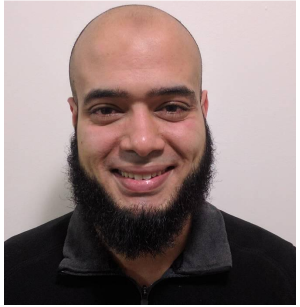
Data Structures

Self-balancing binary search tree

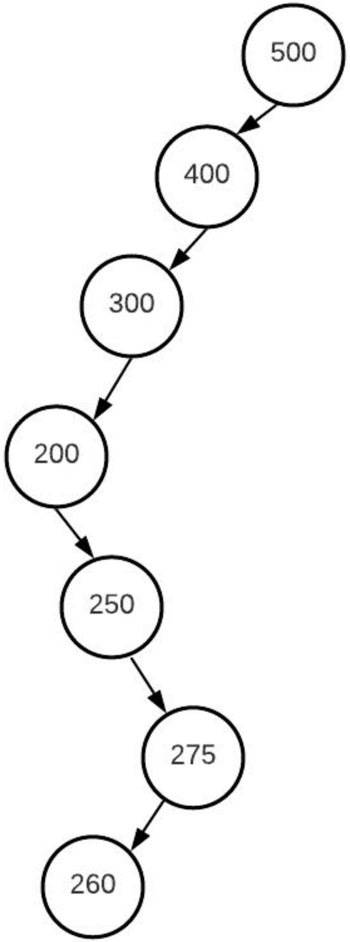
Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

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



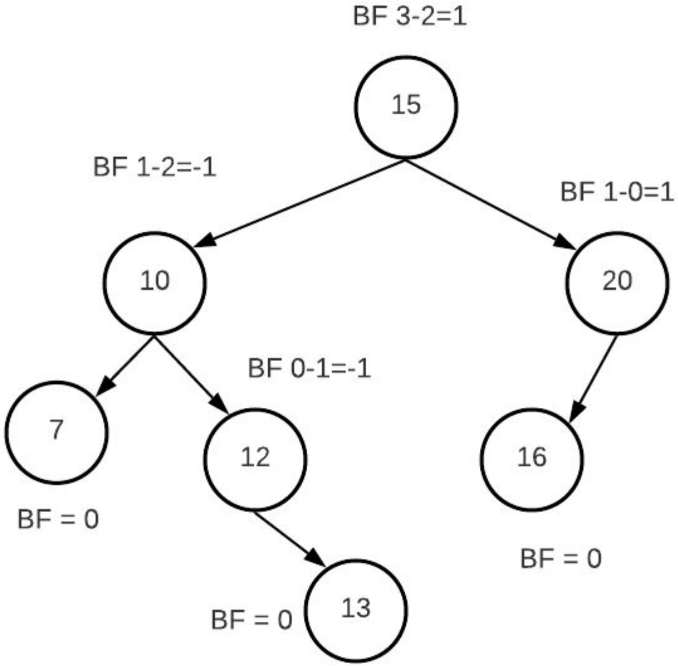
Recall: Degenerate BST

- Each node has 1 child
- From a performance perspective, it is similar to a linked list, with several operations being done in O(n)
- However, in any tree, the bigger the gap between log(n) and the tree height, the less efficient the tree will be!



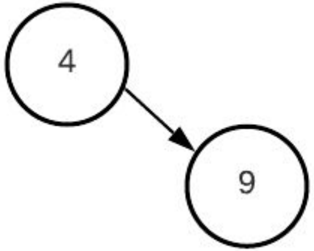
Recall: Balanced BST

- The **difference** in height between the left and right subtrees is never greater than 1.
  - Height(left) - Height(right)
  - **For visualization**, assume height(leaf) = 1
- Let’s compute this difference for each node (**Balance factor**)
  - -7 means = right height greater by 7
  - -1, 0, 1 ⇒ BBST
  - |bf| > 1 ⇒ imbalanced tree



Consider height and balance_factor()

- In the last line of insert() function, we call: update_height()



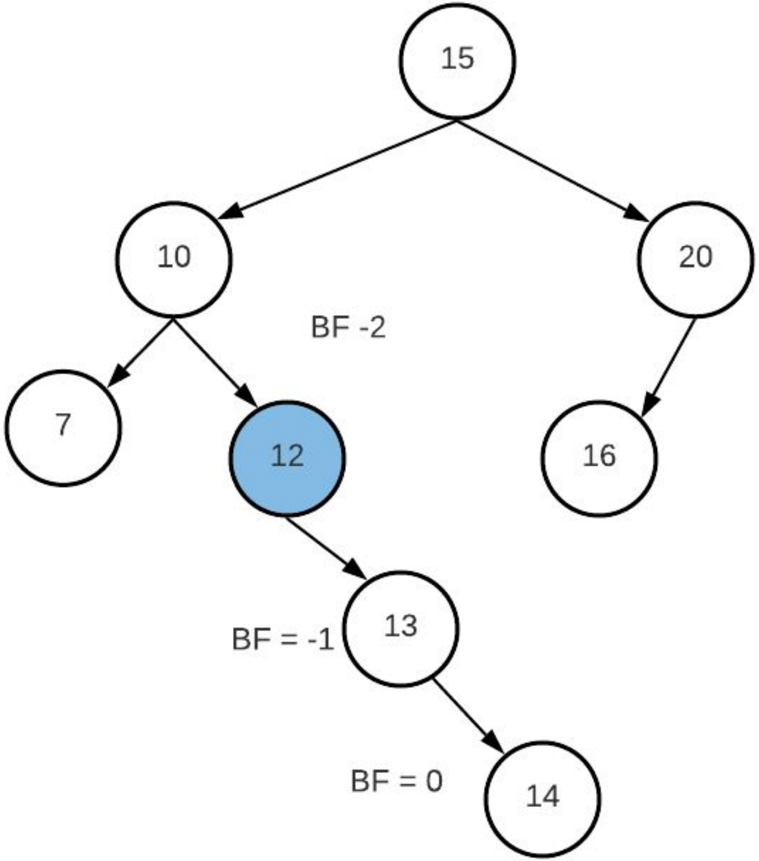

```python
class Node:
    def __init__(self, val=None, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
        self.height = 0  # 0 for leaf

    def ch_height(self, node):  # child height
        if not node:
            return -1  # -1 for null
        return node.height  # 0 for leaf

    def update_height(self):  # call in end of insert function
        self.height = 1 + max(self.ch_height(self.left), self.ch_height(self.right))

    def balance_factor(self):
        return self.ch_height(self.left) - self.ch_height(self.right)

    def is_leaf(self):
        return not self.left and not self.right
```

Maintaining BBST

- Insertion in BST depends on input order - and will generate unbalanced trees.
- To maintain themselves as a BBST, **self-balance** BST trees follow **a change and fix** approach
  - E.g. Insert a new element in BST
  - Do we have an unbalanced BST?
  - Yes ⇒ Fix the tree so that the balance factor is once again between -1 to 1 (i.e. |BF| <=1)
- There are several trees that maintain themselves to remain as a BBST
  - **AVL Trees**: one of the oldest and simplest ways
  - Red-Black Trees, Splay Trees, Treaps
- This section focuses on AVL Trees

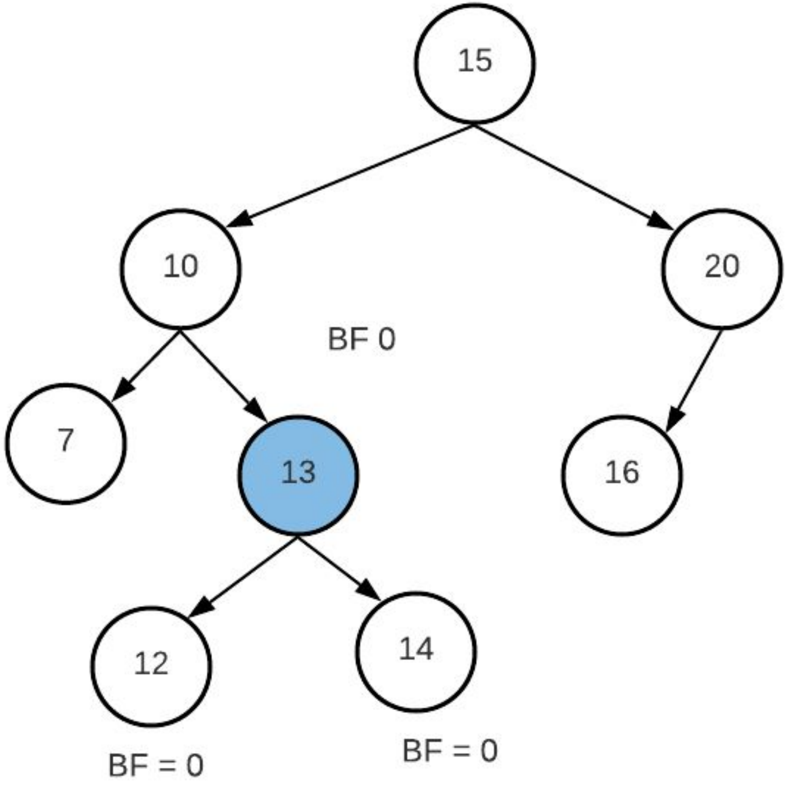
Change (insert 14)

- Let’s insert 14
- Now, recompute the Balance Factor
- We reach leaf nodes recursively, then go back to the parents, and so on
- Once you detect |BF| > 1, , then we know this particular subtree is no longer BBST
- Think for 2 minutes about how to restructure subtree (12) to make it a balanced BST



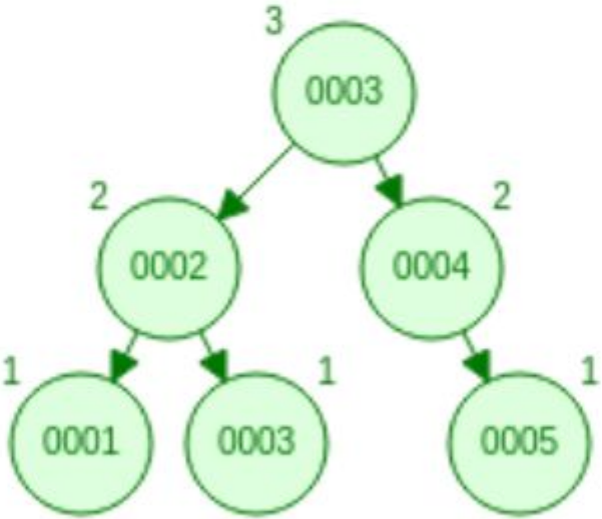
Fix (Tree Rotation)

- If we pushed node 13 **up** and node 12 **down** left, this subtree is fixed
  - Observe, we did not change other subtrees
  - Observe, the tree remains BBST
- This kind of systematic change is called **tree rotation**
- If, after insertion, we keep fixing corrupted sub-trees from the bottom up, we can maintain a BBST



AVL Demo

- Many websites help you to visualize insertion/deletion in AVL trees with real-time online demos
- Try inserting a batch of unsorted numbers, and watch the tree **restructure** itself to accommodate them



“Acquire knowledge and impart it to the people.”

“Seek knowledge from the Cradle to the Grave.”