CS & IT ENGINEERING

Data Structures & Programming

Tree

Lec- 05

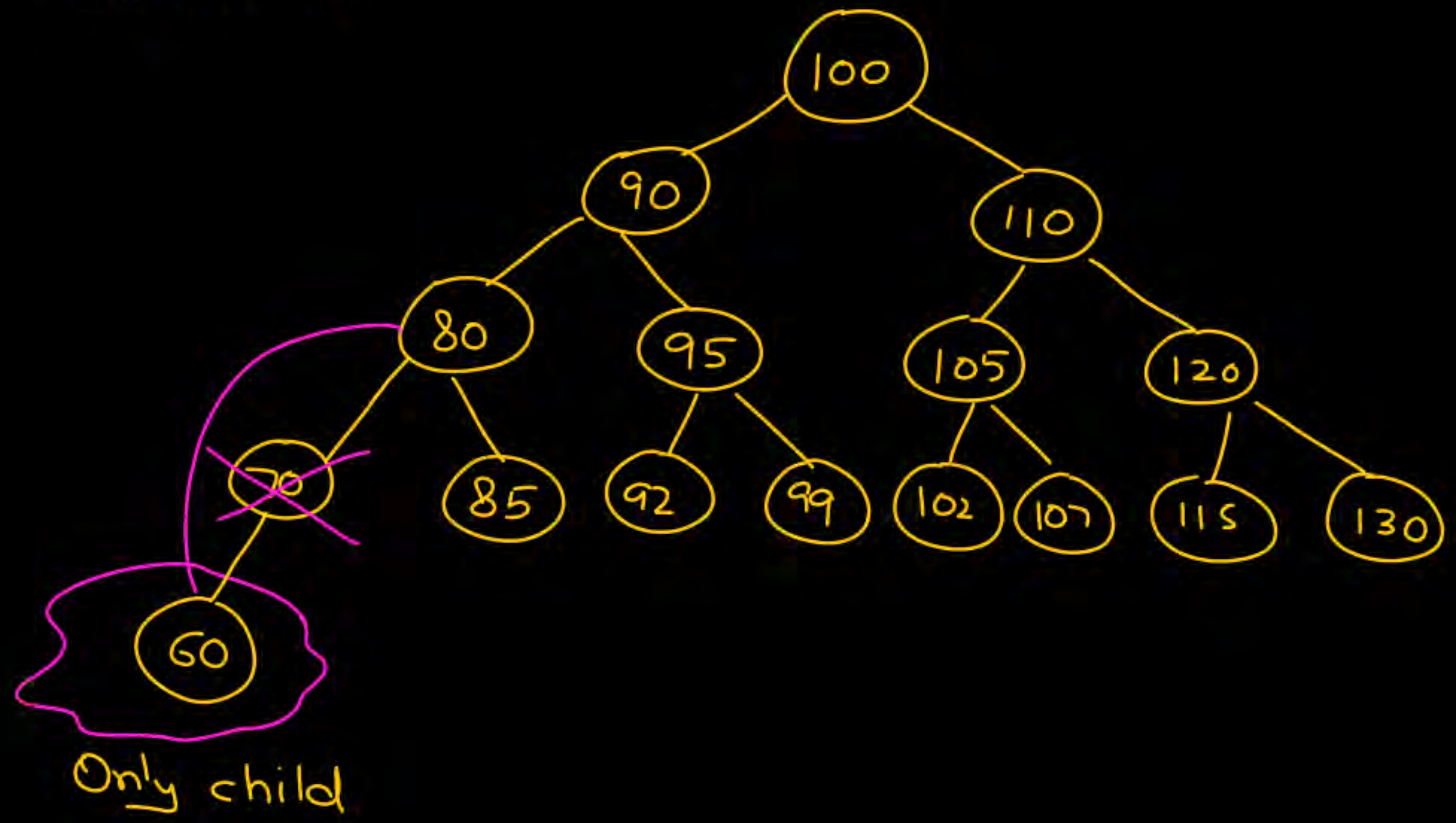
By- Pankaj Sharma Sir

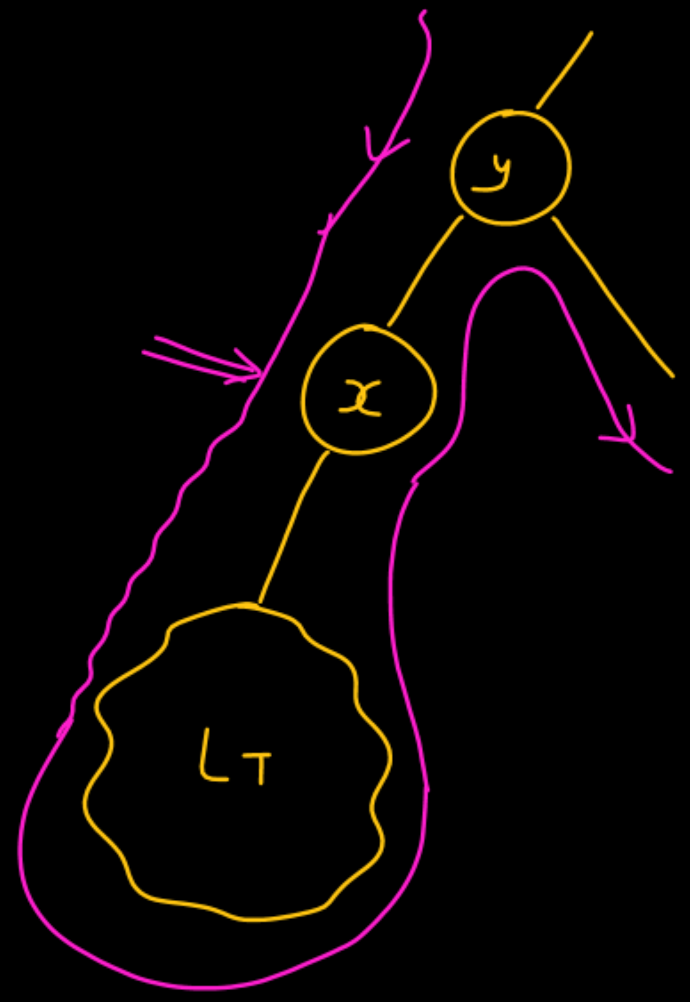
TOPICS TO BE COVERED

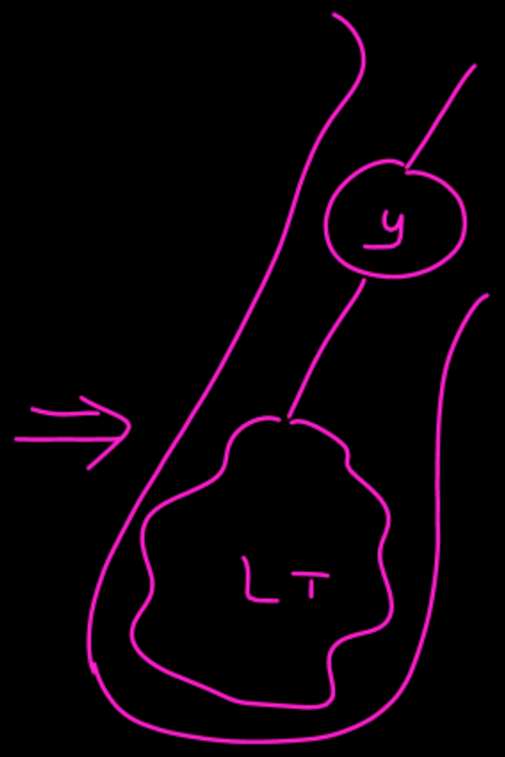
Tree 05

Deletion of a node having One child

Delete 70







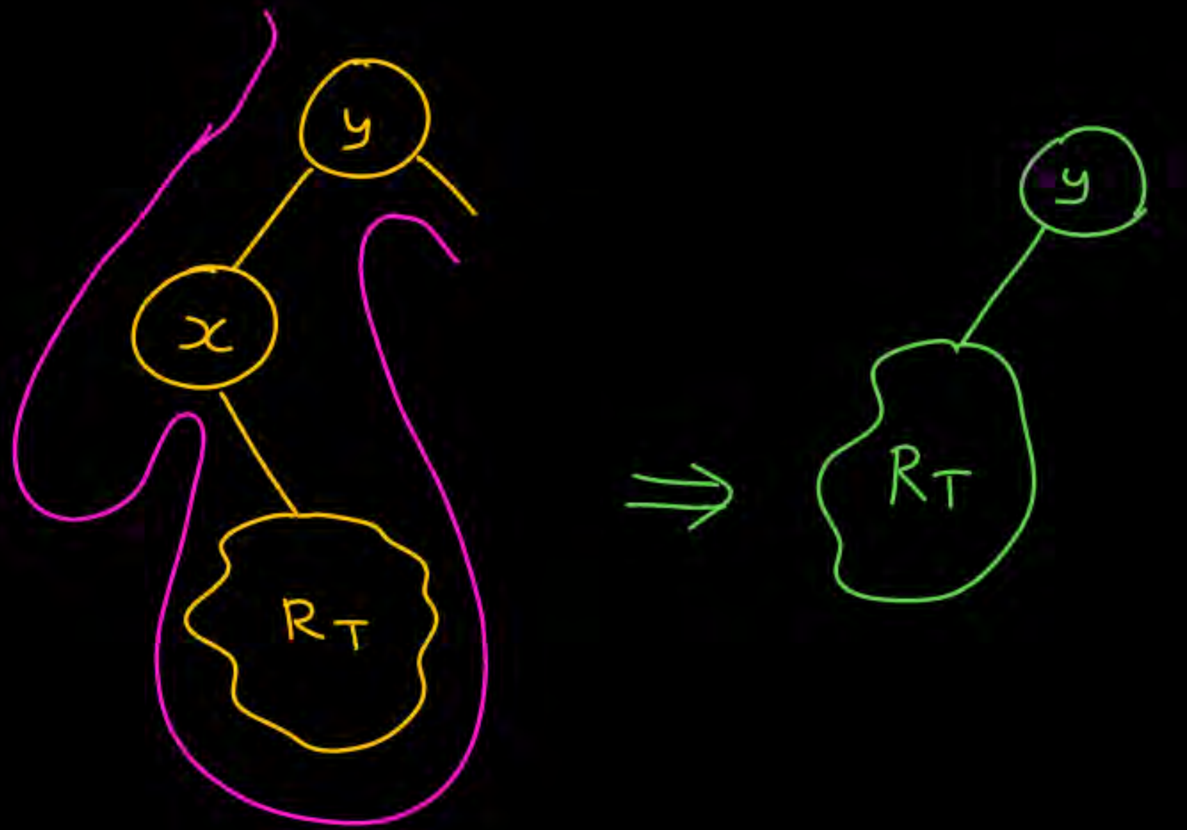
Inorder : L_T, $\cancel{x}$, y,

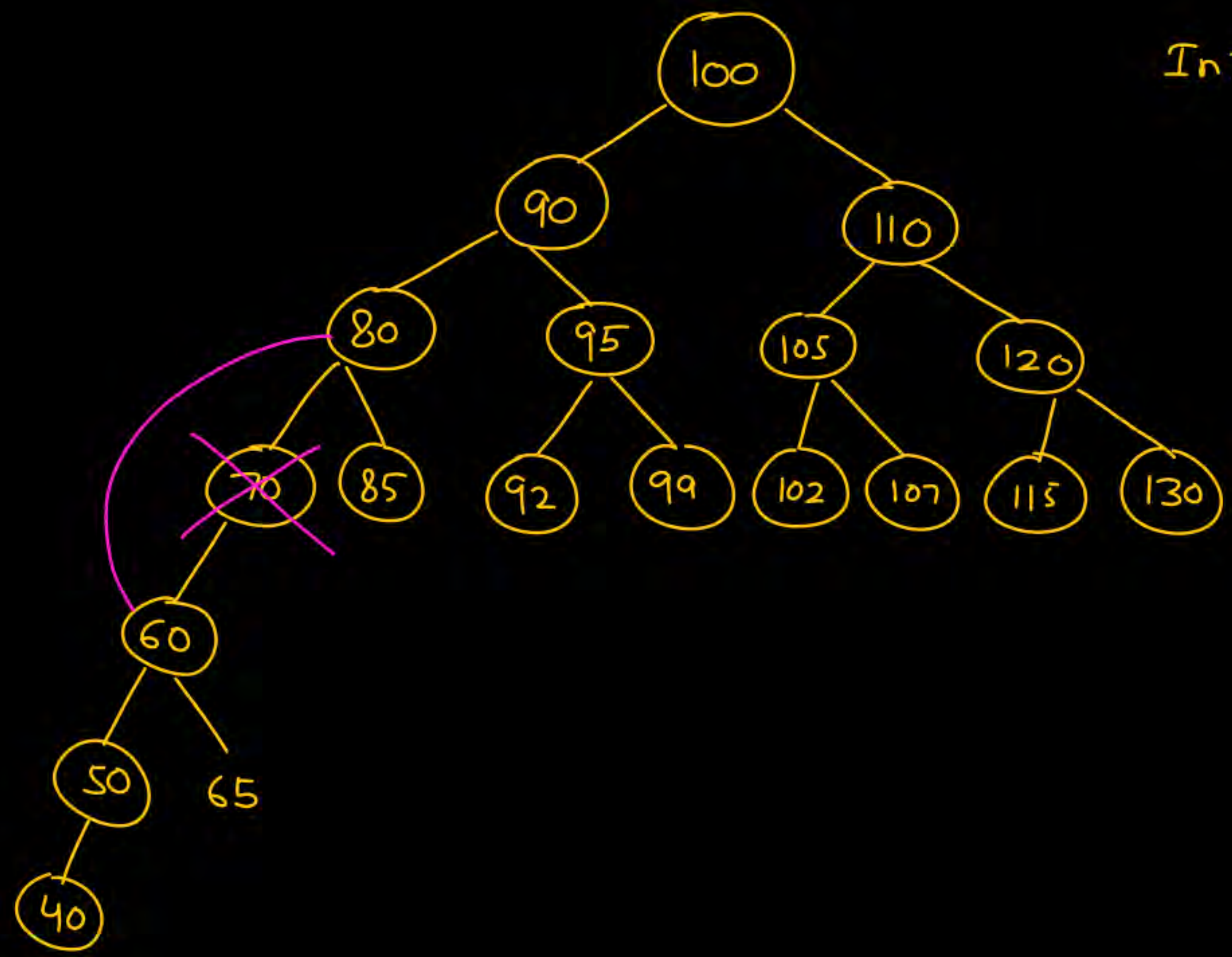
$\underbrace{L_T}$, y

L_T, y

InOrder: $\cancel{x}, R_T, y$

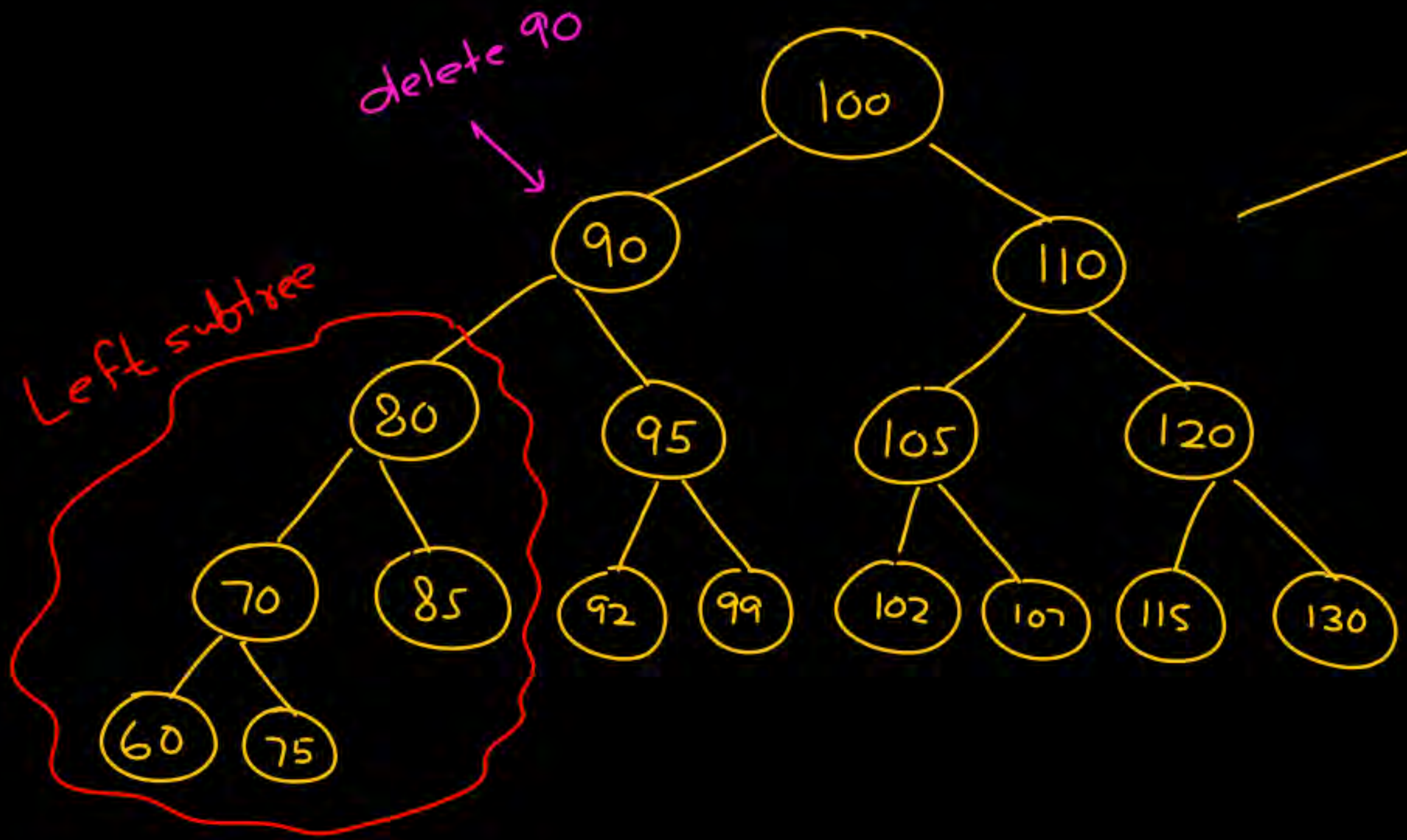
R_T, y



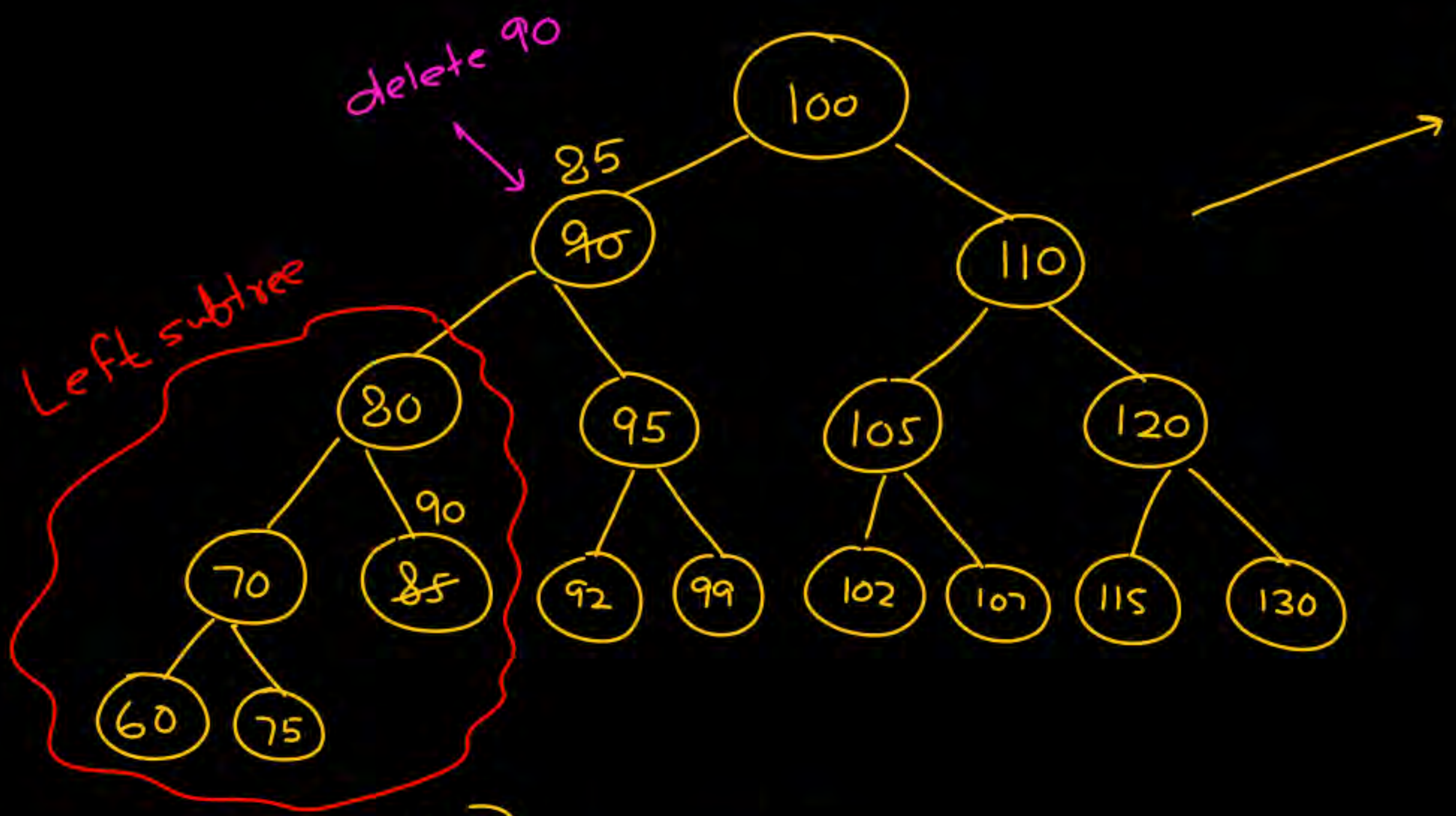


In: $\underbrace{40, 50, 60, 65}_{LT}$, ~~70~~, 80, 85, 90, 92, 95, 99, 100,

102, 105, 107, 110, 115, 120, 130

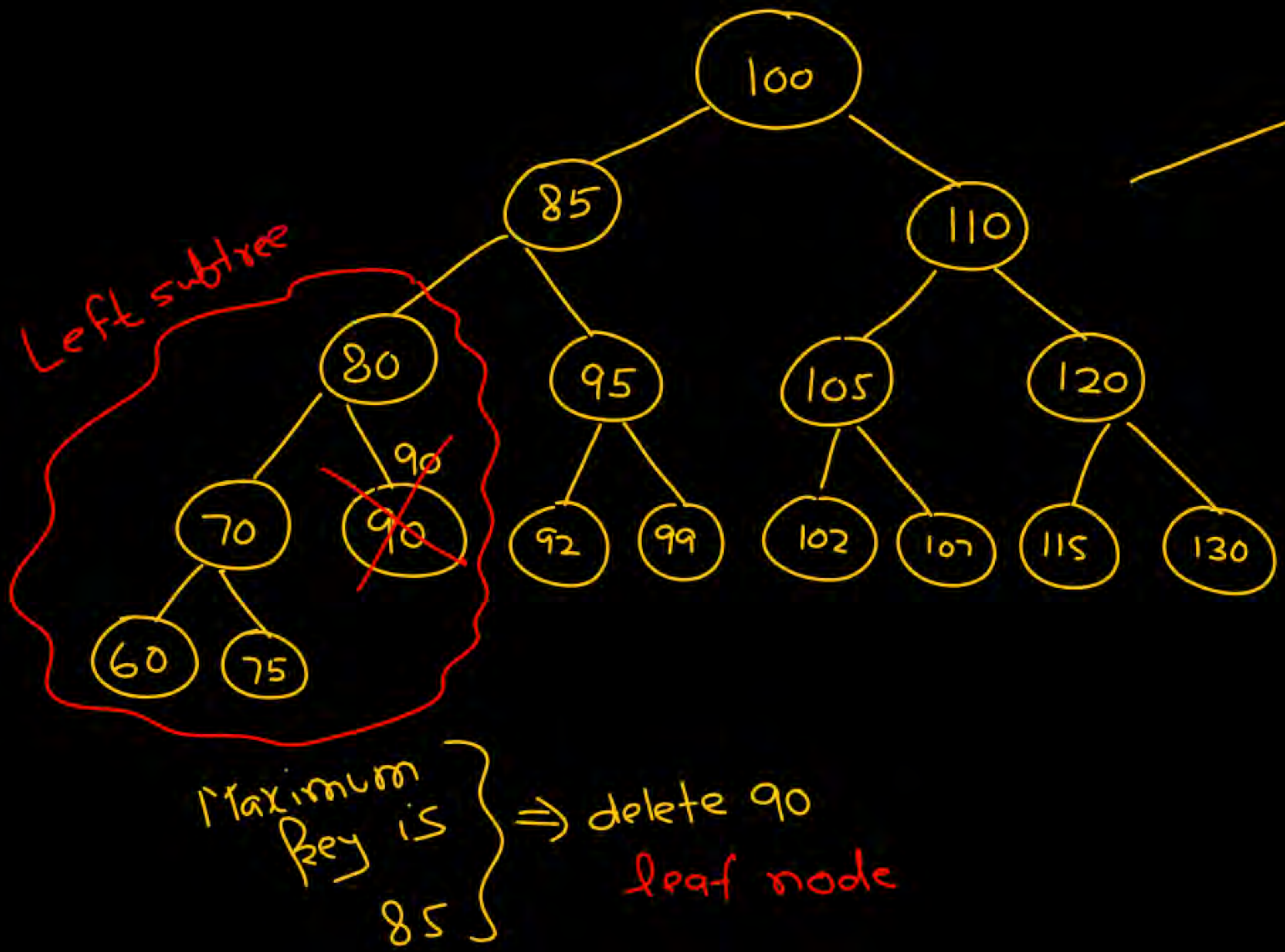


1st way: Replace the node(key) to be deleted with the maximum key in the left-subtree of node & then perform deletion

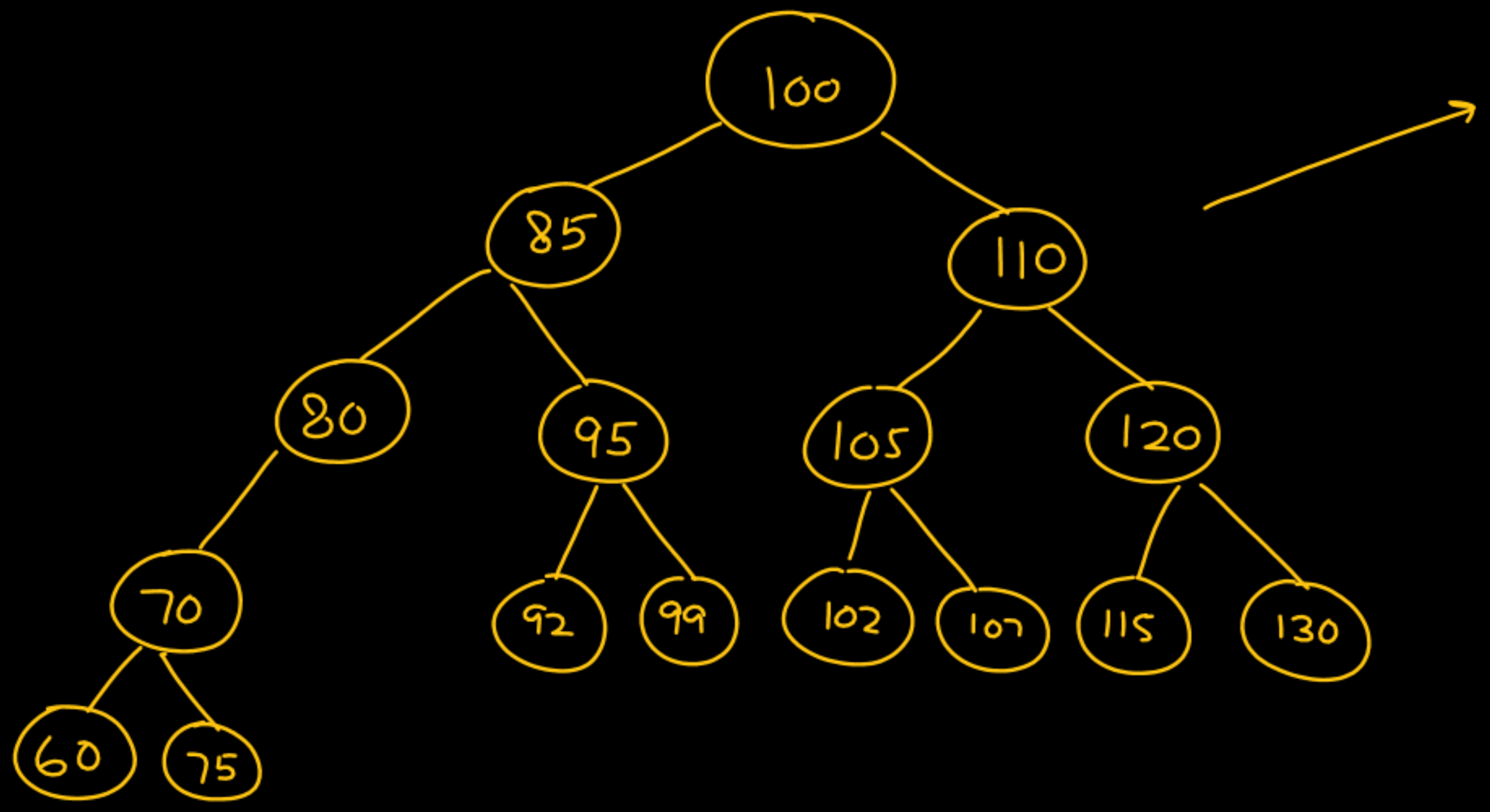


1st way: Replace the node(key) to be deleted with the maximum key in the left-subtree of node & then perform deletion

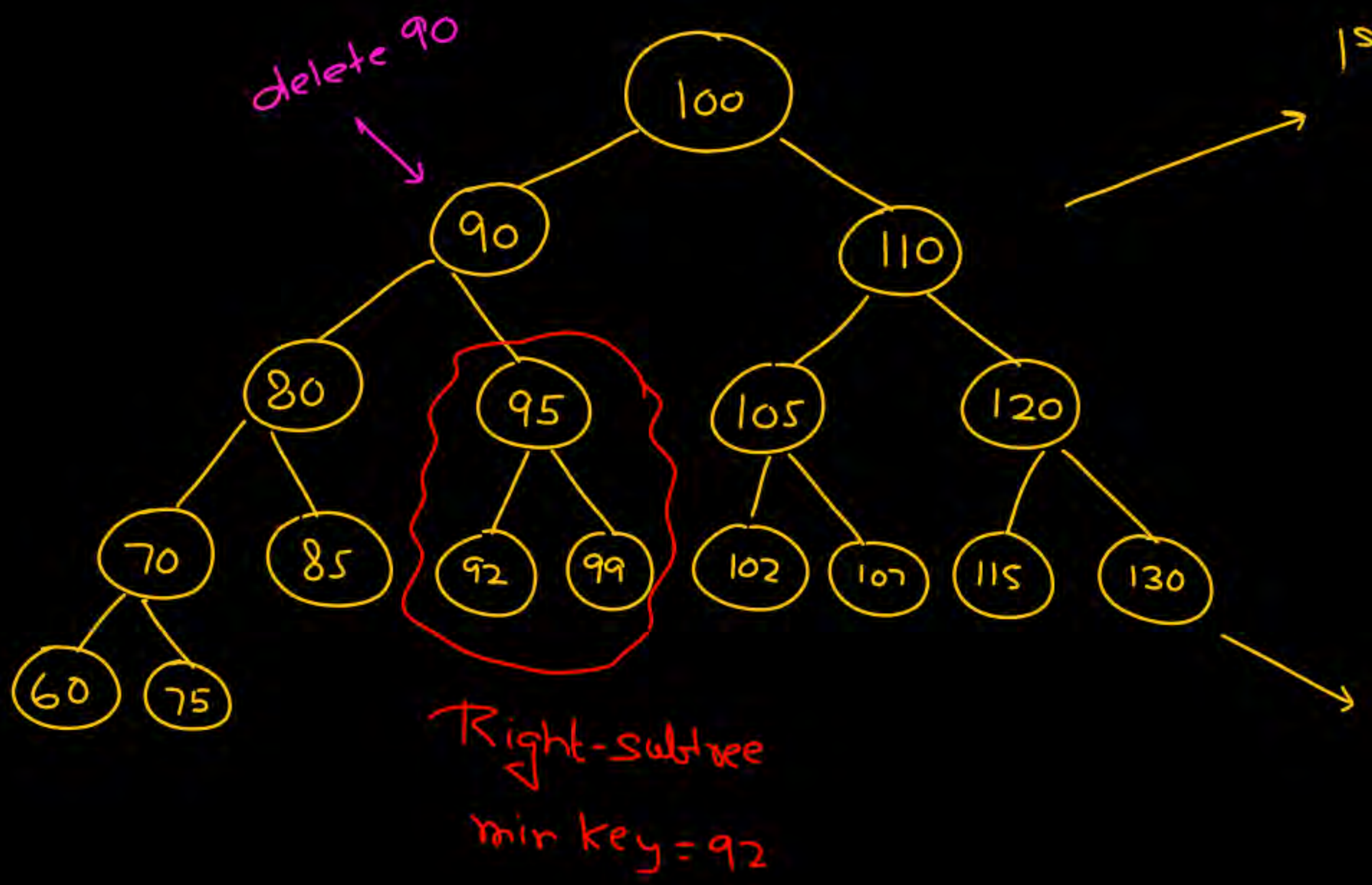
$\left.\begin{array}{l}\text{Maximum key is}\\ \text{85}\end{array}\right\} \Rightarrow$ delete 90



1st way: Replace the node(key) to be deleted with the maximum key in the left-subtree of node & then perform deletion

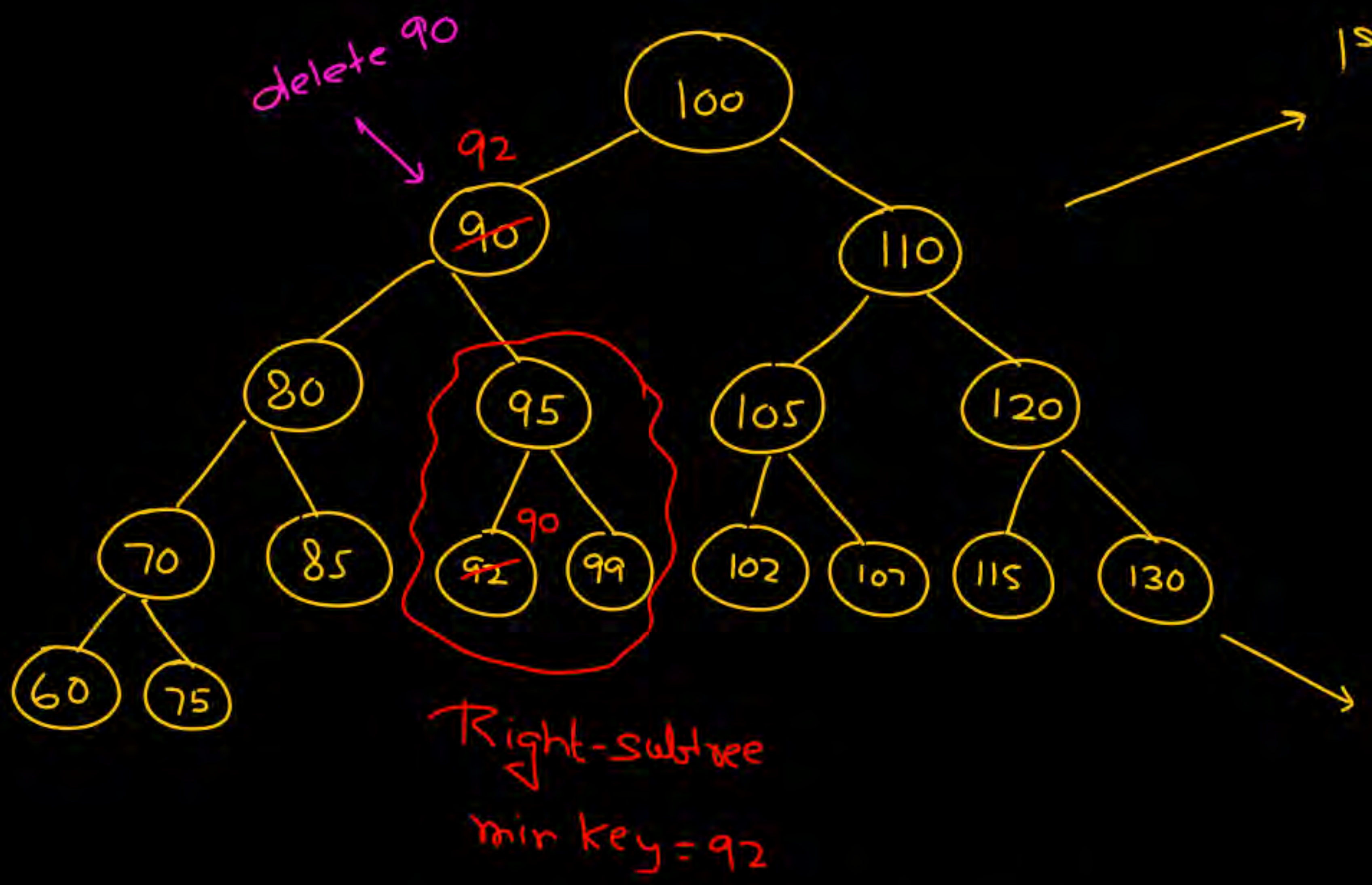


1st way : Replace the node(key) to be deleted with the maximum key in the left - subtree of node & then perform deletion



1st way: Replace the node(key) to be deleted with the maximum key in the left-subtree of node & then perform deletion

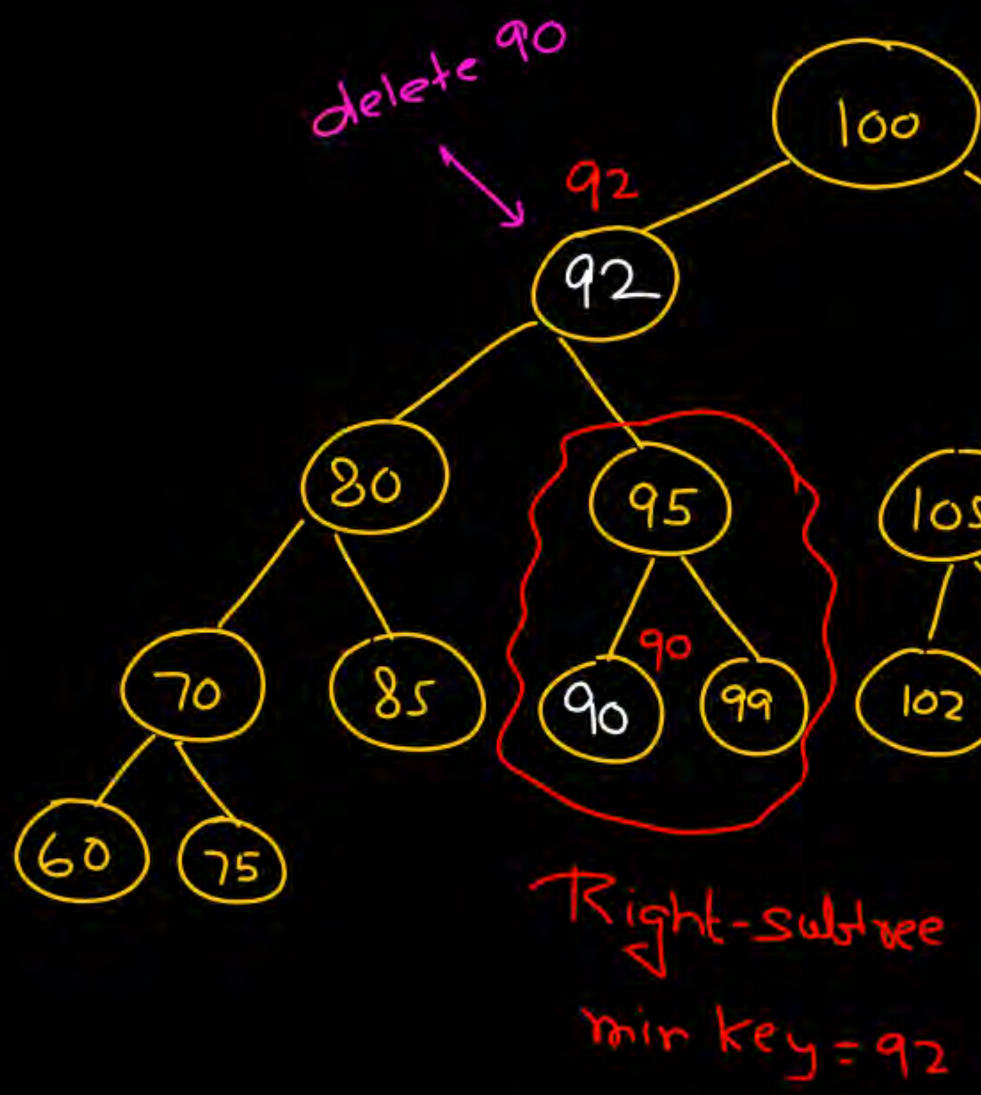
2nd way: Replace the node(key) to be deleted with the minimum key in the right sub-tree of the node & then perform deletion



1st way: Replace the node(key) to be deleted with the maximum key in the left-subtree of node & then perform deletion

2nd way: Replace the node(key) to be deleted with the minimum key in the right sub-tree of the node & then perform deletion

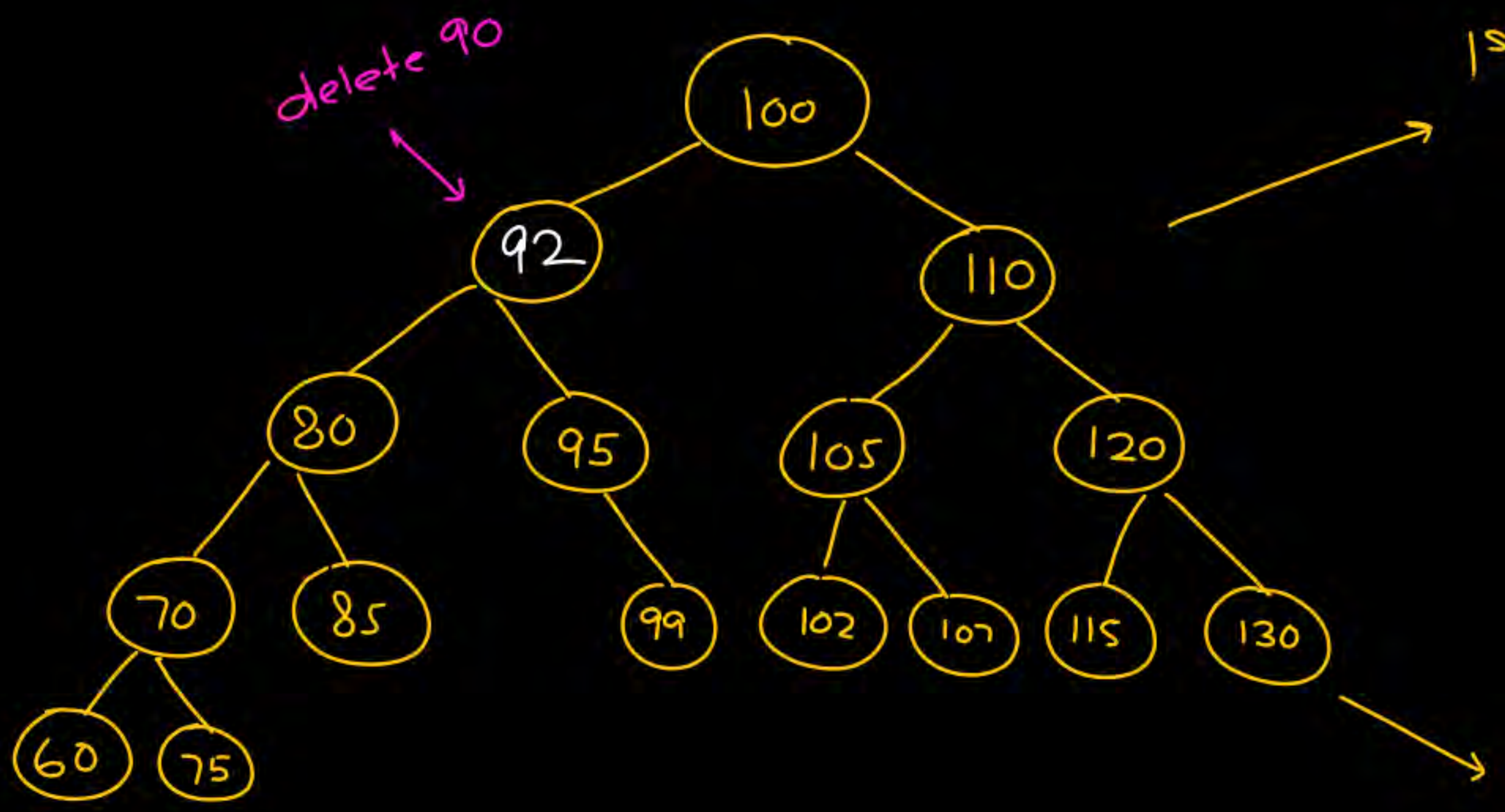
Now ⇒ u need to delete a leaf node



Now ⇒ u need to delete a leaf node

1st way : Replace the node(key) to be deleted with the maximum key in the left - subtree of node & then perform deletion

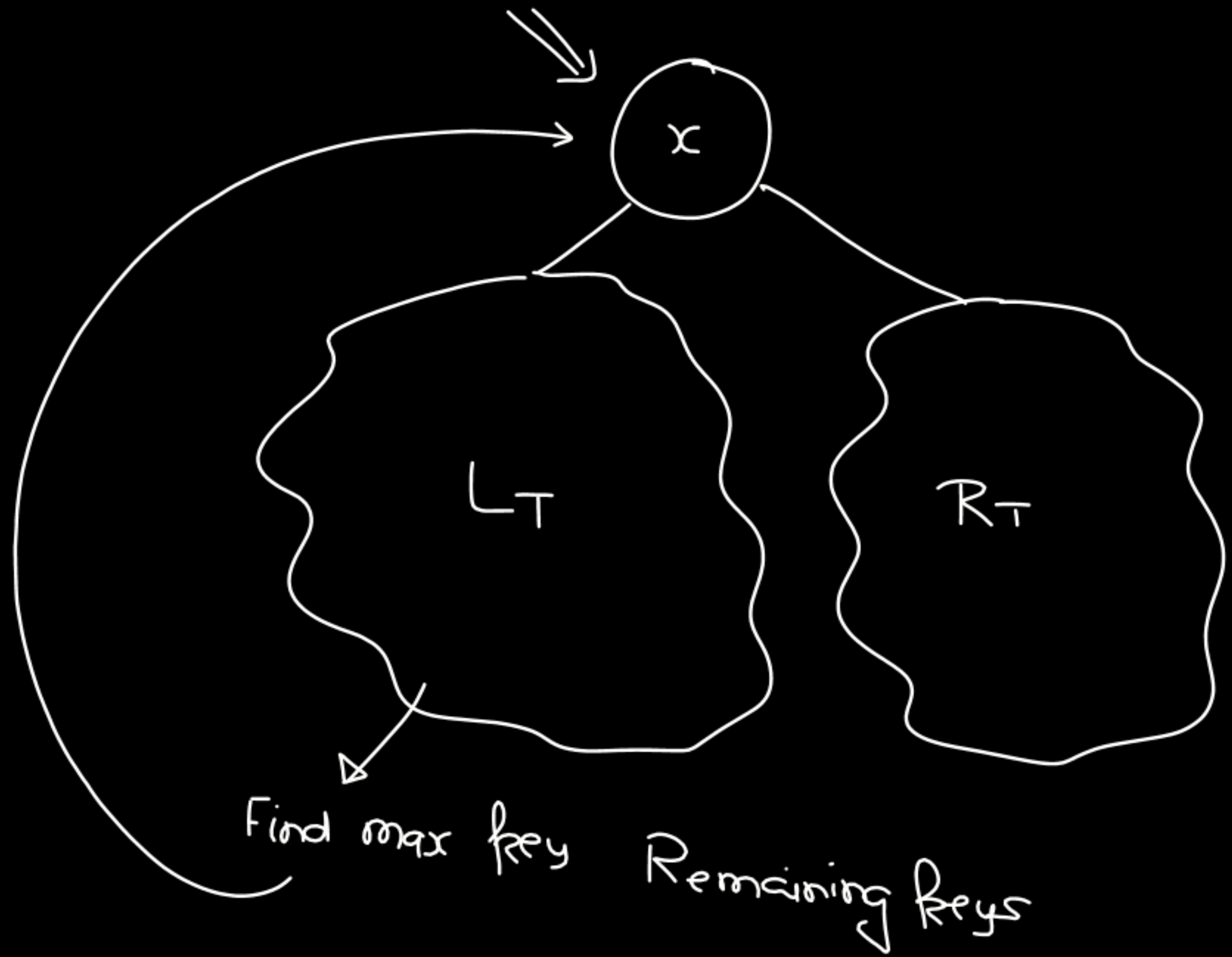
2nd way : Replace the node(key) to be deleted with the minimum key in the right sub-tree of the node & then perform deletion

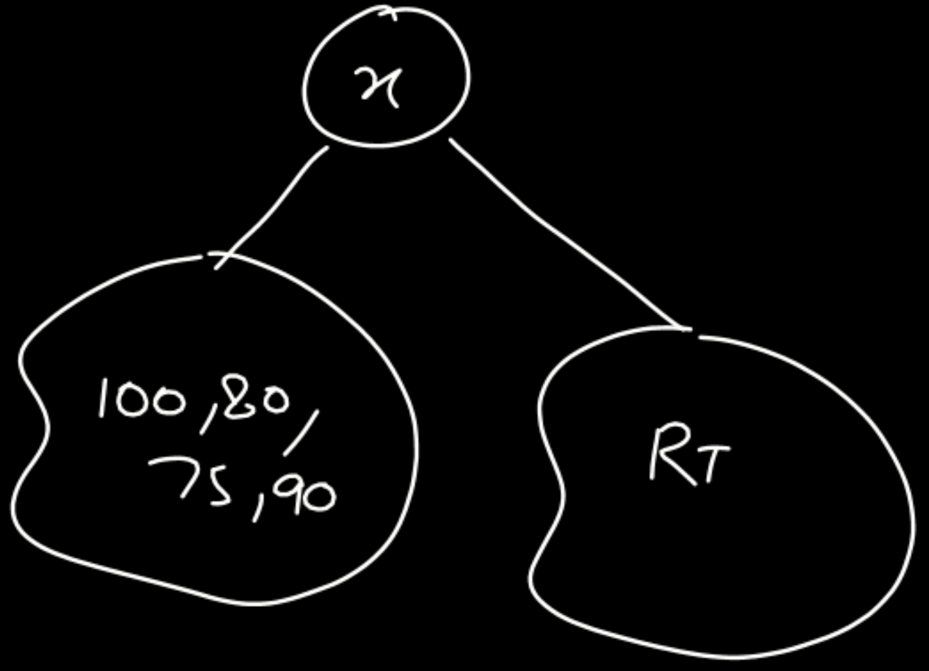


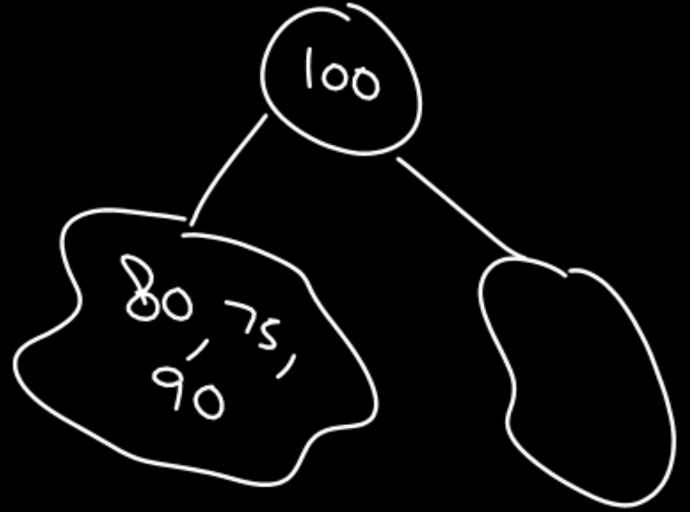
1st way: Replace the node(key) to be deleted with the maximum key in the left-subtree of node & then perform deletion

2nd way: Replace the node(key) to be deleted with the minimum key in the right sub-tree of the node & then perform deletion

1st way :



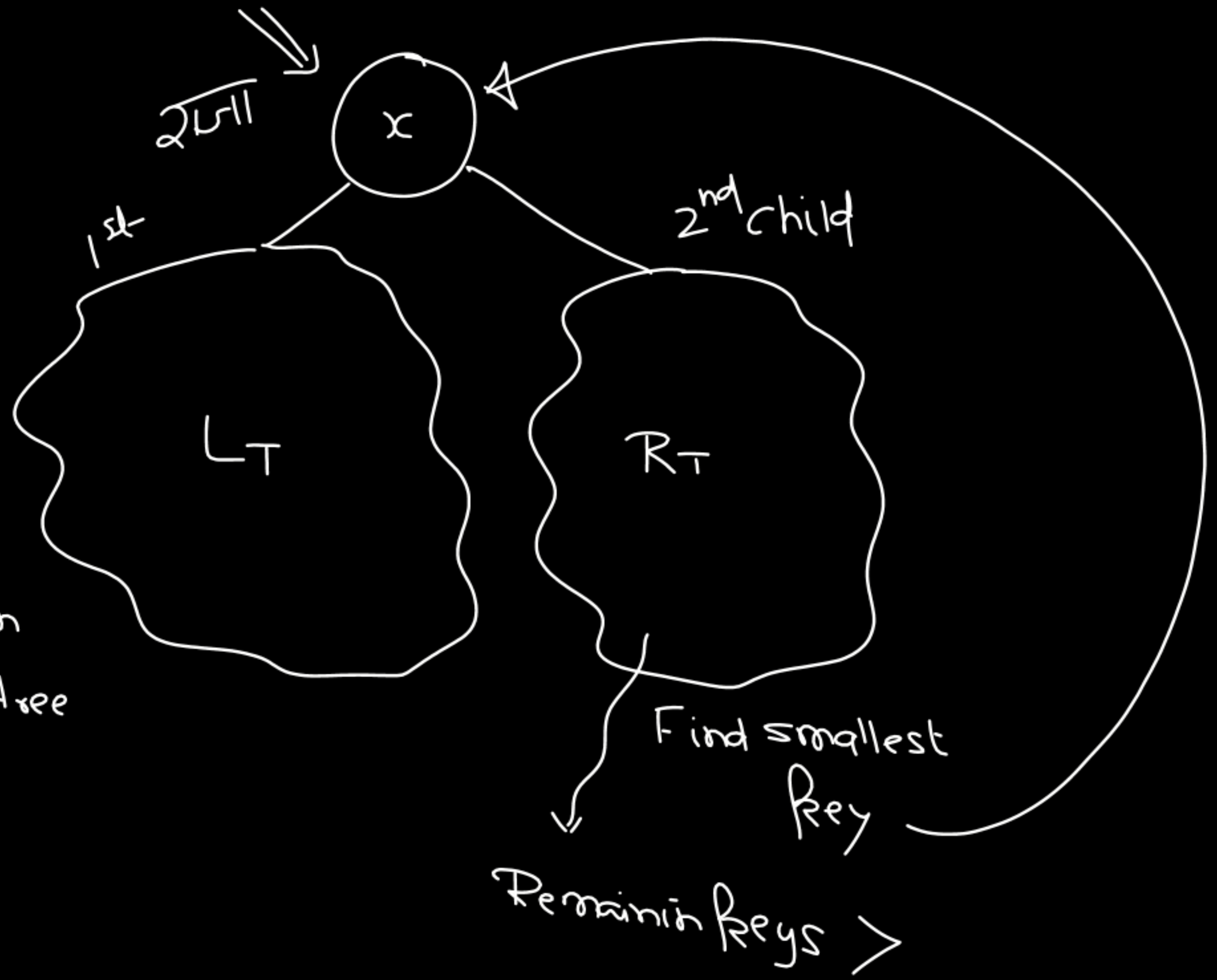


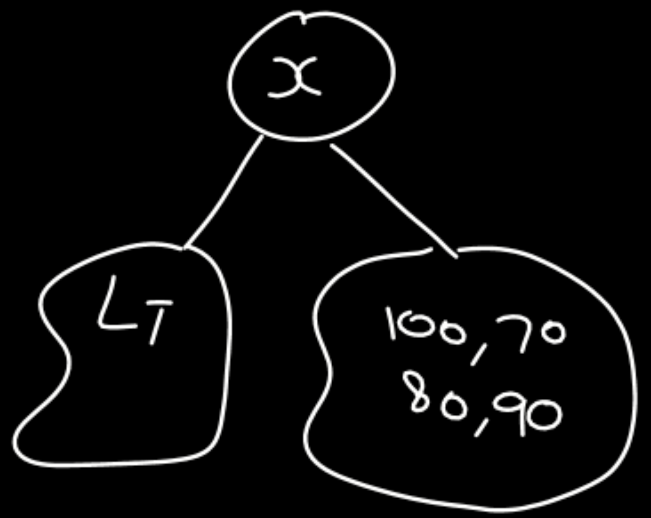


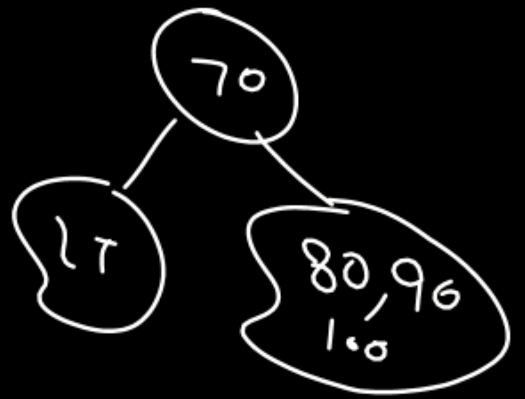
2nd way

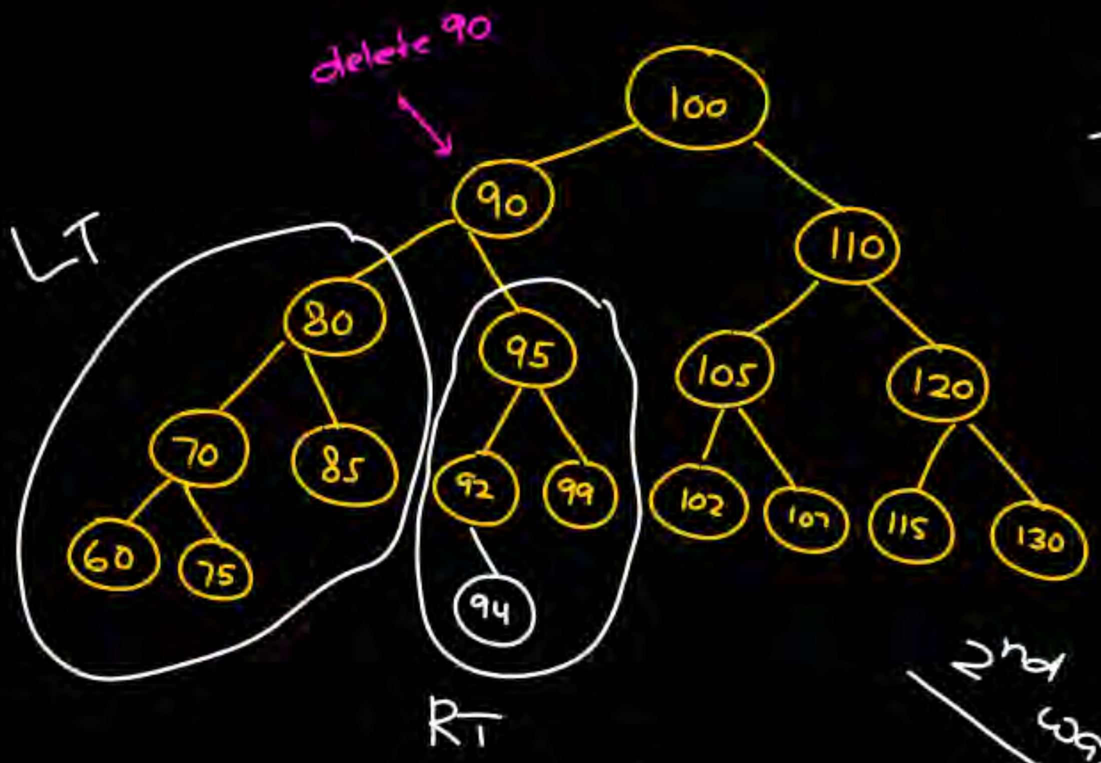
L_T, x, R_T

All keys of left-tree, ~~x~~, All keys in right, subtree

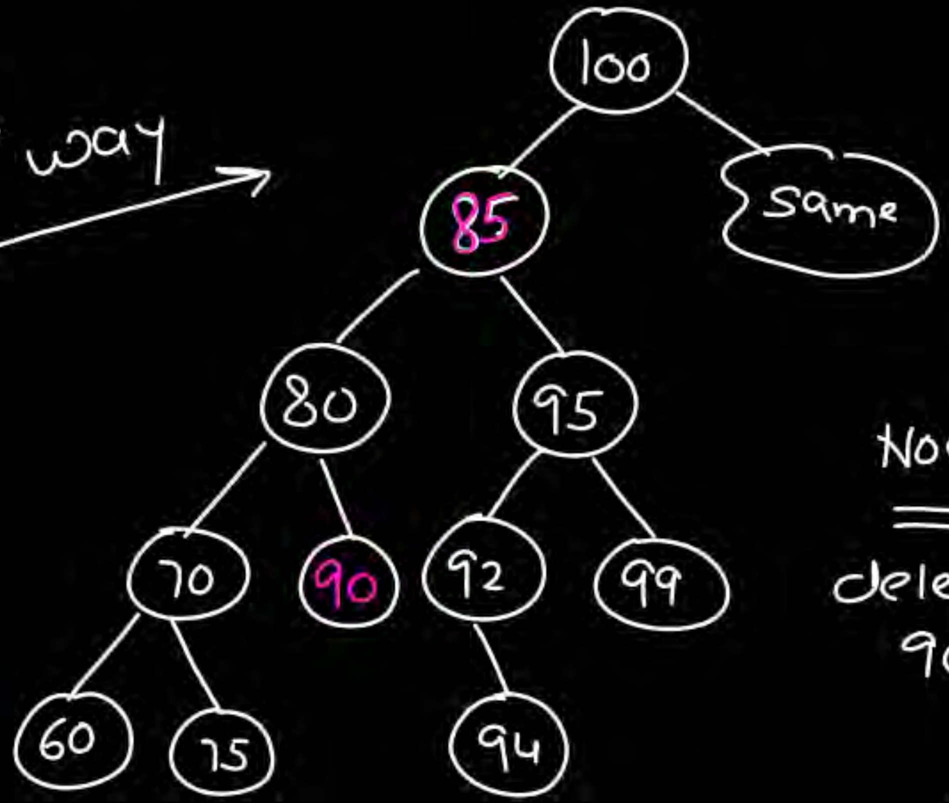




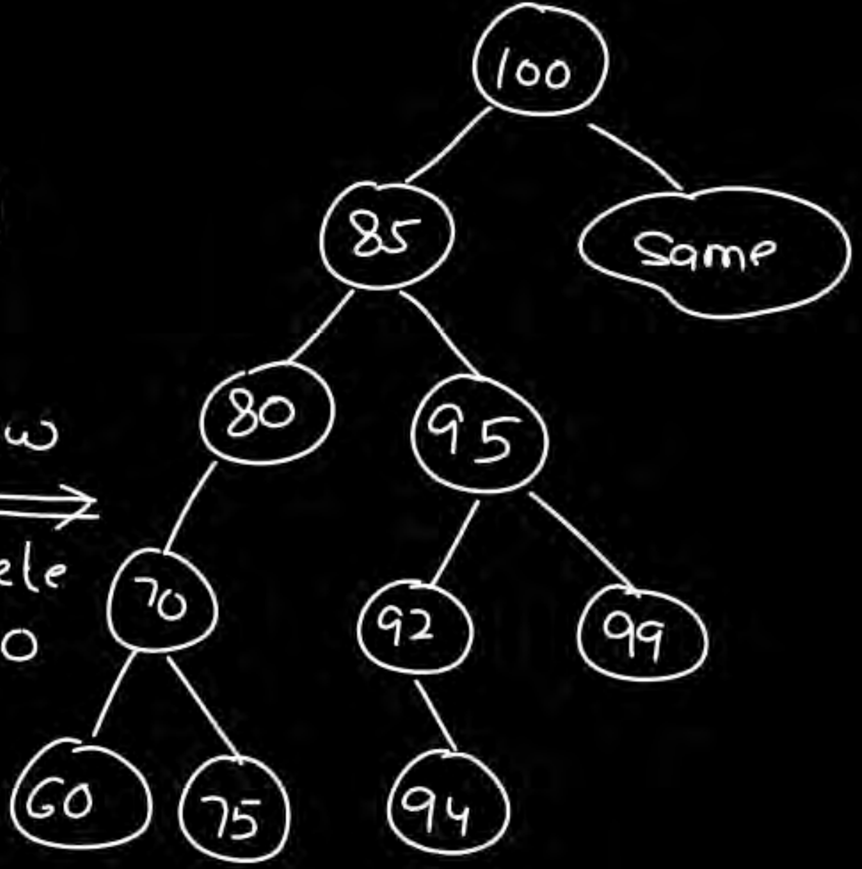




1st way $\longrightarrow$



Now $\Longrightarrow$ delete 90

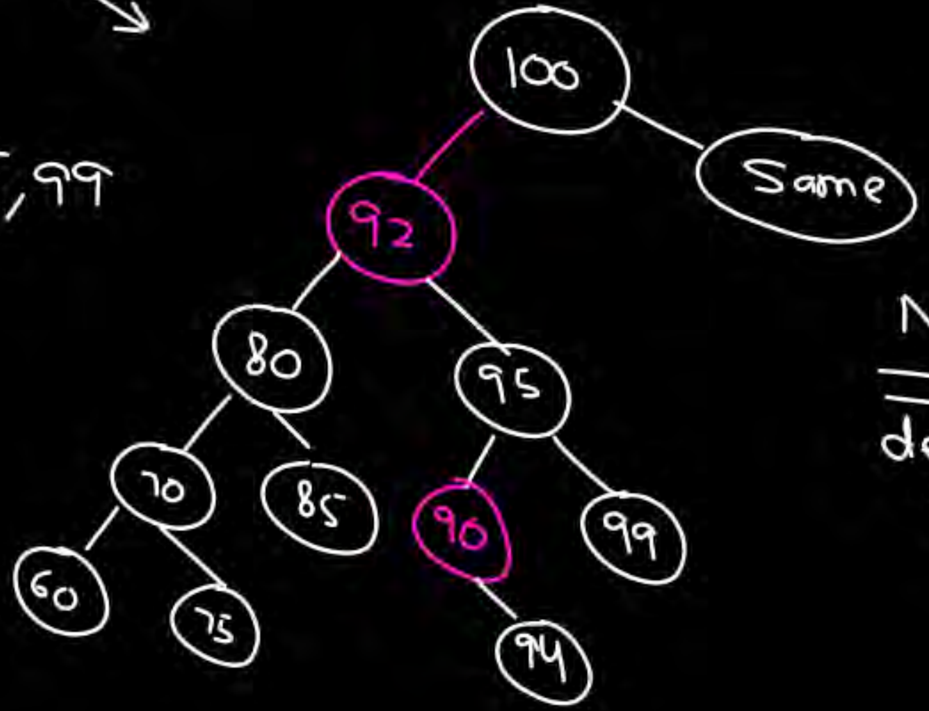


2nd way $\searrow$

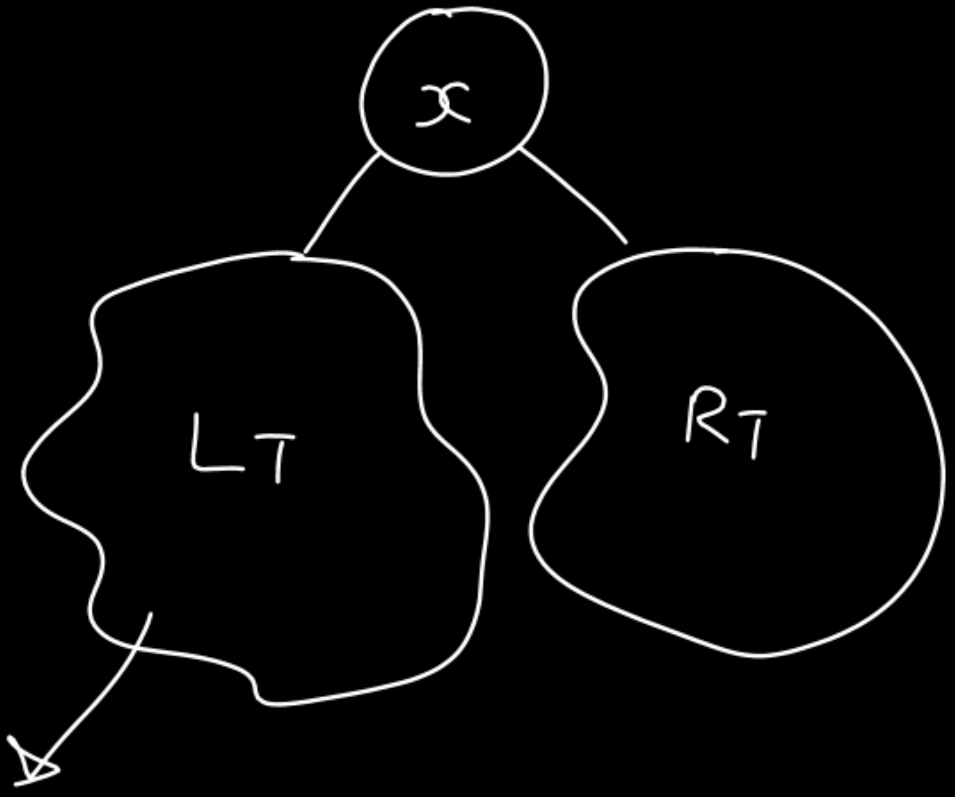
60, 70, 75, 80, 85, $\boxed{90}$, 92, 94, 95, 99

Inorder predecessor

Inorder successor



Now $\Longrightarrow$ delete 90 deletion of node with 1-child

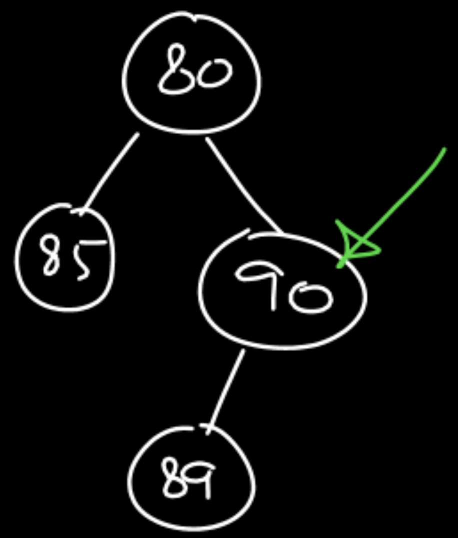


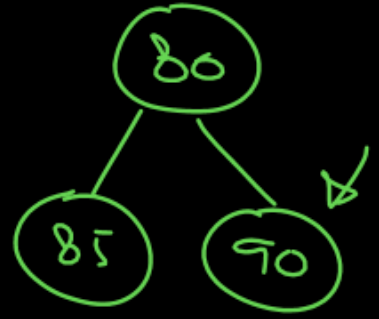
node with largest key

$\Rightarrow$ can have 0 or 1 child

smallest Key

can not have a left child





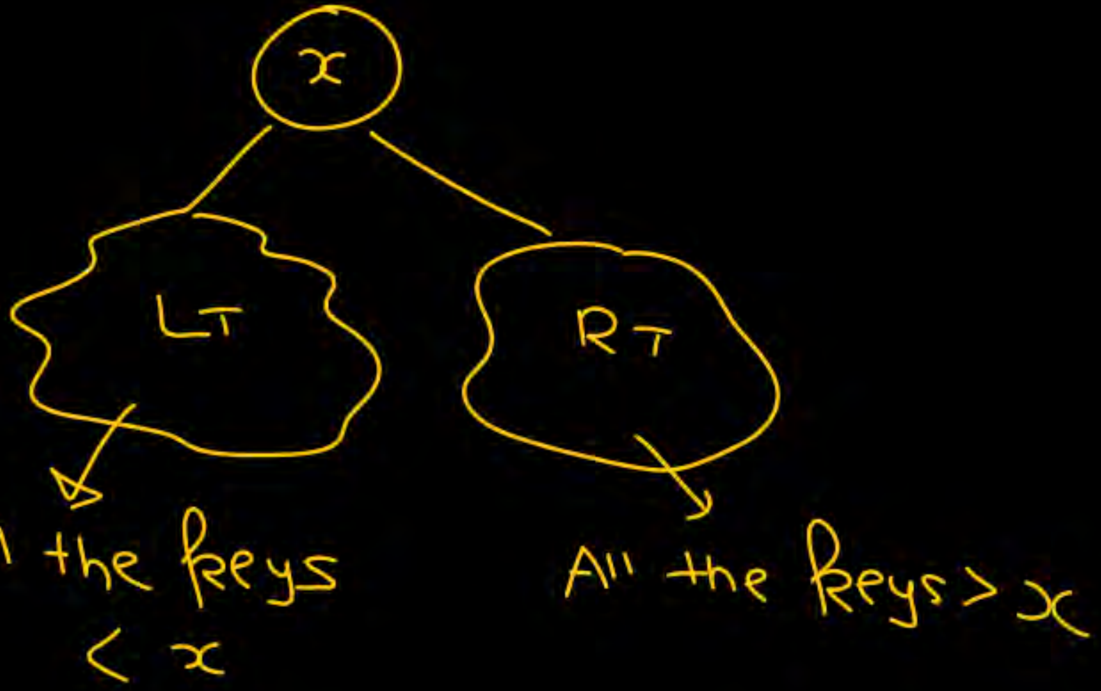
can not have Right child X

AVL - Tree

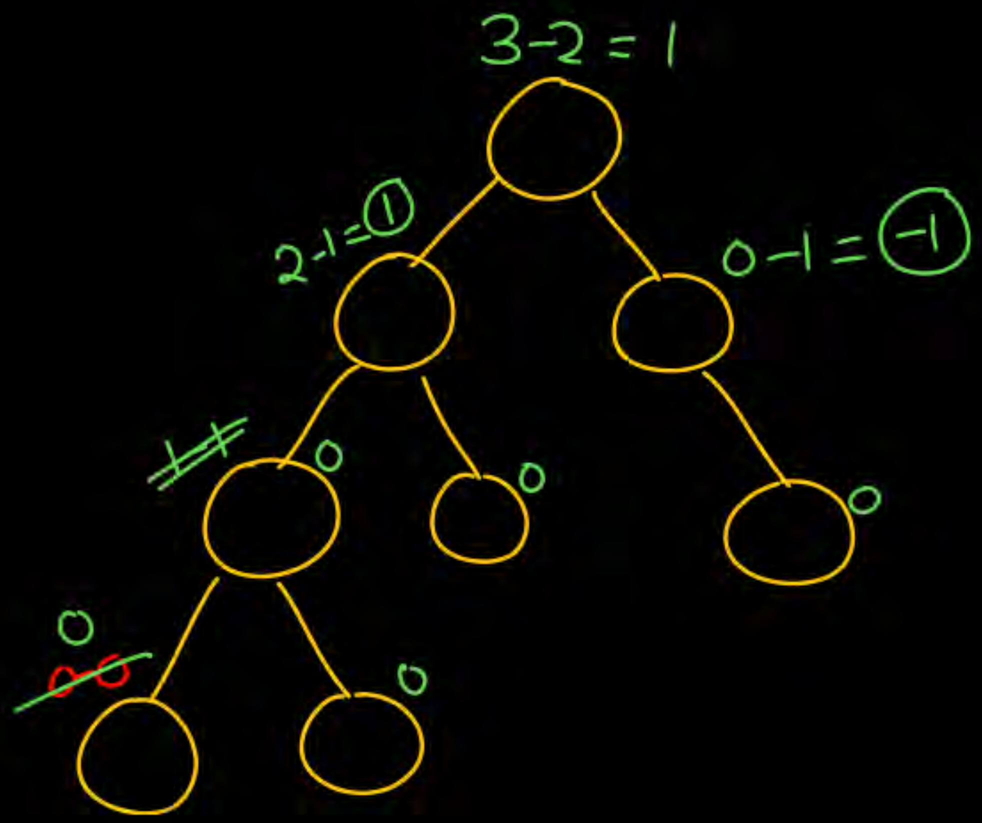
Every node satisfies 2 property

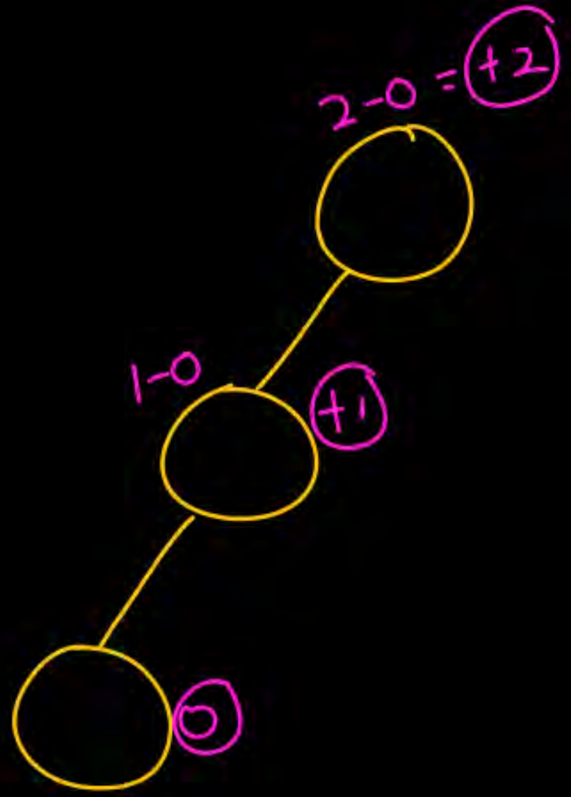
Height Balanced

① BST property :

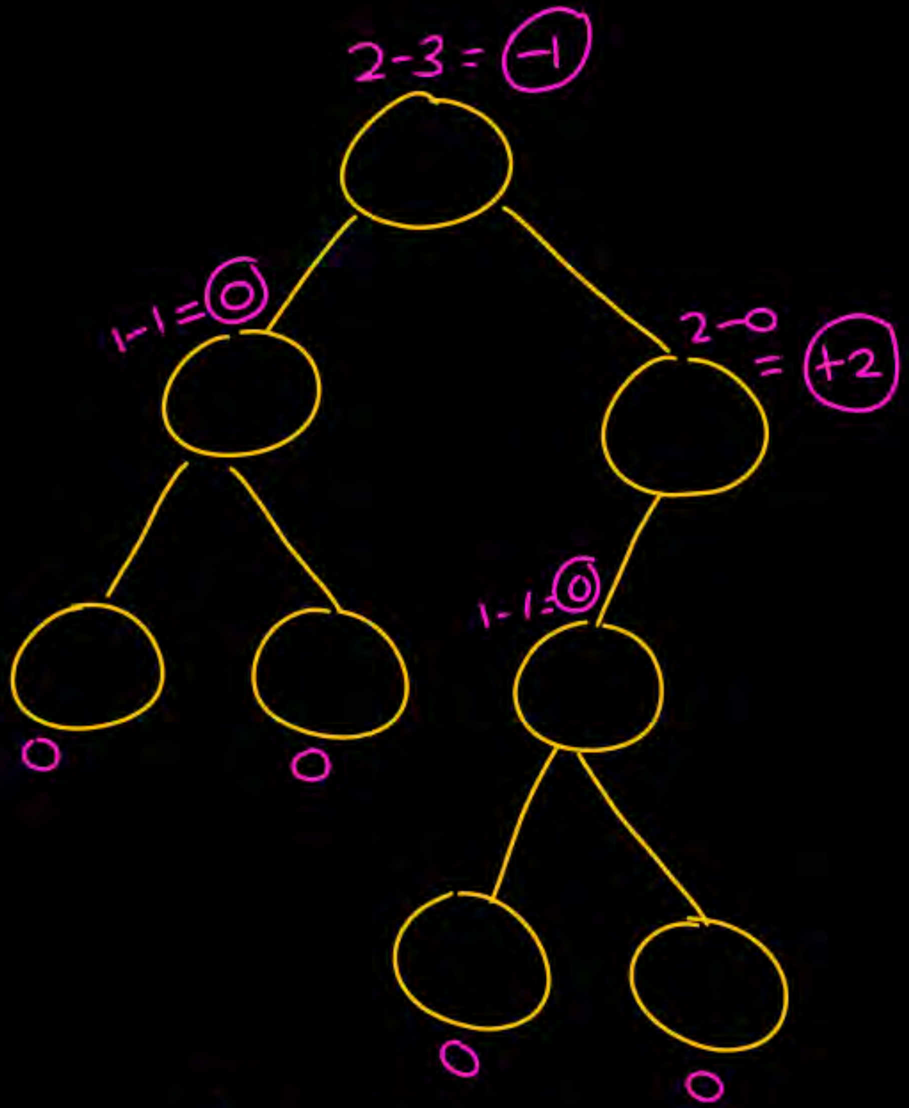


② AVL tree property : The balancing factor of a node can be either 0, -1 or +1.

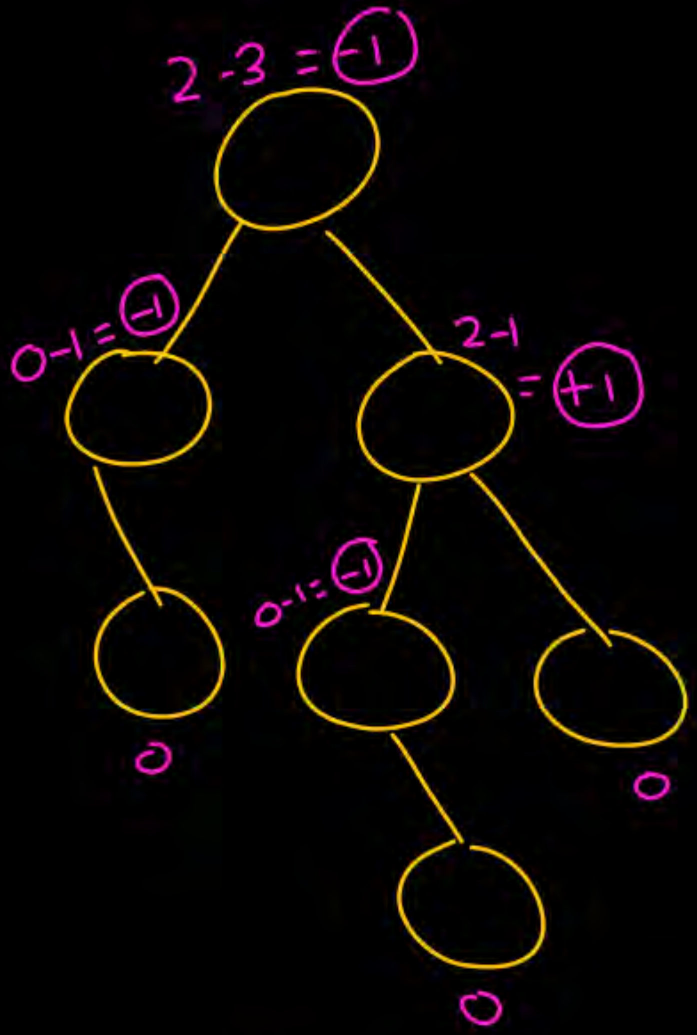

$3-2=1$
$2-1=\boxed{1}$
$0-1=\boxed{-1}$
~~1~~
0
0
0
0
~~00~~
0



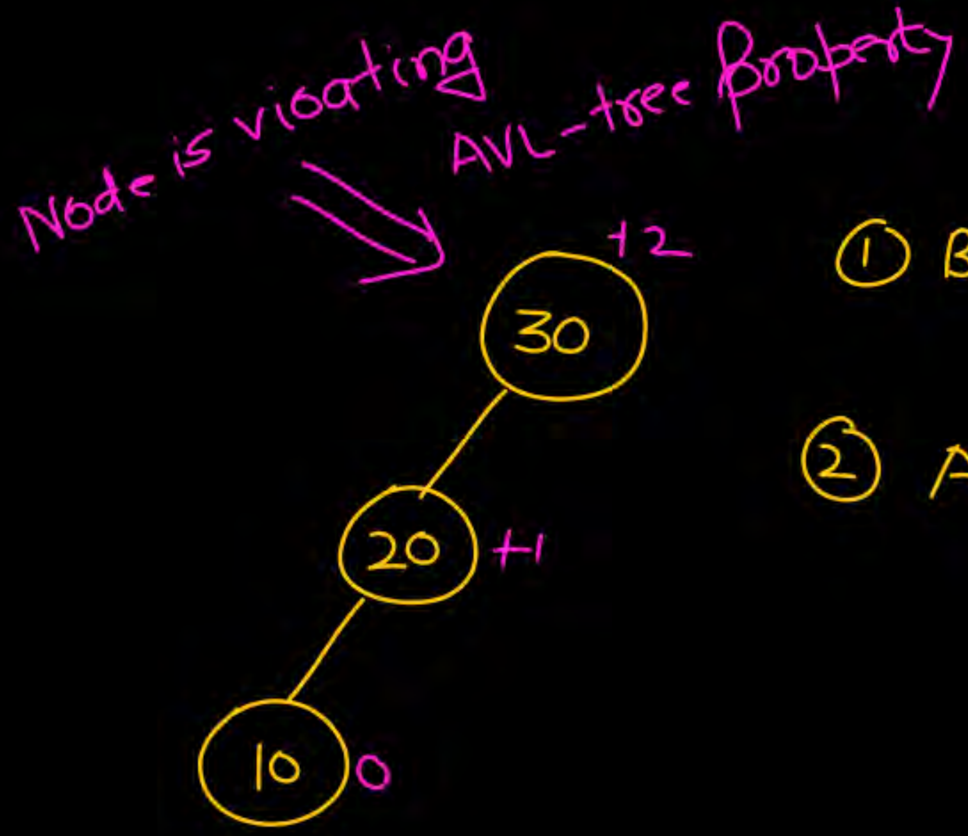
AVL tree
property ✗



AVL tree
property ✗



AVL tree
property ✓



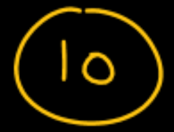
① BST property ✓

② AVL - tree Property

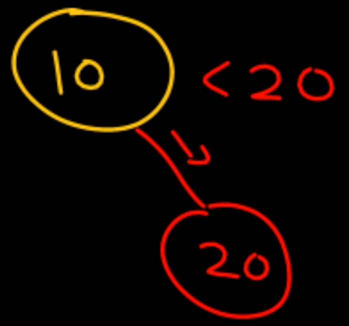
↳ The balance factor of a node can be 0, +1, or -1

Const. an AVL tree by inserting keys 10, 20, 30 in same order.

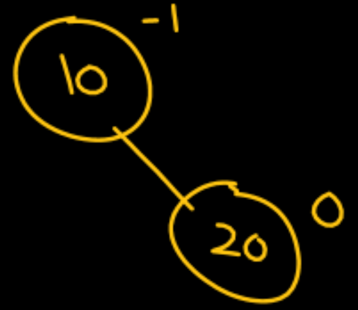
(i) Insert 10



(ii) Insert 20

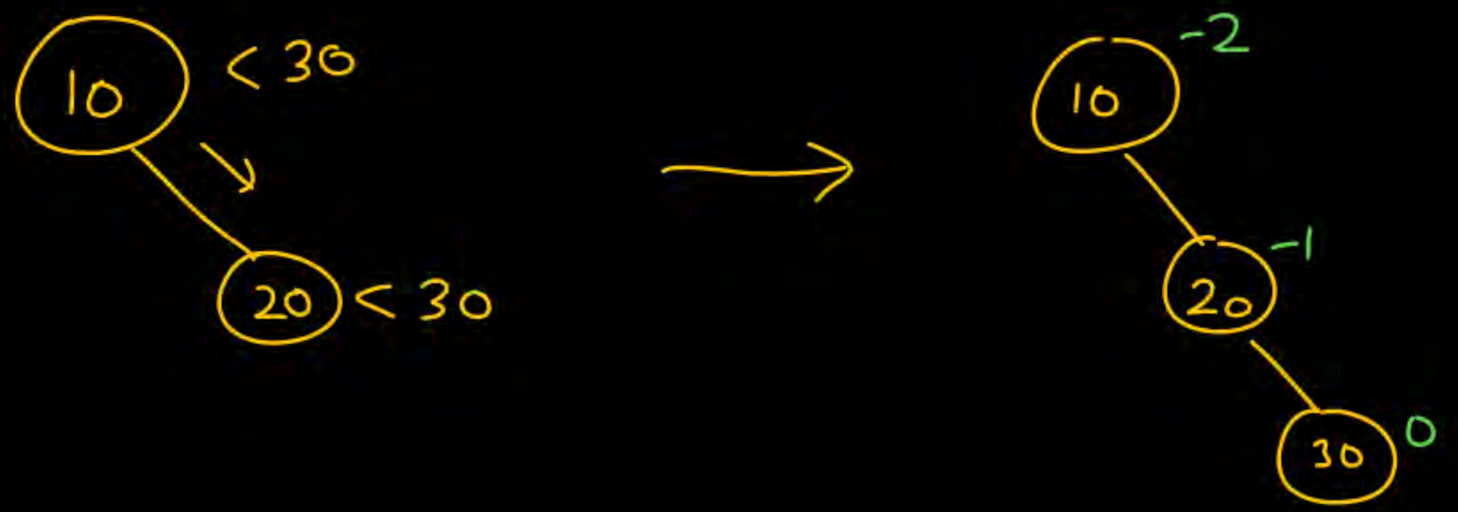


Insert a key ⟹ same as BST

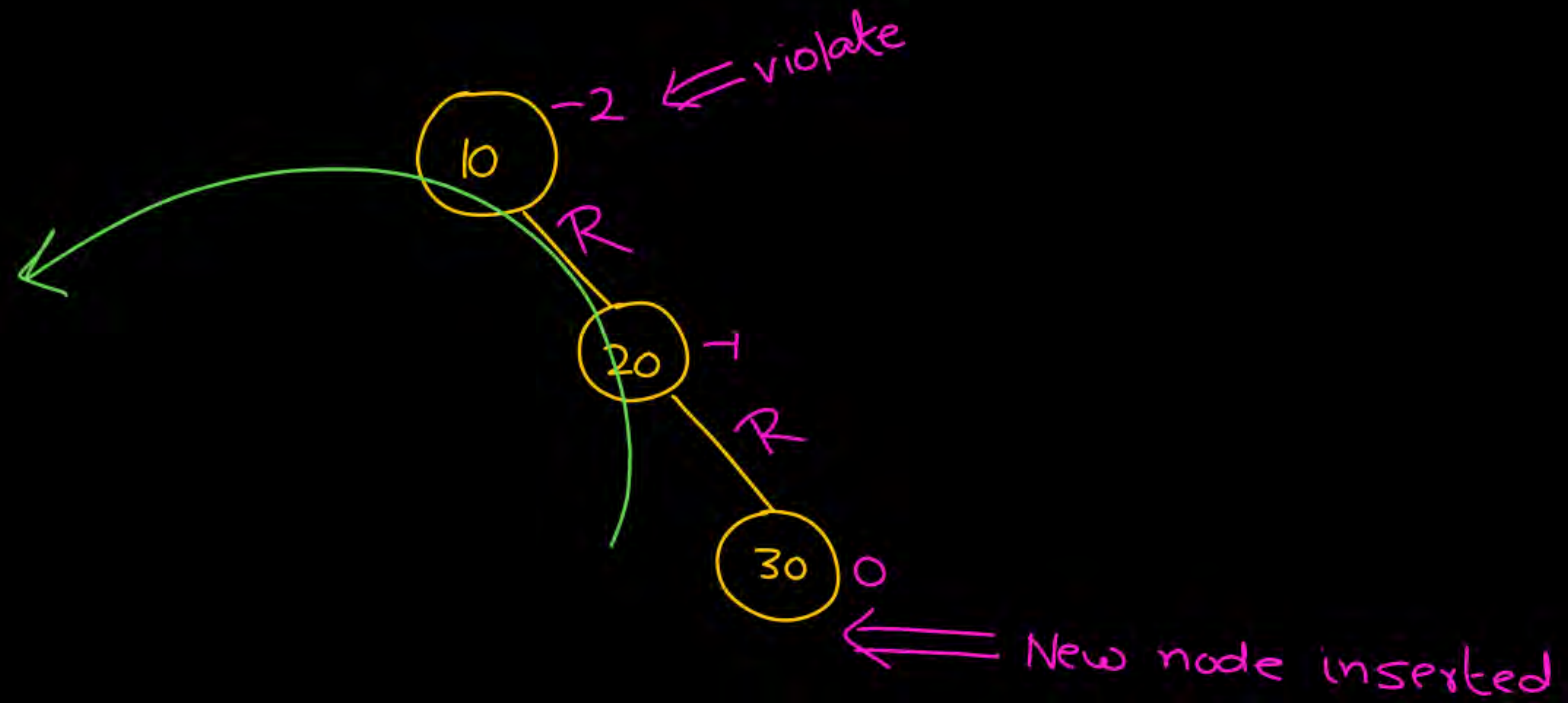


AVL tree

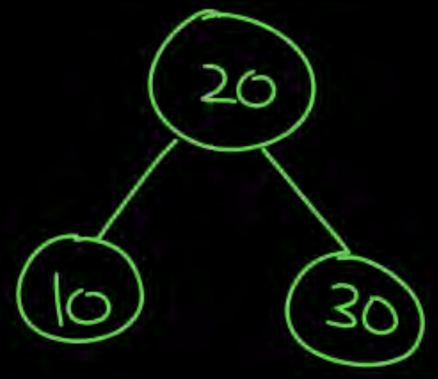
(iii) Insert 30

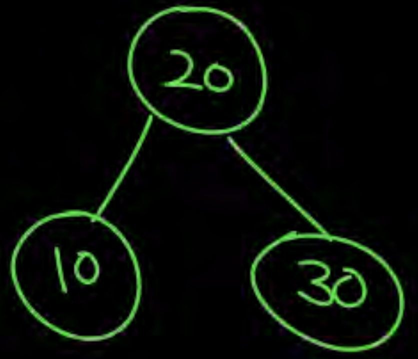


Insertion of key may cause the bal. factor of some node other than 0, -1, +1 (unbalanced) $\Rightarrow$ To balance the tree $\Downarrow$ rotations are required.

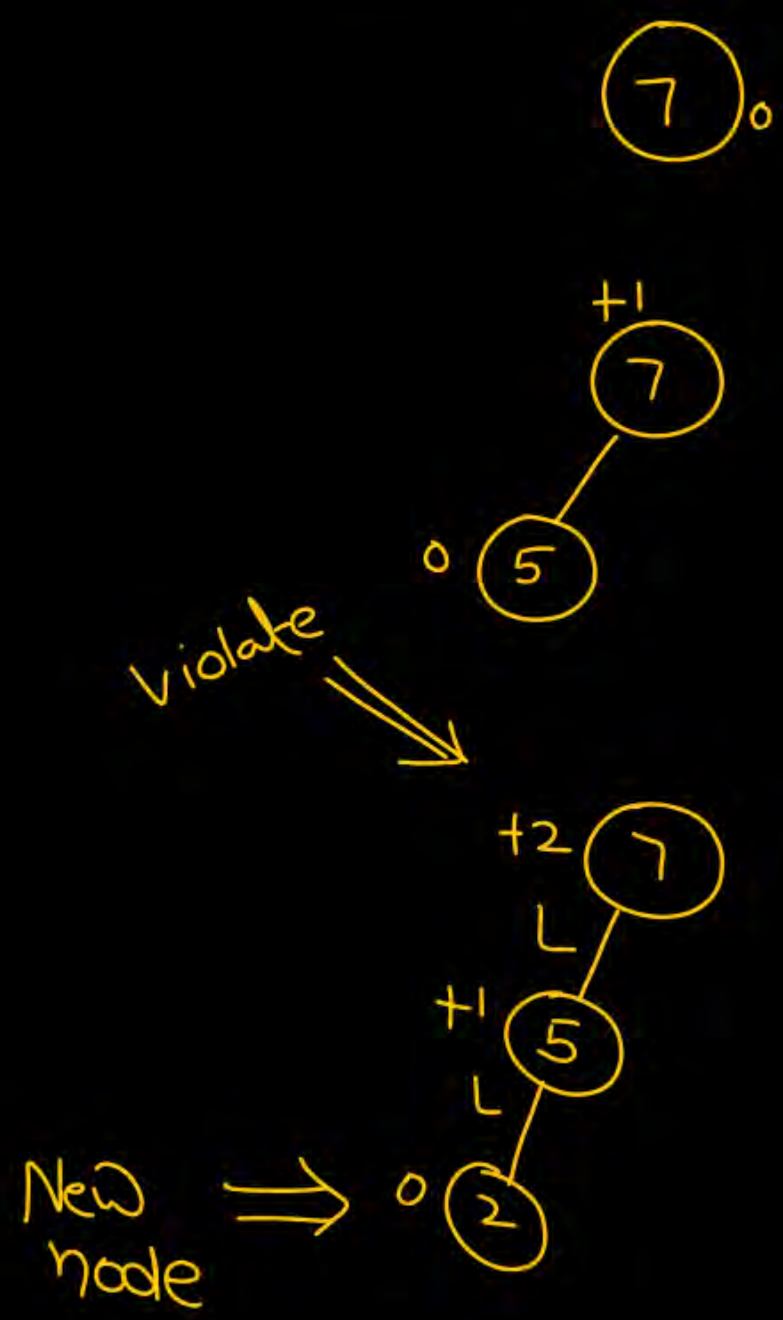


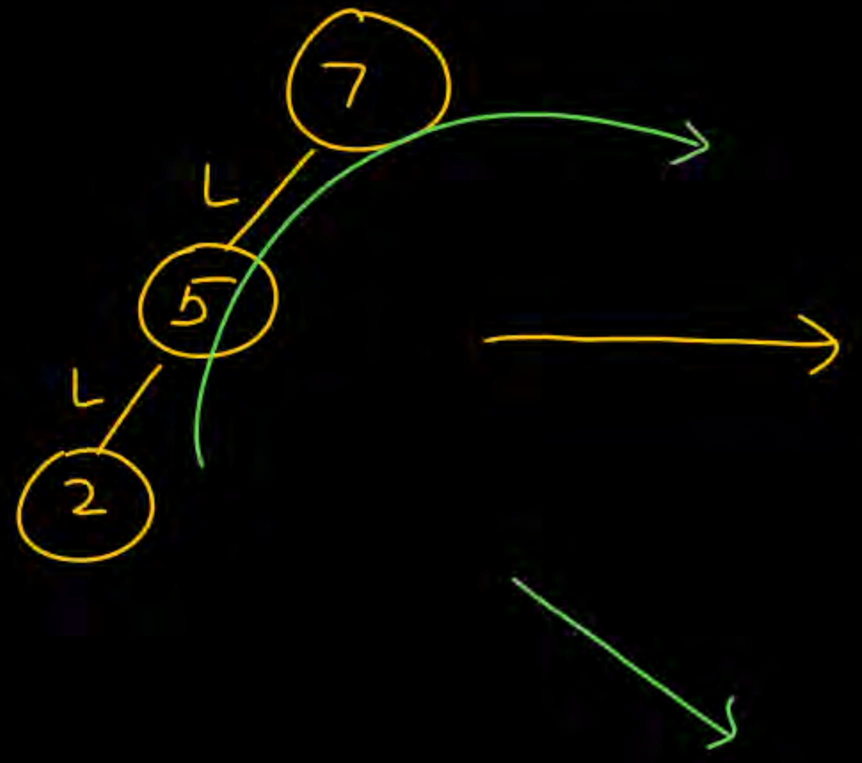
10, 20, 30

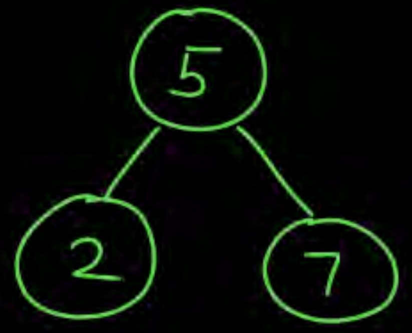




Insert 7, 5, 2



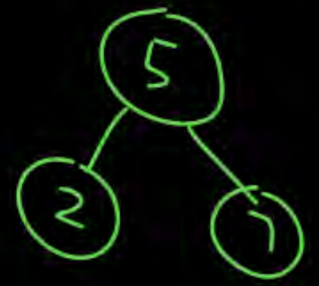




arrange keys in asc. order

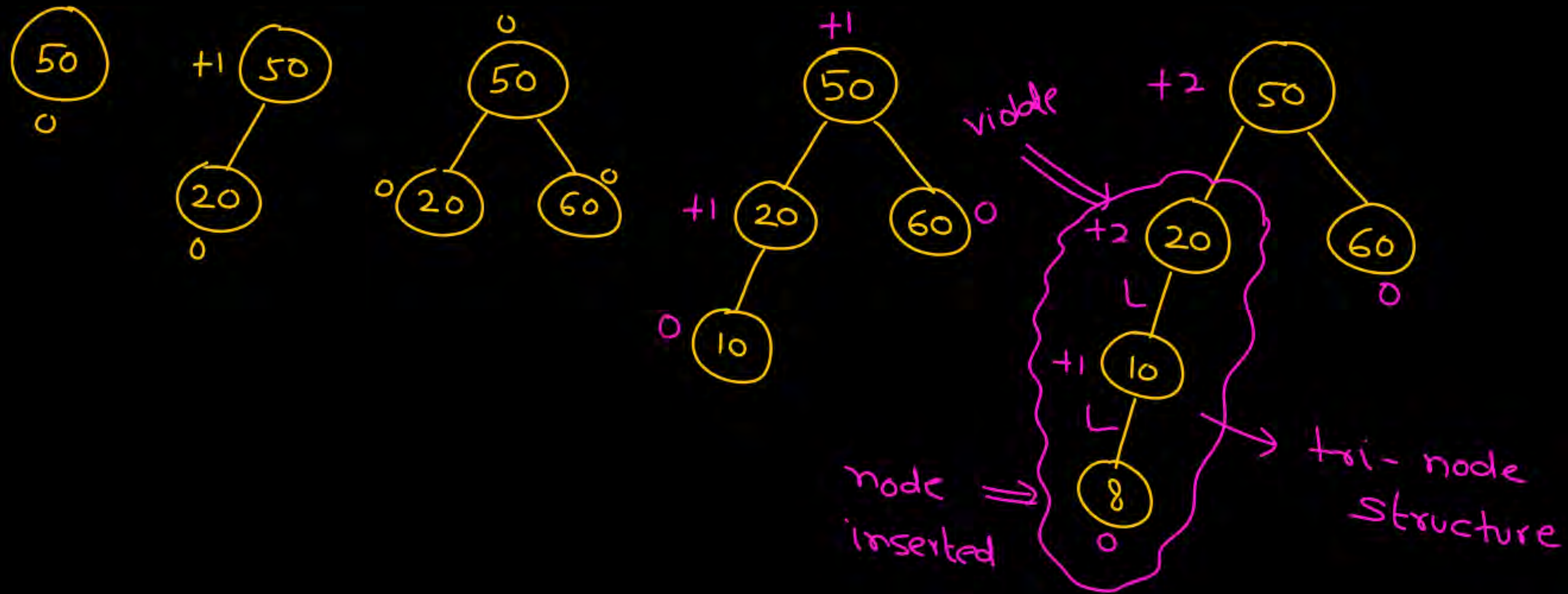
2, 5, 7

⇓



Const. AVL tree by inserting keys in given order.

50, 20, 60, 10, 8, 15, 32, 46, 11, 38

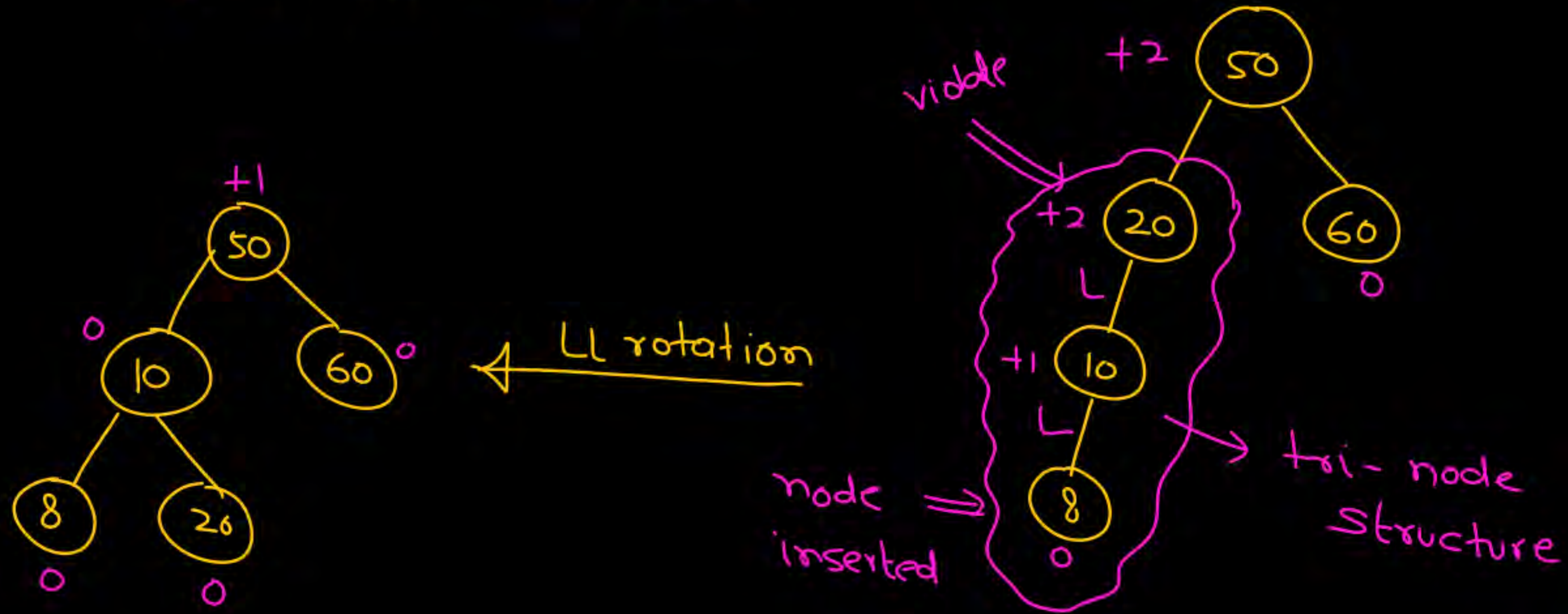


Const. AVL tree by inserting keys in given order.

50, 20, 60, 10, 8, 15, 32, 46, 11, 38

$\longrightarrow$

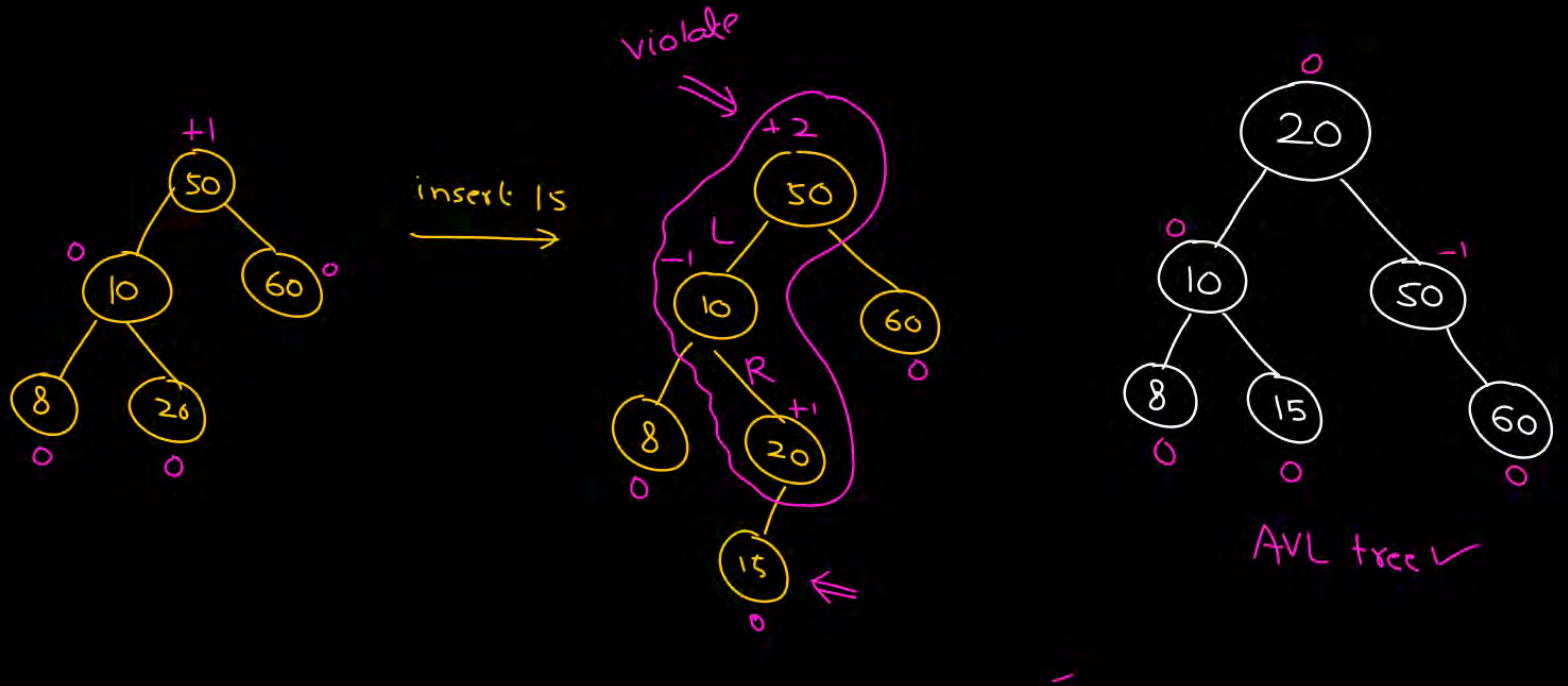
8 10 20 asc. order



Const AVL tree by inserting keys in given order.

50, 20, 60, 10, 8, 15, 32, 46, 11, 38

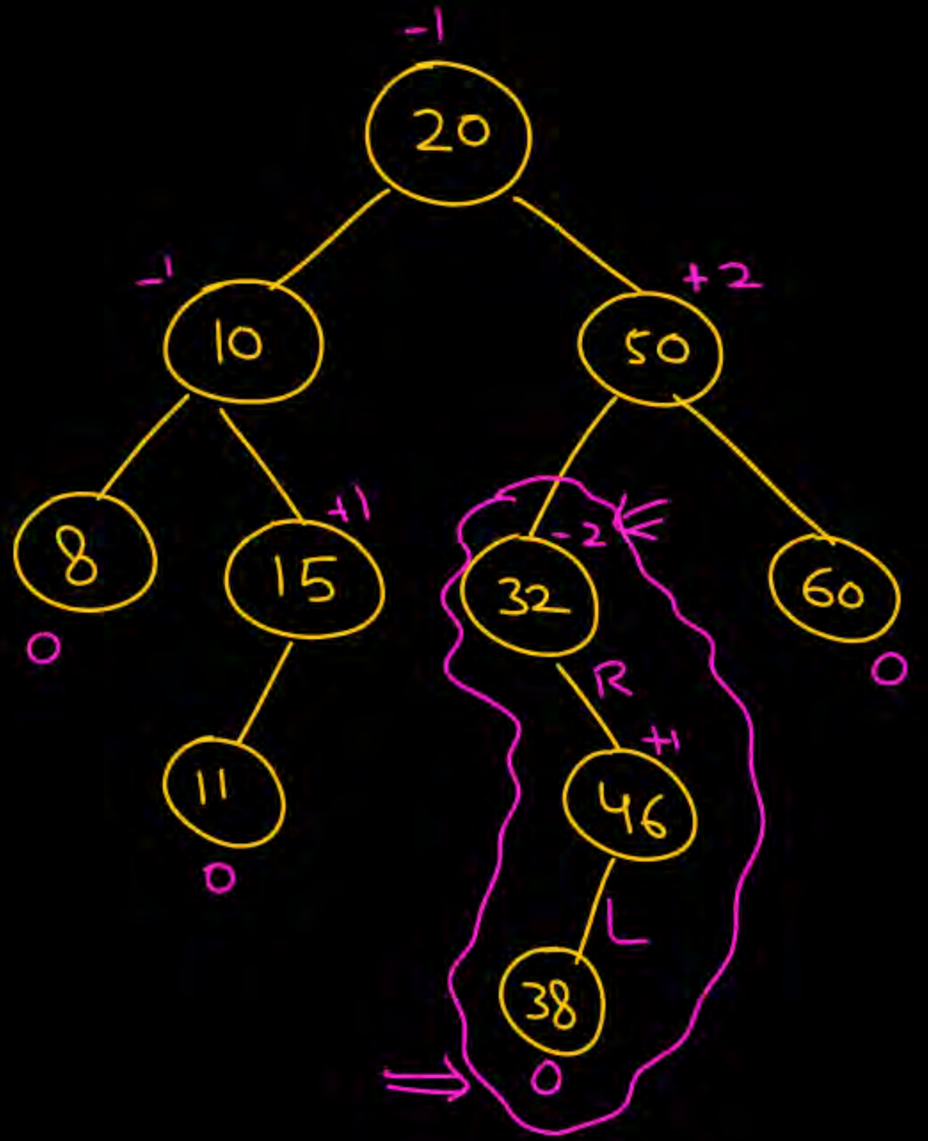
10, 20, 50

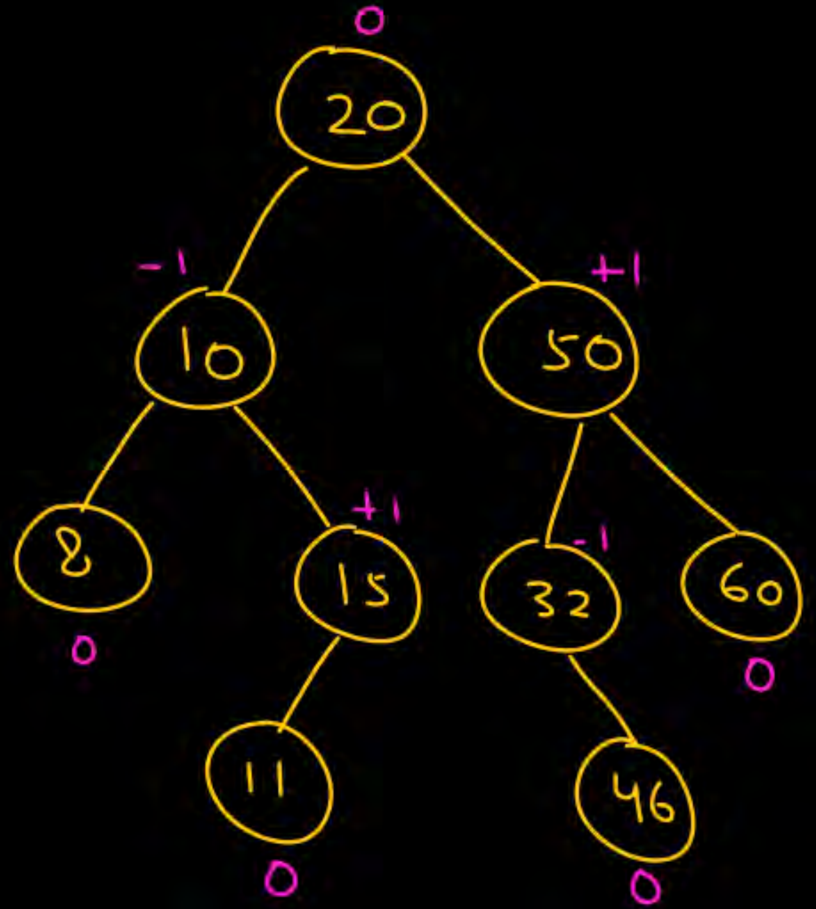


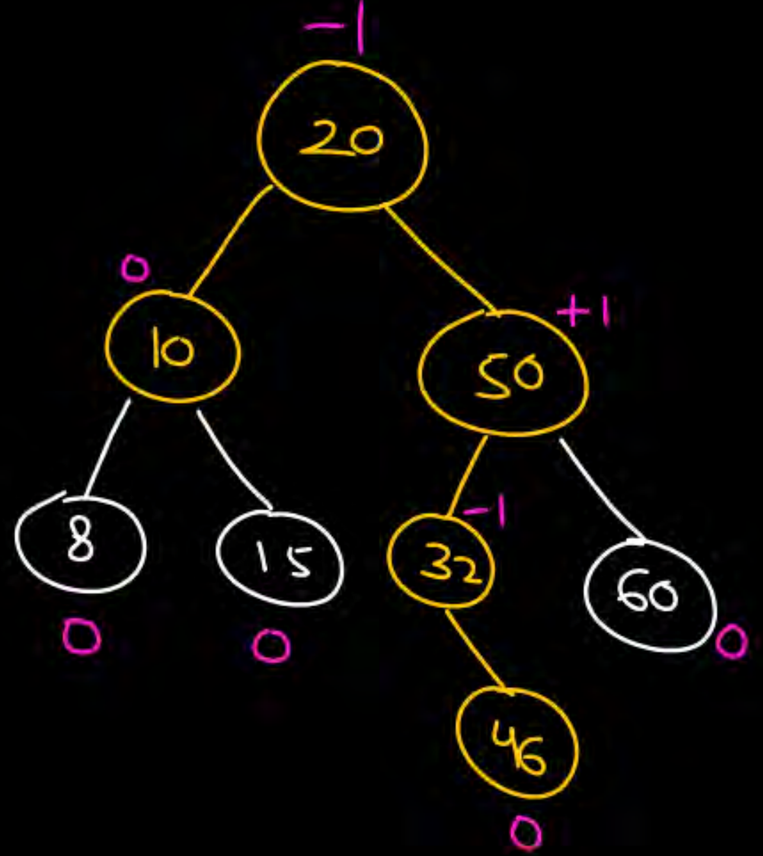
Const. AVL tree by inserting keys in given order.

50, 20, 60, 10, 8, 15. 32, 46, 11, 38 ✓

10, 20, 50



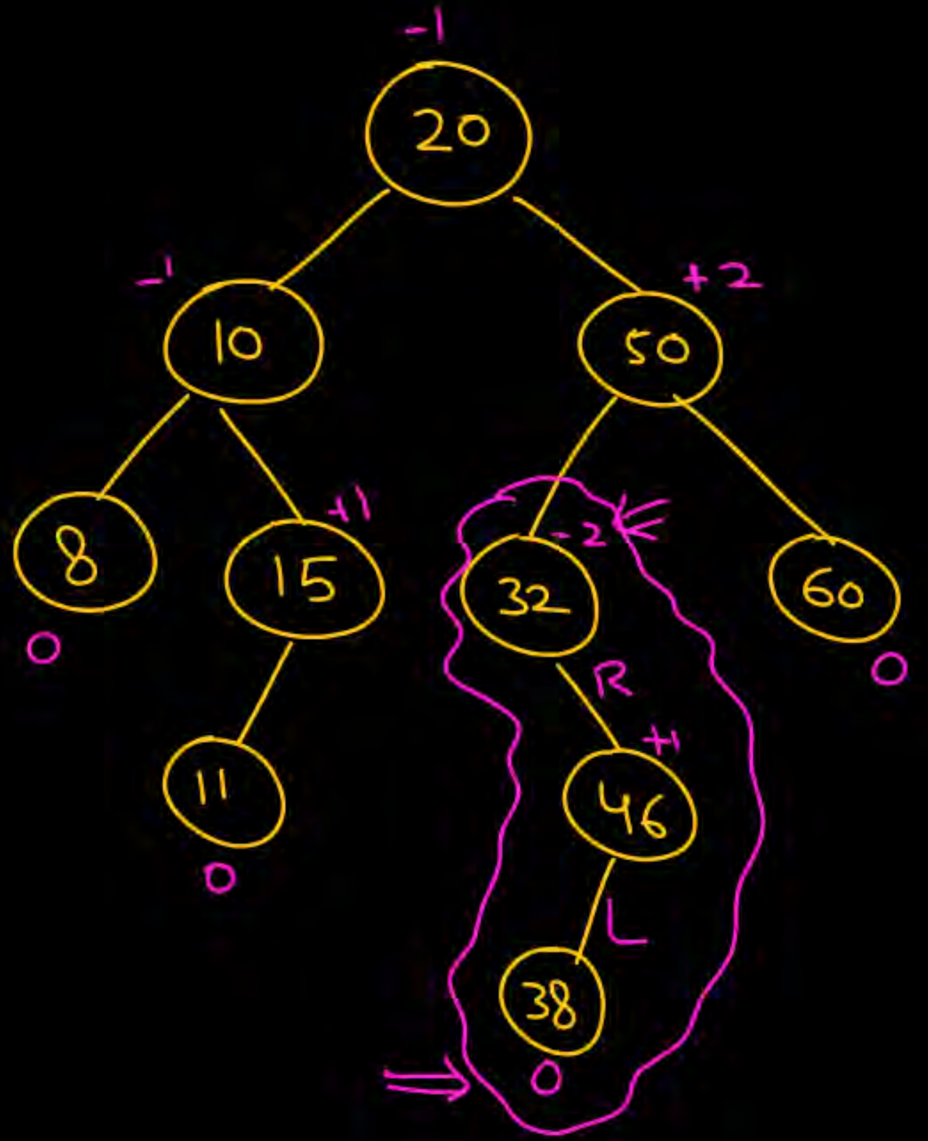


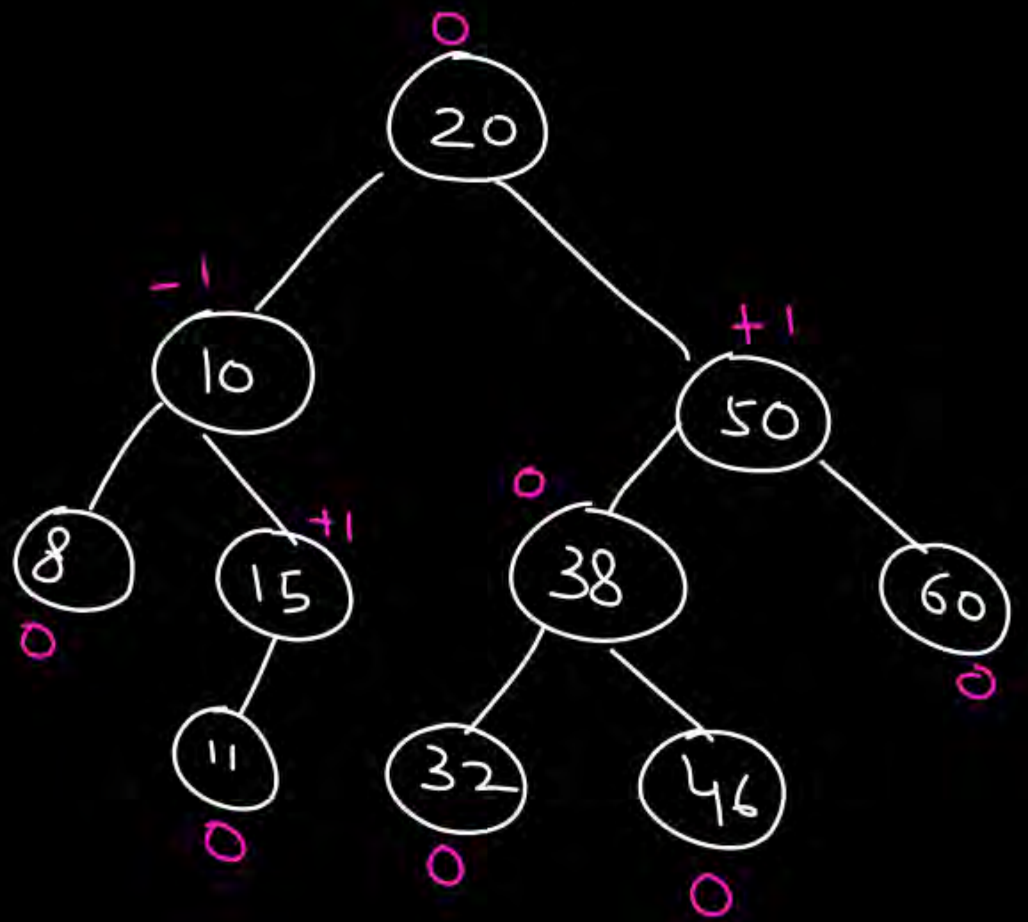


Const AVL tree by inserting keys in given order.

50, 20, 60, 10, 8, 15. 32, 46, 11, 38 ✓

32, 38, 46





$$H, I, J, B, A, E, C, F, D, G, K, L \left.\vphantom{X}\right\} \underleftrightarrow{\text{H.W}}$$

THANK YOU GW SOLDIERS !