
Your Ultimate Guide To Landing
Top AI roles


DECODE
AiML

2.18

Introduction to AVL Tree

DECODE AiML

→ The disadvantage of a Binary search Tree (BST) is that the worst case time complexity of search is $O(n)$

→ If the BST is balanced, we can ensure $O(\log n)$ search time

→ one of the most popular balanced BST is AVL Tree.

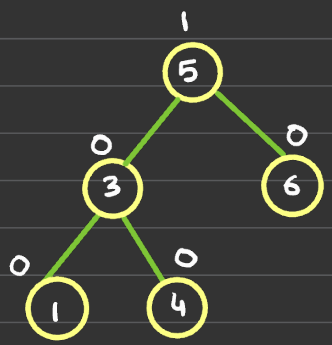
→ Height of AVL tree always remains $O(\log n)$

→ A non-empty binary Tree T is an AVL tree iff given T^L and T^R to be left and right subtree of T and $h(T^L)$ and $h(T^R)$ to be height of subtree T^L and T^R respectively.

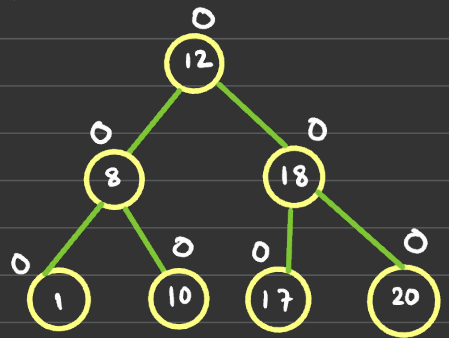
$$|h(T^L) - h(T^R)| \leq 1$$

→ $(h(T^L) - h(T^R))$ is called **balance factor** and is valid at every node.

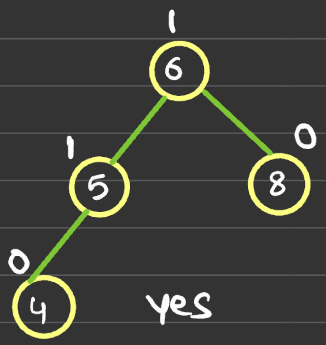
Q. Which of the below trees are balanced BST (AVL) and which are not?



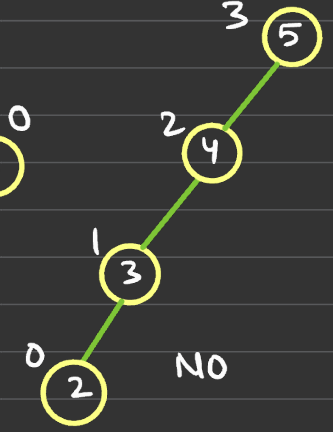
Yes



Yes

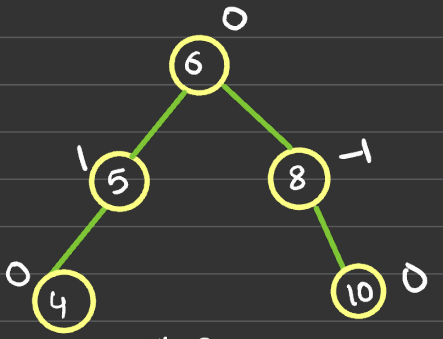


Yes

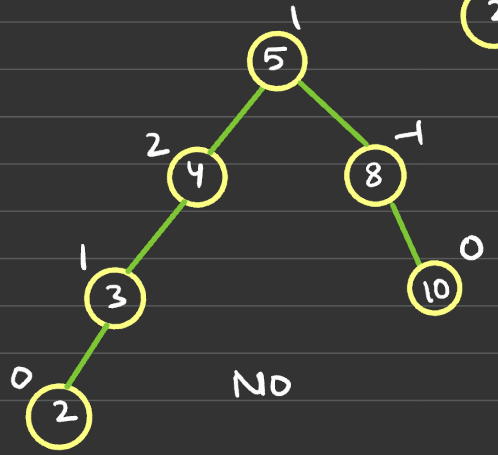


NO

No



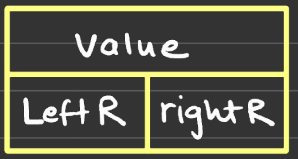
Yes



NO

AVL Implementation in Python

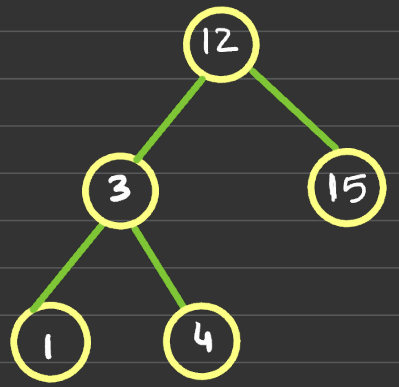
```
Class TreeNode:
    def __init__(self,value):
        self.value = value
        self.left = None
        self.right = None
```



Node structure

Key operations in a BST

1. **Insertion**
   - → Insert a key in a BST style
   - → If unbalanced, perform rotation.
2. **Search**
   - → Search for a key in a BST
3. **Deletion**
   - → Delete a key in a BST style
   - → If unbalanced, perform rotation.



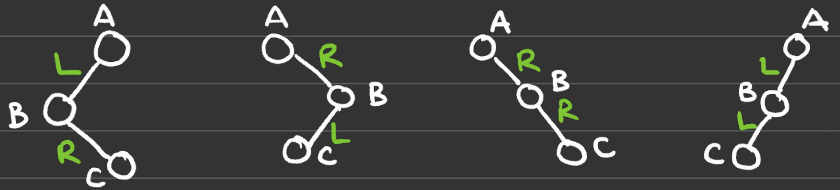
→ If after insertion/deletion, the balance factor of any node in the tree is affected → Try Rotation techniques.

→ Rotation is a tree-restructuring operation used to restore balance when the balance factor of a node goes outside the allowed range {-1, 0, +1}.

Rotation in a AVL Tree

① To perform the rotation, it is necessary to identify a specific node A whose |balance factor| > 1 and which is nearest ancestor to inserted/deleted node on path from inserted/deleted node to root.

② Rotation is performed on node A

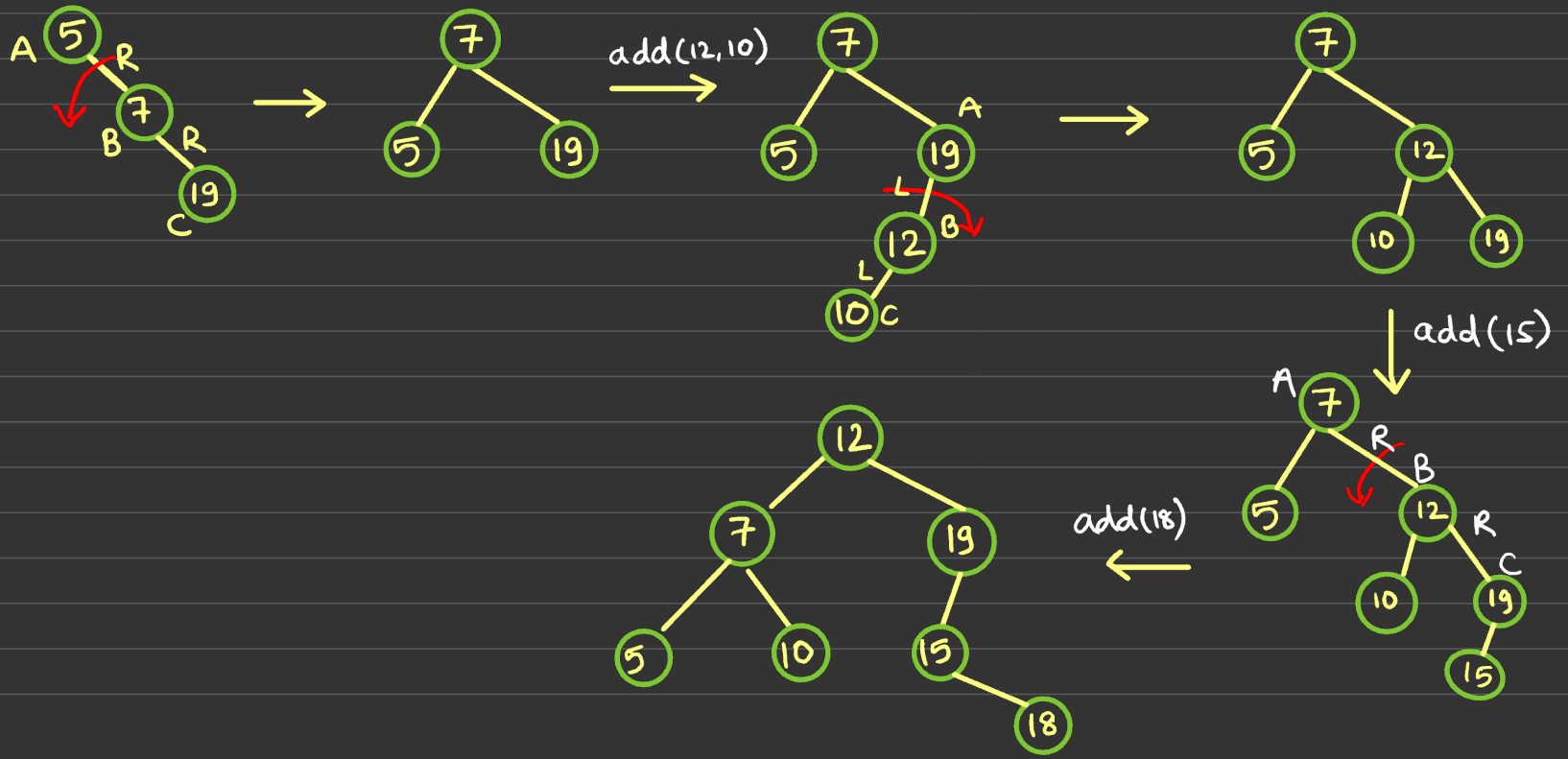


③ For LL and RR rotation → make A child of B
for LR and RL Rotation → make A and B child of C

Insertion in an AVL Tree

Q. Generate AVL Tree for below values.

5, 7, 19, 12, 10, 15, 18



Summary

→ Search in AVL → Same as BST

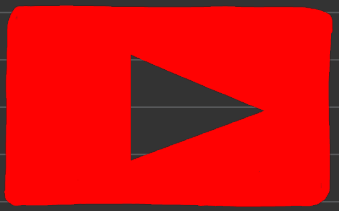
→ Insertion in AVL → BST Insertion + balancing (rotation if needed)

→ Deletion in AVL → BST Deletion + balancing (rotation if needed)

→ For insertion, only one rotation is needed. But for deletion, sometimes multiple rotation may be needed.

→ AVL ensures height remains O(logn), unlike an unbalanced BST.

→ Time Complexity of Insert, search and delete in AVL Tree is O(logn)

Like  Subscribe