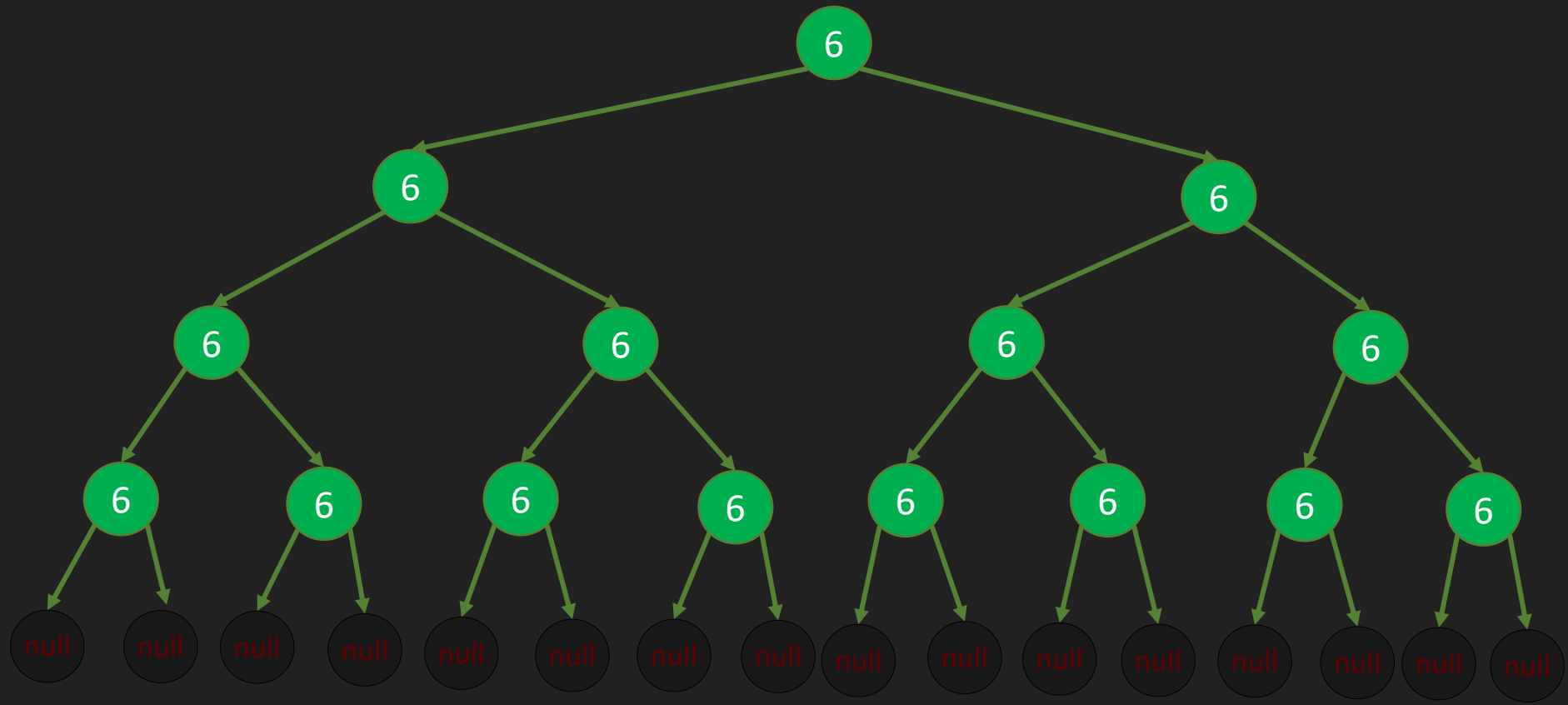
Data Structure

Tree - Binary Tree - Binary Search Tree

1. Định nghĩa

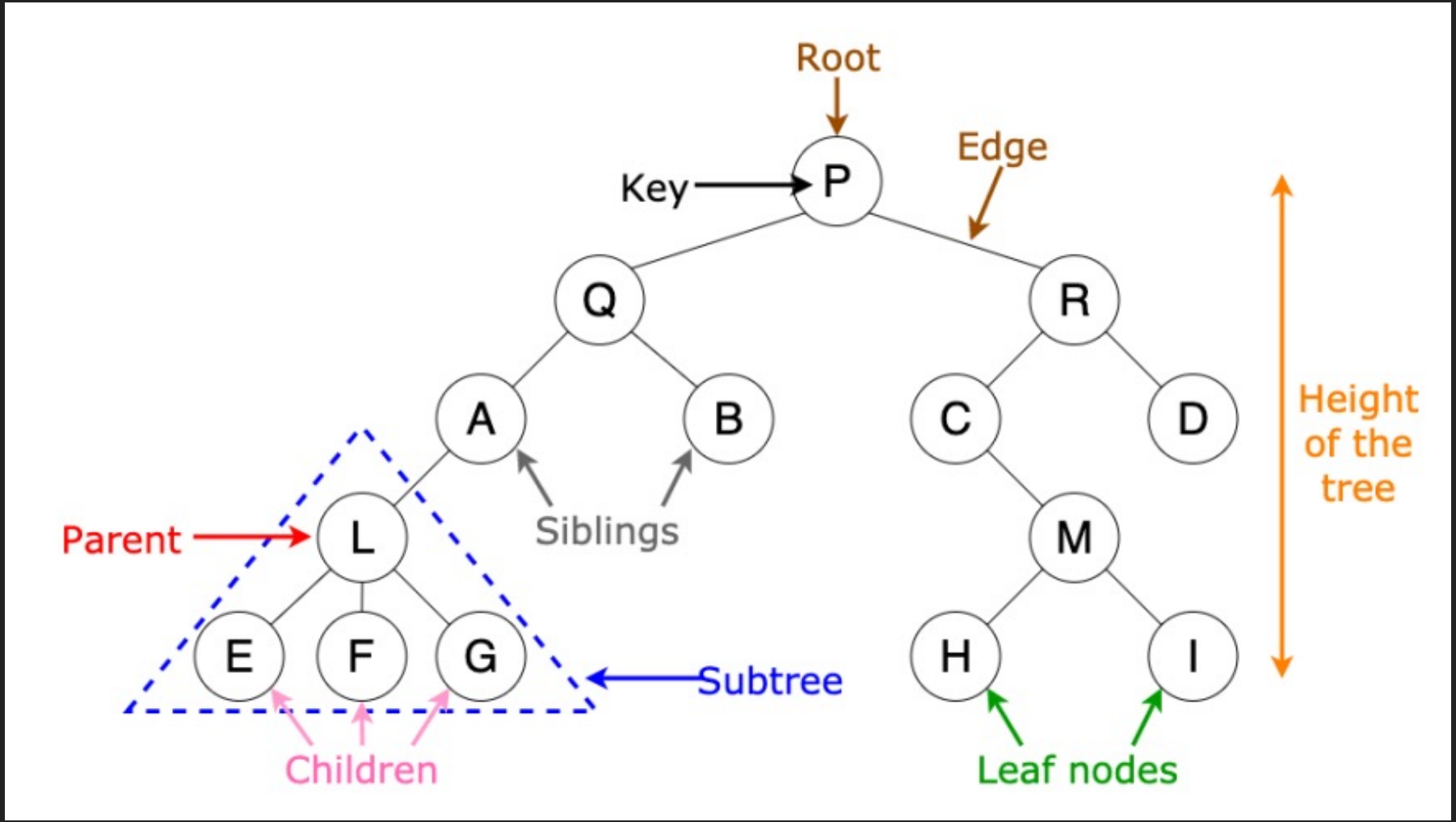
Linked List

Binary Tree



1. Định nghĩa

- Tree

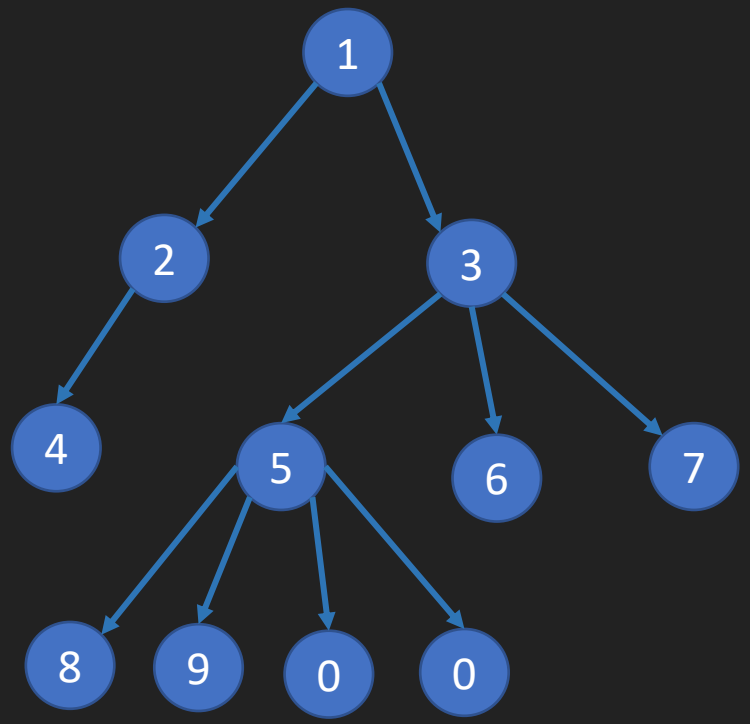


2. Ứng dụng của Tree trong thực tế

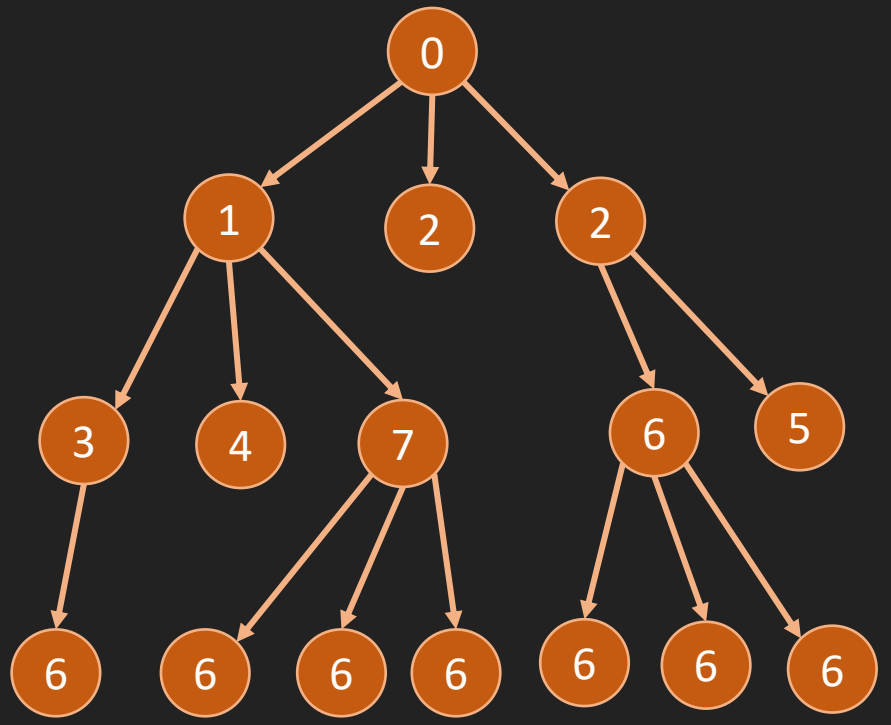
- ✓ Cây thư mục trong ổ cứng máy tính
- ✓ Mục lục của một quyển sách
- ✓ Giả phả của một dòng họ
- ✓ Biểu thức toán học
- ✓ ....

3. Phân loại

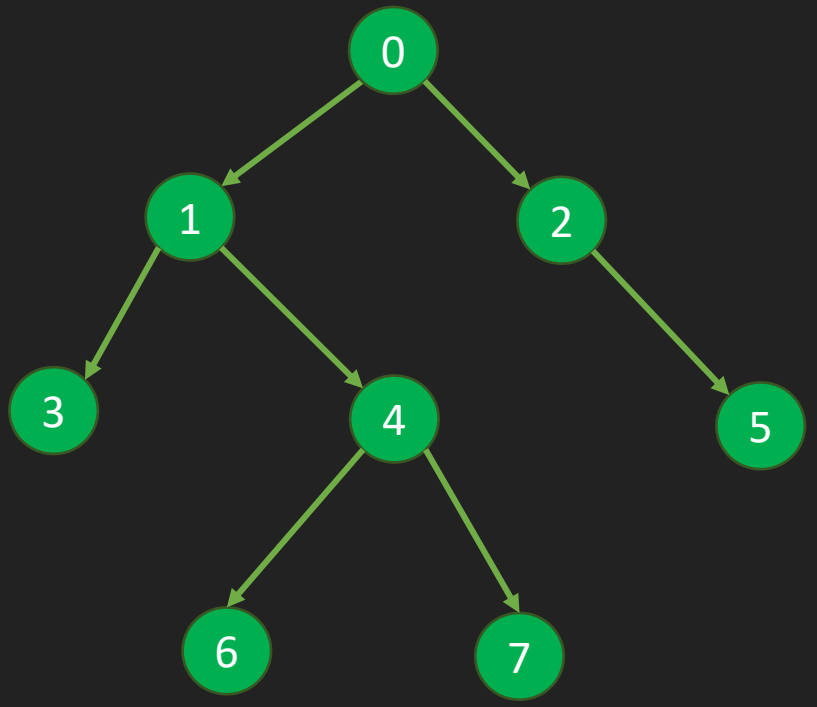
❖ Dựa trên số child node



➢ **Tree**
(N-ary Tree)



➢ **Ternary Tree**



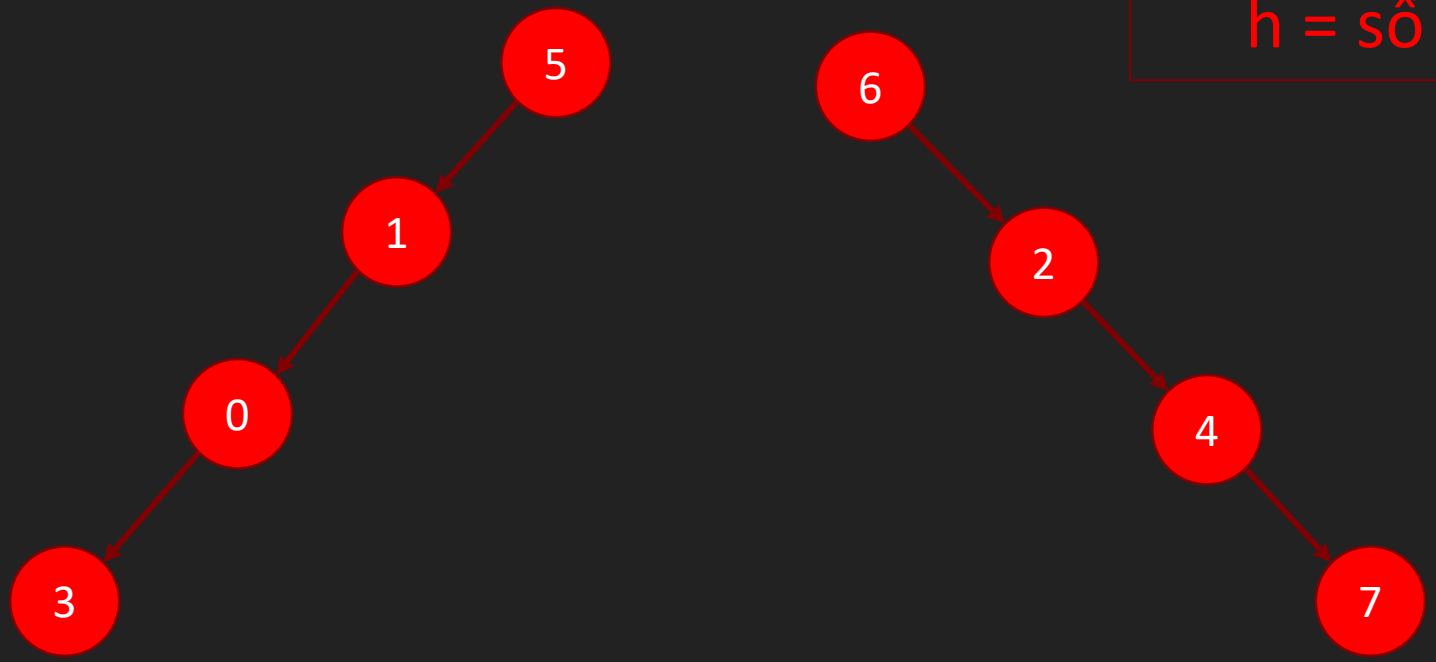
➢ **Binary Tree***

3. Phân loại

❖ Dựa trên tính chất của cây

➢ Cây nhị phân suy biến:

- Suy biến thành linked list.
- Chiều cao của cây:
  h = số node - 1.

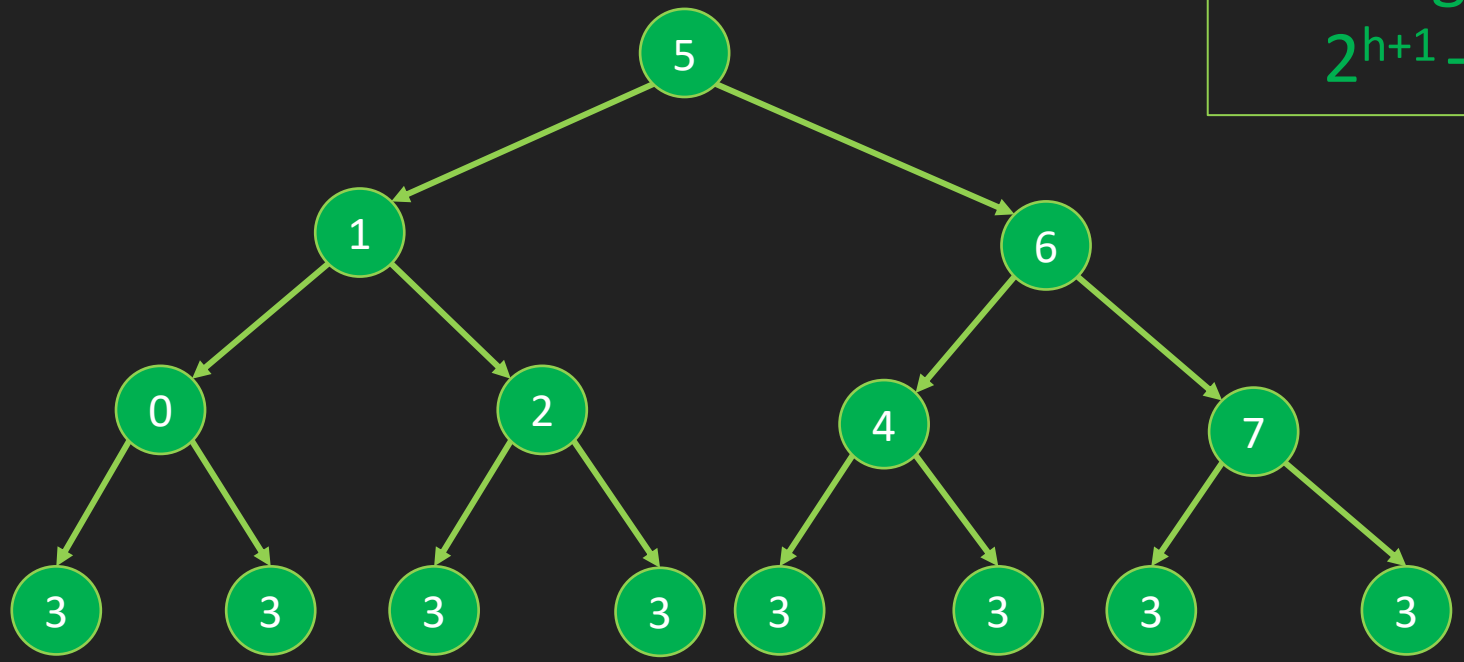


3. Phân loại

❖ Dựa trên tính chất của cây

➢ Cây nhị phân hoàn chỉnh:

- Số node ở độ cao h là 2^h
- Tổng số node ở cây độ cao h là: $2^{h+1} - 1$.

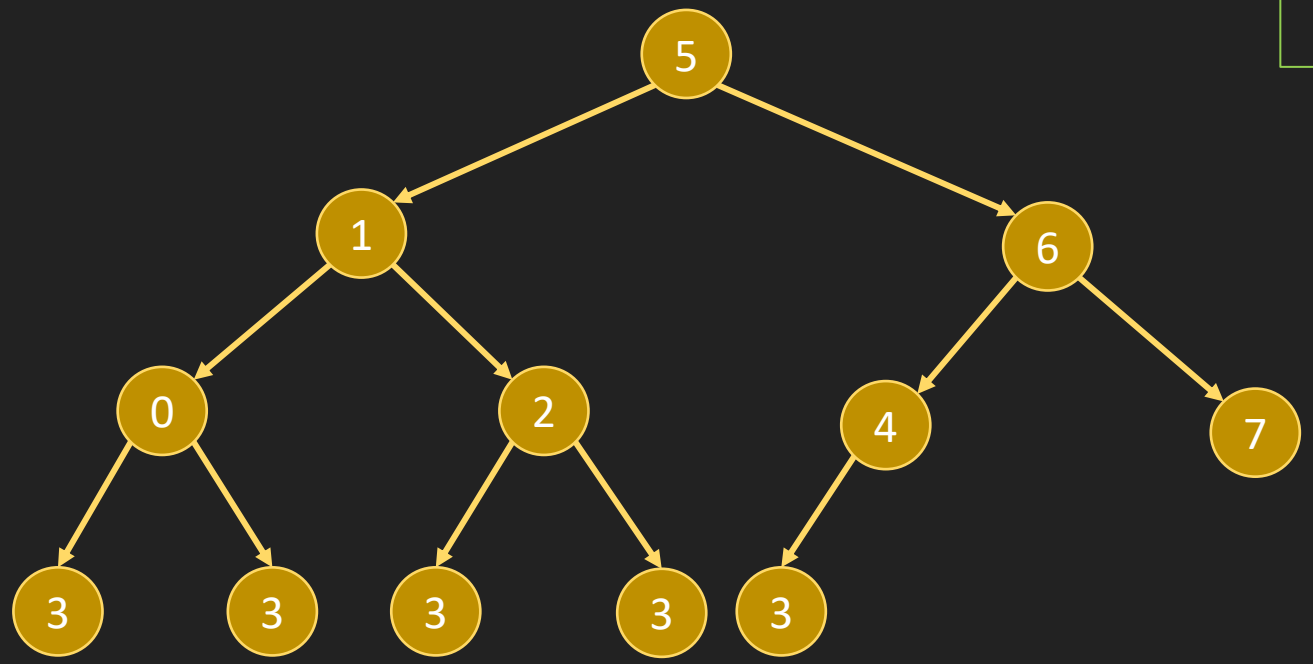


3. Phân loại

❖ Dựa trên tính chất của cây

➢ Cây nhị phân gần hoàn chỉnh:

- Nếu bỏ hết node ở độ cao h, ta thu được 1 ***cây hoàn chỉnh***.

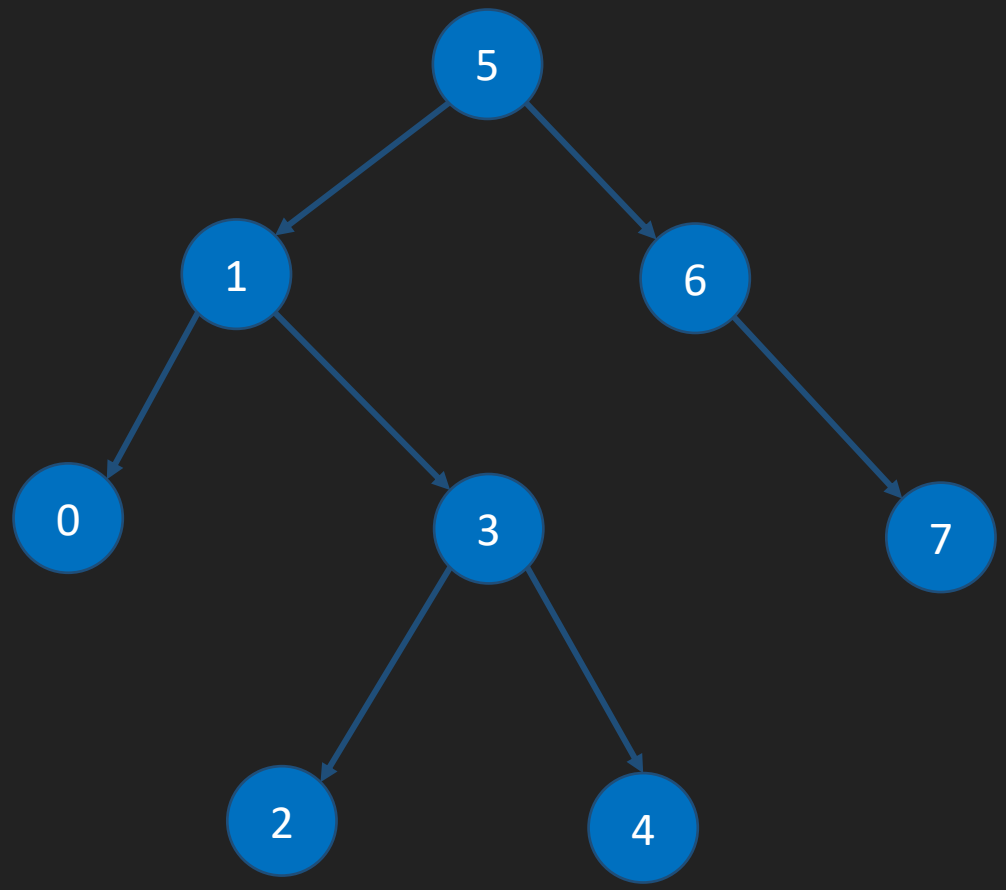


3. Phân loại

❖ Dựa trên tính chất của cây

➢ Binary Search Tree (BST):

(Cây nhị phân tìm kiếm)



- ✓ Tất cả các node con bên trái đều nhỏ hơn node hiện tại.
- ✓ Tất cả các node con bên phải đều lớn hơn node hiện tại.

❑ Tác dụng: Có yếu tố sắp xếp, tạo thuận lợi cho bài toán tìm kiếm.

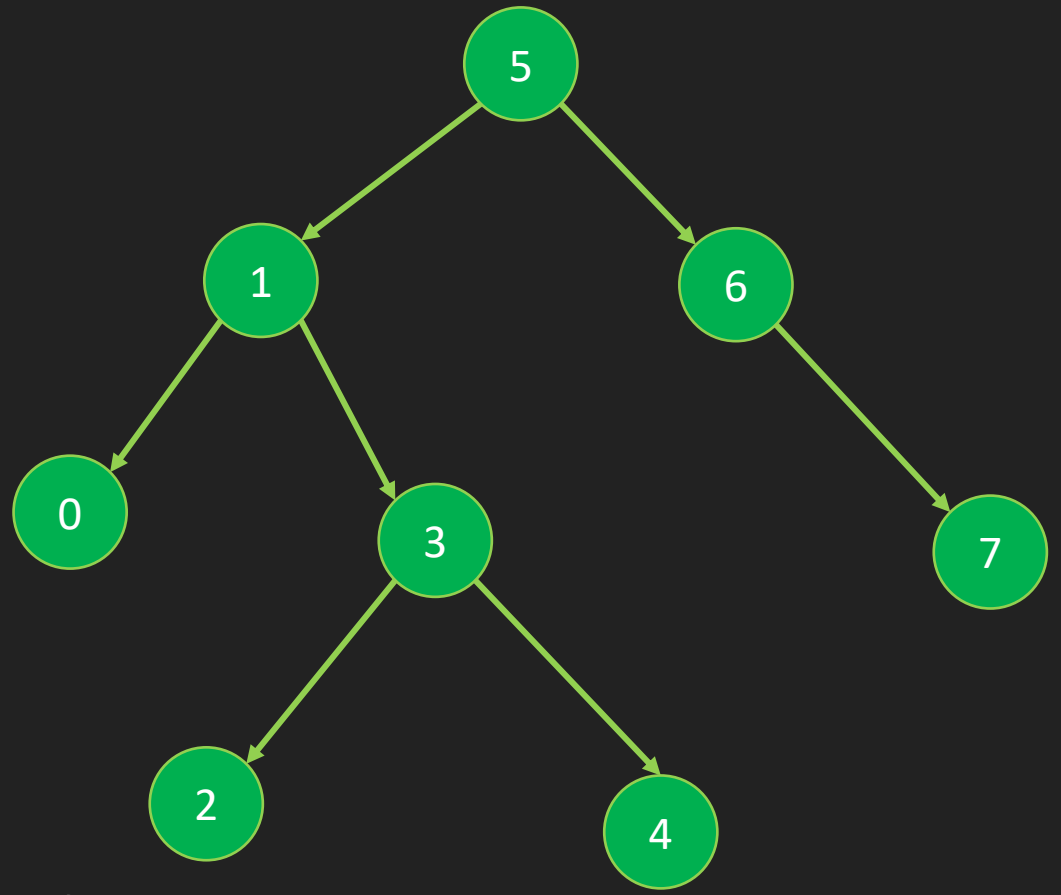
3. Phân loại

❖ Dựa trên tính chất của cây

➢ AVL Tree (Self-balancing BST):

(Cây nhị phân tìm kiếm tự cân bằng)

- Cây AVL là cây BST mà tại 1 node bất kỳ chiều cao của cây con trái và cây con phải lệch nhau không quá 1 đơn vị.



❑ Tác dụng: Giúp cho cây luôn có chiều cao thích hợp để xử lý bài toán.

3. Phân loại

- N-ary Tree
- Binary Tree *
- Binary Search Tree *
- AVL Tree
- Heap
- Trie
- ...

4. Biểu diễn cây nhị phân

- Sử dụng cấu trúc liên kết

```
public class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;
}
```

- Sử dụng cấu trúc mảng

5. Các thao tác trên cây (BST)

- Xây dựng / tạo cây
- Thêm, xoá Node trên cây
- Tìm kiếm trên cây *
- Khảo sát / Duyệt cây **

5. Các thao tác trên cây (BST)

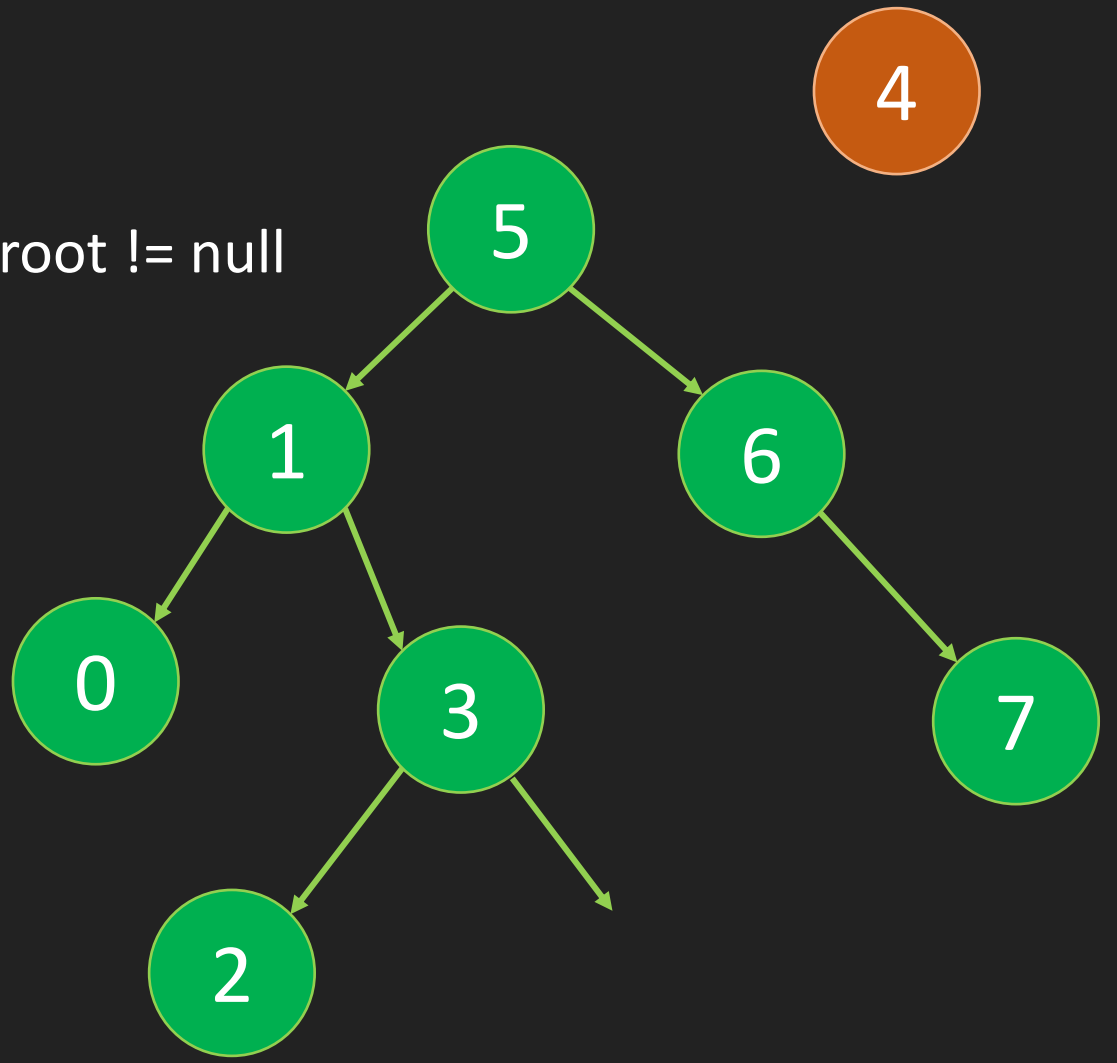
- Xây dựng / tạo cây

5. Các thao tác trên cây (BST)

- Thêm node vào cây

✓ TH1: root = null

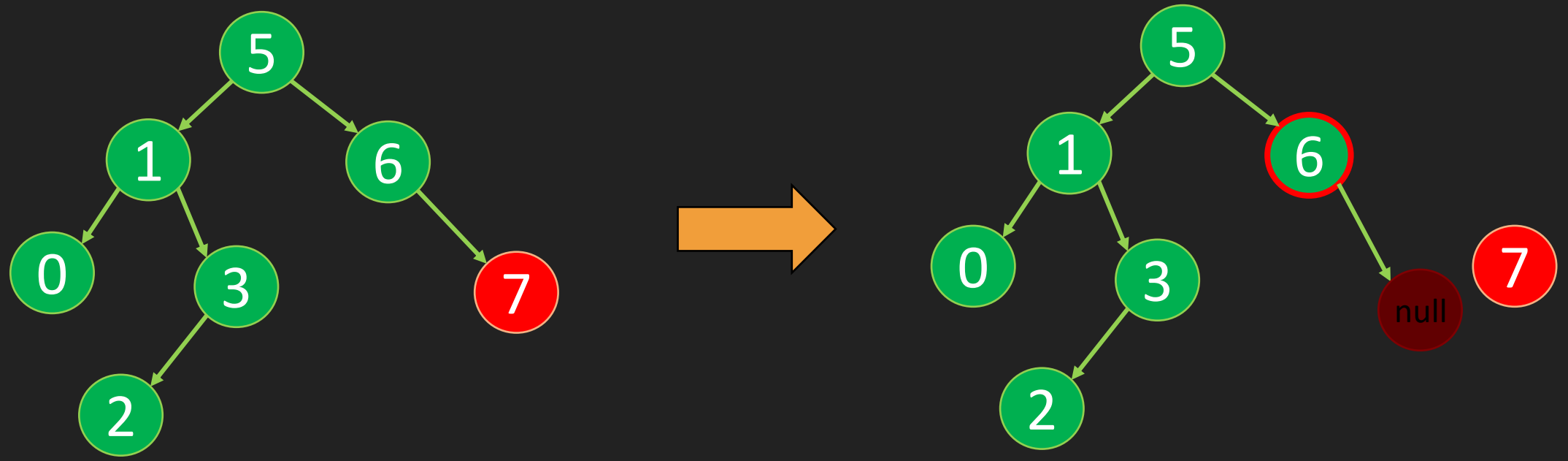
✓ TH2: root != null



- Practice: ✓ **701. Insert into a Binary Search Tree**

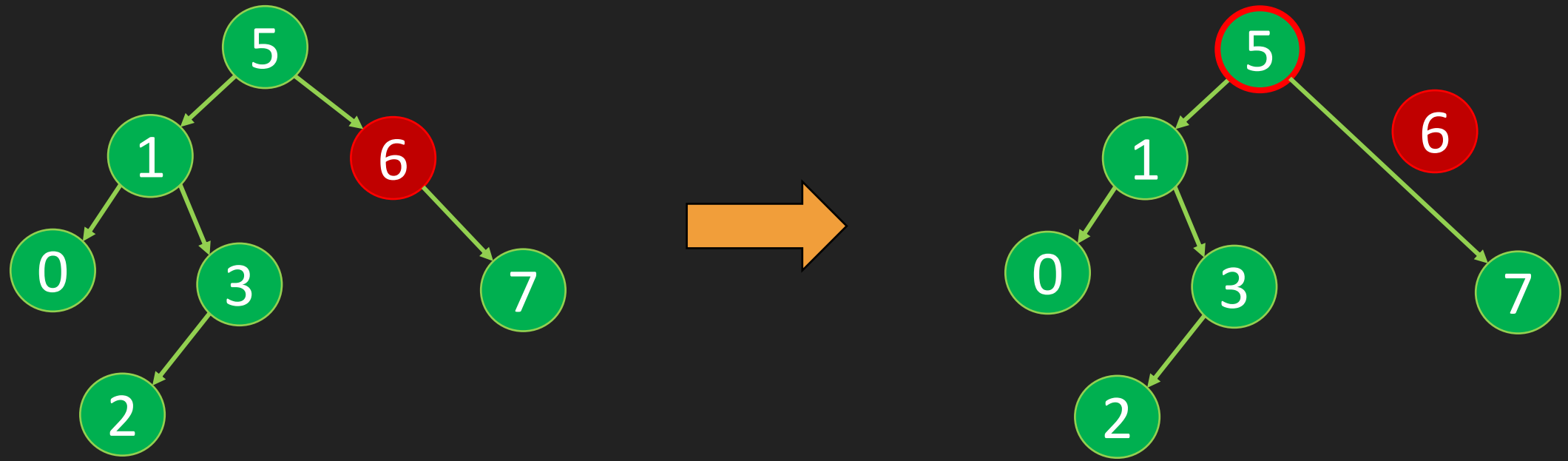
5. Các thao tác trên cây (BST)

- Xoá node trong cây
  - B1: Xác định node x cần xoá
  - B2: Xoá node x
    - ✓ TH1: x không có node con (node lá):



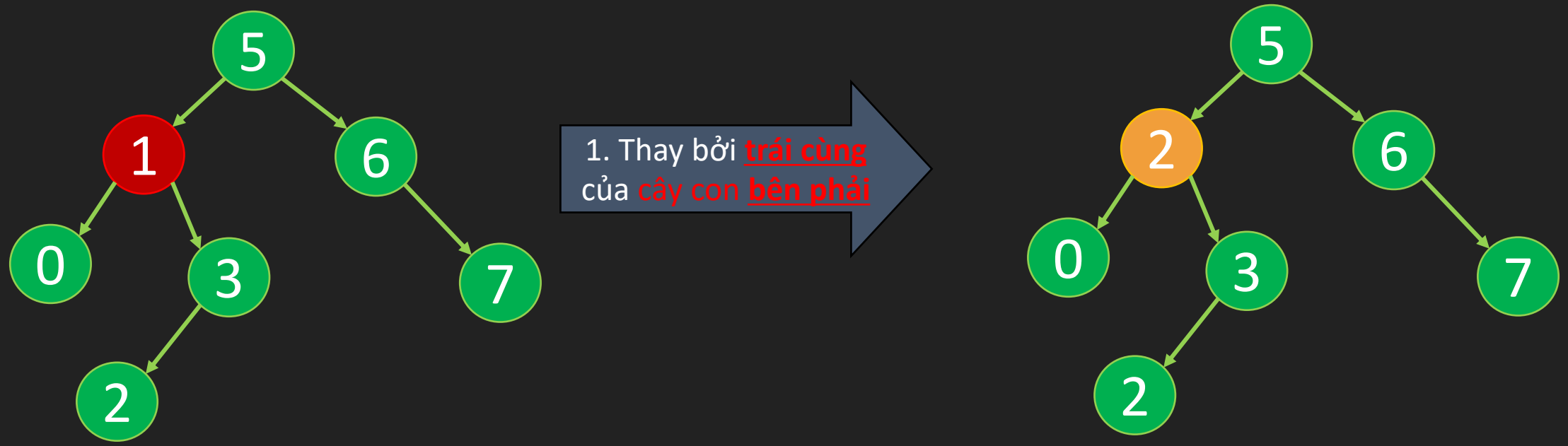
5. Các thao tác trên cây (BST)

- Xoá node trong cây
  - B1: Xác định node x cần xoá
  - B2: Xoá node x
    - ✓ TH2: x **có một** node con:



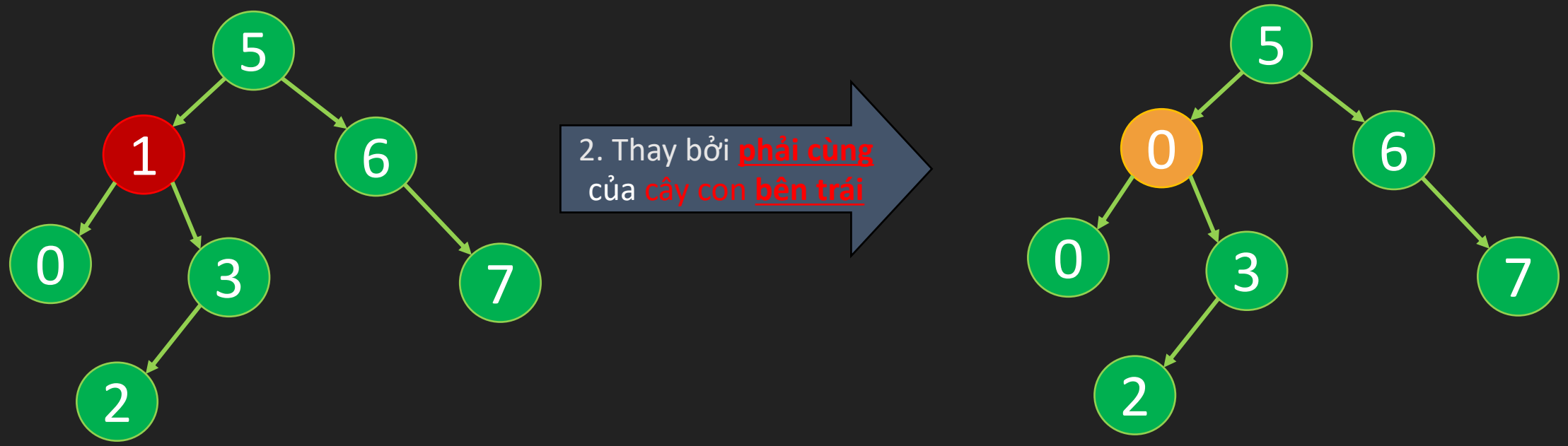
5. Các thao tác trên cây (BST)

- Xoá node trong cây
  - B1: Xác định node x cần xoá
  - B2: Xoá node x
    - ✓ TH3: x **có hai** node con:



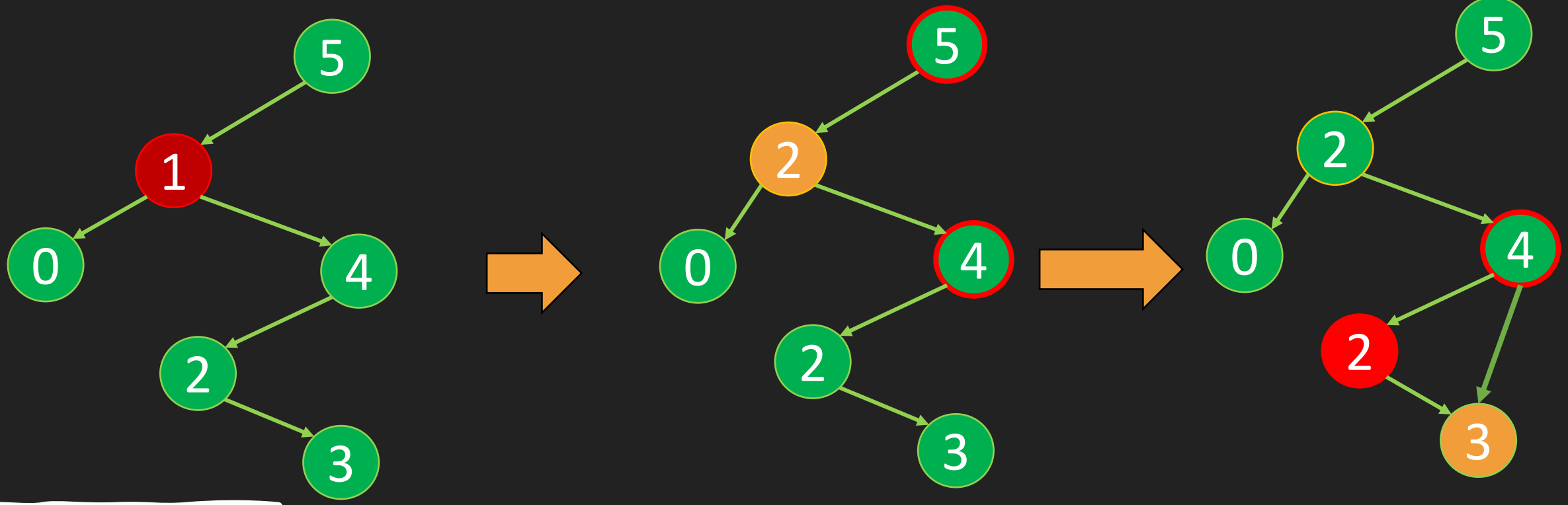
5. Các thao tác trên cây (BST)

- Xoá node trong cây
  - B1: Xác định node x cần xoá
  - B2: Xoá node x
    - ✓ TH3: x **có hai** node con:



5. Các thao tác trên cây (BST)

- Xoá node trong cây
  - B1: Xác định node x cần xoá
  - B2: Xoá node x
    - ✓ TH3: x **có hai** node con:



5. Các thao tác trên cây (BST)

- Tìm kiếm node trong cây

- Practice:
  - ✓ **700. Search in a Binary Search Tree**

5. Các thao tác trên cây (BST)

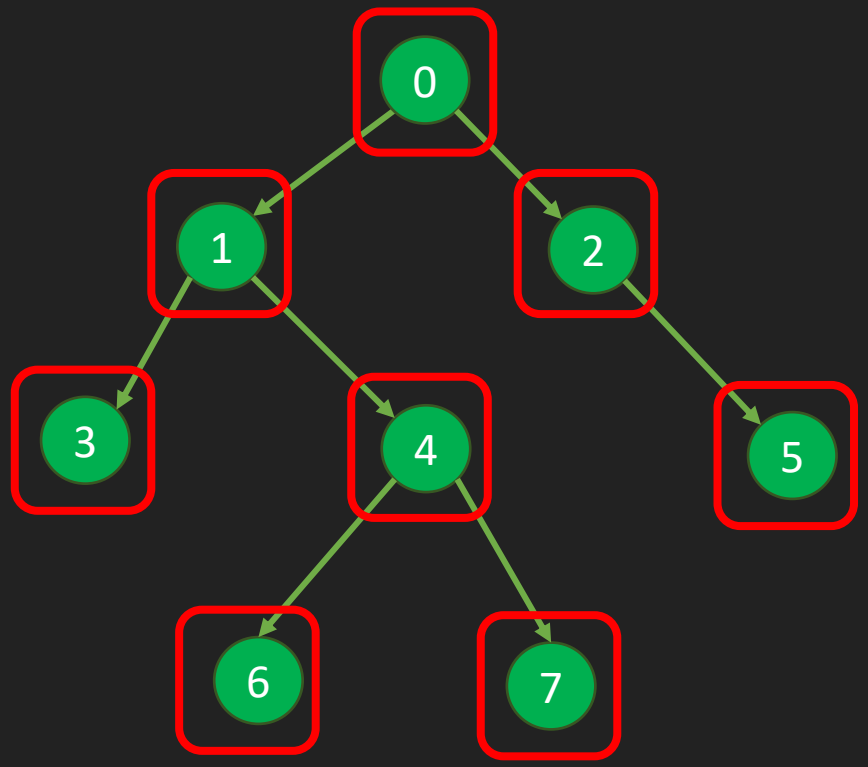
➢ Duyệt cây

There are 3 types of traverse:

- ❖ Pre-order Traversal: [Node -> L -> R]
- ❖ In-order Traversal: [L -> Node -> R]
- ❖ Post-order Traversal: [L -> R -> Node]

5. Traverse a Tree

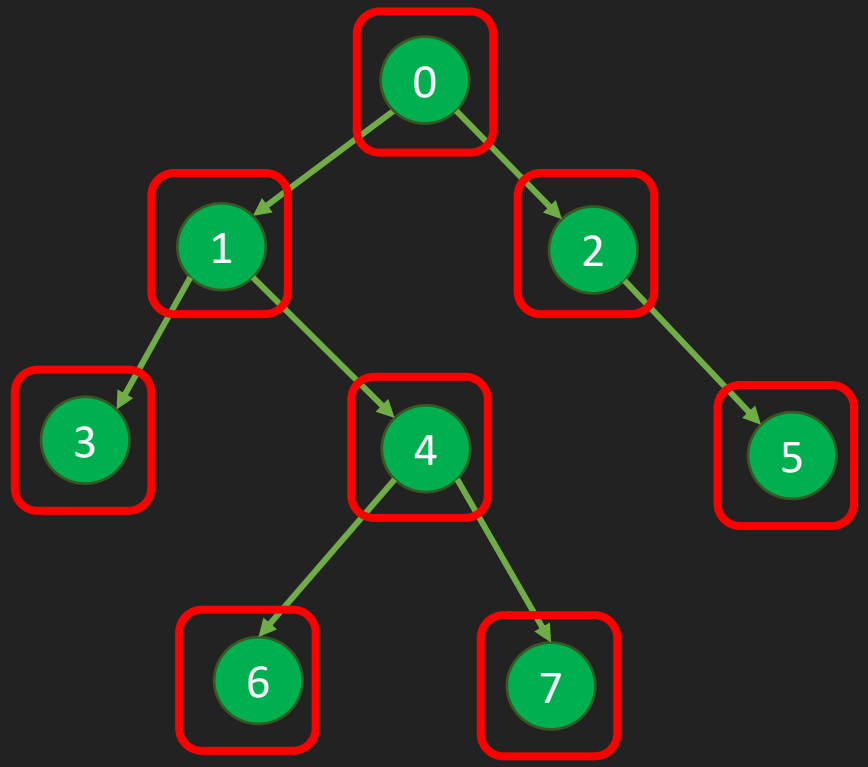
❖ Pre-order Traversal (N-L-R)



0	1	3	4	6	7	2	5

5. Traverse a Tree

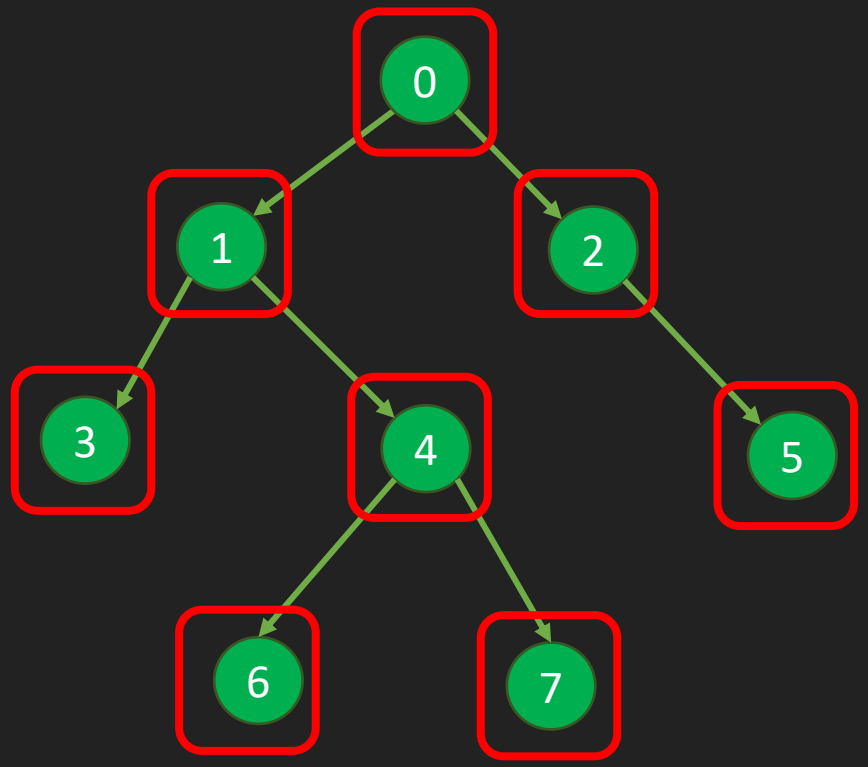
❖ In-order Traversal (L-N-R)



3	1	6	4	7	0	2	5

5. Traverse a Tree

❖ Post-order Traversal (L-R-N)



3	6	7	4	1	5	2	0

5. Traverse a Tree

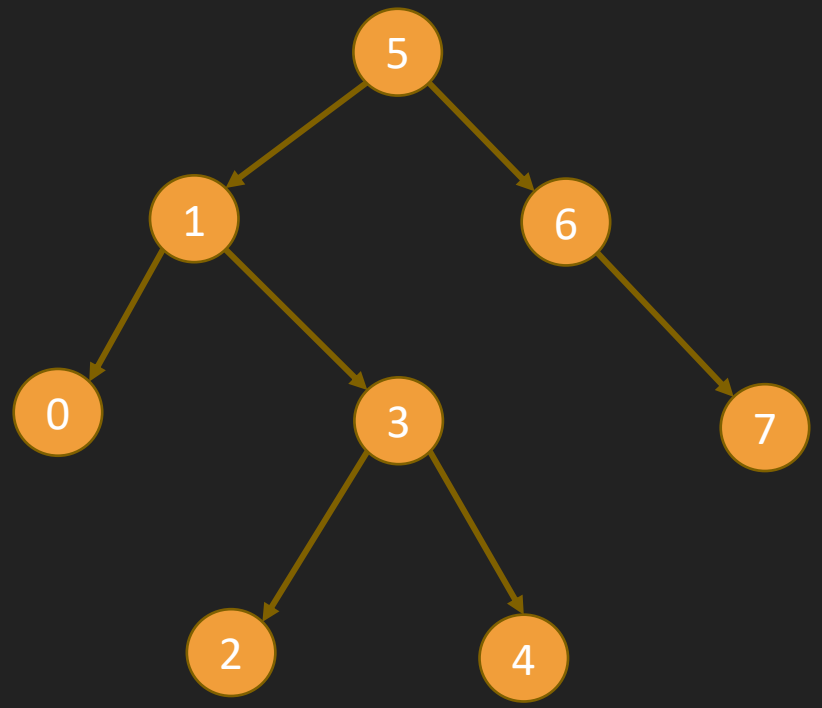
❖ Practice

- ✓ 144. Binary Tree Preorder Traversal
- ✓ 94. Binary Tree Inorder Traversal
- ✓ 145. Binary Tree Postorder Traversal
- ✓ 102. Binary Tree Level Order Traversal * (BFS)

6. Working with Binary Tree and BST

- ✓ 104. Maximum Depth of Binary Tree
- ✓ 112. Path Sum
- ✓ 173. Binary Search Tree Iterator
- ✓ 98. Validate Binary Search Tree

7. Binary Search Tree



- Question: Print in ascending order?
- Practice:

Data Structure & Algorithm

Please Like and Subcribe