Data Science & AI
&
NIC - Param

Python-For Data Science

Binary Tree

Lecture No.- 03

By- Pankaj Sharma Sir

Recap of Previous Lecture

Topic Trees Part 02

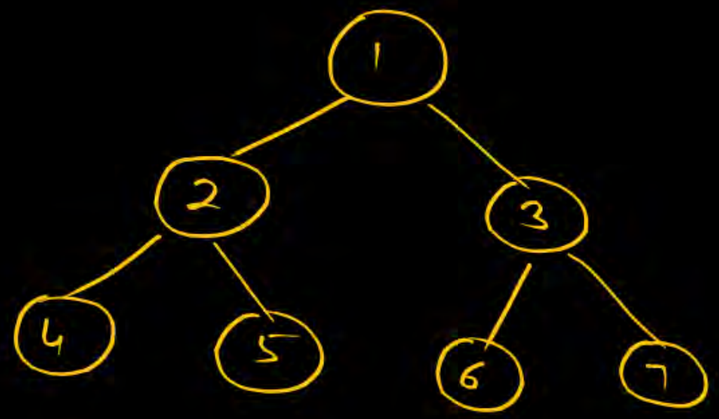
Topics to be Covered

Topic Trees Part 03

① Pre, Post → 1 or many

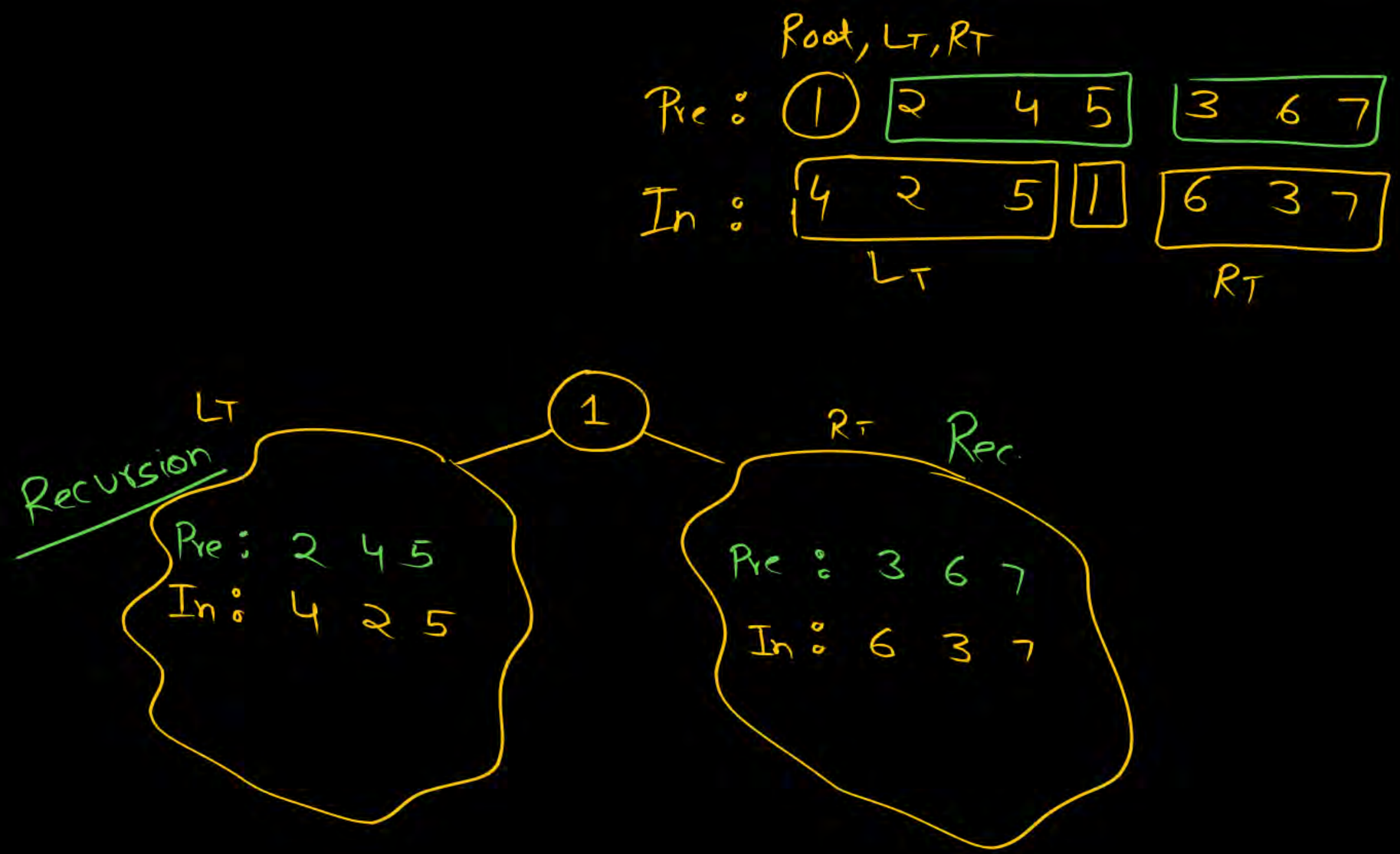
② Pre, In → unique

③ Post, In → unique



Pre: 1 2 4 5 3 6 7

In: 4 2 5 1 6 3 7


Root, LT, RT
Pre : 1 2 4 5 3 6 7
In : 4 2 5 1 6 3 7
LT
RT
LT
1
RT
Rec.
Recursion
Pre : 2 4 5
In : 4 2 5
Pre : 3 6 7
In : 6 3 7

Root, LT, RT

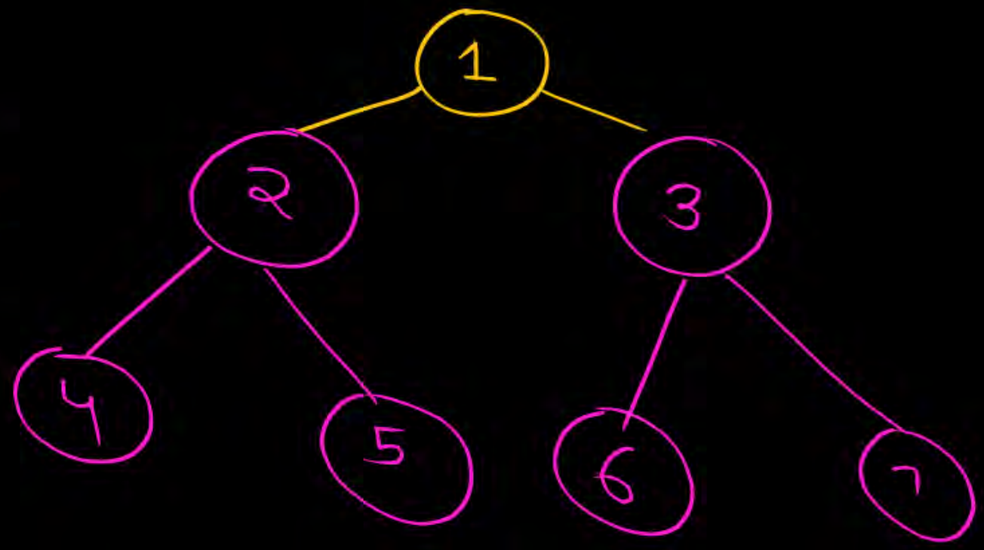
Pre : 1 | 2 4 5 | 3 6 7

In : 4 2 5 | 1 | 6 3 7

LT RT

Pre : 2 4 5

In : 4 2 5

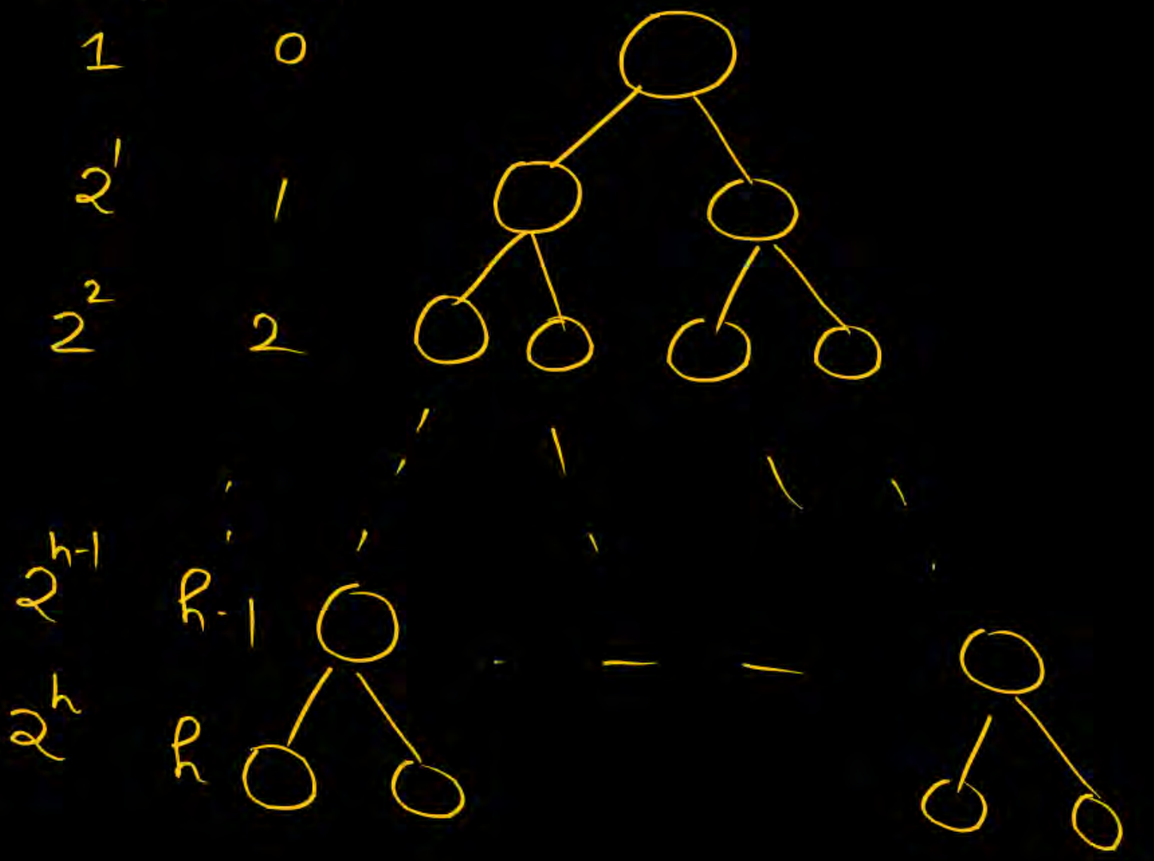


Pre : 3 6 7

In : 6 3 7

Tree

Nodes	level
1	0
2^1	1
2^2	2
2^{h-1}	h-1
2^h	h



Q/ max. no. of nodes in a binary tree of h height.

h+1 terms

$a(r^n-1)/(r-1)$

$$N = 2^0 + 2^1 + 2^2 + \cdots + 2^h$$

$$N = 1 + 2^1 + 2^2 + \cdots 2^h$$

$$= \frac{1(2^{h+1}-1)}{2-1}$$

$$= 2^{h+1} - 1$$

$$\boxed{N = 2^{h+1} - 1}$$

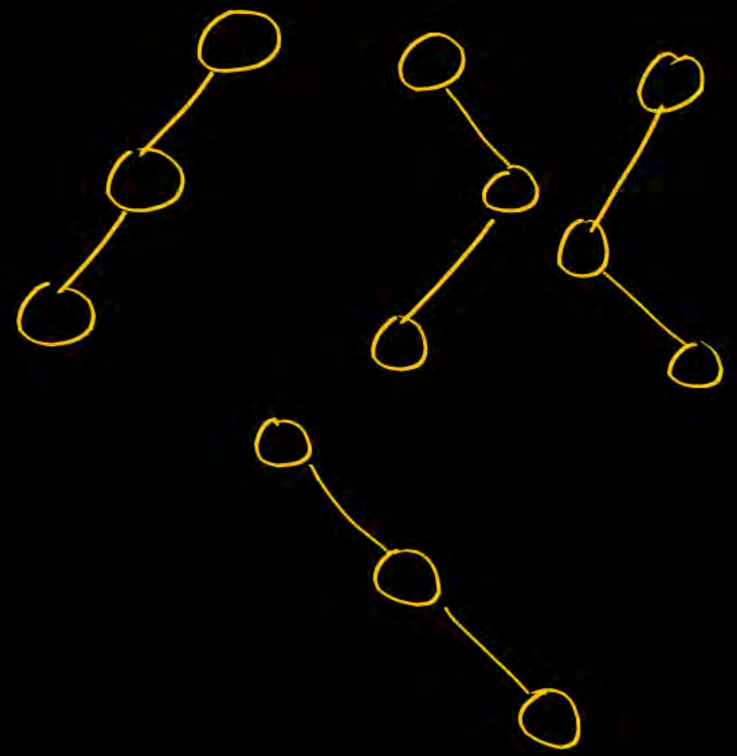
Q min no. of nodes in a binary tree of h height?

$$n_{min} = h+1$$

$$n_{max} = 2^{h+1} - 1$$

$h = 2$

$n = 3$



$n = h+1$

$h = n-1$

$n = 2^{h+1} - 1$

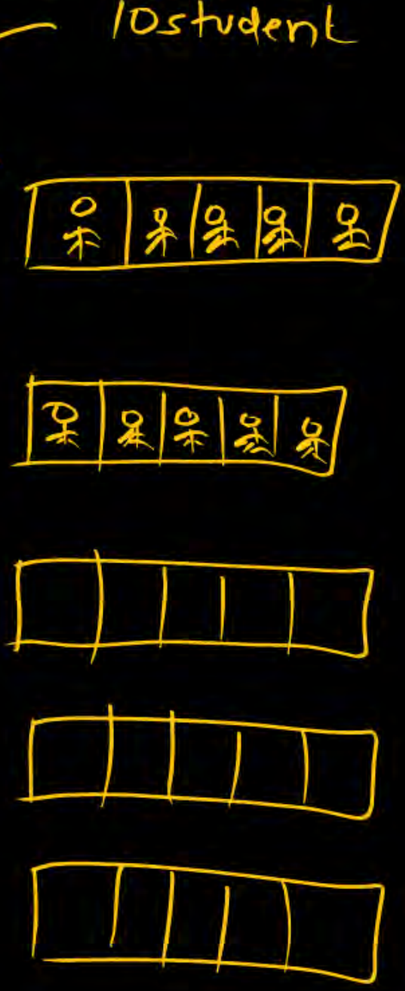
$n+1 = 2^{h+1}$

$\log(n+1) = (h+1)\log 2$

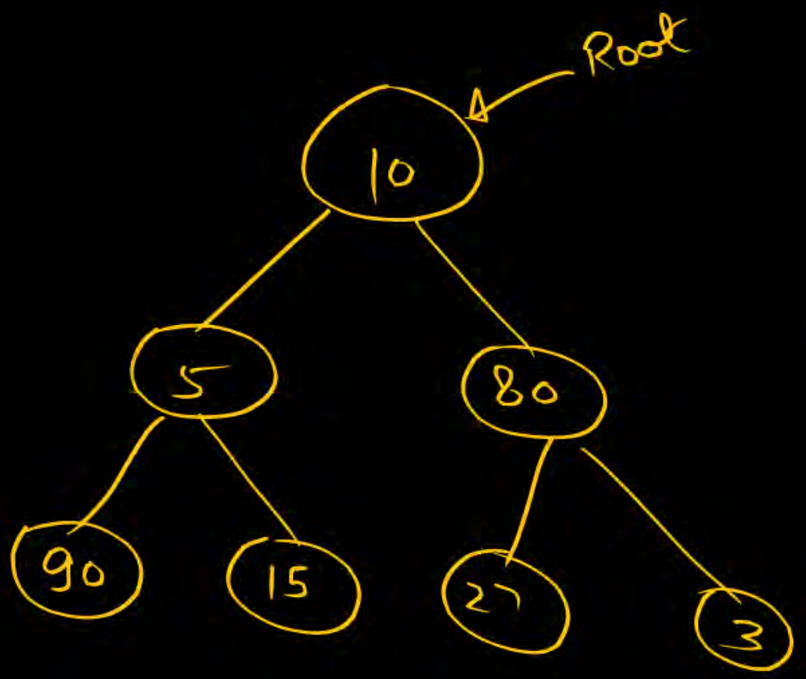
$h+1 = \dfrac{\log(n+1)}{\log 2}$

$h+1 = \log_2(n+1)$

$h = \log_2(n+1) - 1$



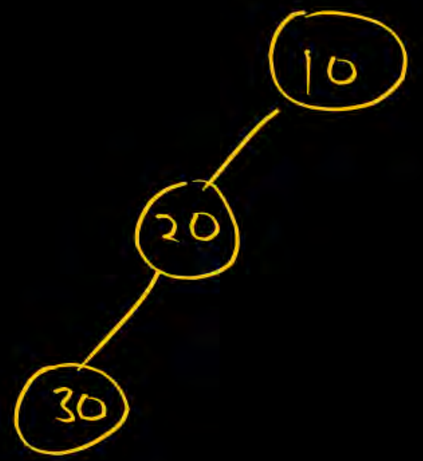
Q Given a binary tree and an element x, find whether x is in the tree or not.



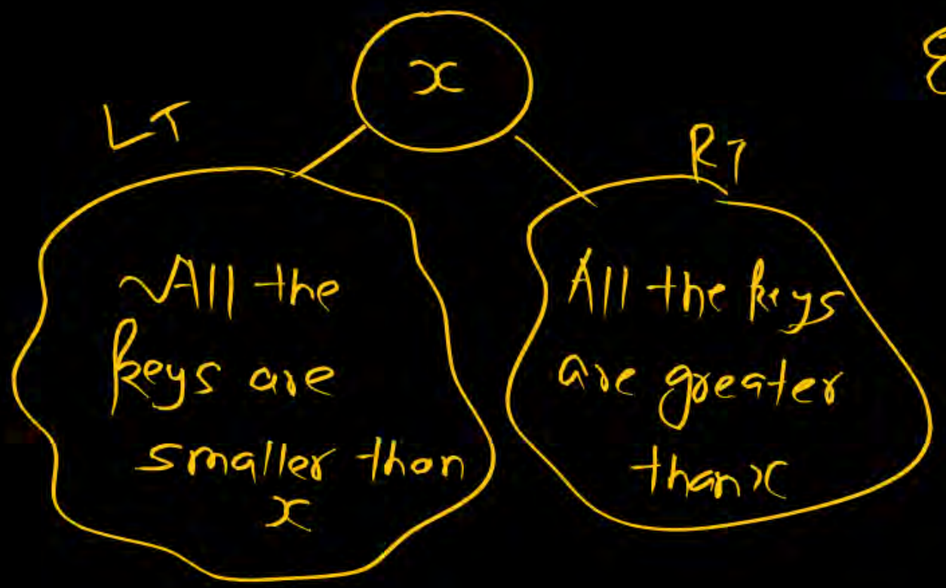
x = 100

→ Preorde
→ Post
→ In

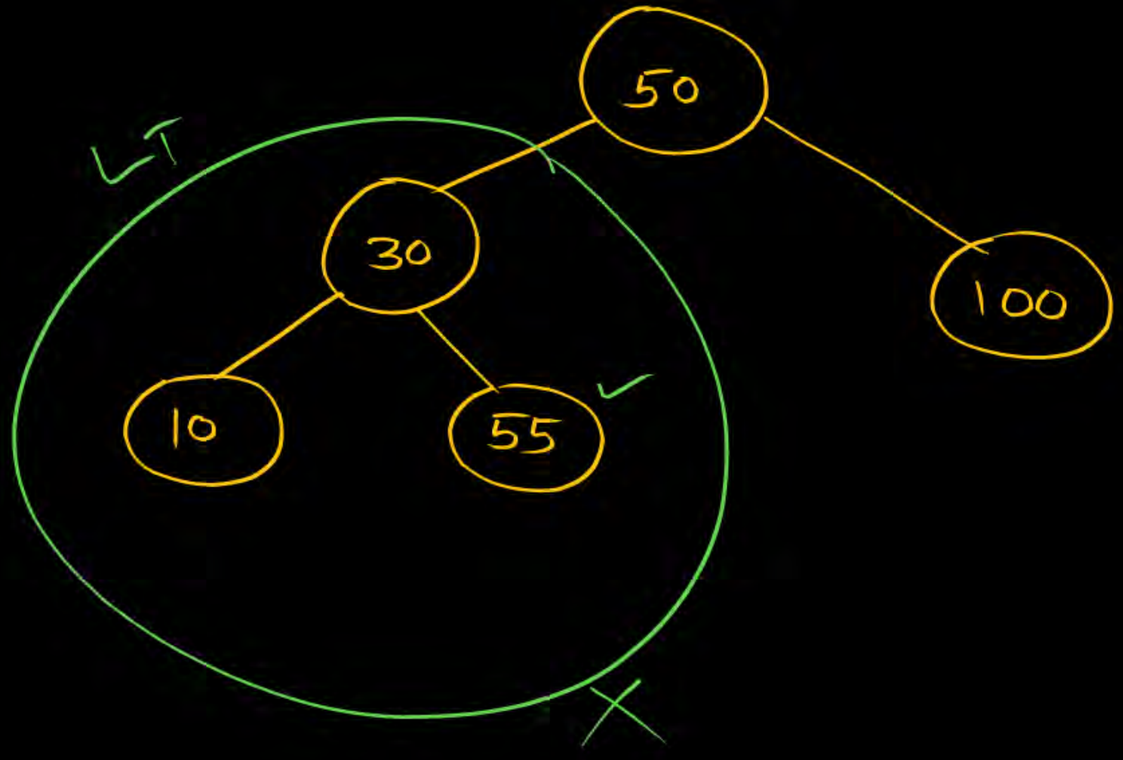
} O(n)

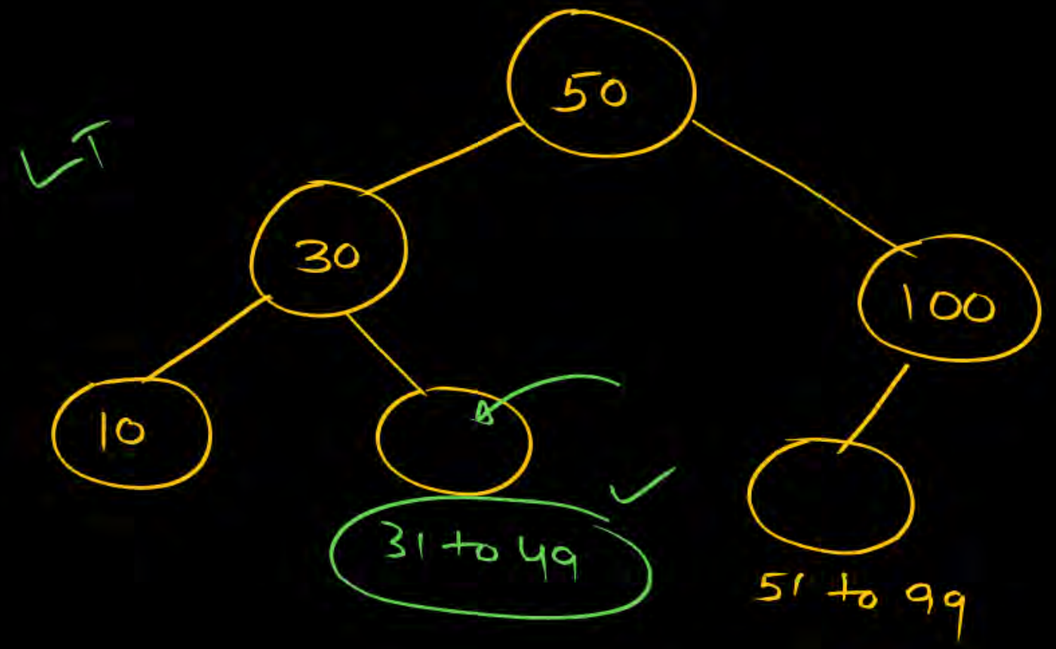


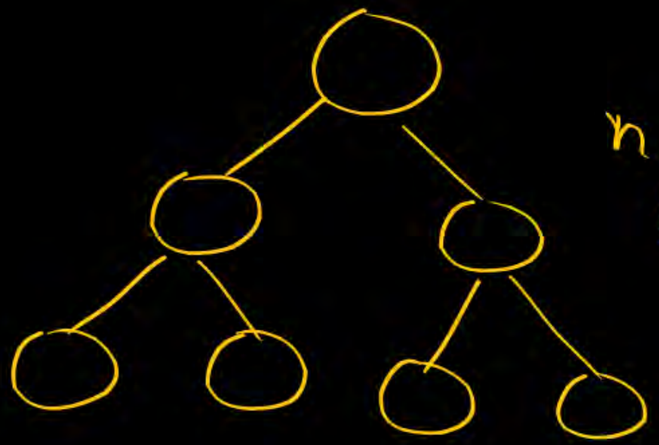
Binary Search Tree



Every node satisfy this prop


50
30
100
10
55


50
30
100
10
31 to 49
51 to 99

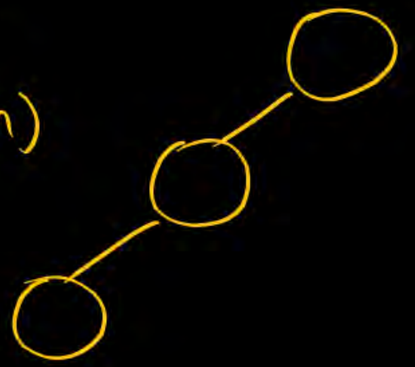


$n = 2^{h+1} - 1$

$h = O(\log_2 n)$

$n = h + 1$

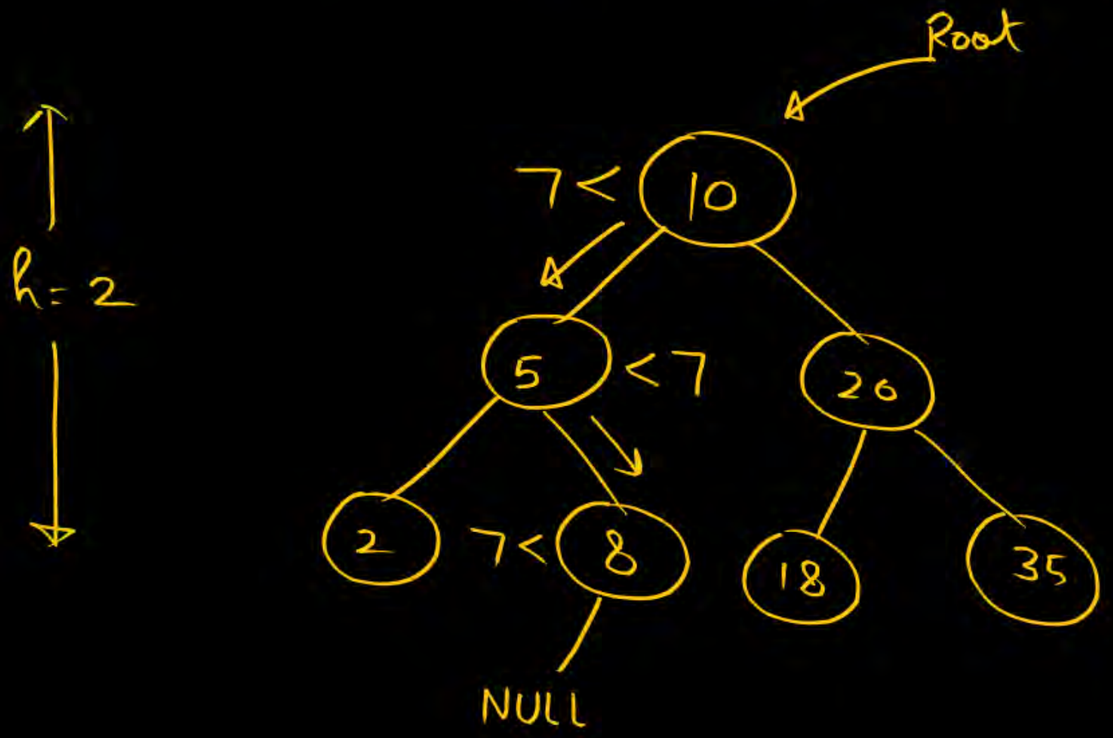
$h = O(n)$



skewed tree

Search in BST

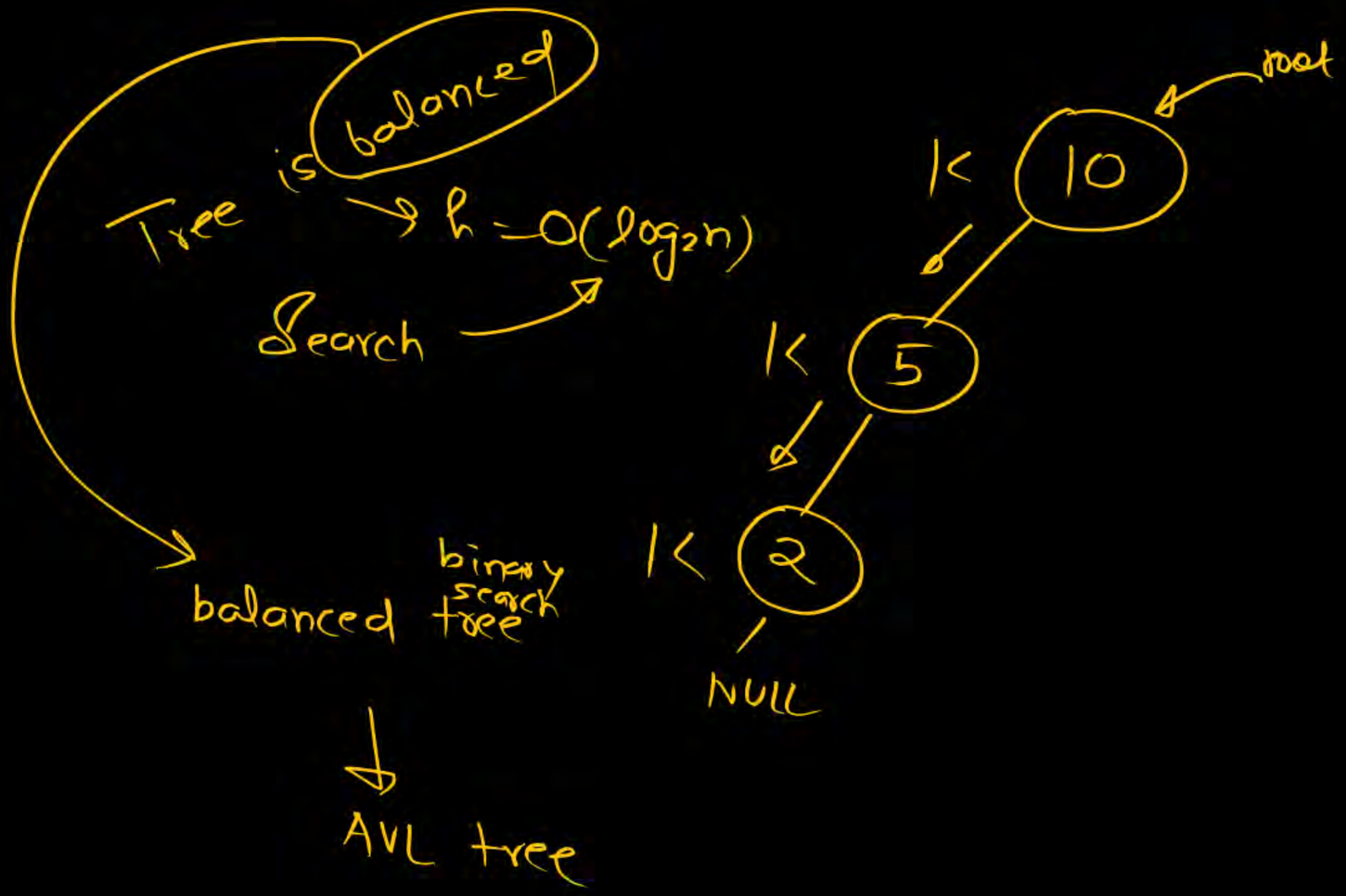
x = 7



Worst case

comp ⇒ O(h)

= O($\log_2 n$)



$n = 1$

No. of com = $O(h)$

$= O(n)$

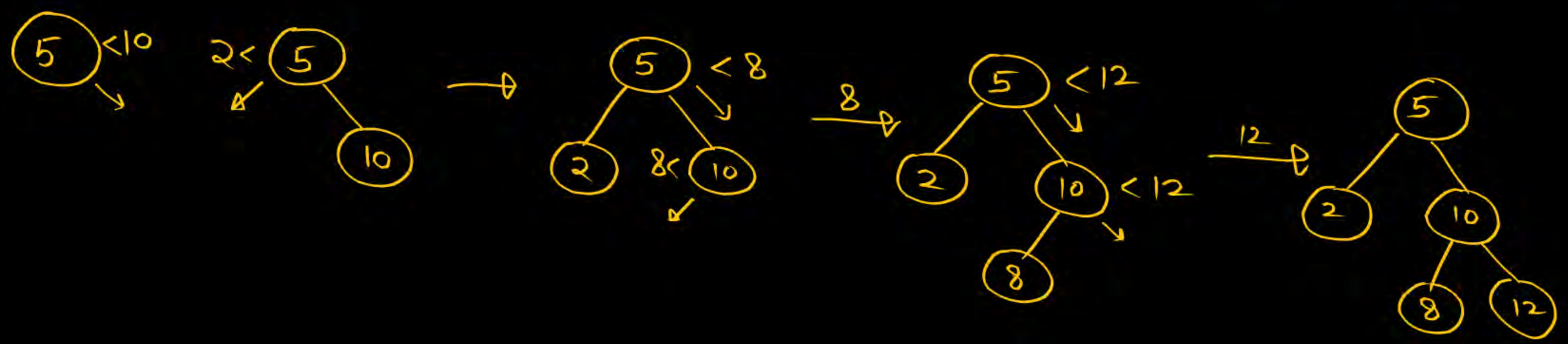
BST

↳ search $\Rightarrow O(n)$

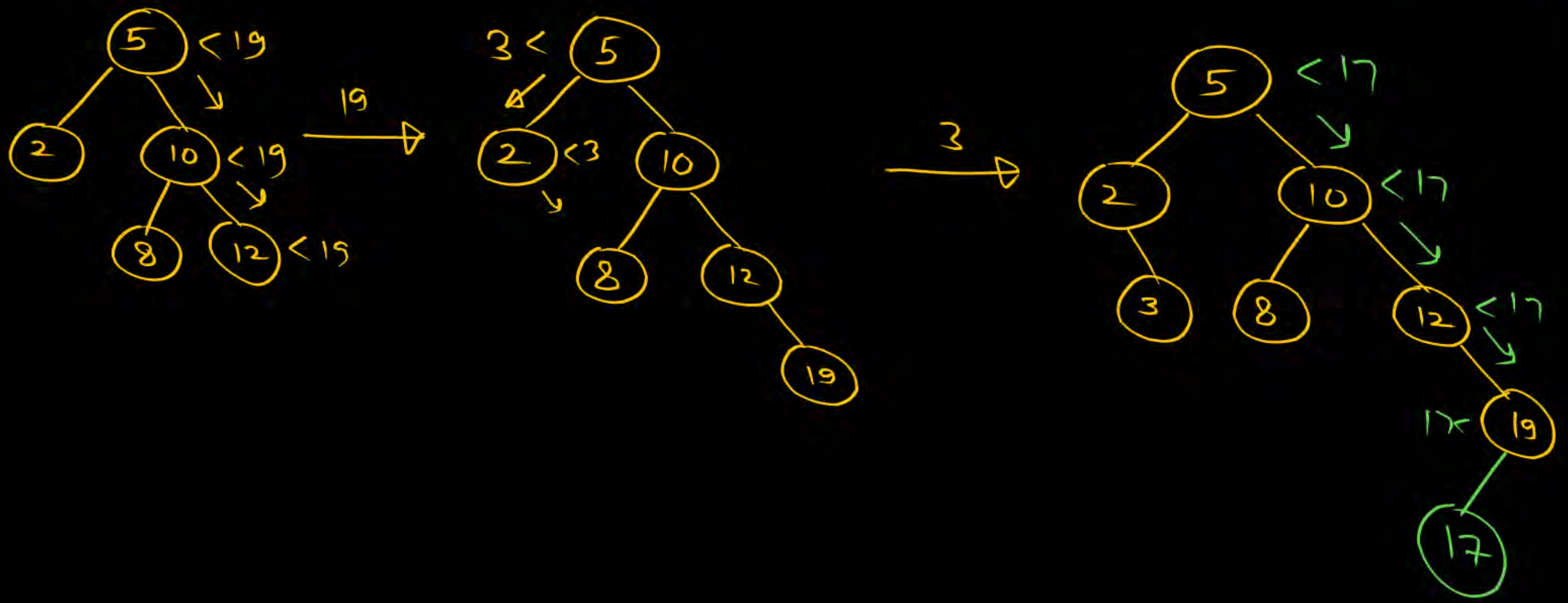
① BST ⇒ Inorder traversal

↳ increasing order of keys.

Q Insert keys 5, 10, 2, 8, 12, 19, 3, 17 into a initially Empty BST.



Q Insert keys 5, 10, 2, 8, 12, 19, 3, 17 into a initially Empty BST.

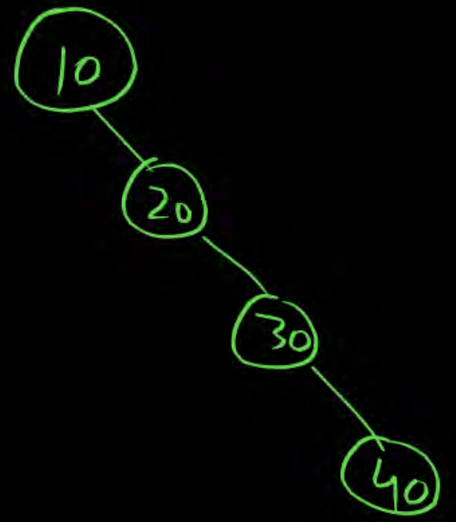


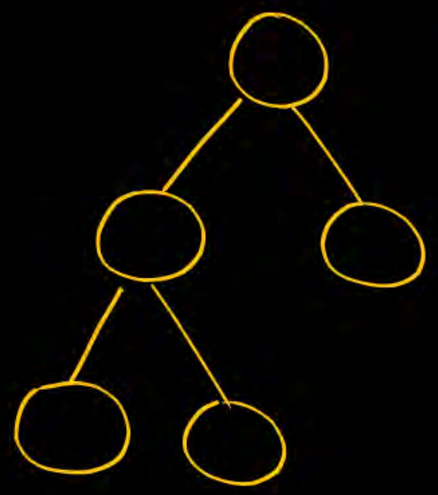
Search → O(n)
Traversal

Insert → Search in Binary tree → O(n)

Insert a key

deletion → O(n)



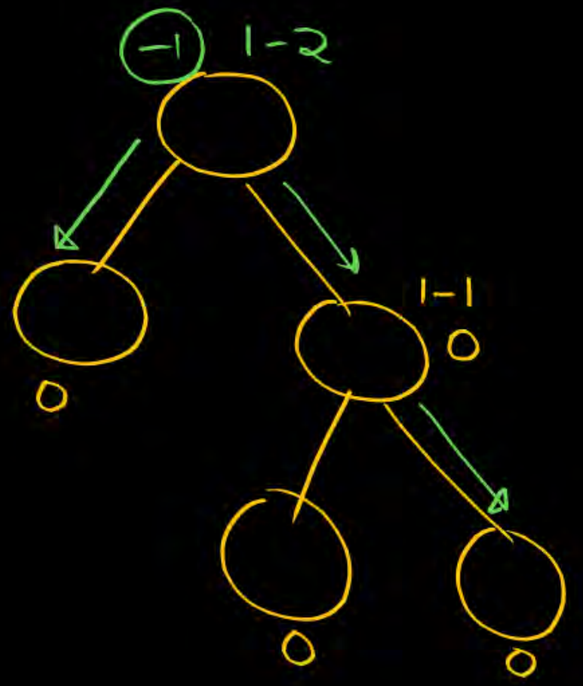


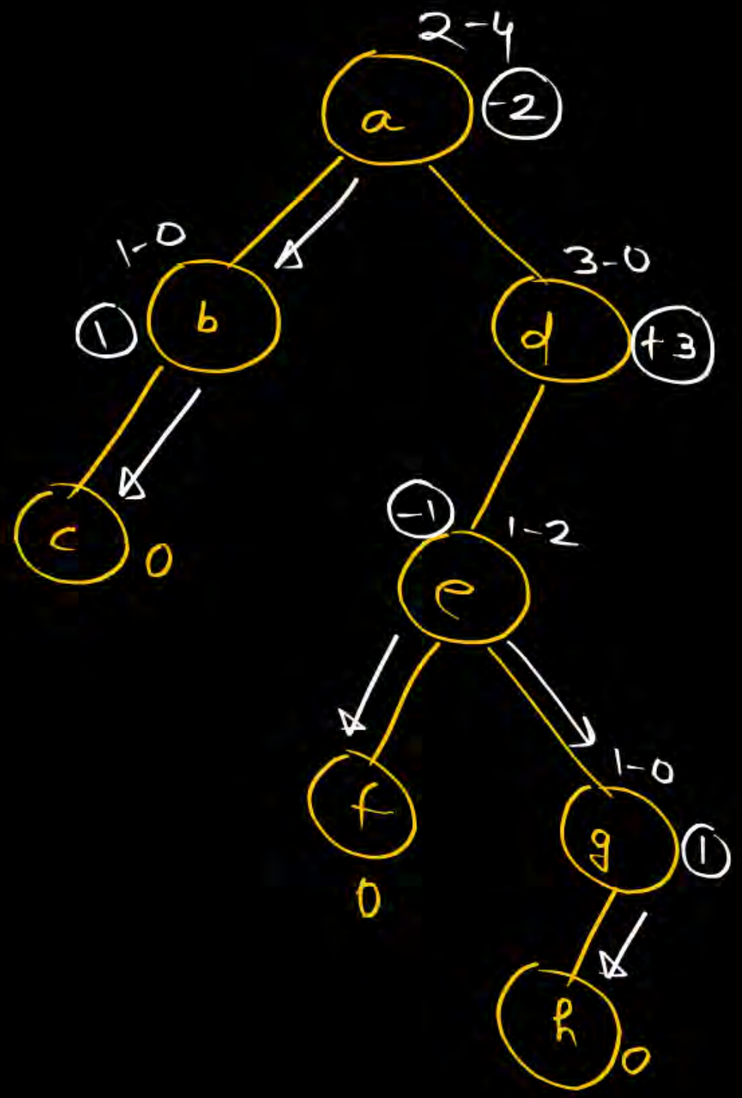
Is it a BST or not?

BST property का structure से कोई लेना देना नहीं है

√AVL - tree

① Balancing factor : is related to structure




2-4
a
-2
1-0
b
1
3-0
d
+3
c
0
-1
1-2
e
f
0
1-0
g
1
h
0

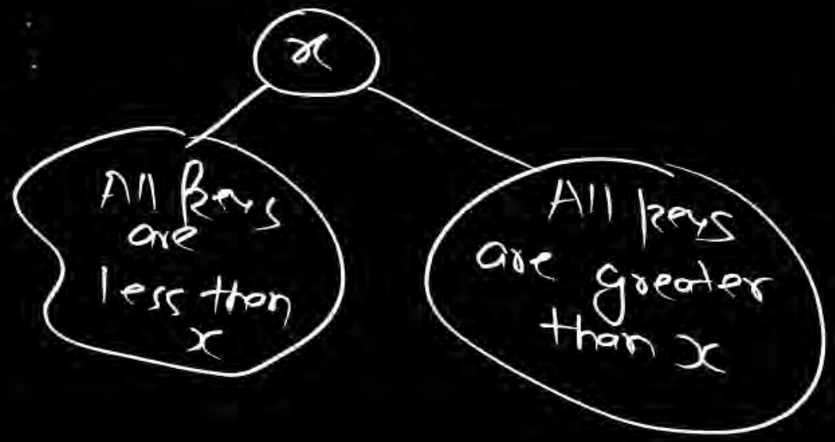
AVL Search tree

Height balanced

20% ✓

Every node satisfy 2 property

(i) BST property :



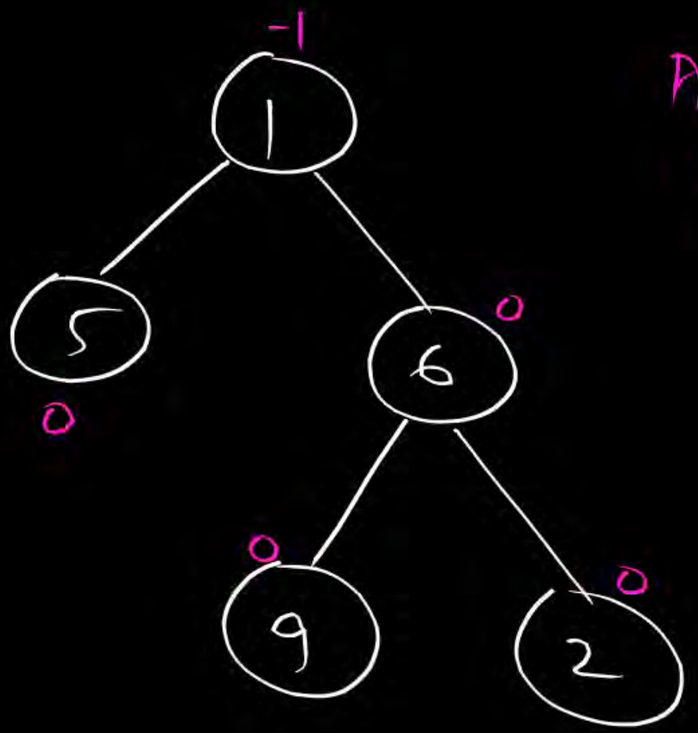
80:1 → 20:1

(ii) AVL tree property :

Balancing factor of Every node must be

−1, 0 or +1

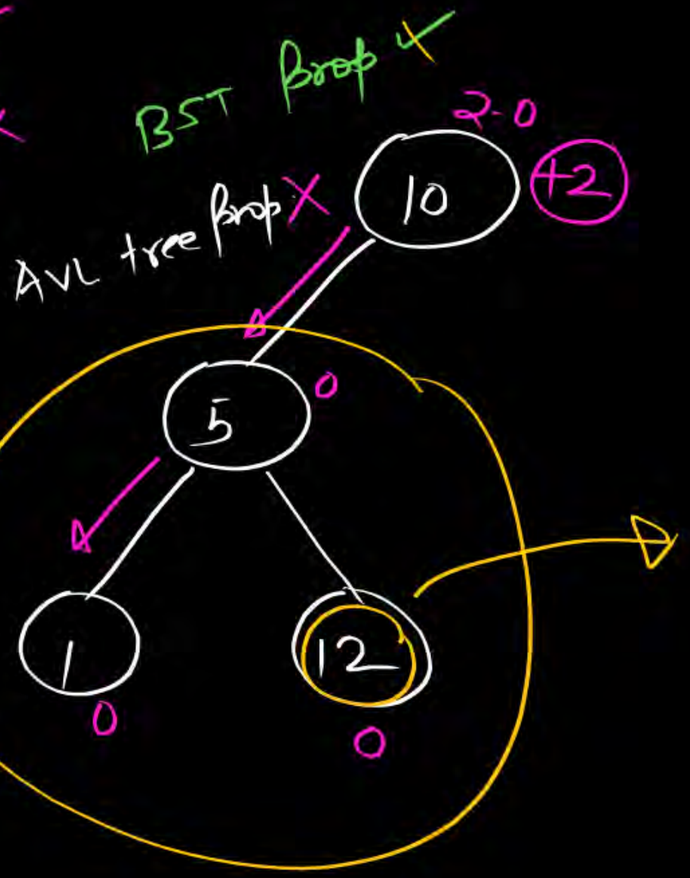
1.



Not a AVL tree

AVL tree prop ✓
BST tree prop ✗

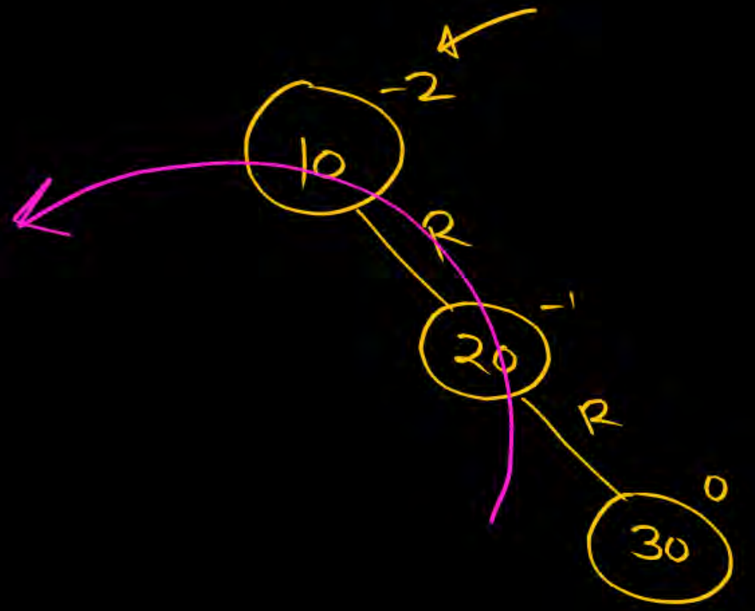
BST prop ✓



Insert → same as BST

but everytime ⇒ check for Bal. factor.

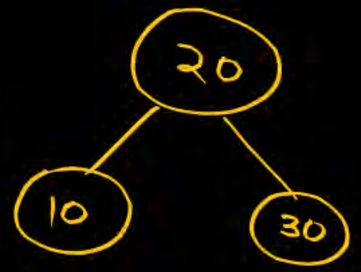
10, 20, 30, 40, 5, 15, 2

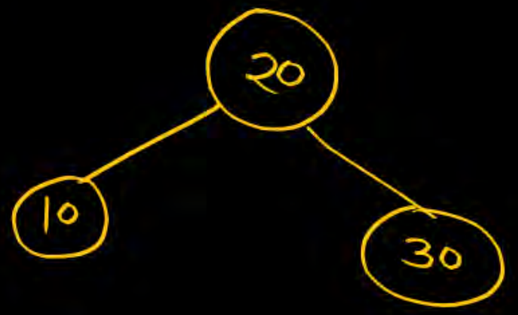


Tri-node structure

10 20 30

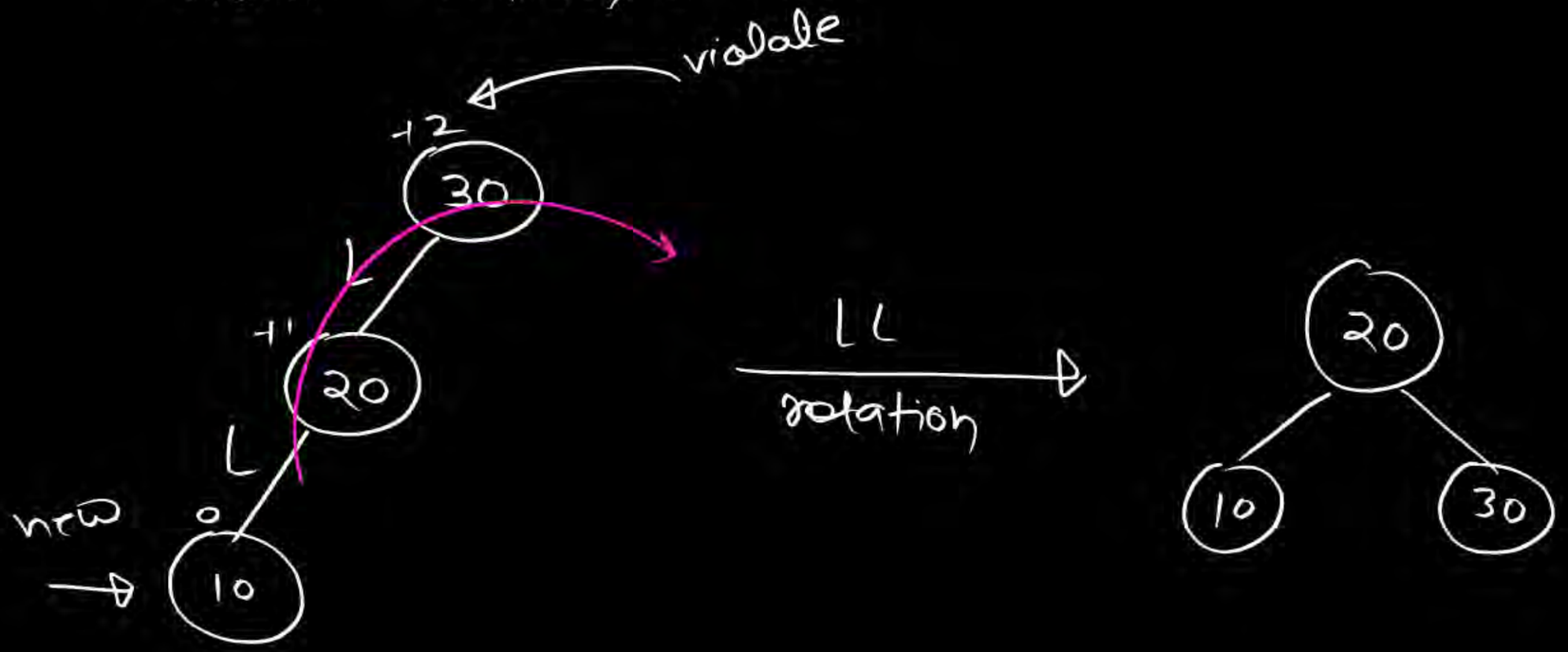
RR rotation



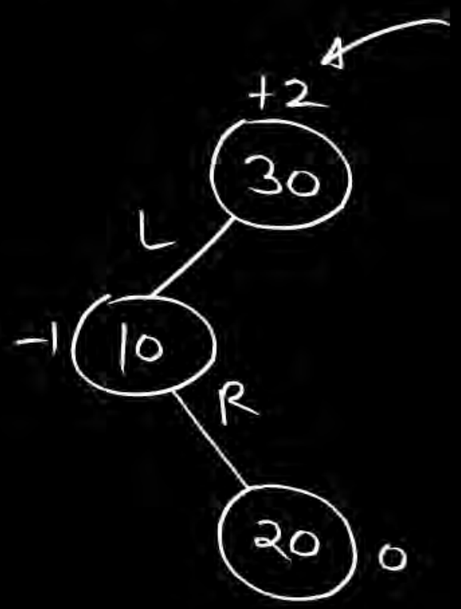


Insert 30, 20, 10

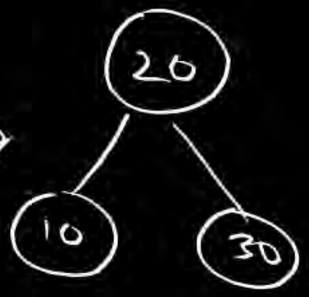
10 (20) 30



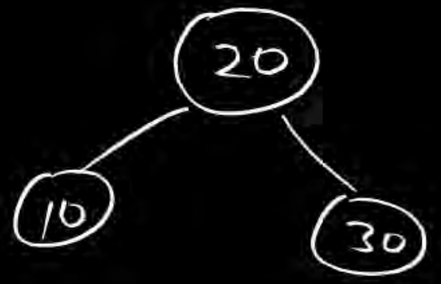
30, 10, 20



LR rotation

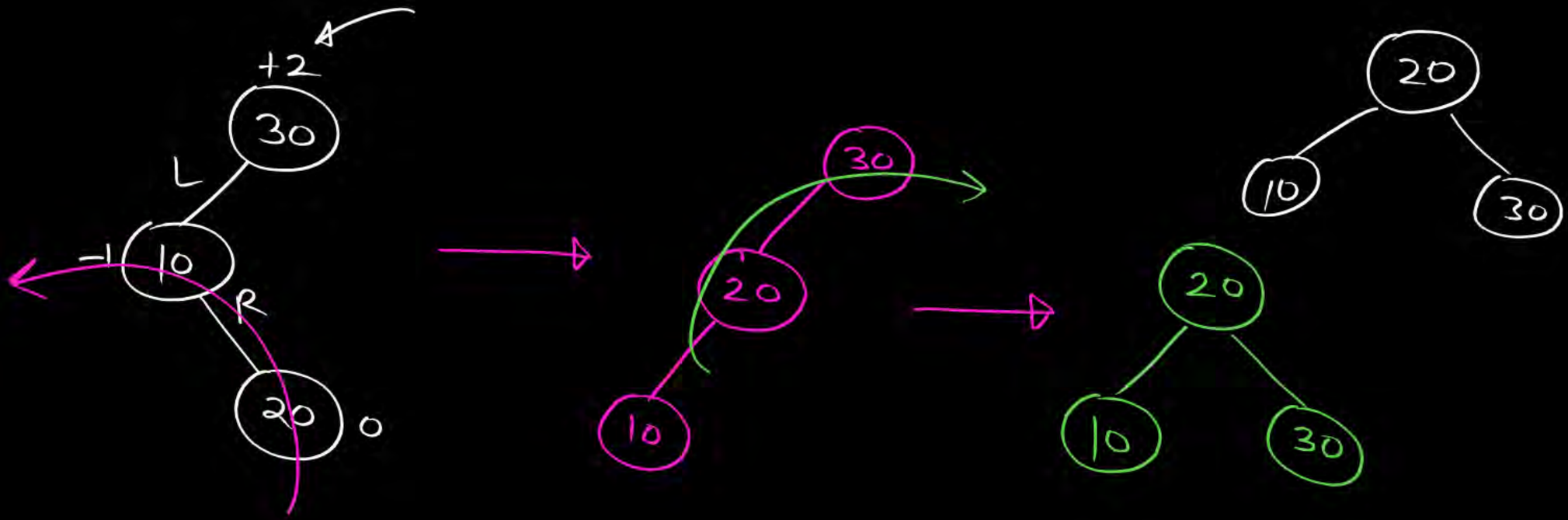


10 20 30

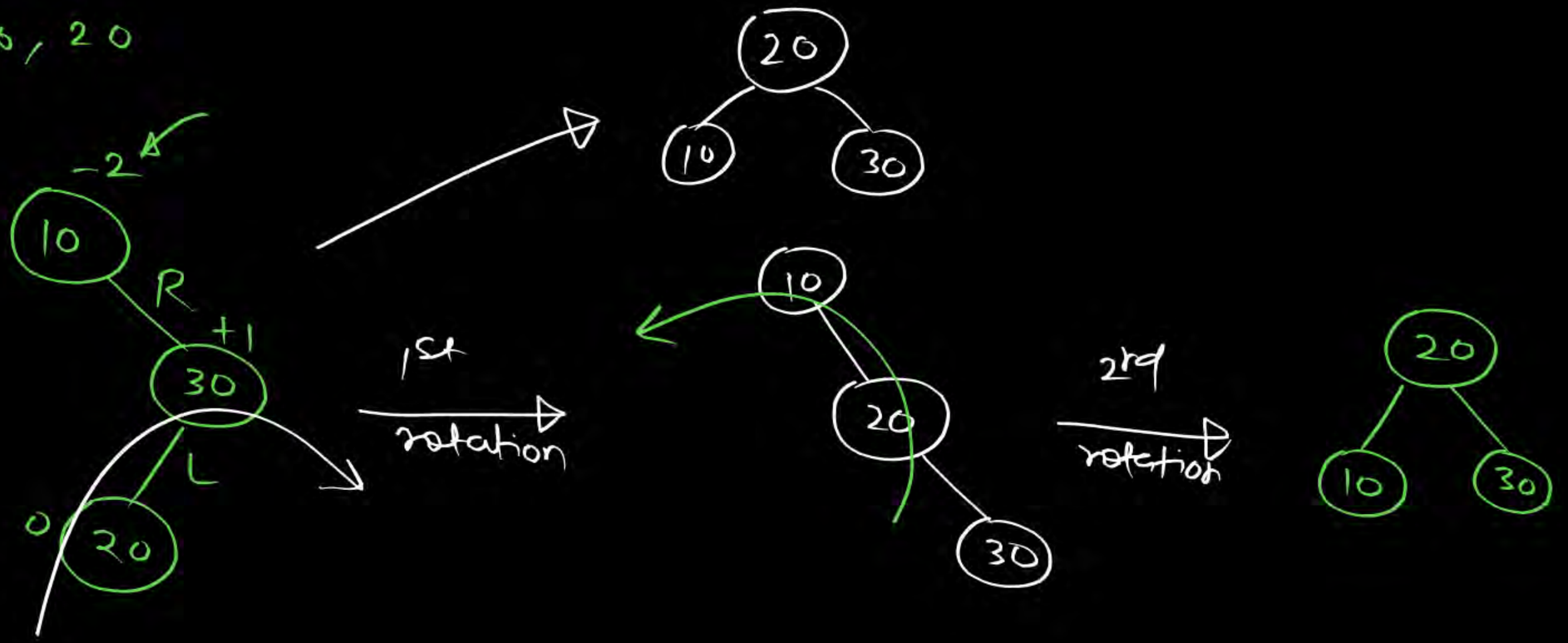


30, 10, 20

10 20 30

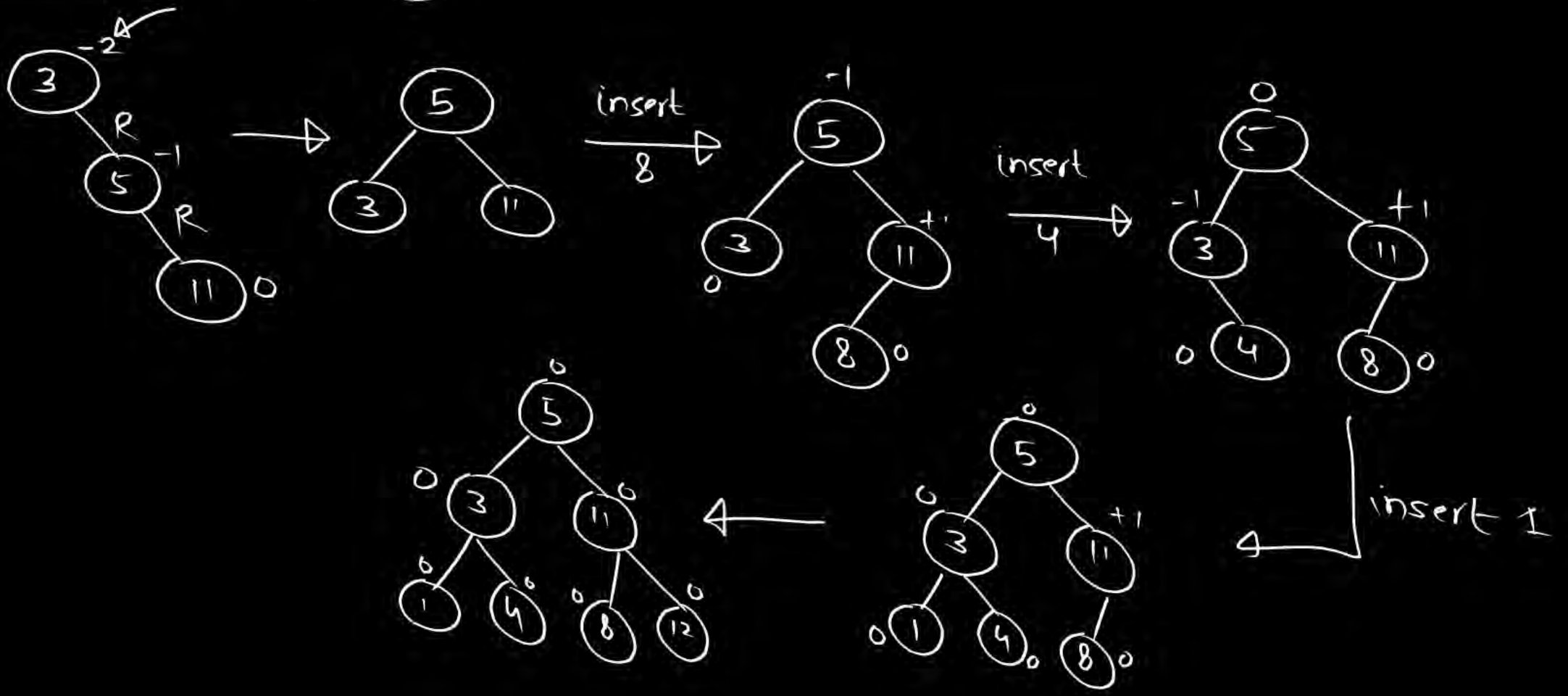


10, 30, 20



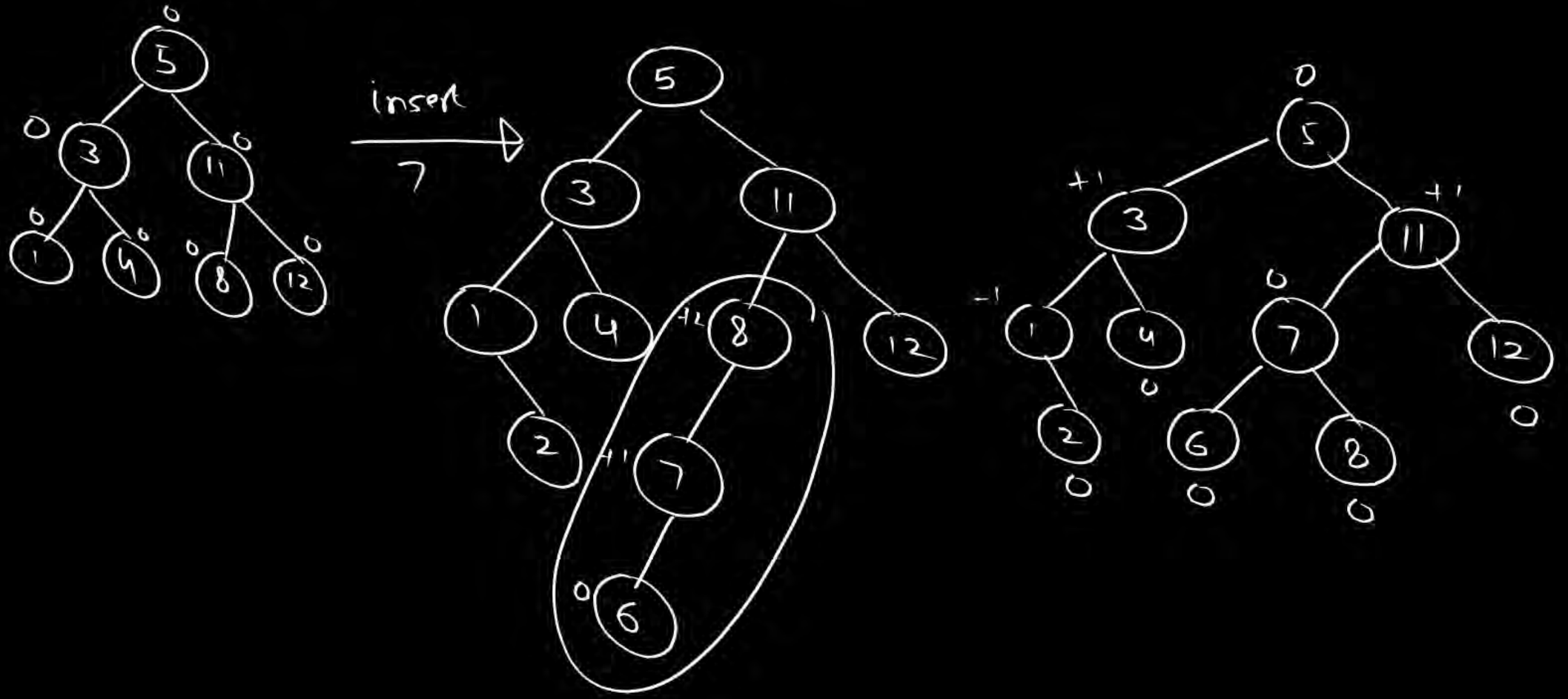
Insert keys into initially Empty AVL tree :

3, 5, 11, 8, 4, 1, 12, 7, 2, 6

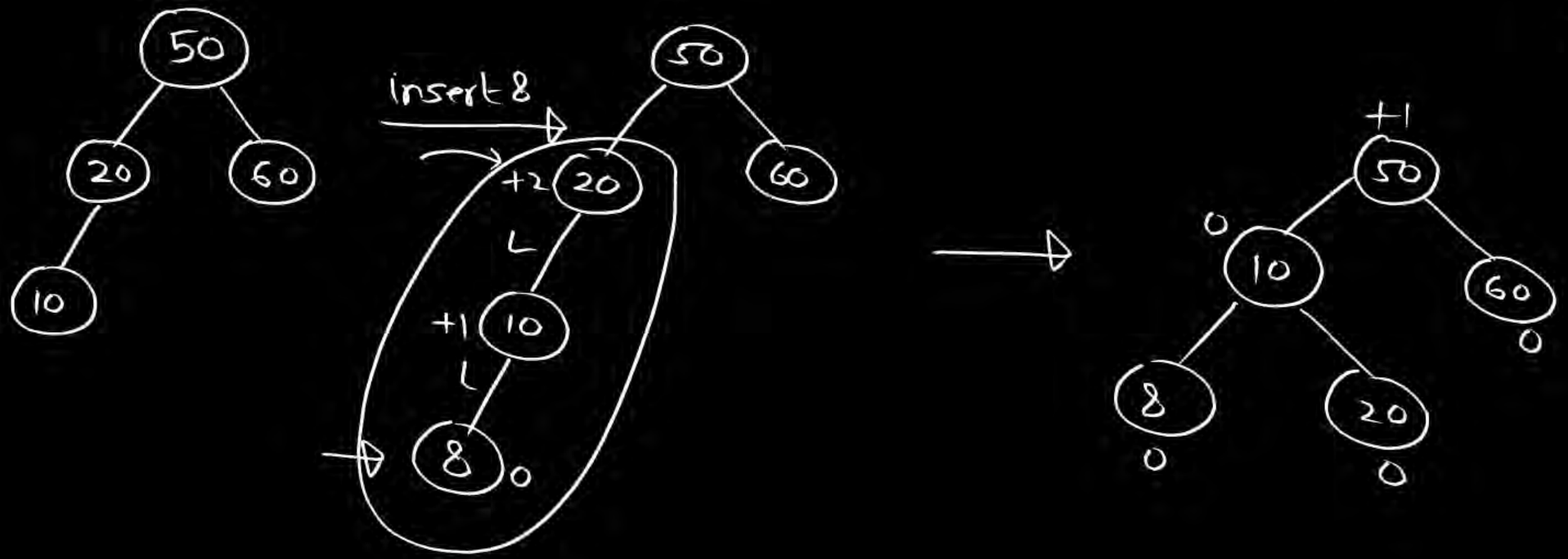


Insert keys into initially Empty AVL tree :

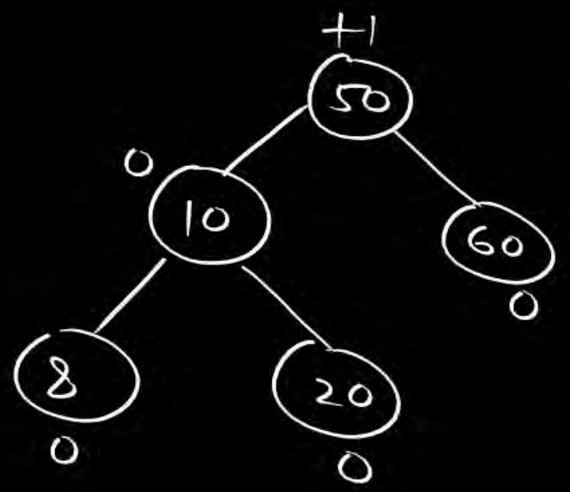
3, 5, 11, 8, 4, 1, 12, 7, 2, 6



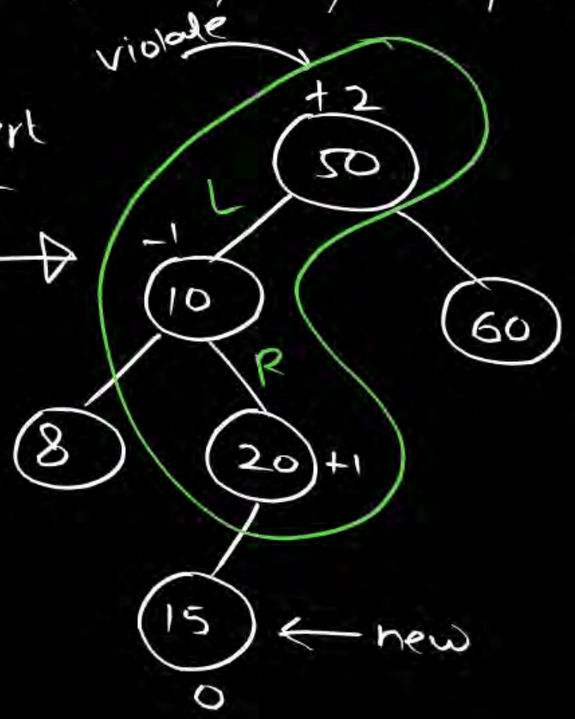
Keys : 50, 20, 60, 10, 8, 15, 32, 46, 11, 48



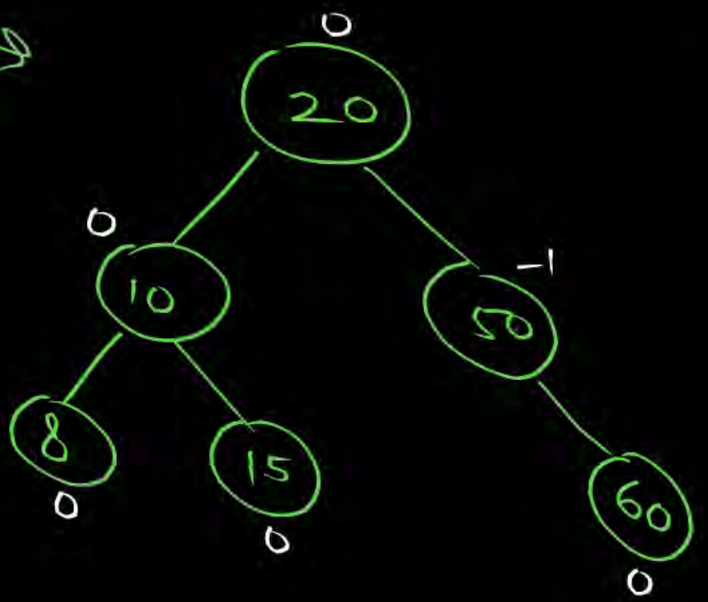
Keys : 50, 20, 60, 10, 8, 15, 32, 46, 11, 48



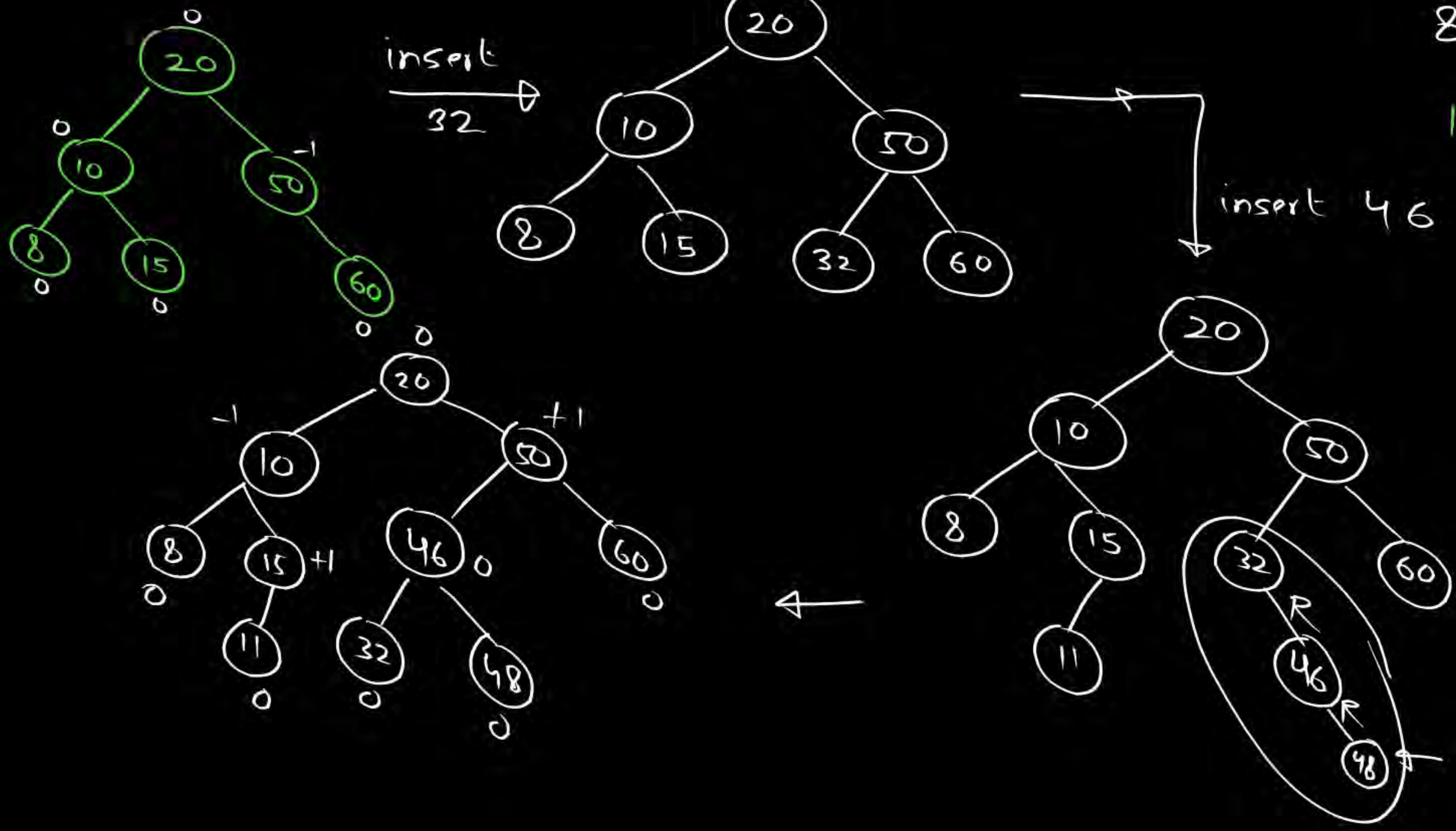
insert 15



L root R
8 10 20
root
10 20 50



Keys : 50, 20, 60, 10, 8, 15, 32, 46, 11, 48



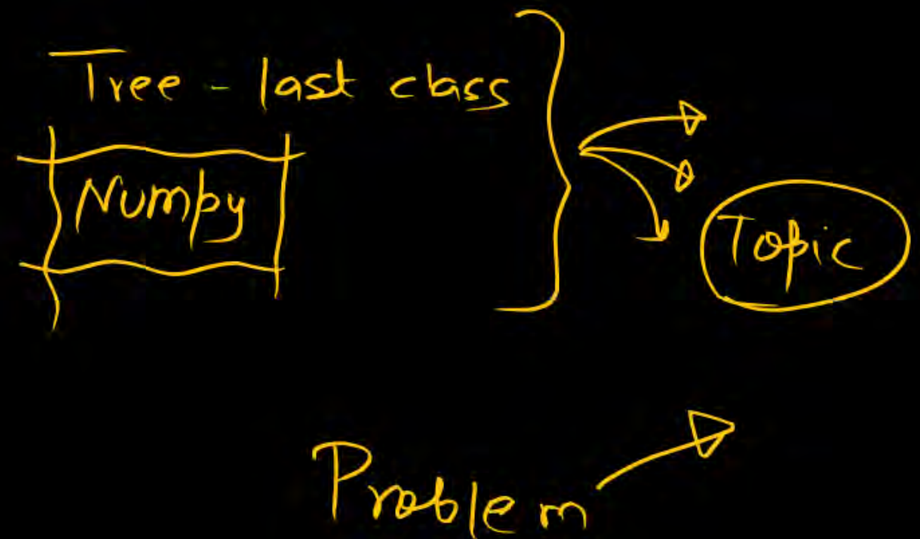
L root R
8 10 20
10 20 50

H, I, J, B, A, E, C, F, D, G, K, L

Step by step

30min

t.me/PWpankajsirP


Tree - last class
Numpy
Topic
Problem

THANK - YOU