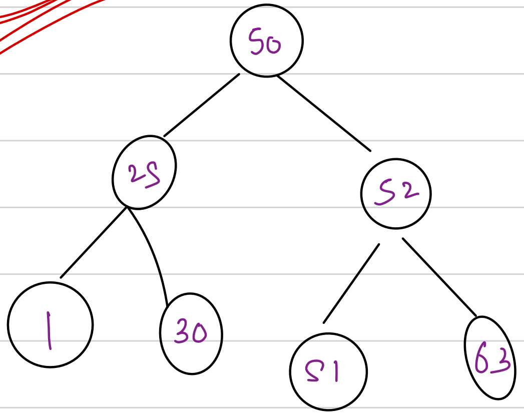
Binary Search Tree

can optimize

This DS improves the efficiency of searching

find 30



→ for any root node of a subtree in the whole tree

all the nodes on the LST of root are less than the root, & all in the RST are greater than root.

In the best case scenario, in one iteration, a BST will eliminate 1/2 of the nodes either from LST or RST

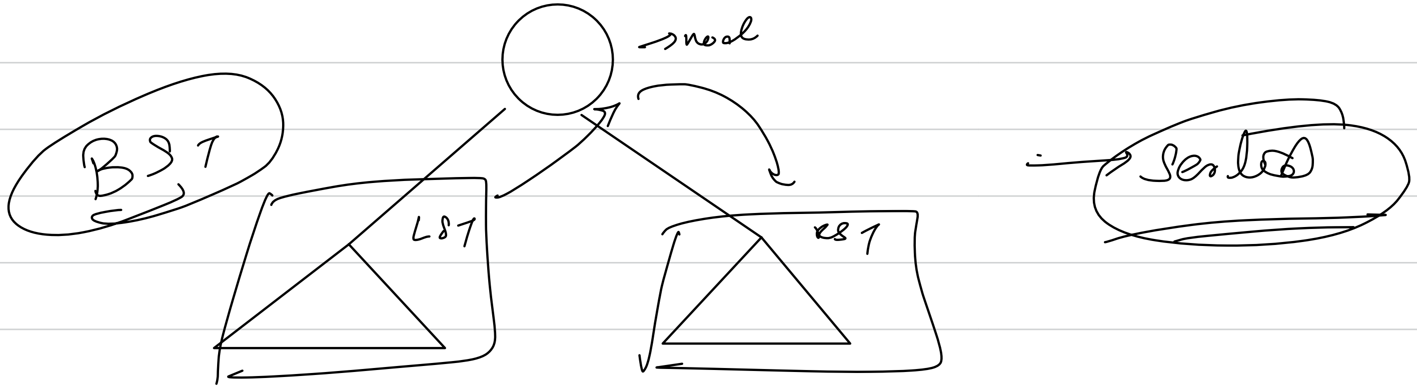
↳ The best case will occur for a B-BST and in a BBST TC of search = O(log n)

↳ Worst case can come for a skewed tree.
TC → O(n)

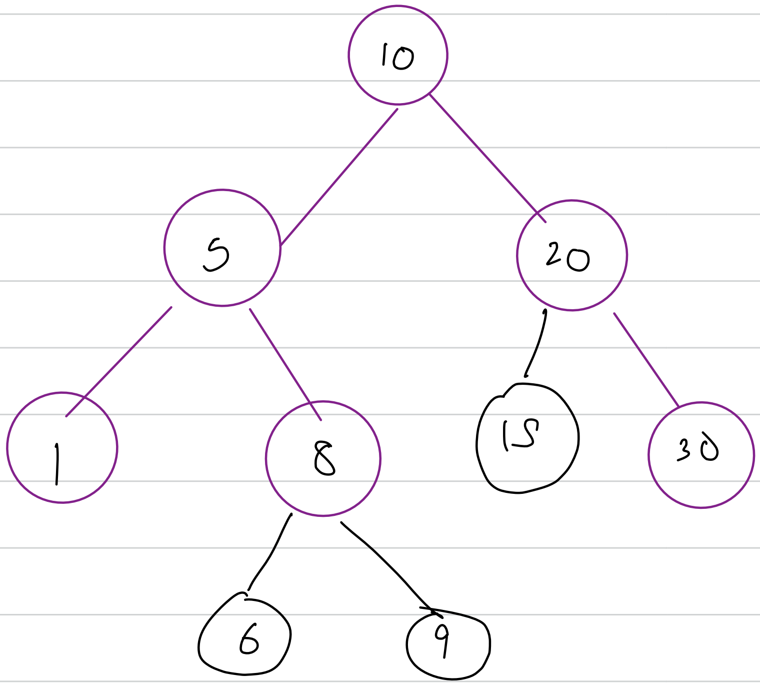
↳ General BST → O(height) → O(n)

↪ Due to the fact that all nodes in a LST is less than root & all nodes in the RST is greater that root,

the Inorder of a BST is always Sorted



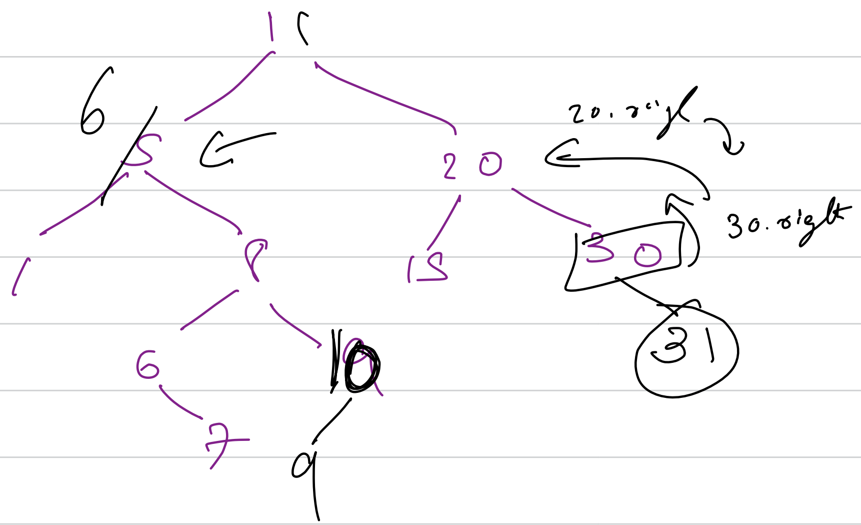
↳ How to insert an element in a BST



insert $\rightarrow$ 15

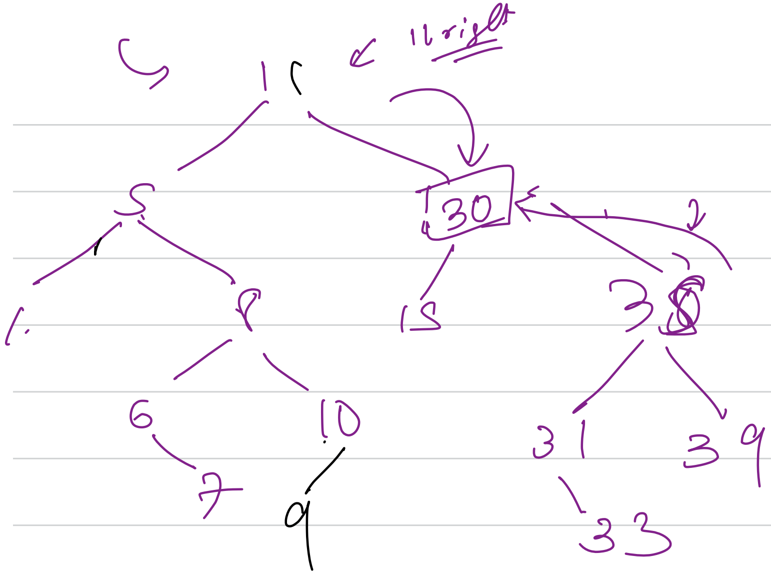
$\downarrow$

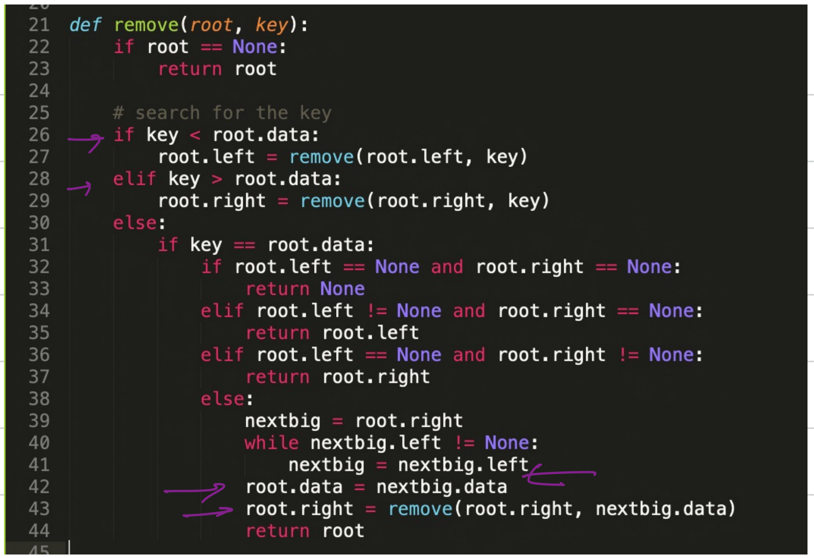
$O(h)$



10 5 1 8 6 7 9 20 15 30 — Pre

↳ we can find the leftmost node of RST, swap it with root, the call delete in the RST for the smallest node.




```python
def remove(root, key):
    if root == None:
        return root

    # search for the key
    if key < root.data:
        root.left = remove(root.left, key)
    elif key > root.data:
        root.right = remove(root.right, key)
    else:
        if key == root.data:
            if root.left == None and root.right == None:
                return None
            elif root.left != None and root.right == None:
                return root.left
            elif root.left == None and root.right != None:
                return root.right
            else:
                nextbig = root.right
                while nextbig.left != None:
                    nextbig = nextbig.left
                root.data = nextbig.data
                root.right = remove(root.right, nextbig.data)
                return root
```

level order level wise

→ VVI

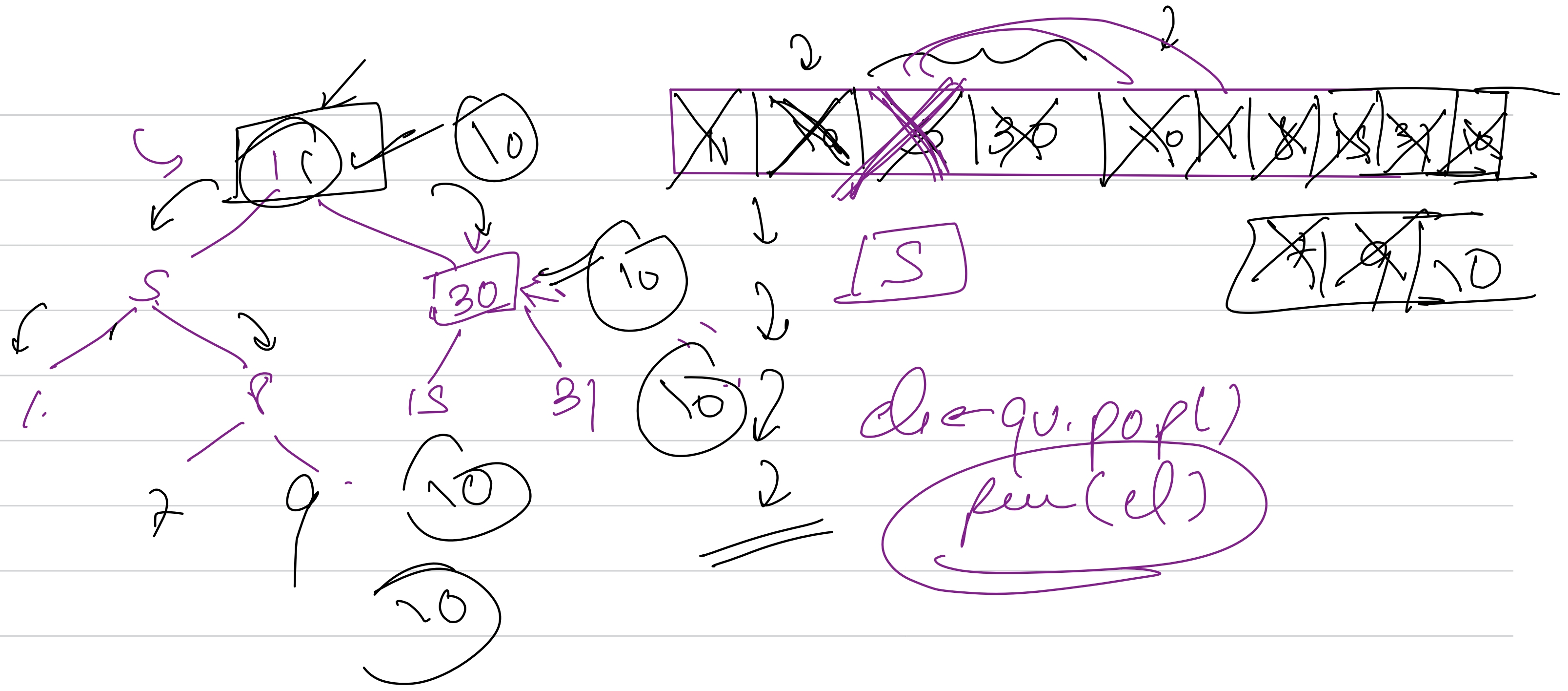
11
5 30
1 8 15 31
6 10
7 9

→ Tree level order → BFS → FCFS → queue

Preorder, Inorder, Postorder → DFS

~~4~~ ~~5~~ ~~3 0~~, 1, 8, 15, 37

To print level wise, we need an indicator which tells when to shift in a new line or when a level is completed


11
10
5
30
10
1
8
15
31
7
9
10
20
5
dequ.pop()
fun(el)