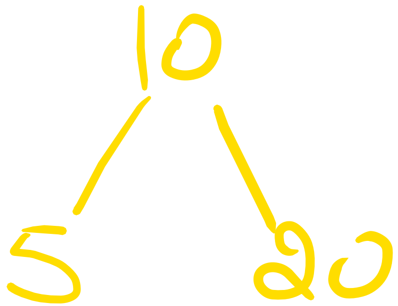
BST :->

inorder , preorder and postorder of

BST .

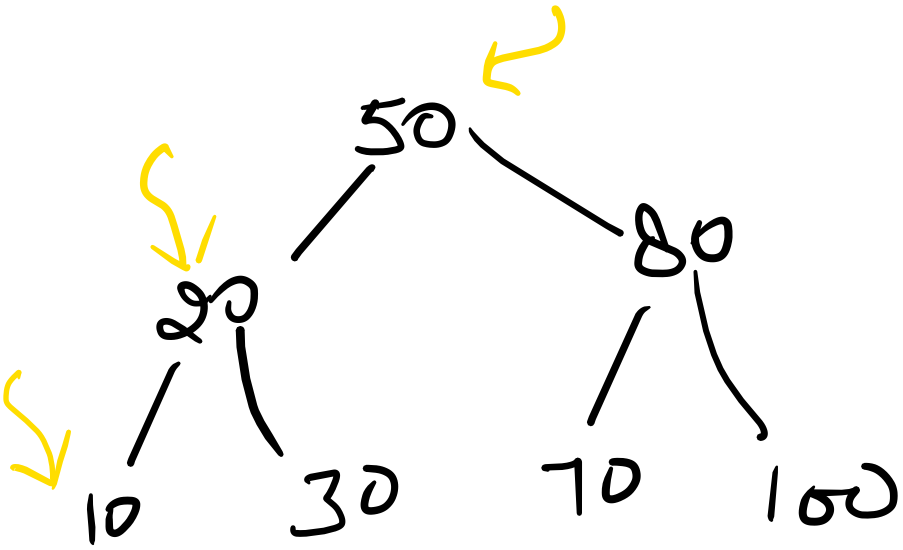


preorder :- 10 5 20

Inorder :- 5 10 20

postorder :- 5 20 10

Sorted in BST



Left, right, Pf(root)

Sorted

preorder:-

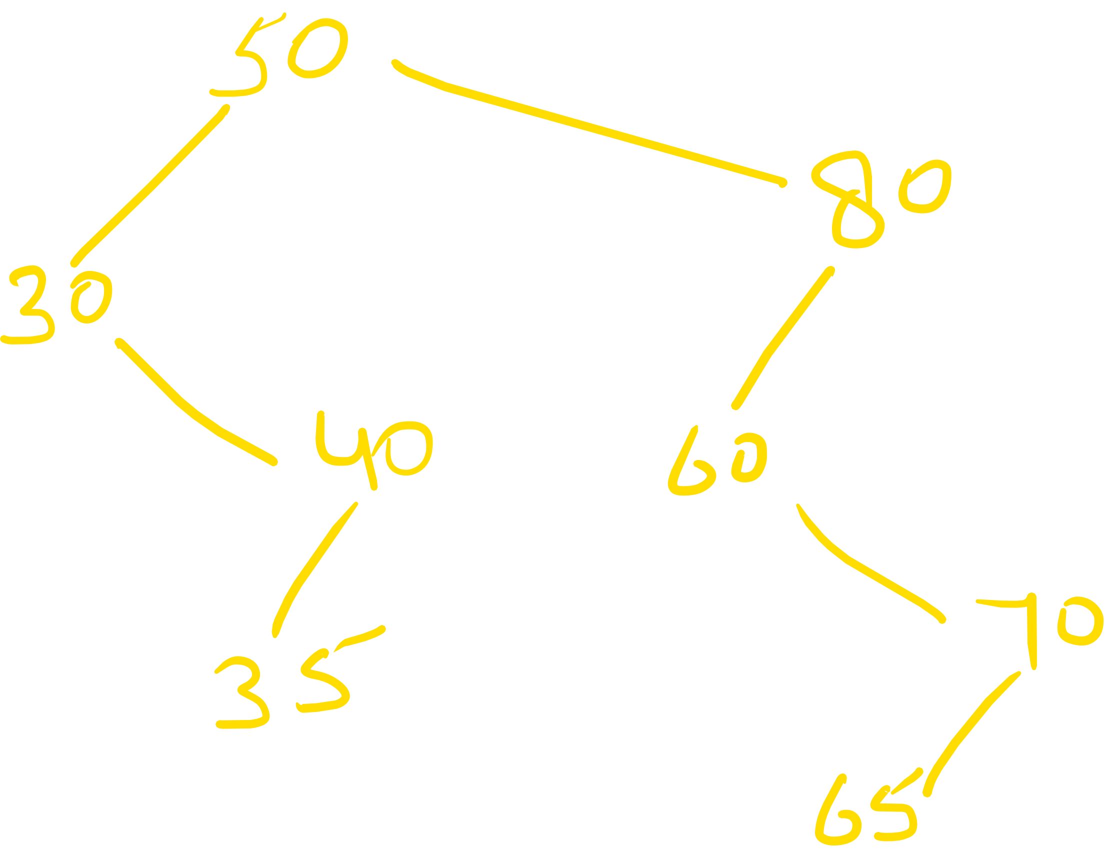
50, 20, 10, 30, 80, 70, 100

inorder:-

10, 20, 30, 50, 70, 80, 100

postorder:-

10, 30, 20, 70, 100, 80, 50



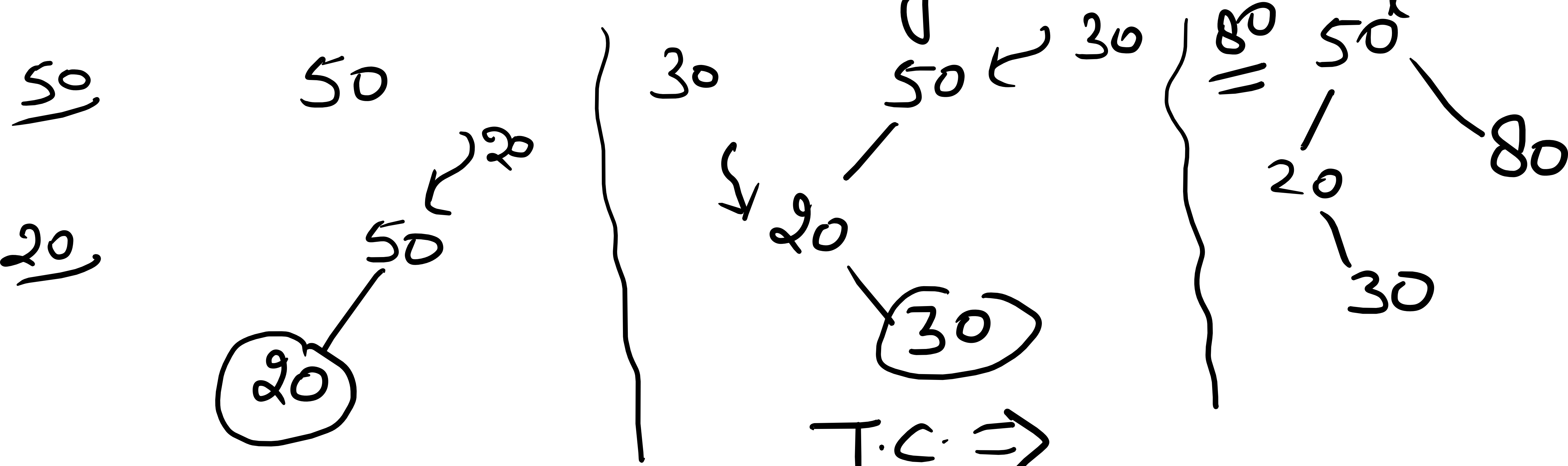
inorder 30, 35, 40, 50,
60, 65, 70, 80

preorder
50, 30, 40, 35, 80, 60, 70, 65

postorder
35, 40, 30, 65, 70, 60, 80,
50

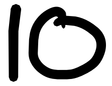
Construction of BST

element came one after another.

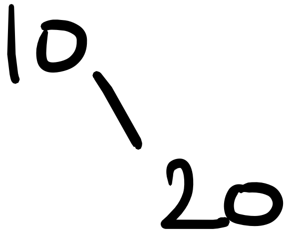


T.C. ⇒

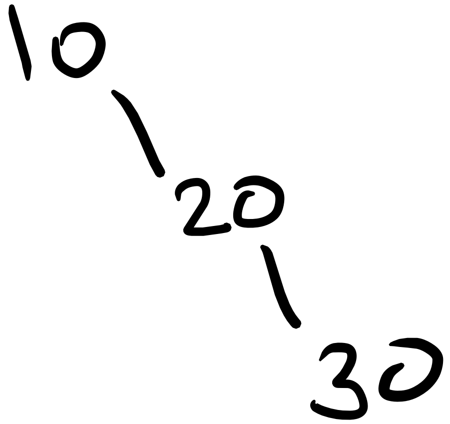
0



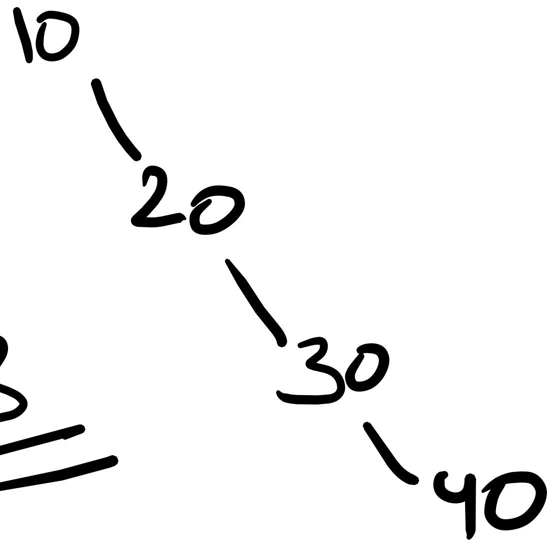
1



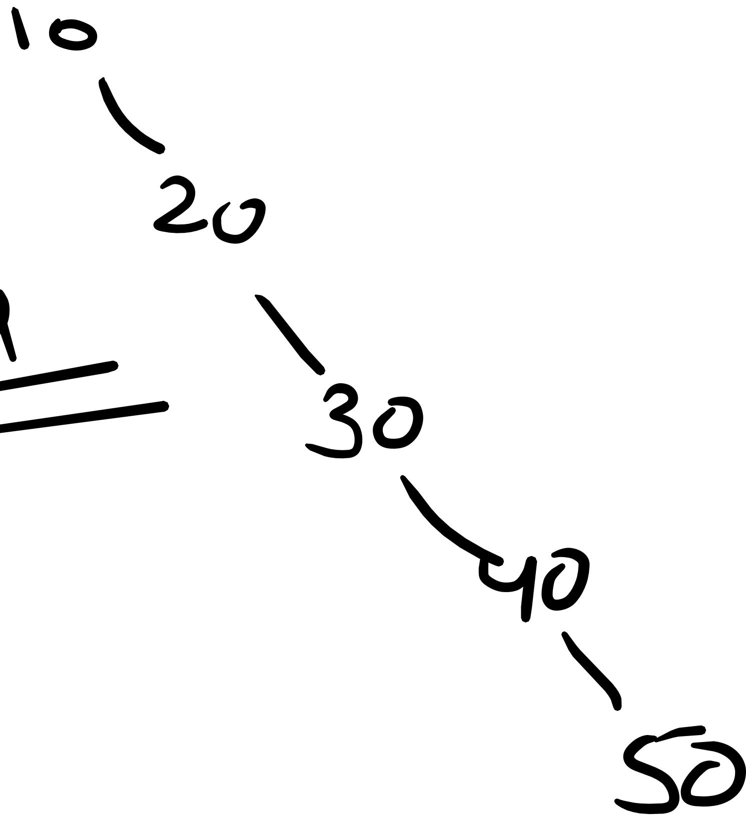
2



3

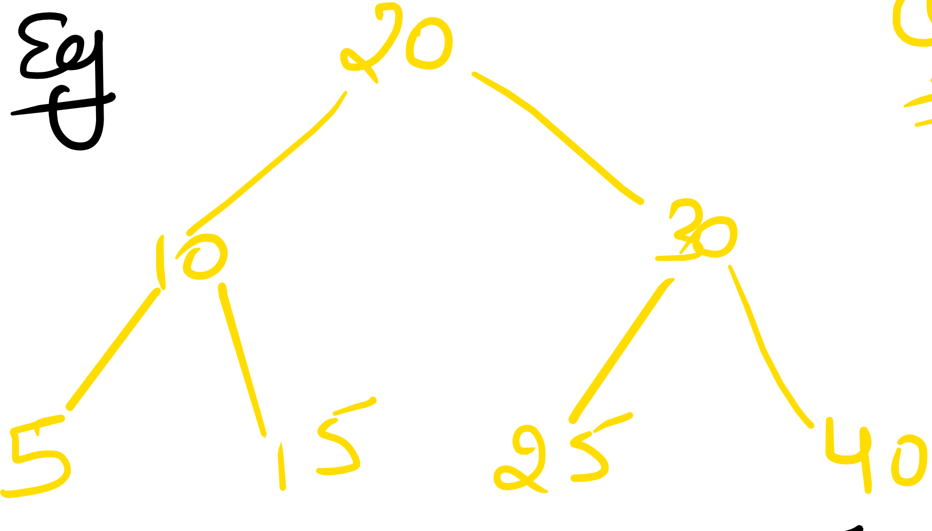


4



T.C. $\Rightarrow O(n^2)$

Eg

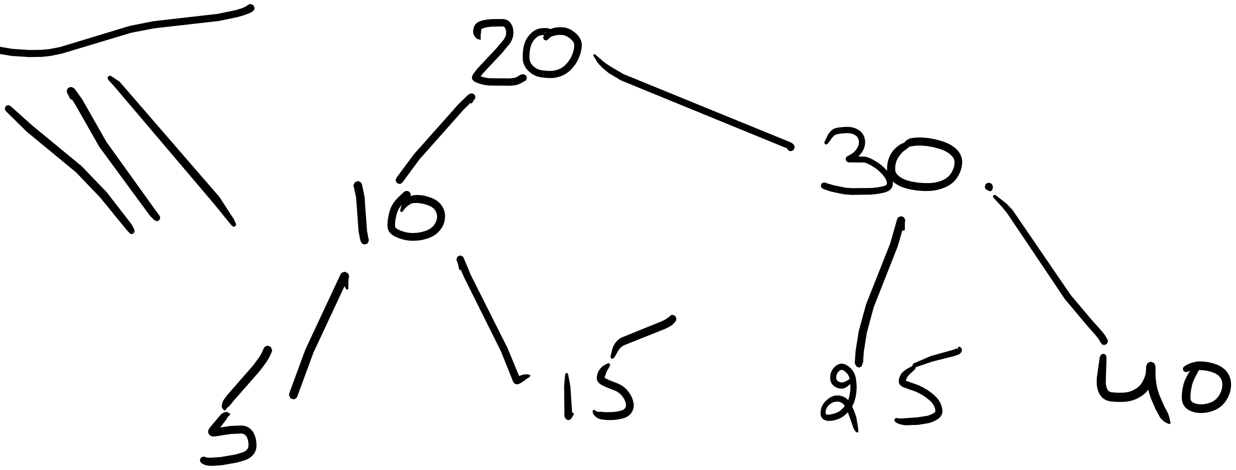


Q) Given ?

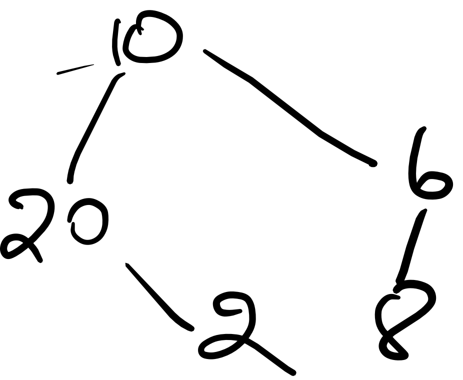
Inorder :- (5, 10, 15), 20, 25, 30, 40

preorder :- 20, 10, 5, 15, 30, 25, 40

BST = ?



Given inorder & preorder of BT



Preorder = 10, 20, 2, 6, 8

Postorder = 2, 20, 8, 6, 10

Inorder = 20, 2, 10, 8, 6

Given

Construct BT ?

apply linear search to find where to insert

T.C to create BT if inorder & preorder is given

$\Rightarrow O(n^2)$?

Q) what is the time complexity to create BST if inorder & preorder(postorder) is given?

sol

preorder: - ⊖ ⊖ ⊖ _ _ _

inorder(sorted): _ _ _ _ _ _

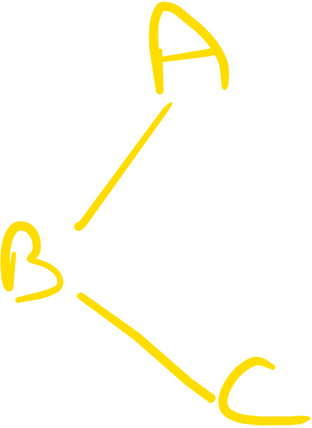
⊖

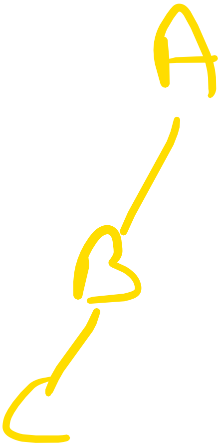
apply Binary search

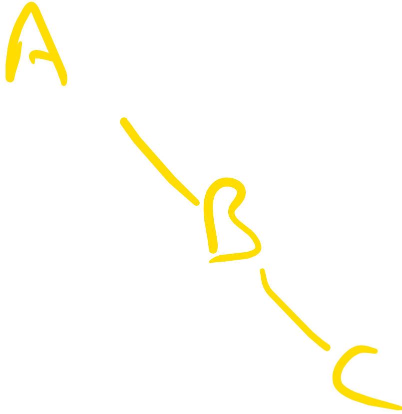
✓

$\Rightarrow O(n\log n)$

(*)







preorder =	ABC	ABC	ABC	ABC
postorder =	CBA	CBA	CBA	CBA

Q) let preorder & postorder is given, Can we construct a unique BT?

Sol) NO

But, if inorder & (preorder / postorder) is given, then we always construct Unique BT

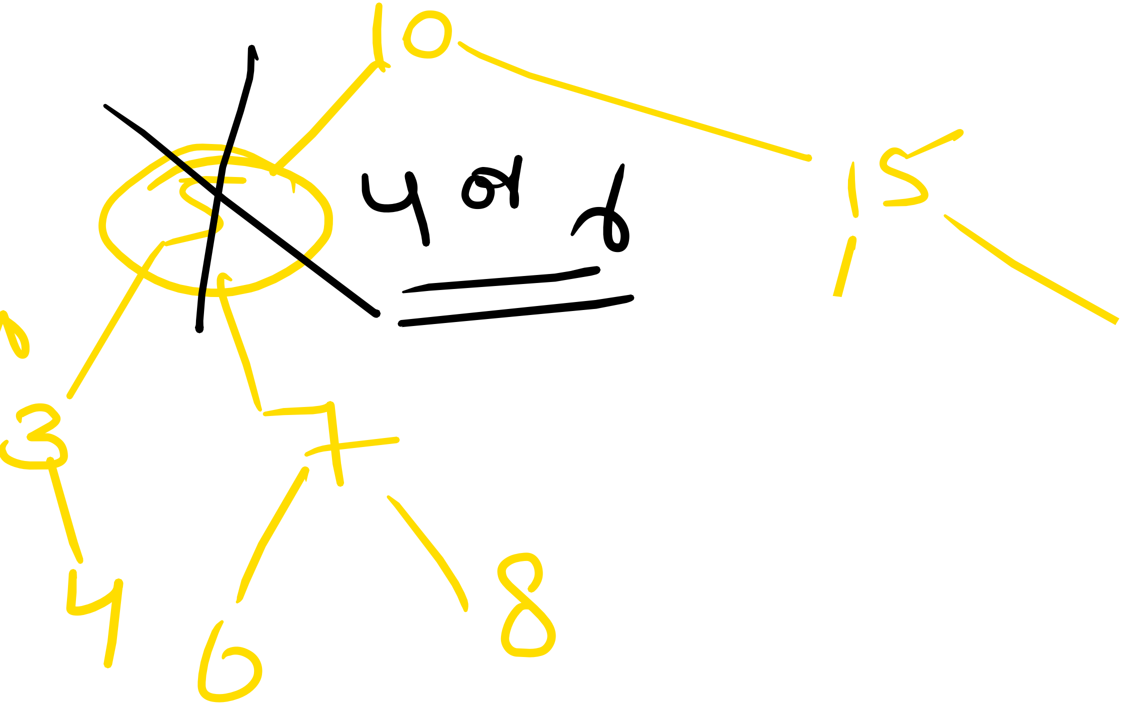
deletion in BST when element has 2-child

inorder

3, 4, 5, 6, 7, 8, 10, 15

4 = inorder predecessor

6 = inorder successor



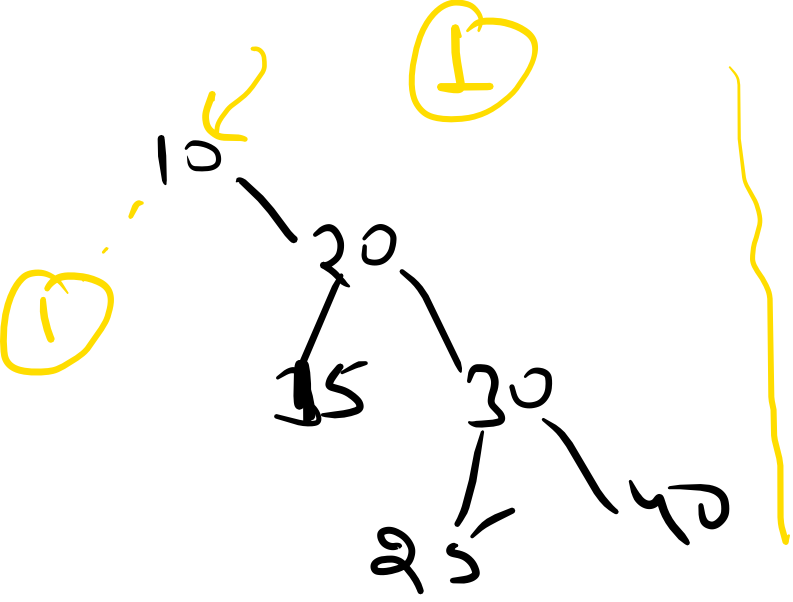
T.C. of BST

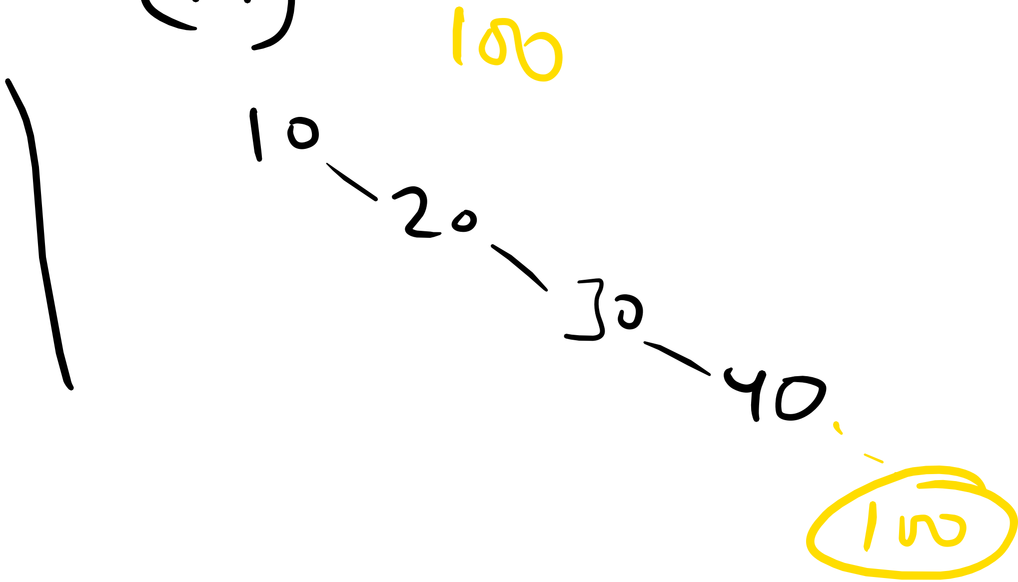
Insert an element

BC = $\Omega(1)$

AC = $\Theta(\log n)$

WC = $O(n)$





Search! →

$B.C = \Omega(1)$

$A.C. = O(\log n)$

$W.C. = O(n)$ ✓

delete = $B.C = \Omega(1) + \text{delete}(\Omega(1)) = \Omega(1)$

$AC = O(\log n) + \text{''} = O(\log n)$

$W.C = O(n) + \text{''} = O(n)$ ✓

Disadvantage of BST

In w.c., insertion, deletion & search will take $O(n)$ time.

→ becoz of skewed tree (left / right) i.e. height is not balance.

Balanced BST :-> AVL

AVL tree :-> Balanced BST

Balance factor :-> Height of left subtree −
(node) Height of right subtree.

Balance factor = $\{0, 1, -1\}$

<u>Problem</u> :->

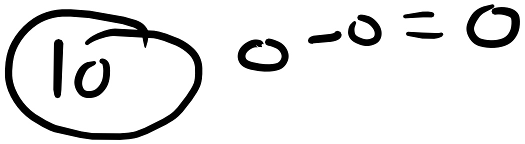
L-L problem

R-R problem

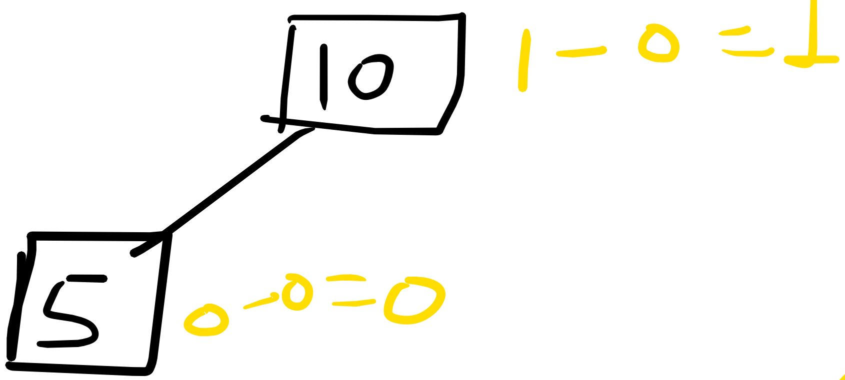
L-R problem

R-L problem.

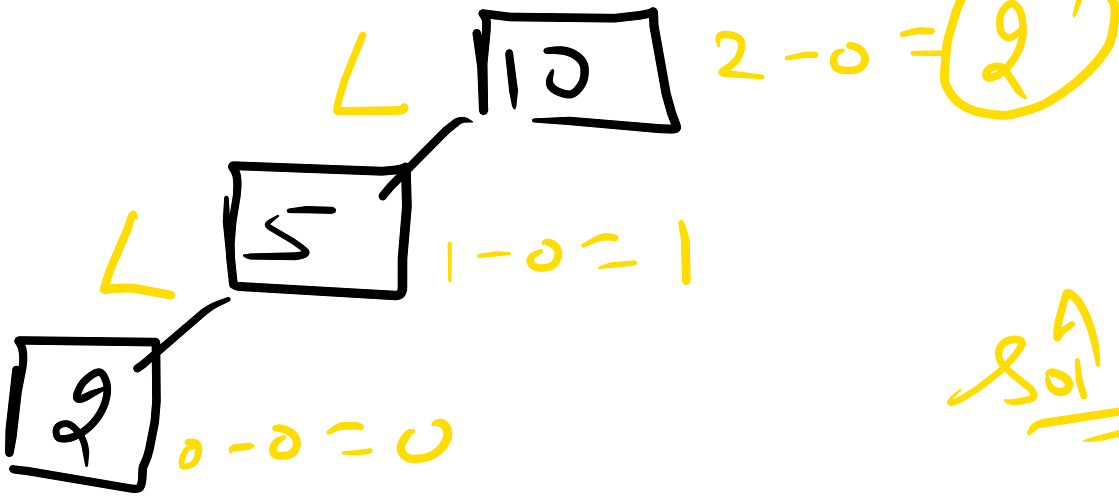
Eg. 10



5



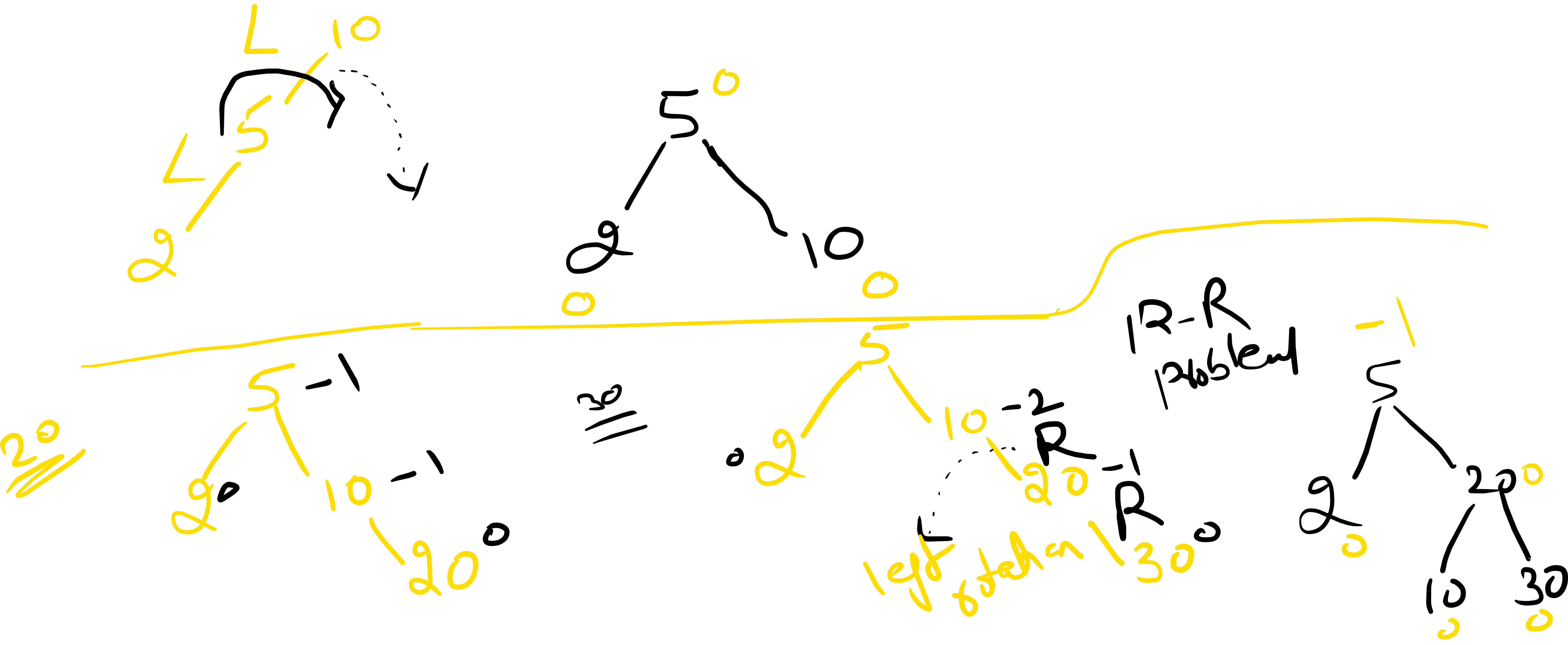
2



violates BF

L-L problem ?

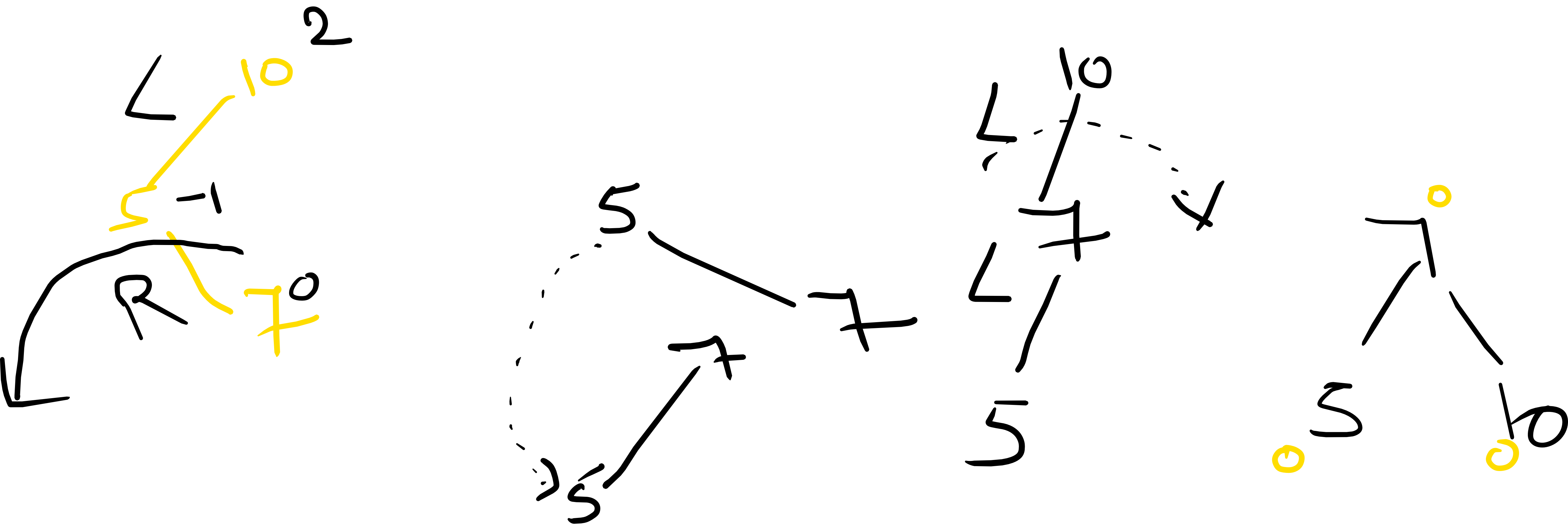
Sol^n do a right rotation


L
10
5
L
2
5
0
2
10
0
0
5 -1
20
2 0
10 -1
20 0
30
5
0 2
10 -2
R
20 -1
R
left rotation
30 0
R-R
Problem
-1
5
2
0
20 0
10
30
0
0

L-L → do a right rot w.r.t. upper edge

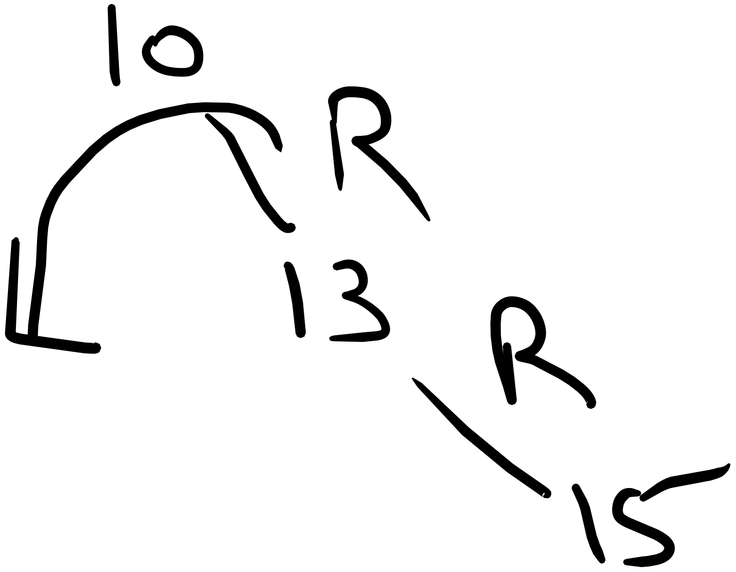
R-R → " left " "

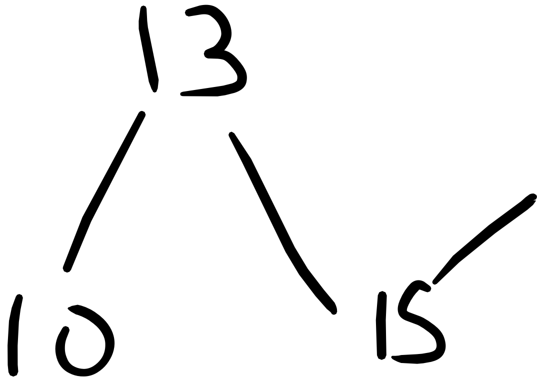
L-R ⇒ left rot^n & then right rotation



R-L

R-L





when L-R problem

a) do left rota^n (lower edge)

b) do right rot^n (upper edge)

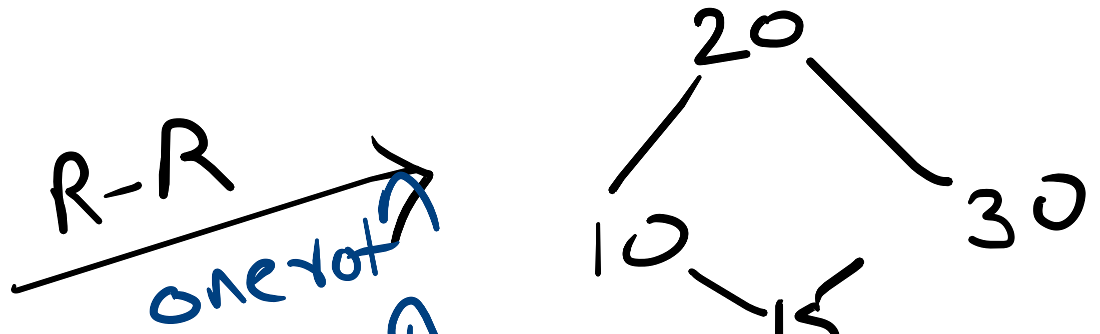
when R-L problem

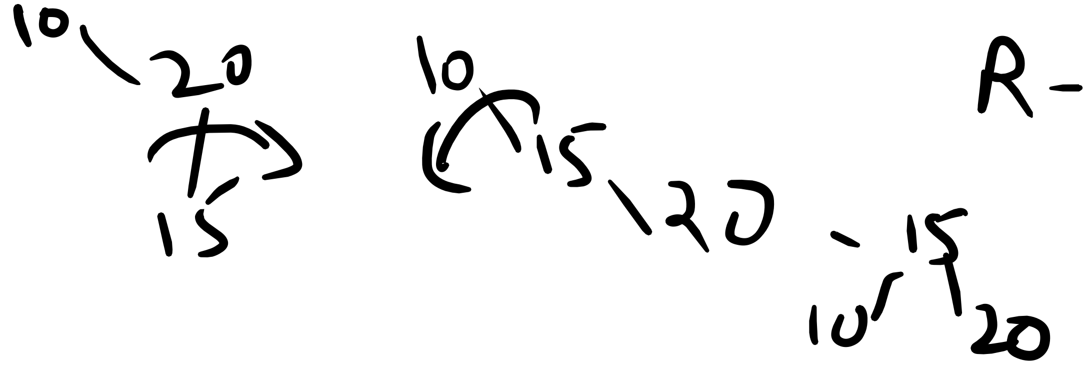
a) do right rot^n (lower edge)

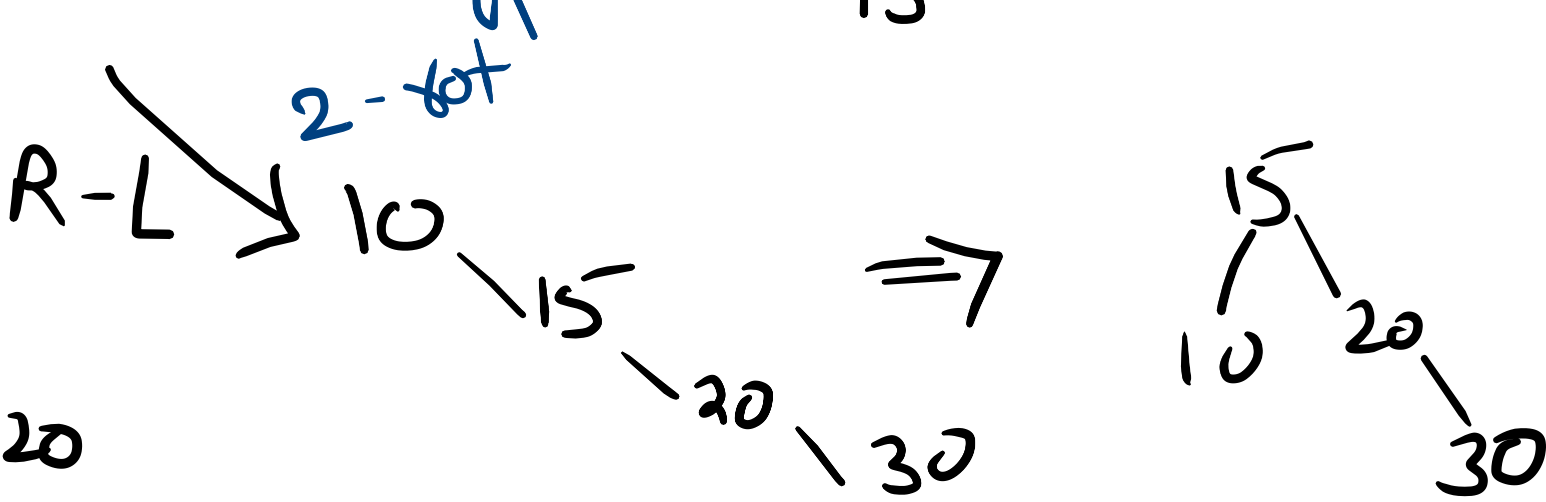
b) do left rot^n (upper edge)

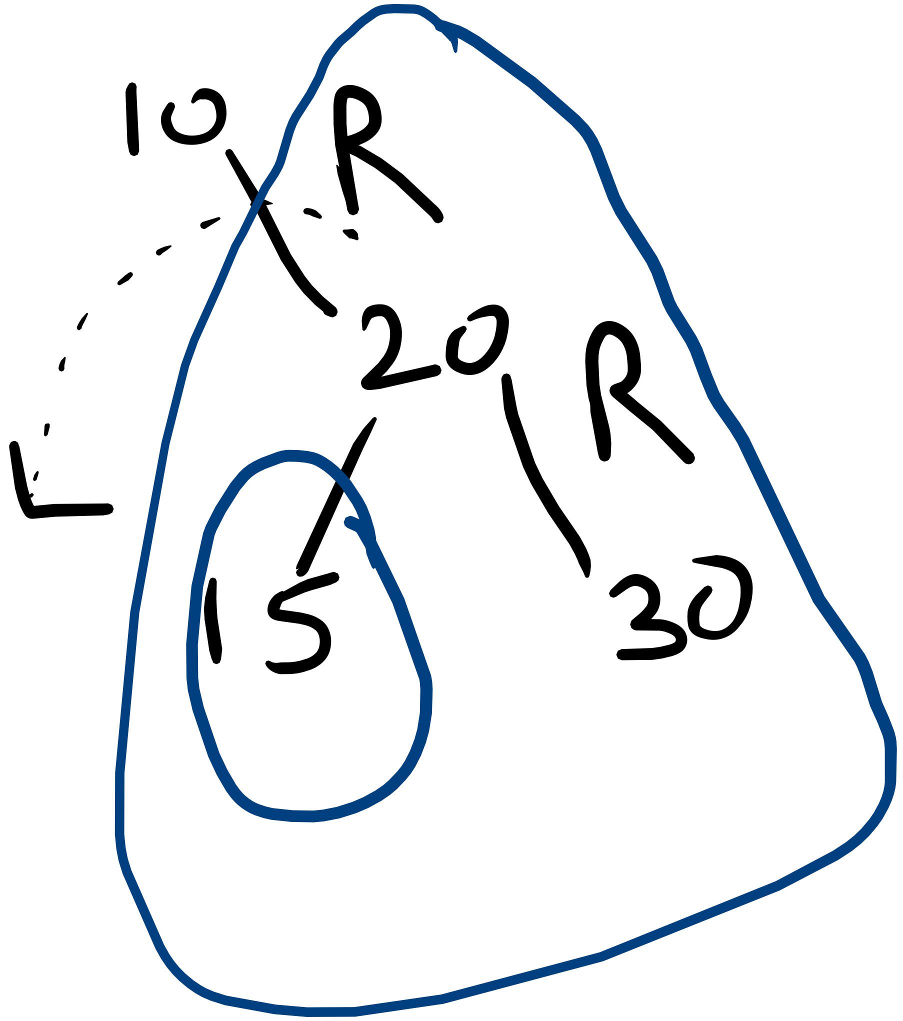
Eg

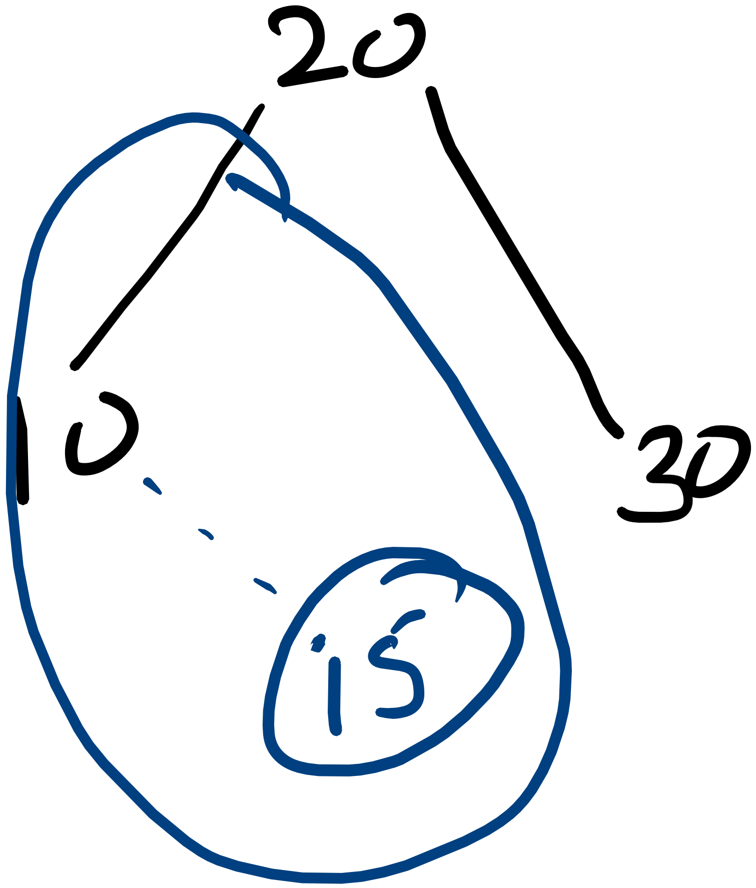
So, 10 has R-R or R-L problem.







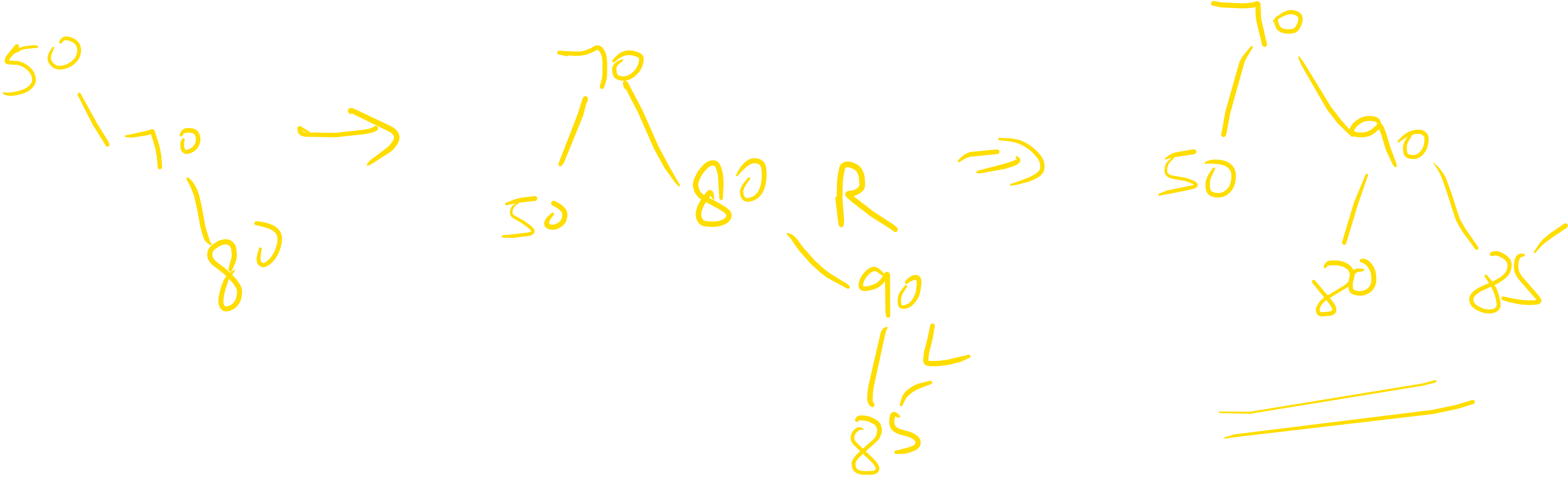

10
R
20
R
L
15
30


20
10
30
15

Note:- If a node have both R-R & R-L (similarly L-L & L-R) then min. no. of rot^n possible when we select problem as R-R (L-L) respectively.

Eg Construct AVL

50, 70, 80, 90, 85,



T.C :-

Construct AVL :->

Balance

↳ $n\left(\log n + c\right)$

$\Rightarrow O(n\log n)$

Insert : →

B.C $= c + c = \Omega(c)$

A.C $= \lg n + c = \Theta(\lg n)$

W.C $= \lg n + c = O(\lg n)$

Deletion

$B.C = c + c = \Omega(c)$

$A.C. = \log n + c = \Theta(\log n)$

$W.C. = \log n + c = O(\log n)$