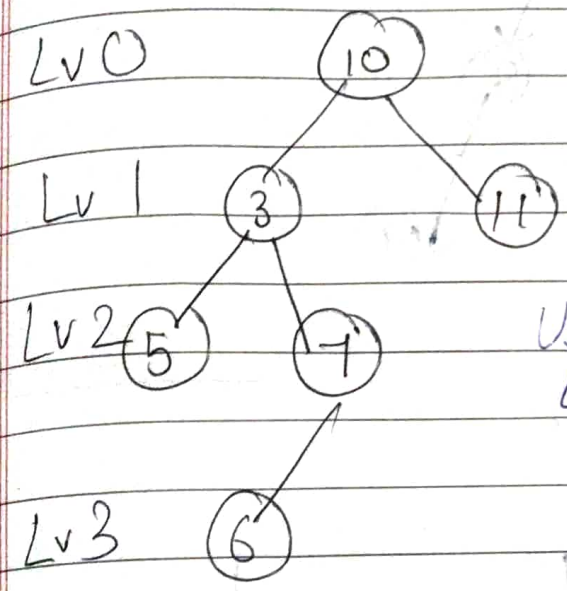
Trees.

RANKA
DATE / /
PAGE

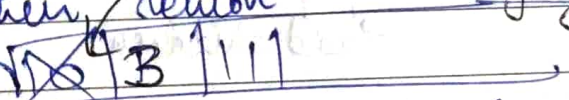
Level Order Traversal.



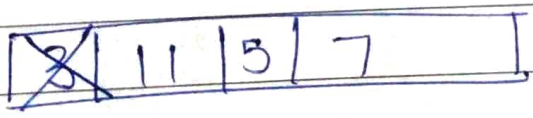
Using Queue → why?
↳ FIFO.

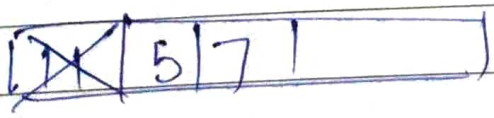
put 10 & then remove

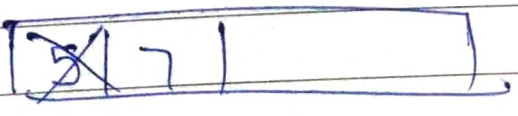
tum niklo, bacche chor jayiyo :D

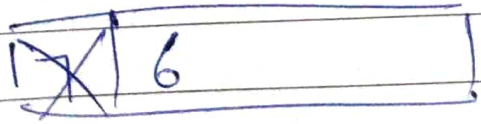


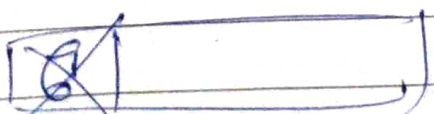
jab nikalloge then put its left & right child there.

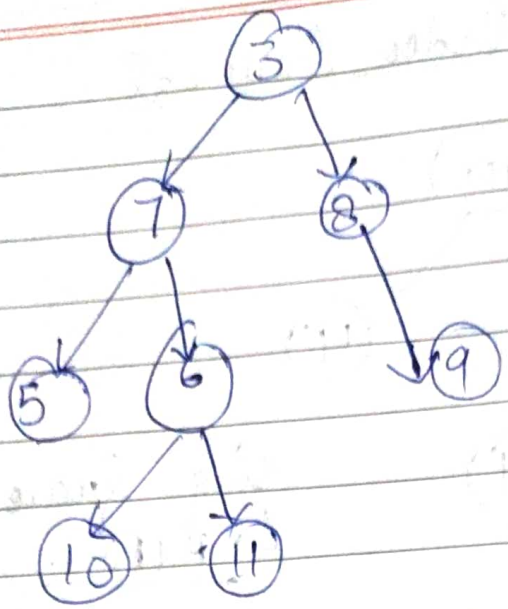












3

starting

| 3 | NULL |

while (!q.empty())

| ~~3~~ | NULL | 7 | 8 | NULL |

3 ko pakda

not NULL → print

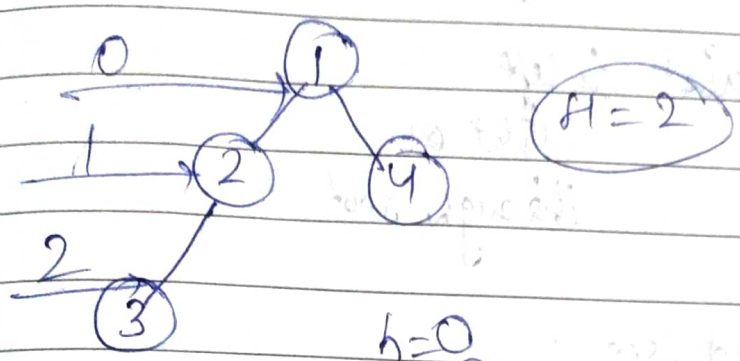
→ pop

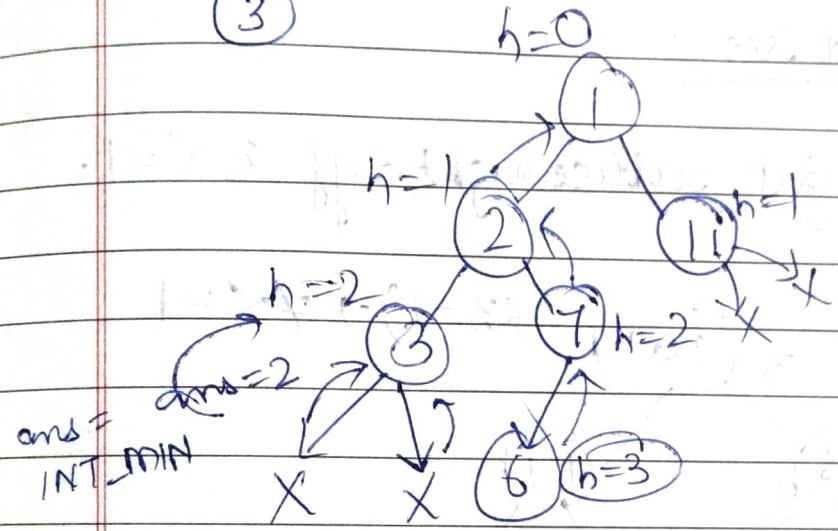
→ add children

NULL → new line

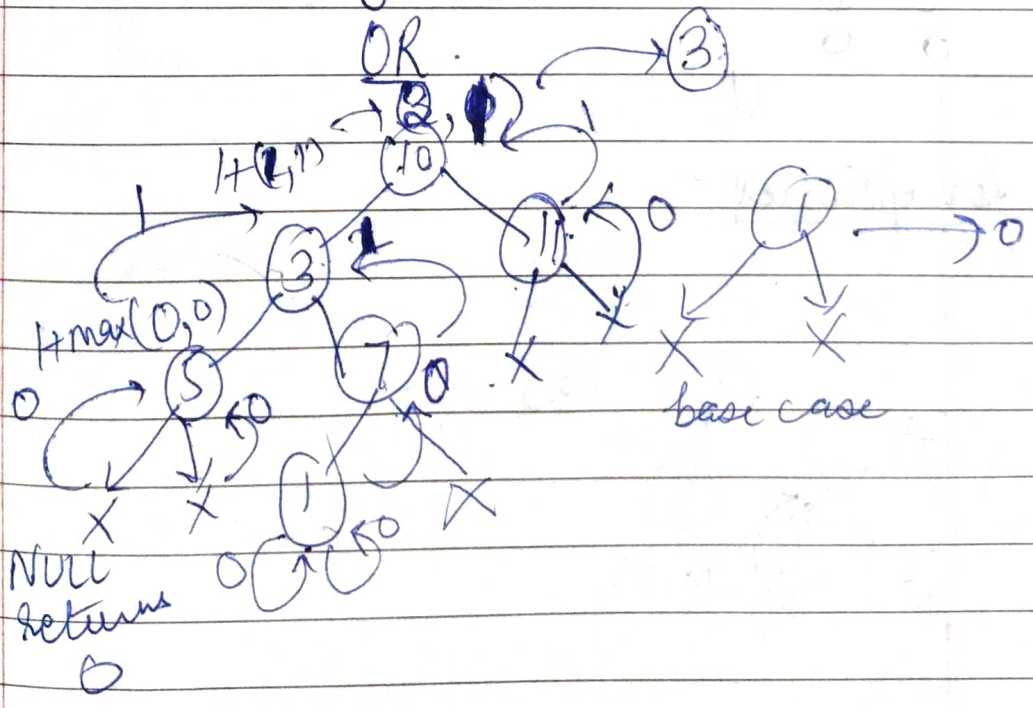
RANKA
DATE / /
PAGE

Height of tree
(max depth)





On reaching NULL → Traversal completed



RANKA
DATE / /
PAGE

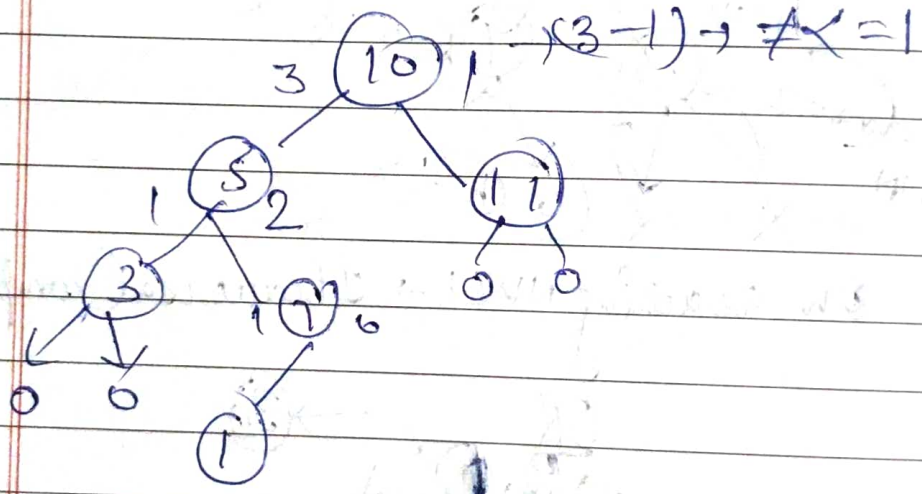
Diameter of tree :

length of longest path b/w any 2 nodes.

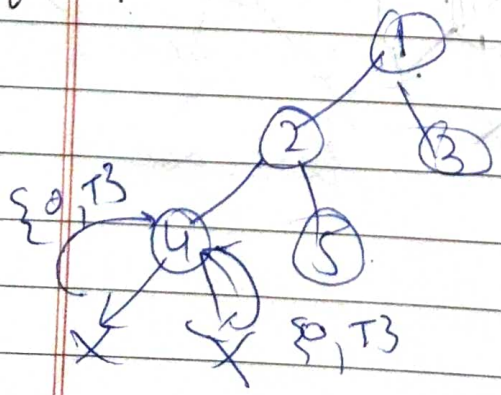
can exist in left
right or
through root

Balanced Tree :

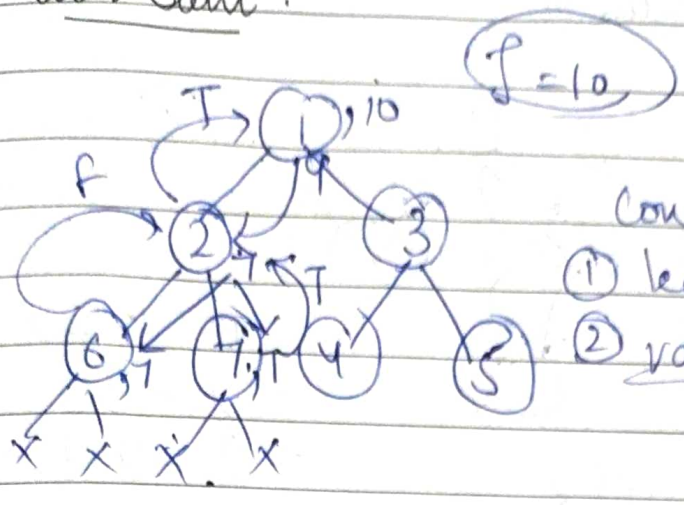
left & right subtree height diff is <= 1.

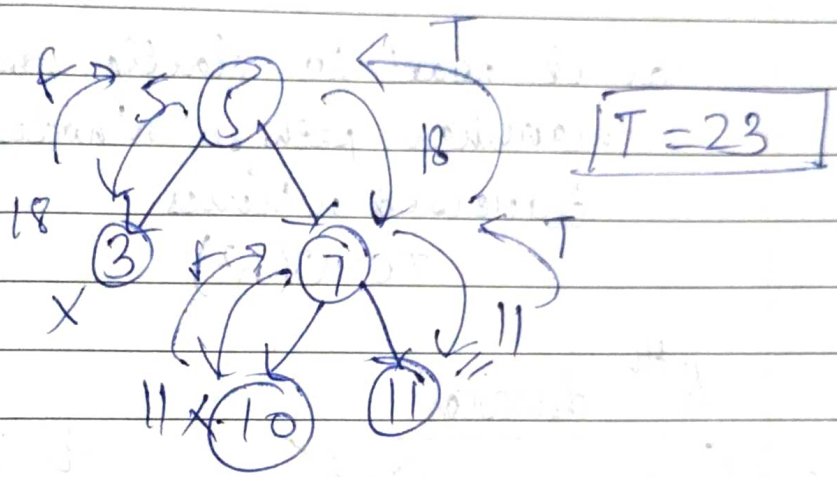


for optimised -



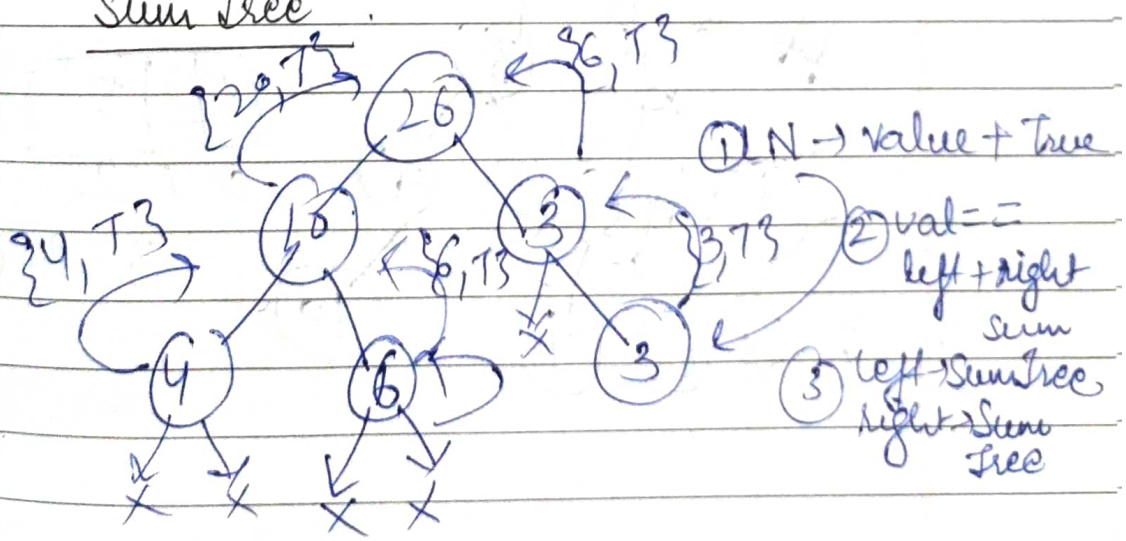
Path Sum.





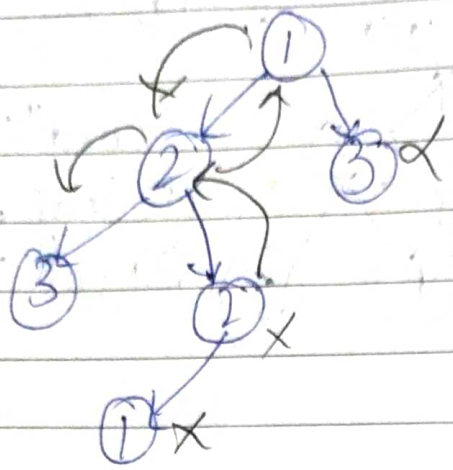
~~Lowest Common Ancestor~~

Sum Tree.



RANKA
DATE / /
PAGE

Path Sum III



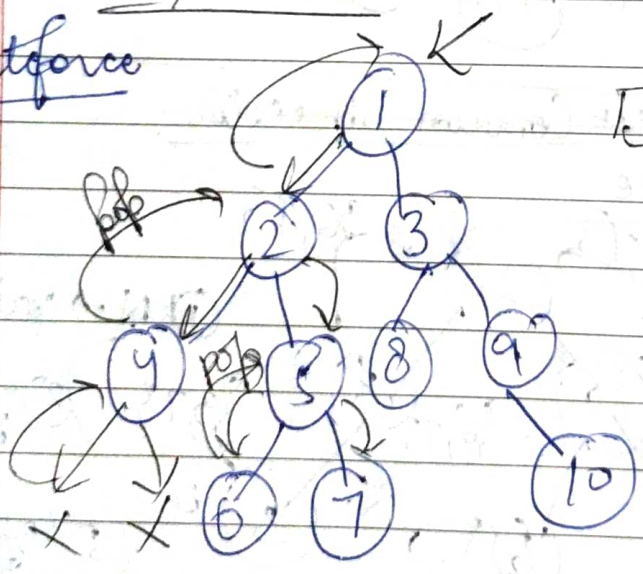
1 - 2 - 2
2 - 3
2 - 2 - 1

har ek node ko starting node maankar path so n nodes traversed n times

TC → $O(n^2)$

K^{th} Ancestor

brutforce



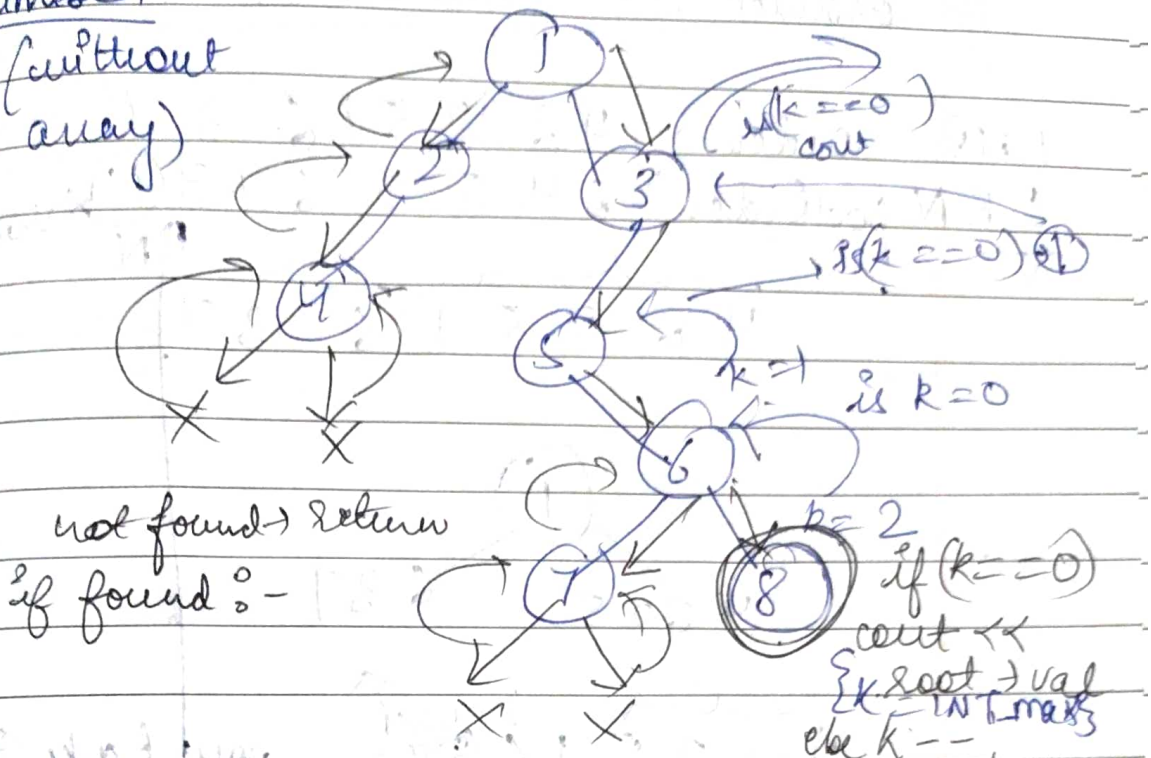
a

1	2	4	5	8	9

1	2	5
0	1	2

3 - 2 = 1

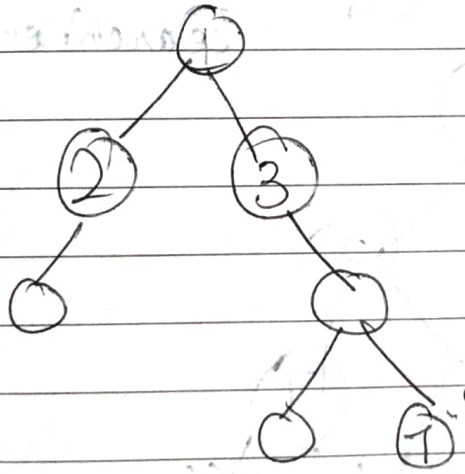
optimised – (without away)



bool found = false

not found → return

if found :-



element = 7.

K = 2

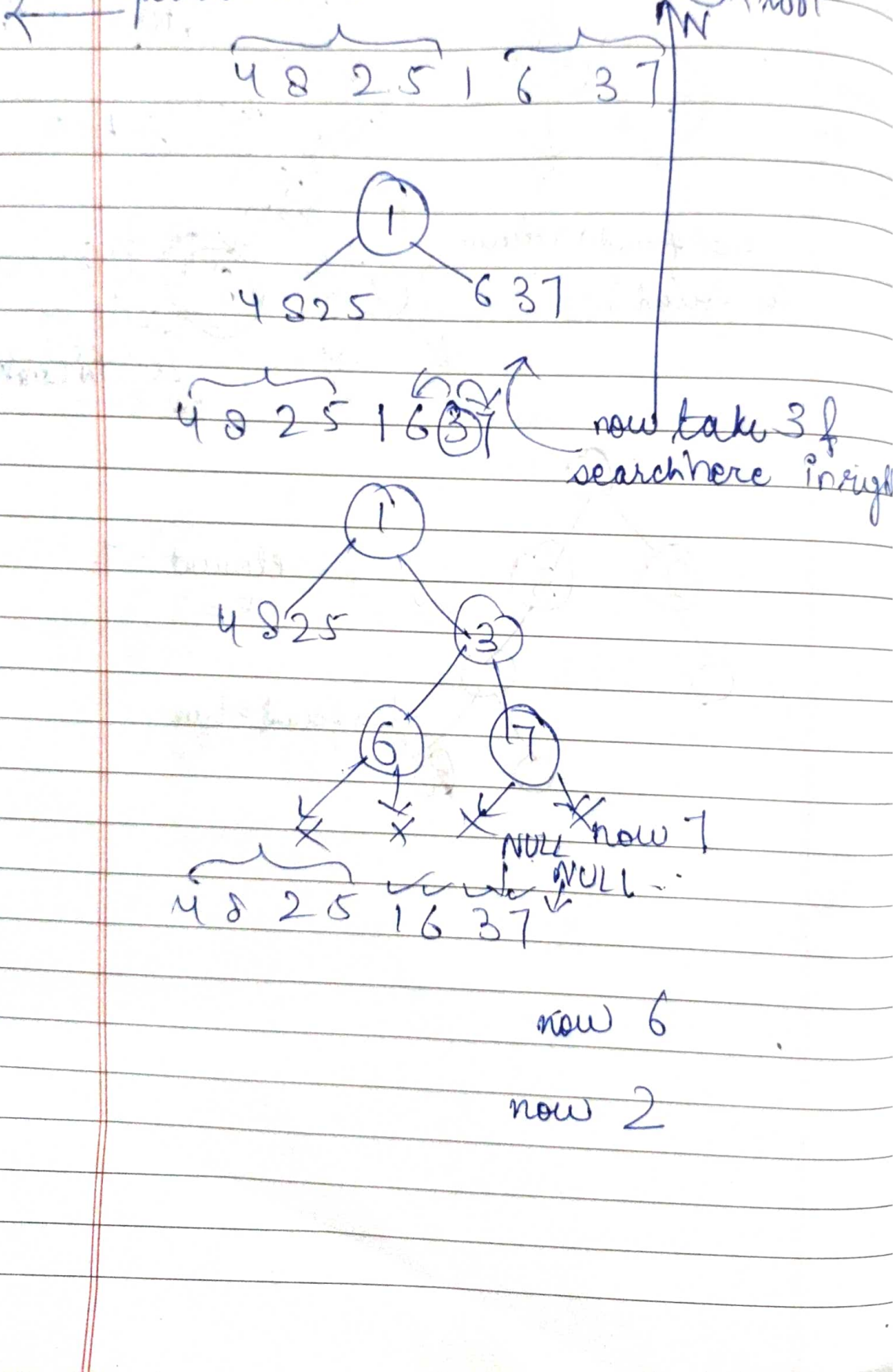
→ found = true.

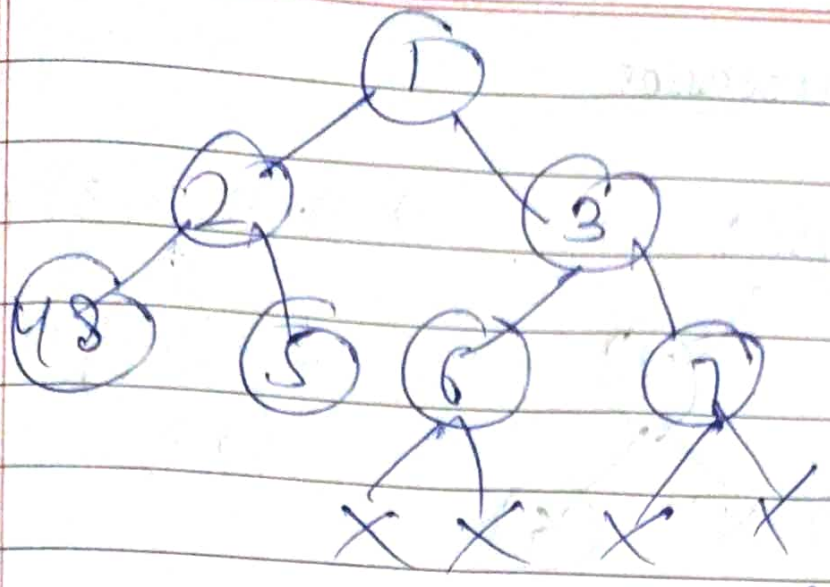
Imp → Adobe

Construct BT from inorder & postorder

LNR inorder → 4 8 2 5 1 6 3 7

LRN postOrder → 8 4 5 2 6 7 3 1 ← root





① post order index starts from n.
② first RC then LC.