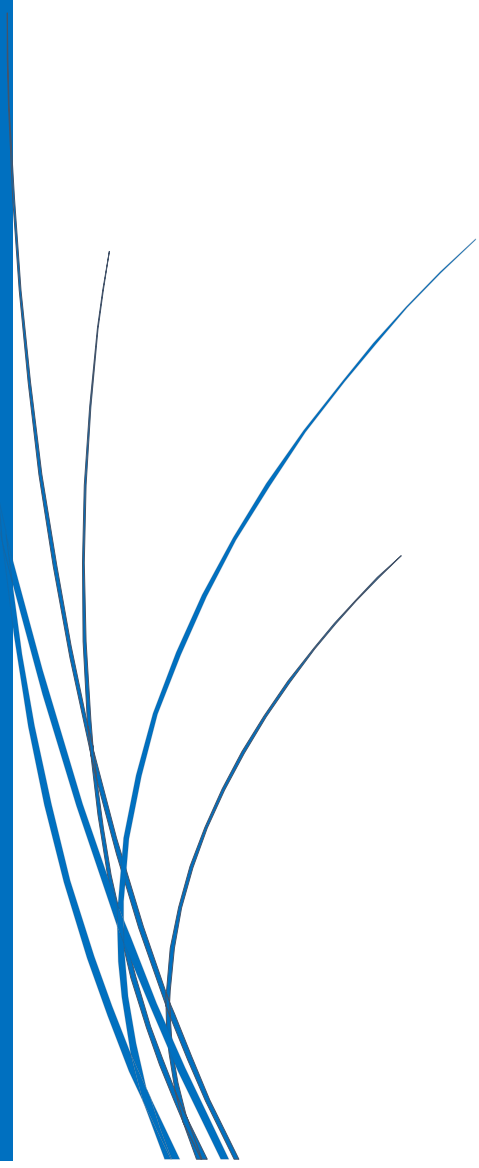
Made By:
Kenan Gazwan

AVL Trees

ICS202-Summary

King Fahd University of
Petroleum and Minerals



Telegram: @KenanGazwan

(AVL Trees)

✓ Balance & Height

Balanced Tree: A binary tree has a difference in height of both subtrees of any node in the tree either 0 or 1.

- A balanced tree of n nodes has a height of log n.
- The imbalanced tree can reach a height close to n.
- The height of trees is important because the runtime of BST operations (Insertion, search) is closely related to height.

Height: Max number of nodes in path from root to any leaf node.
or in other words: the longest path from the root to any leaf node.

✓ AVL Tree

A binary search tree that uses modified insert and delete operations to stay balanced as its elements change.

- Many of the AVL tree operations depend on height, so each node keeps track of its subtree height as a field.
- Rebalancing operations cost: O(1)
- AVL tree's height is always $O(\log n)$
- **The main idea of AVL Tree: It always balances itself.**

Balance Factor (BF): (height of right subtree – height of left subtree) for any node.

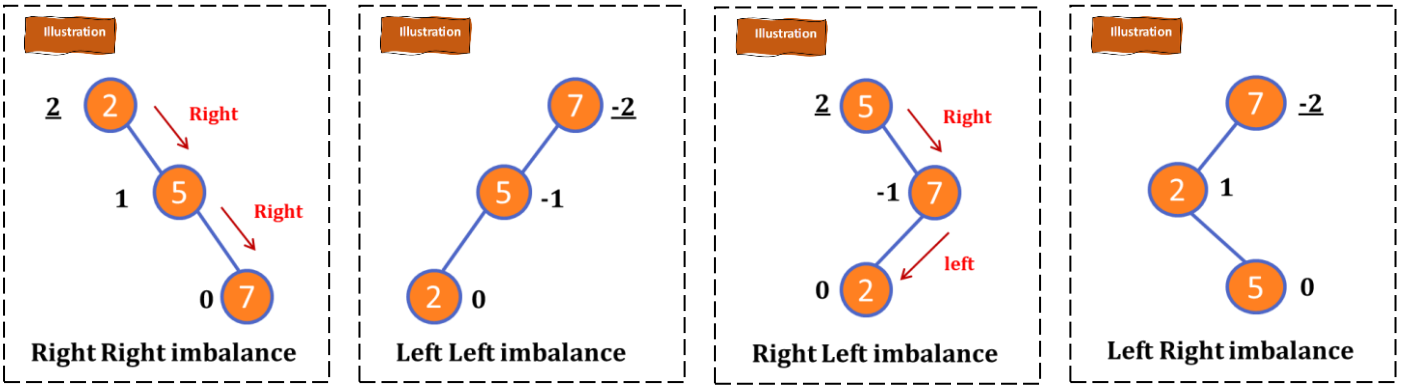
- AVL tree maintains the BF in each node equal to: 0, 1, or -1.

(For all AVL operations, assume the tree was balanced before the operation began.)

AVL Tree (Insertion\Deletion):

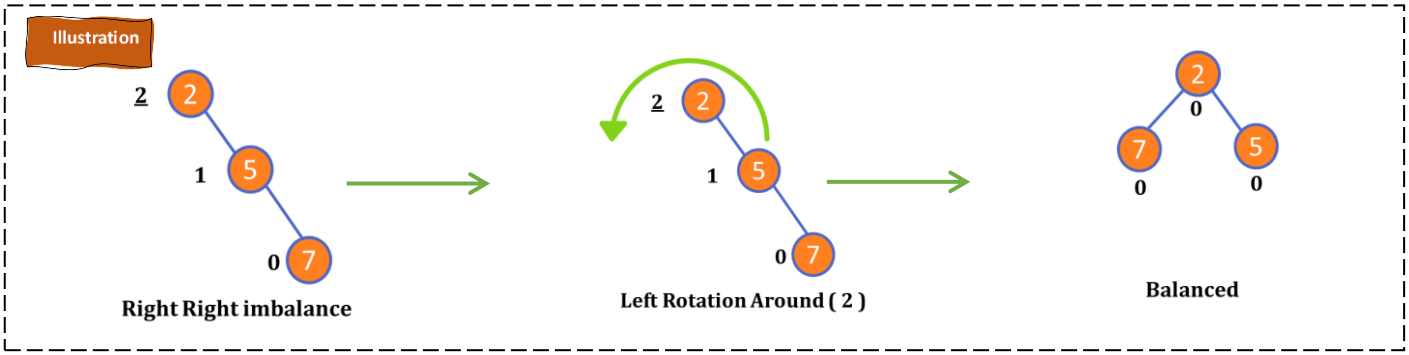
- First, the node is inserted\deleted in the same way as BST.
- Then, go back from the new node to the root and check the BF for each node on the path.
    - Either: all nodes are still balanced or stop on the first node that has a BF of (2)/(-2).

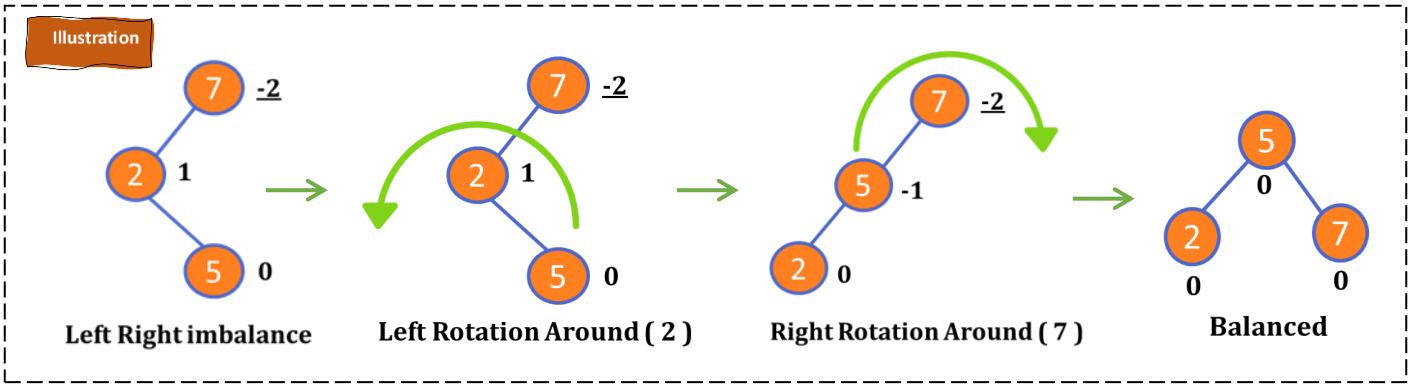
- The imbalanced node has one of these cases:
  - **Right Right – imbalance**
  - **Left Left – imbalance**
  - **Right Left – imbalance**
  - **Left Right – imbalance**



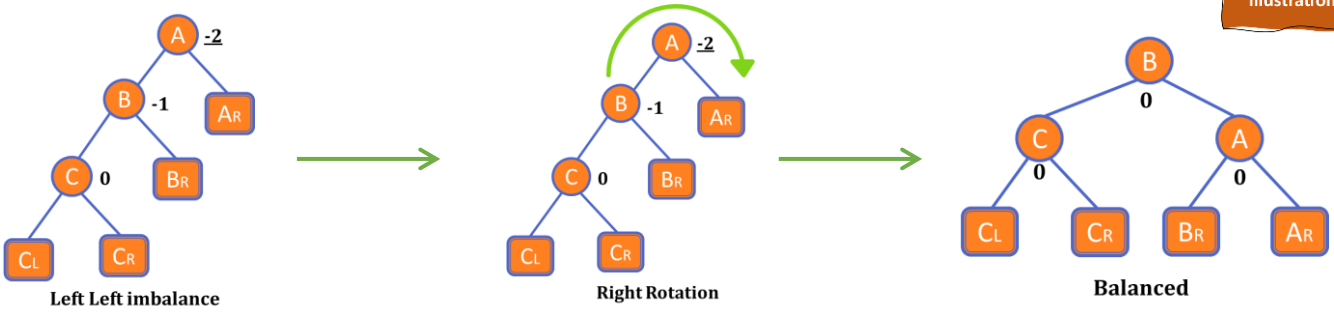
How to Fix Each?

Imbalance Type	Fixing Steps
Right Right	Single **Left** Rotation around the node
Left Left	Single **Right** Rotation around the node
Right Left	○ Single **Right** Rotation around the **Right** Child ○ Single **Left** Rotation Around the Node
Left Right	○ Single **Left** Rotation around the **Left** Child ○ Single **Right** Rotation around the node

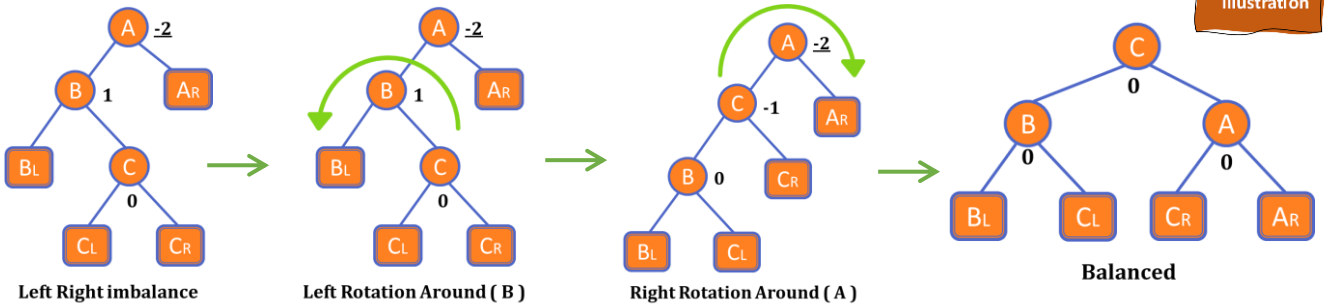




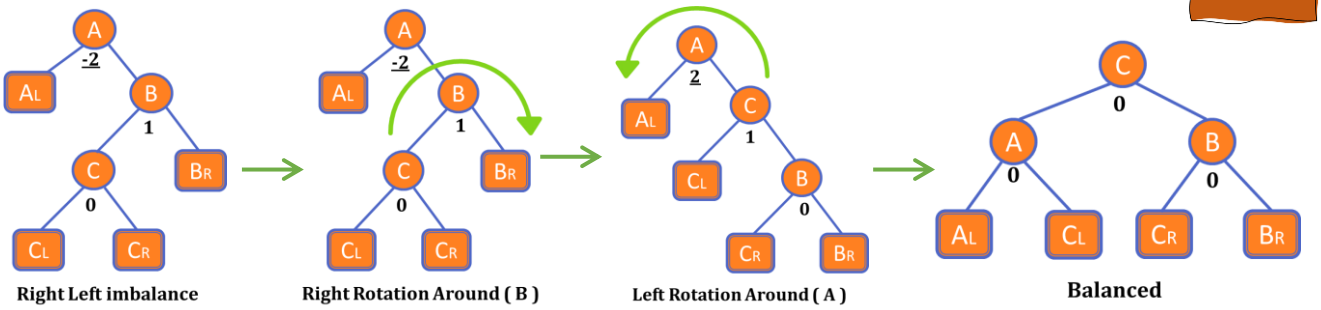
Illustration



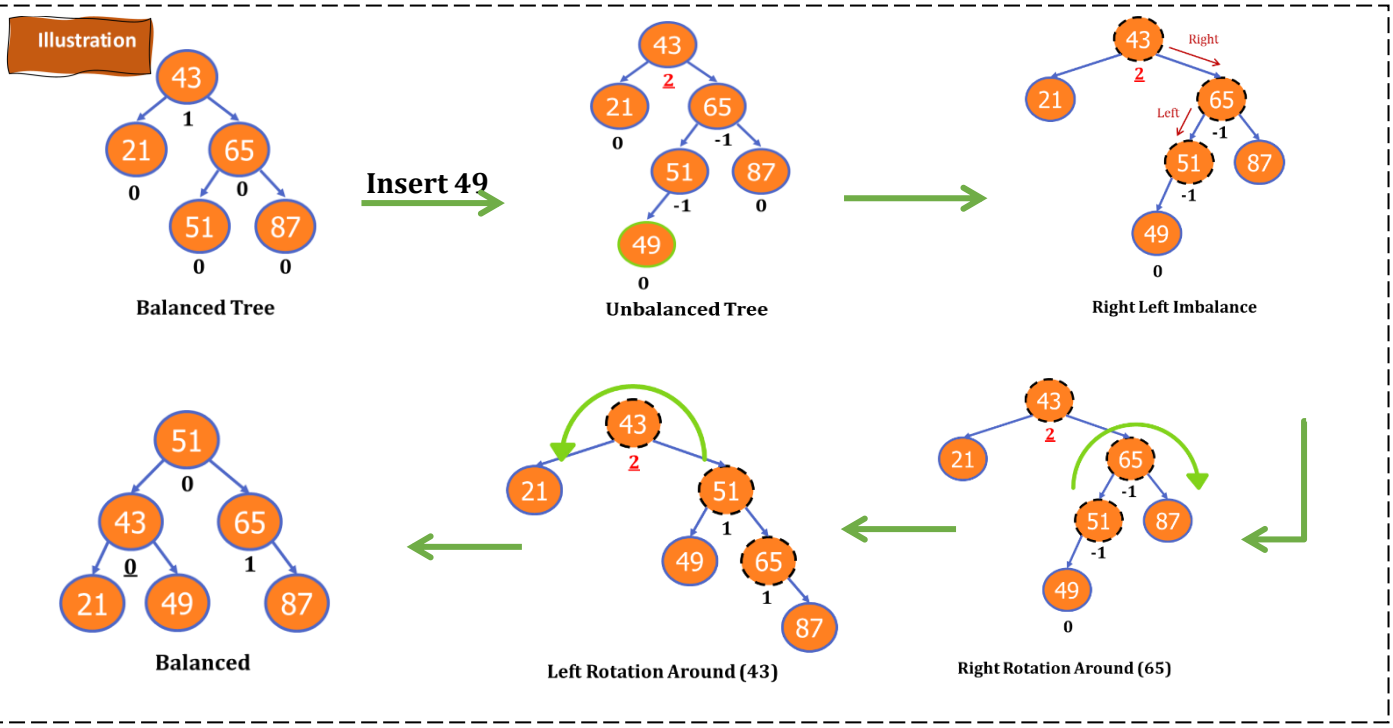
Illustration

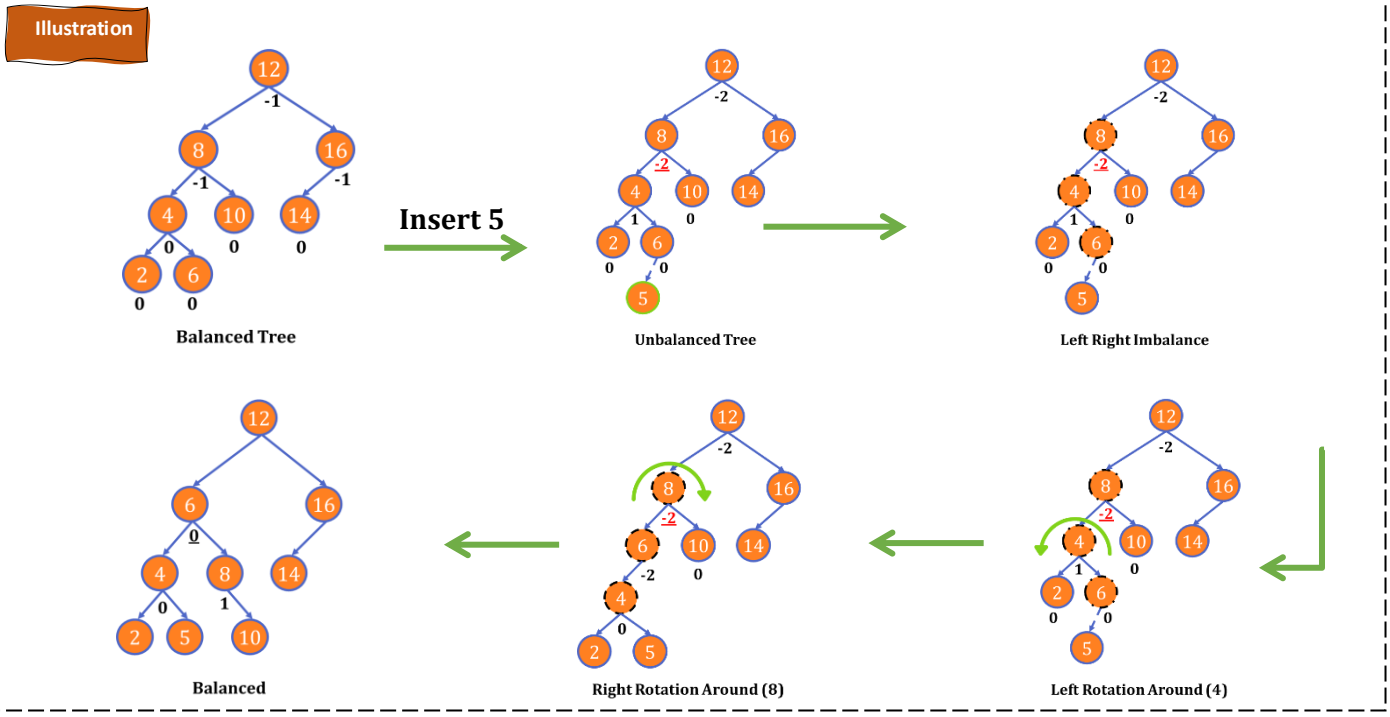


Illustration

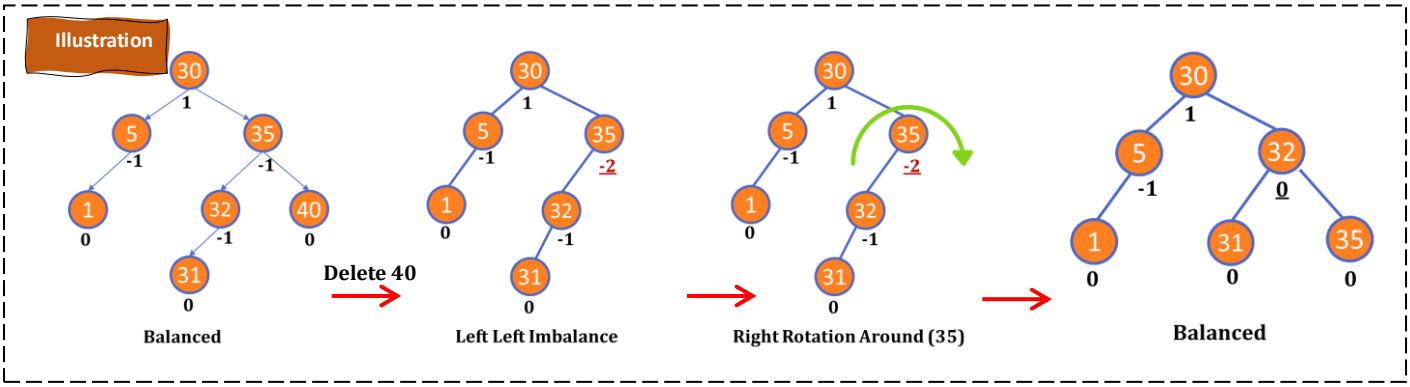


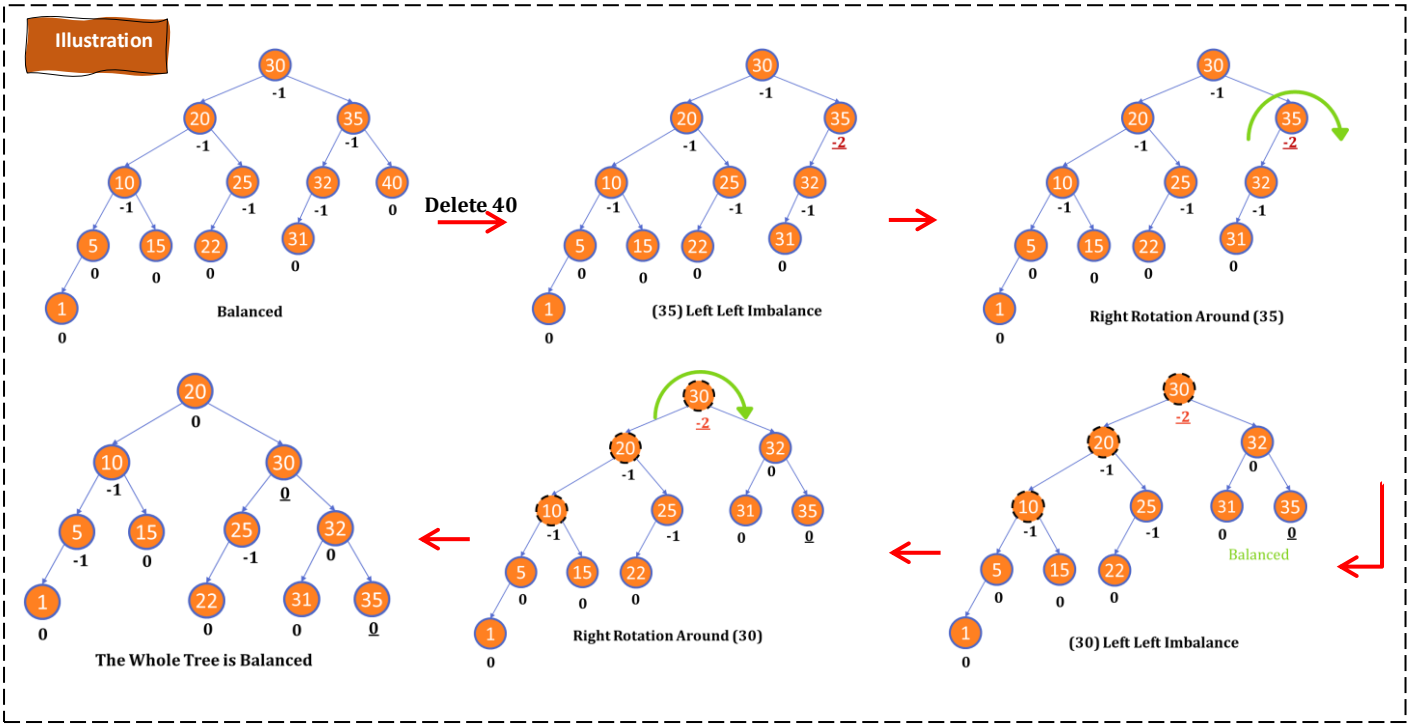
Rebalancing After Insertion:





Rebalancing After Deletion:





Pros and Cons of AVL Trees:

Arguments for AVL trees:

- Search, insertion, and deletion are always O(log n).
- The height balancing adds no more than a constant factor to the speed of insertion.

Arguments against using AVL trees:

- Difficult to program/debug.
- It costs more space for a balance factor.
- Asymptotically faster but rebalancing costs time.