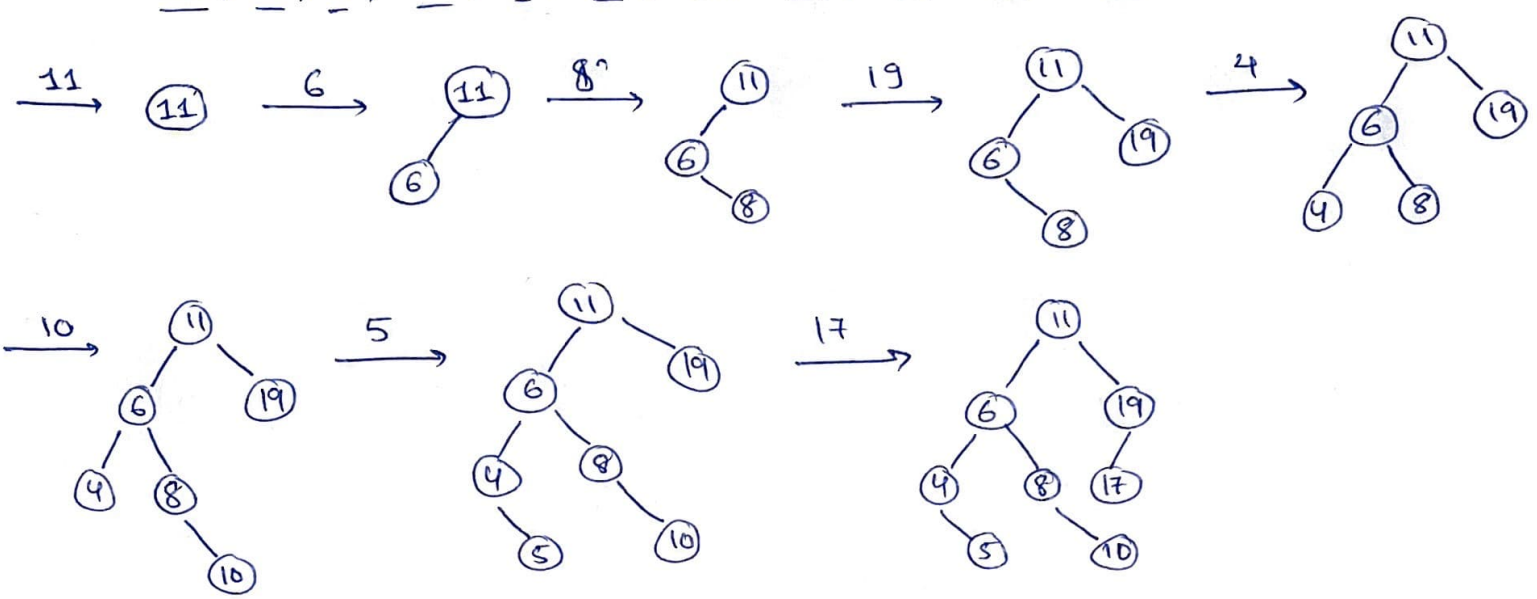
Binary Search Trees (BST)

- get(K) ⟶ returns value V associated with key K, if exists
- put(K, V) ⟶ associates value V with key K
- remove(K) ⟶ removes the entry with K & returns value, or null

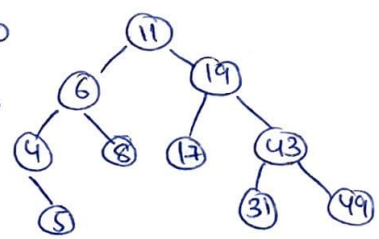
⇒ Insertion [Draw BST by inserting from left to right]

11, 6, 8, 19, 4, 10, 5, 17, 43, 49, 31

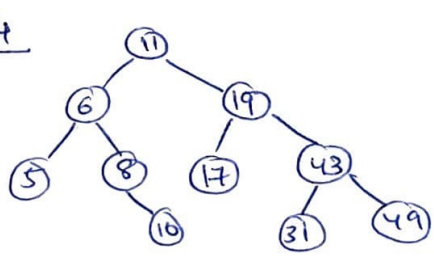


⇒ Deletion – ① 0 Child → Directly delete
② 1 Child → Replaced with its child itself
③ 2 children → (i) inorder predecessor
(ii) inorder successor
> replace

① Del 10



② Del 4



③ Del 11

(i) Replace 11 with 10

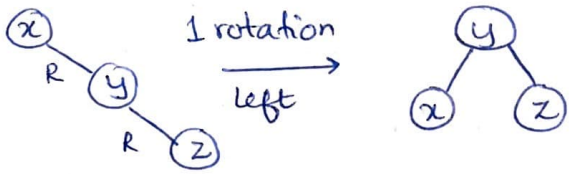
(ii) Replace 11 with 17

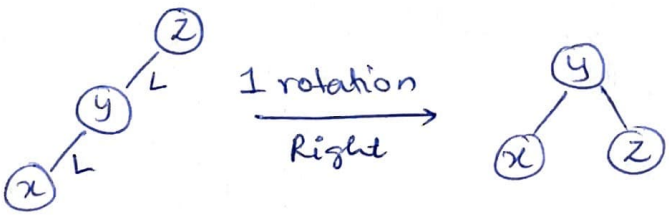
AVL Trees

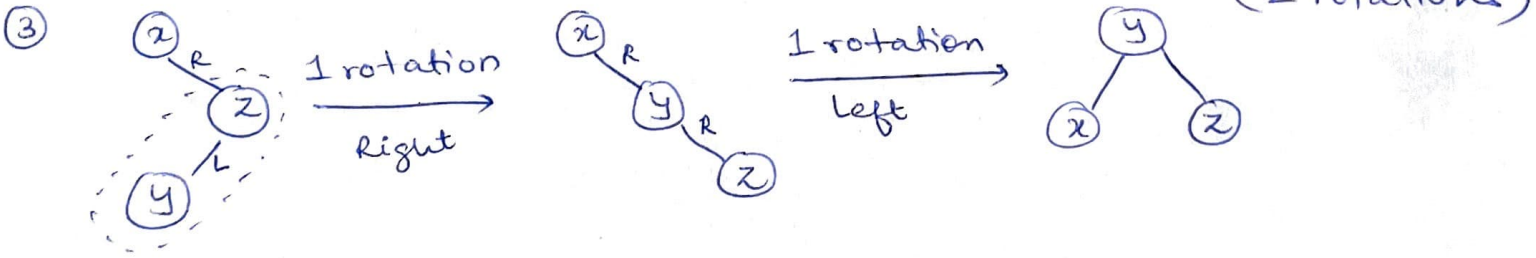
- It is a BST
- The height of left subtree − The height of right subtree is $\{-1, 0, 1\}$ (Balance Factor)

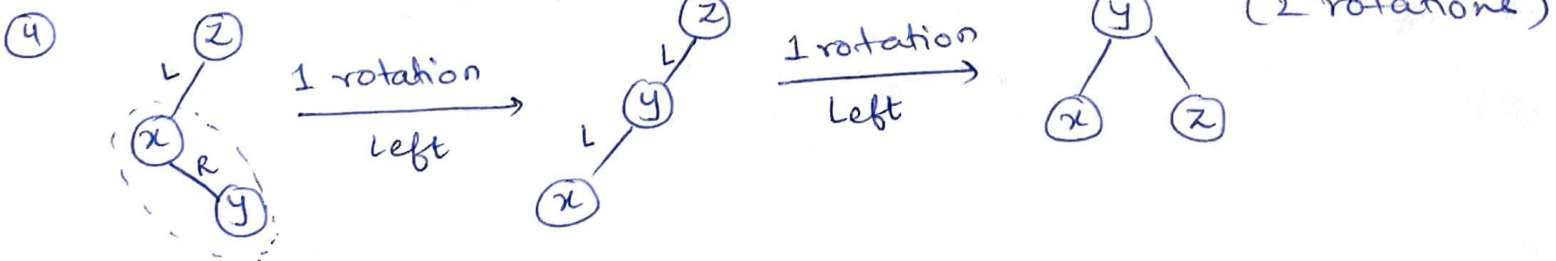
Rules:-

Consider : $x < y < z$

① 

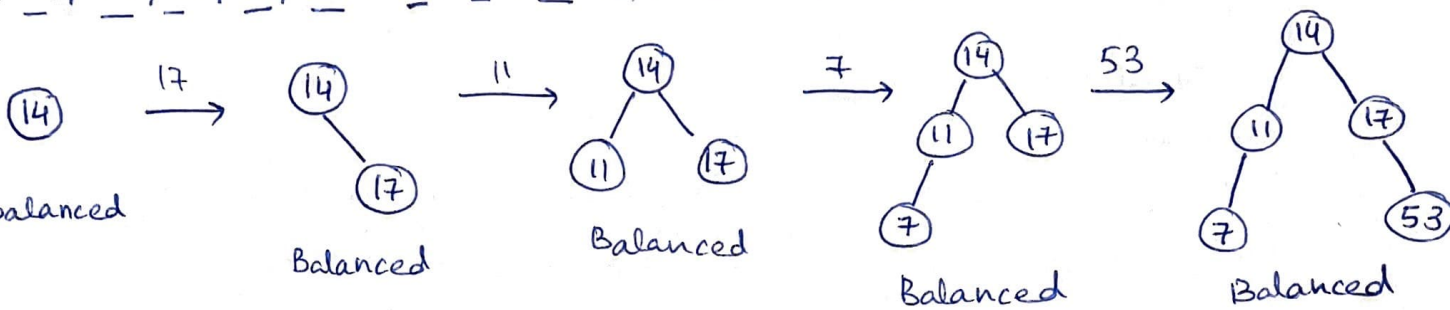
② 

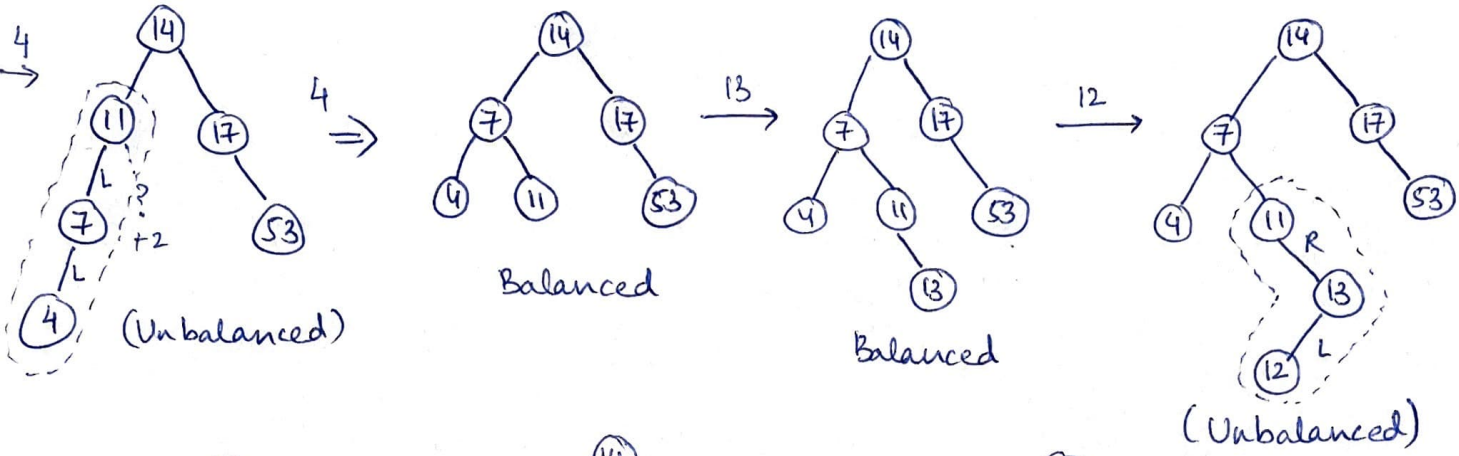
③ 

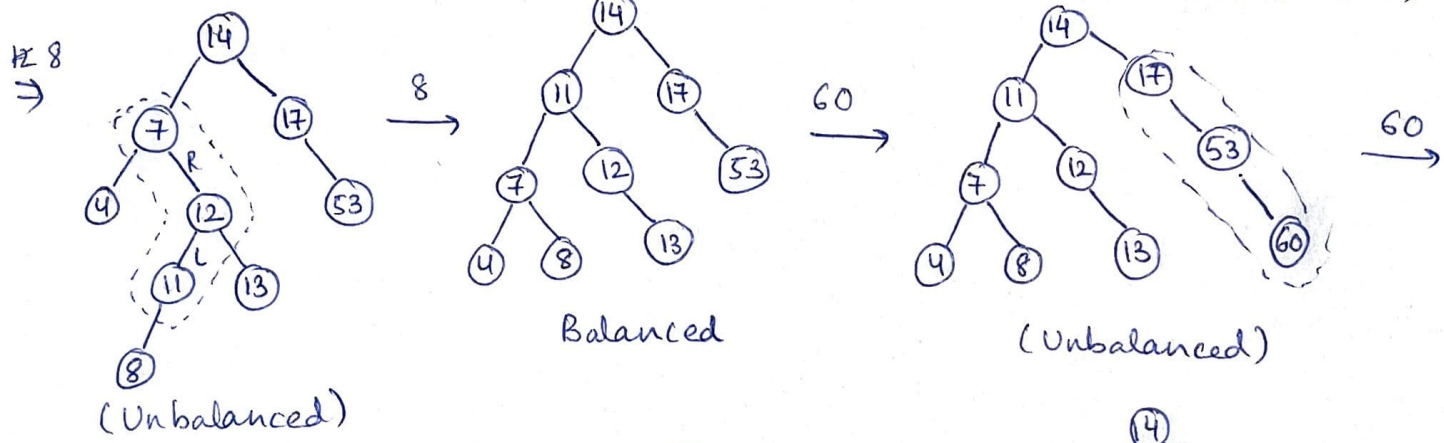
④ 

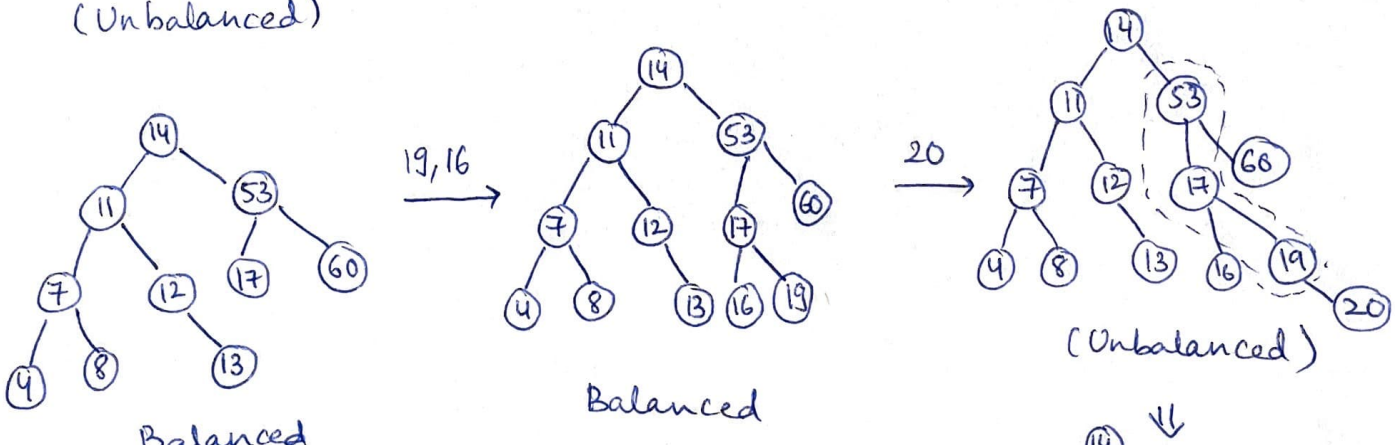
AVL Insertion

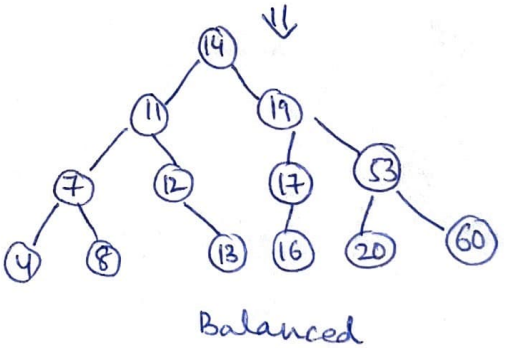
[14, 17, 11, 7, 53, 4, 13, 12, 8, 60, 19, 16, 20]





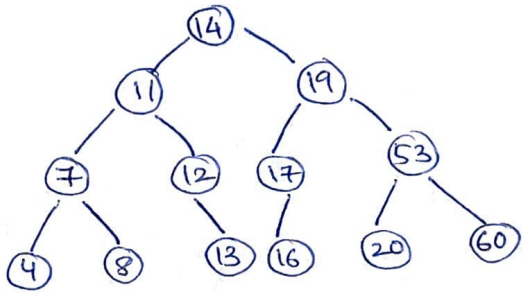




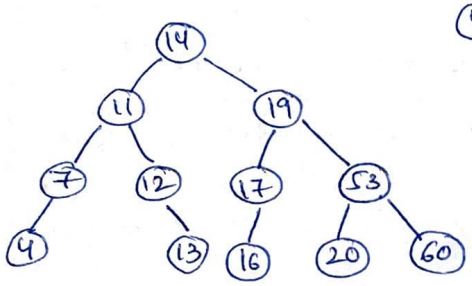


AVL Deletion

[8, 7, 11, 14, 17]

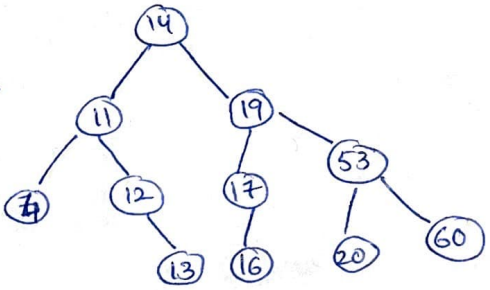


Del 8 →



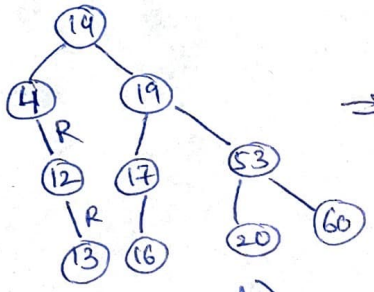
Balanced

Del 7 →

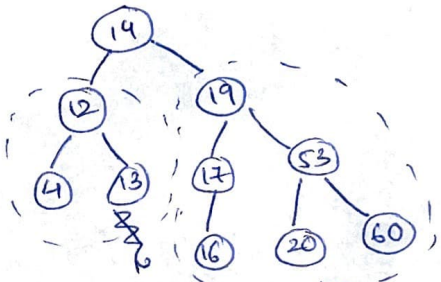


Balanced

Del 11 →



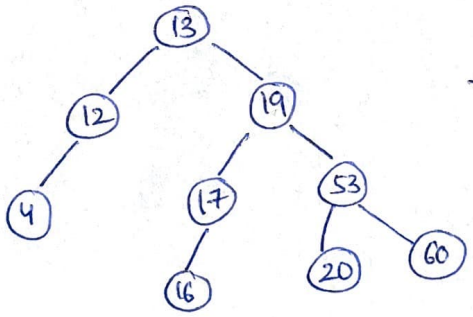
(unbalanced) ⇒



Balanced

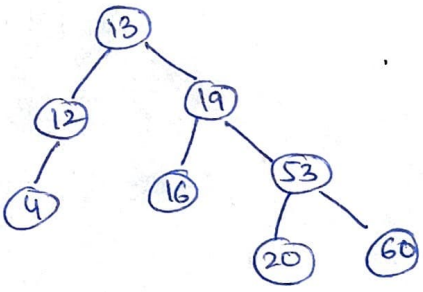
(To remove 14)

Del 14 →



Balanced

Del 17 →



Balanced