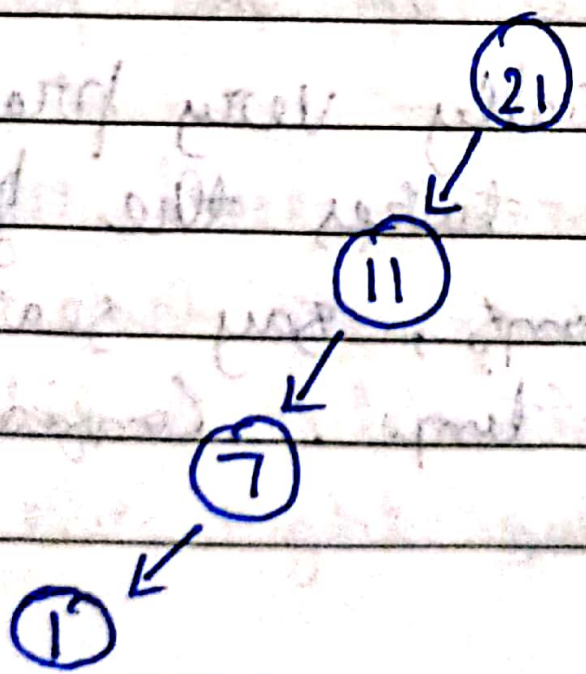
AVL Trees:

→ Suppose we are told to create a Binary Search tree out of some elements given. Suppose the numbers were {1, 11, 7, 21}. And to avoid any labour, we simply sort the elements and write them as shown:



This is indeed a Binary search tree, though skewed. There is no violation of any of the rules of a binary search tree. But any operation here has a complexity of O(n). But we expected something more balanced, something more dense. Well, that is what an AVL tree is.

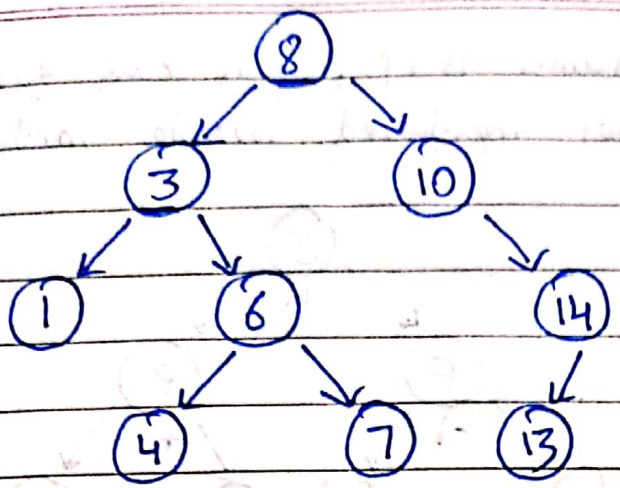
→ An AVL tree is needed because:

1) Almost all the operations in a binary search tree are of order O(h) where h is the height of the tree.

2) If we don't plan our trees properly, this height can get as high as n where n is the number of nodes in the BST (skewed tree).

3) So, to guarantee an upper bound of O(log n) for all these operations we use balanced trees.

→ This is actually very practical. Because when a BST takes the form of a list, our operations, say searching, starts taking more time. Consider the search tree below.



To search 1, in this BST, you would need only 3 operations, while to search in a list type binary search tree having the same elements, this would have taken 9 operations.

→ What are AVL trees?

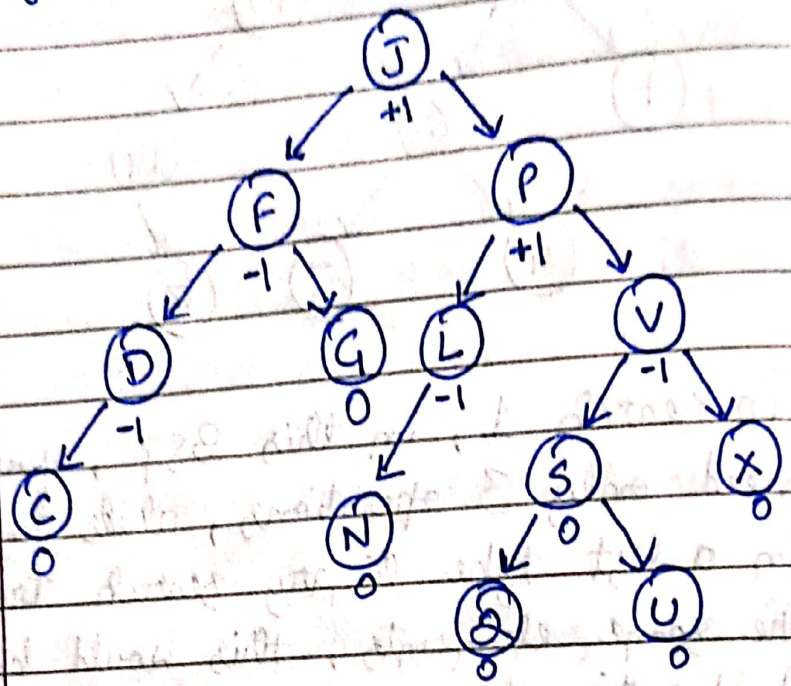
1) AVL trees are height balanced search trees. Because most of the operations work on O(h), we would want the value of h to be minimum possible, which is log(n).

2) Height difference between the left and the right subtrees is less than 1 or equal in an AVL trees.

3) For AVL trees, there is a balance factor BF, which is equal to the height of the left subtree subtracted from the height of the right subtree. If we consider

Date

the below BST, you can see the balance factor mentioned beside each node.

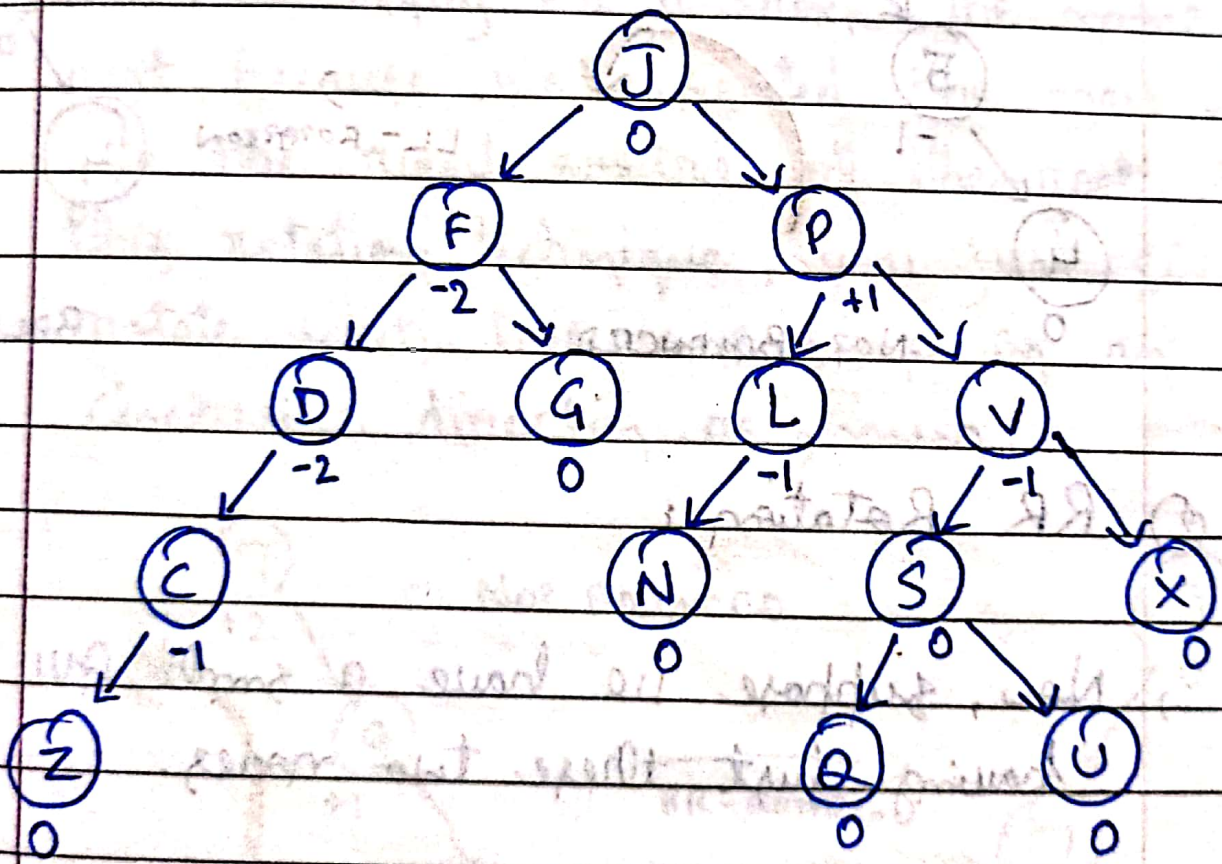


You can see, none of the nodes above have a balance factor more than 1 or less than -1. So, for a balance tree to be considered an AVL tree, the value of |BF| should be less than or equal to 1 for each of the nodes, i.e., $|BF| \leq 1$.

4) And even if some of the nodes in a BST have a |BF| less than or equal to 1, those nodes are considered balanced. And if all the nodes are balanced, it becomes an AVL.

→ An AVL tree gets disturbed sometime when we try to inserting a new

element in it. For example, in the above AVL tree, if we try inserting an element Z at the end of the leftmost element, the balanced factor gets updated for each of the nodes following above. And the tree is no more an AVL tree.



And to avoid this unbalancing, we have an operation called rotation in AVL trees. This helps maintain the balancing of nodes even after a new element gets inserted.