CS & IT ENGINEERING

Data Structure

Tree

Chapter- 5

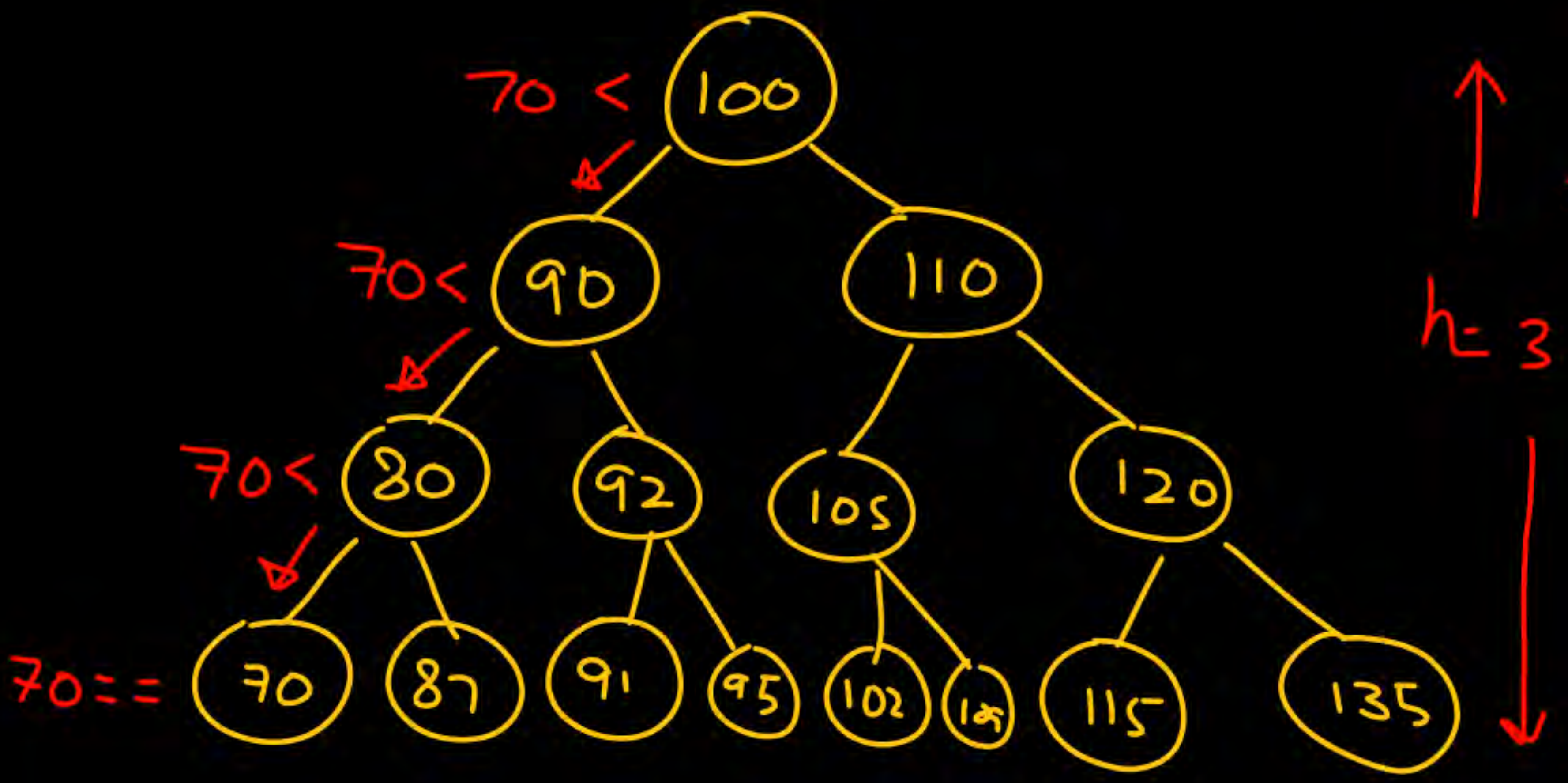
Lec- 05

By- Pankaj Sharma sir

TOPICS TO BE COVERED

- Tree-V

Search in a BST



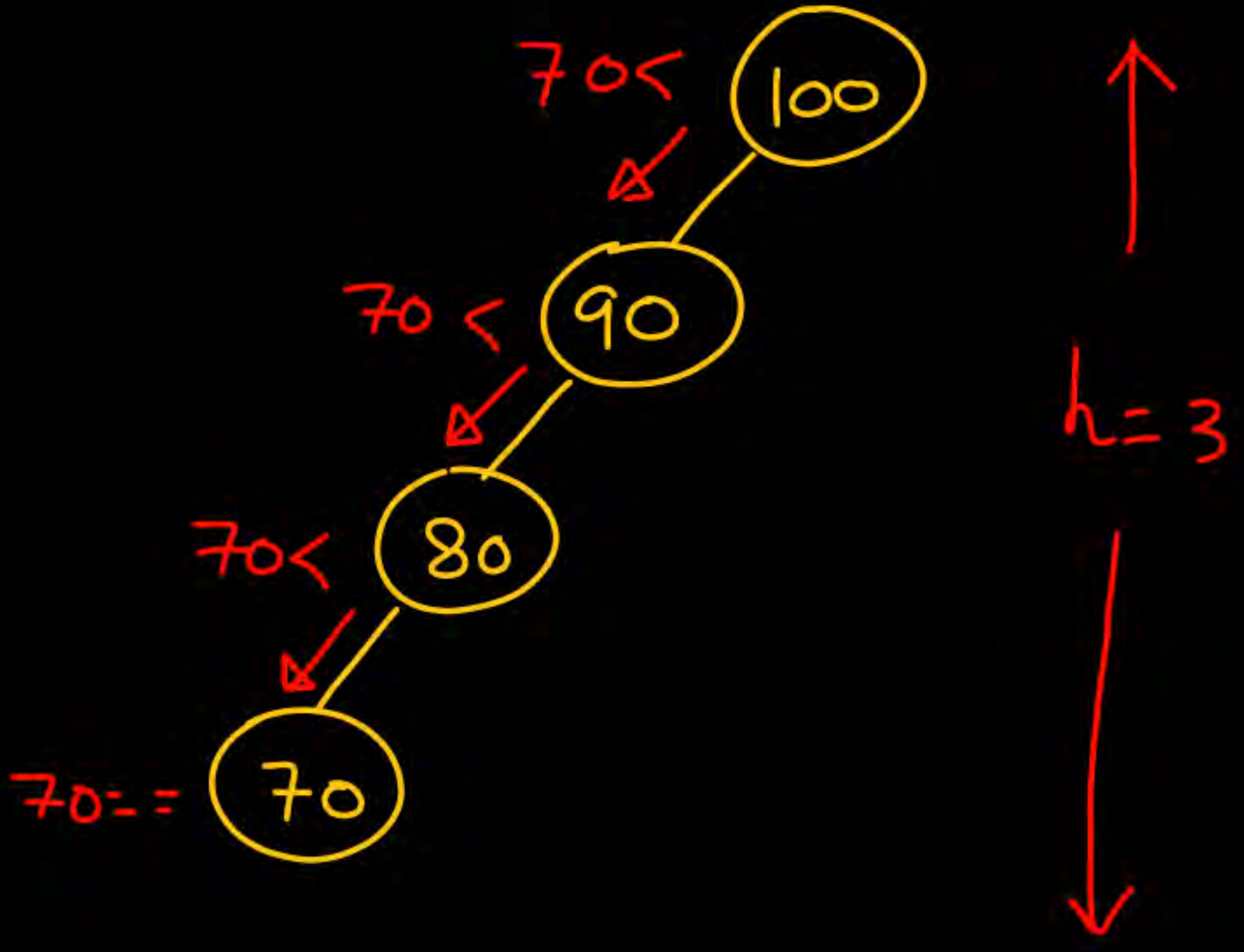
Key = 70

$\#\text{comp.} = 4$

$= h+1$

$$\boxed{h = O(\log_2 n)}$$

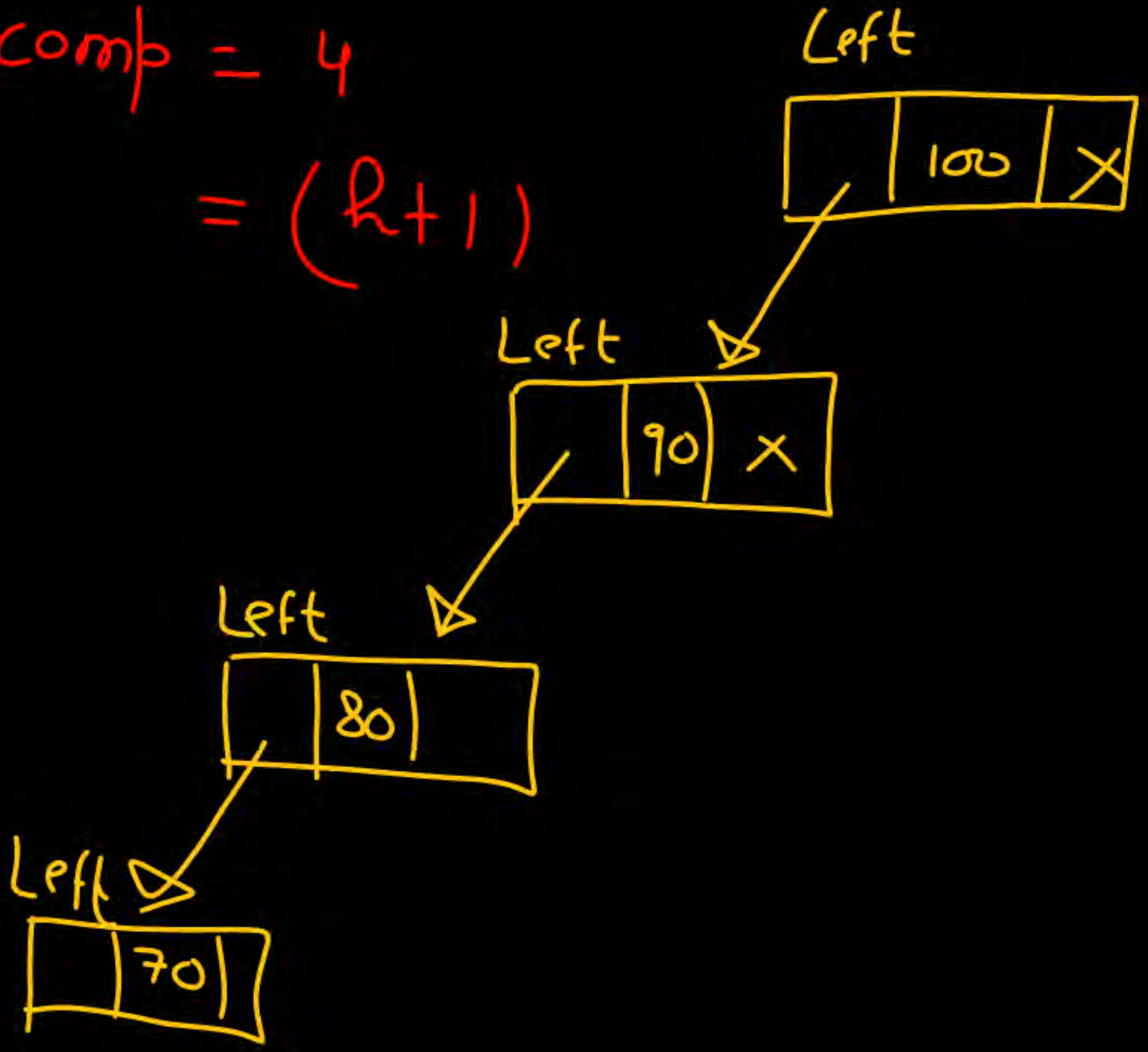
Left Skewed tree



Key = 70

comp = 4

$= (h+1)$



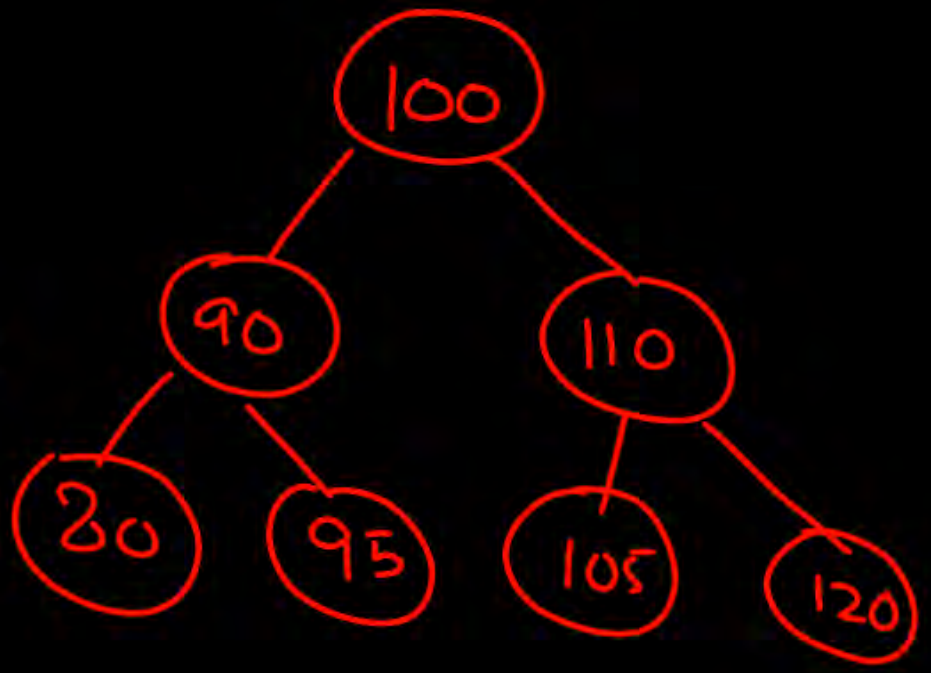
# comp ⇒ $h+1$ → $O(n)$ skewed

→ $O(\log_2 n)$ FBT ✓

Insert :

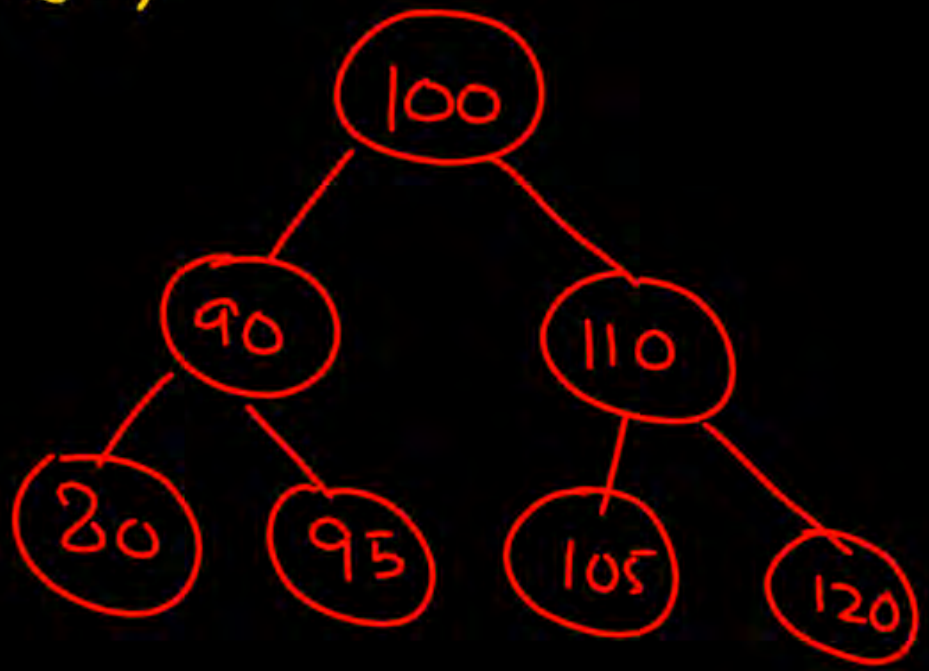
$\boxed{70}$

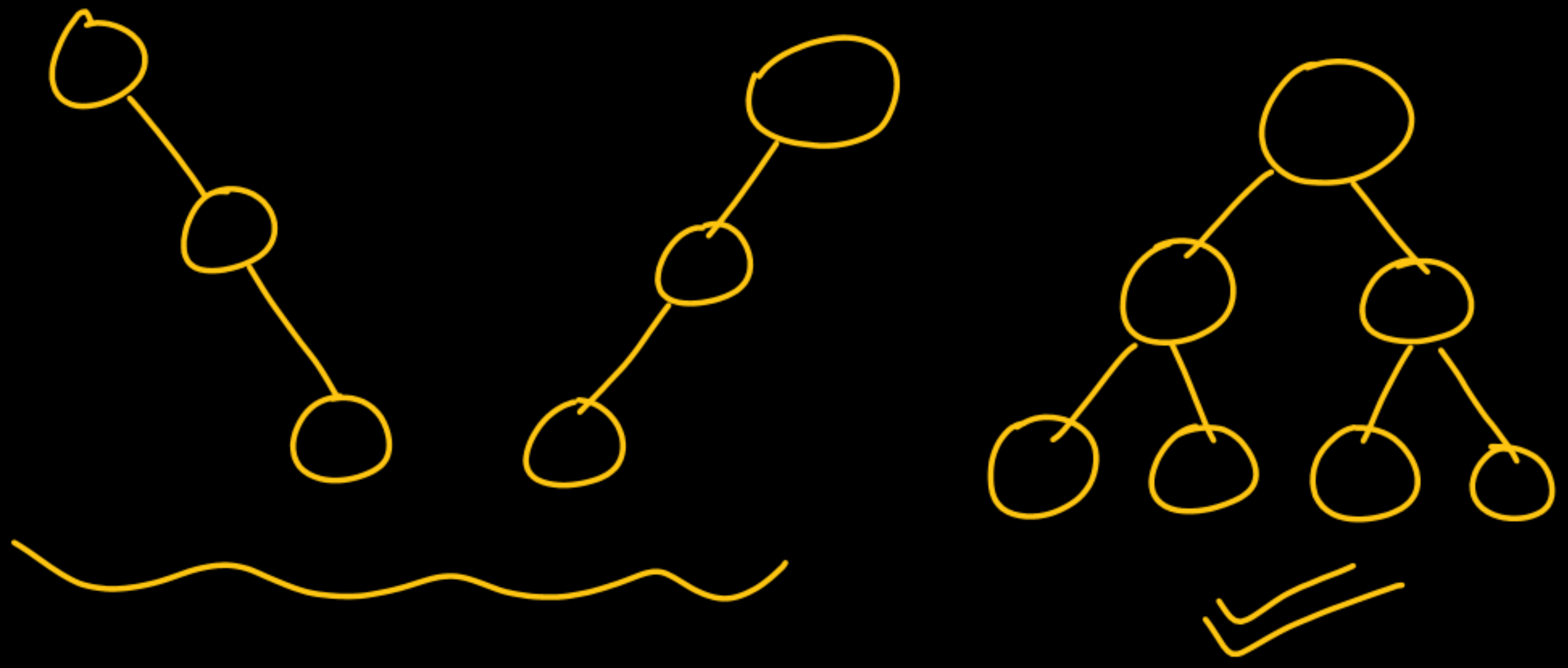
a) Search 70 → $O(n)$



$$\begin{tikzcd}
 & O(n) \quad \text{skewed} \\
\# \text{comp} \Rightarrow h+1 \arrow[ur] \arrow[dr] & \\
 & O(\log_2 n) \quad \text{FBT} ✓
\end{tikzcd}$$

Insert → $O(n)$





Deletion from a BST

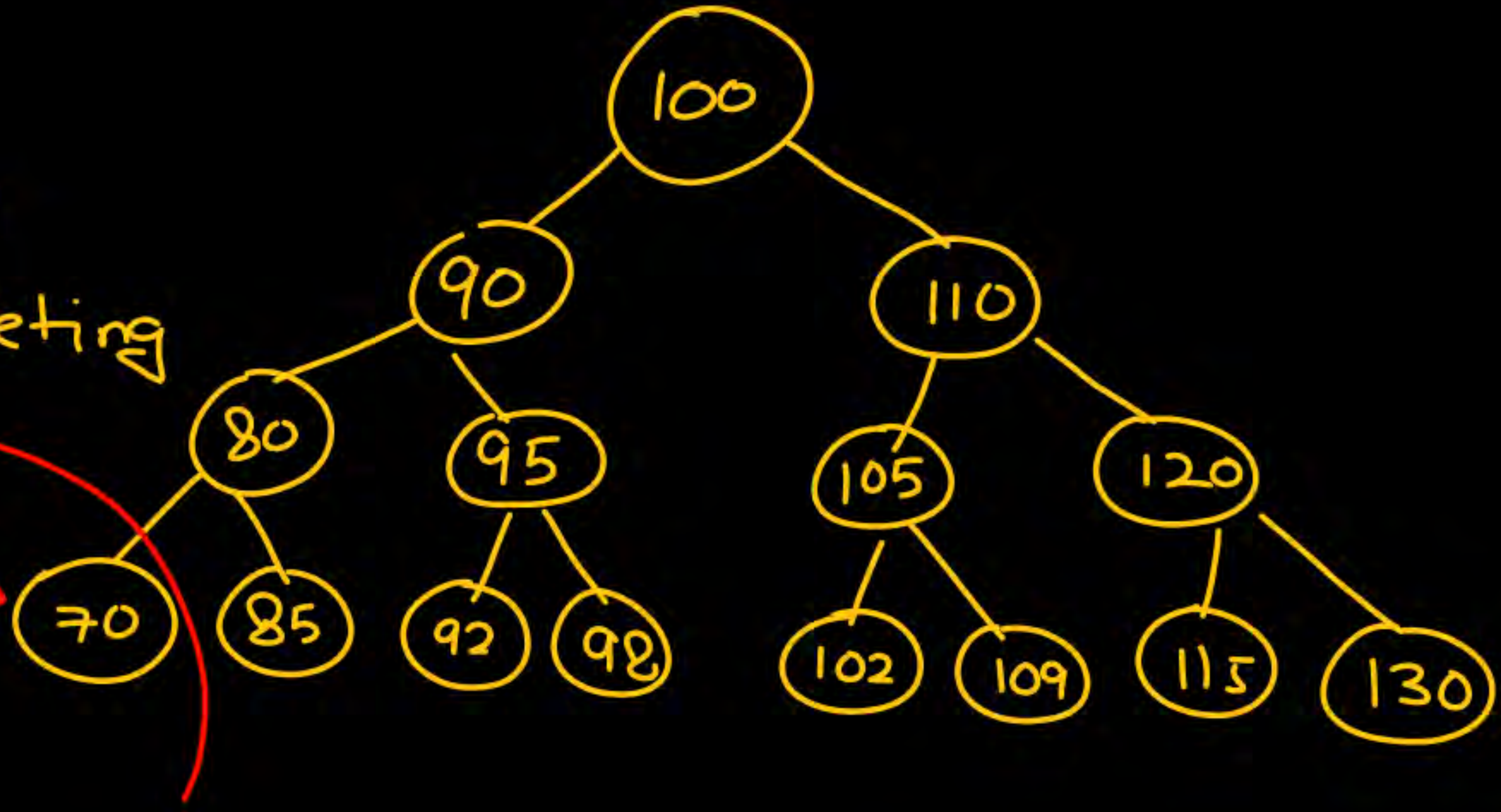
(i) Node with 0 - child (leaf node)

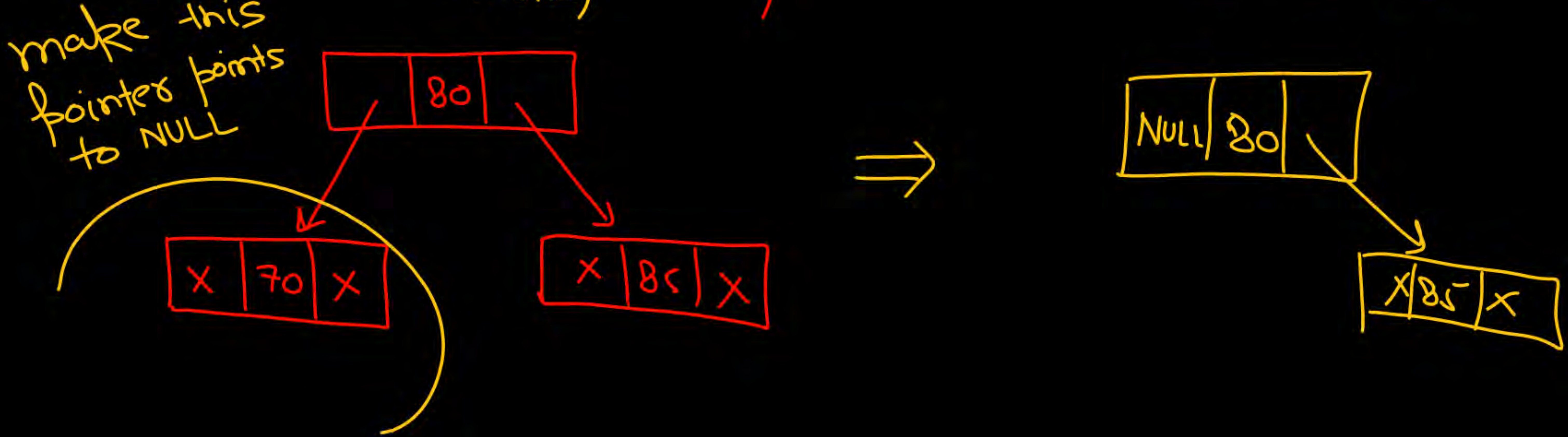
(ii) Node with 1 - child

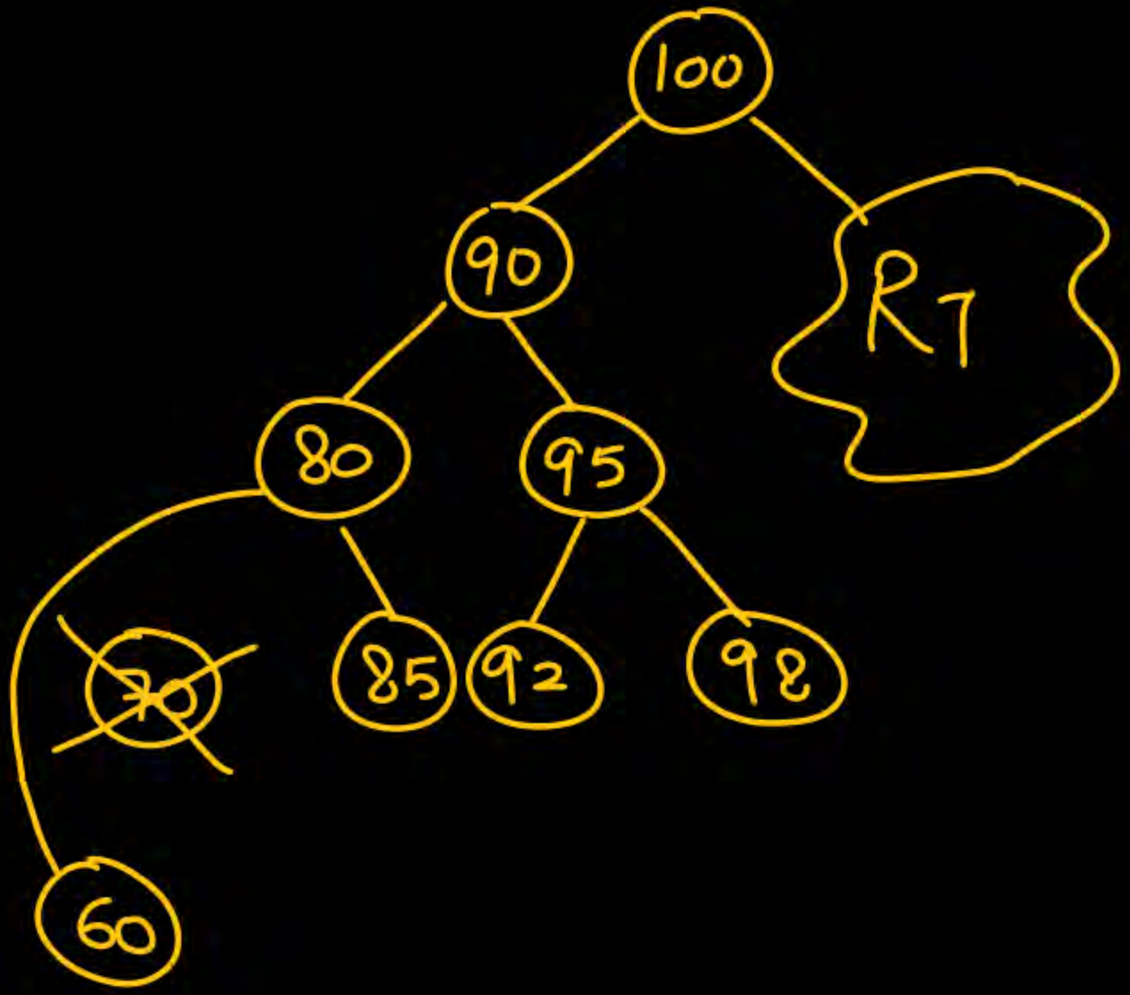
(iii) Node with 2 - child.

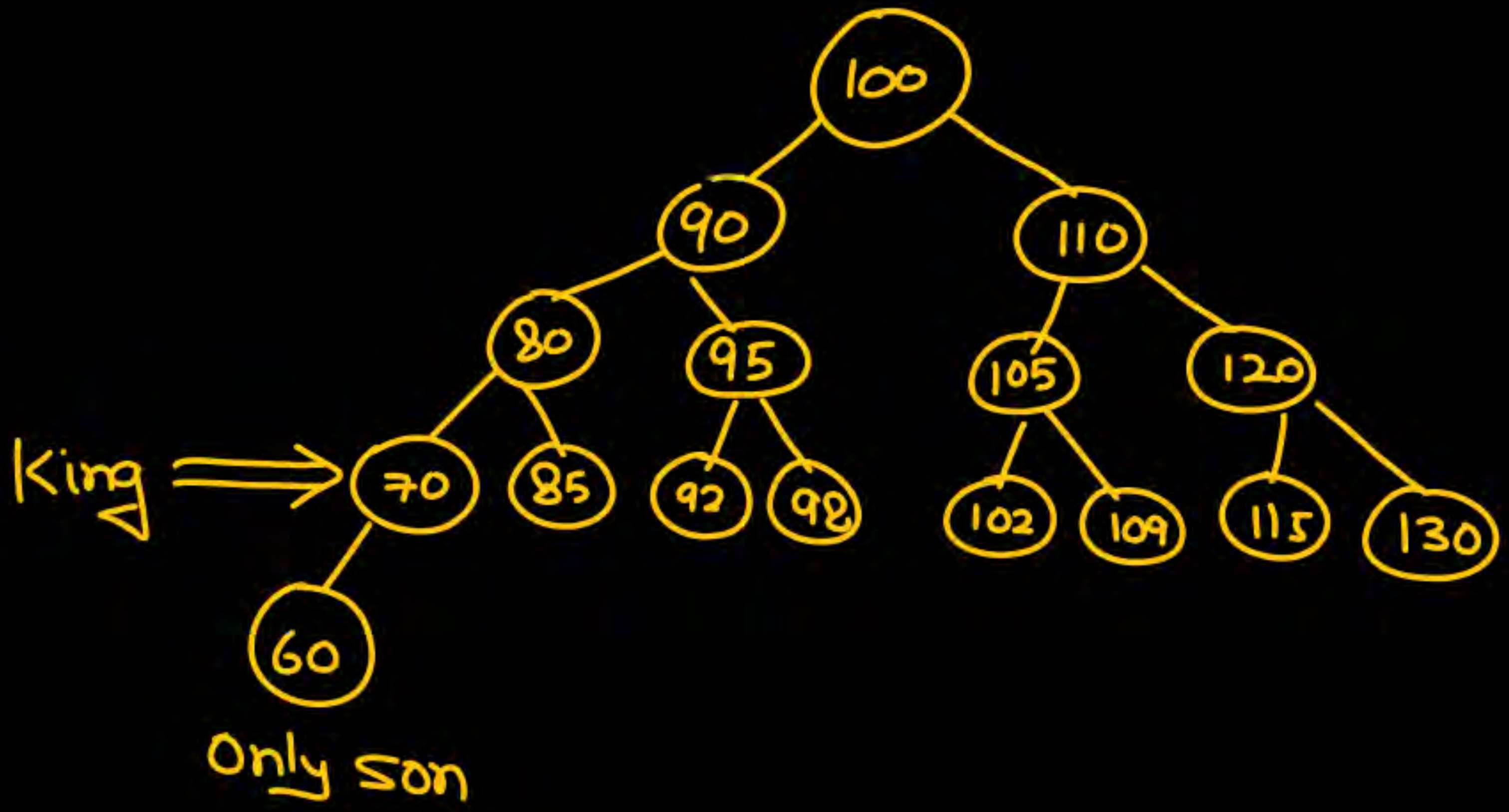
Key = 70

we need to identify the parent pointer of the deleting node, which is pointing to node (to be deleted)




make this pointer points to NULL

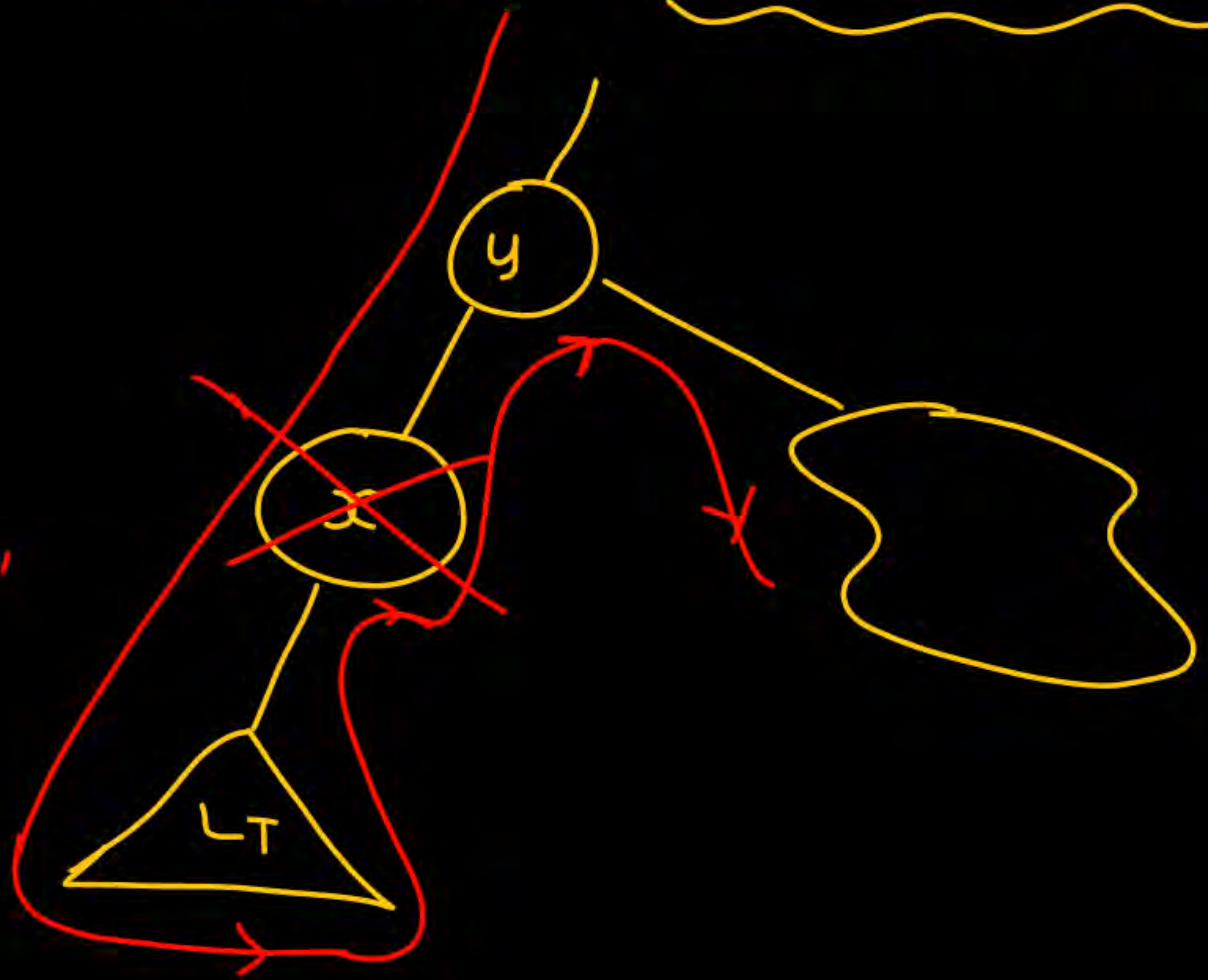

100
90
RT
80
95
70
85
92
98
60


100
90
110
80
95
105
120
King
70
85
92
98
102
109
115
130
60
Only son

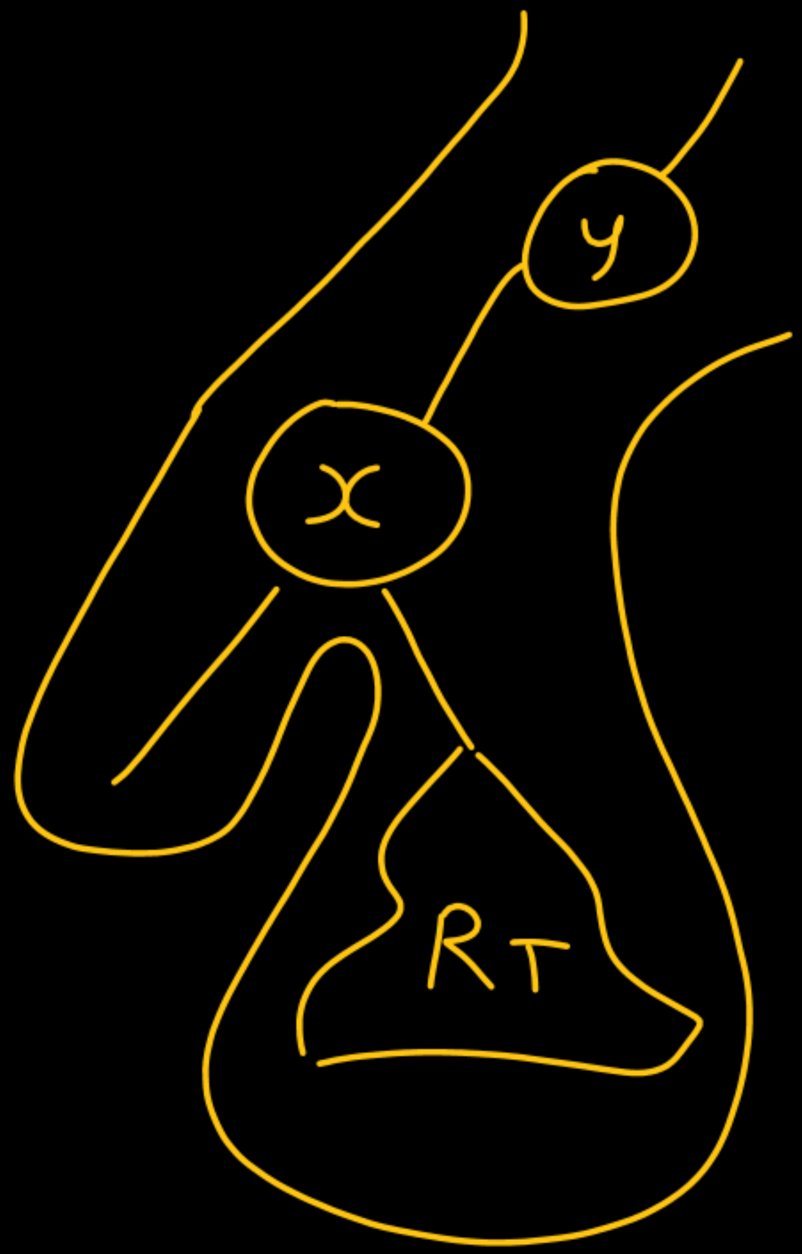
Why BST

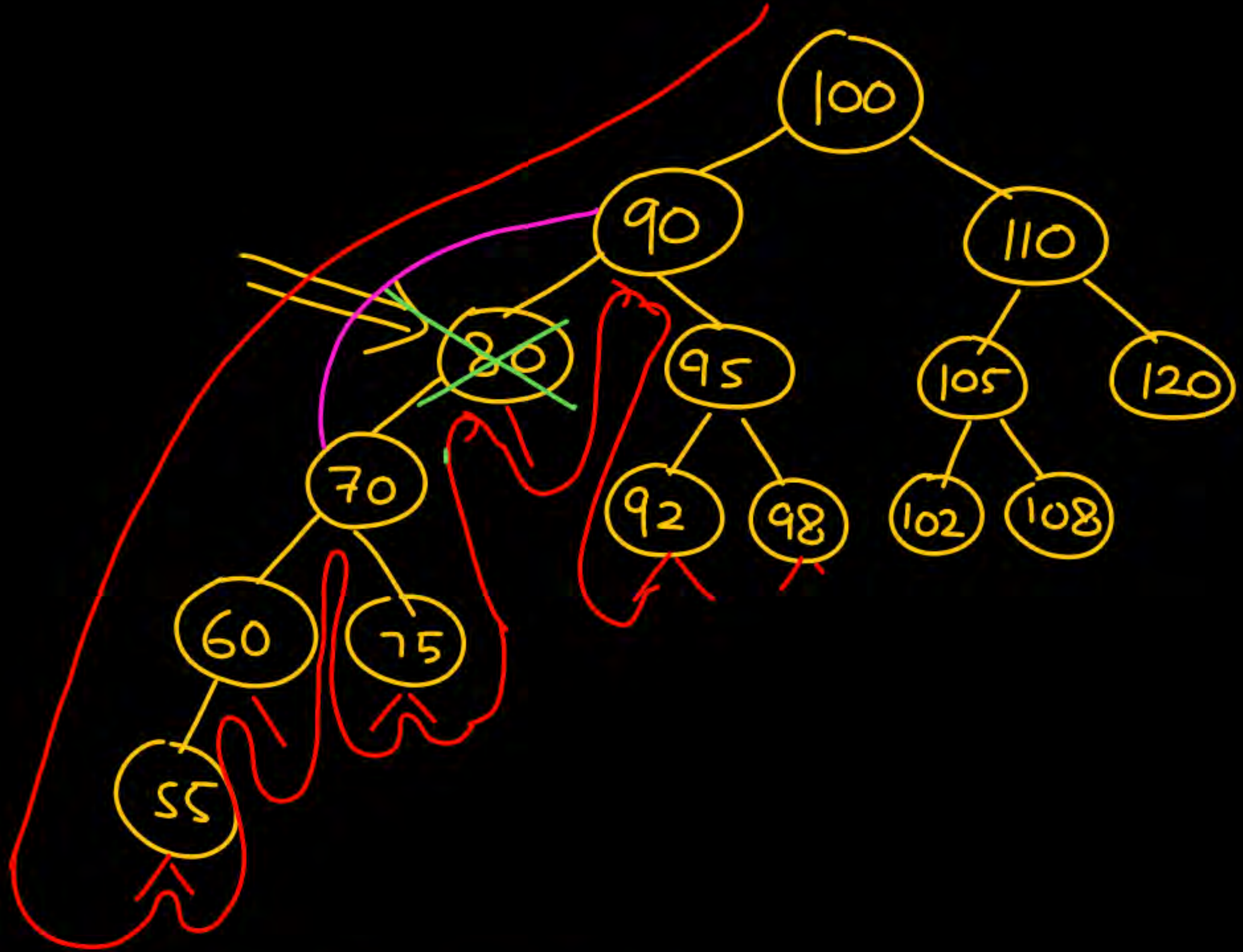
Inorder traversal

$\underbrace{L_T}\ x \qquad ,y,$



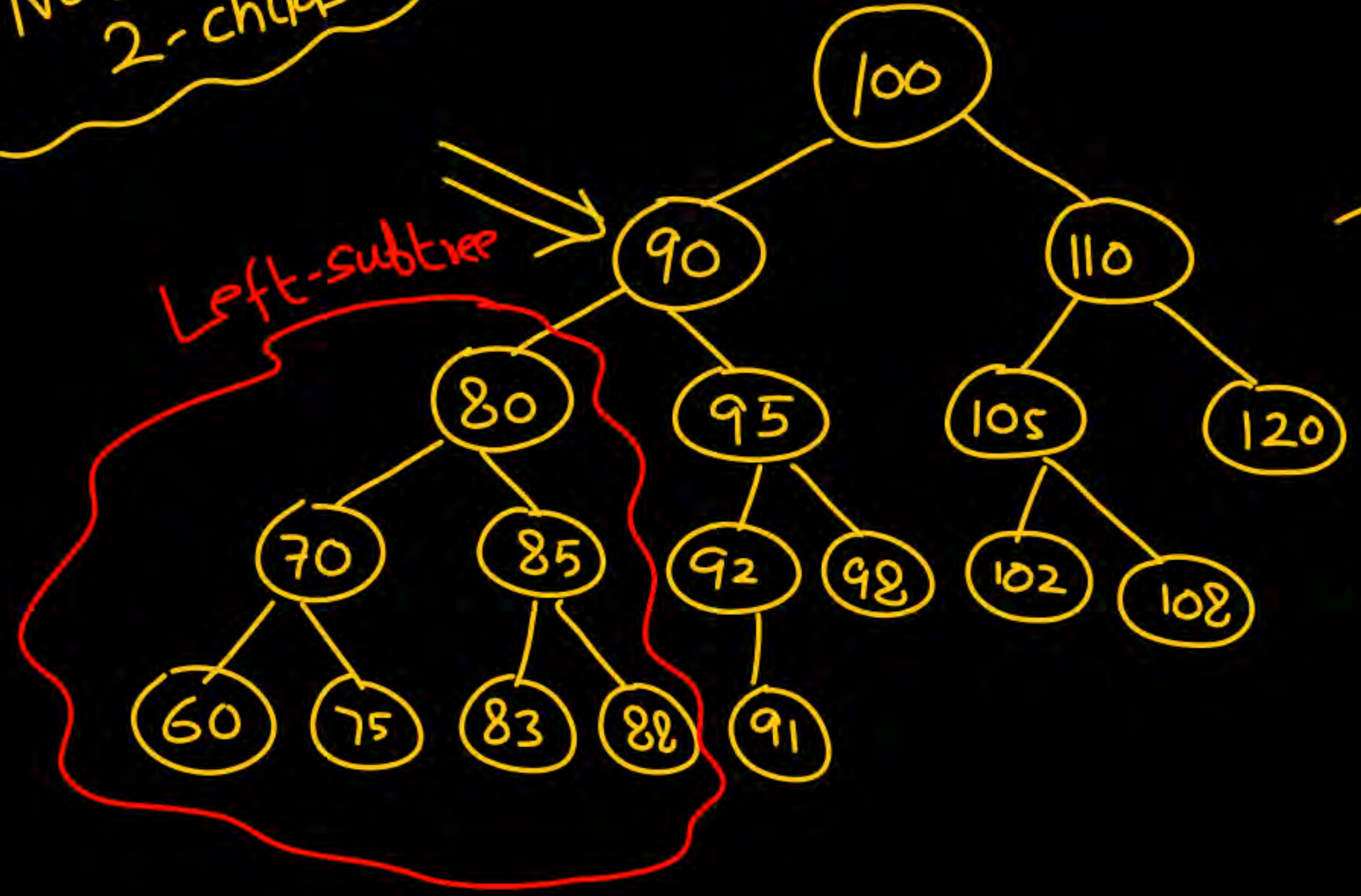
~~x~~, RT, y





$\underbrace{55\ 60\ 70\ 75}_{LT}$, ~~80~~, 90, 92, 95, 98, - -

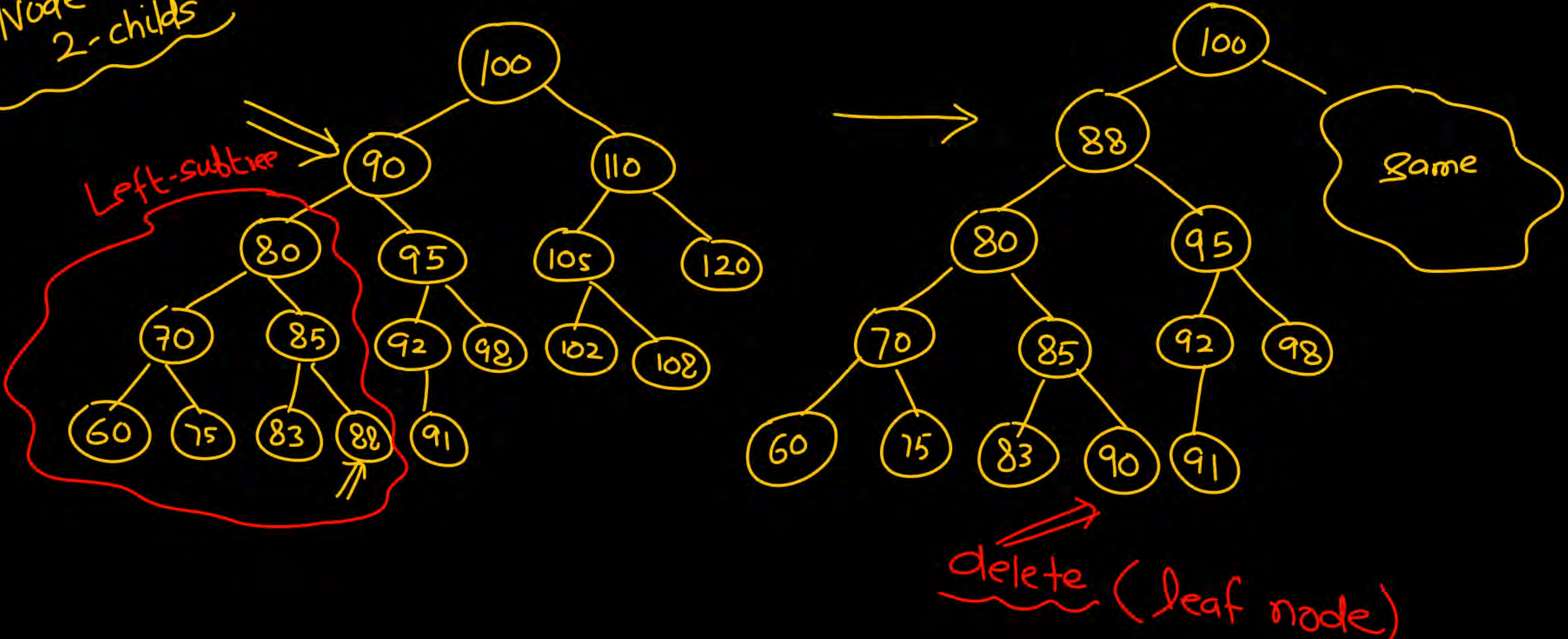
Node with 2-childs



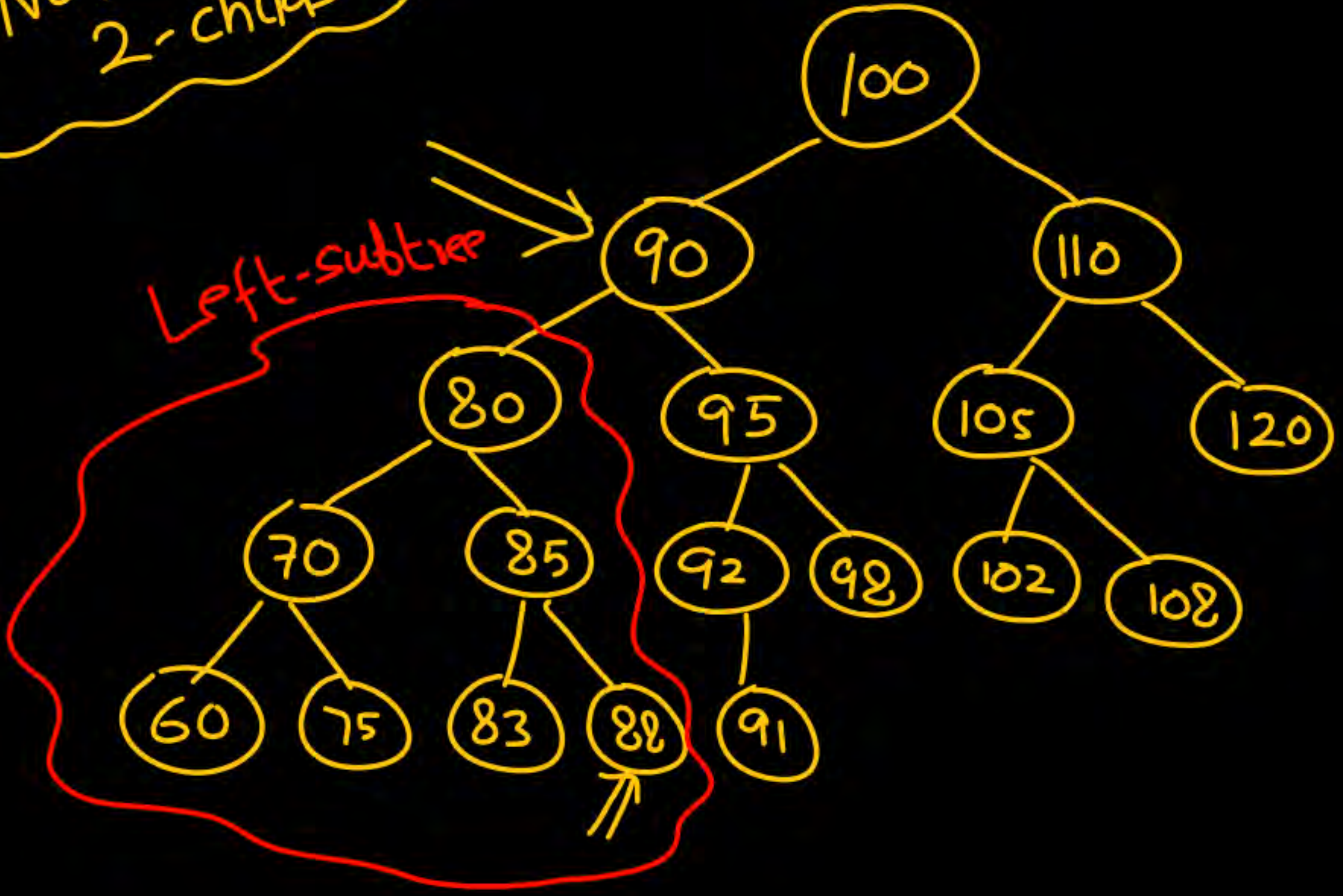
1st way : Replace the node(key) to be deleted with the maximum key from Left-Subtree & then perform deletion

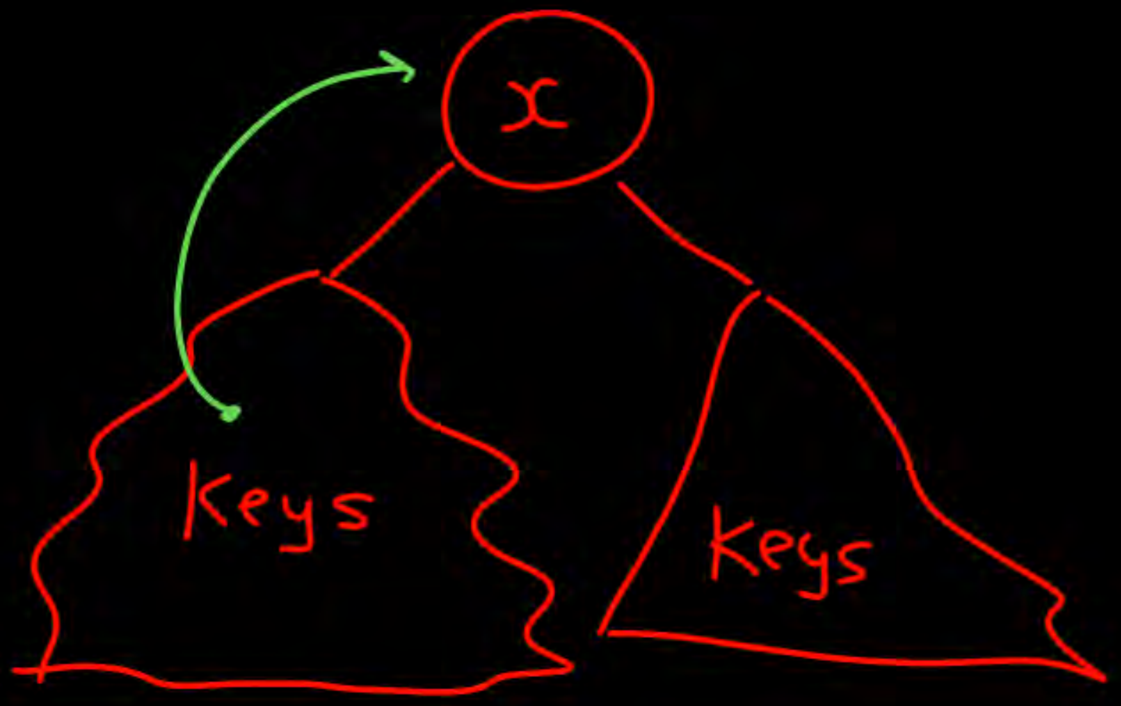
2nd way : Replace the node (Key) to be deleted with the min. key from right subtree & then perform deletion.

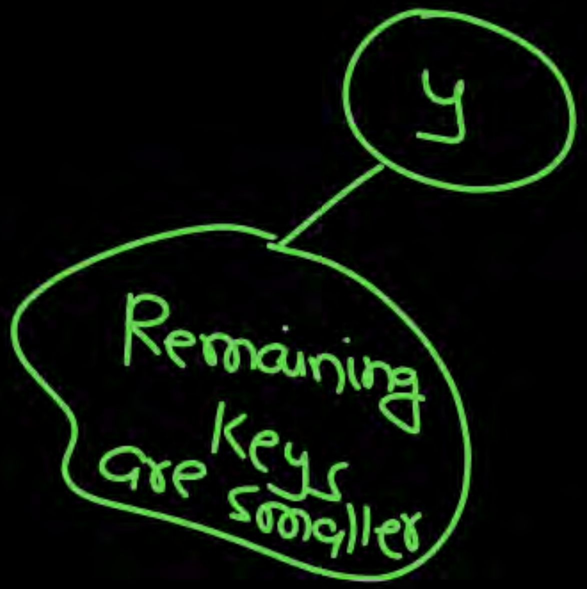
Node with 2-childs



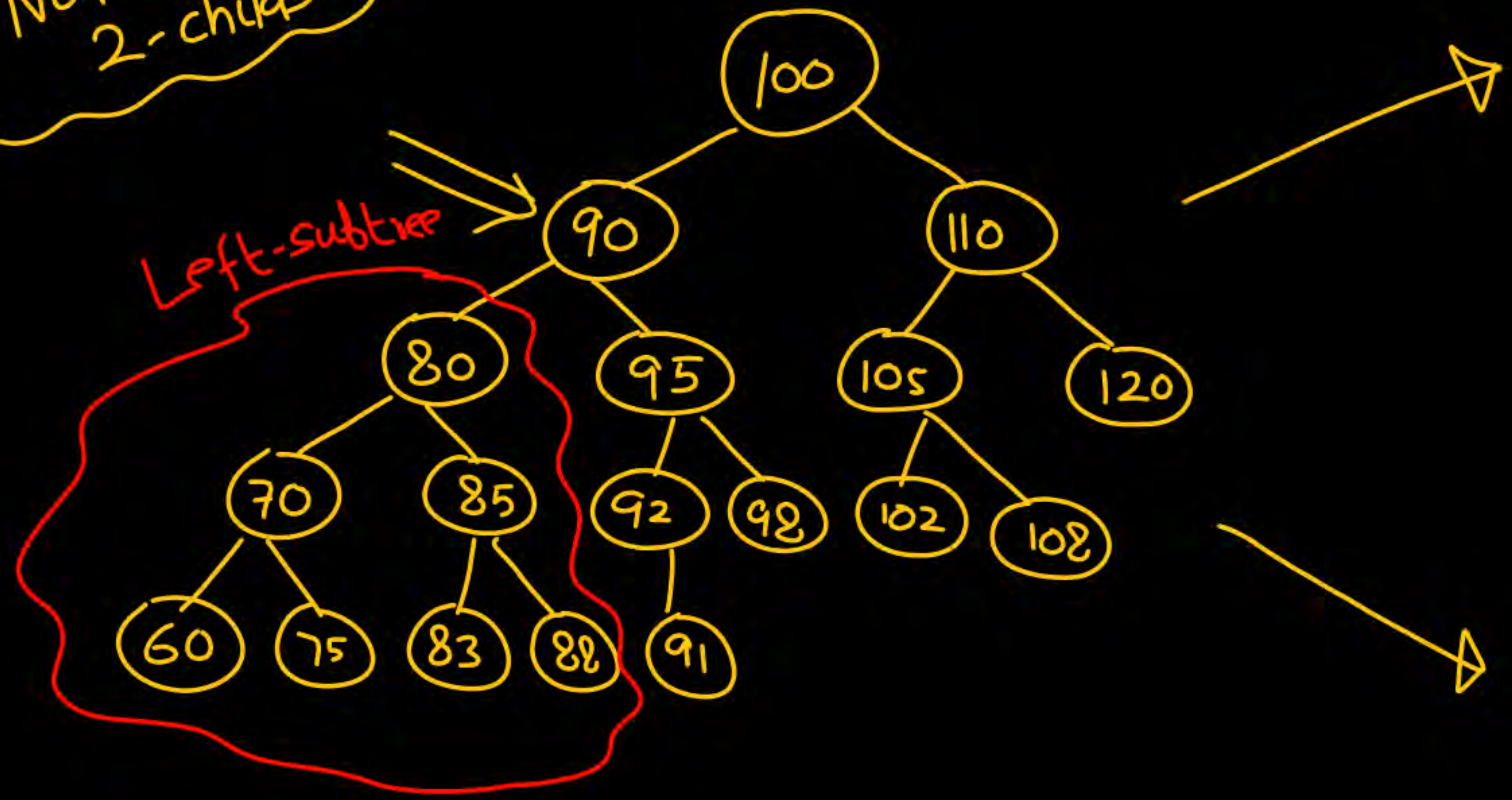
Node with 2-childs







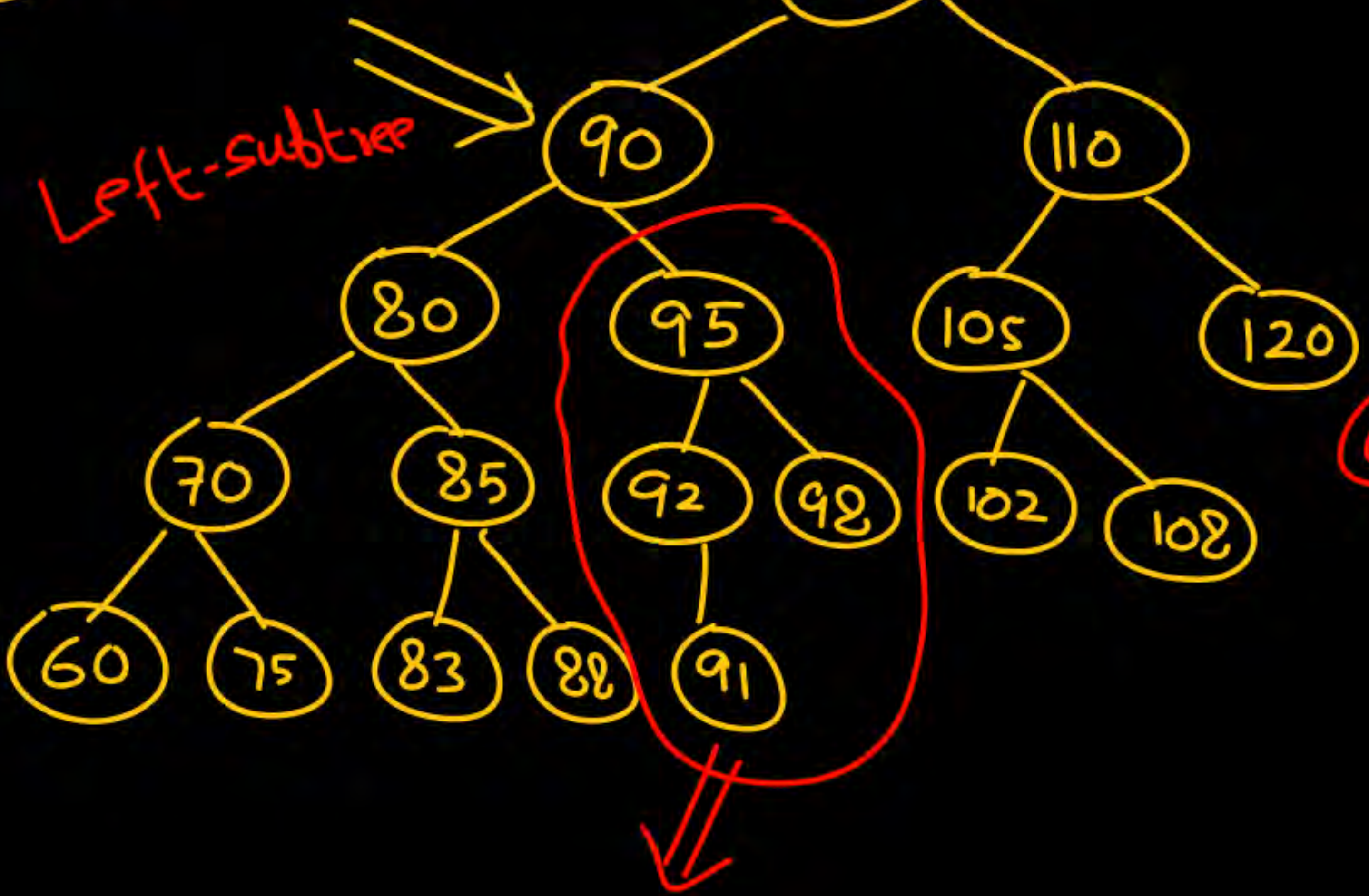
Node with 2-childs

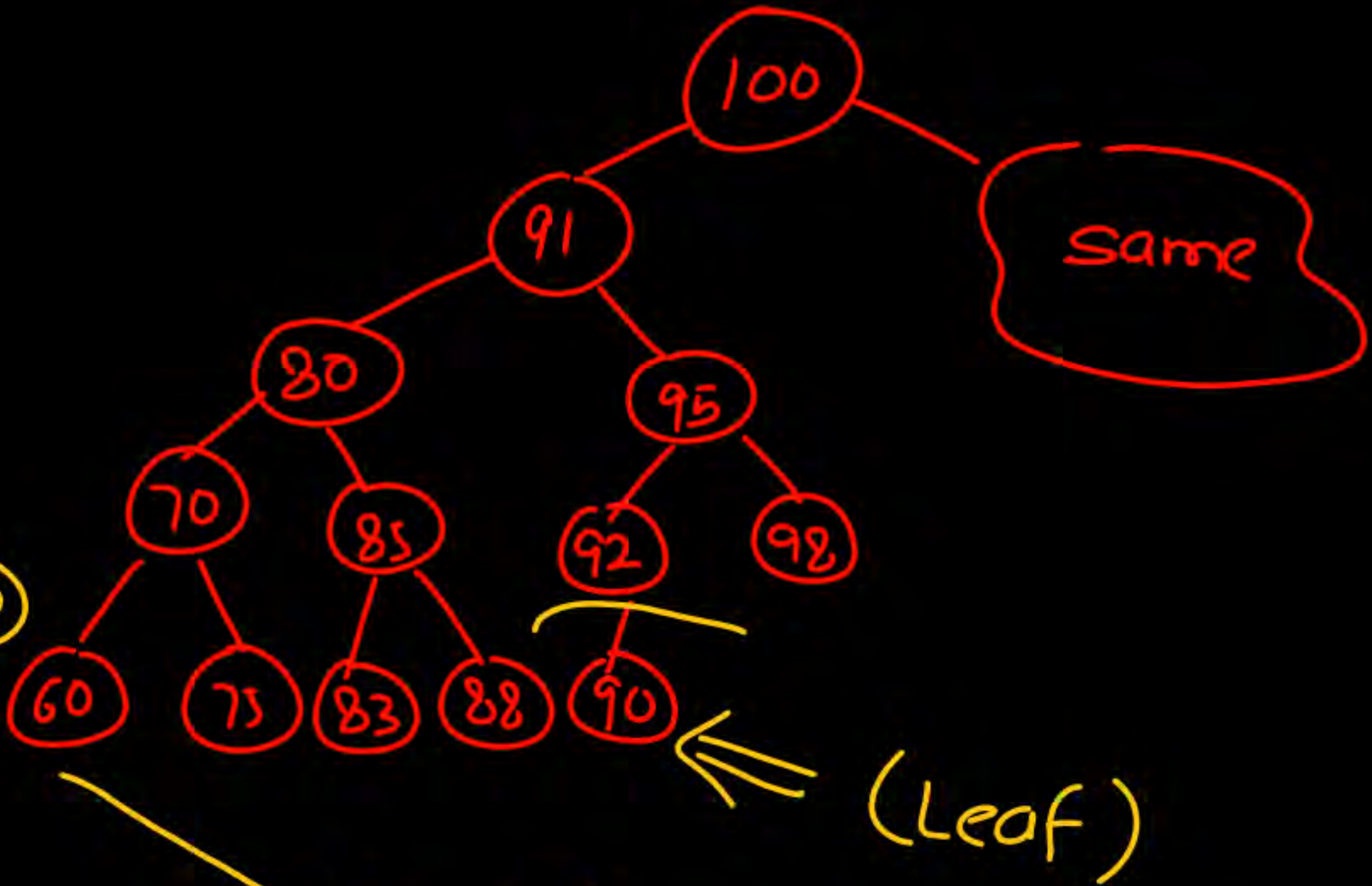


1^{st} way : Replace the node(key) to be deleted with the maximum key from Left-Subtree & then perform deletion

2^{nd} way : Replace the node (Key) to be deleted with the min. key from right Subtree & then perform deletion.

Node with 2-childs



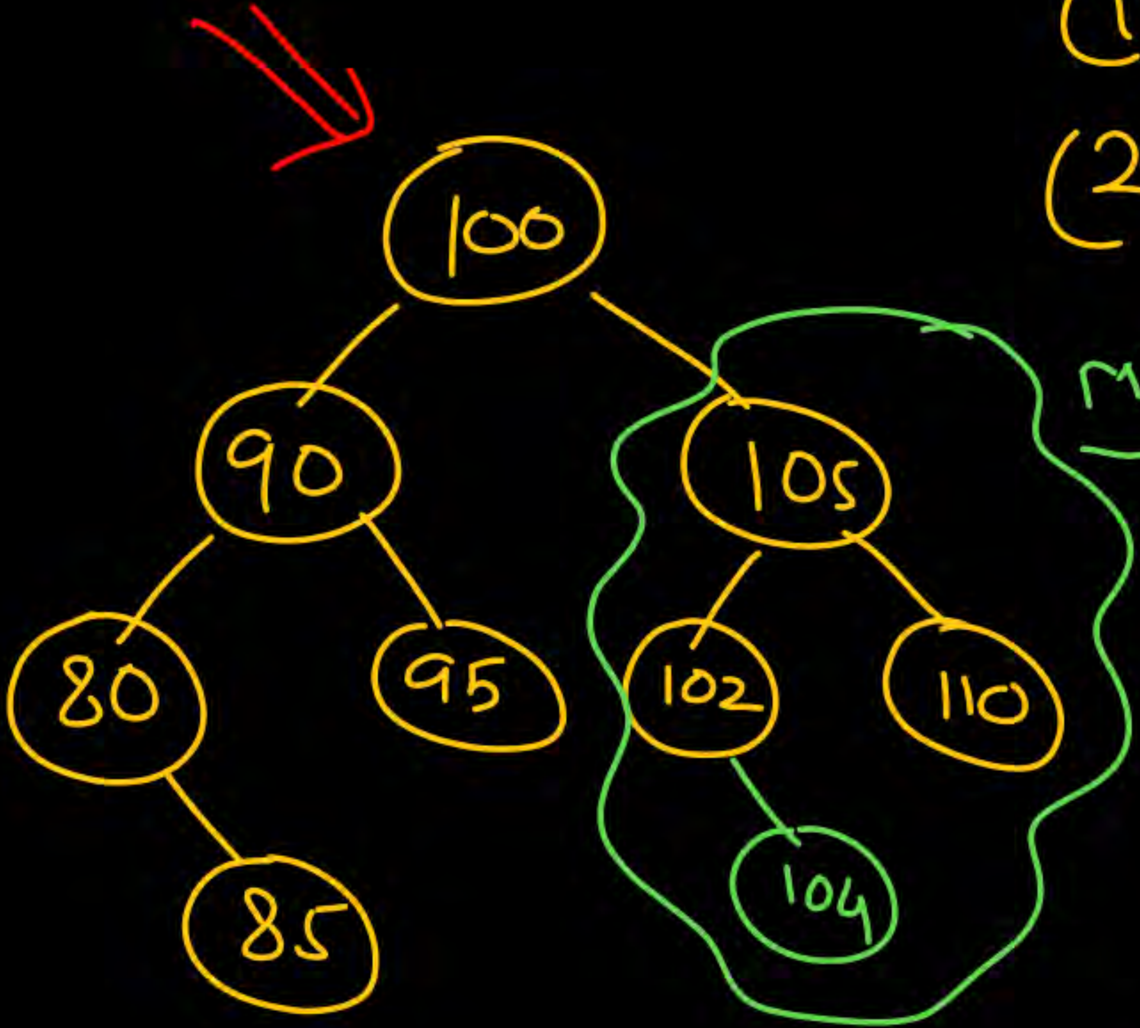


2nd way : Replace the node (Key) to be deleted with the min. key from right subtree & then perform deletion.

Node with min value can have 0 or 1 child

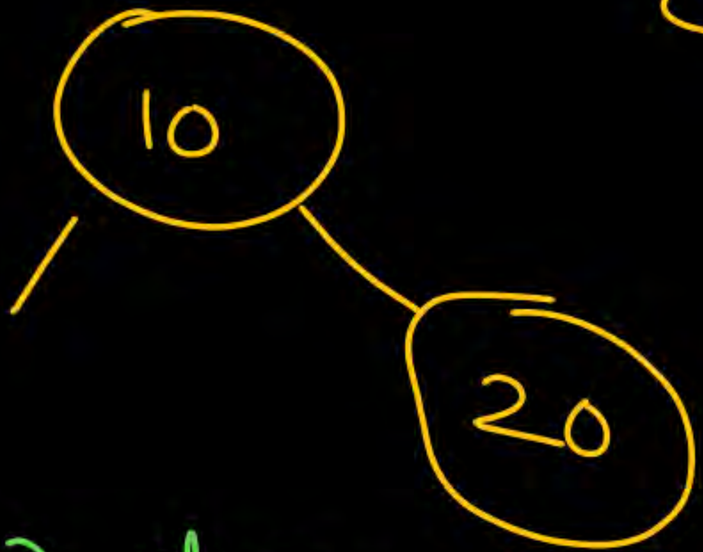
(1) leaf node

(2) It may have right child



min value

⟹



can not have a left child

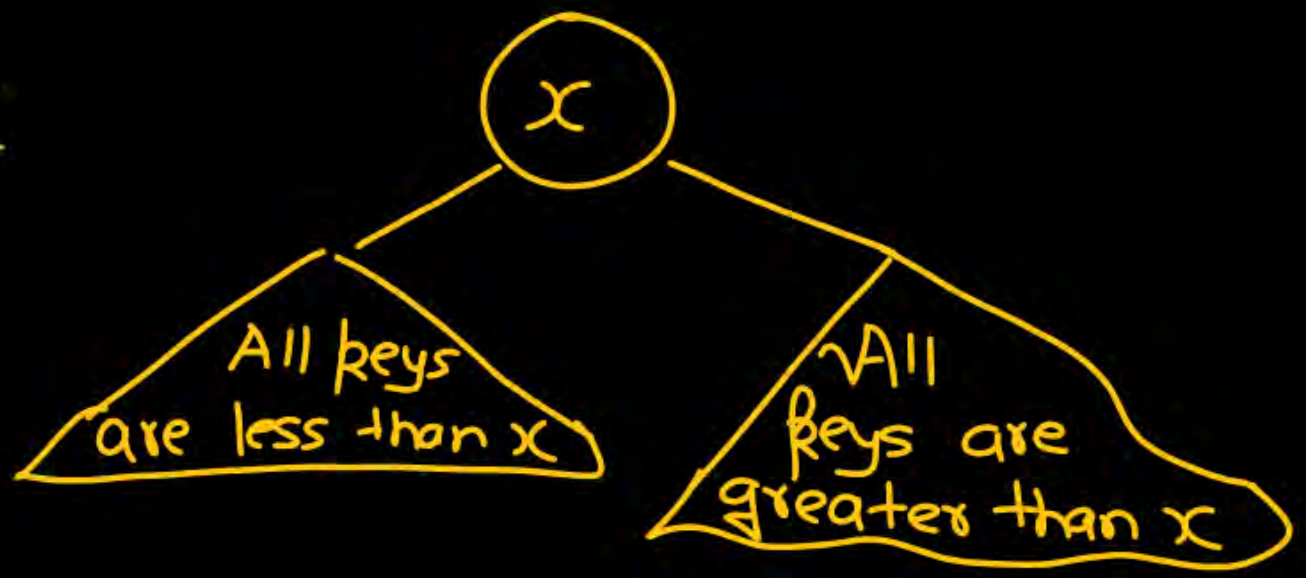
Why 102 does not have a left child

Balanced BST

AVL tree   height balanced tree

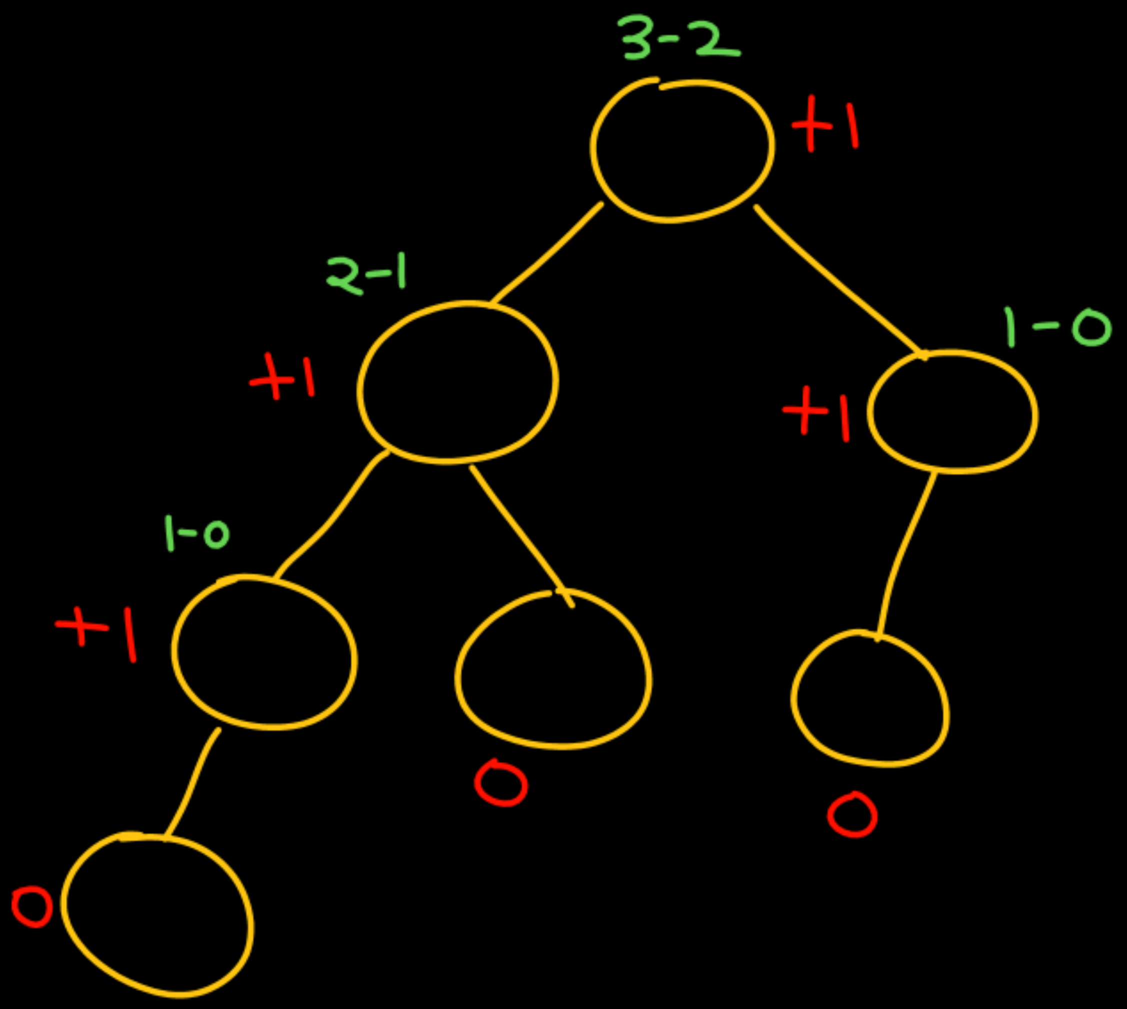
AVL tree, every node satisfies 2 properties

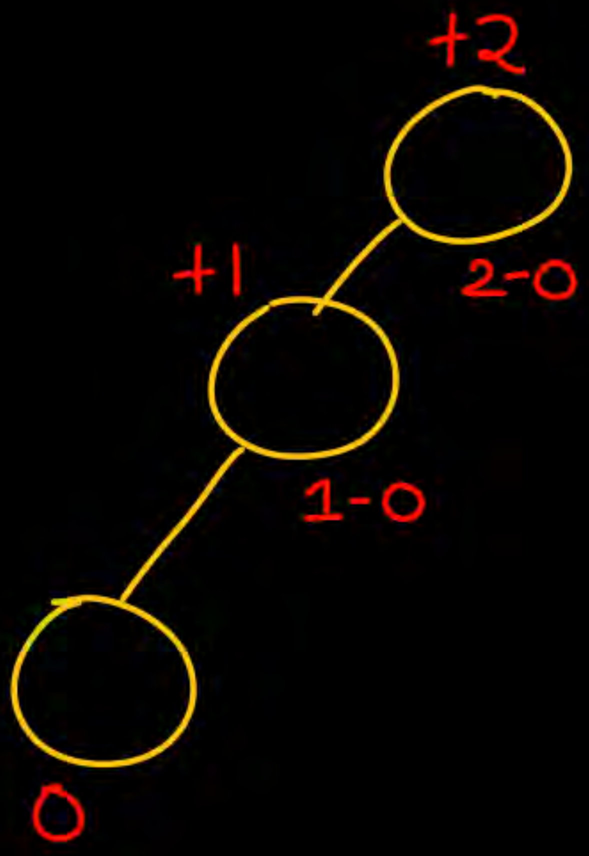
1) BST property :

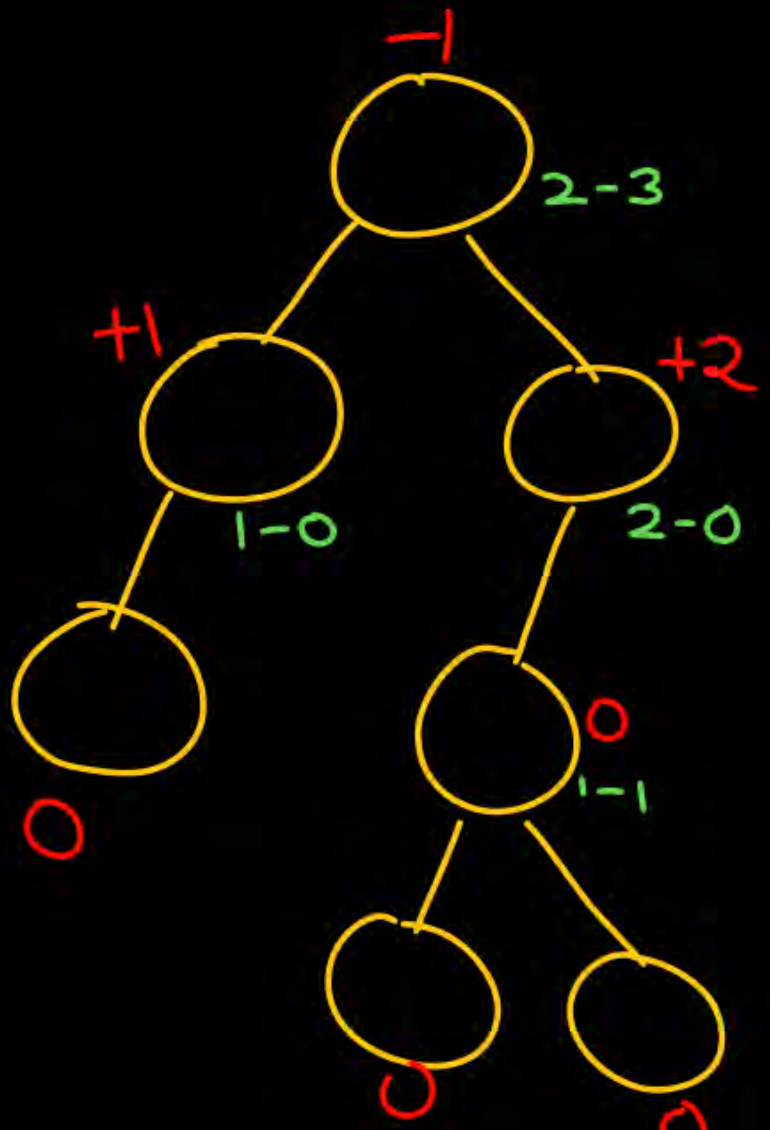


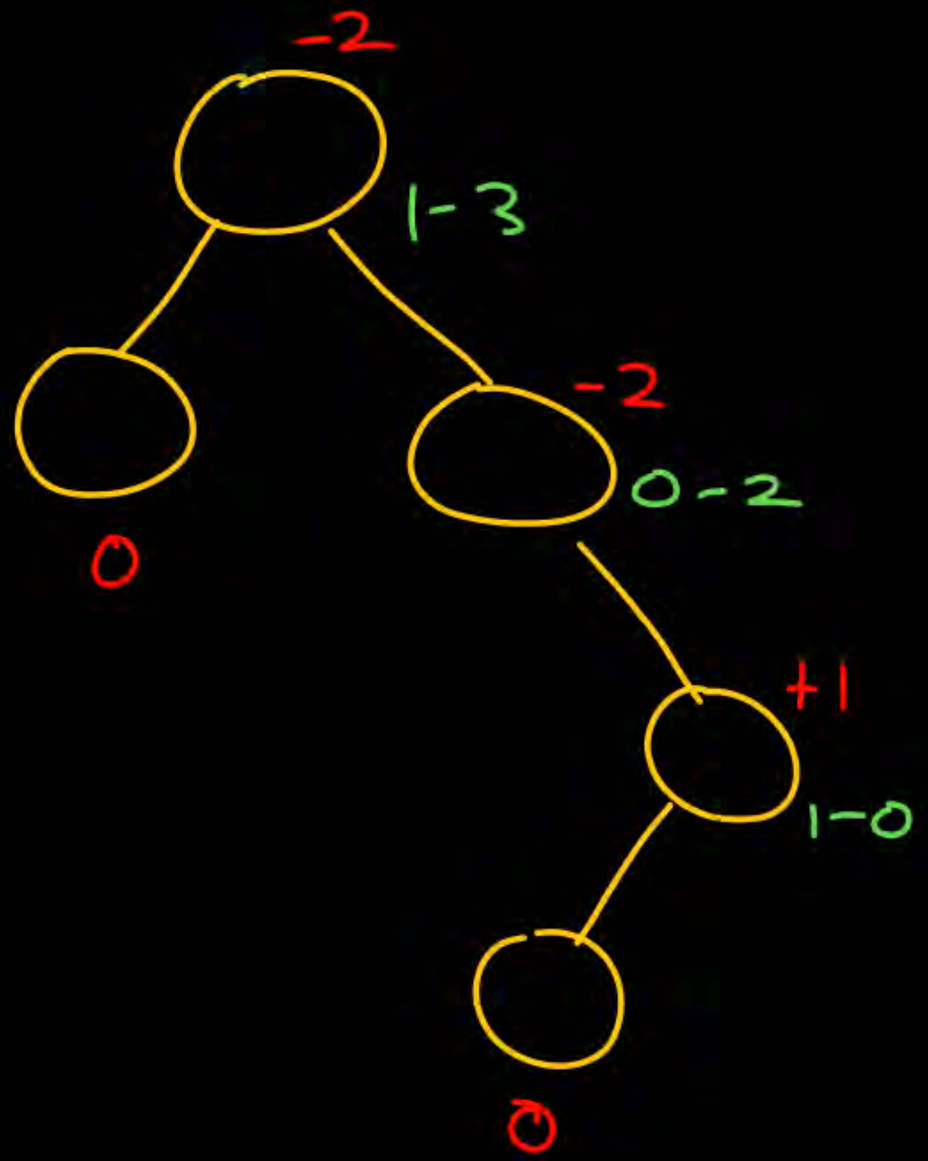
2) AVL tree property : The balancing factor of each node is either 0, -1 or +1.

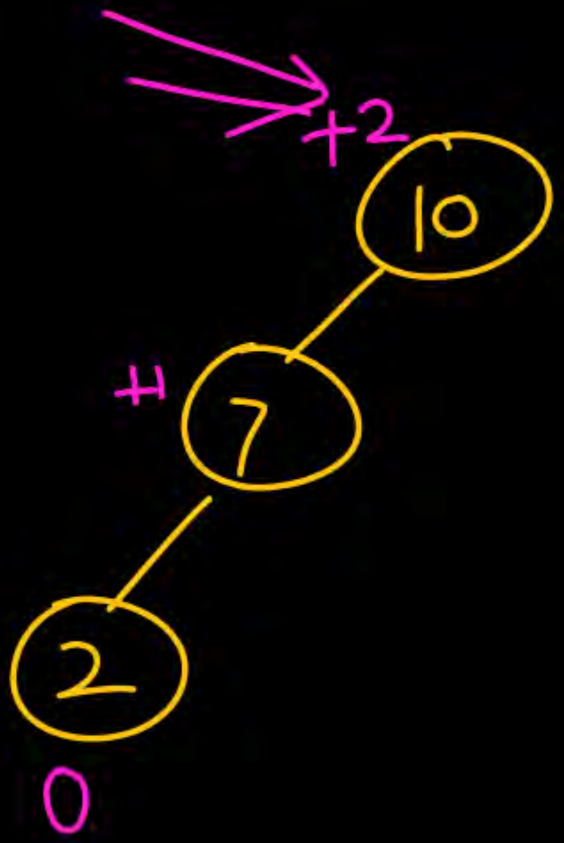
Balancing factor :




+2
2-0
+1
1-0
0


-1
2-3
+1
1-0
0
+2
2-0
0
1-1
0
0


-2
1-3
0
-2
0-2
+1
1-0
0



AVL tree

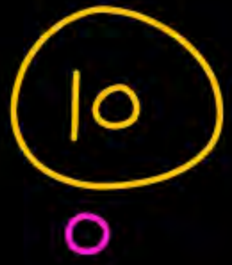
① BST property satisfied ✓

② AVL tree property ✗

→ bal. factor of each node can be +1, -1 or 0.

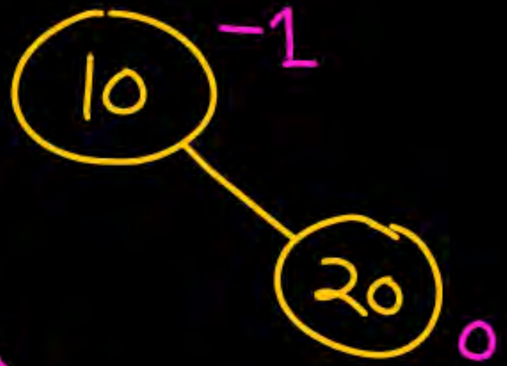
Const. AVL tree by inserting Key 10, 20, 30 in order ⇒ insert a node just like BST

a)



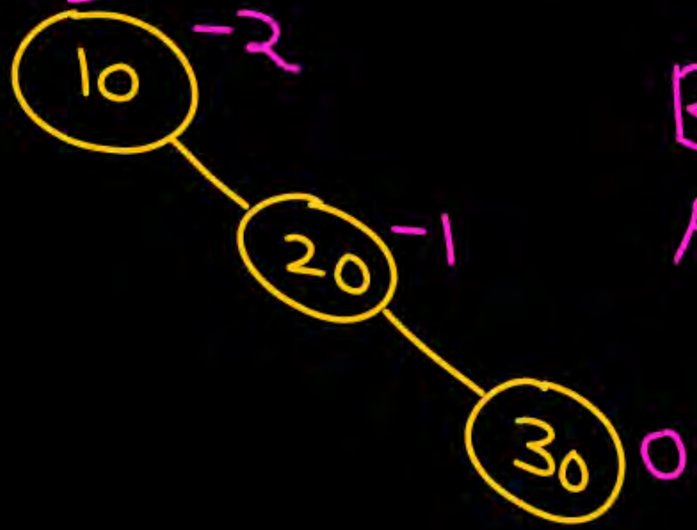
AVL tree ✓

b)



AVL tree ✓

c)



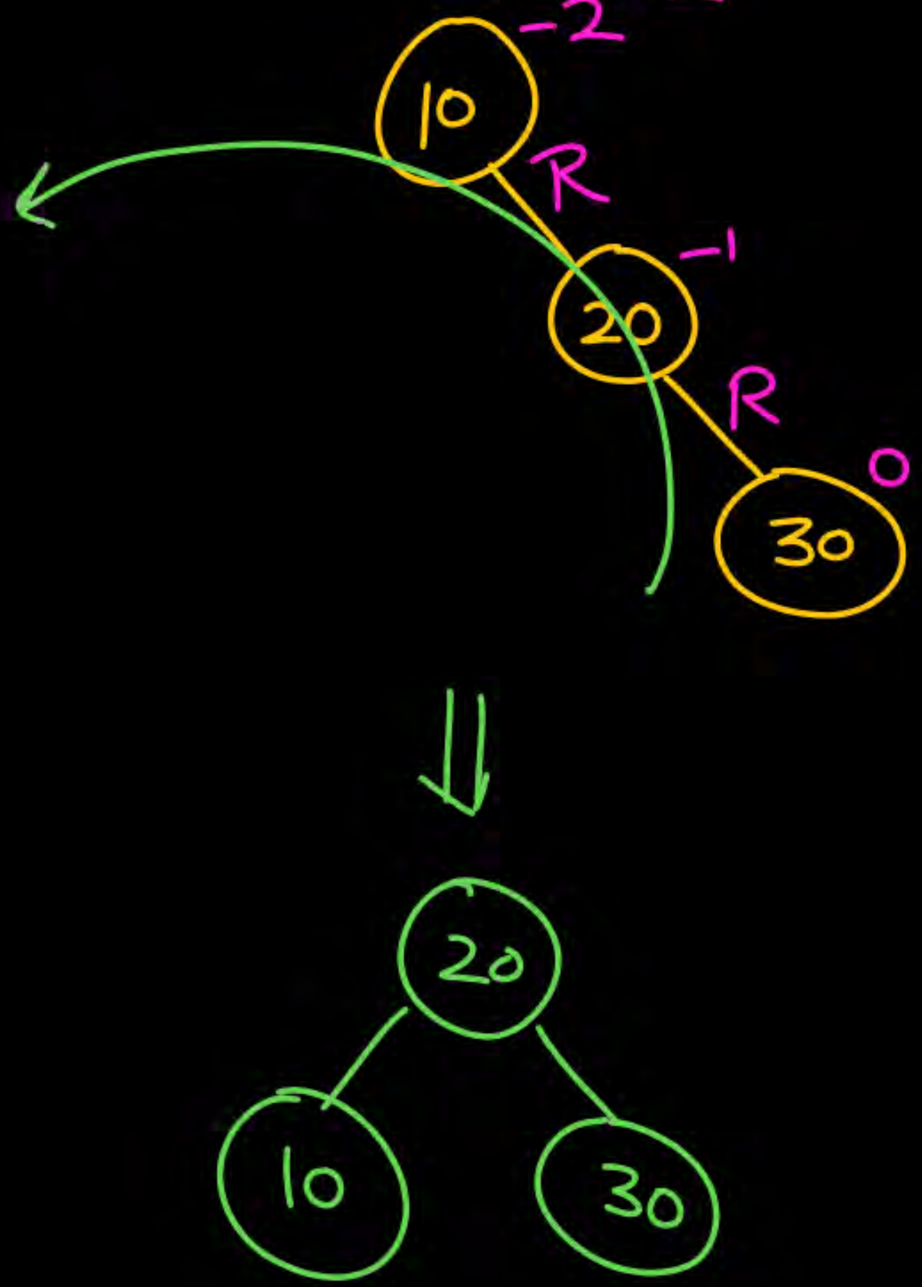
BST ✓

AVL X

Insertion of key may cause Bal. factor of nodes Other than -1,0,+1 (Unbalanced)

⇓

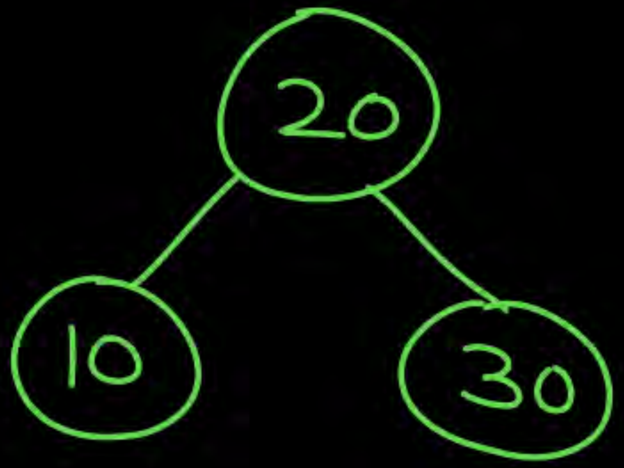
To balance the tree ⇒ rotations are performed.

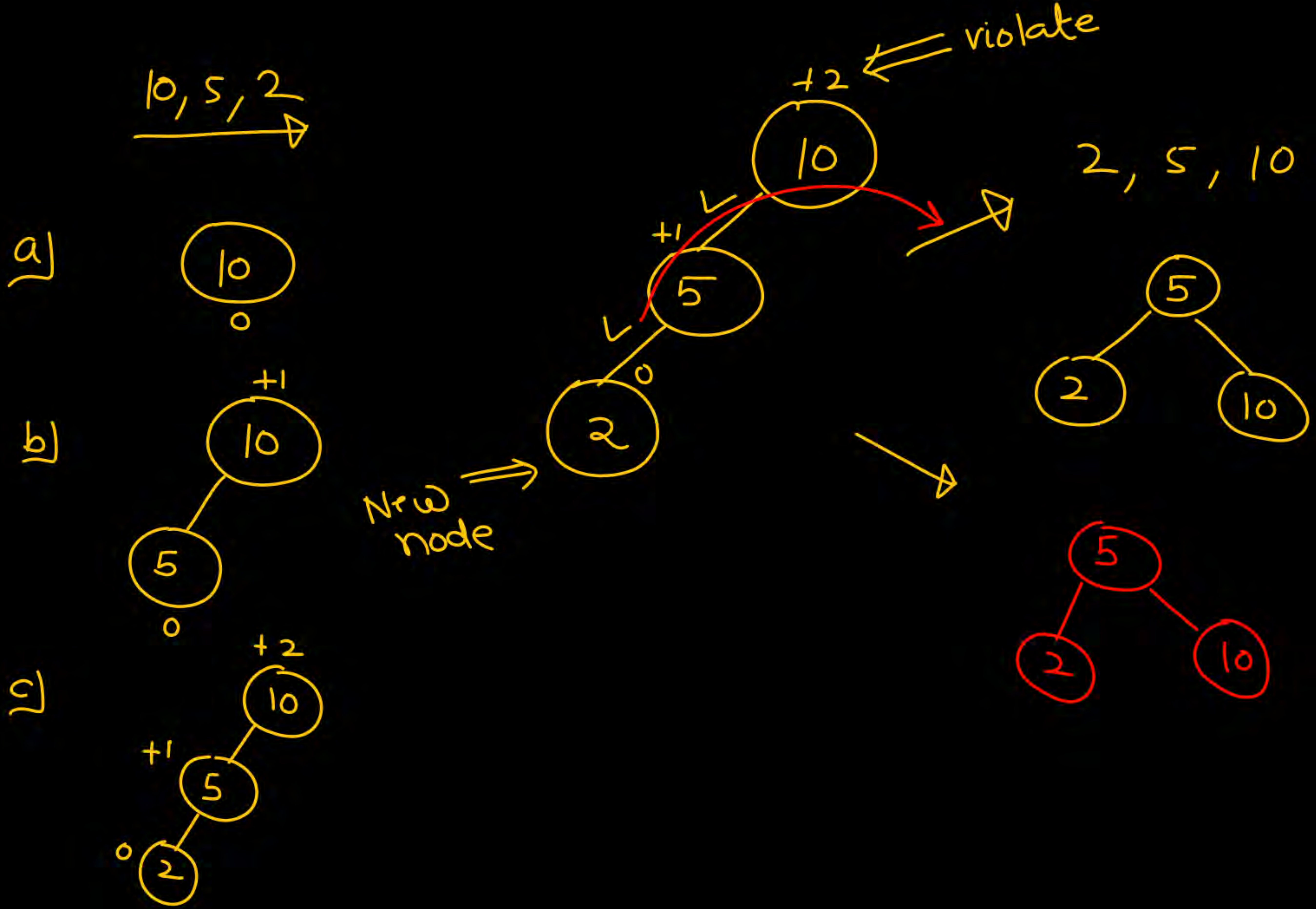


violation of AVL tree property

new node

2nd way: 10, 20, 30




10, 5, 2
a)
10
0
b)
+1
10
5
0
c)
+2
10
+1
5
0
2
New node
+2
violate
10
+1
5
0
2
2, 5, 10
5
2
10
5
2
10

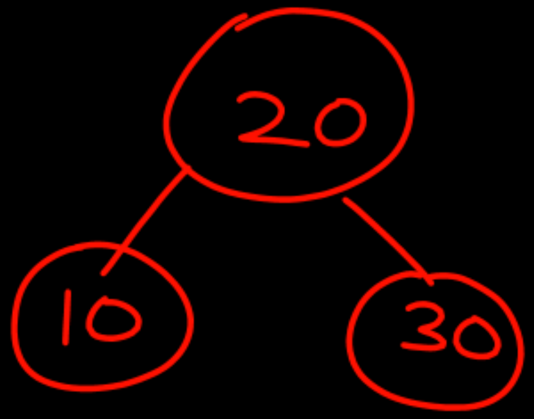
Rotations

① LL
② RR

} Single rotations

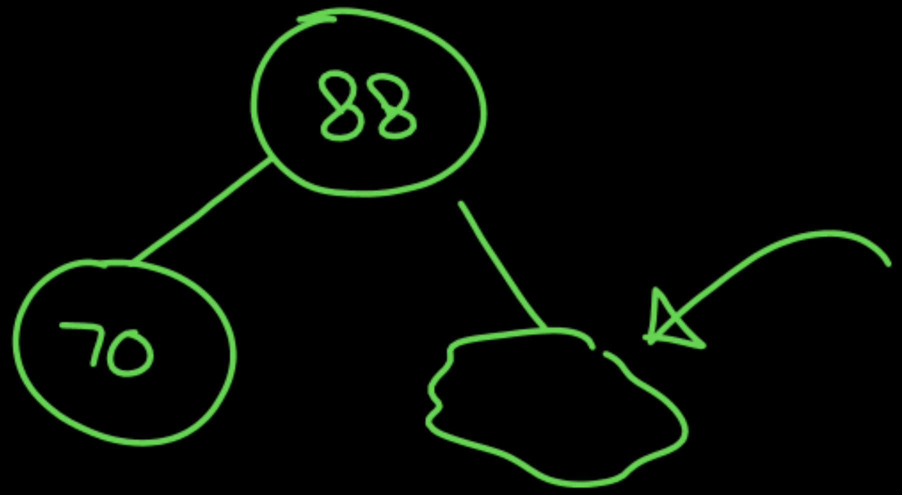
③ LR
④ R L

} double rotations



Deleting a node with 2-childs ⟹ deletion of node having 0-child / 1-child.

Claim: Node with maximum key, can have atmost one child



⟹ leaf node

OR

it can have left child

but it can not have right child.

THANK YOU GW SOLDIERS !