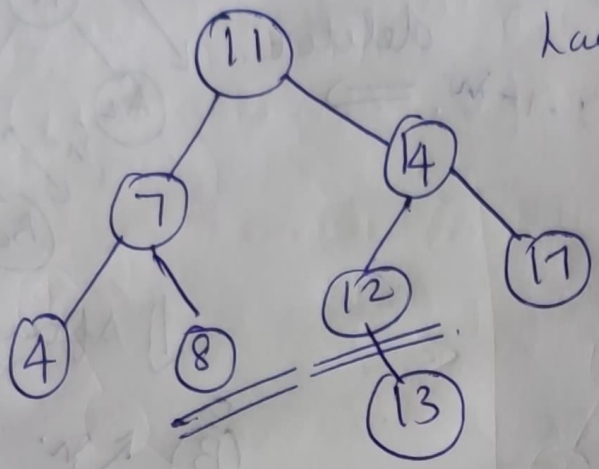
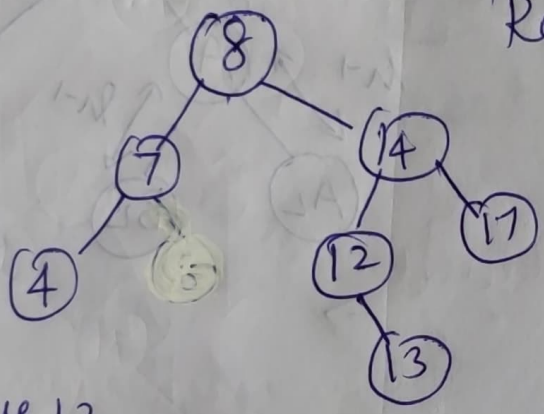
Remove 11

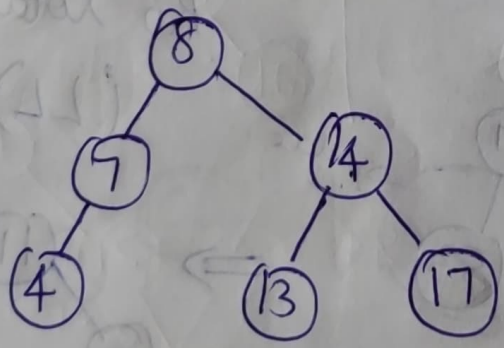
Small in right or Large in left.



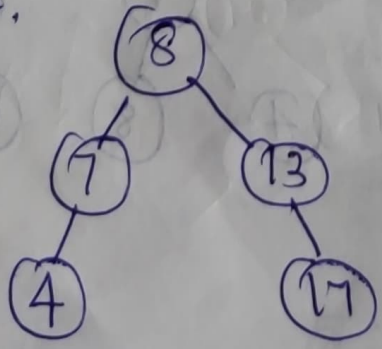
Rotation free deletion.

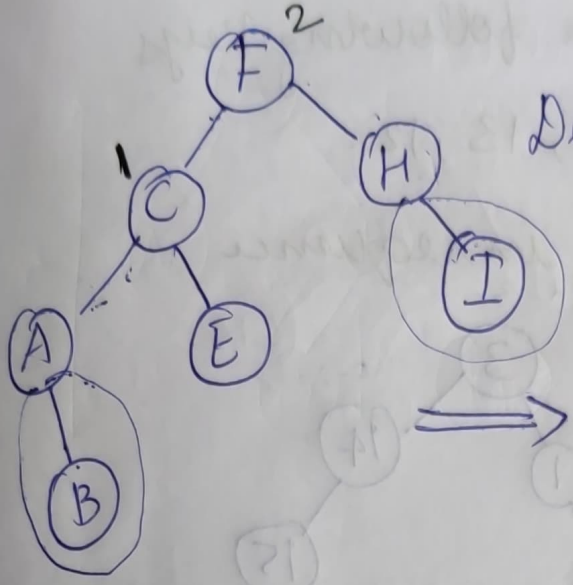


Remove 12.



Remove 14.

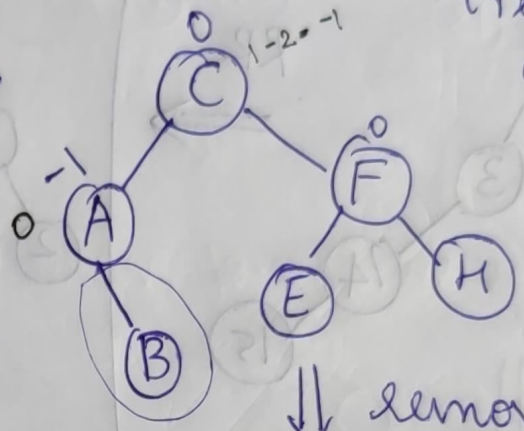




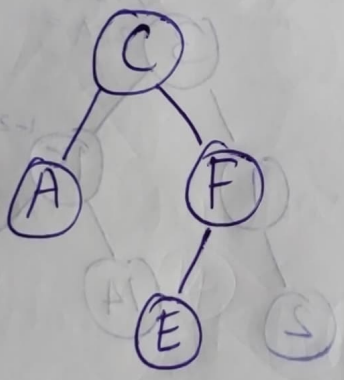
Delete [I, B, H] in sequence

↓

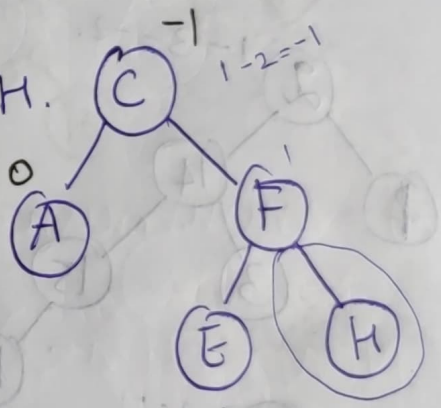
R_1. in same tree after one another.



⇓ remove B.



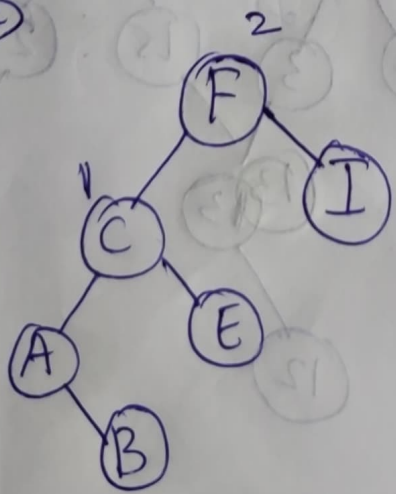
remove H.



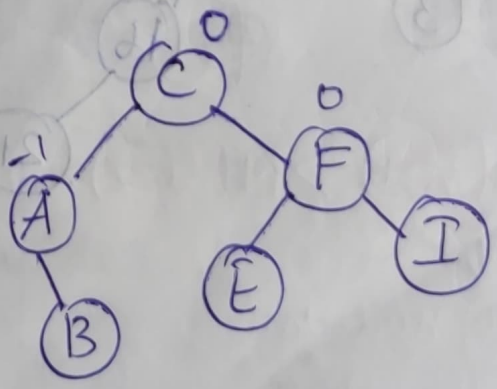
Construct (Independent) deletion.

B ⇒ no rotation.

H ⇒



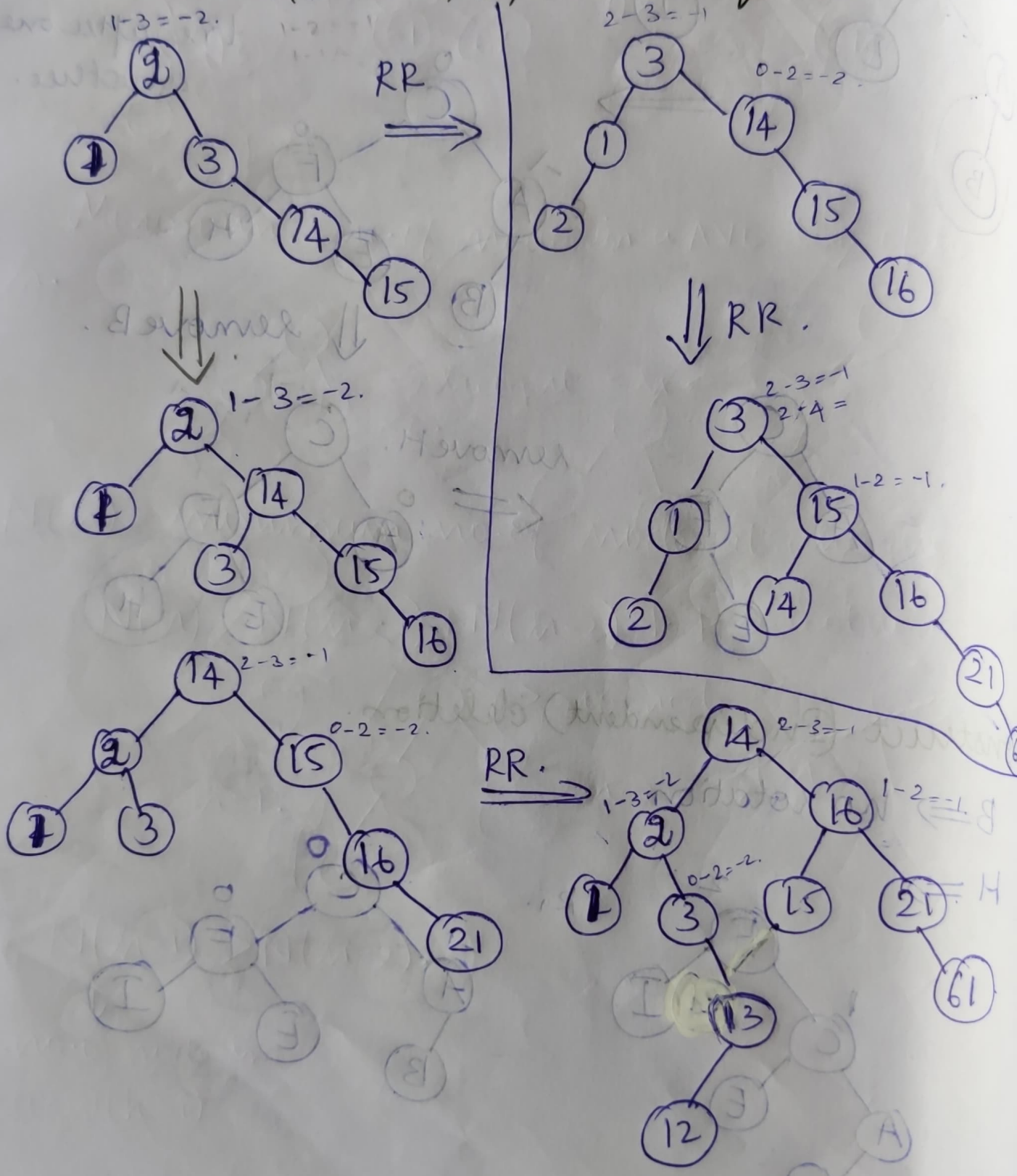
R_1

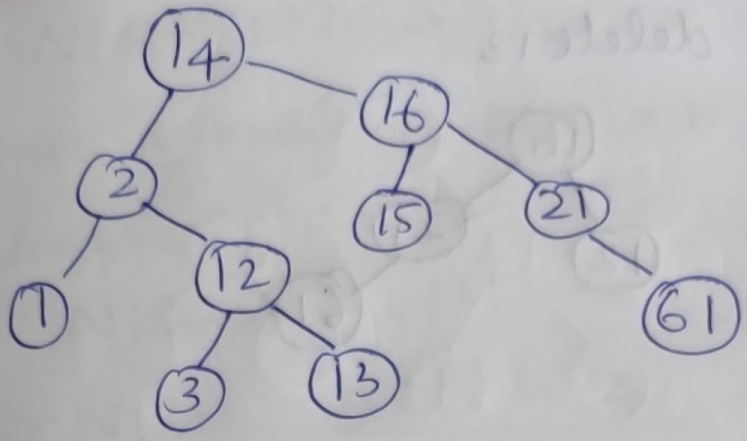


Construct AVL tree with following keys.

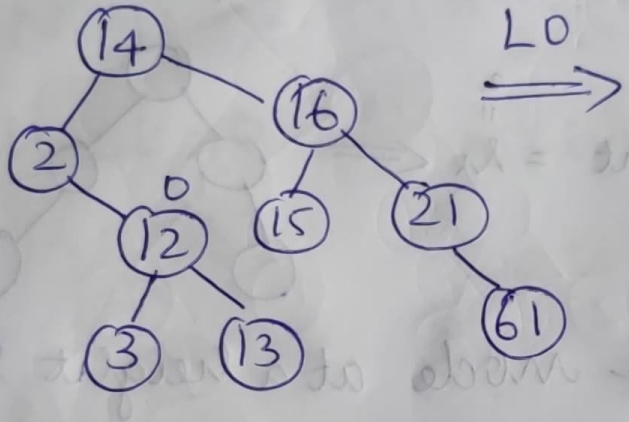
1, 2, 3, 14, 15, 16, 21, 61, 13, 12.

delete 1, 14, 15, 3, 2, 13 in sequence

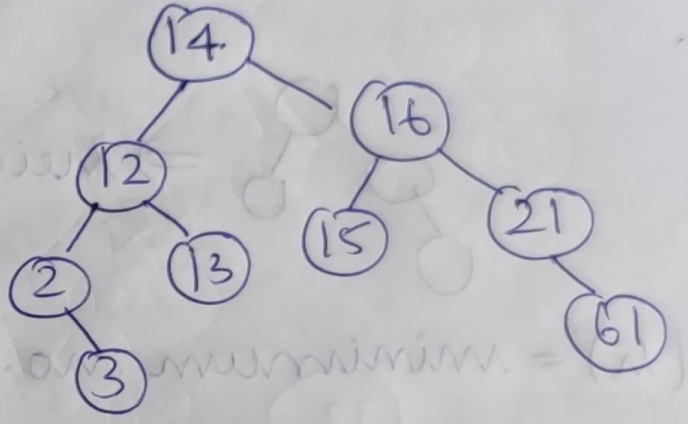




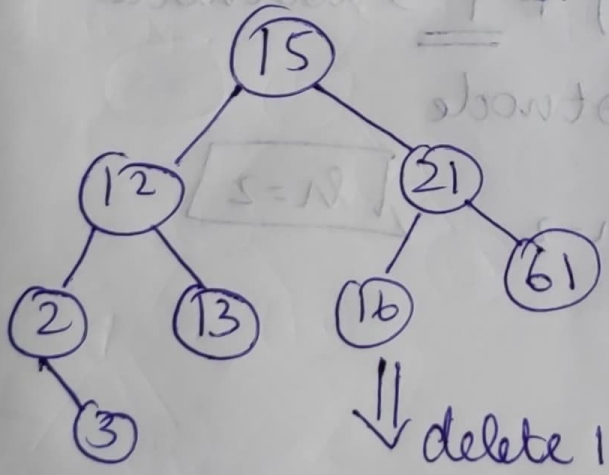
delete 1.



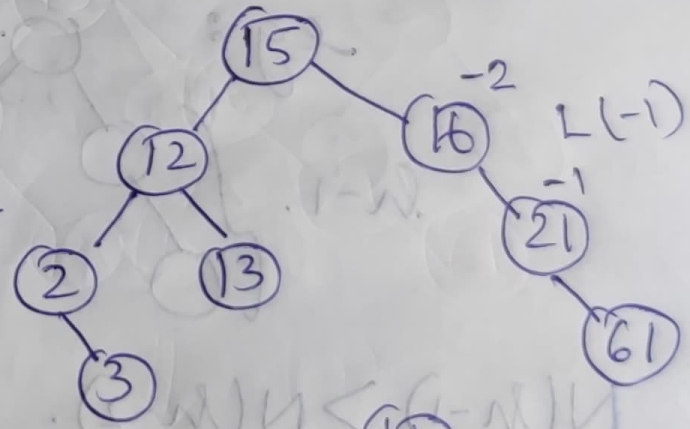
LO

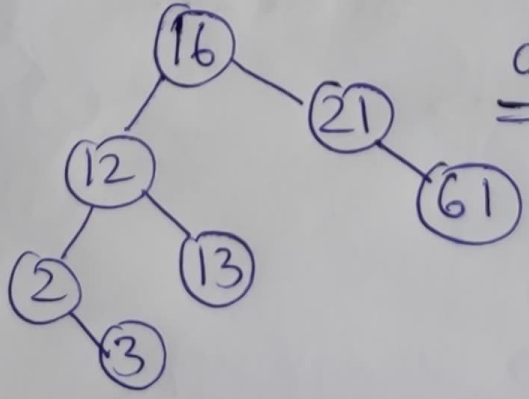


⇓ delete 14.

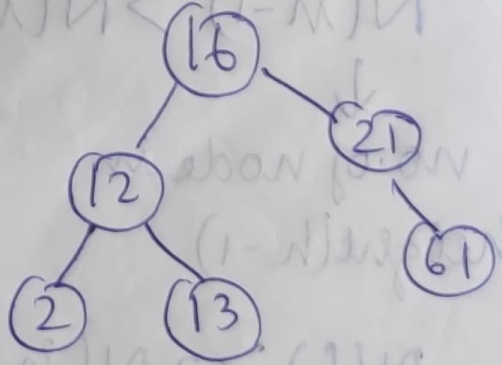


⇓ delete 15.

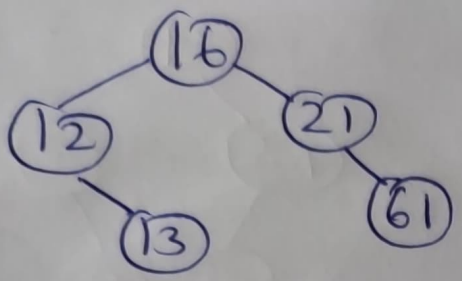




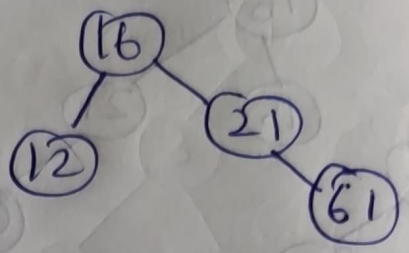
delete 3.



delete 2

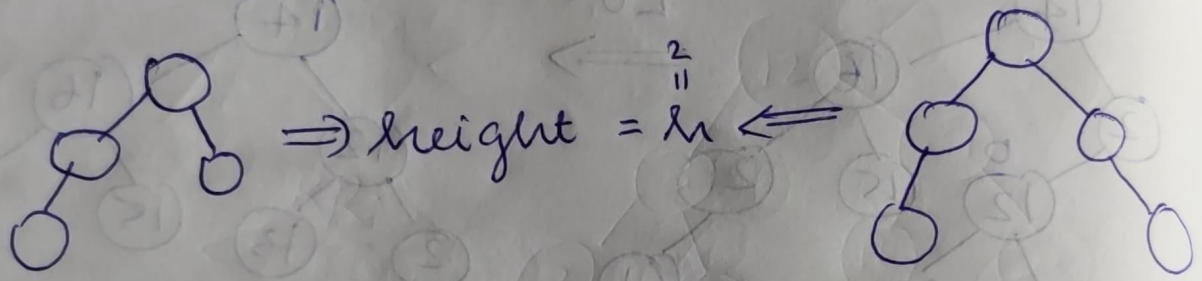


delete 13



Complexity Analysis

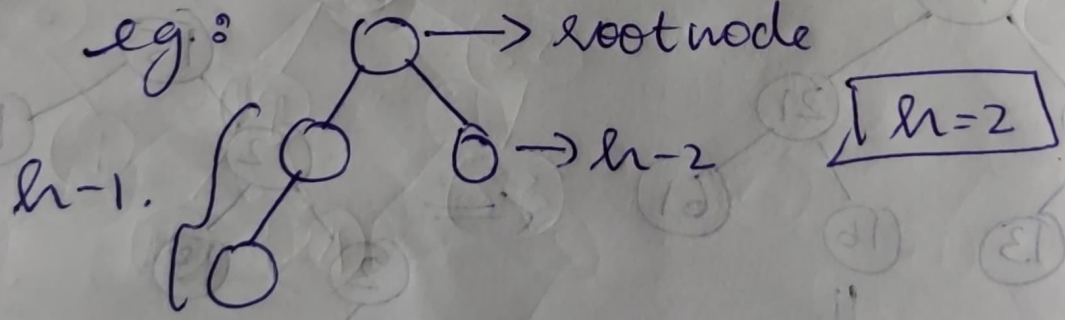
Minimum no. of nodes in a AVL tree of height h.



N(h) = minimum no. of node at a height h.

$N(h) = N(h-1) + N(h-2) + \underline{\underline{1}}$ → rootnode.

eg:



$N(h-1) > N(h-2)$.

↓

no. of node in height(h-1)

$N(h) \geq 2N(h-2) + 1$.

$N(h) > 2N(h-2)$

Initial condition $N(0) = 1$.

$h = h-2$

$N(h-2) > 2N(h-4)$

$N(h) > 4N(h-4)$ $\quad 2 \cdot 2N(h-4)$

$N(h) > 2 \cdot 2 \cdot 2 N(h-6) \Rightarrow N(h) > 2^3 N(h-6)$

$N(h) \geq 2^i N(h - 2 \times i)$.

$h - 2i = 0$.

$i = h/2$.

$N(h) = 2^{h/2} N(0)$

$\log N(h) > h/2$.

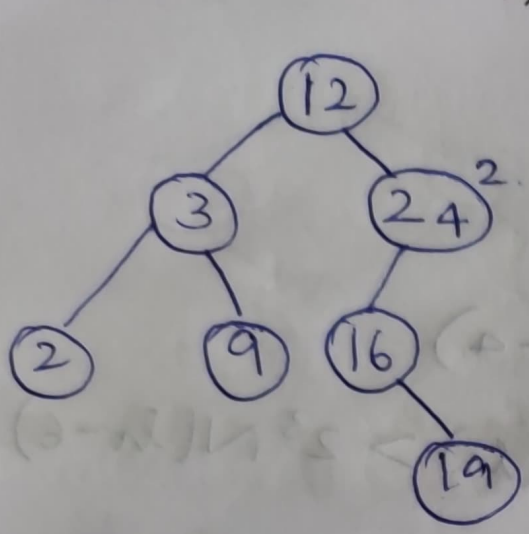
$h < 2 \log N(h)$.

n = min. no of nodes at a height h

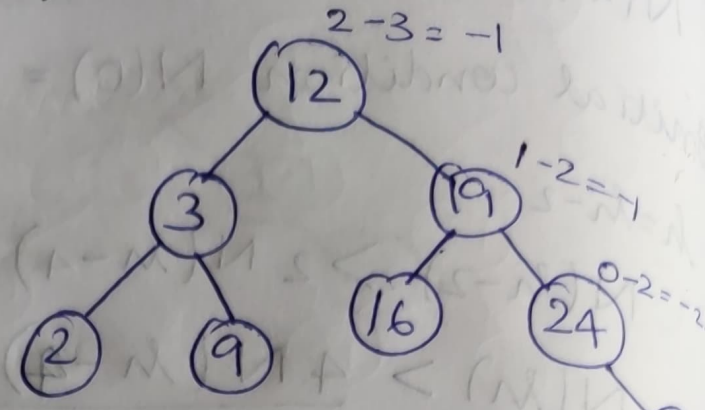
$h < 2 \log_2 N$

$h \simeq O(\log n)$.

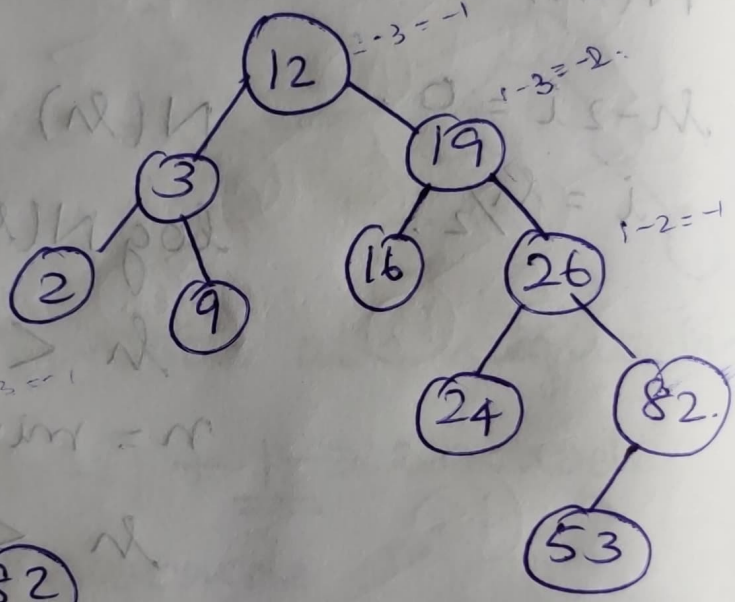
HW . 12, 24, 3, 9, 16, 2, 19, 26, 82, 53 (Insert).



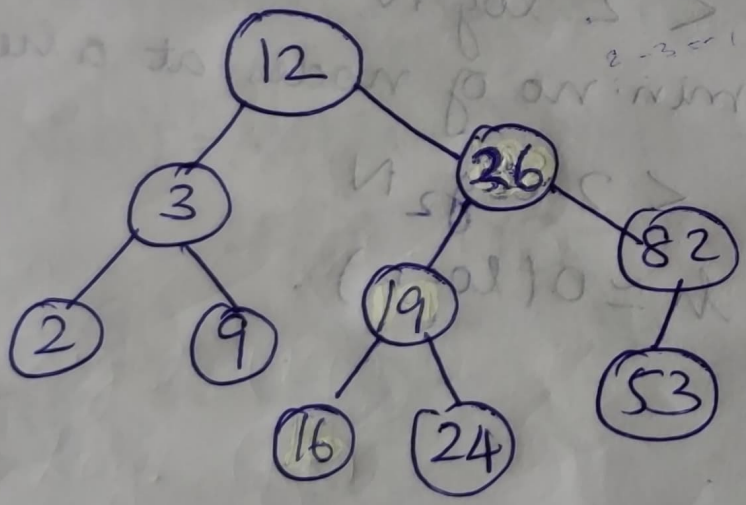
LR



RR.

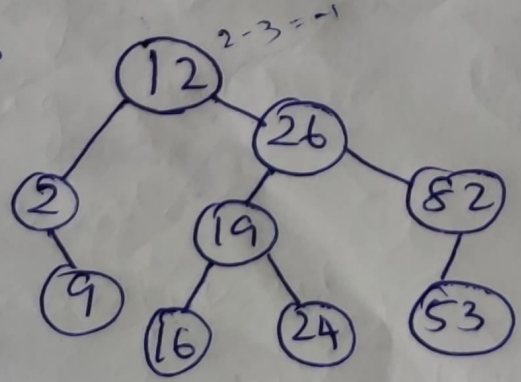


RR.

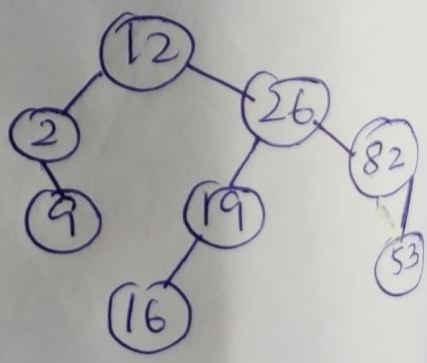


Delete 3, 24, 82, 9, 19.

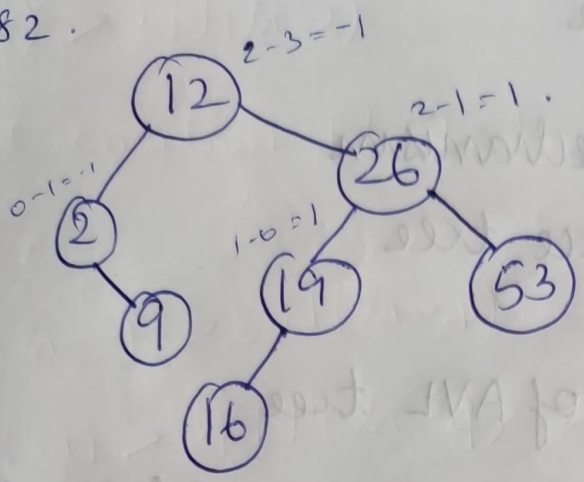
delete 3



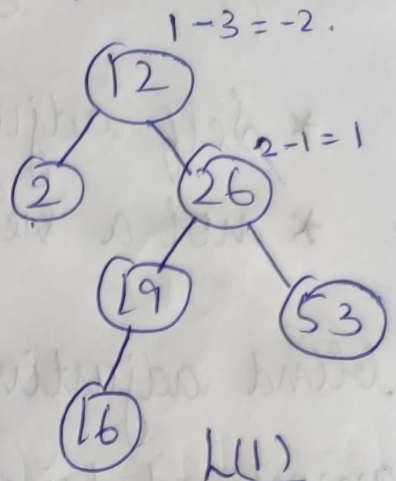
delete 24.



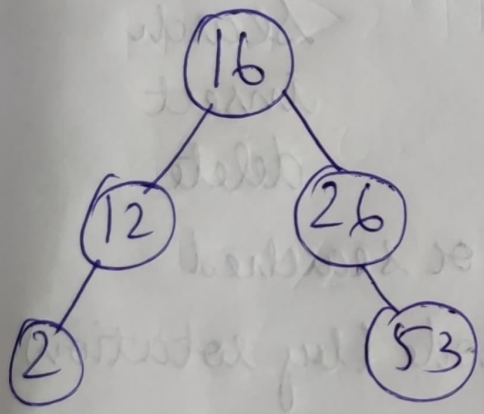
delete 82.



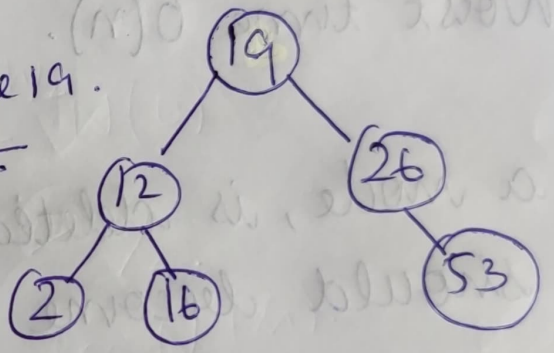
delete 9.



LL(1)
RL.



delete 19.



Splay tree

* Self adjusting mechanism
* not a help balance tree

blind adjusting version of AVL tree

amortized time for all operation is $O(\log n)$ (together).

Worst time $O(n)$.

search
insert
delete

if a node, is deleted, inserted or searched it should become the root node (by rotation)

Rotation

Zig - Left (LL)

Zag - right. (RR)

Zig Zig (LL)

Zag Zag. (RR)

Zig Zag (LR)

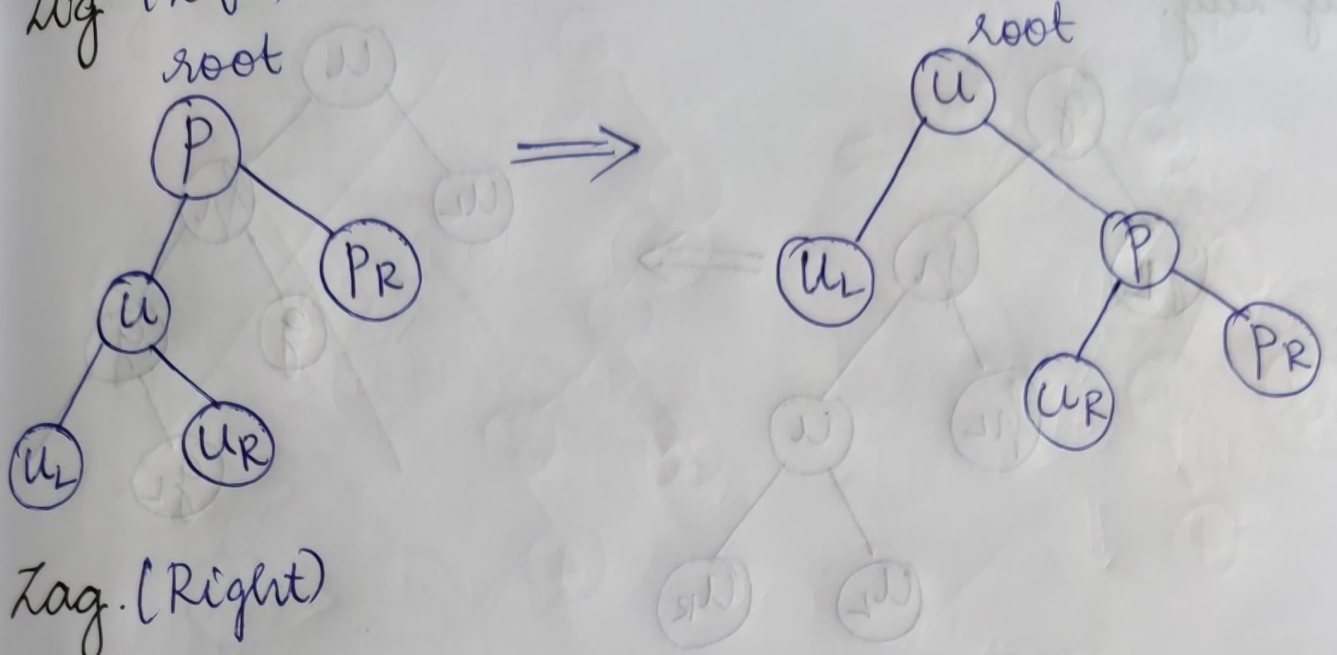
Zag Zig (RL)

p = parent

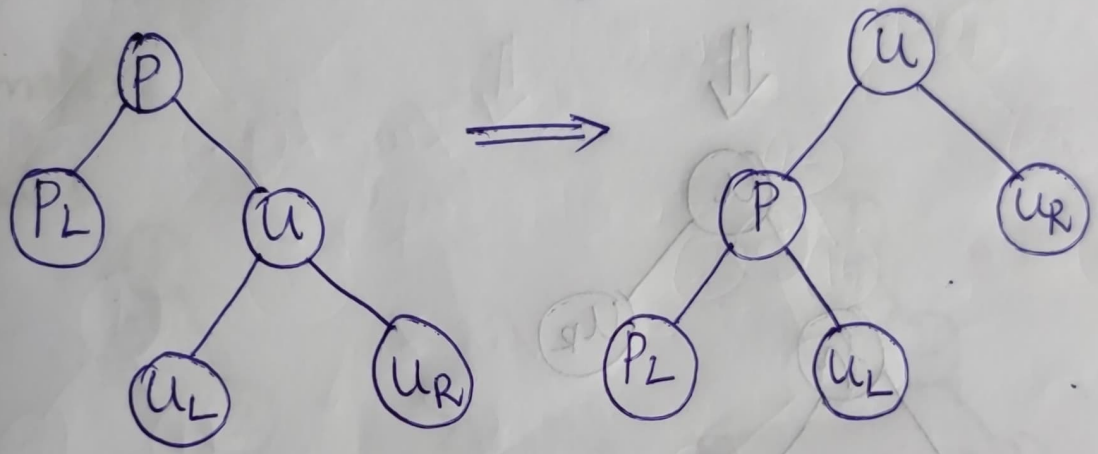
g = grandparent

u = node (which under go splay)

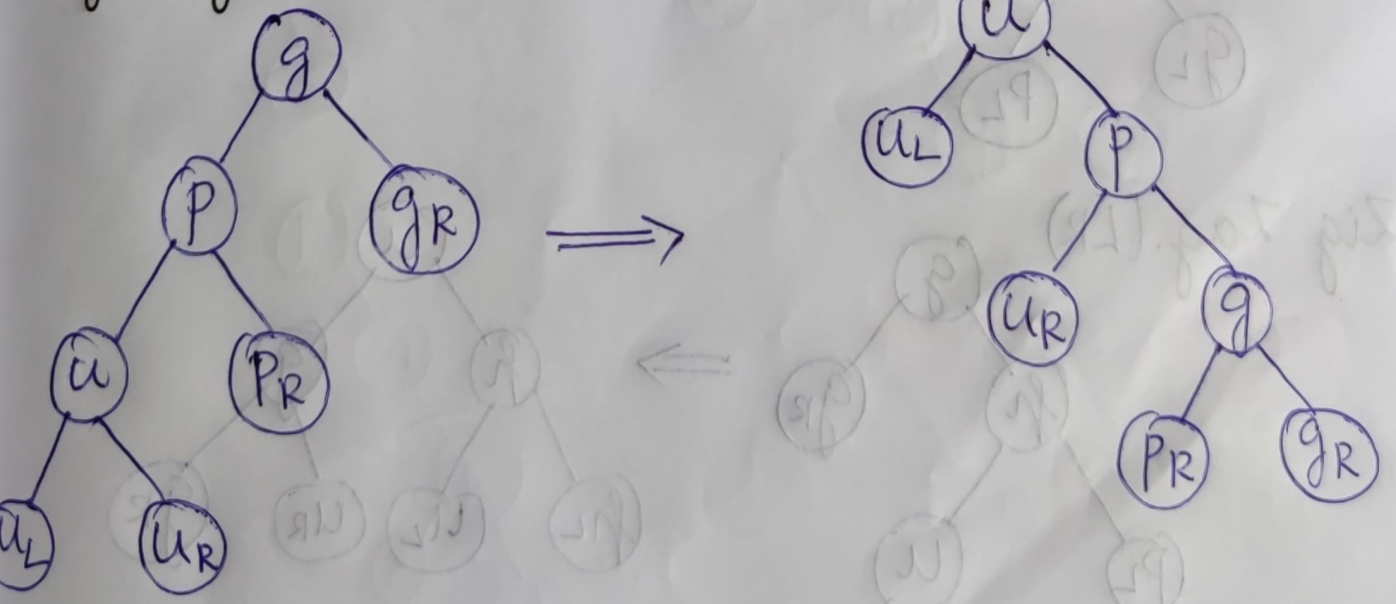
Zig (left)



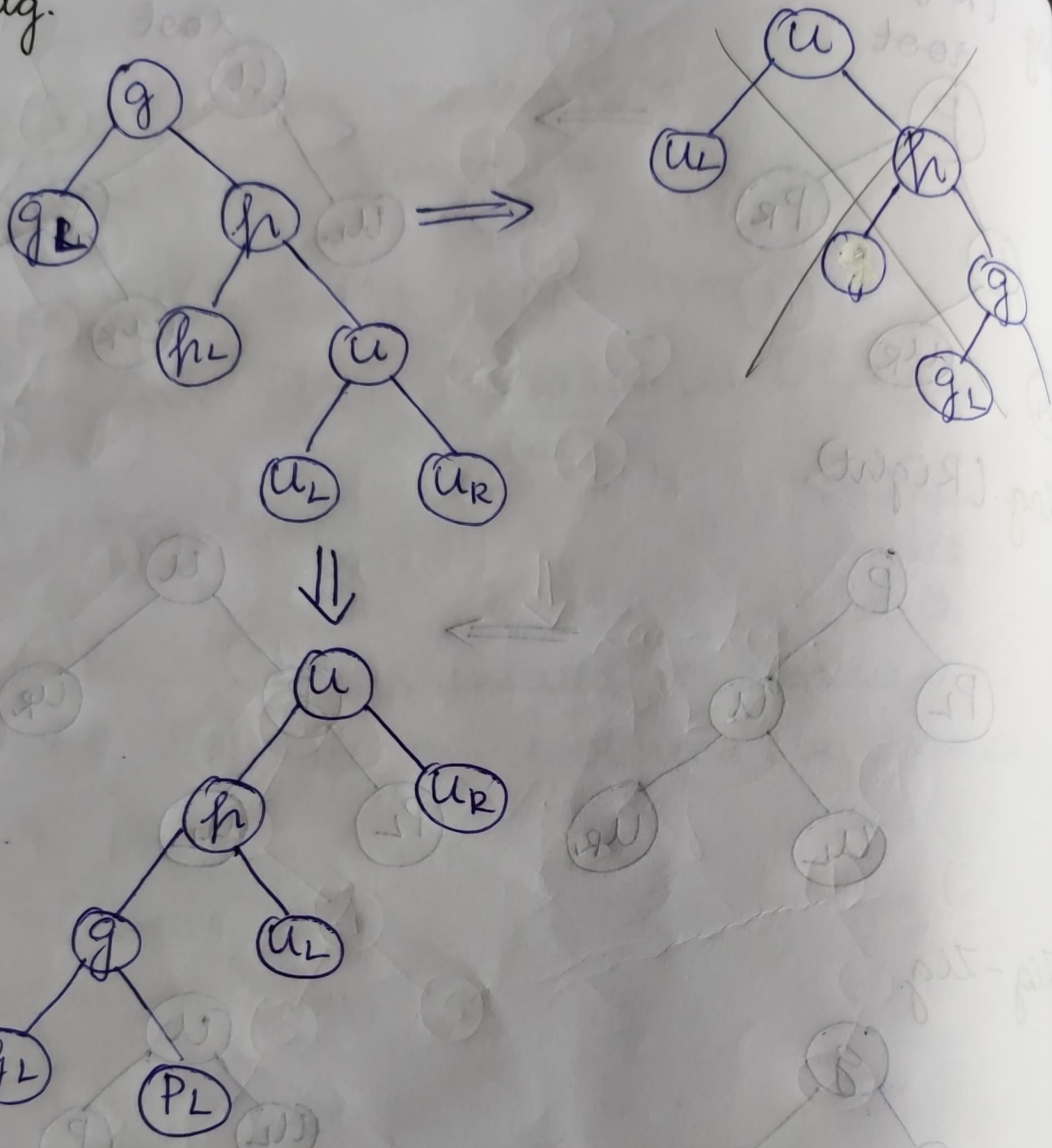
Zag. (Right)



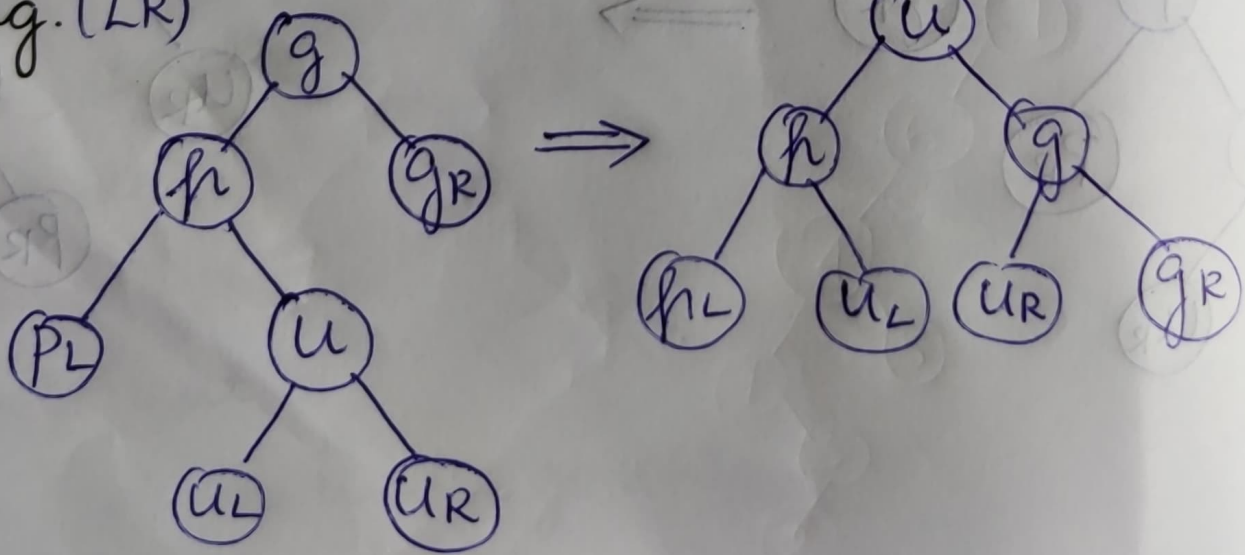
Zig-Zig.



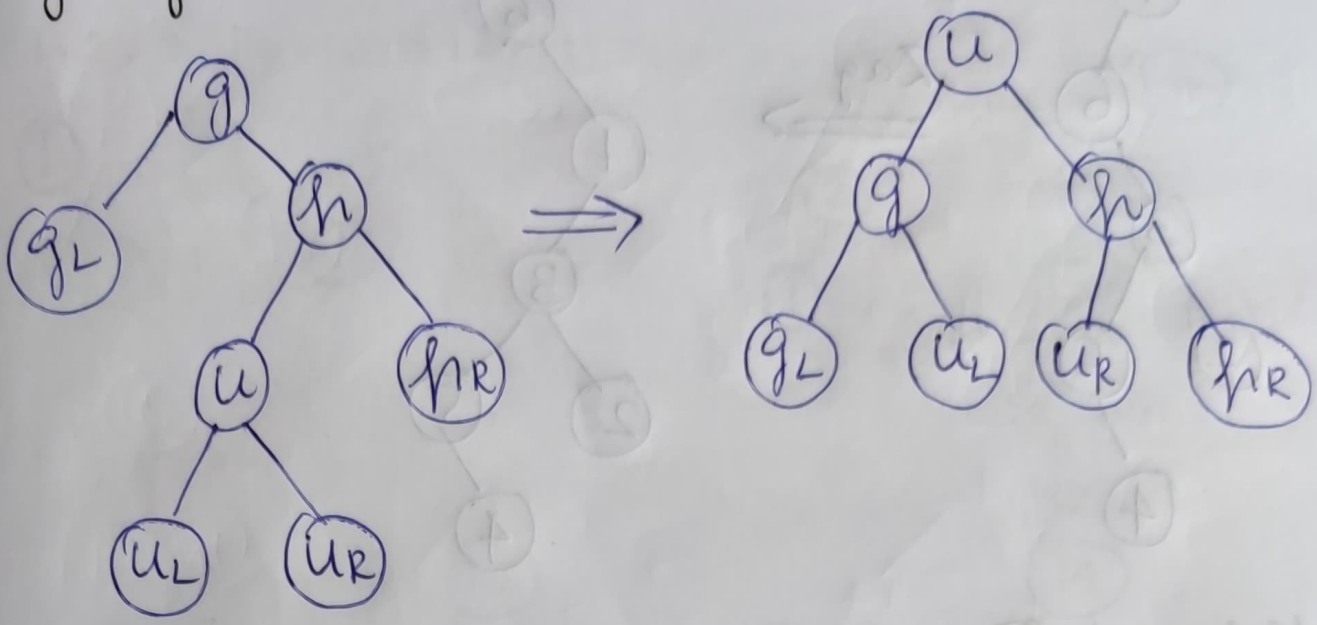
Zag-Zag.



Zig Zag. (LR)

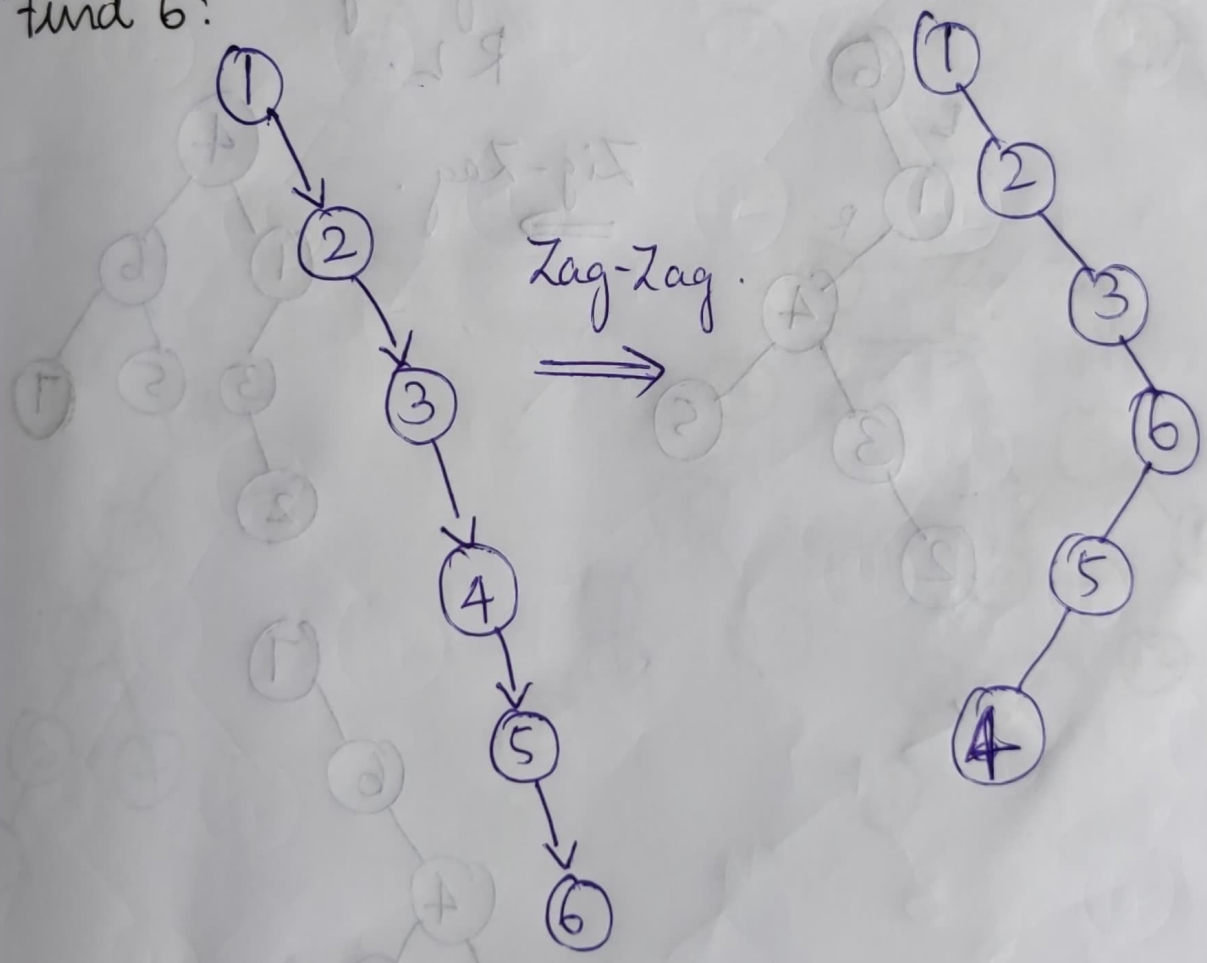


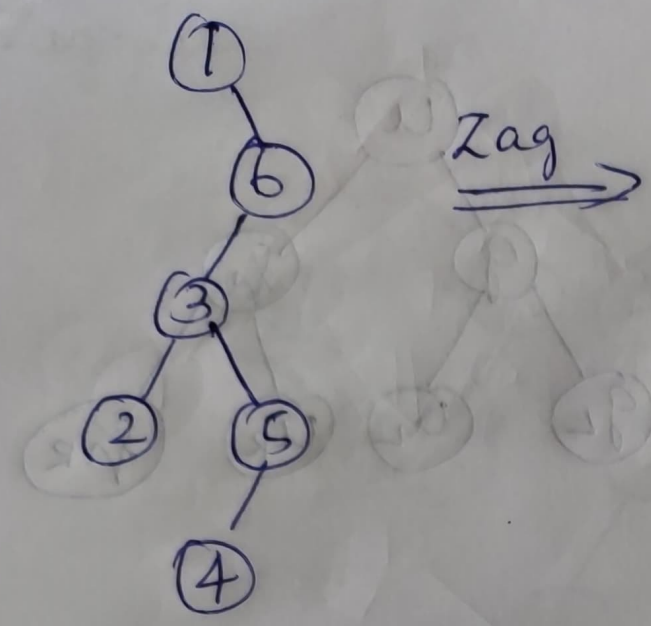
Zag Zig. (RL)



Example:

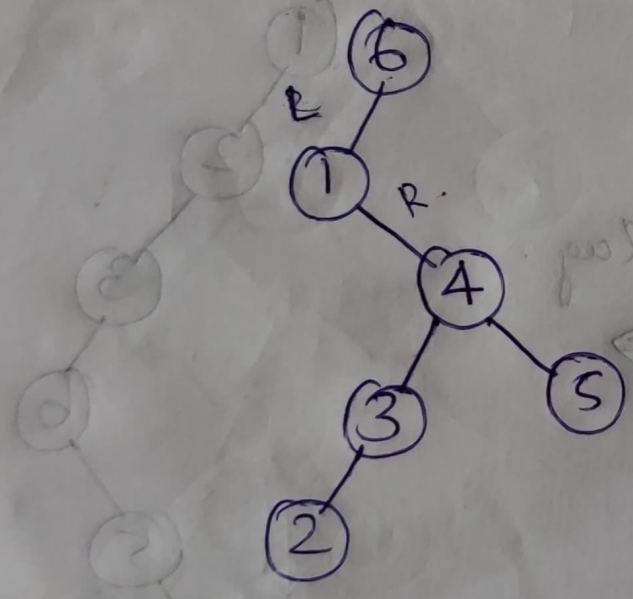
Find 6:



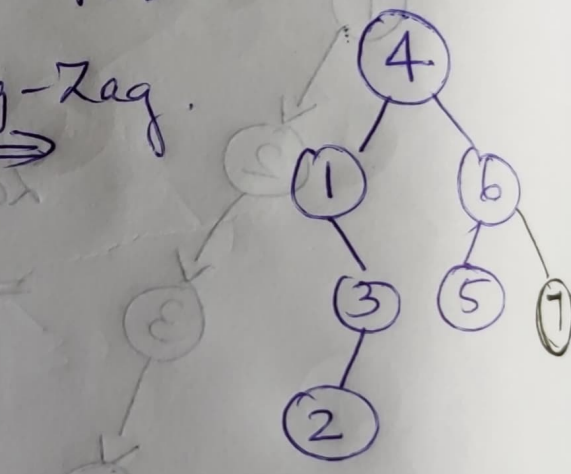


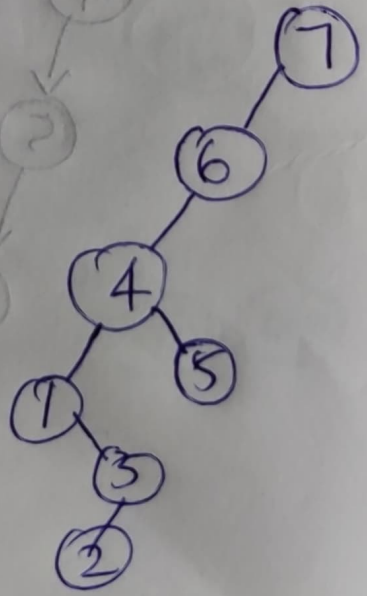
Splay at (4)

⇓ Zag-Zig.
RL.



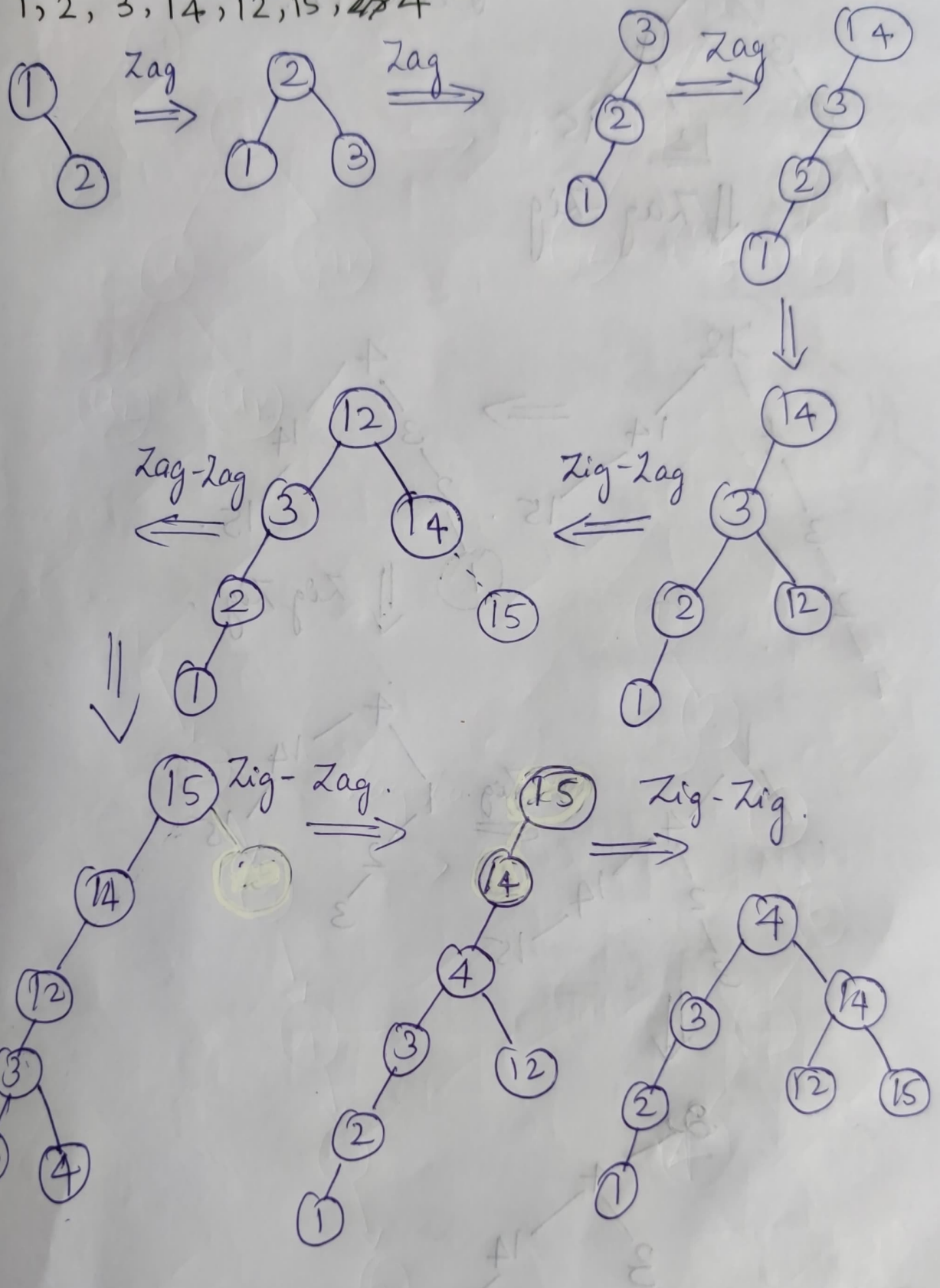
Zig-Zag.



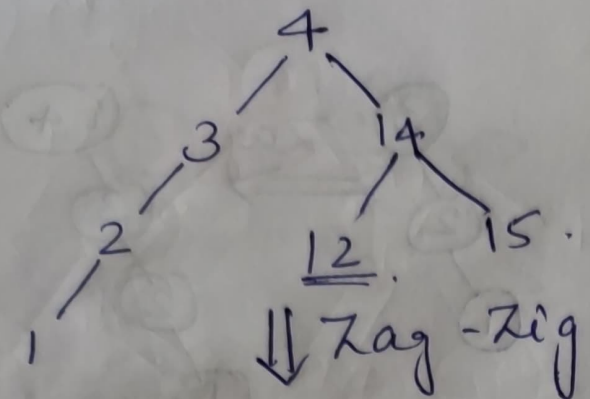


Construct Splay tree

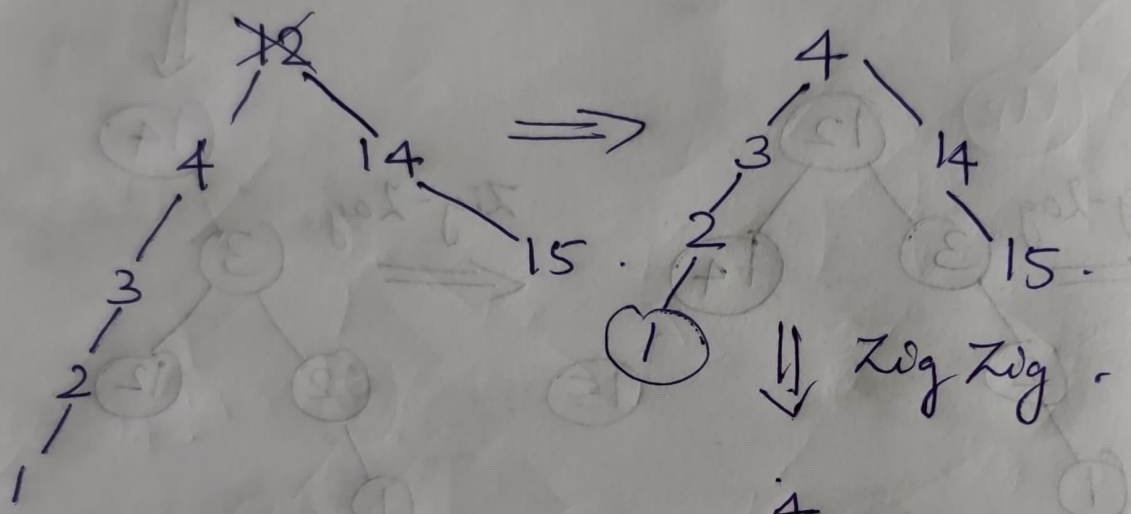
1, 2, 3, 14, 12, 15, ~~25~~ 4

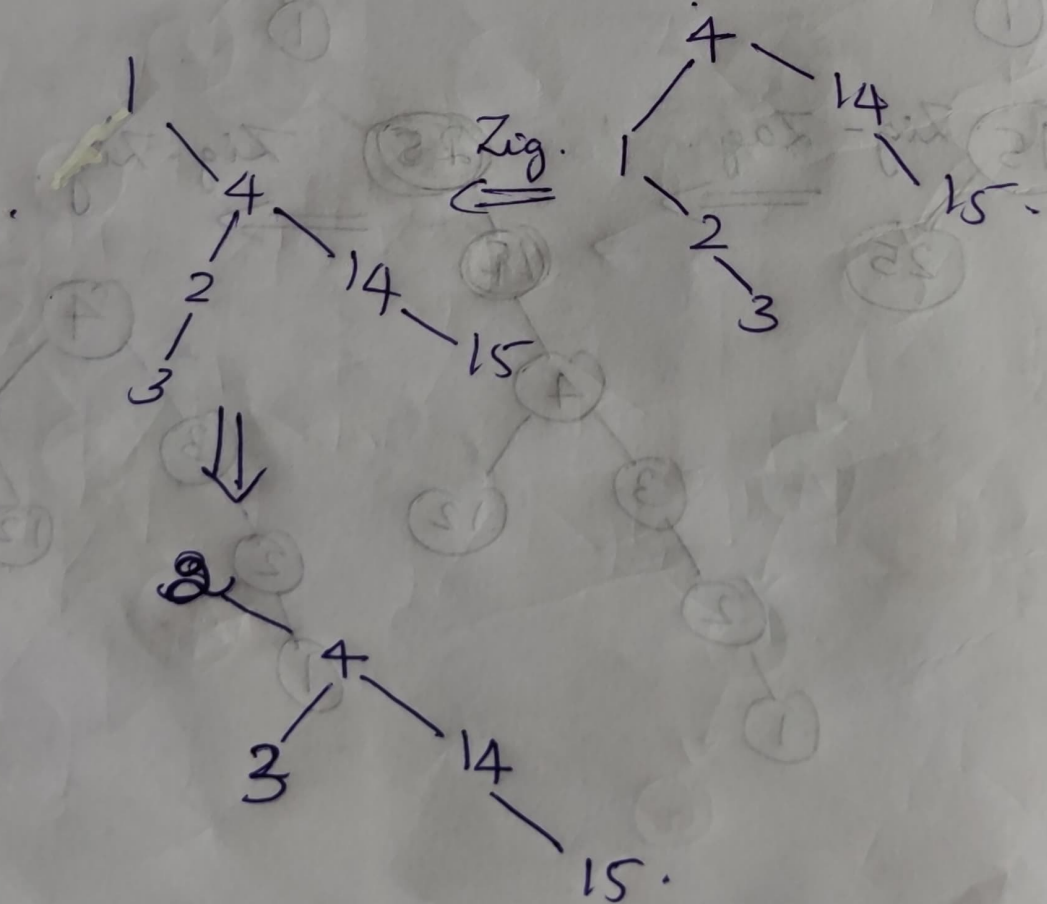


Delete 12.



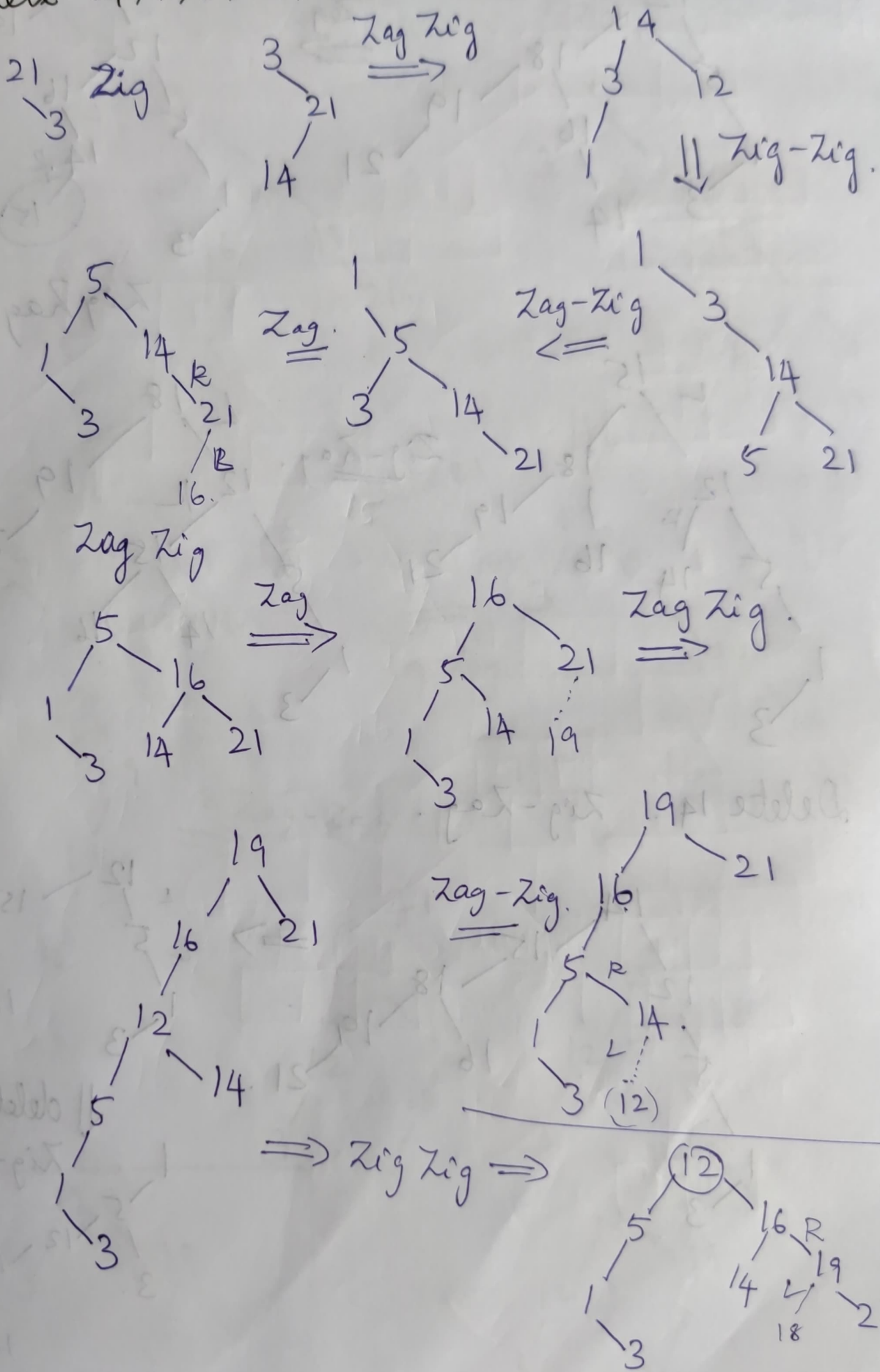
⇓ Zag - Zig



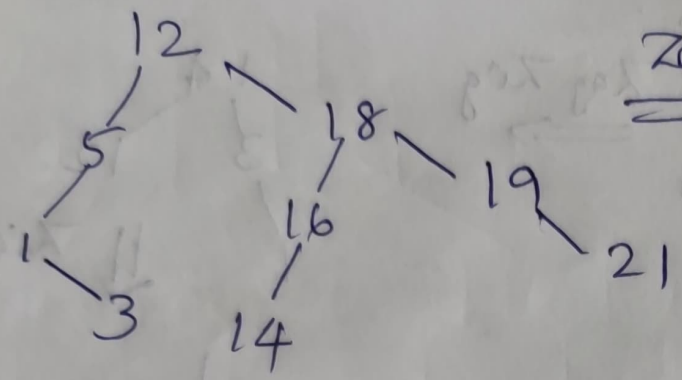


Insert 21, 3, 14, 1, 5, 16, 19, 12, 18, 15.

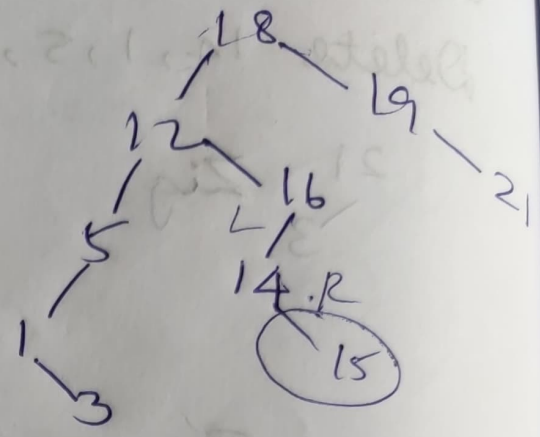
Delete 14, 1, 5, 21, 15.



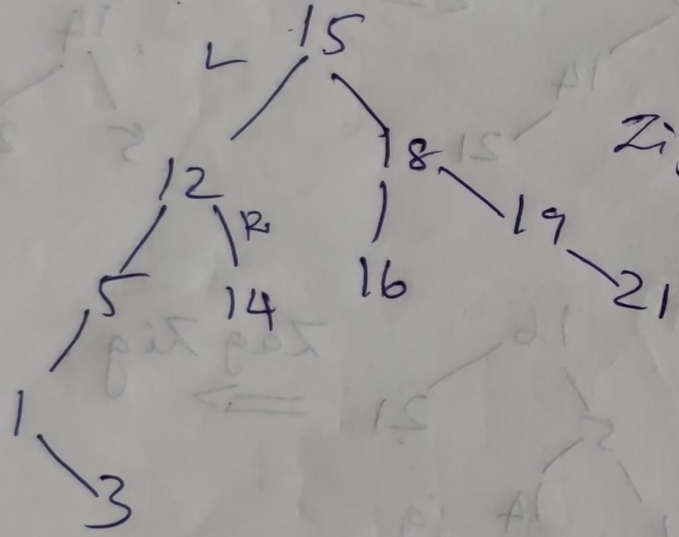
Zag Zig



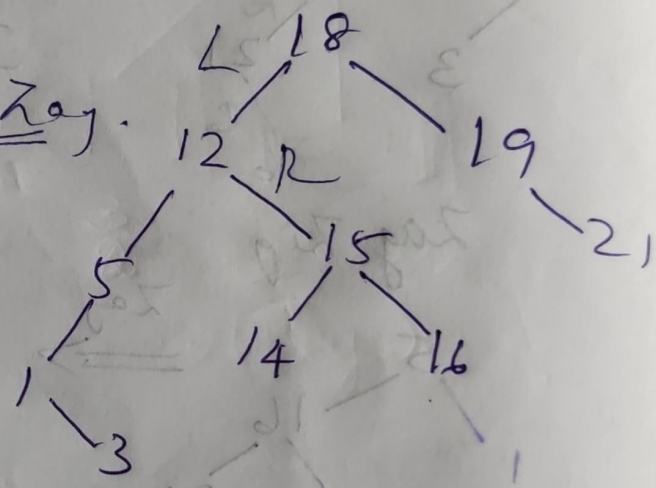
Zag



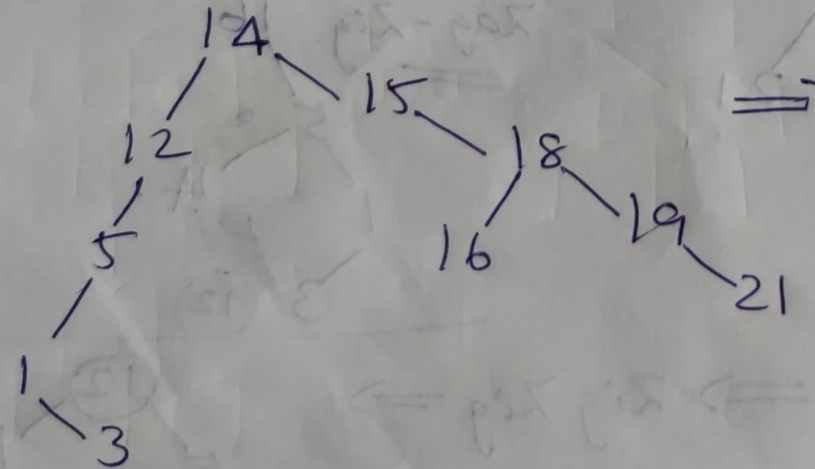
Zig Zag.

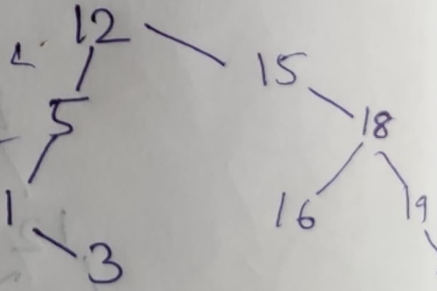


Zig-Zag.

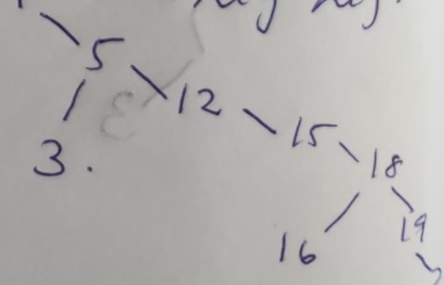


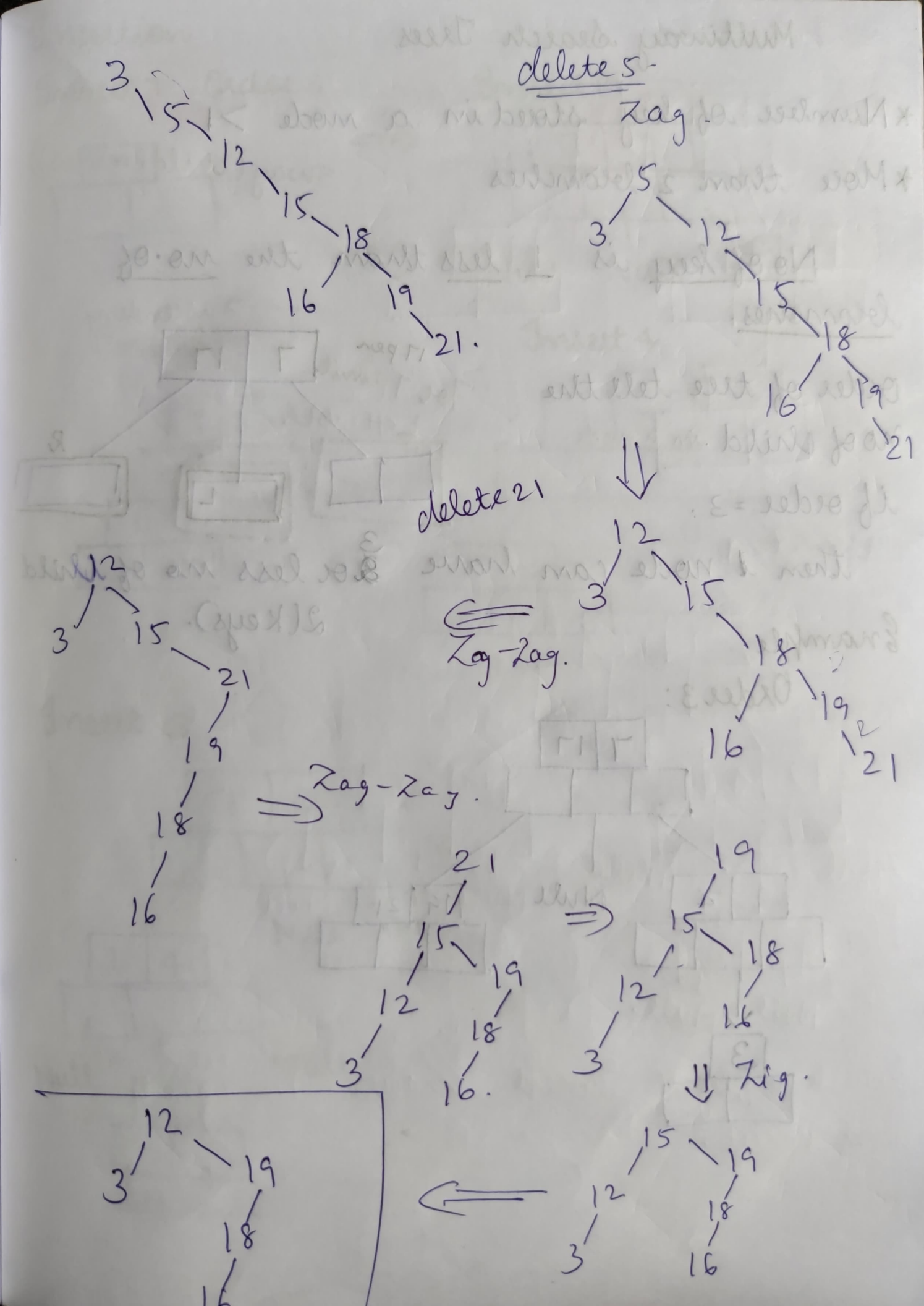
Delete 14. Zig-Zag.





⇓ delete 1
Zig-Zig.

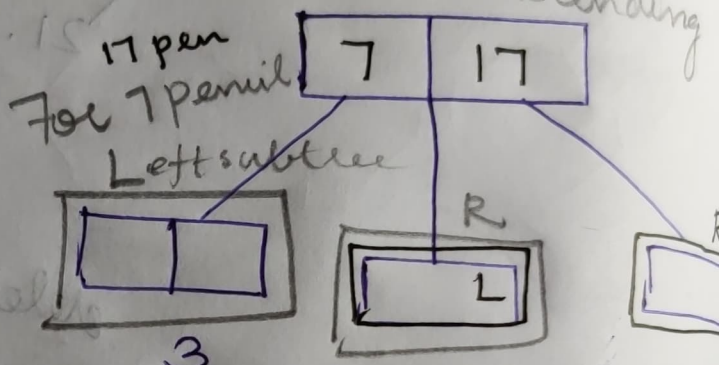



delete 5-
Zag.
delete 21
Zag-Zag.
Zag-Zag.
Zig.

Multiway Search Trees

* Number of key stored in a node >1
* More than 2 branches

<u>No of key</u> is <u>1 less</u> than the <u>no. of branches</u>.



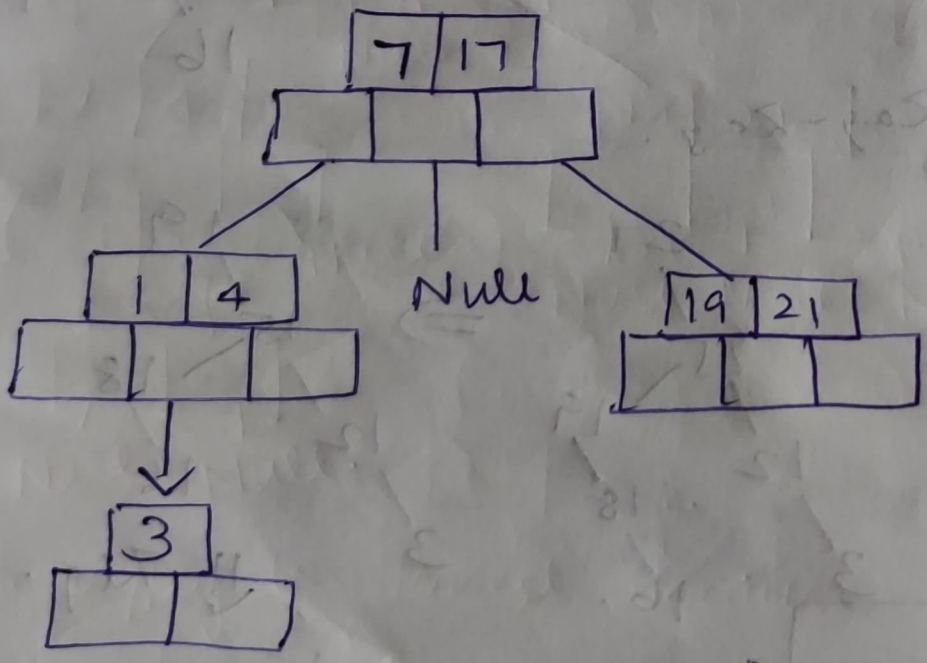
Order of tree tell the no of child in a node.

if order = 3.

then 1 node can have 3 or less no of children 2(keys).

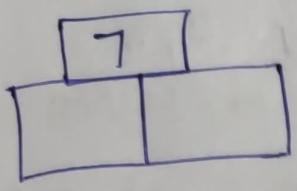
Example

Order 3:



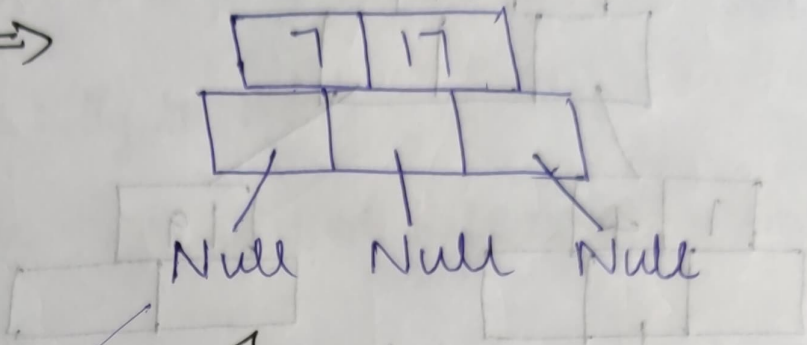
Insertion

Insert 7. Order 3

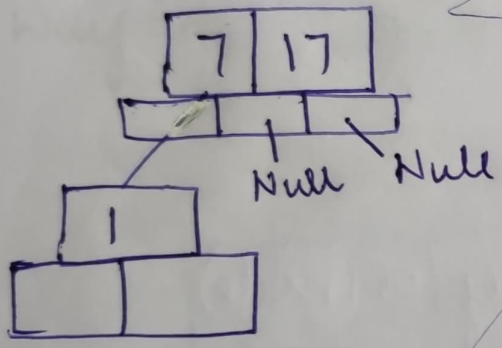


⟹

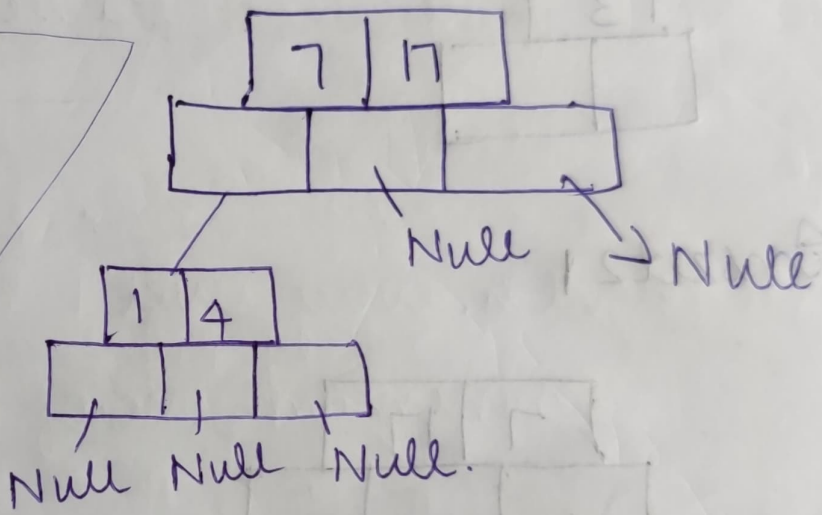
Insert 17



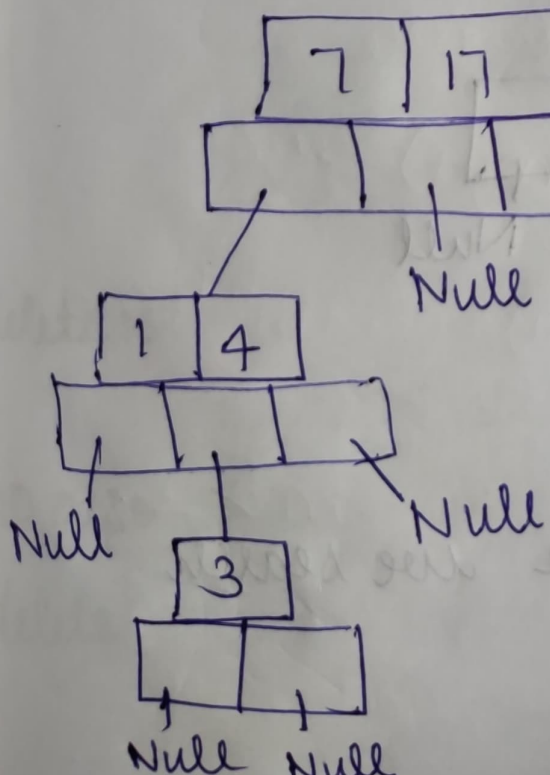
Insert 1



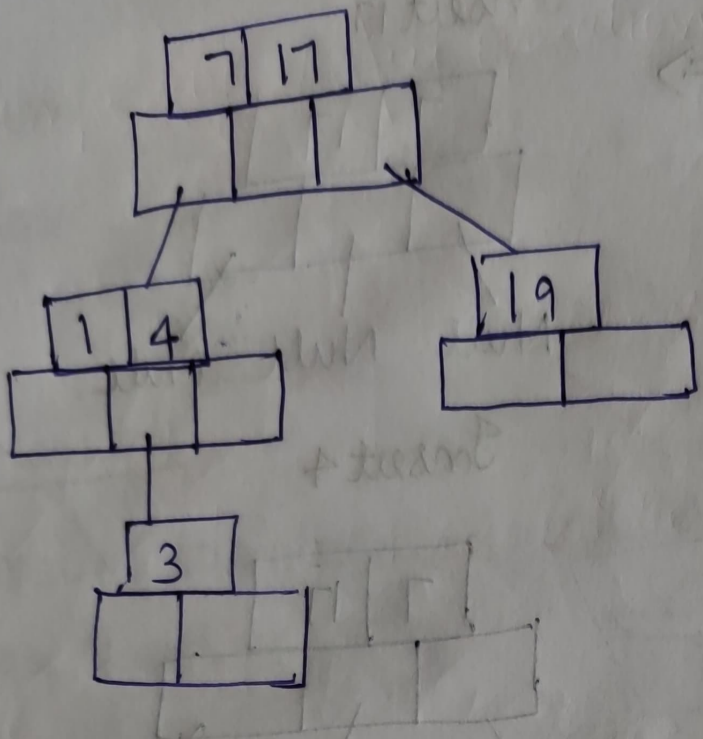
Insert 4



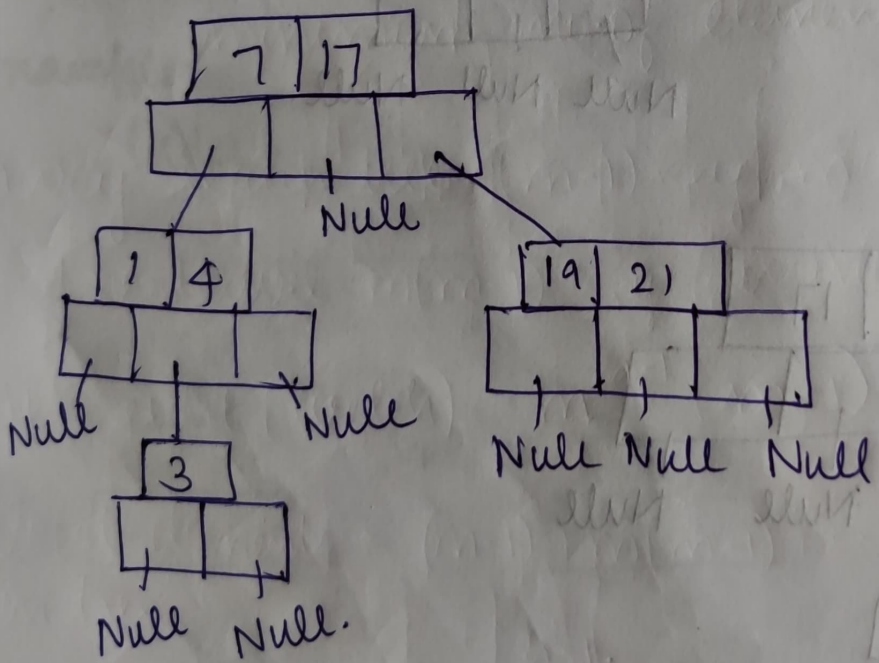
Insert 3



Insert 19.



Insert 21



Search is like to Insert because we search and insert.

Deletion.

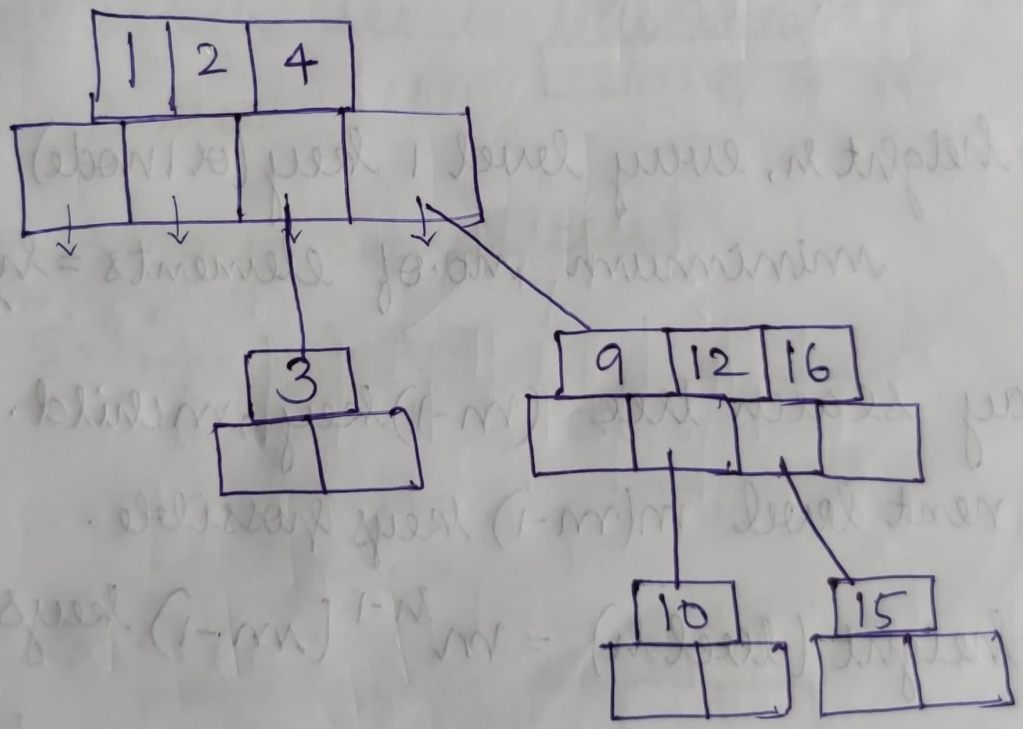
(i) deleting a leaf node and adjust pointers

(ii) only L/R subtree: (lly to BST)

(iii) Both L & R subtree greatest of left or smallest of right.

ascending.

m:way = 4 4, 2
1, 2, 3, 4, 3, 9, 12, 16, 10, 15.



delete 15

delete 15 its a leaf node.

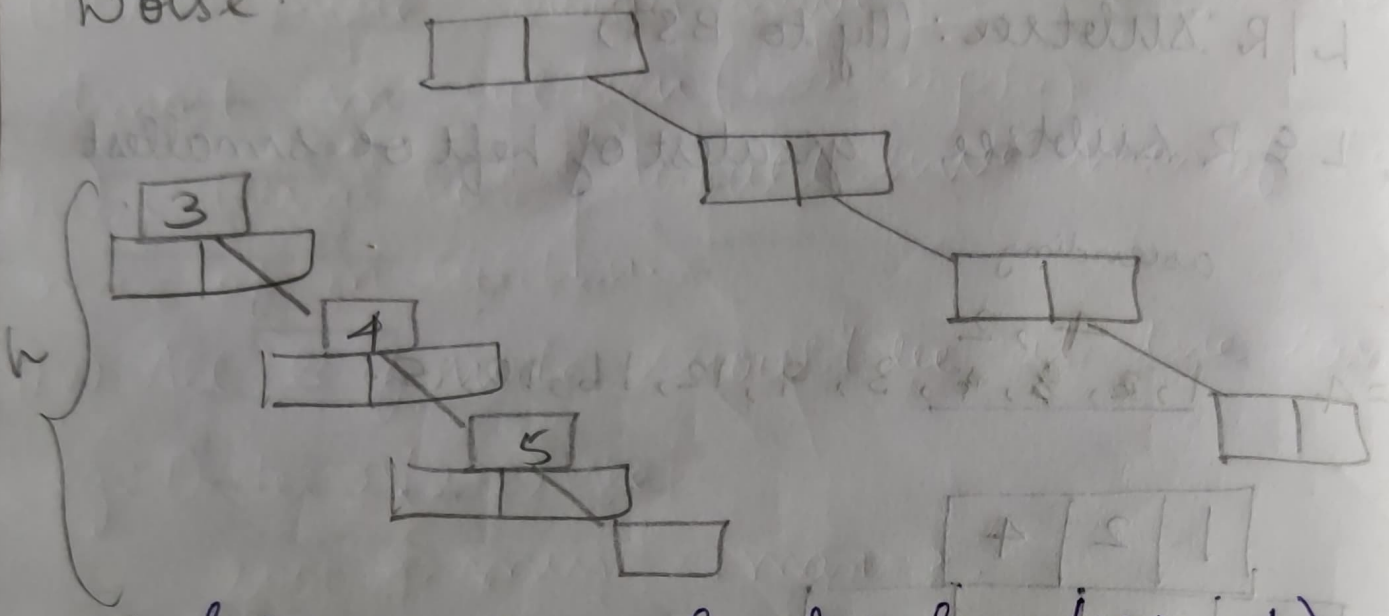
10 also deleted like that.

delete 4 and replace by 9

Analysis of M-way search trees

Complexity of any operation is $O(h)$.

Worse



1. height h, every level 1 key (or 1 node)
   minimum no. of elements $= h$ //.

2. m way search tree $(m-1)$ key, m child.
   at next level $m(m-1)$ keys possible.

At height (level h) $= m^{h-1}(m-1)$ keys

Total elements in tree $= (m-1) + m(m-1) + \cdots + m^{h-1}(m-$

$= m^h - 1$ //. $\quad m > 2$

$\frac{a(r^n-1)}{r-1} \quad \frac{1(m^{h-1+1}-1)}{m-1}$

m way search tree has n elements

Worse case: height of tree is n

O(n).

Best case

$$n = m^h - 1$$

$$\log_m n = h - \log_m m$$

$$\log_m (n+1) = h.$$

m way search tree is balanced is called as B tree

Construct 3 way search tree

5, 1, 4, 3, 6, 9, 2.