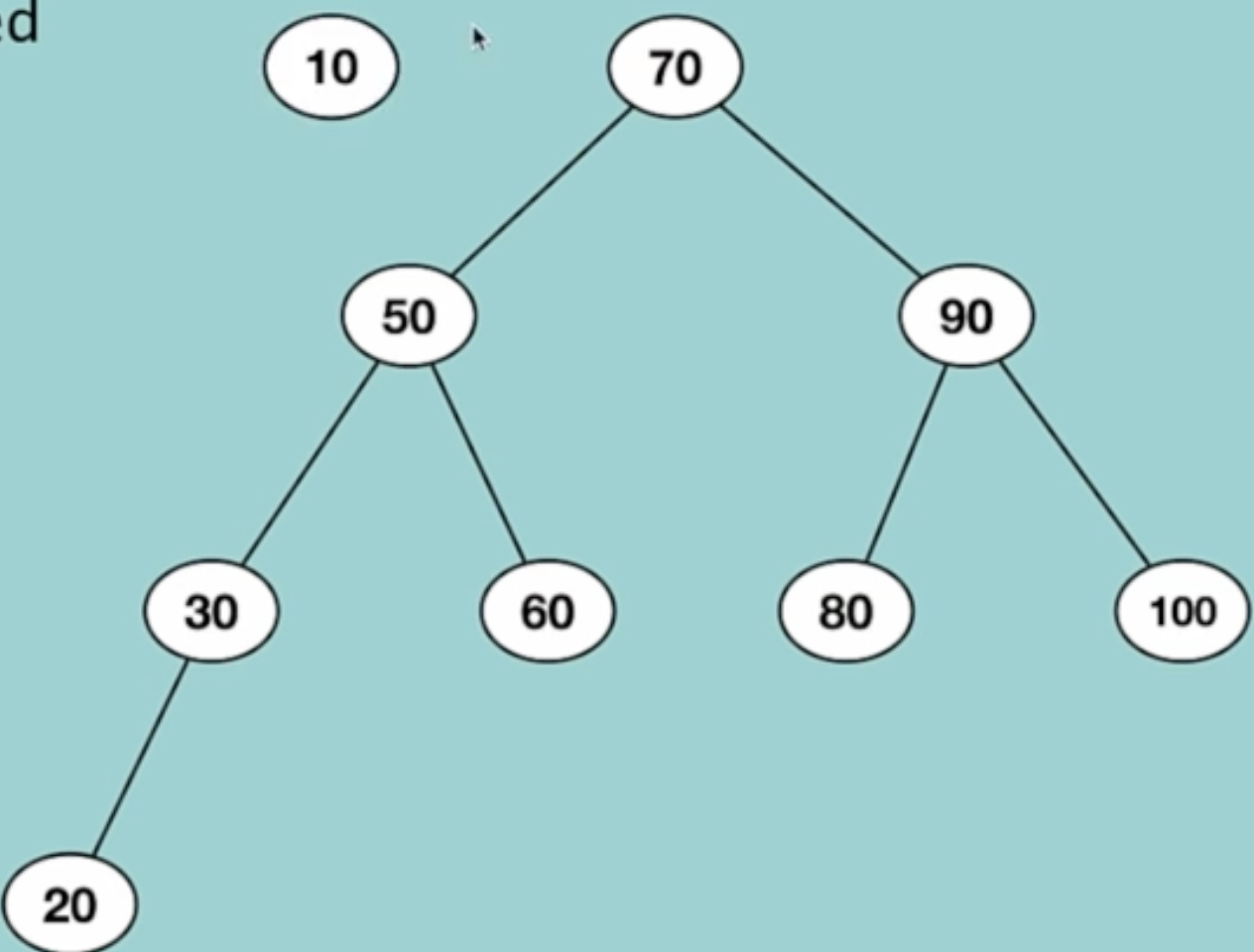
Insertion in AVL trees:

To insert a node in AVL tree (which rotation is requires after insertion) we have to divide the case on 4 conditions:

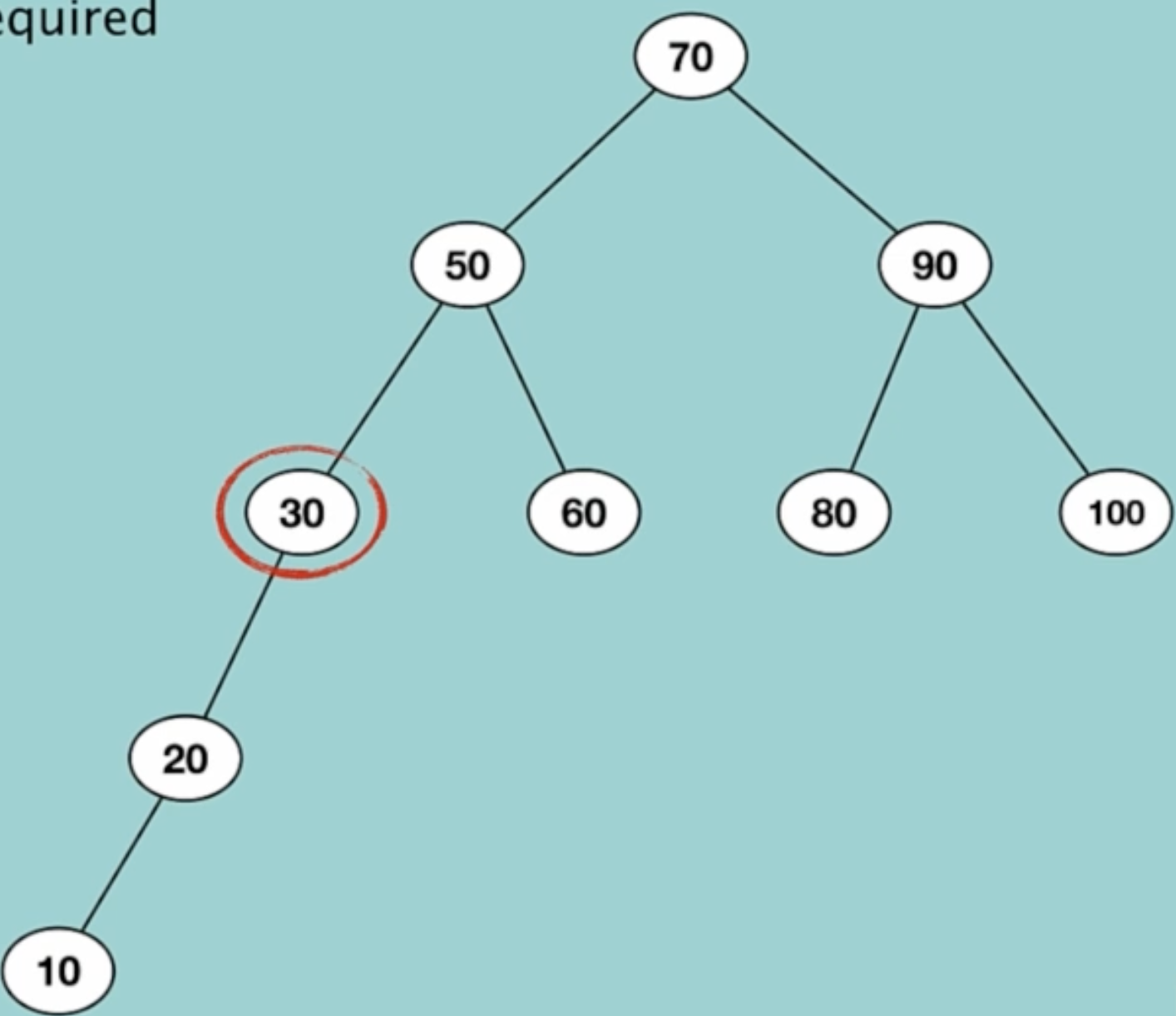
Right Right case - Left Left case – Right left case – Left Right case

# LL condition:

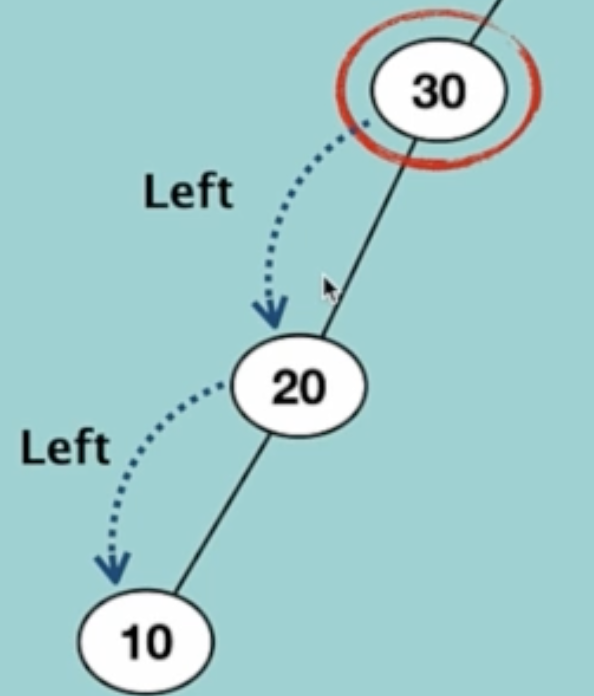
Let’s say we want to insert the node which has a value of 10 to this BST:



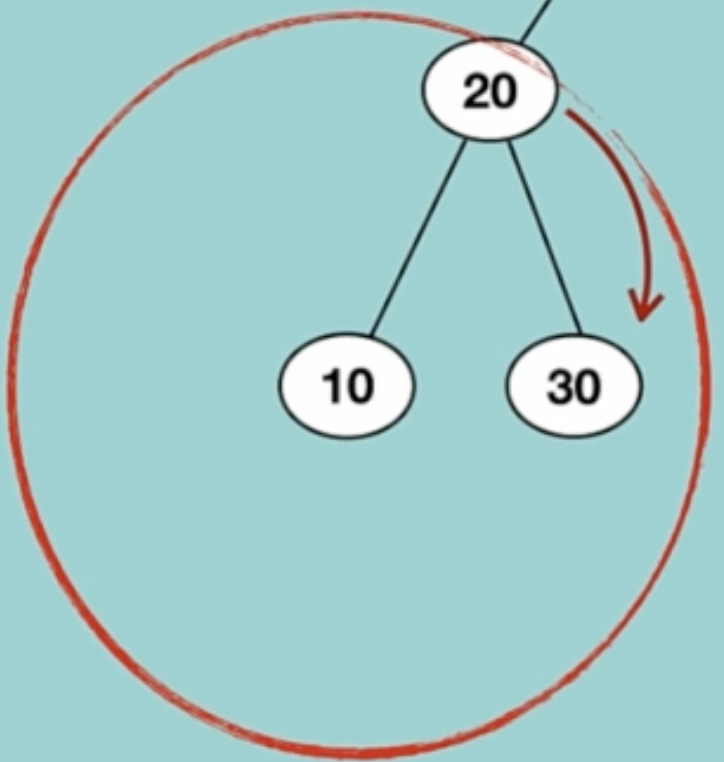
After the insertion the BST will look like this:



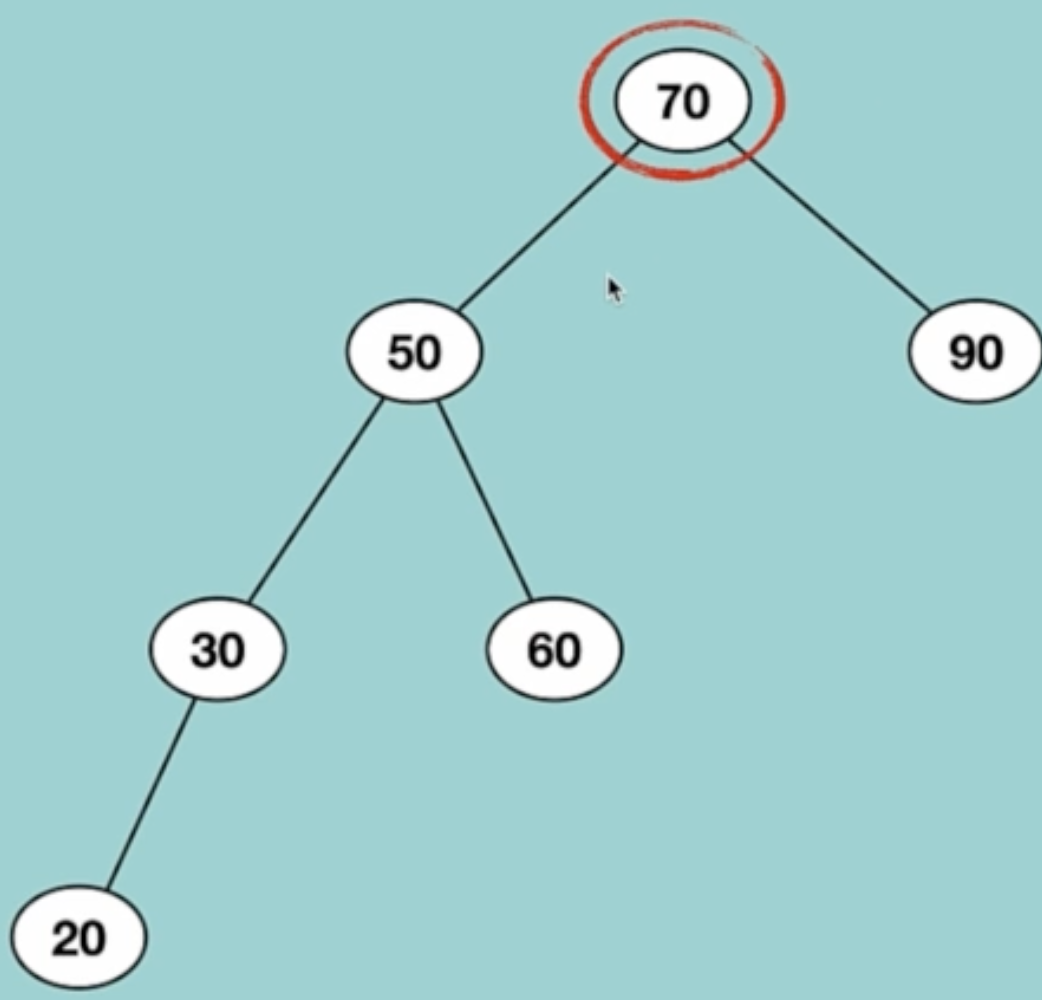
And if we check the balancing condition of the parents node of 10 when we encounter node that has value of 30 we see the height of left sub-tree is 2 and the height of right sub-tree is 0 the difference is: 2 and it’s bigger than one so know we need rotation to balance the tree.



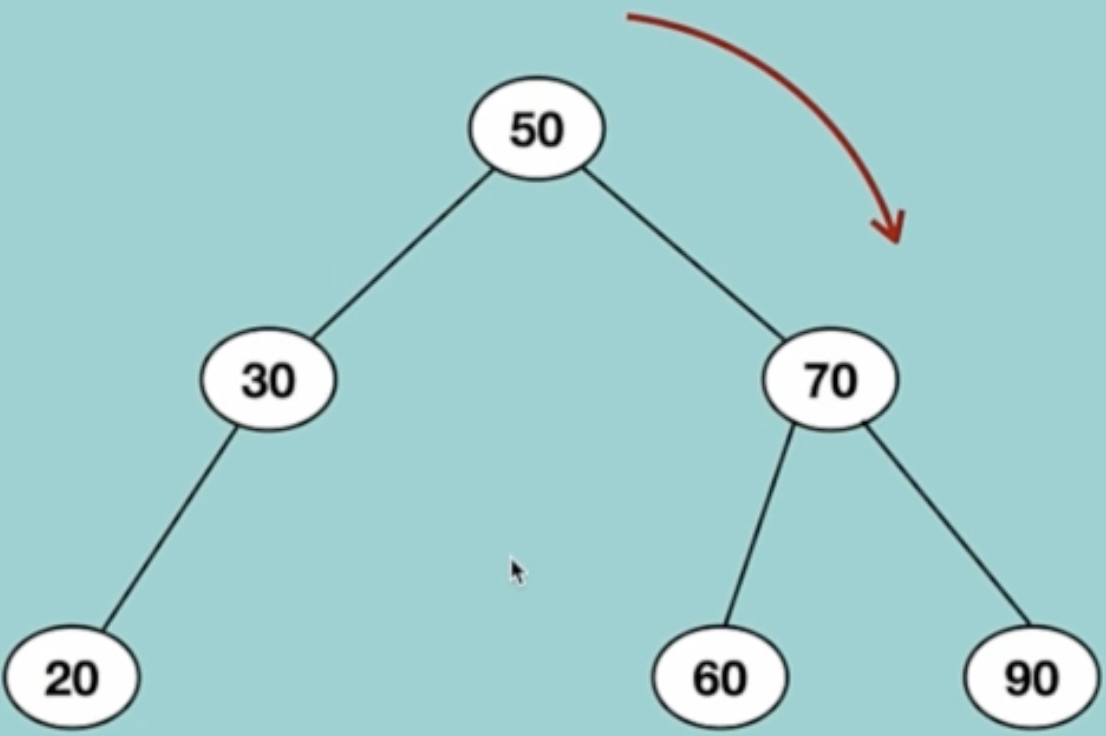
The solution is when we find the Node that cause this problem we move the causing problem Node to the right child and the left-Child goes up :



Just an other example :



We search for a grand child of 70 and we see it has two one, we chose the one that has more height which in this case is 30



Algorithm of Left Left (LL) Condition:

rotateRight(disbalancedNode):

newRoot = disbalancedNode.leftChild

disbalancedNode.leftChild = disbalancedNode.leftChild.rightChild

newRoot.rightChild = disbalancedNode

update height of disbalancedNode and newRoot

return newRoot

# Left Right condition: