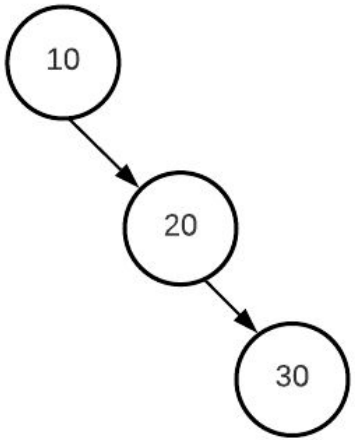
Data Structures

Unbalance Types

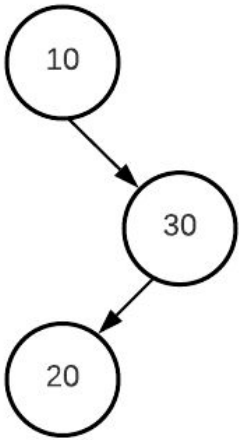
Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

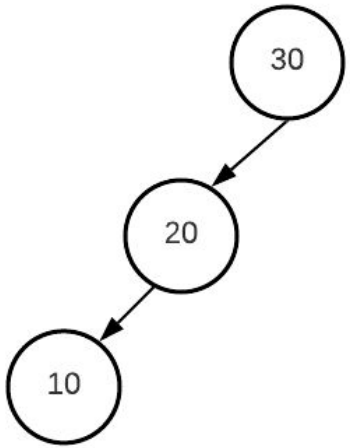
4 unbalanced cases (|BF| > 1)



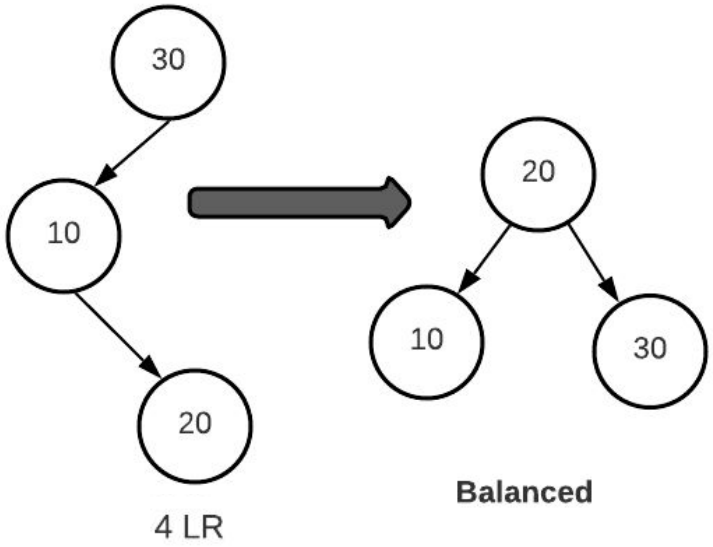
1 RR



2 RL



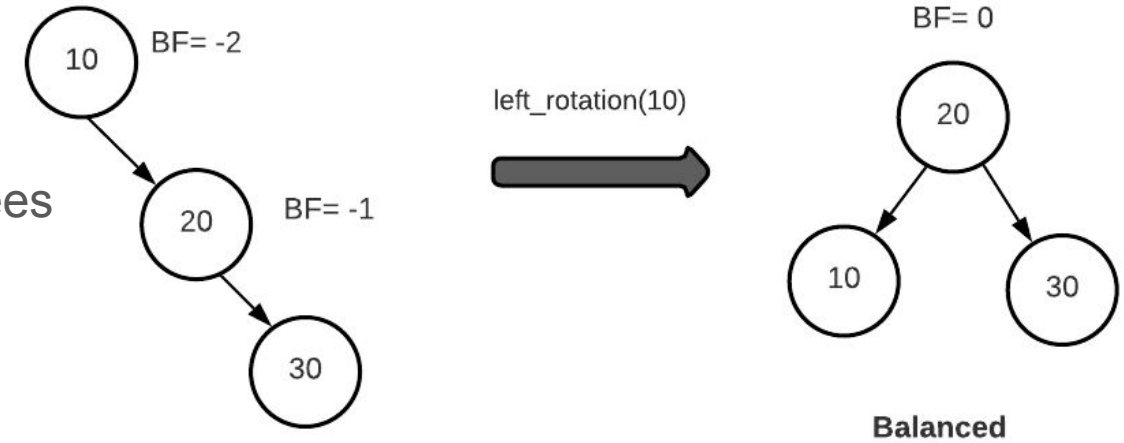
3 LL



4 LR

Case 1: Right-Right unbalanced tree

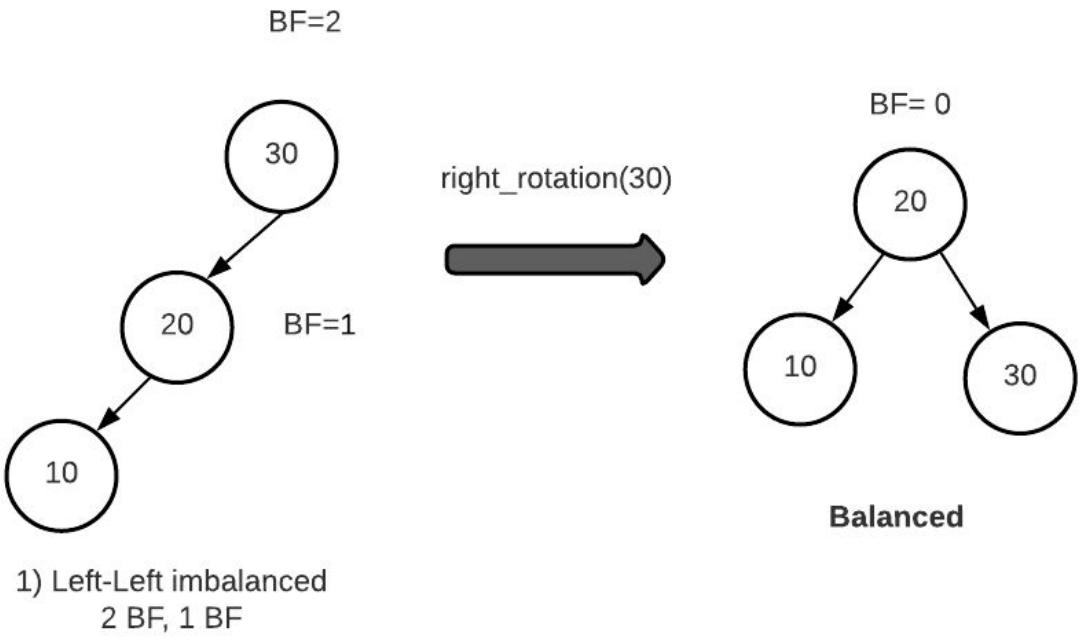
- Occurs when a node is inserted on the right side
  - The BFs are -2 and -1
- Apply left rotation on the root
- node = left_rotation(node)
- For simplicity, we omitted subtrees for nodes 10, 20, and 30



1) Right Right unbalanced
-2 BF, -1 BF

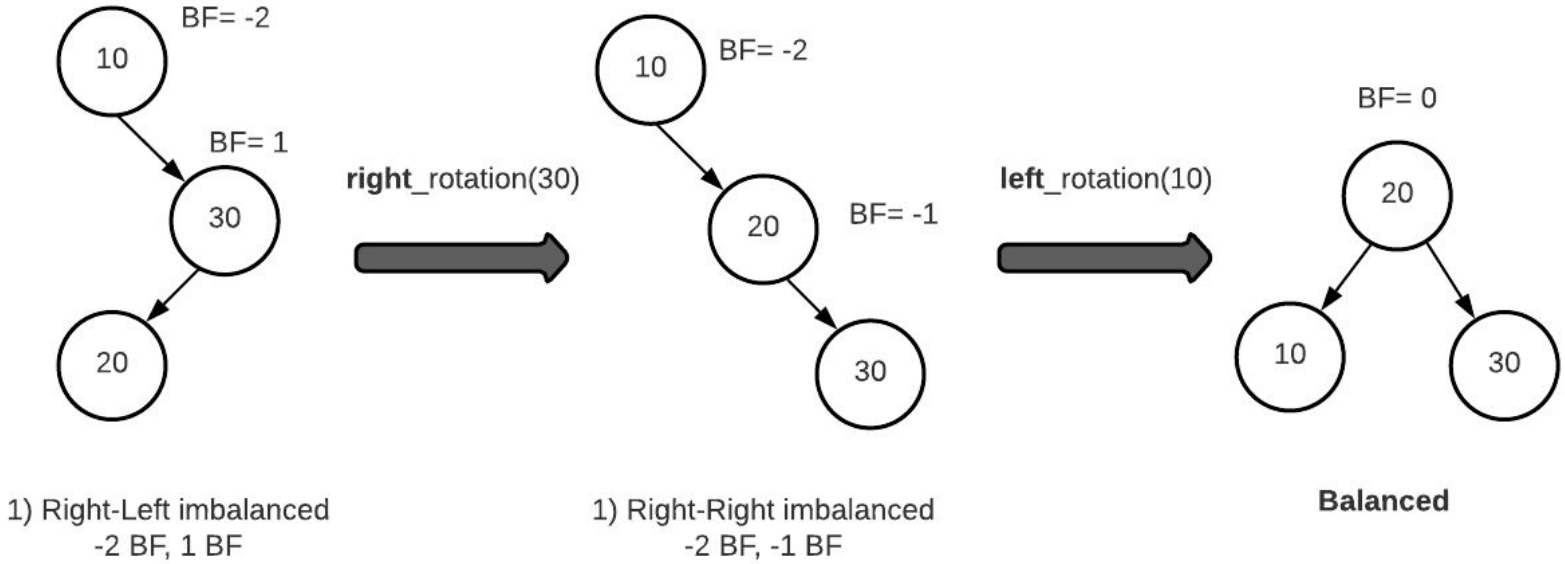
Case 2: Left-Left unbalanced tree

- The complete opposite case
  - BF: 2 1
- Do right rotation on the root
- node = right_rotation(node)



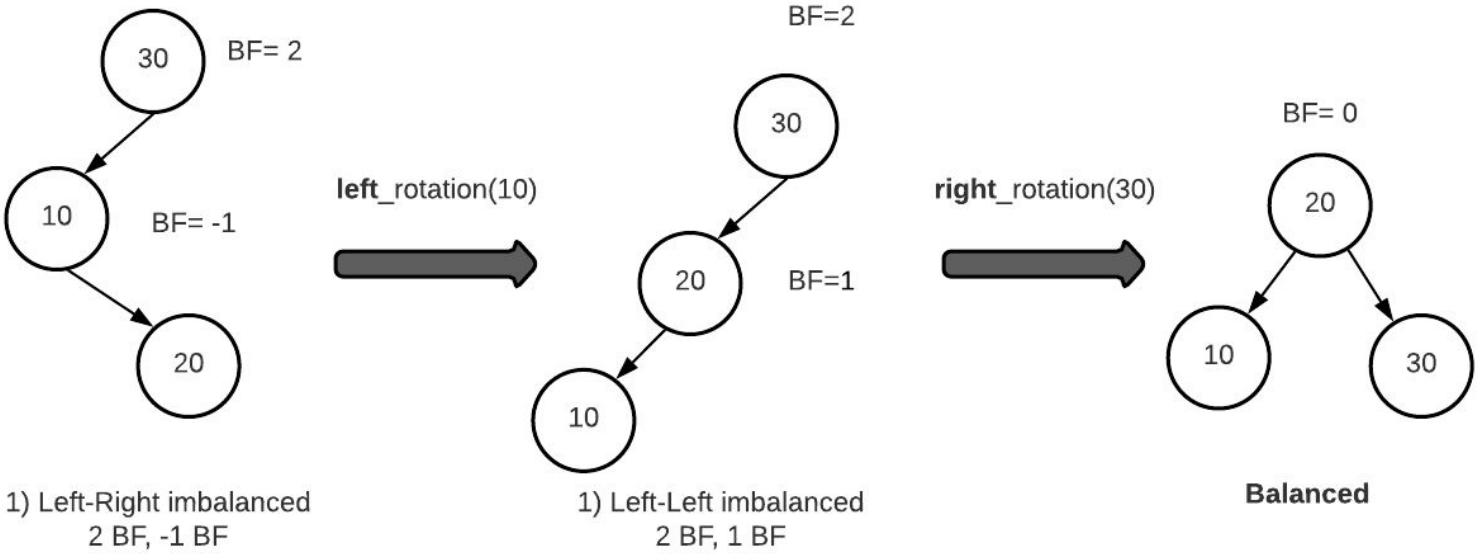
Case 3: Right-Left unbalanced tree

- The trick here is to convert it first to right-right unbalance tree case
- node.right = right_rotation(node.right)
- node = left_rotation(node)



Case 4: Left-Right unbalanced tree

- The trick here is to convert it first to left-left unbalance tree case
- node.left = left_rotation(node.left)
- node = right_rotation(node)



Handling all four cases

```
def balance(self, node):
    if node.balance_factor() == 2:                  # Left
        if node.left.balance_factor() == -1:        # Left Right
            node.left = self._left_rotation(node.left)  # To Left Left

        node = self._right_rotation(node)           # Balance Left Left
    elif node.balance_factor == -2:
        if node.right.balance_factor() == 1:
            node.right = self._right_rotation(node.right)

        node = self._left_rotation(node)

    return node
```

Notes

- All of the functions introduced are O(1)
- In a general BST, there's practically no limit to the balance factor!
- But the balance() function assumes only: {-2, -1, 0, 1, 2}
- **Why?**
- Because AVL follows a **change-then-fix** approach
- The tree will always be balanced {-1, 0, 1} (i.e. |BF|<=1)
- After insertion or deletion, the tree may become unbalanced; with one more step up or down
    - That is: {-2, -1, 0, 1, 2}
- We immediately fix any corruption using a 'bottom-up' style ⇒ {-1, 0, 1}

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."