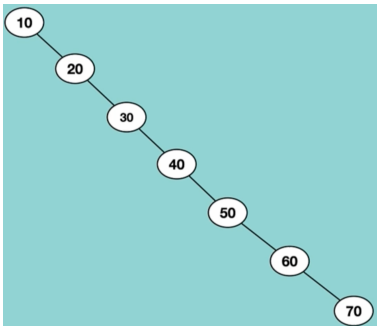
Why do we need AVL Tree?

Look at this example:
We have these numbers and we want to insert them in BST (Binary search Tree)

10, 20, 30, 40, 50, 60, 70

Using the algorithm that indicates if a Node value is greater than root node it should go in the Right sub-Tree then after inserting all these elements our Tree now looks like this:

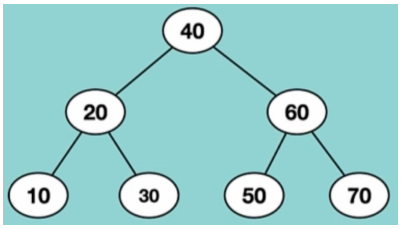


For example, if we want to search for a Node that has a value of 60, we should start from the root Node then right sub-tree then right sub-tree and again and again until we finally find the Node that has the value of 60 in it.

So, this is not very efficient because it has O(n) time complexity the search algorithm that we know has O (log n) time complexity.

In order for us to avoid these time-consuming obstacles, we should use AVL trees.

A balanced BST look like this (time complexity is O (log n):



AVL trees uses some rules to always make sure that by inserting nodes to the BST our tree is balanced.

The property of Being balanced in the trees is so important that can enhance the performance of doing different operations like inserting a node, deleting a node and ….