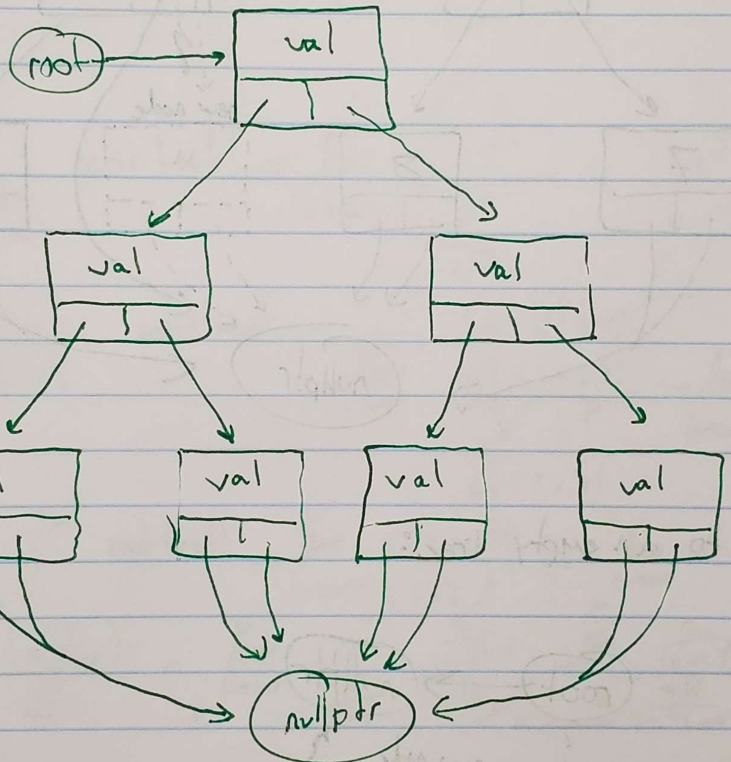
Binary Search Tree

A Node should contain:

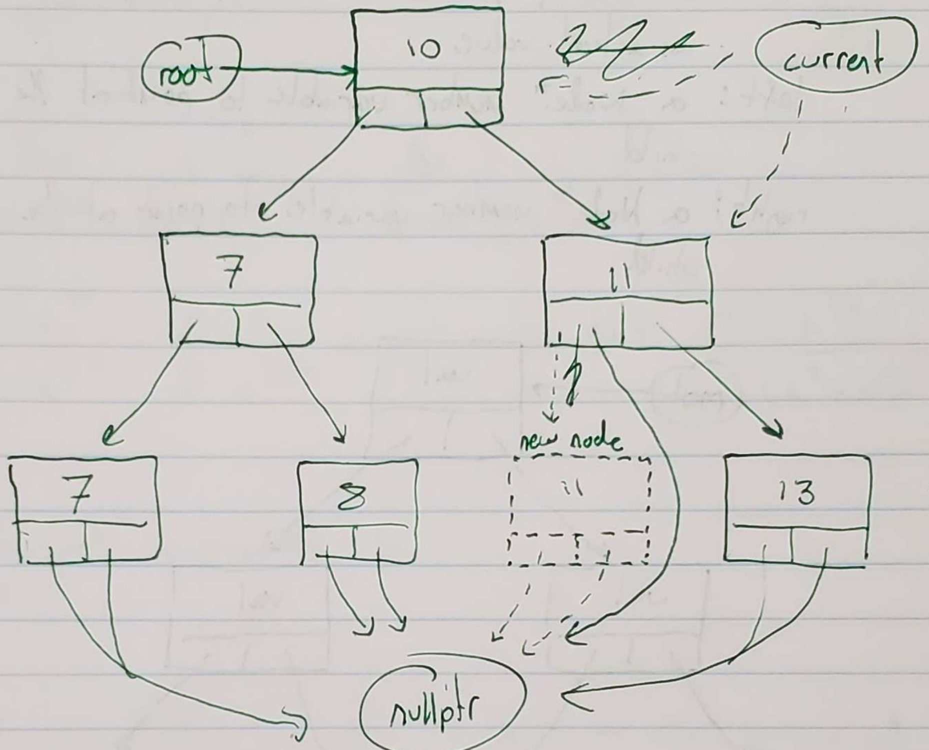
val: a ~~[illegible]~~ member variable (probably int) to store an actual value

left: a Node* member variable to point at the left child

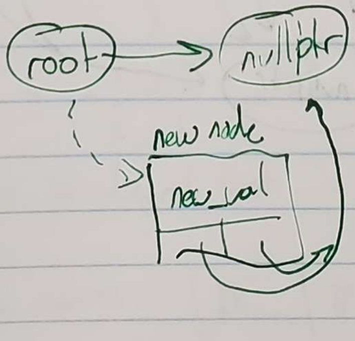
right: a Node* member variable to point at the right child



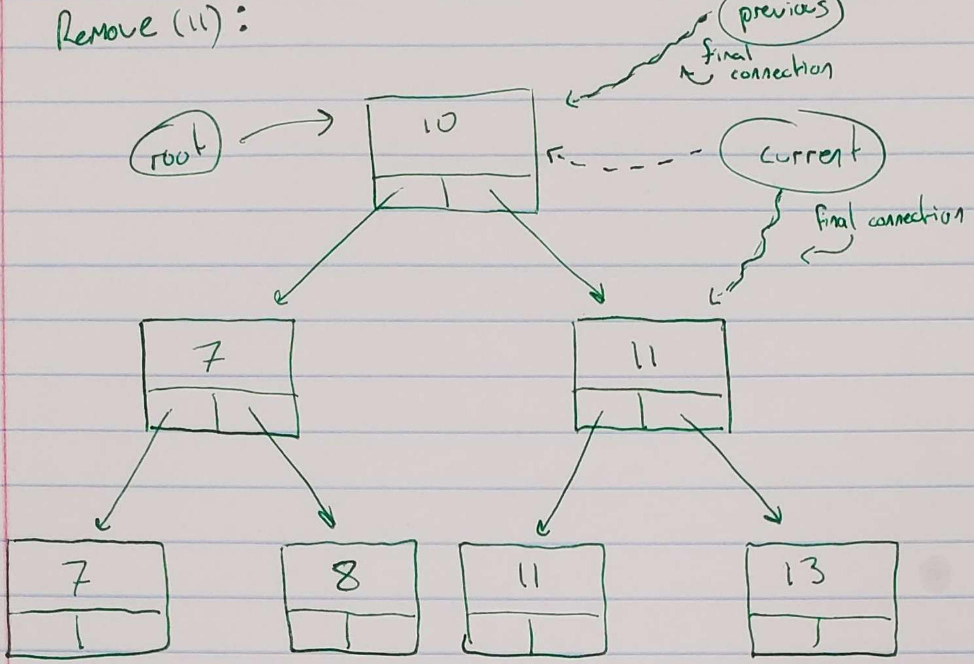
Insert (11):



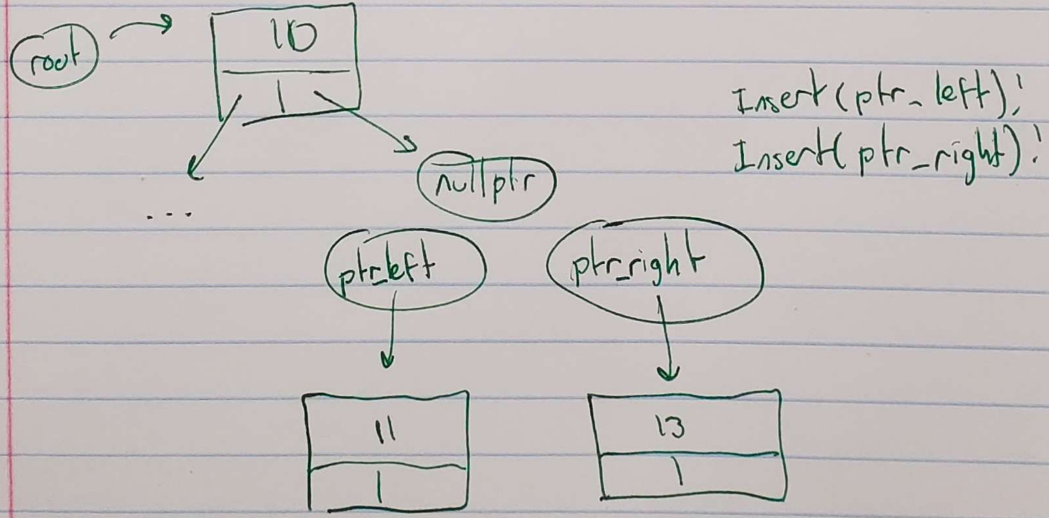
Insert to an empty tree:



Remove (11):



Iterate current through the BST untill desired value is found.
Set the correct direction of previous to nullptr (in this example we set previous→right = nullptr). Then make temporary pointers ~~with~~ to left and right nodes of what we want to delete. Then delete current.



Now insert ptr_left and then ptr_right using the same algorithm. The only difference is that this node that we insert does not point to nullptr, but instead has a sorted tree beneath it. That doesn't, however, change the algorithm.