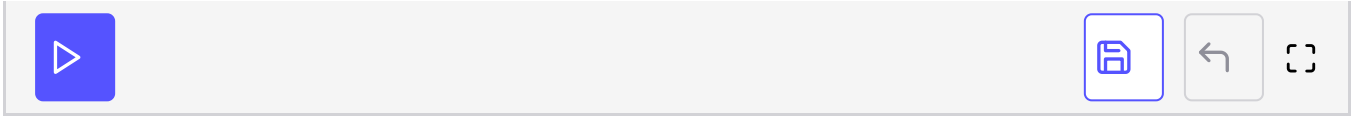
Insertion in Binary Search Tree

(Reading time: 4 minutes)

The Javascript function to add a node in the binary search tree is as follows:

```
addNode(data) {
  const newNode = new Node(data);
  if (!this.root) {
    this.root = newNode;
  } else {
    this.insertNode(this.root, newNode);
  }
}

insertNode(node, newNode) {
  if (newNode.data < node.data) {
    if (!node.left) {
      node.left = newNode;
    } else {
      this.insertNode(node.left, newNode);
    }
  } else {
    if (!node.right) {
       node.right = newNode;
    } else {
       this.insertNode(node.right, newNode);
    }
  }
}
```



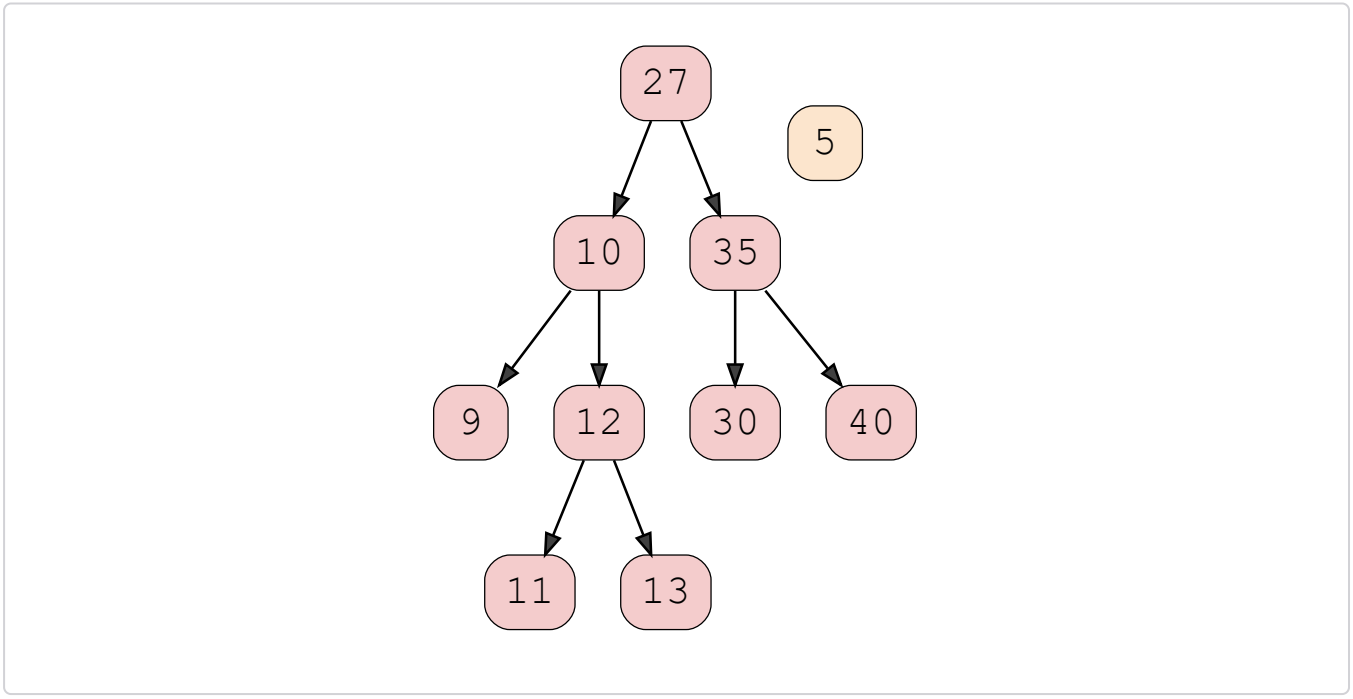
BST after inserting '5'

First, we create our new node. If there is no root in the tree, meaning that there are no nodes at all, the new node is the new root. Otherwise, if the root is already present, we invoke the **insertNode** function that receives two arguments: the root, as we need to check all values in the tree later on, and the new node.

In the **insertNode** function, we check if the currently checked node's data is lower or higher than the new node's data. If it's lower, meaning we go left, we

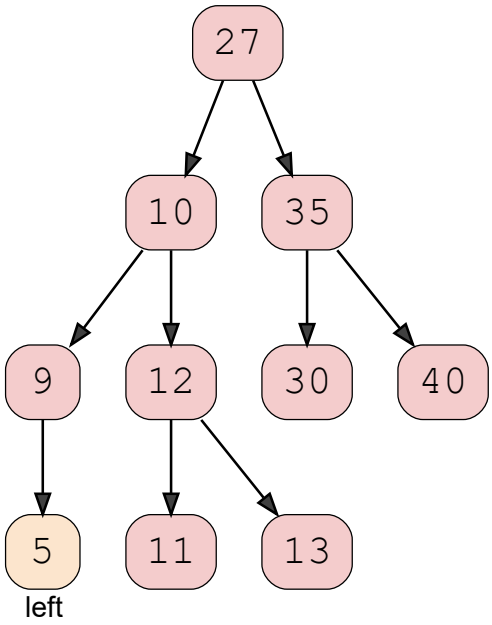
first check if there is a value already. If there isn't, the currently checked

node's left value is equal to the new node. Otherwise, we invoke the **insertNode** function again, to keep on walking through the tree until we find the right value. The node's left node is now passed as the first argument, and we check again whether it's value is lower or higher than the new node's value. This keeps on going, until we find the place for the new node. The same goes for the right values, but the other way around! Let's say we want to add a new node with data 5.



There is a root, so the **insertNode** function gets called as `!this.root` returns false. Within this function, we get to the first if-statement: is the data of the new node is smaller than the currently checked node? In this case, 5 is smaller than 27, so it returns true. Then, we get to the second if-statement. Is there no left node? Yes, there is, there is a node on the node's left with the value of 10, so it returns false.

This means that the **insertNode** function gets called again, only this time with the node which data is **10** as the first argument. We again get to the first if-statement, which still returns true. The node already has a left node, so the insertNode function gets called yet again, only this time with the currently checked node's left node as the first argument: the node with the value of **9**. Again, the first if-statement returns true (5 is smaller than 9), but this time, `!node.left` also returns true! The currently checked node's left value is now equal to the new node. This results in:



In the next lesson, I will talk about deleting a node from a binary search tree.