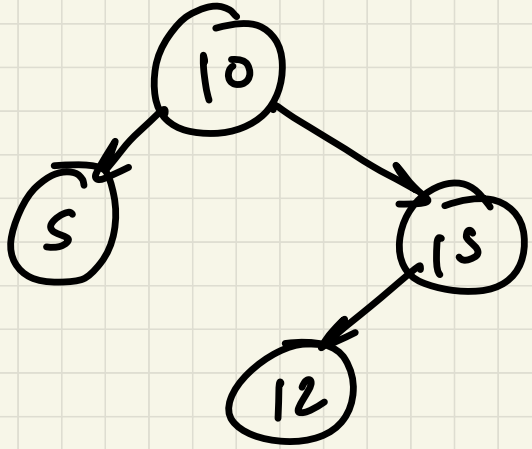
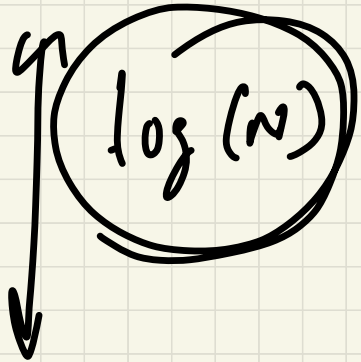
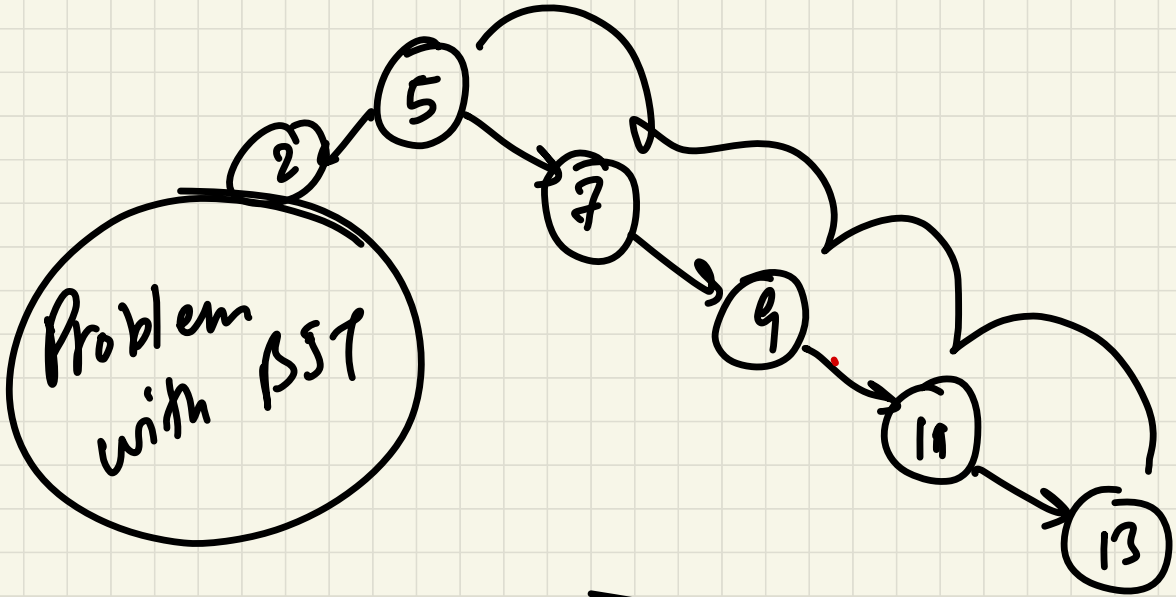
AVL

- Not confusing
- A lot of moving parts
  - very simple

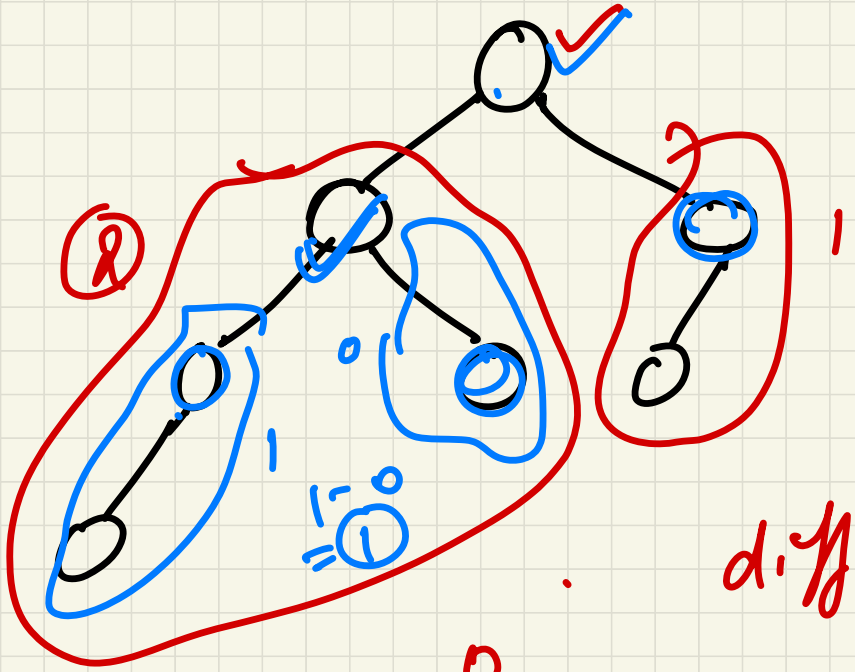






$\log(n) \times$
$O(N)$

Define the problem

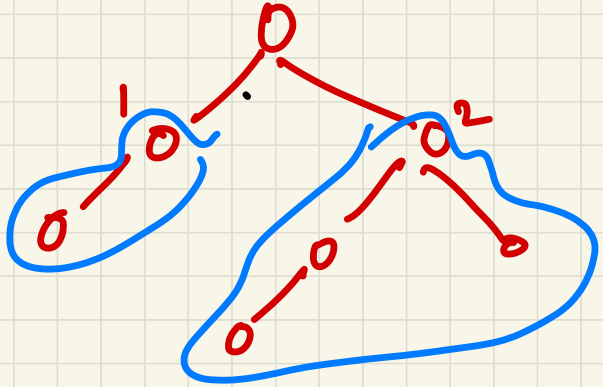


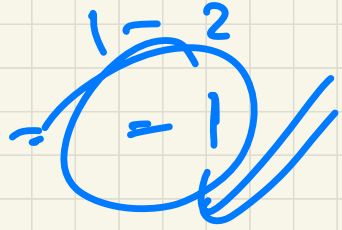
2 - 1

= 1

diff ≤ 1

1 or 0

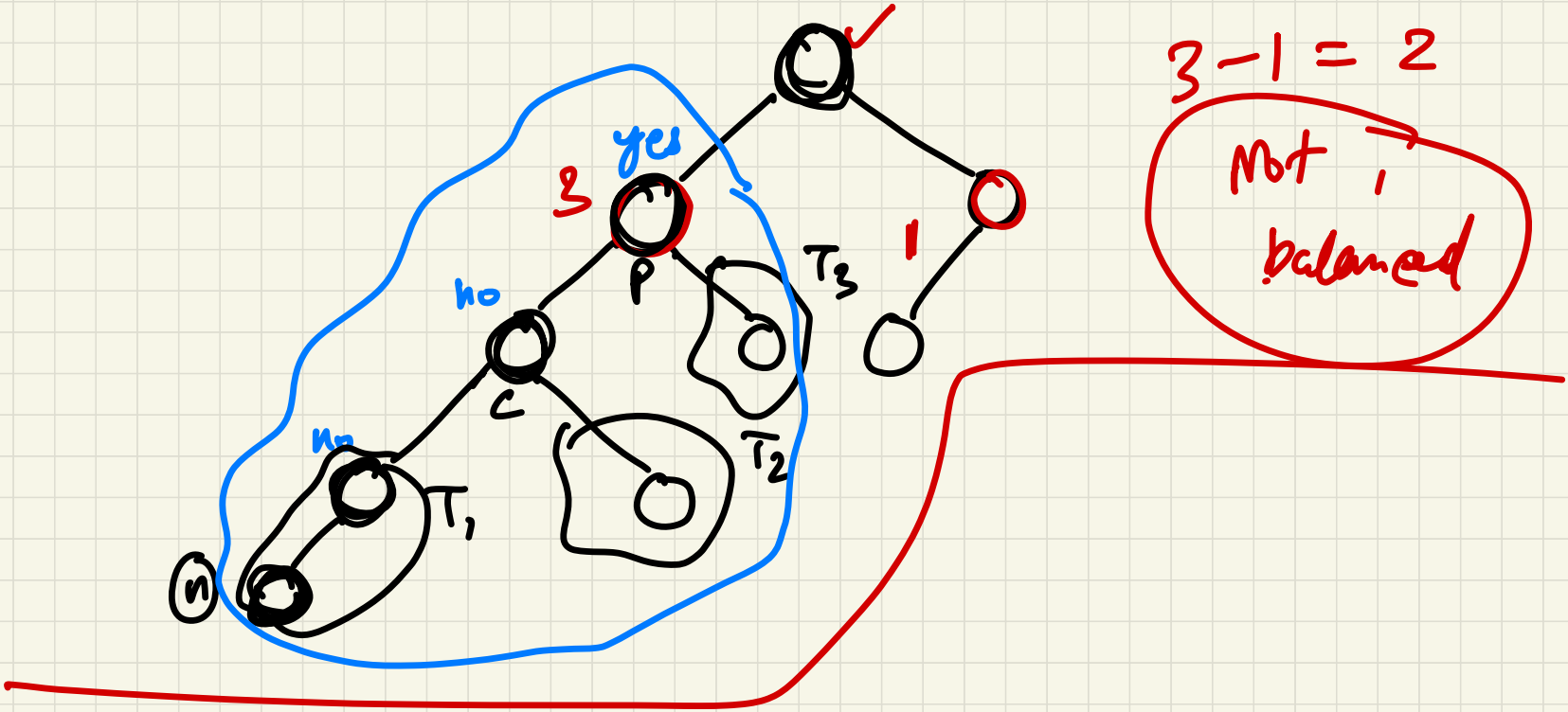




For every Node, $h(l) - h(r)$

$= -1, +1, \text{or } 0$

⇔ For every Node in the tree, the diff in height of left and right subtree of that node, $\leq 1 \Rightarrow$ Balanced tree.

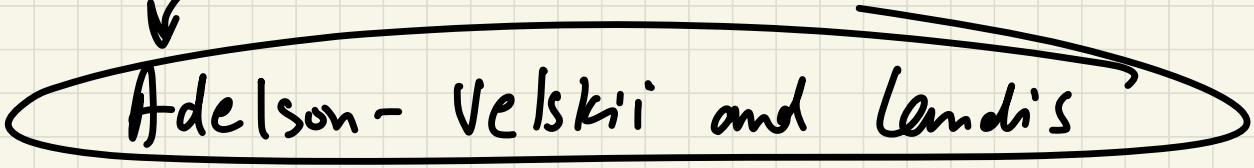


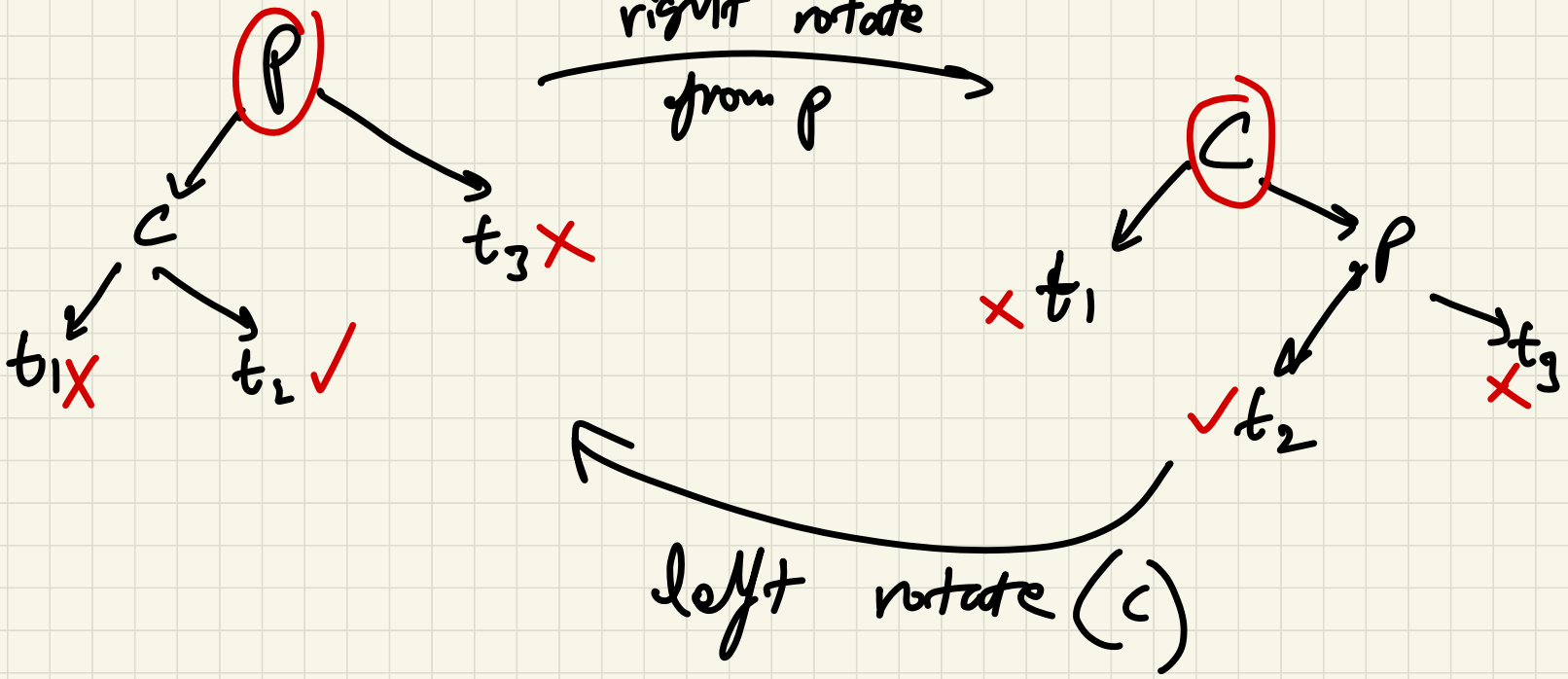
3 - 1 = 2

Not 1 balanced

Solution? Self balancing binary trees

Example: AVL



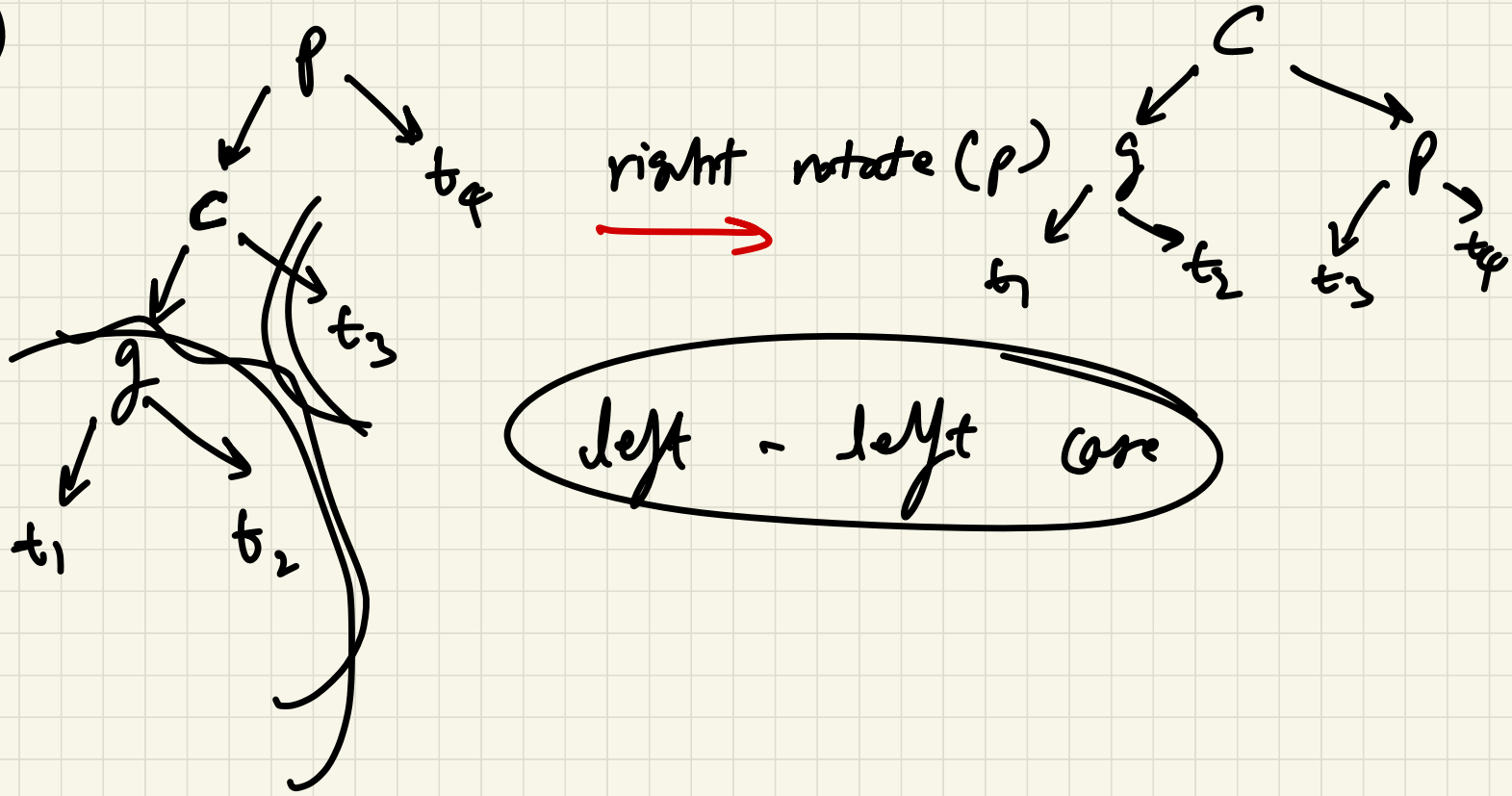

right rotate
from p
p
c
t_3
t_1
t_2
c
p
t_1
t_2
t_3
left rotate (c)

Algorithm

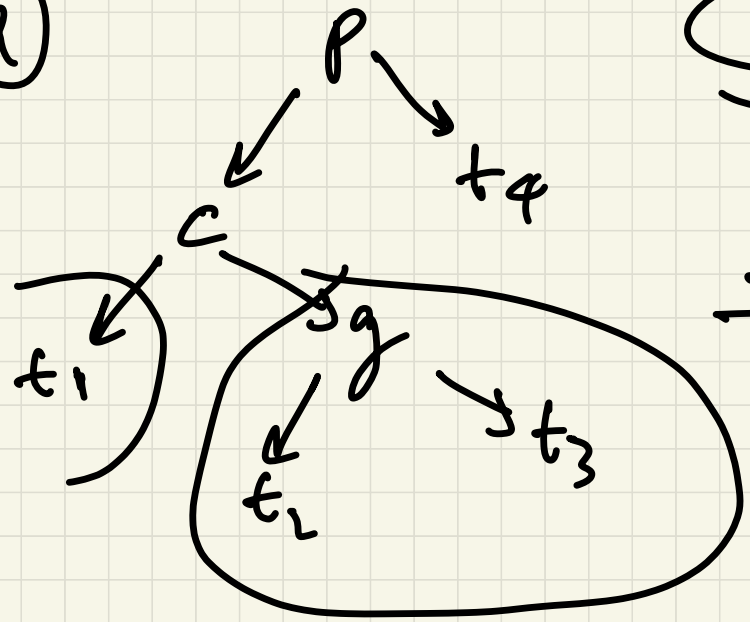
1. Insert normally, node (n)
2. Start from node (n) & find the node that makes the tree unbalanced, bottom up.
3. Using one of the (4) rules; rotate.

4 rules:

① left - left case

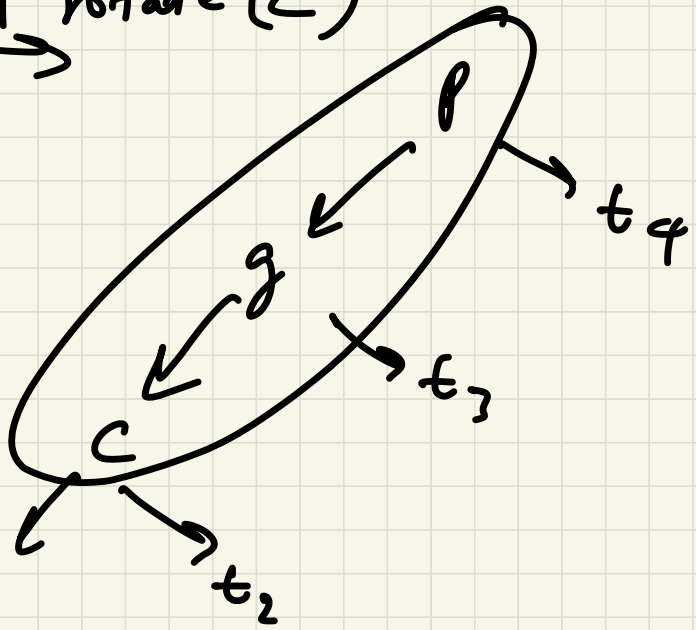


(2)



left - right case

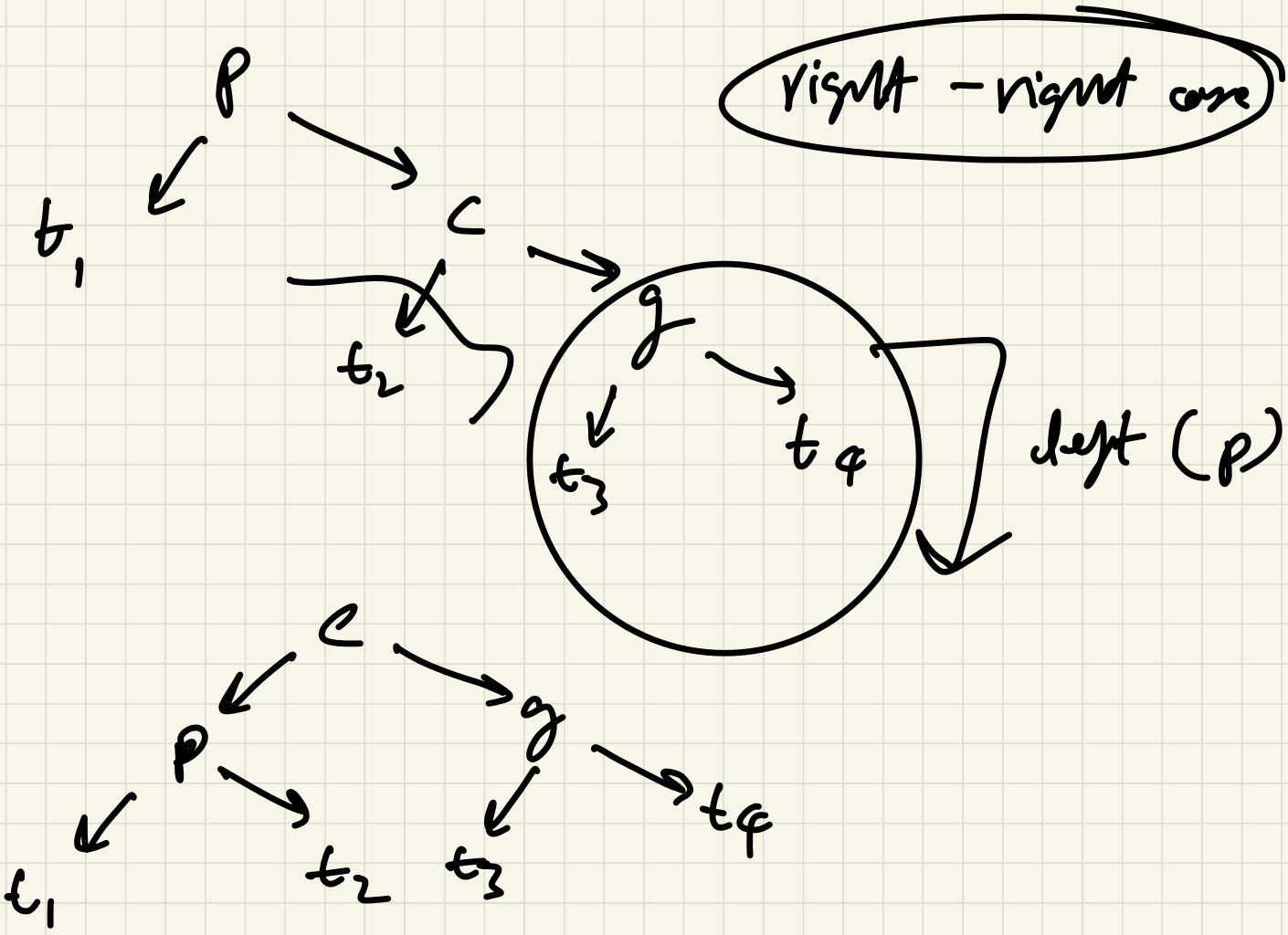
left rotate (c)



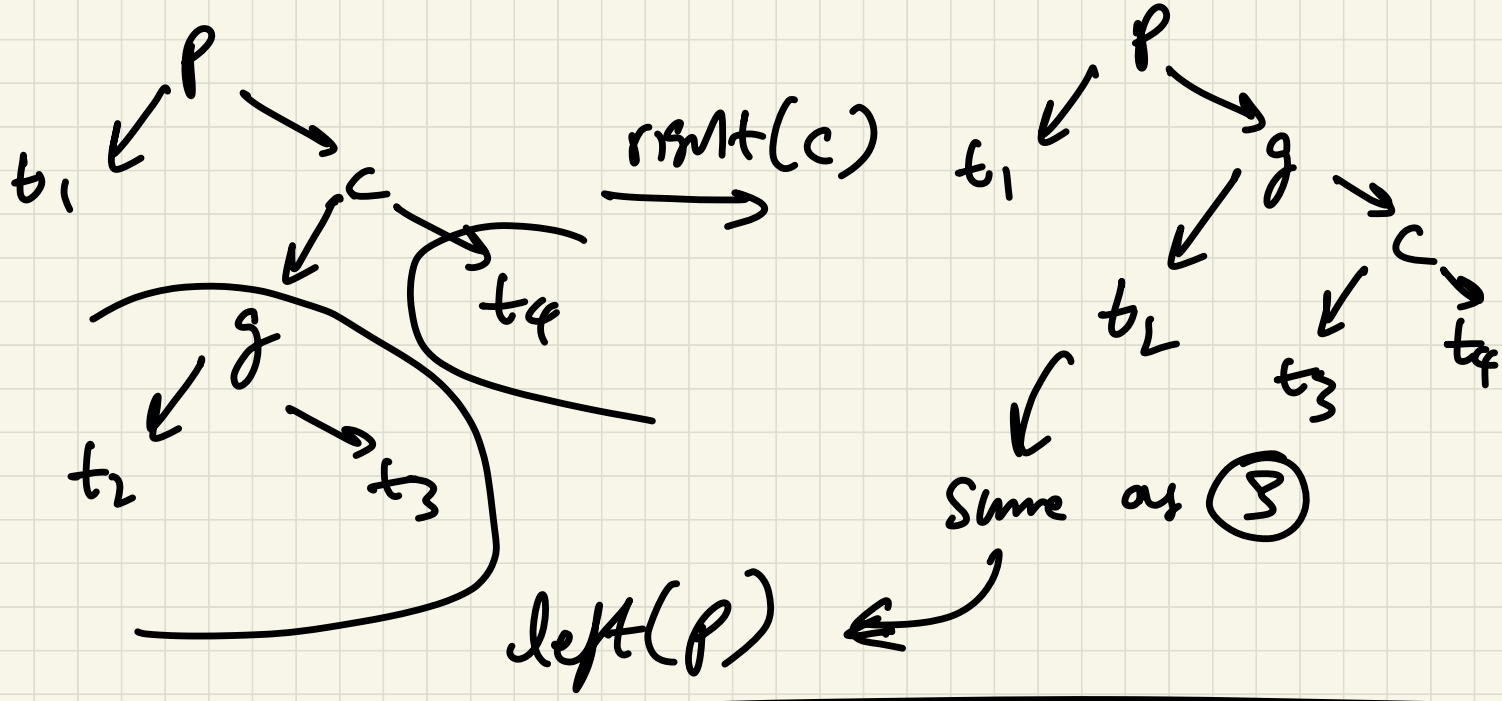
right rotate (p)

same as (1)

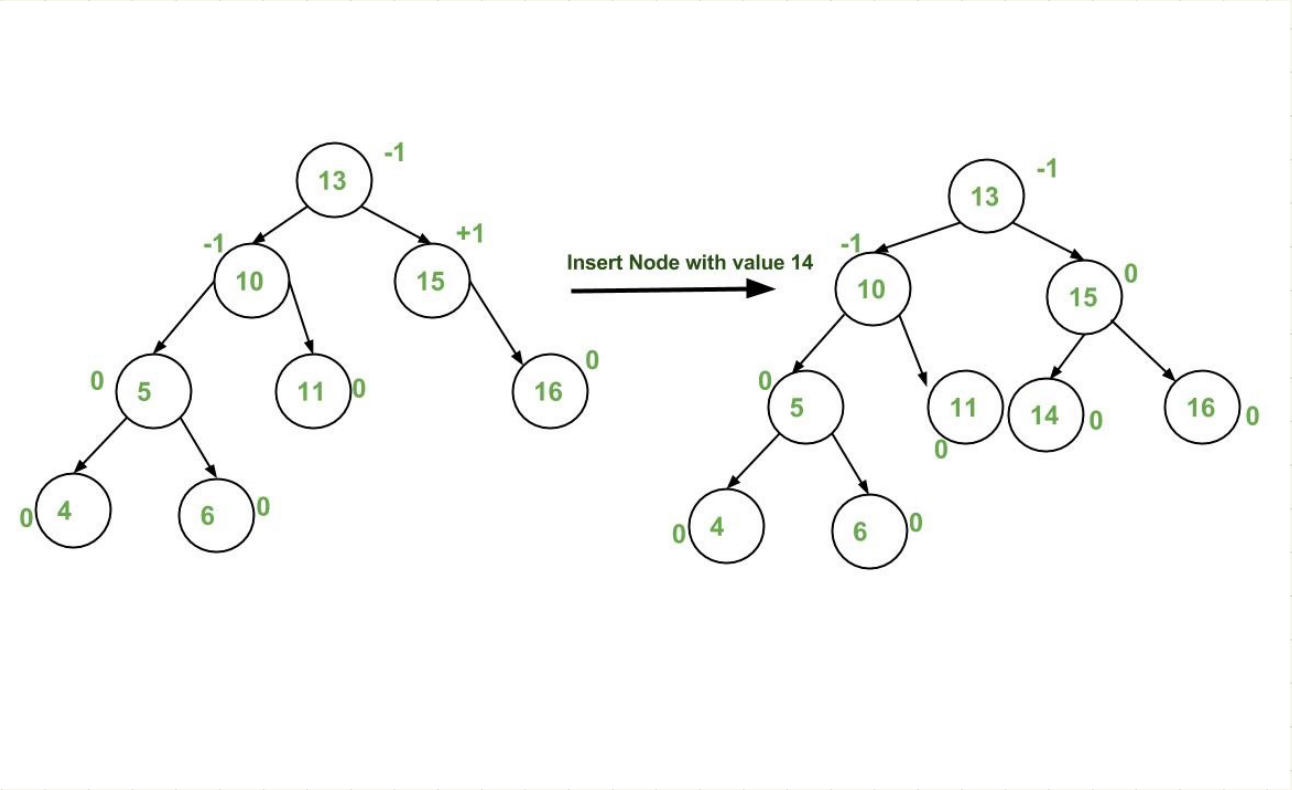
(3)

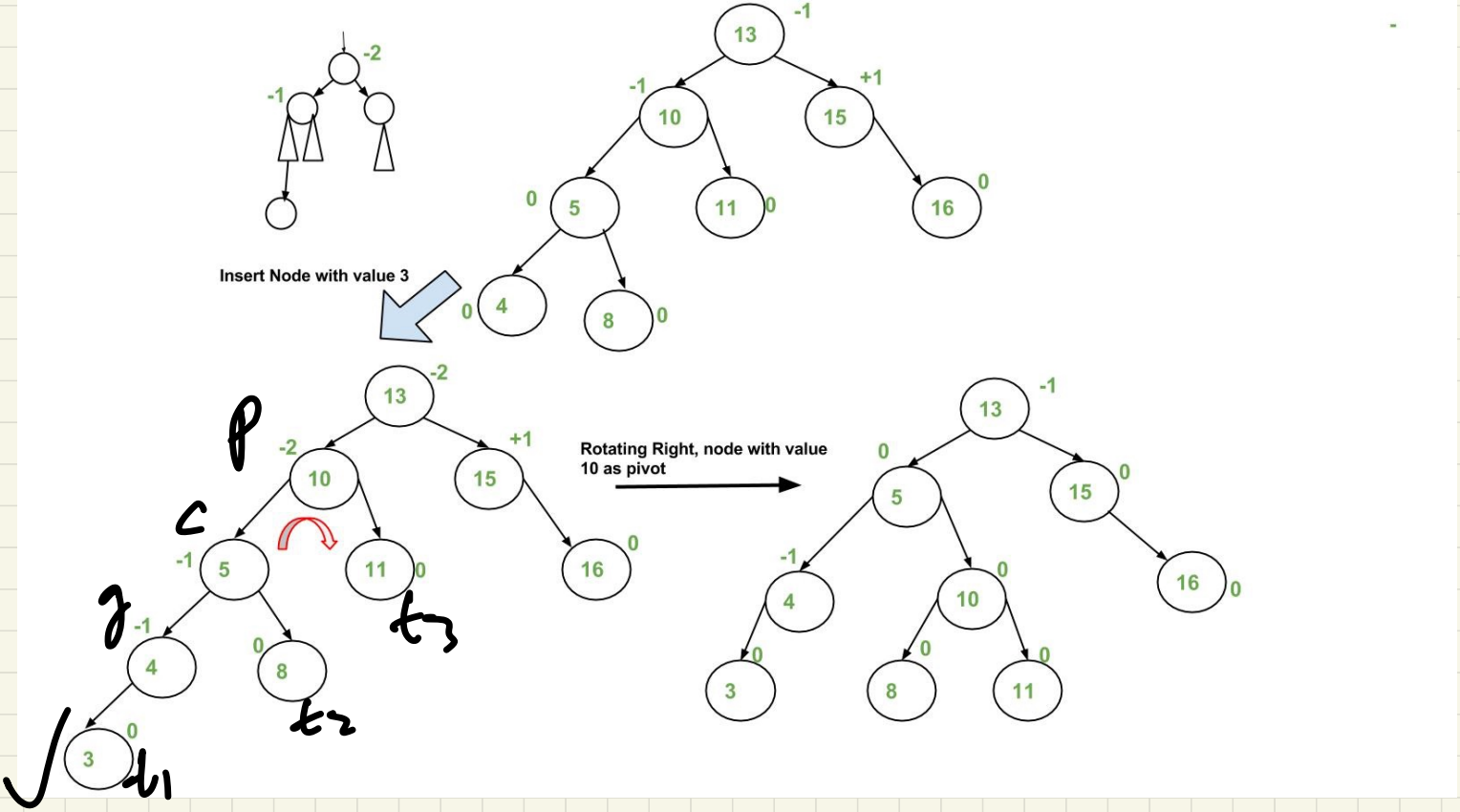


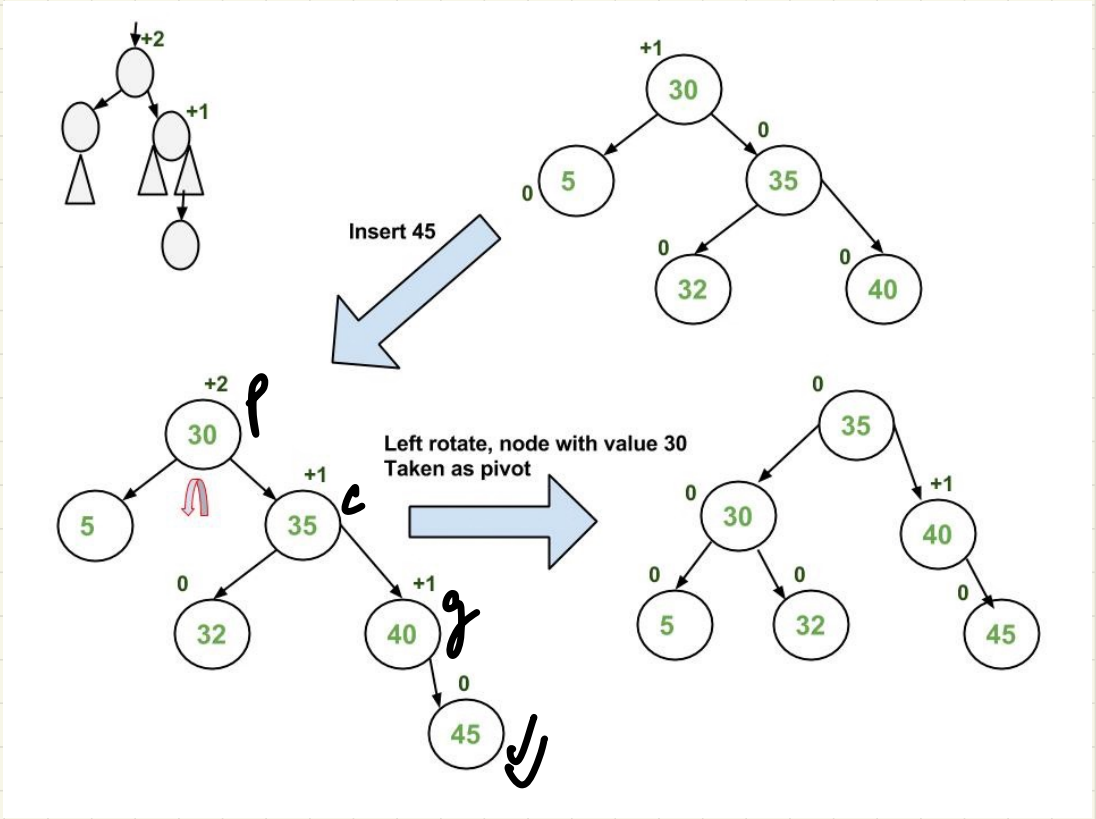
④ right - left case :

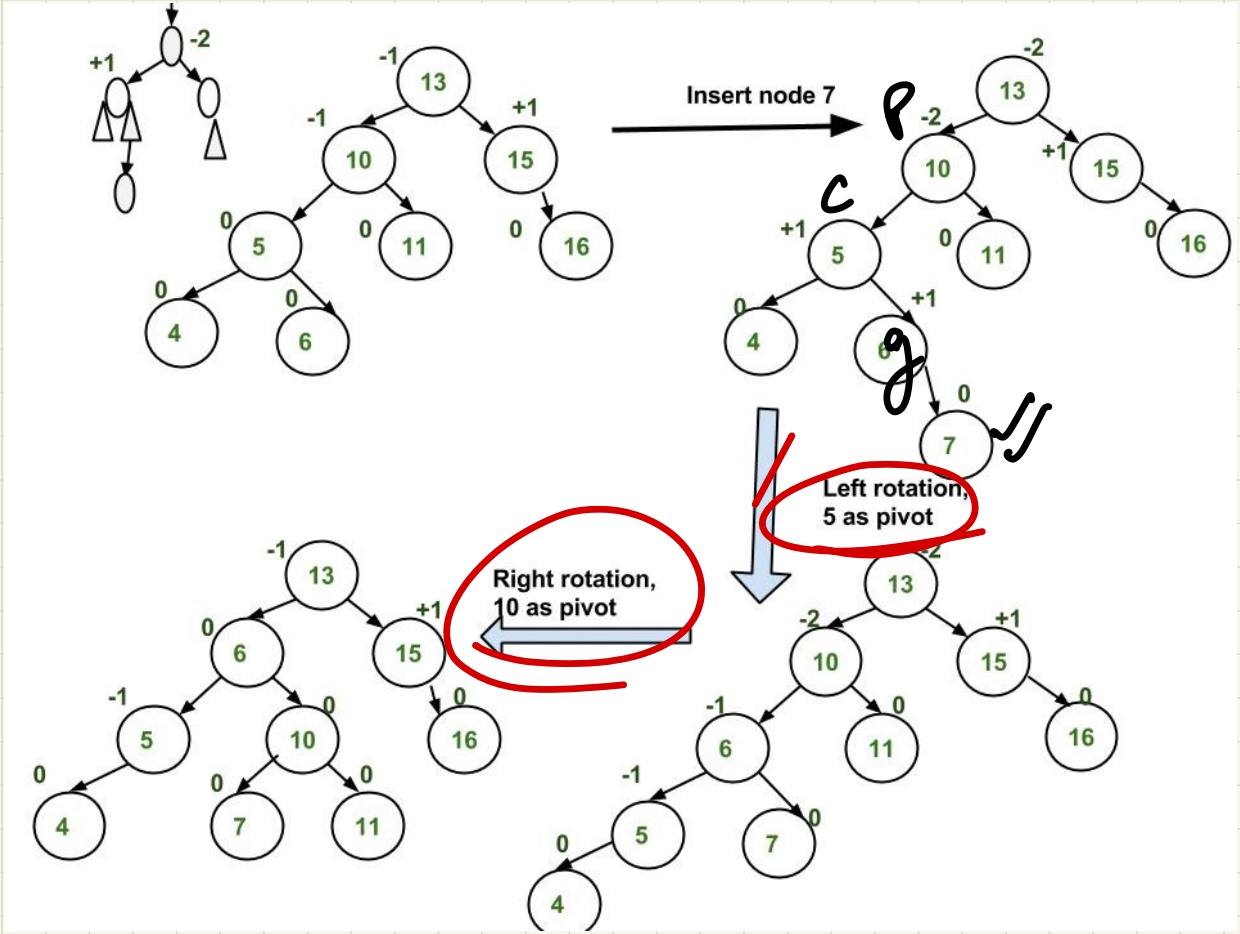


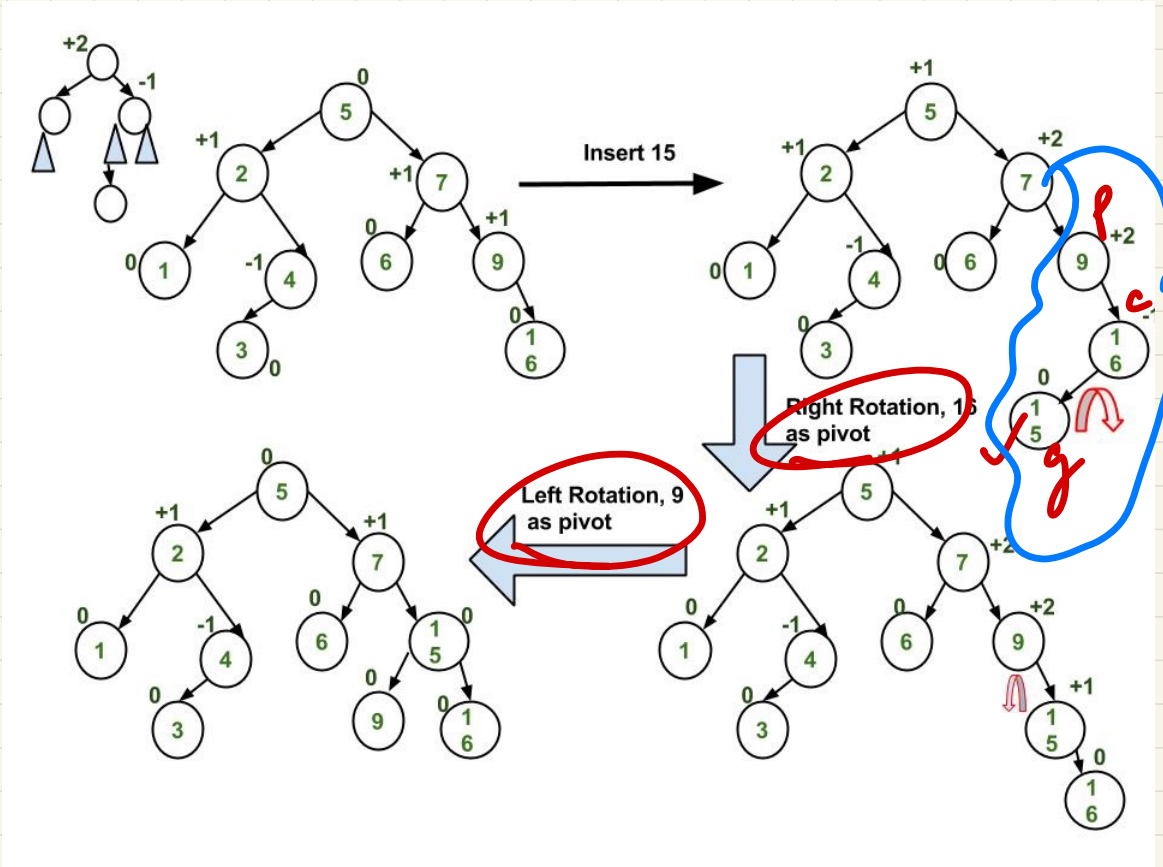
Example :


13
-1
-1
10
+1
15
0
5
11
0
16
0
0
4
6
0
Insert Node with value 14
13
-1
-1
10
0
15
0
5
11
14
0
16
0
0
0
4
6
0


-2
-1
13
-1
-1
10
+1
15
0
5
11
0
16
0
Insert Node with value 3
0
4
8
0
-2
13
p
-2
10
+1
15
c
-1
5
11
0
16
0
g
-1
4
0
8
t3
3
0
t2
t1
Rotating Right, node with value
10 as pivot
-1
13
0
5
15
0
-1
4
0
10
0
16
0
3
0
8
0
11
0


+2
+1
Insert 45
+1
30
0
5
35
0
32
0
40
+2
30
p
5
+1
35
c
0
32
+1
40
g
0
45
j
Left rotate, node with value 30
Taken as pivot
0
35
0
30
+1
40
0
5
0
32
0
45


Insert node 7
Left rotation,
5 as pivot
Right rotation,
10 as pivot


Insert 15
Right Rotation, 15
as pivot
Left Rotation, 9
as pivot
p
c
g
solve this
while tree
is balanced,
or original
whole tree
is balanced

Time complexity :

Activity

O(log n)

Answer

$\log(N) + O(1)$

your notation