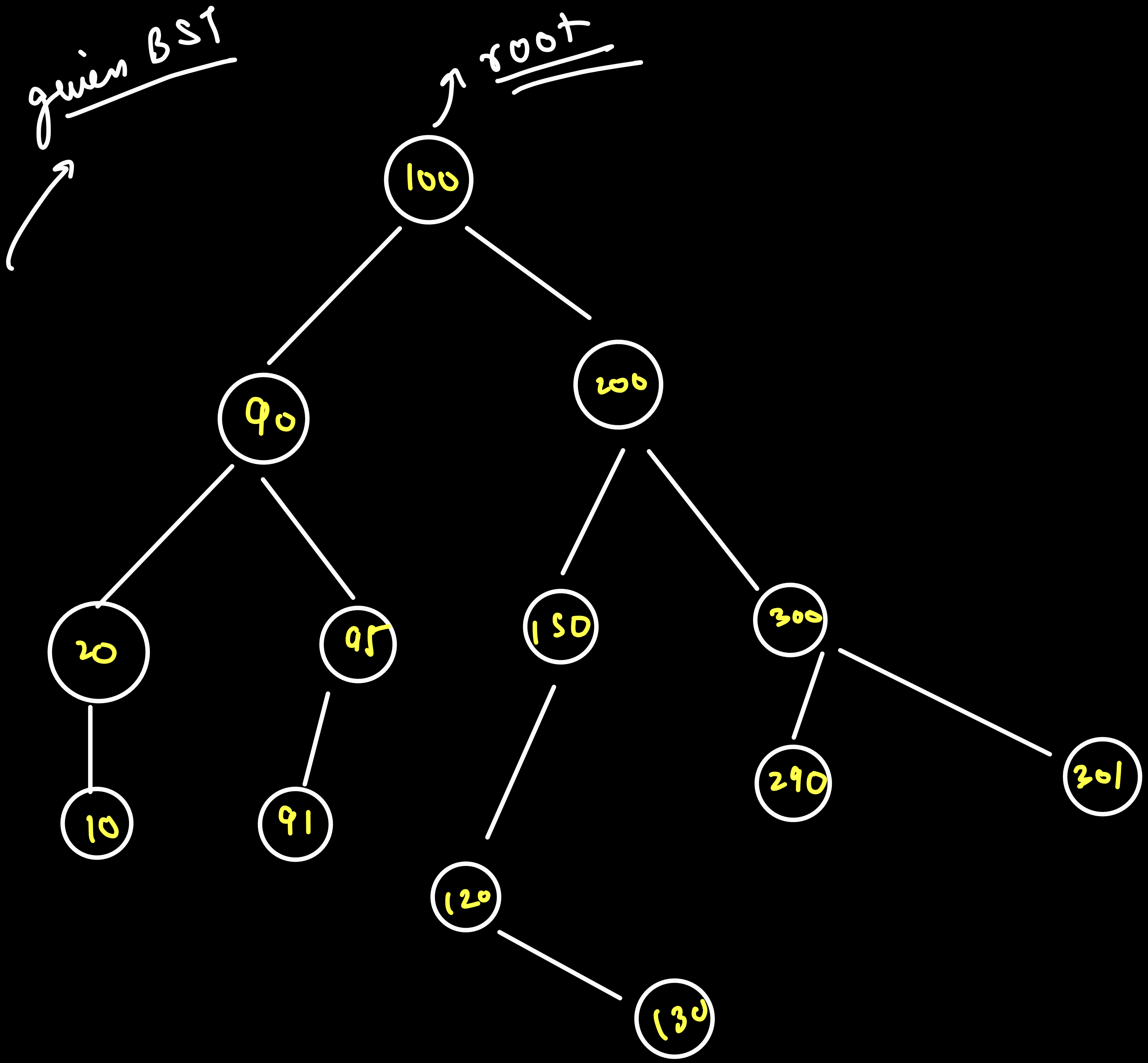
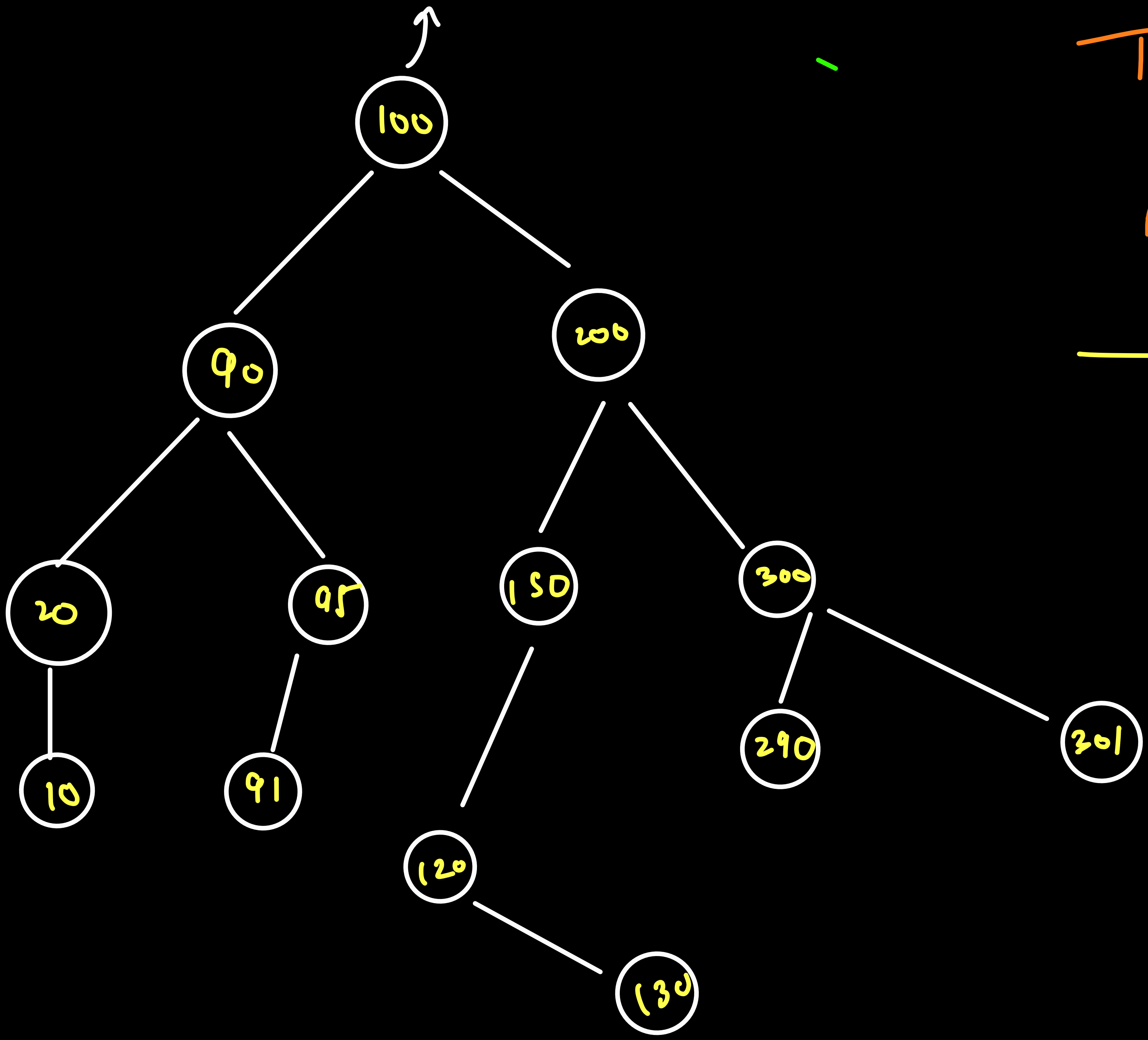

given BST
root
100
90
200
20
95
150
300
10
91
120
290
301
130



There can be 3 type of nodes to be deleted.

→ 1) leaf node

2) subtree root with one child

3) Subtree root with 2 child.

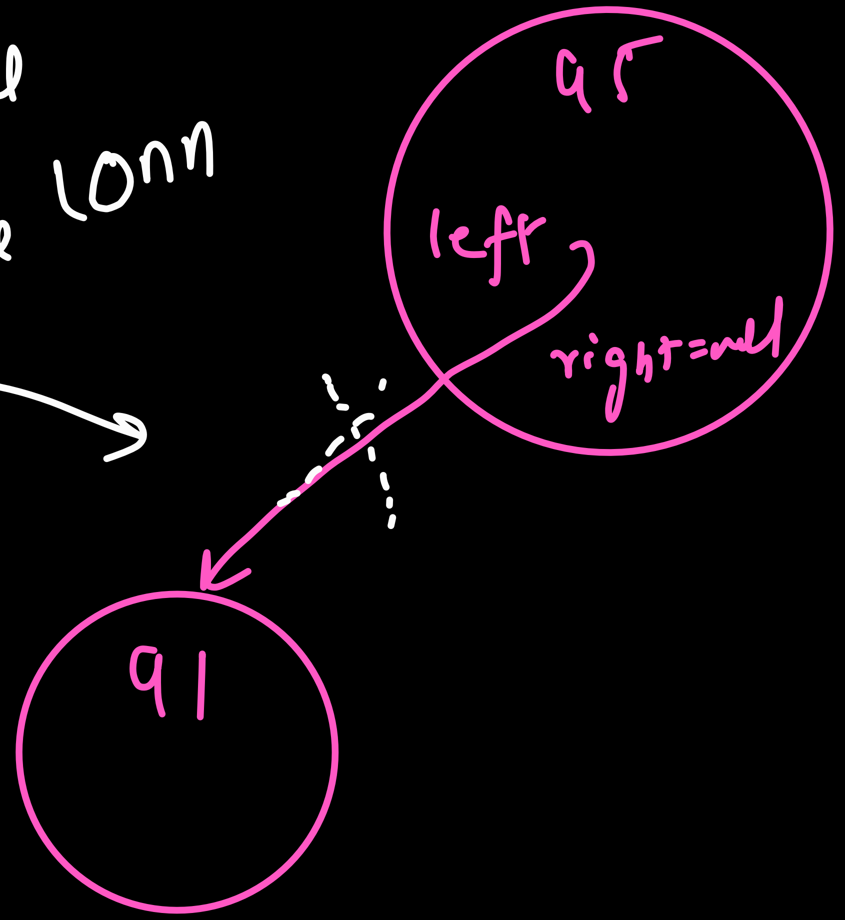
2 step procedure

1) find the node → 2) delete it.

delete (91)

How is 95 and 91 connected ?

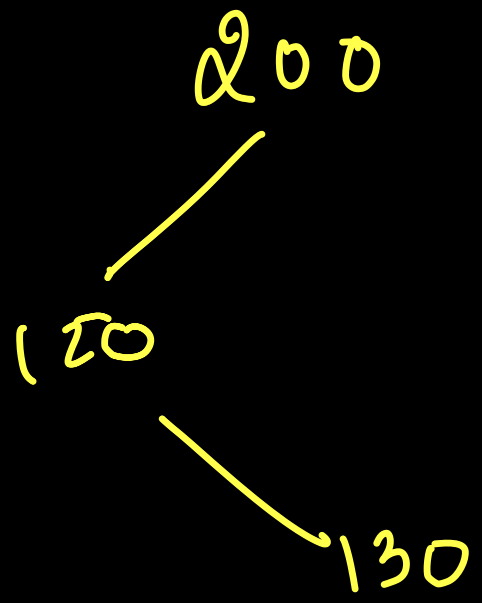
To delete 91 we have to break the conn b/w 95 & 91

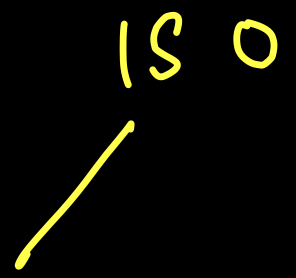


f(95)

```
if (91 < 95)
    95.left = f(95.left, 91)
```

delete (150)





f(200, 150)

if (150 < 200)

200.lft = f(200.lft, 150)

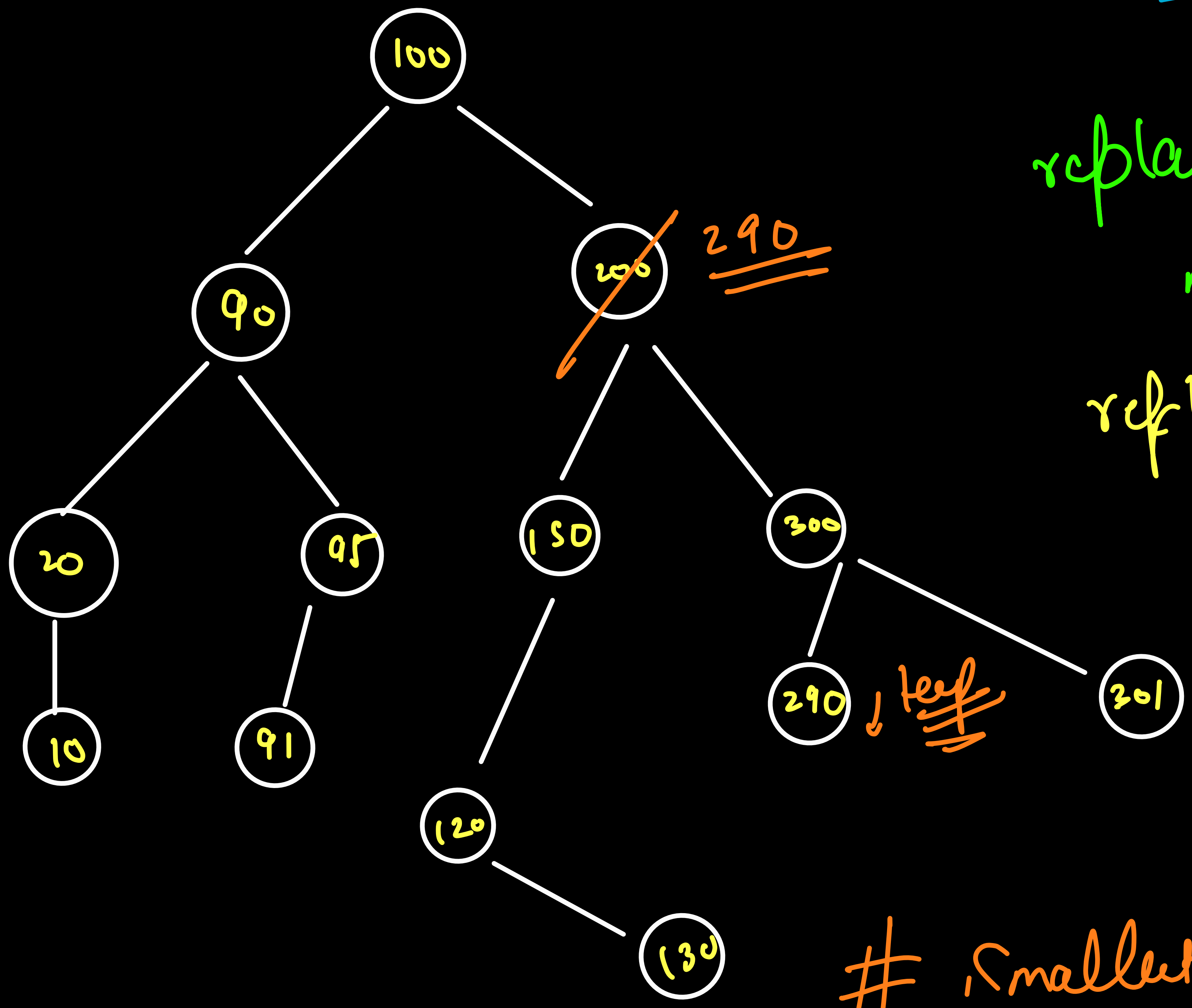
↓

f(150, 150)

delete(200)

200 ↙ ↘

replace 200 with a suitable node



replace 200 with smallest node of R.S.T. or biggest node of L.S.T.

smallest node of RST ⇒ left most node of RST

f(root, value) =

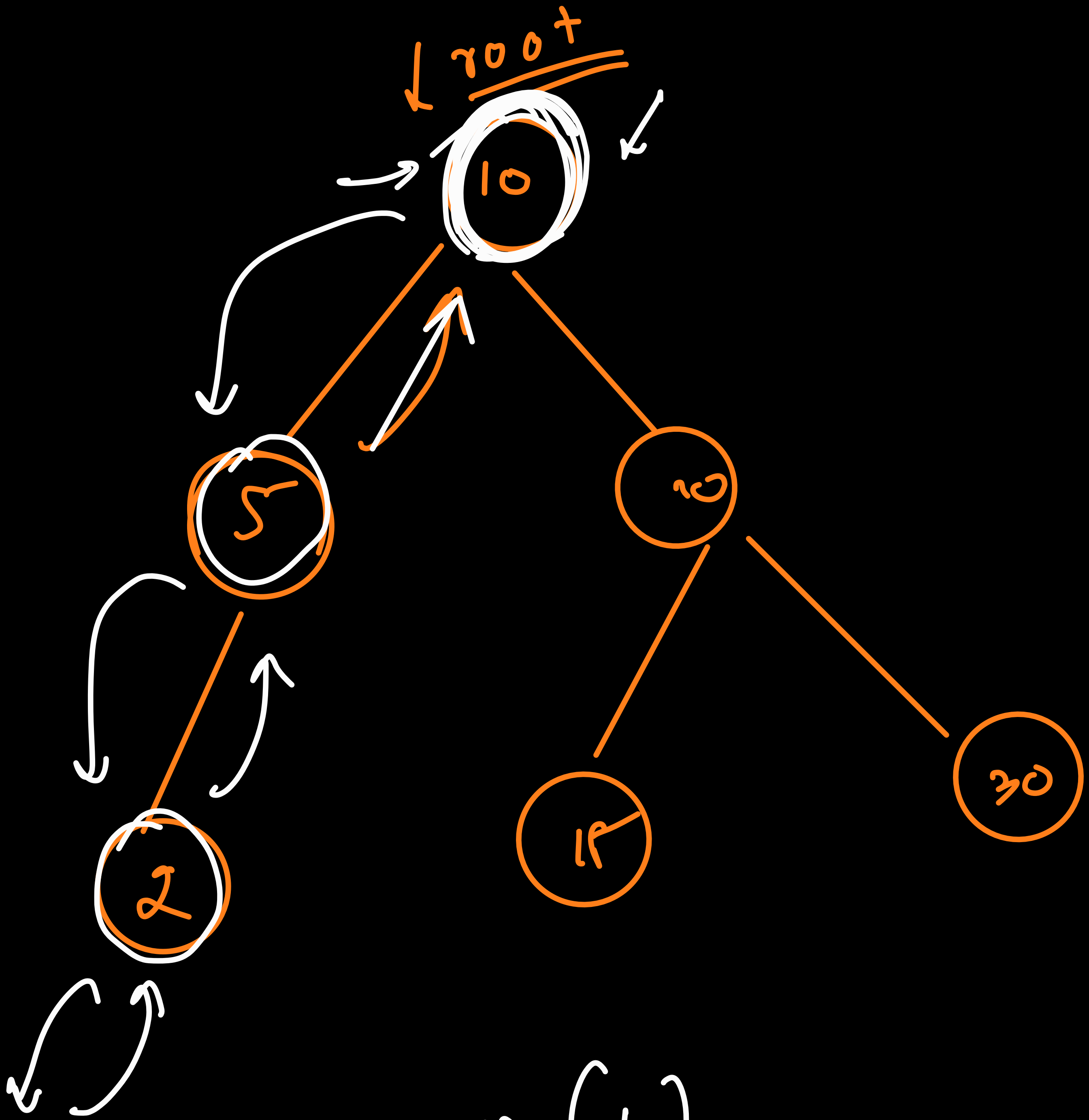
deletes the node with value from subtree of root

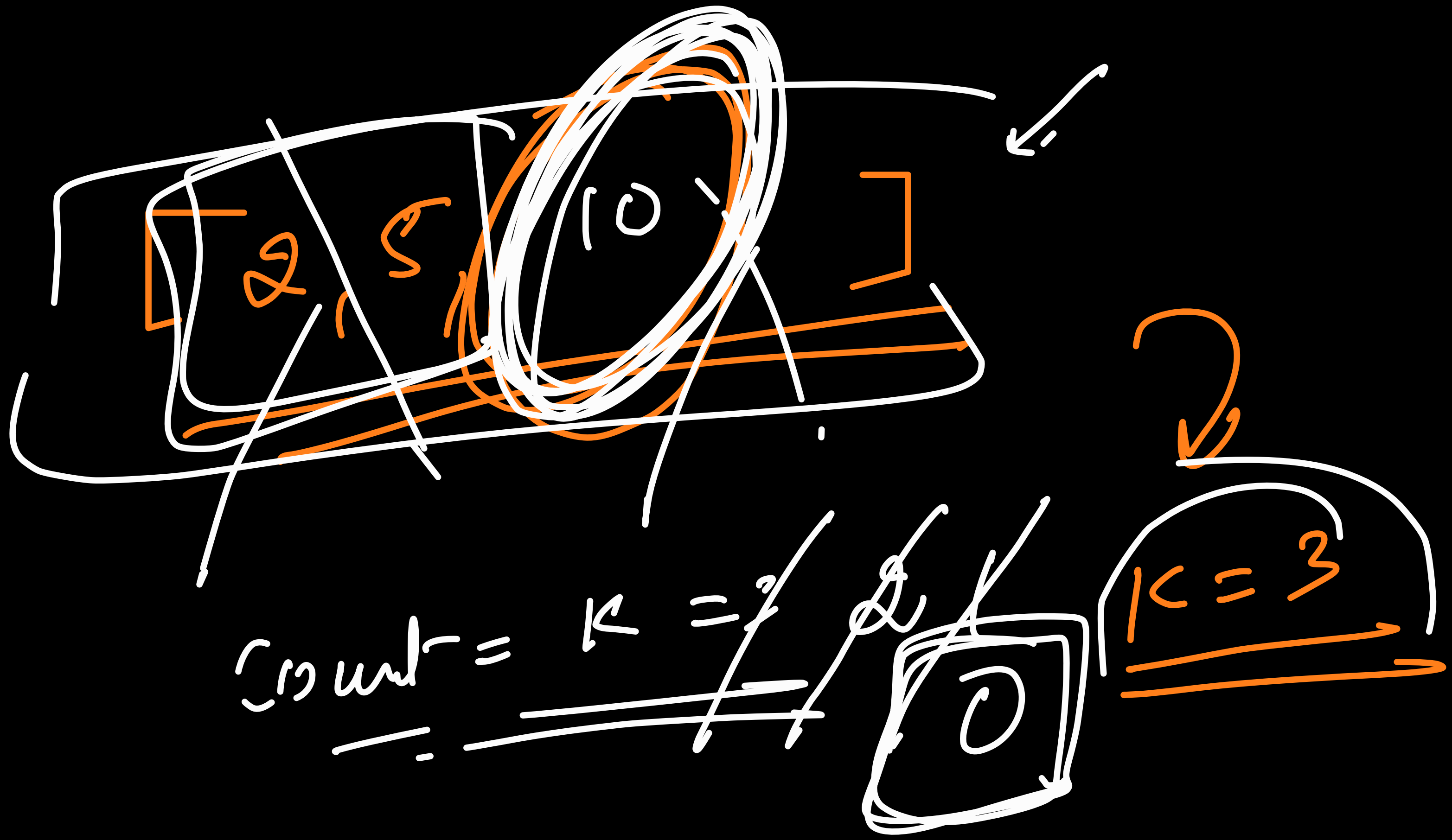
```
# Search
if (value < root.val)
    → go left
    root.left = f(root.left, value)
else if (value > root.val)
    → go right
    root.right = f(root.right, value)
else { → # node is found
    if (node.left == null && node.right == null) {
        # leaf node
        return null;
```

```
else if( node. left == null) {
        retun node. right;
} else if (node. right == null) {
        retun no. de. left;
} else {
    # 2 Subtrees)
    temp = node. right
    while (temp. left != null)
        temp = temp. left.
    node.val = temp. value
    node.right = f( node. right, temp. value)
```

return node;

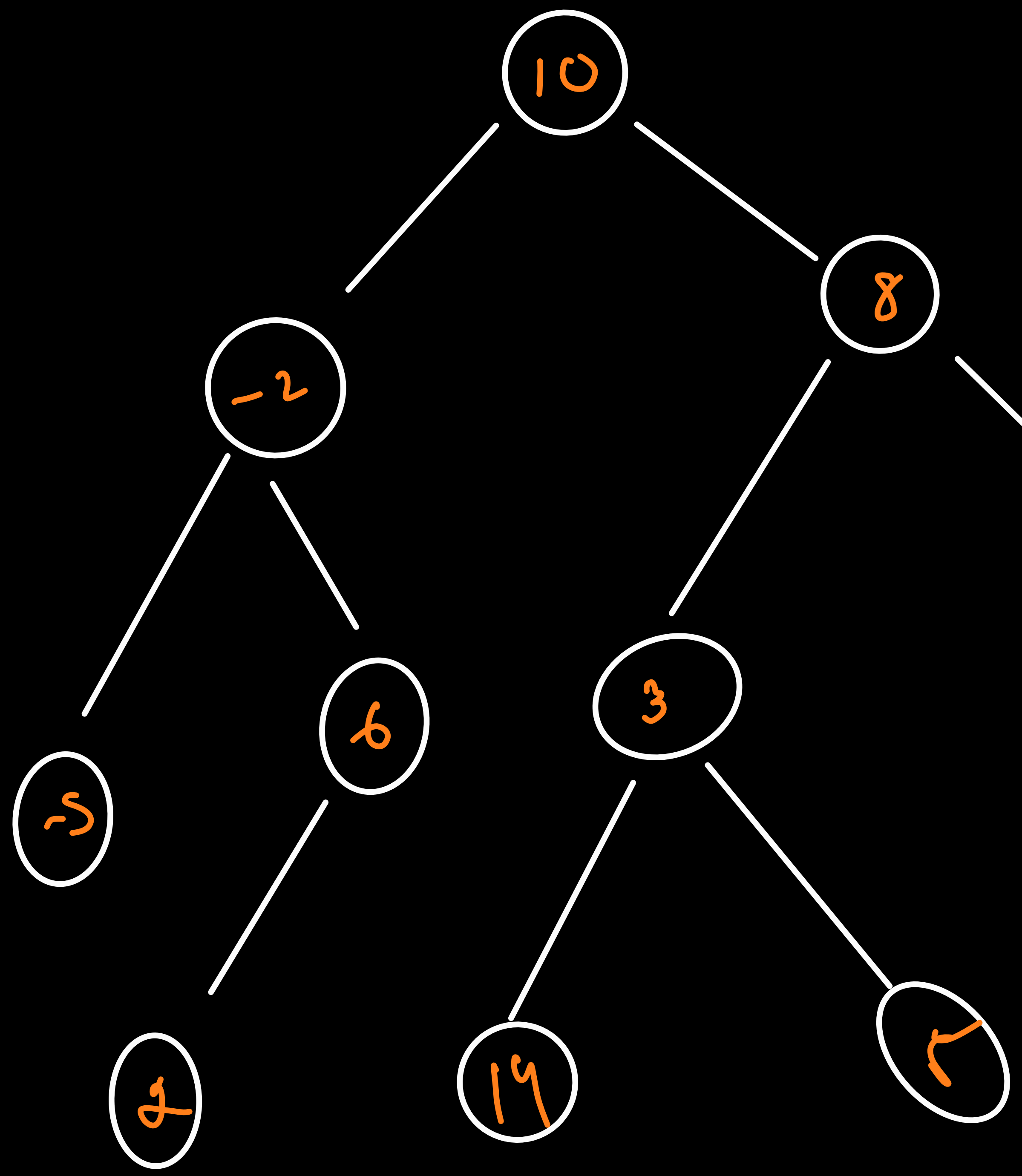
}

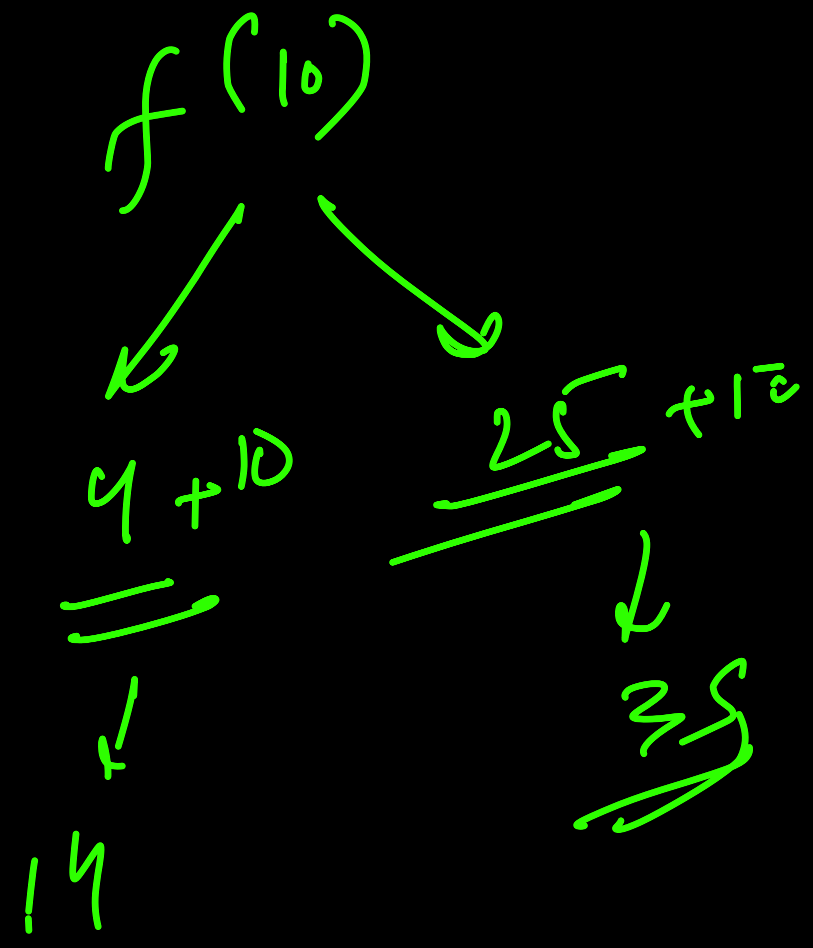




count --

O(n)





14, 25, 10, 39

14+35-10

25+14

39

f(root) =
↓

- leftbs = f(root.left) + root.val
- rightbs = f(root.right) + root.val

max path sum → ans = max(root.val, leftbs, rightbs,
leftbs + rightbs - root.val)
↑

return max(leftbs, rightbs, root.val)

f(root) =

↓

max branch sum or max sum from root to leaf

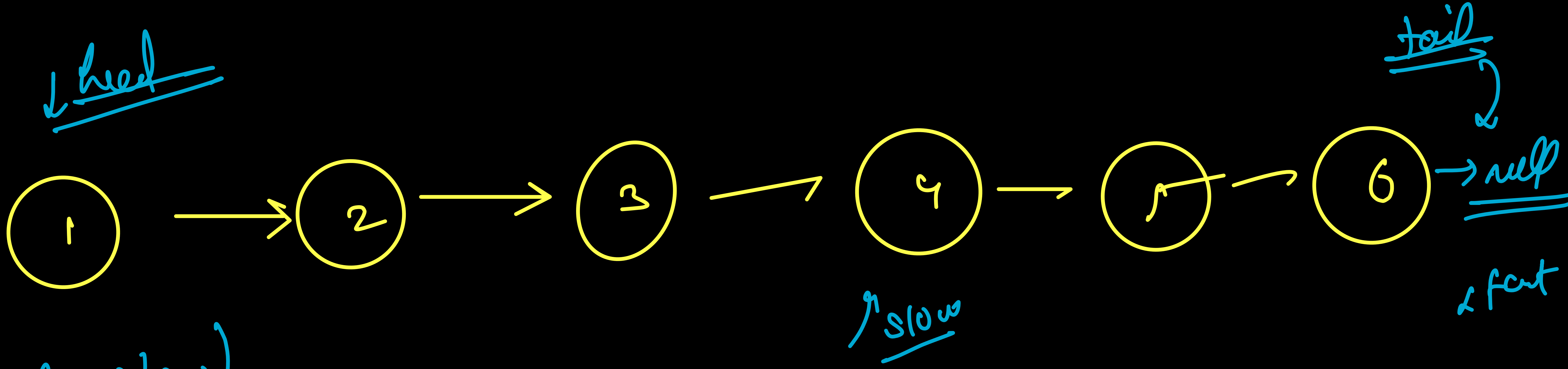
ans ← max path sum

lbs = f(root.left) + root.val

rbs = f(root.right) + root.val

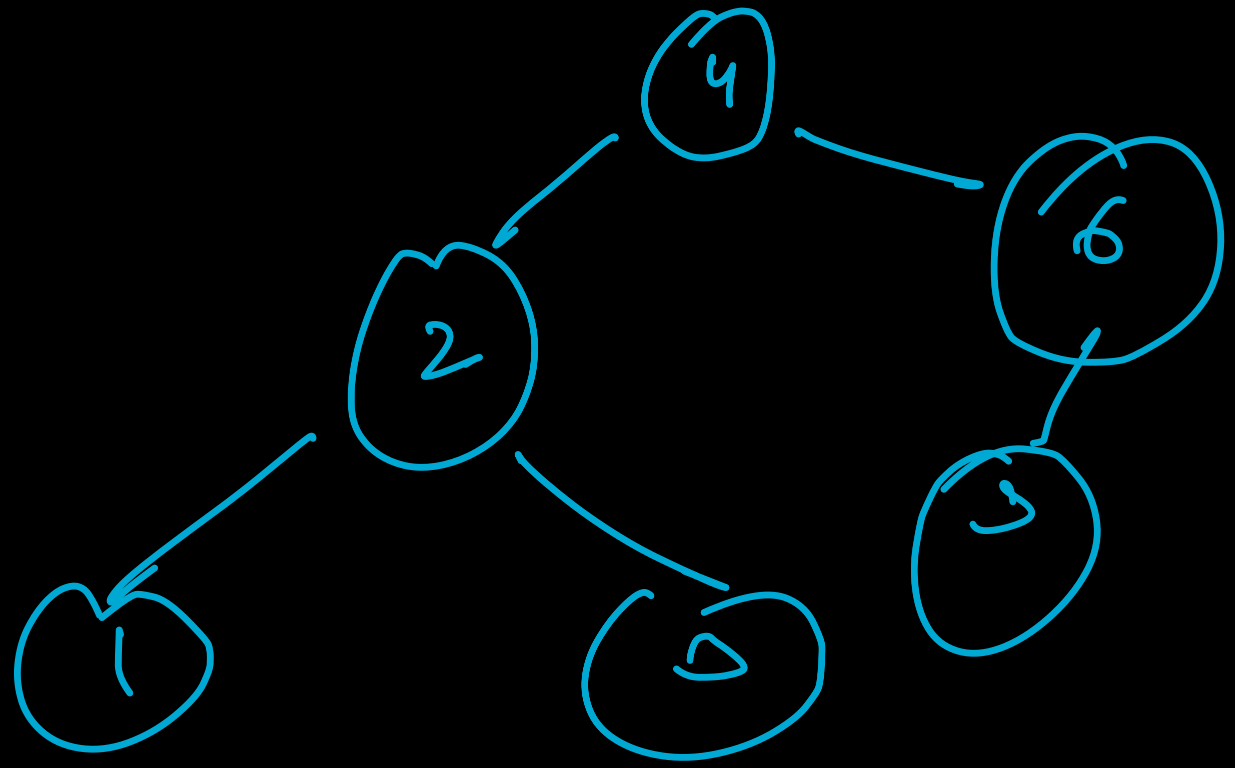
ans = max(ans, lbs, rbs, lbs+rbs - root.val)

return max(leftbs, rightbs, root.val)



f(head, slow)
f(slow.next, tail)
f(head, tail)
↓

create a BB.ST.
from linked list
starts from head node
to tail node.



```
f(head, tail) = {
    midnode = calcmid(head)
    root = new TreeNode(midnode.val)
    root.left = f(head, midnode)
    root.right = f(midnode.next, tail)

    return root.
}
```