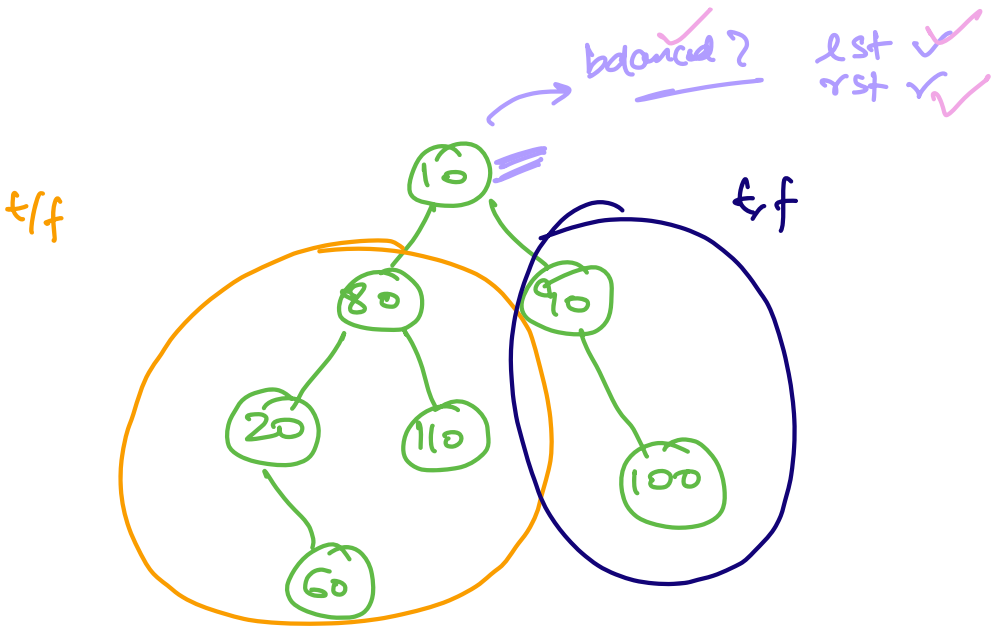
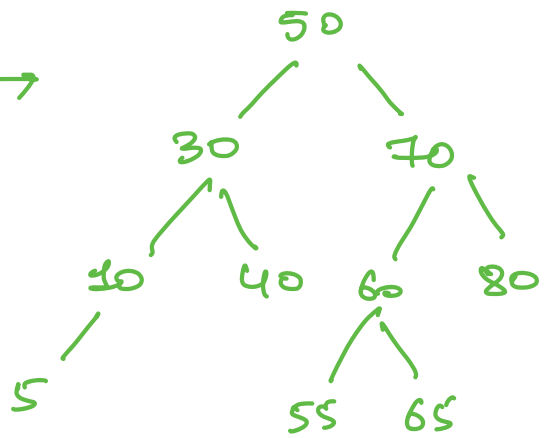


Binary search Tree:

→ BT + left node data < mid < right node data

BT, BST
node map 2 child

BST →



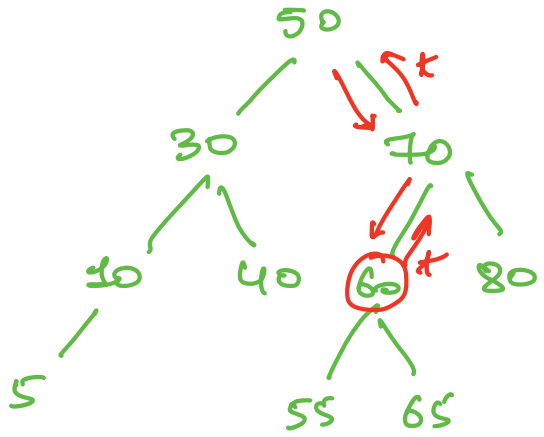
Diameter
↓
BT, BST

Inorder trav: L N R

< N > N

BST Inorder traversal always be in inc order

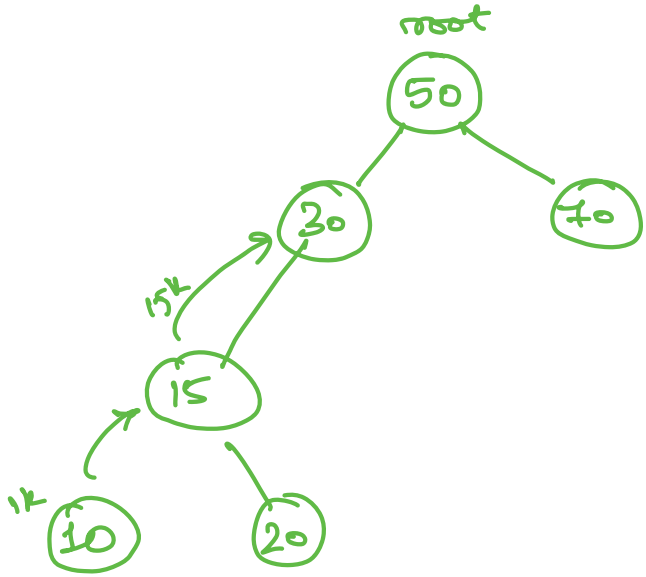
BT is BST ?



find	BT: O(n)	BST: O(h)
max	BT: O(n)	BST: O(h)
ht	Structure	BT ✓
size	Structure	BT ✓
sum	Structure	BT ✓

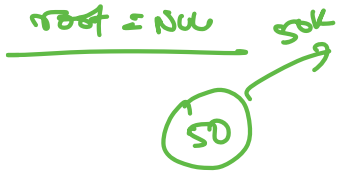
right child = NULL ←

BST Create

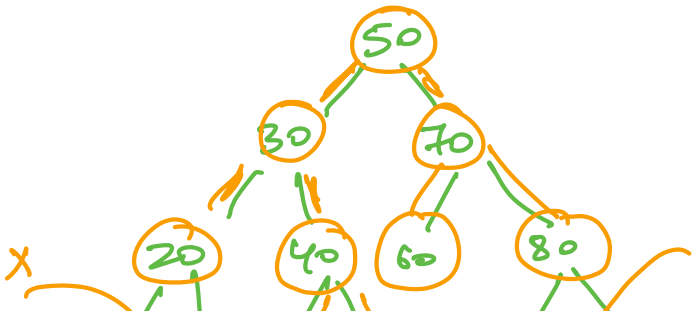


add(50)
add(30)
add(70)
add(15)
add(20)

add(10)



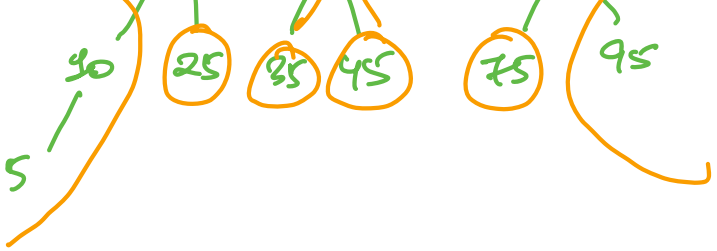
Q:



Range: (25, 75)

BST property

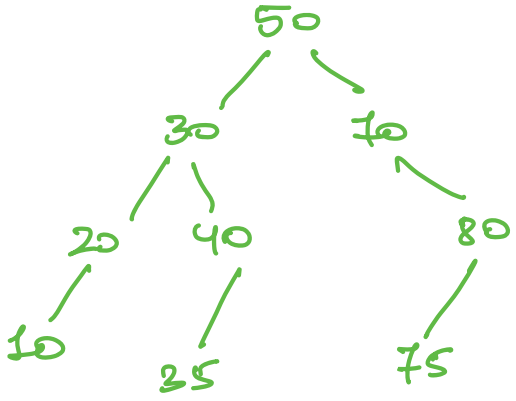
n.d < RR rst



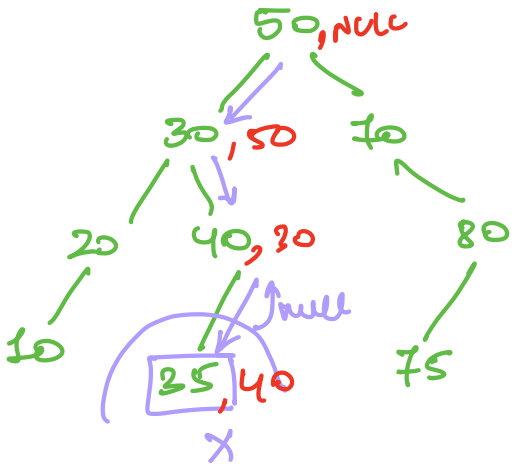
n.d > ul left

ul ≤ n.d ≤ ll left right

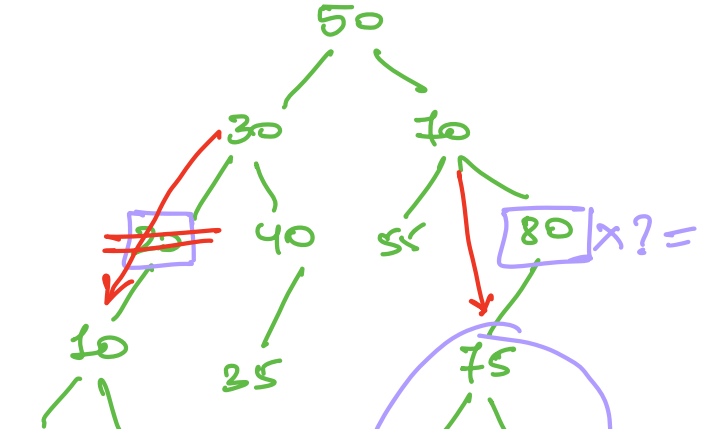
Q: Remove



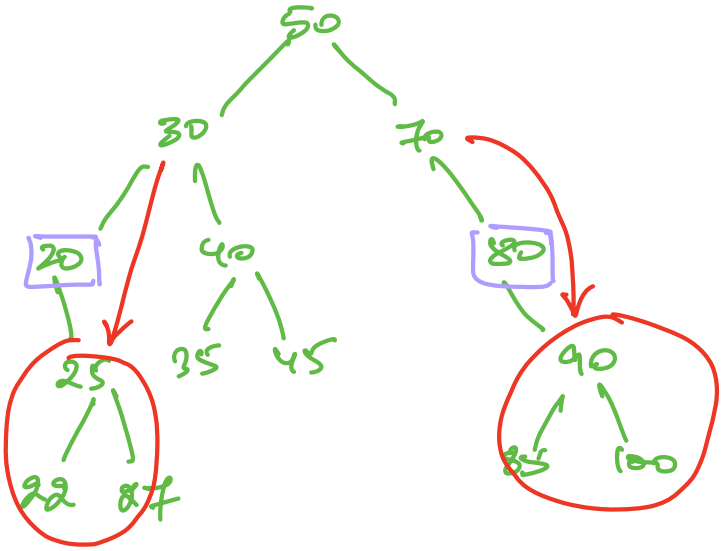
Case 1: Remove left child = NULL
right child = NULL
} leaf node



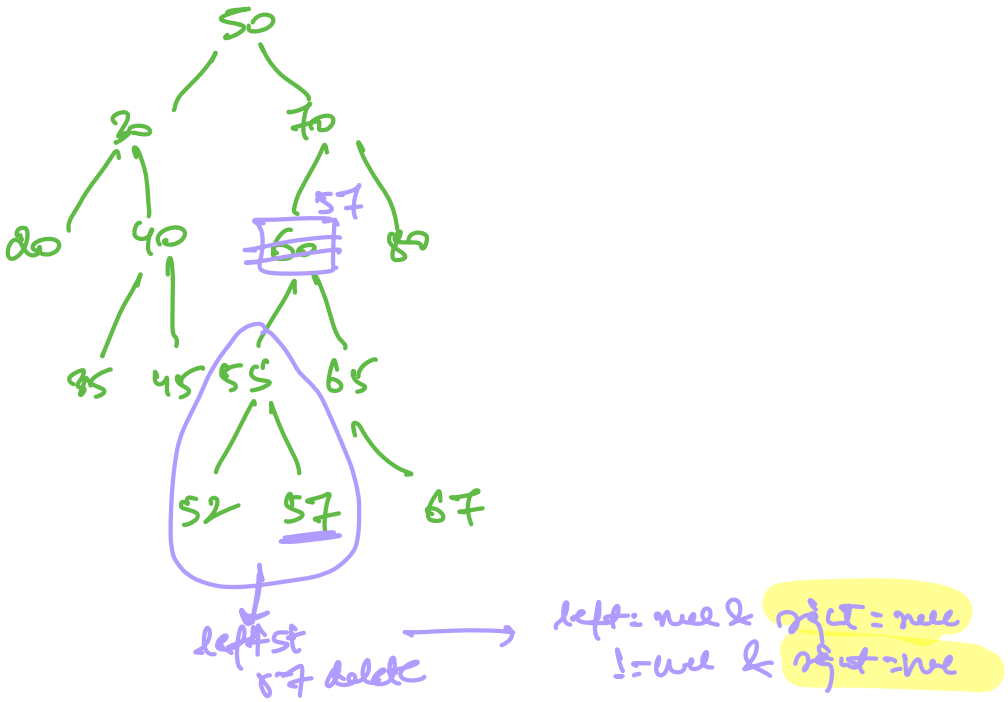
Case 2: left child exist ✓
right child exist X



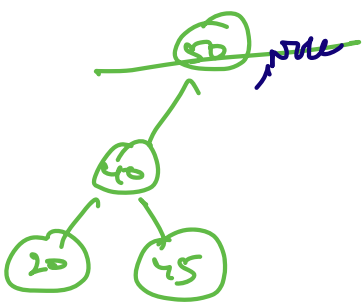
Case 3:

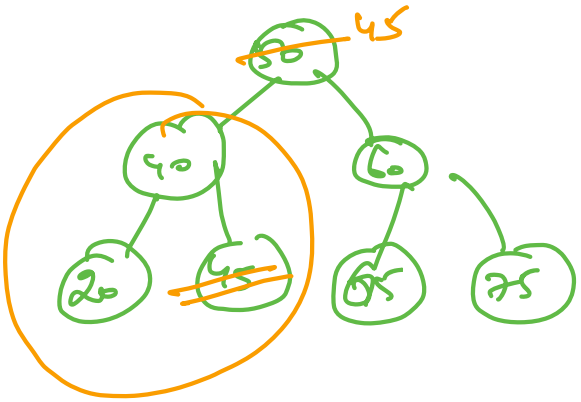


Case 4:

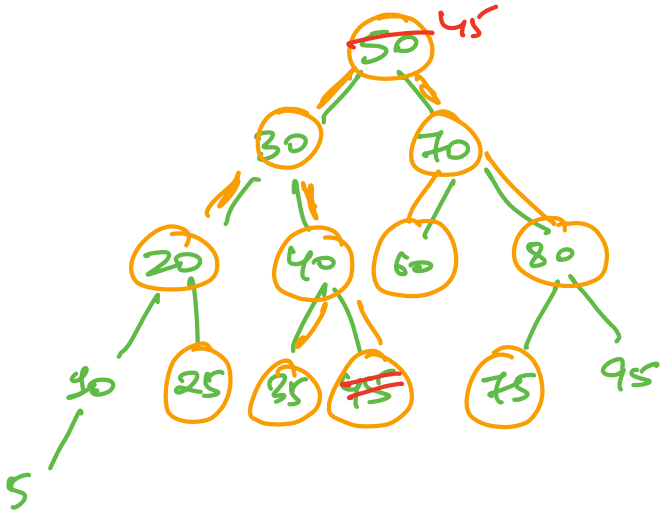


BCE



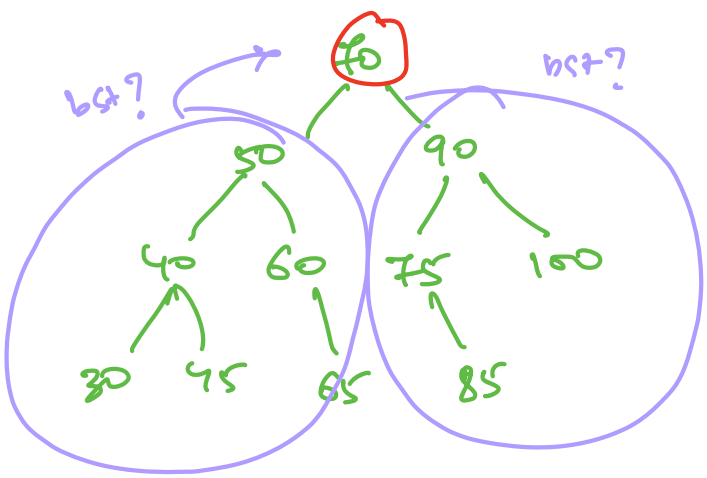


Delete 50



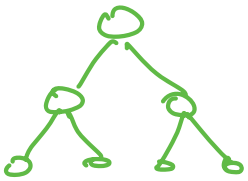
Q: Is BT a BST ?

Inorder



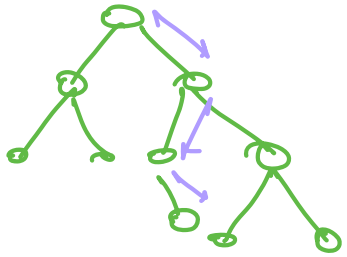
AVL Trees (Self Balancing Trees)

BT find:



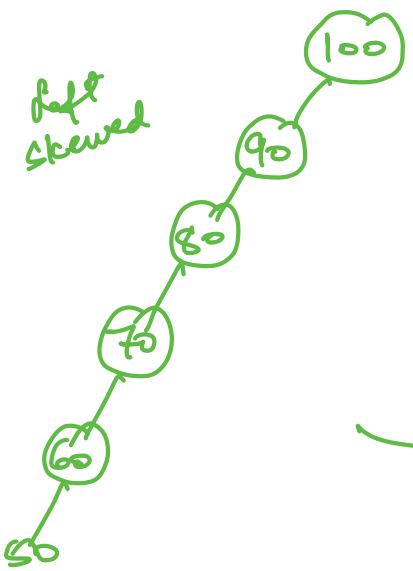
TC: O(n)
↳ no. of nodes

BST find :

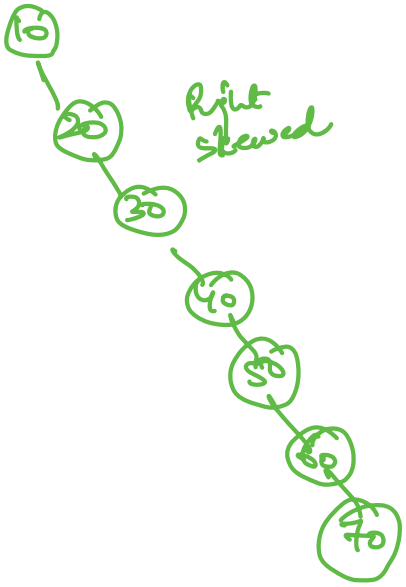


TC: O(h)

BST Unbalanced



left skewed



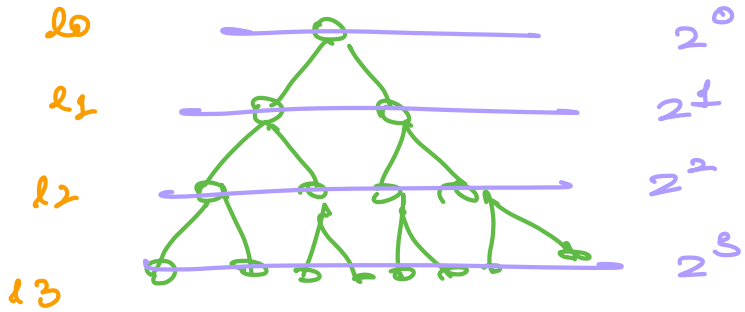
Right skewed

BSTs (Skewed)

h = n

TC: O(n)

BST Balanced



total no. of nodes : $2^0 + 2^1 + 2^2 + 2^3 \cdots 2^h$

$$n = \underbrace{2^0 + 2^1 + 2^2 + 2^3 \cdots 2^h}$$

$$n = \frac{2^{h+1} - 1}{2 - 1}$$

$$n = 2^{h+1} - 1$$

$$n+1 = 2^{h+1}$$

$$\log_2(n+1) = h+1$$

$$\log_2(n+1) - 1 = h$$

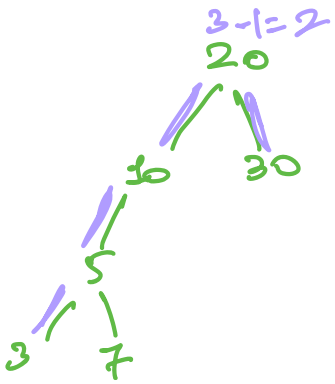
$$h = O(\log_2 n)$$

AVL ⟶ BST + bf = -1, 0, 1

↳ self balancing trees

bf = -1, 0, 1

↳ left ht − rt ht



st method: 2 − 0 = 2

4 different cases:

Case 1: LL case

left left

Eg.

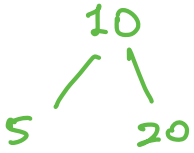
adds

① normally add BST

② new node upward do^n first unbalanced node

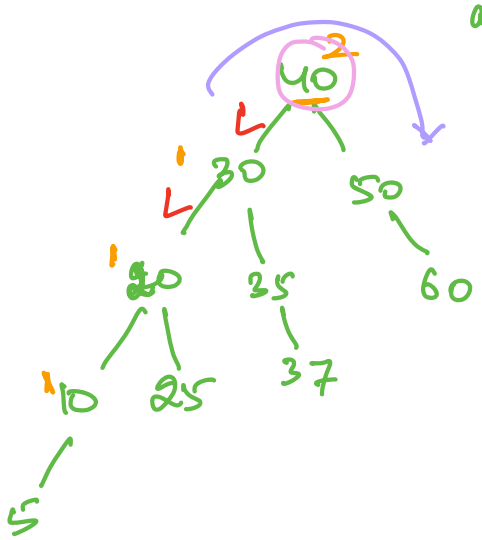
③ unbalanced node
↓
newly added node

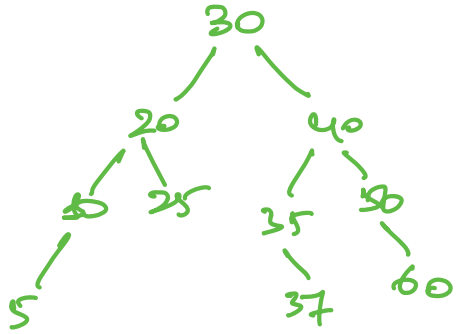
Capture initial movement



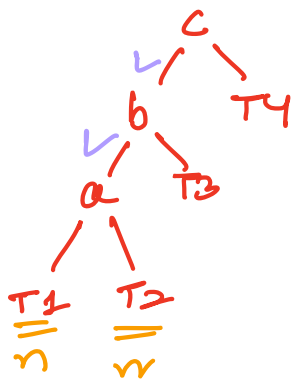
Eg:

adds





SOLUTION for LL Case: Single Right Rotation



n: new node

c: balancing factor violate

b → c left child

a → b left child

Single Right Rotation