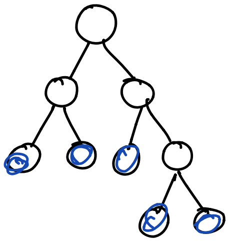
Some terminology

Full binary tree:

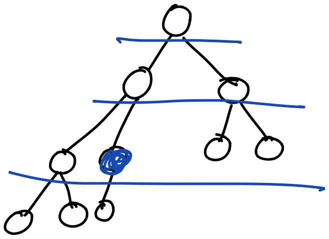
Every node other than the leaves has two children

(Every node has two children or no children)



Complete Binary Tree

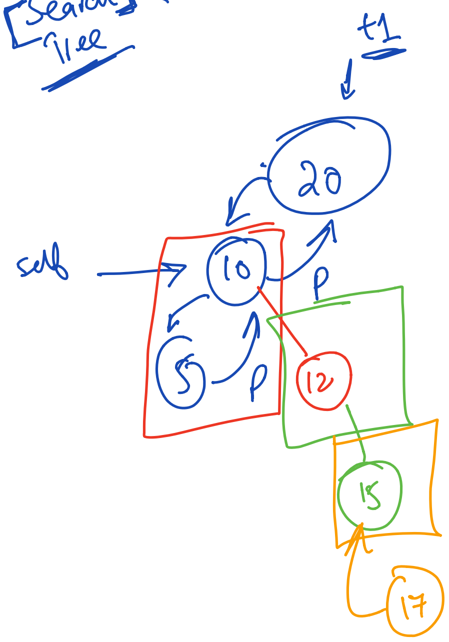
All levels are completely filled (except possibly the last — in which case, all leaves in there must be to the left.)

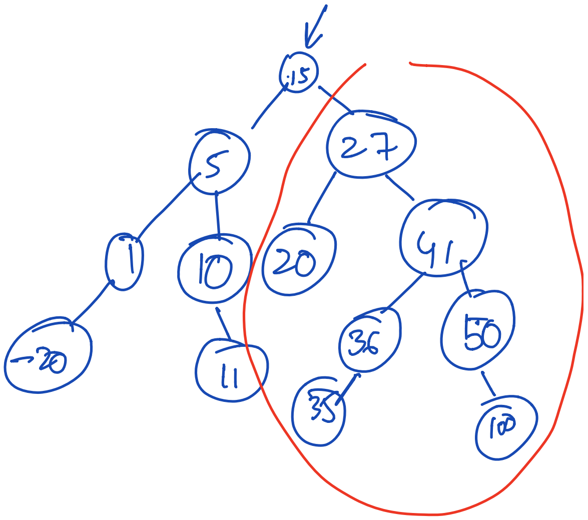


t1 = BST(15)

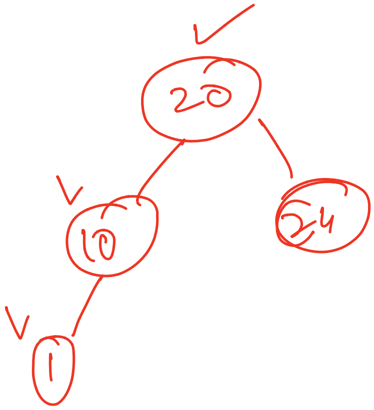
t1.insert(__)

Binary Search Tree

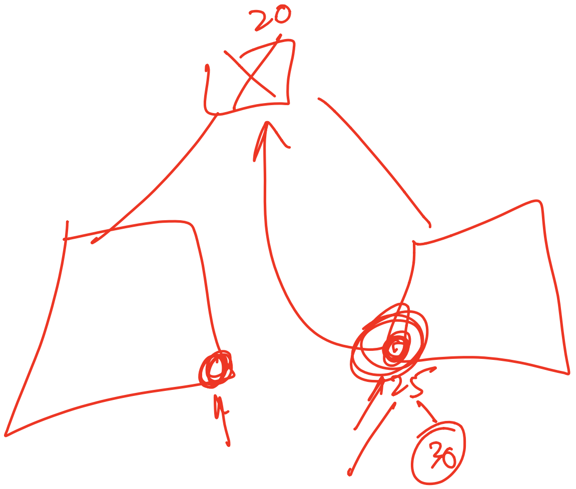


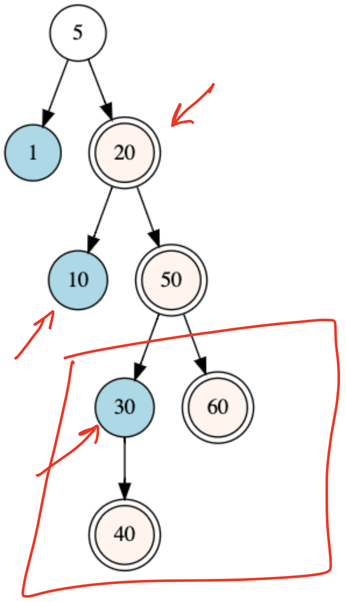


In_order



$\underbrace{1, 10, 20, 24}$


20
4
25
30

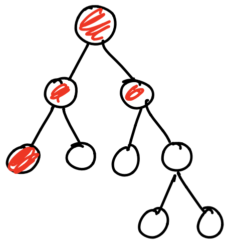

5
1
20
10
50
30
60
40

Full binary Tree

Each node —

0 or 2

children

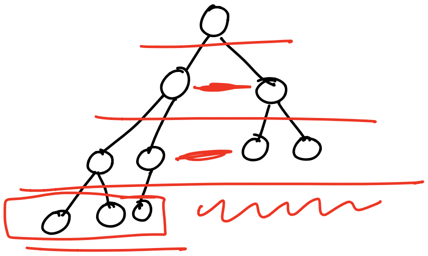


Complete binary Tree

All levels filled

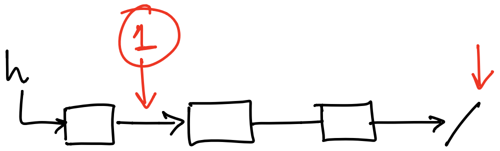
(possibly) except

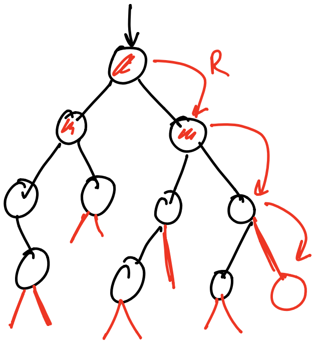
last ⟶ to left.



Linked

Insertion

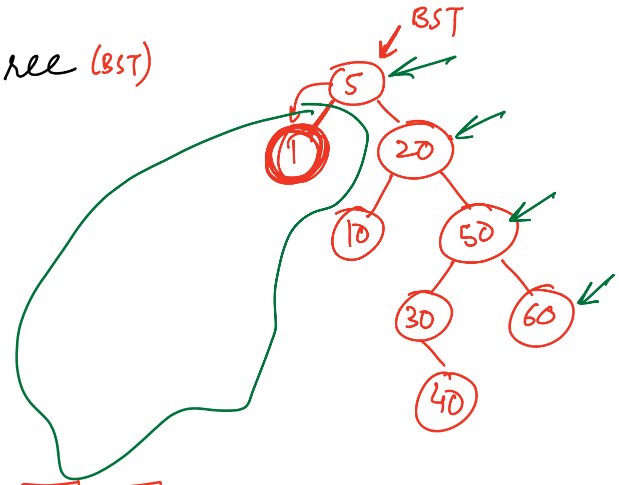




TreeNode

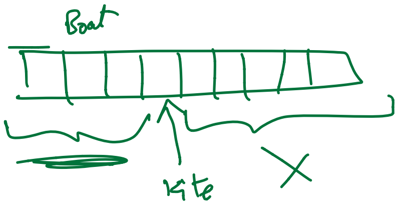
↳ Binary Search Tree (BST)

- Binary
- Search



Insert :

5, 1, 20, 10, 50, 30, 40, 60



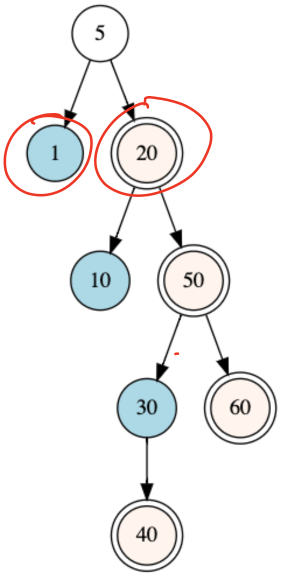
Searching

In-order traversal

< = >

L , I , R

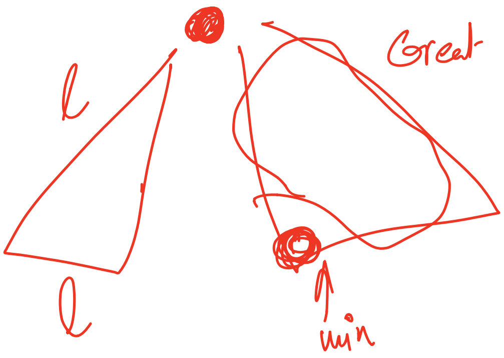
1 , 5, 10, 20, 30, 40, 50, 60

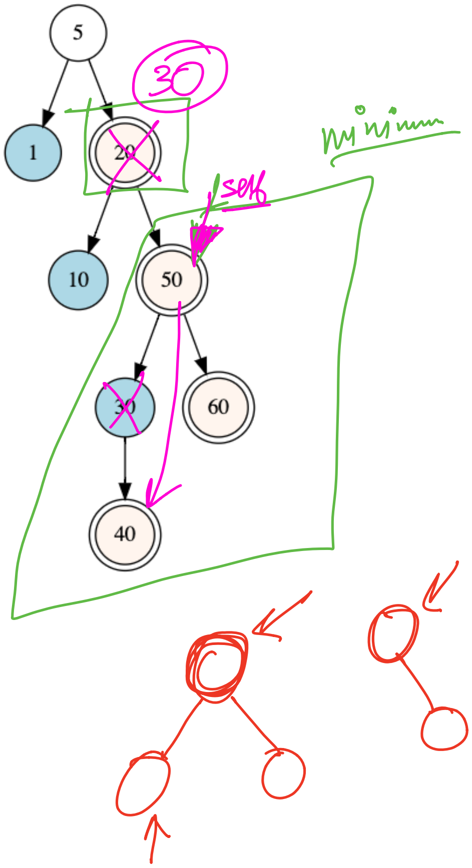


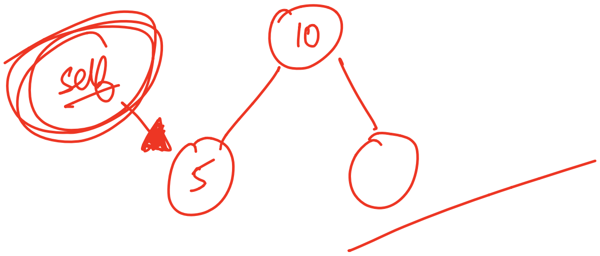
Deletion:

Case 1 : No children
(leaf)

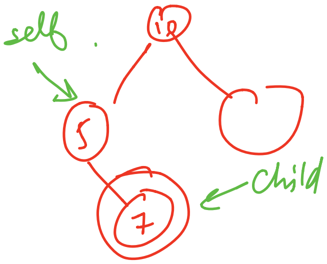
Case 2 : One child
(either child)







$\overbrace{\text{self . parent}}^{10}$



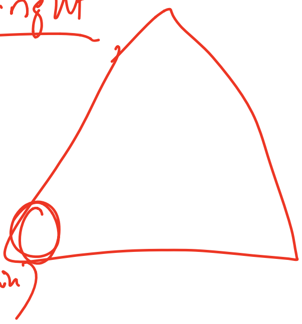
"find child"

```
if self.parent.right == self:
    self.parent.right = node
if self.parent.left = self:
    self.parent.left = node.
set_for_parent(self, node)
```

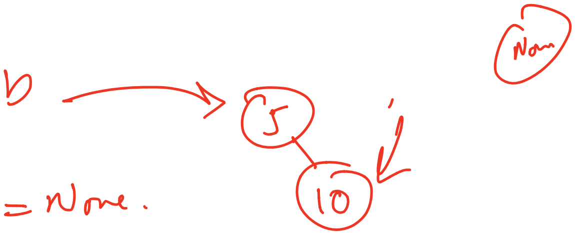
```
if self.right:
    child = self.right
if self.left:
    child = self.left
self.set_for_parent(child)
```

min of self.right

"replace"



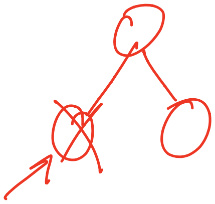
self.right.delete(min)



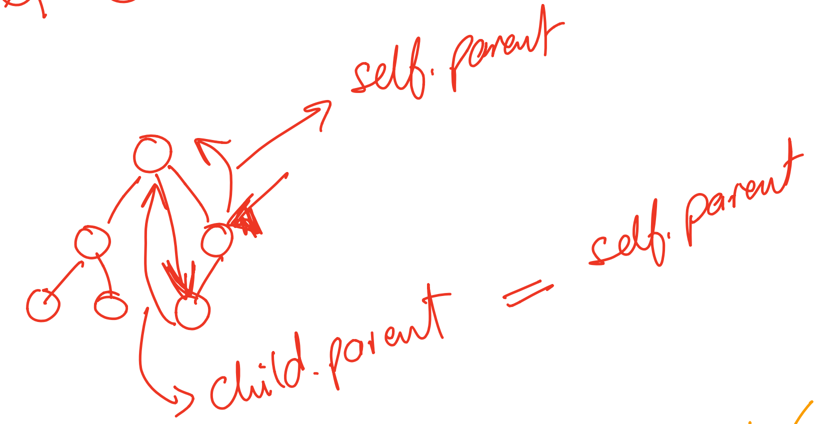
b = None.

$b = \underbrace{b.delete(5)}_{None}$

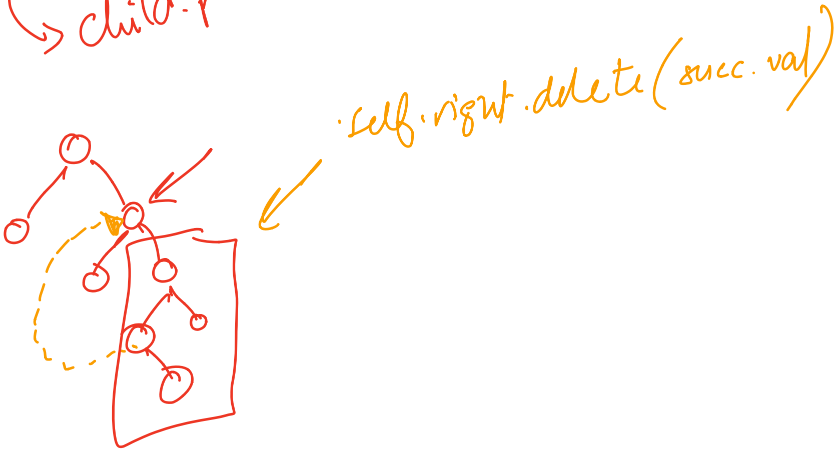
Case 1

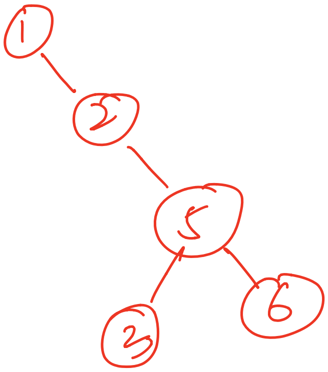


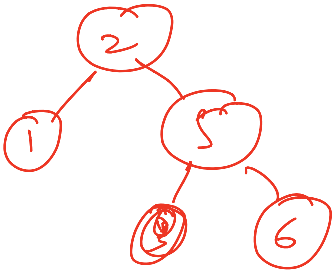
Case 2



Case 3:







RB trees

AVL trees