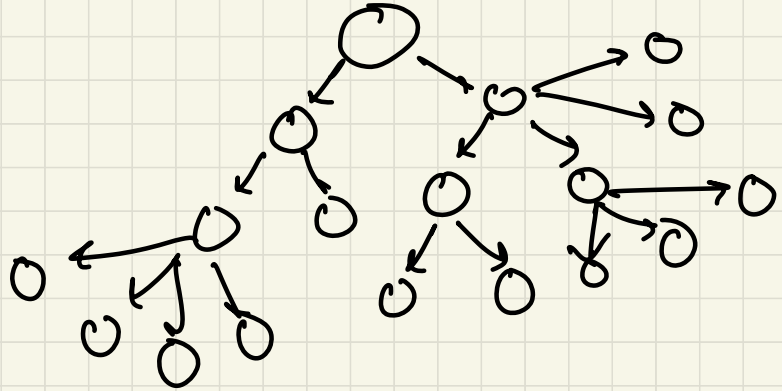
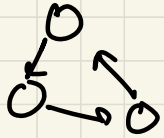
Trees

① Binary Tree
② Binary Search Tree



① Trees cycle exist nahi karti

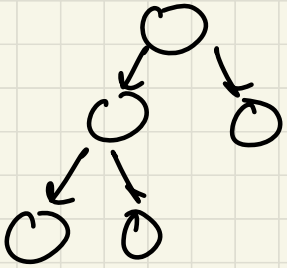


Binary → atmax − 2

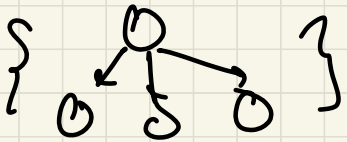
↓

atmost → ②

#

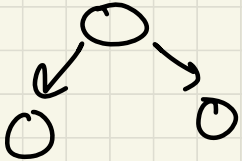


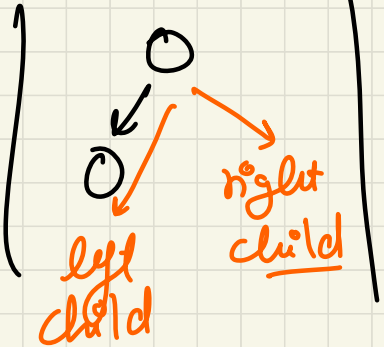
→ yes!

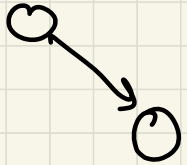


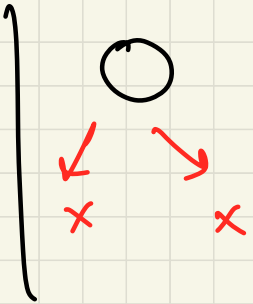
②

In terms of parent



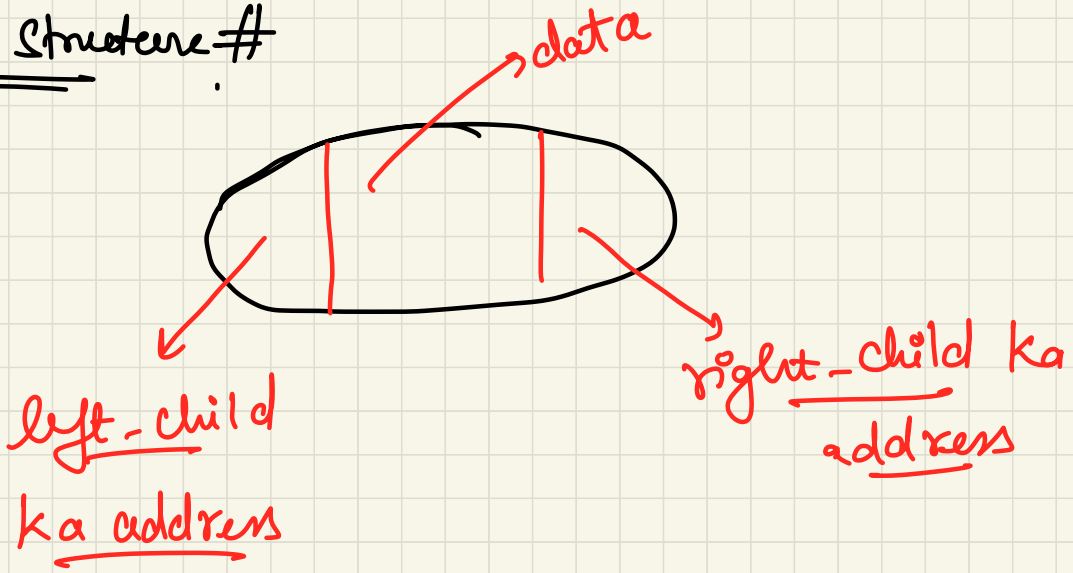




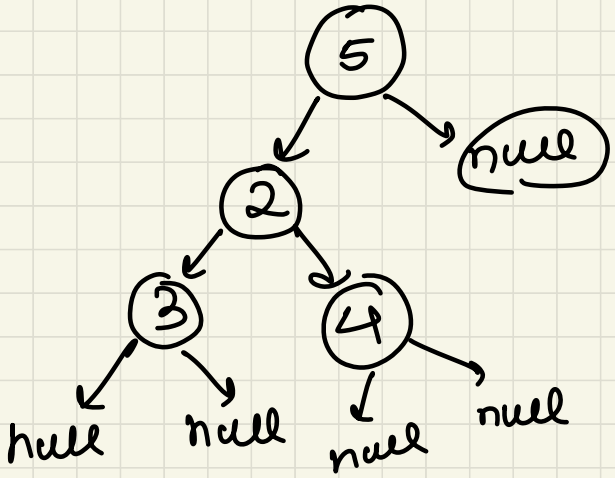


Root Node → ultimate parent.

Tree Node Structure#



input: 5 2 3 -1 -1 4 -1 -1 -1

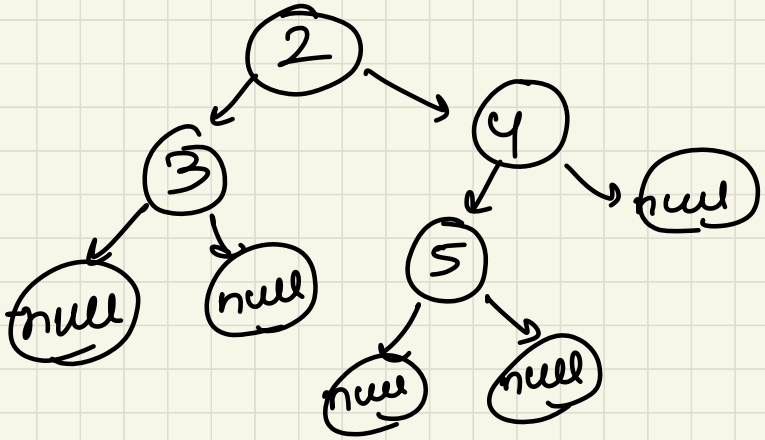


input 2

Input-2 :-

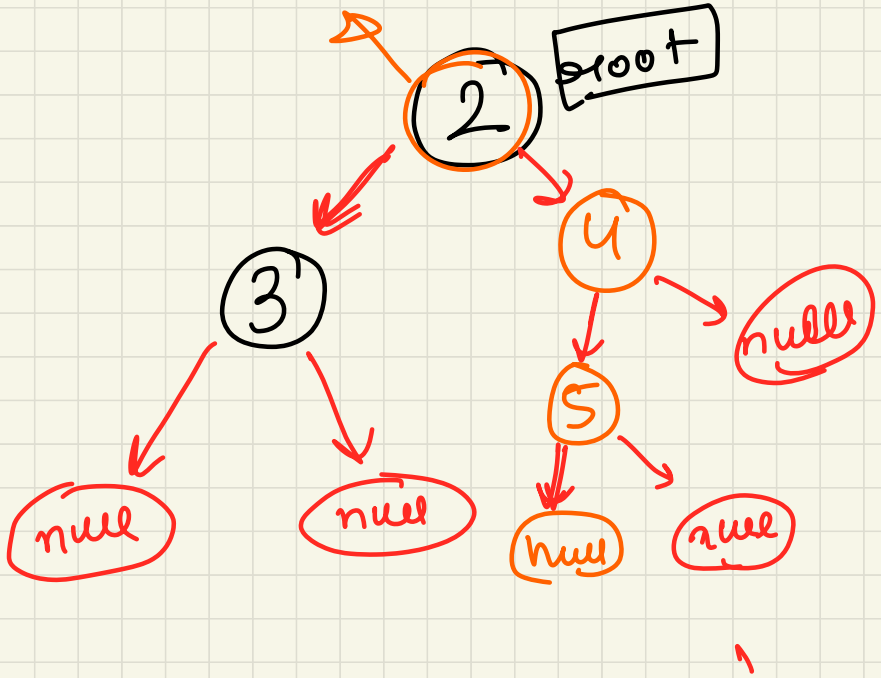
2 3 -1 -1 4 5 -1 -1 -1

Traversal



Random example

2 3 -1 -1 4 5 -1 -1 -1



input stream

2 3 -1

-1 4 5

-1

Types of traversals

① Pre-order traversal

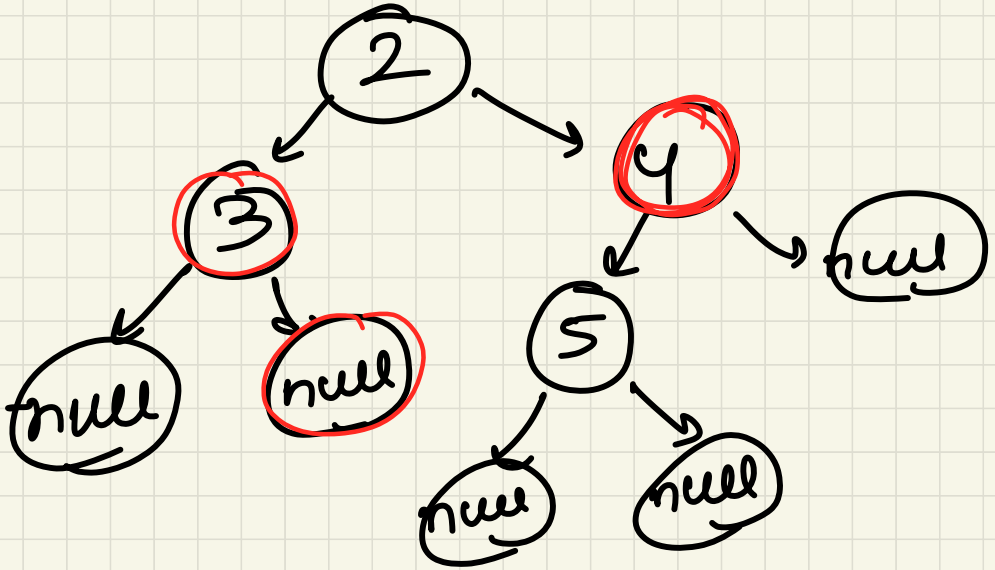
② In order traversal → (BFS)

③ Post-order Traversal → left right root

④ Level-order Traversal

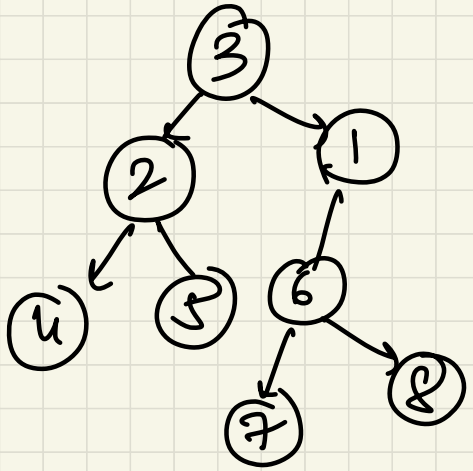
root, left, right ; left root right /

① pre-order traversal → root, left, right



Output :

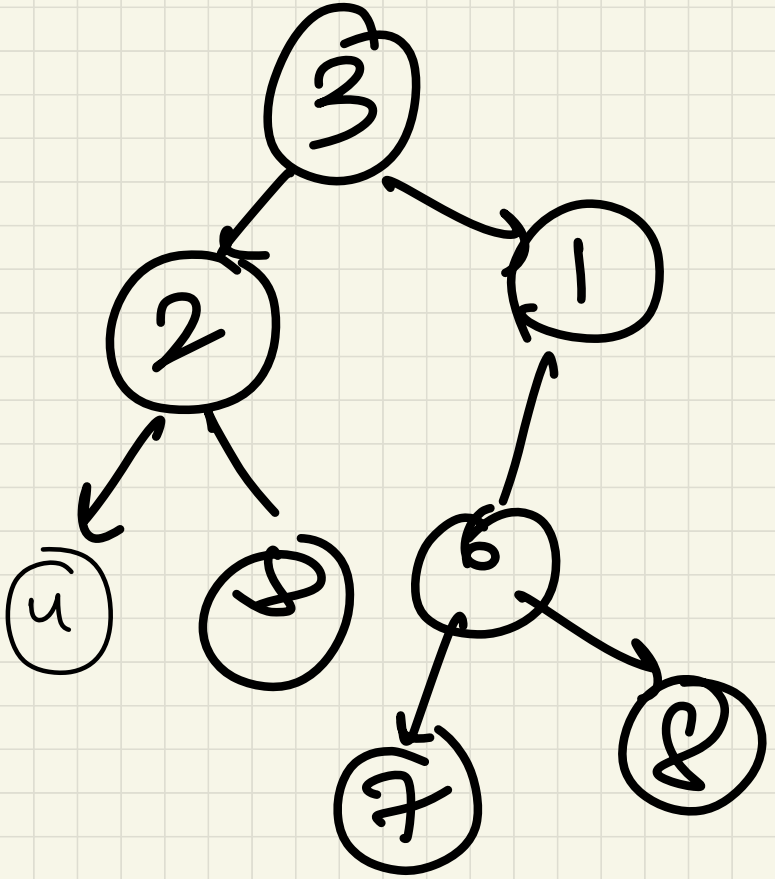
2 3 -1 -1 4 5 -1 -1 -1



pre order traversal

Input :- 3 2 4 -1 -1 5 -1 -1 1 6 7 -1 -1 8 -1 -1 -1

R



Output

3 2

4 -1 -1

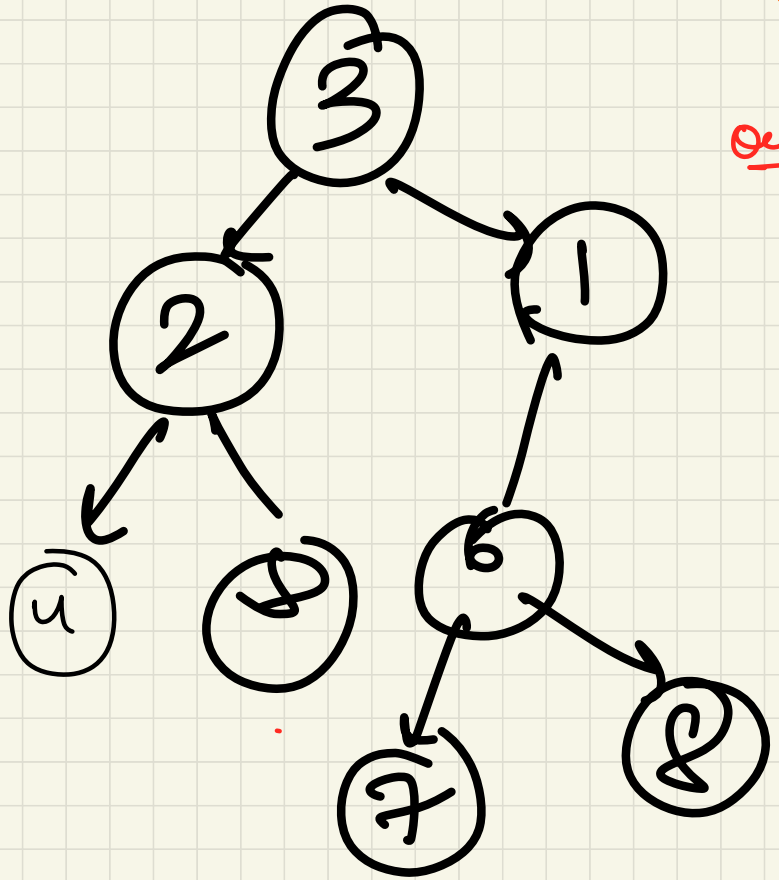
5 1 6

7 8 -1

-1

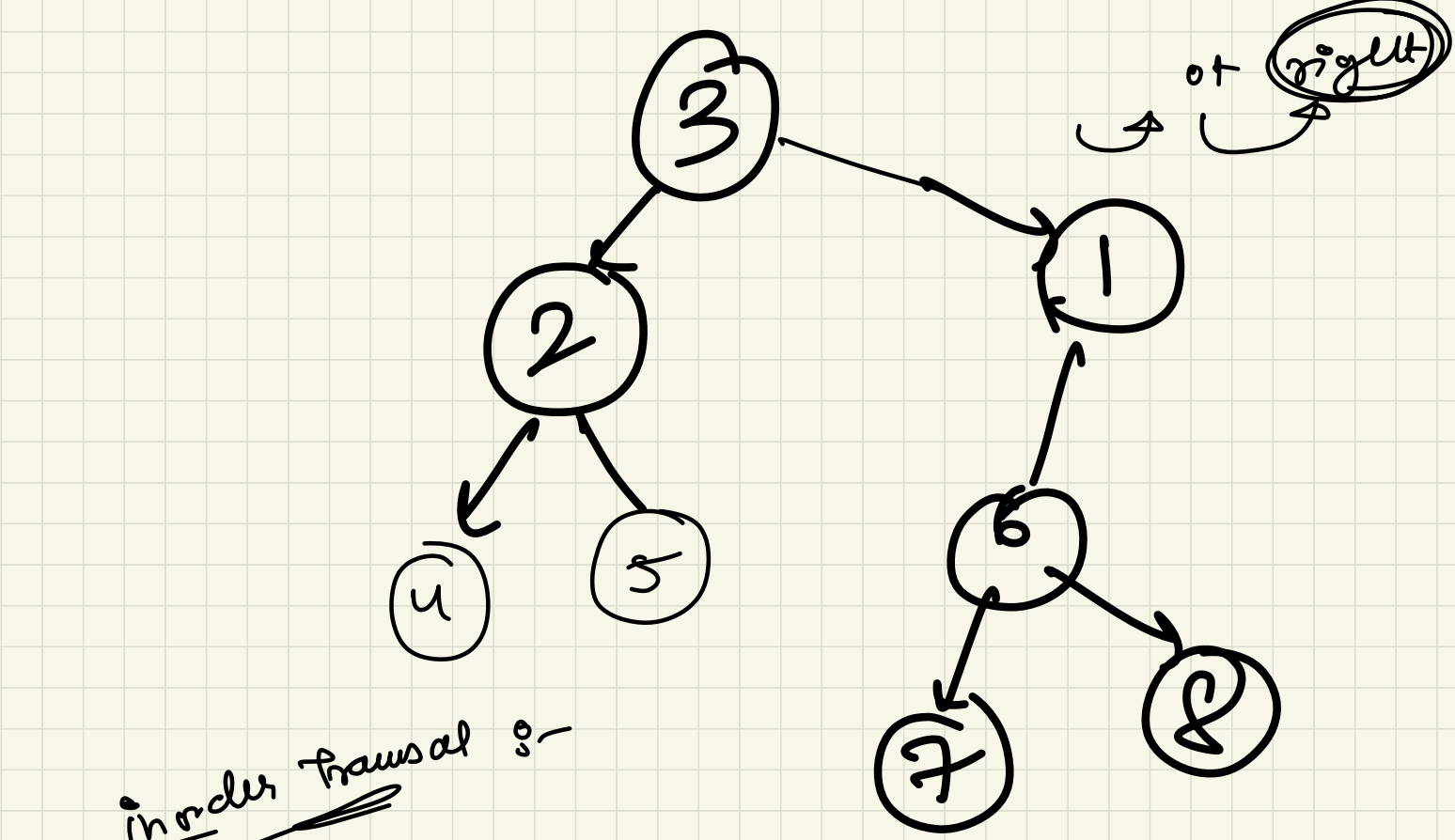
Inorder Traversal

→ left root right



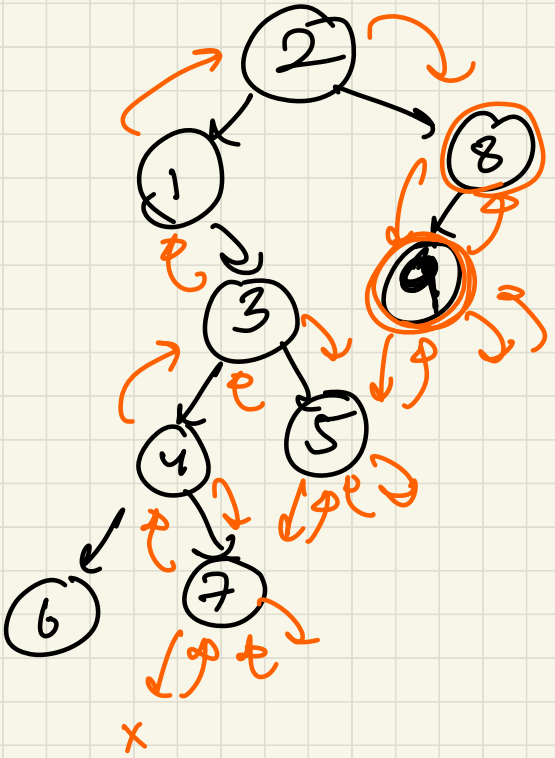
Output :-

-1 4 -1



inorder traversal :-

4 2 5 3 7 6 8 1



Inorder :-

(left, root, right)

inorder - traversal

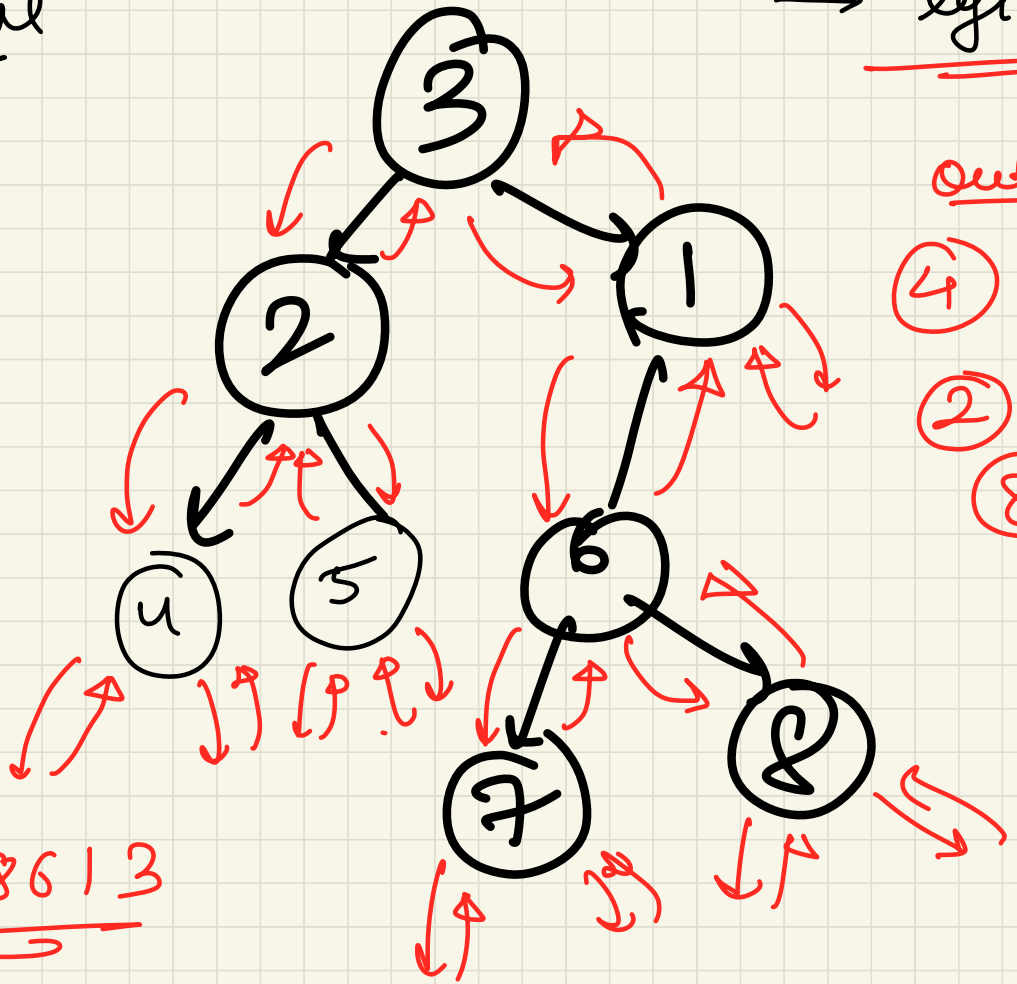
1 6 4 7 3

5 9 8

2

Post-order Traversal

→ left, right, root

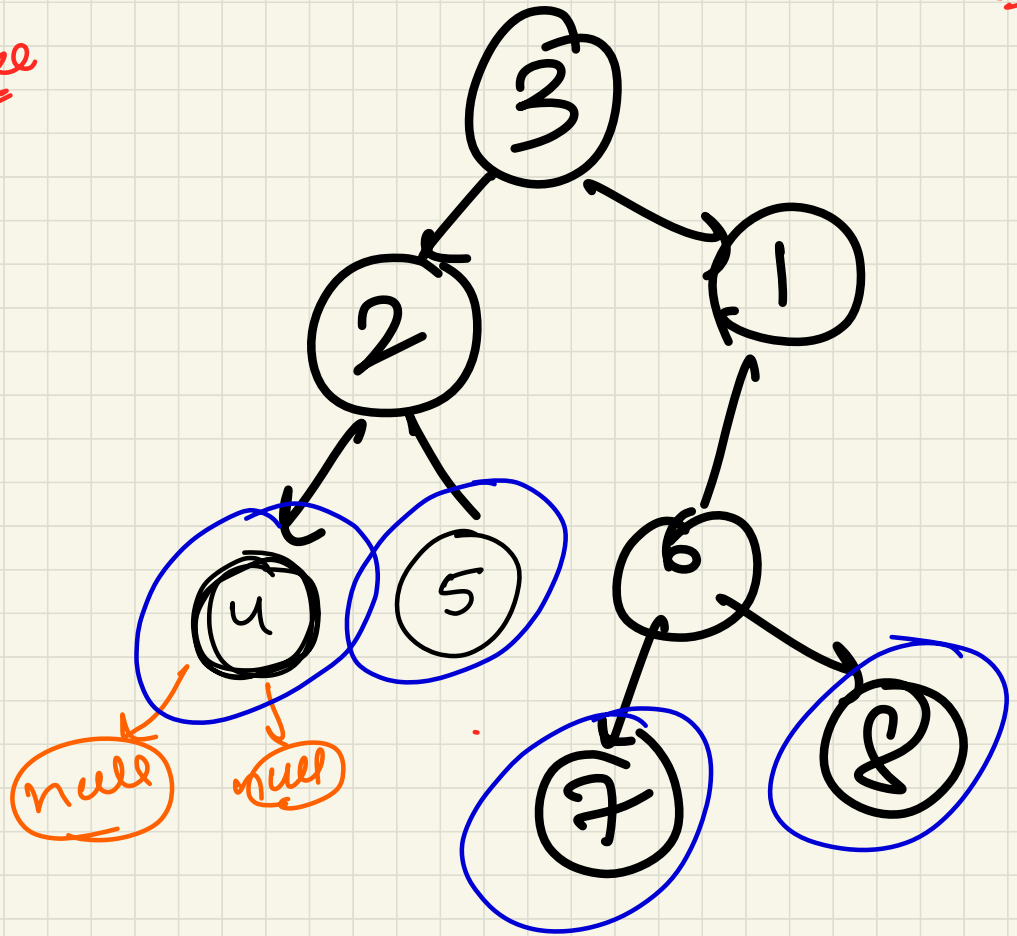


output:

4 5 2 7 8 6 1 3

4 5 2 7 8 6 1 3

Tree

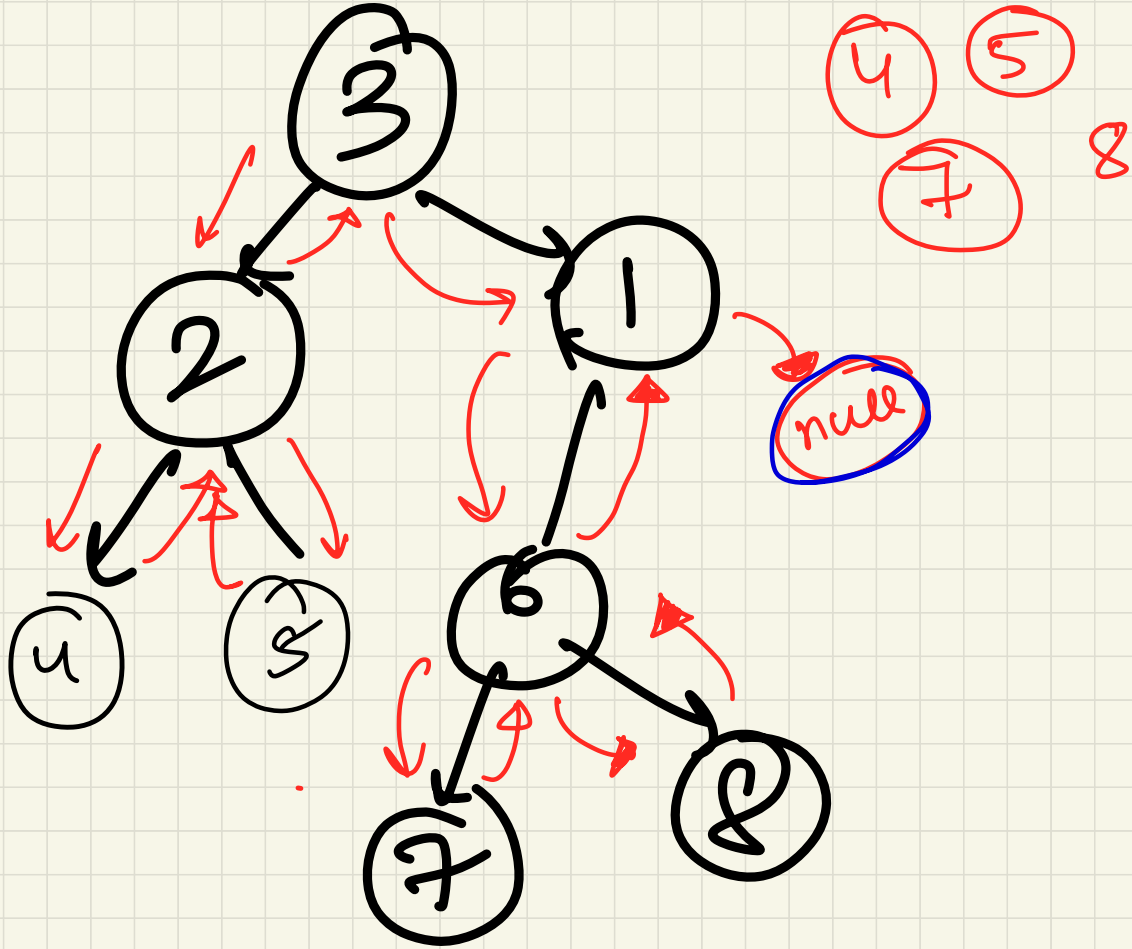


leaf node

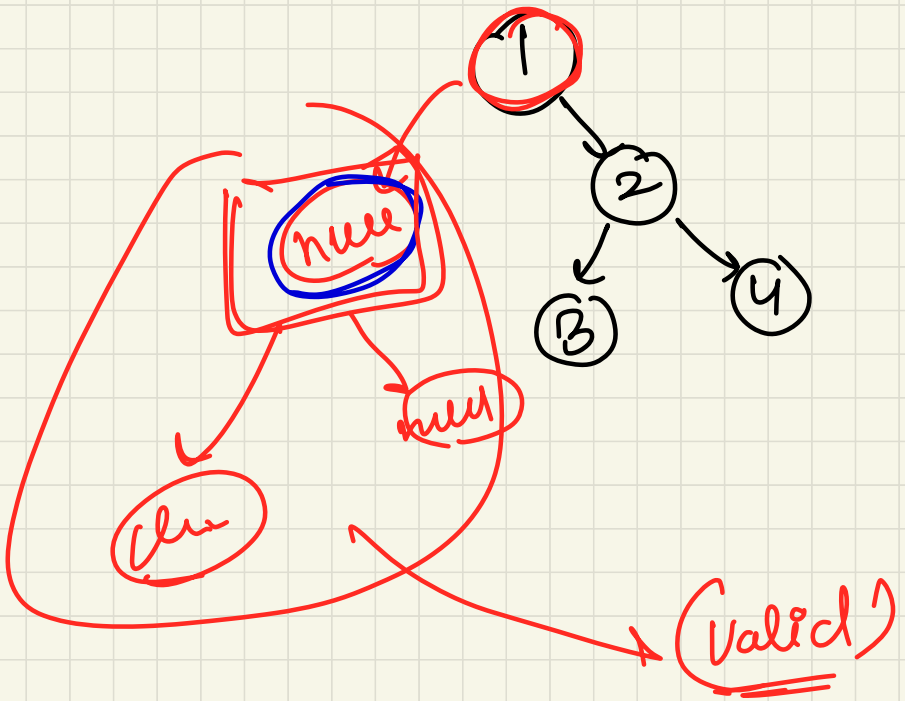
↳ left child == null

↳ right child == null

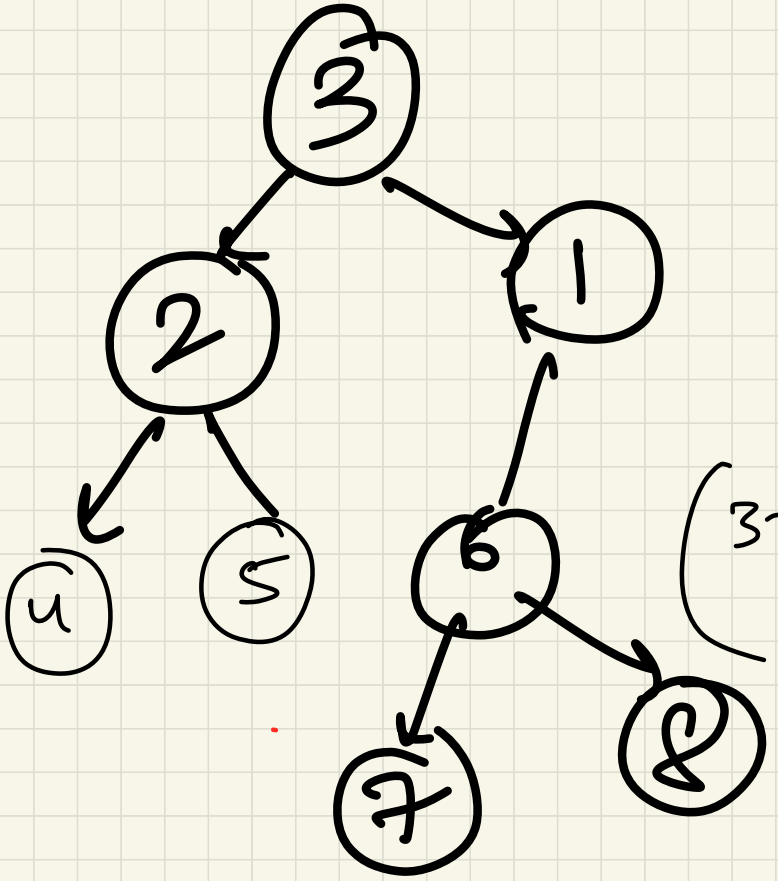
print leaf nodes


3
2
1
4
5
6
7
8
null
4
5
7
8

Answer



#

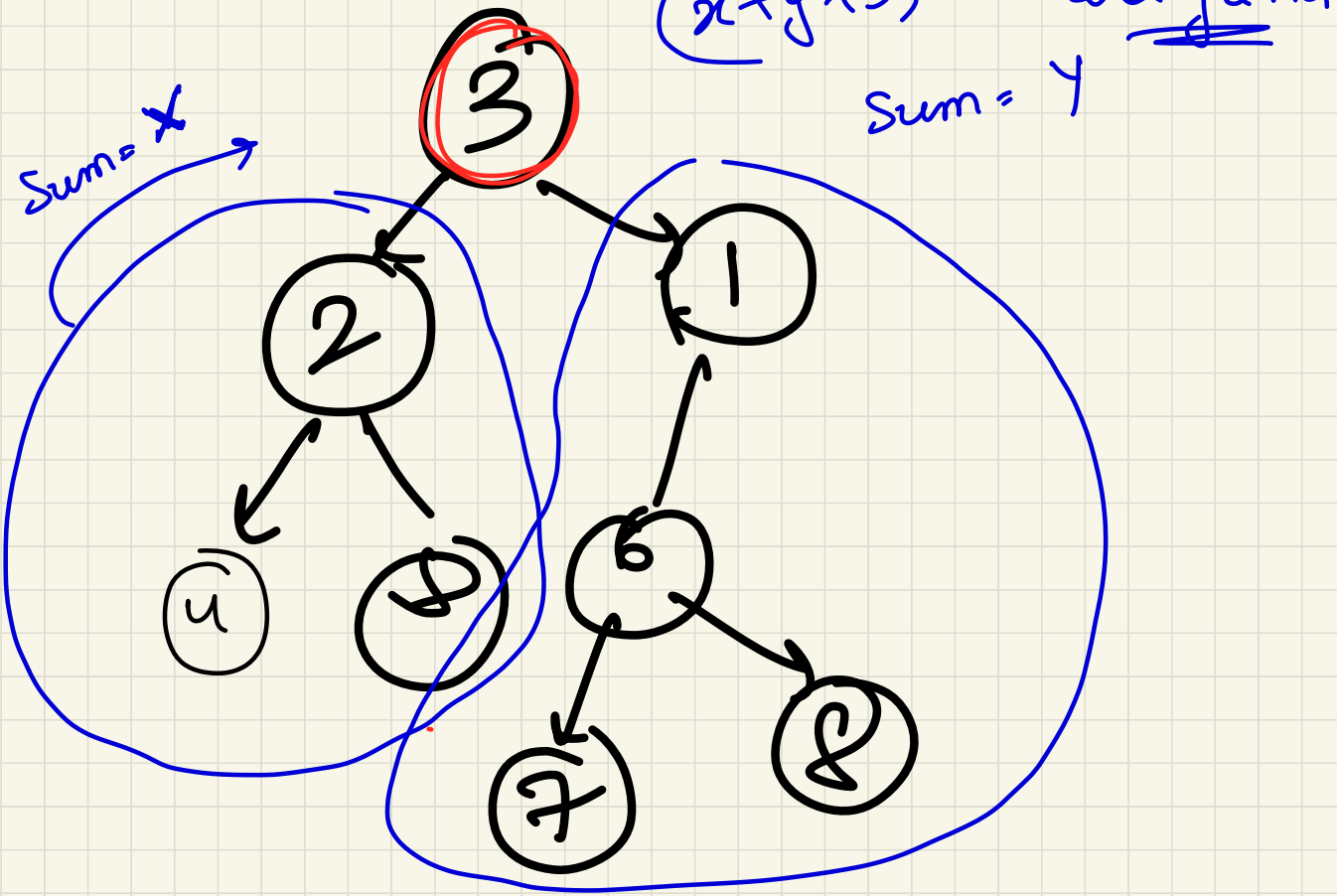


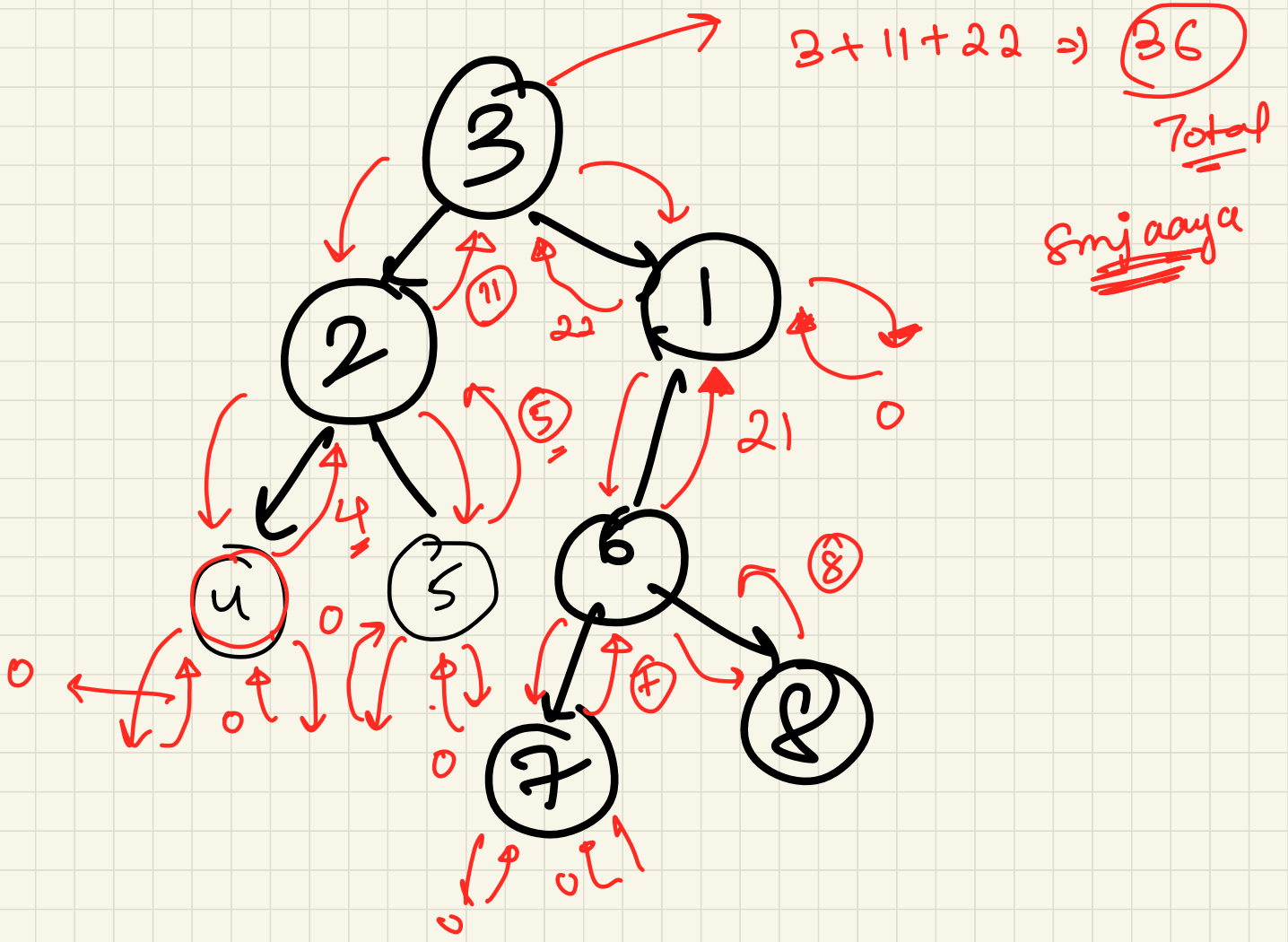
Sum of Complete Tree

Global variable nahi lene

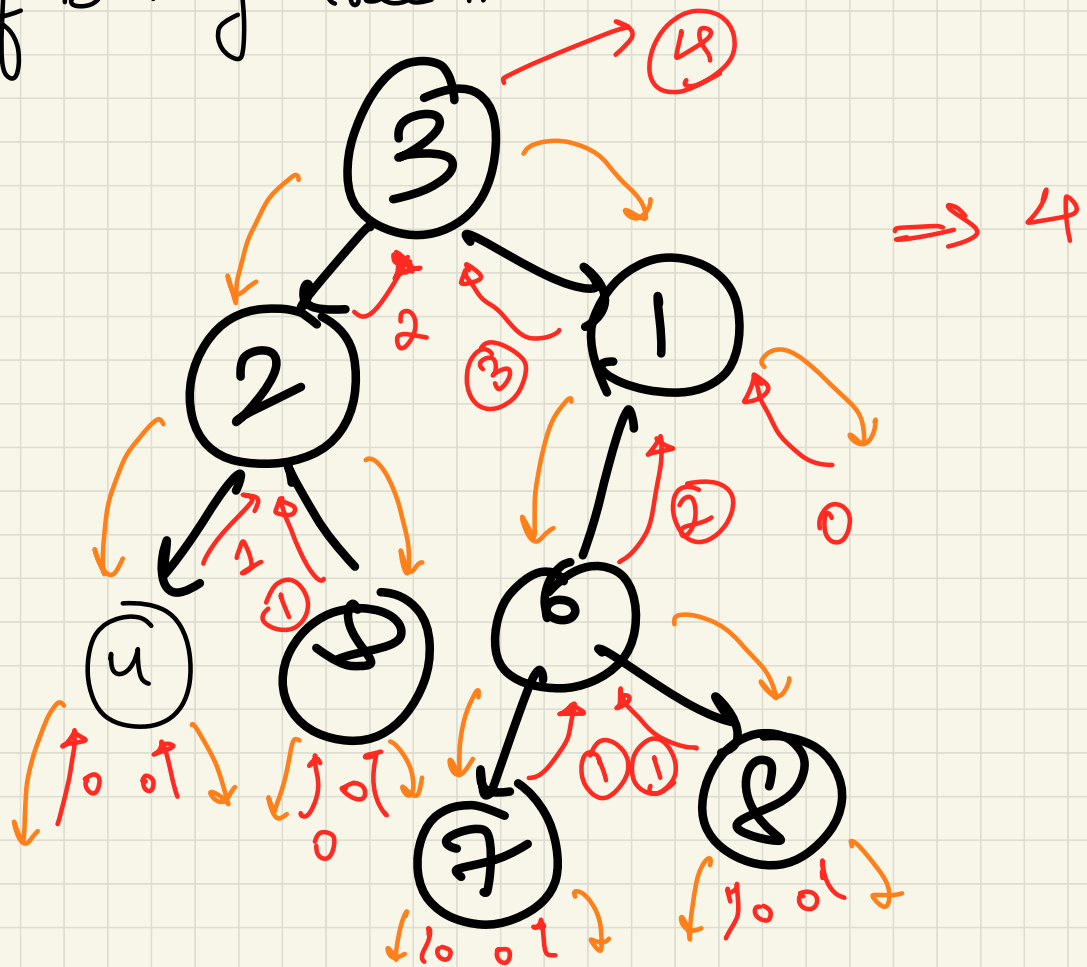
(3+2+1+4+5+6+7+8)

Js

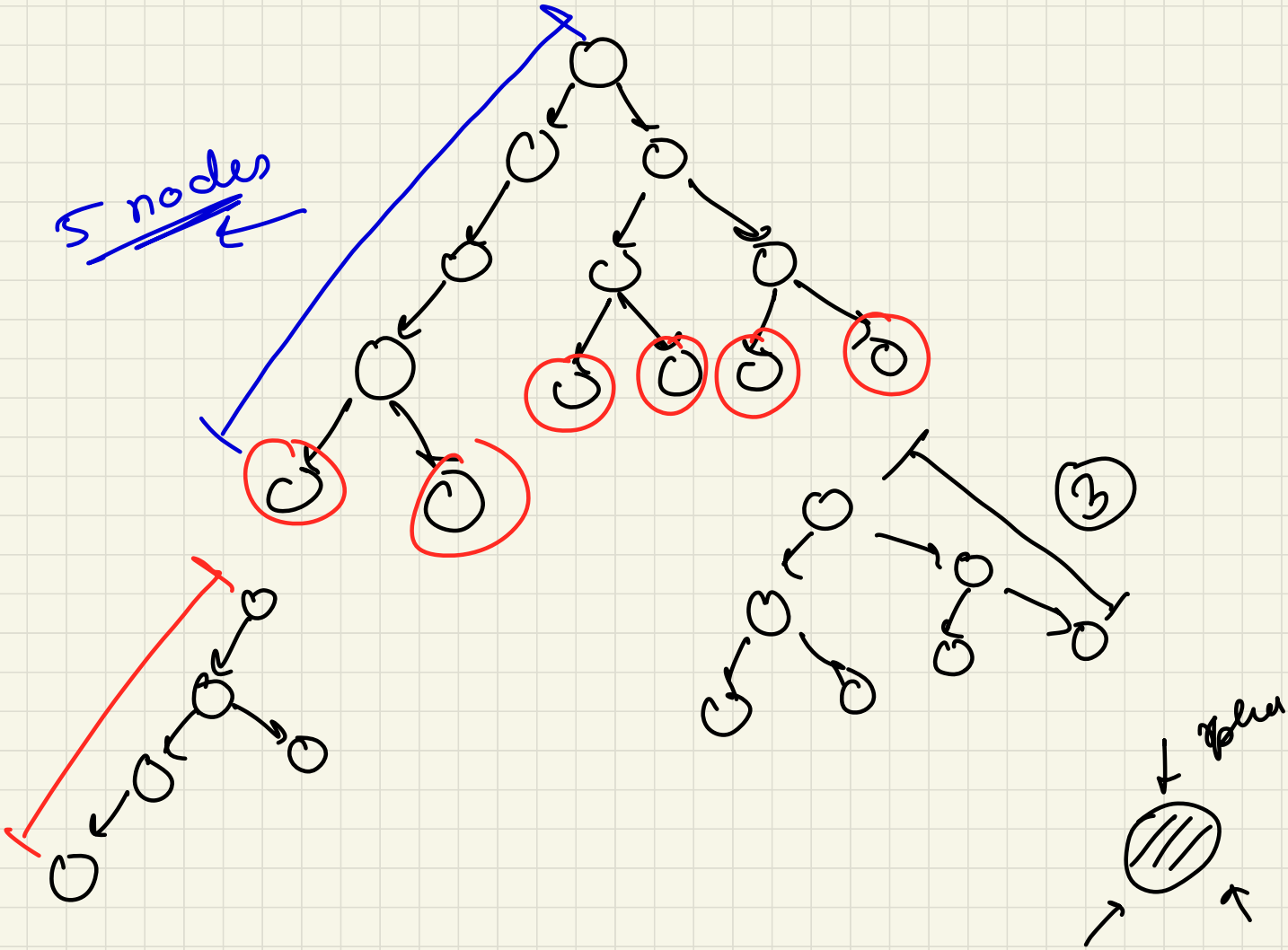

(x+y+3)
Sum = X
Sum = Y
3
2
1
4
9
6
7
2

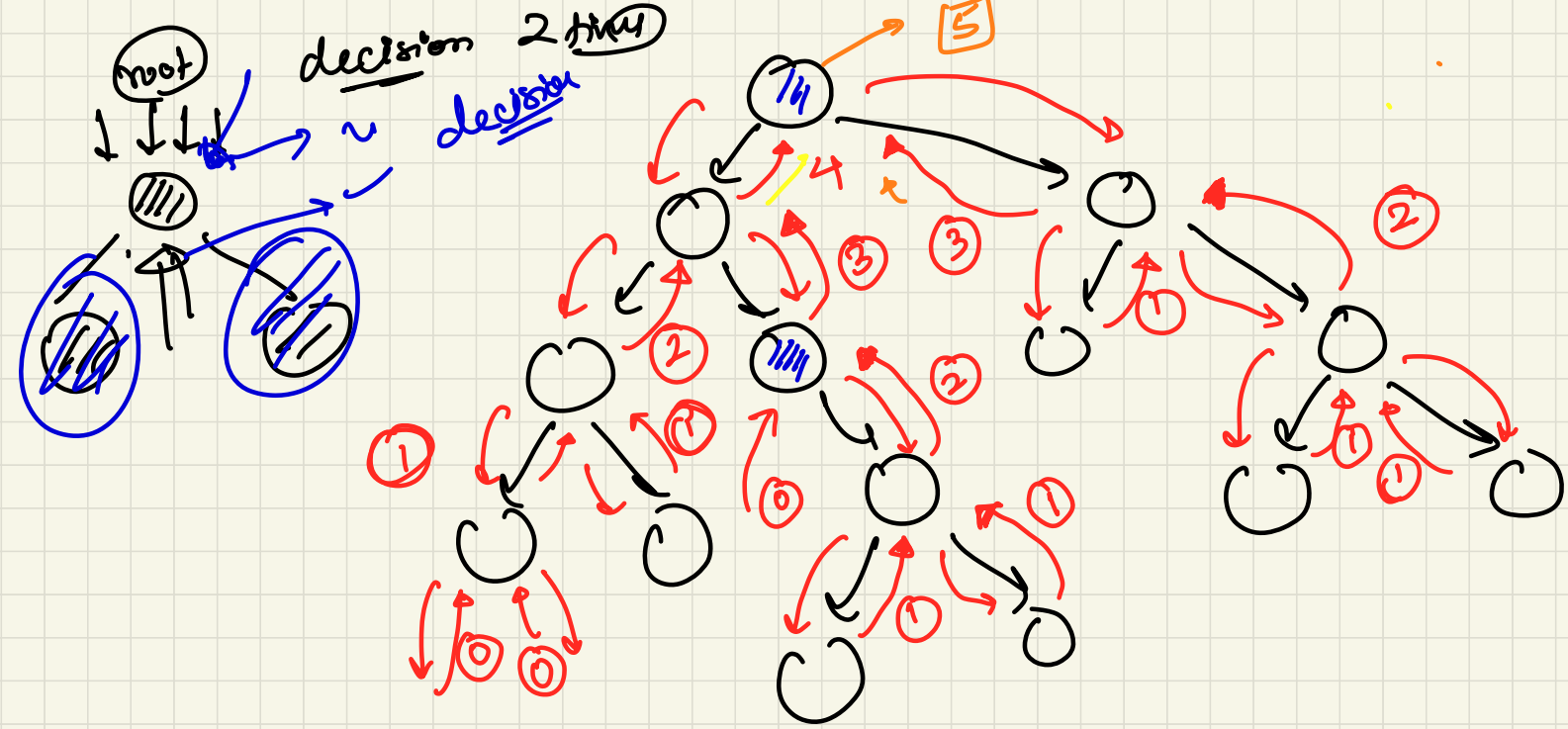

3+11+22 ⇒ 36
Total
Smjaaya
3
2
1
4
5
6
7
8
11
22
5
21
0
4
8
0
0
0
0
0
1
1

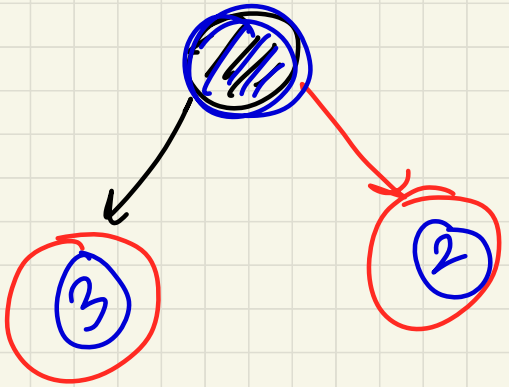
Height of Binary Tree



$\Rightarrow 4$


5 nodes
3
plus

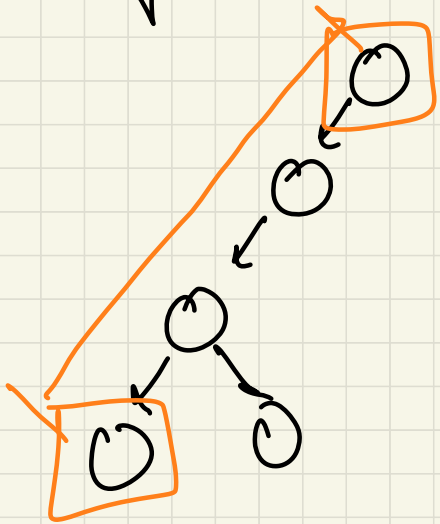

root
decision
decision
5
4
3
2
1
0

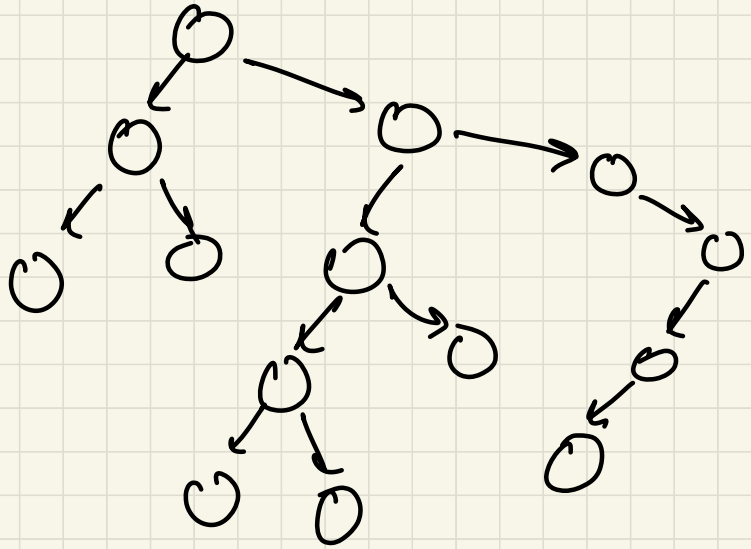


maximum

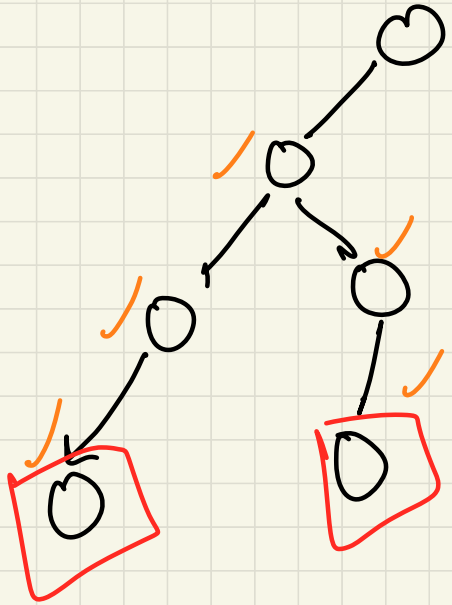
Da Icina

Diameter of a Binary Tree





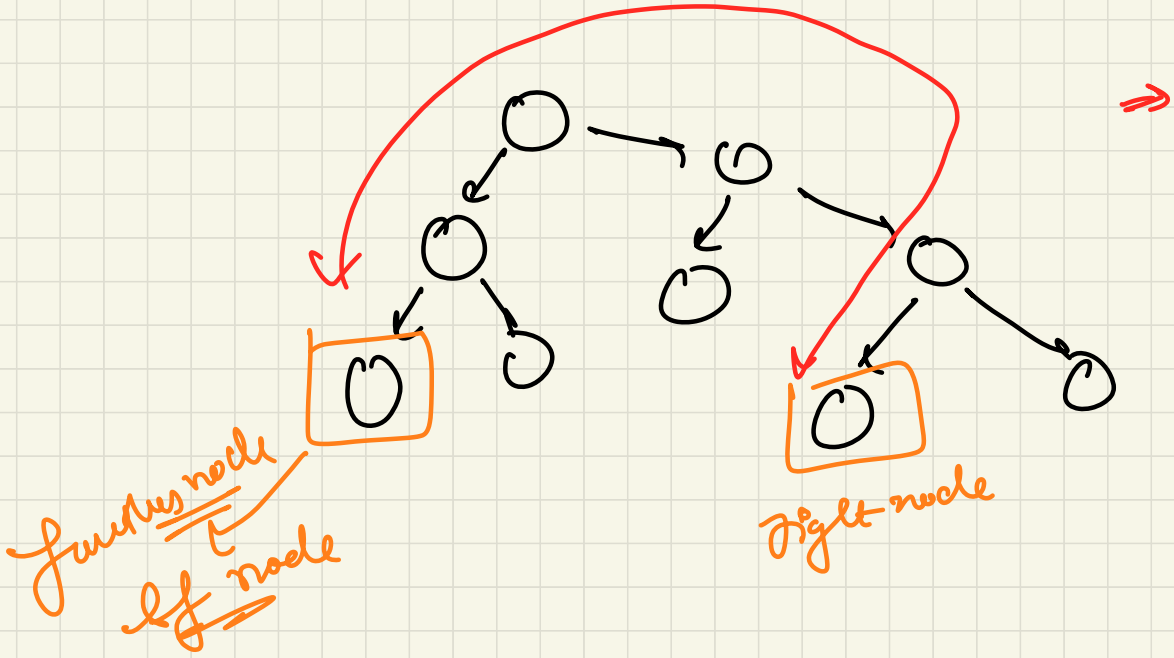
Diameter of Binary Tree

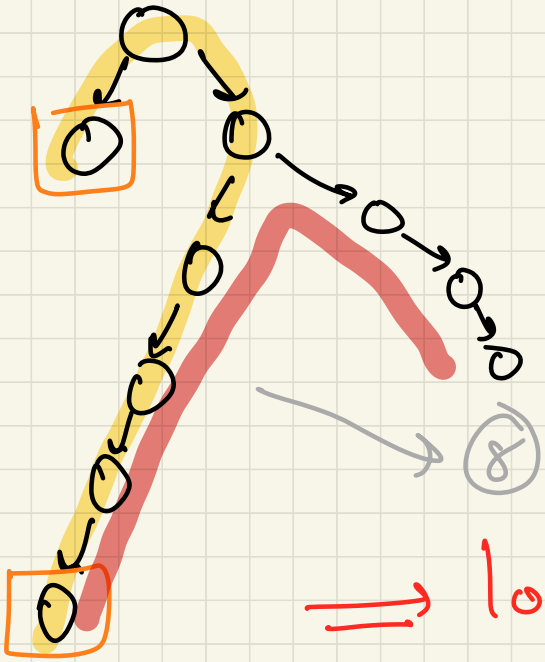


→ 5

Diameter = 5

#logic


Sundays node
=
left node
Right node



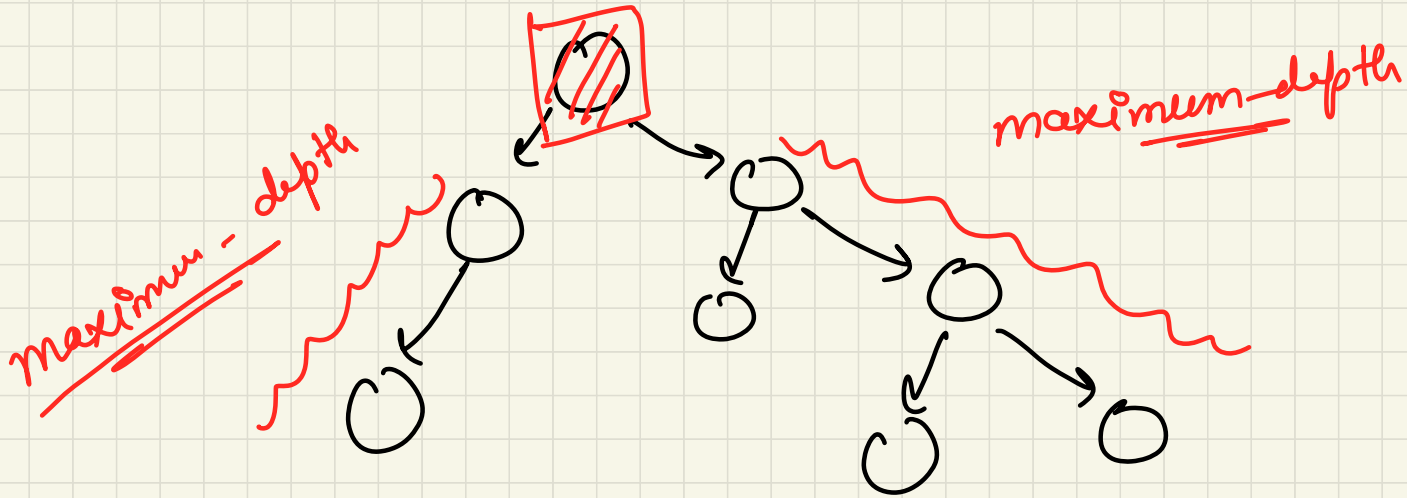
6

logic wrong ans

clear

8

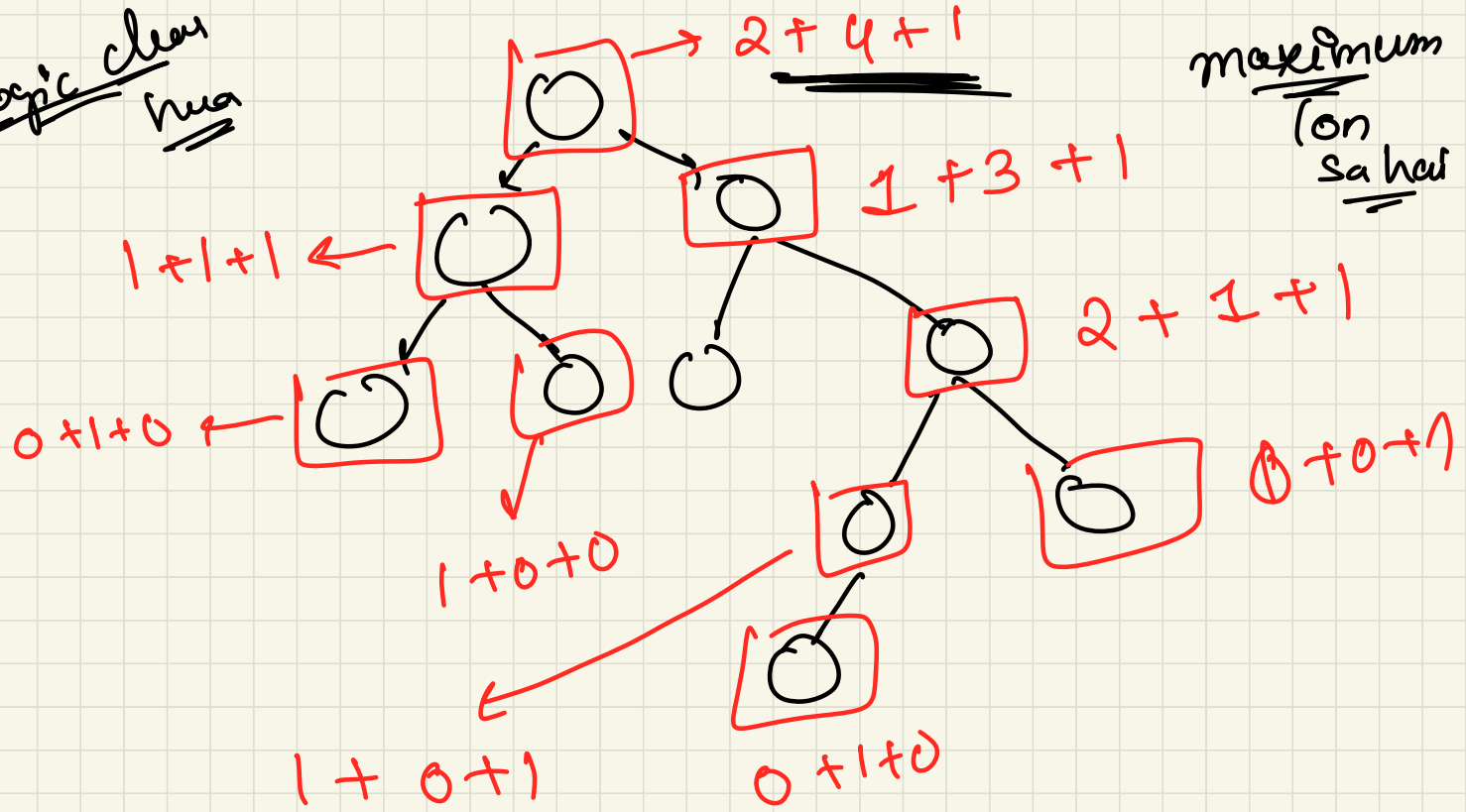
⟹ logic → extend

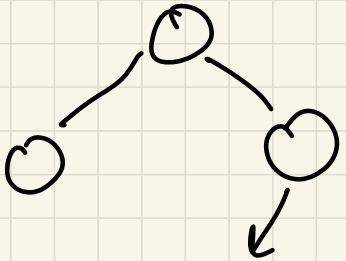


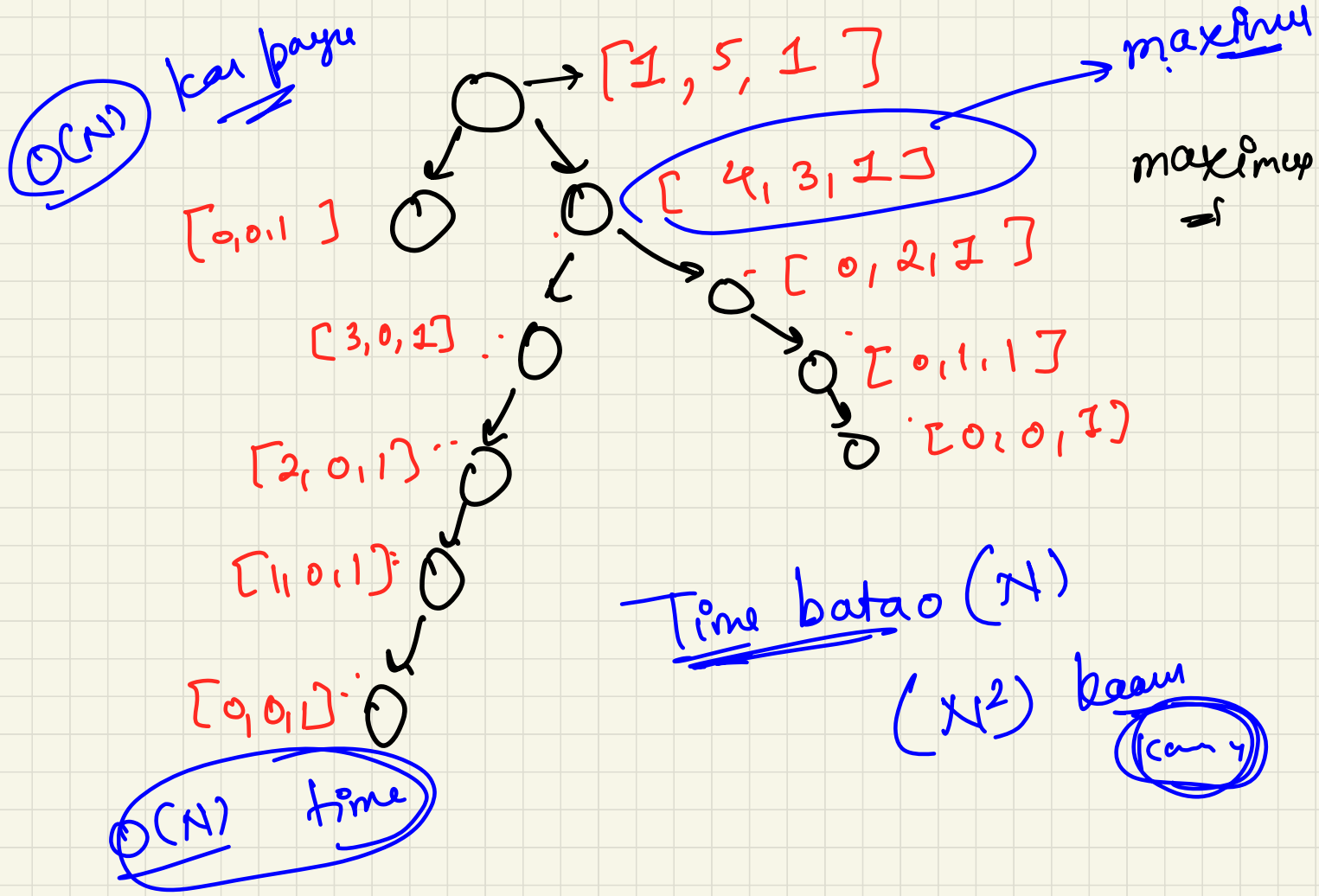
ans => $1 + \max\text{\_left\_depth} + \max\text{\_right\_depth}$

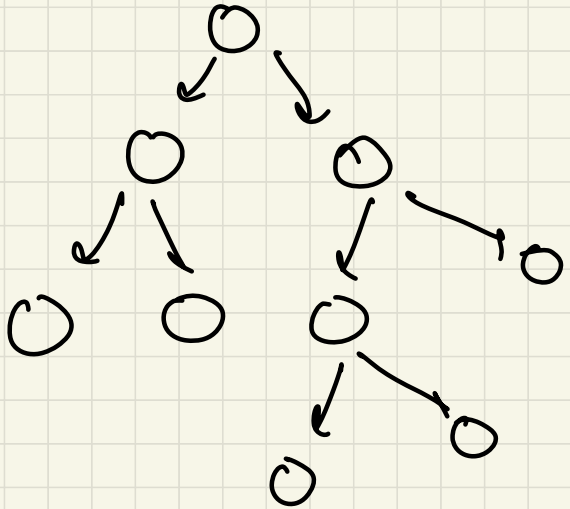
logic clear hua

maximum ton sahi

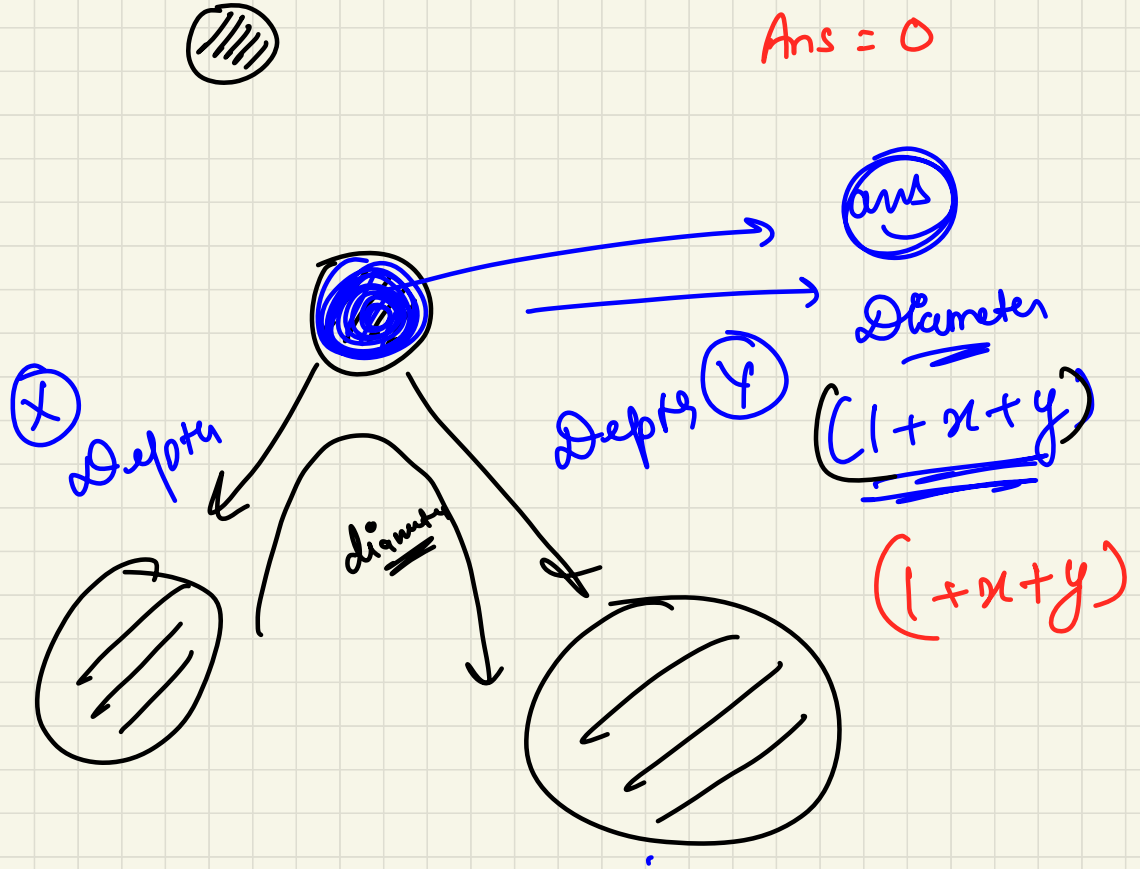




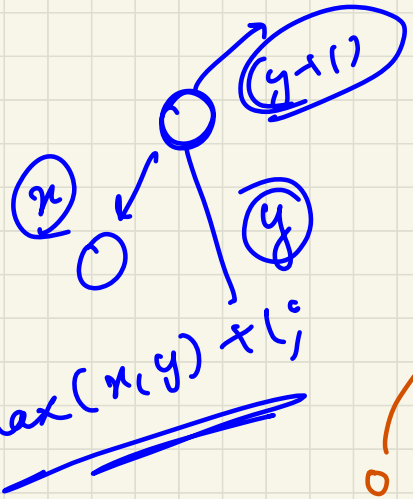

O(N)
kar payu
[1, 5, 1]
maxiny
[4, 3, 1]
maximum
[0,0,1]
[0, 2, 1]
[3,0,1]
[0,1,1]
[0,0,1]
[2,0,1]
[1,0,1]
[0,0,1]
O(N) time
Time batao (N)
(N²) bola
(can y



Ans = 0

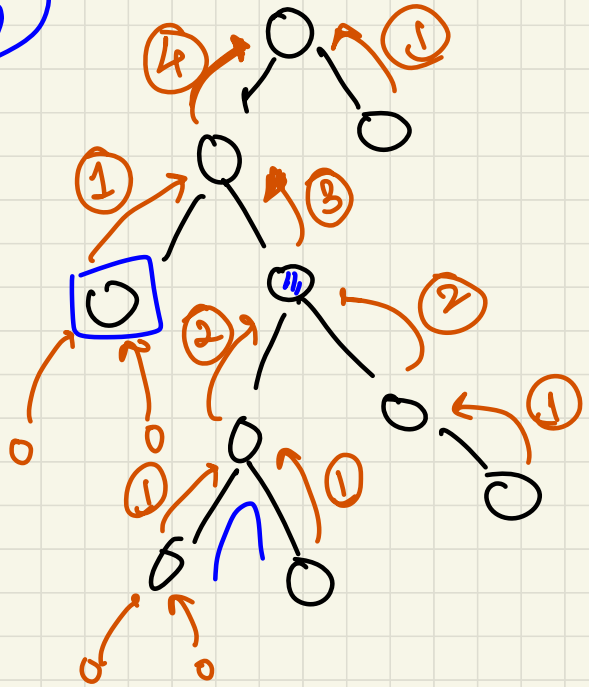


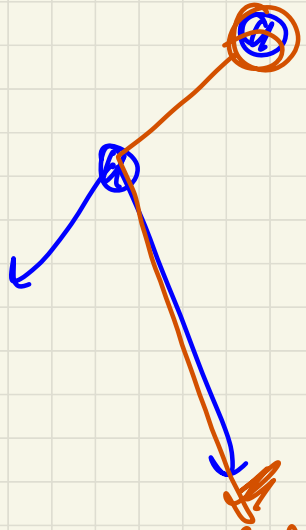
Ans = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6



max(x,y) + 1;

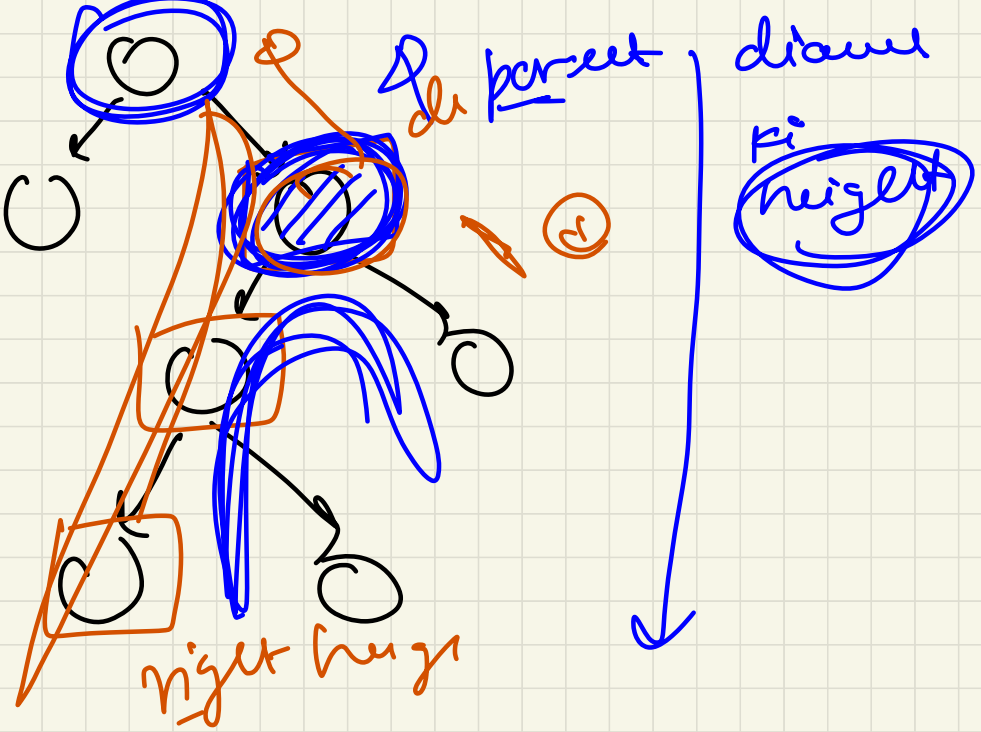
1 + max (x, y)





1 + max (left, right);

kaam aasna



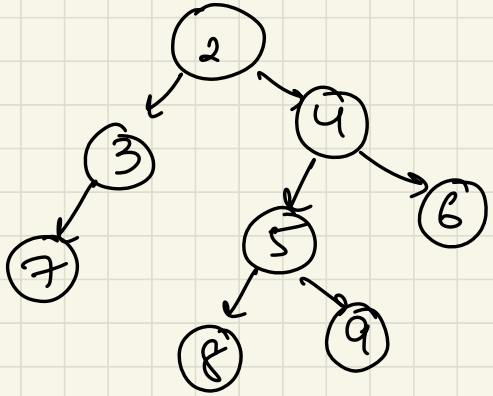
height

$\max(lh, rh) + 1$

level-order traversal

↳ (Breadth first Search) (BFS)

(DFS)
↓
(Recursion)



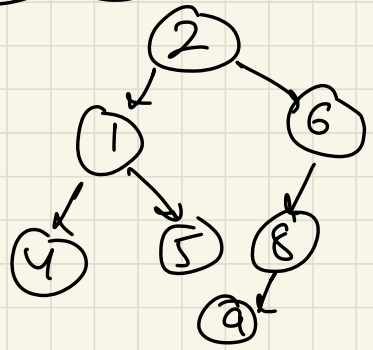
output

2

3 4

7 5 6

8 9



output:

2

1 6

4 5 8

9

level order

queue

Stack → LIFO

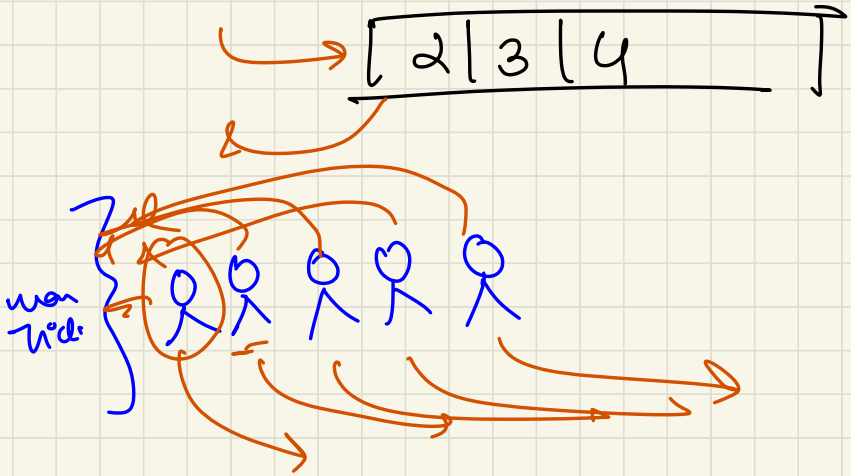
last in first out

(FIFO)

first in first out

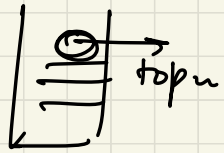
queue ke top par

& queue 2, 3, 4



syntax

- stack <int> st; ⟶ st.top();



- queue <int> q;

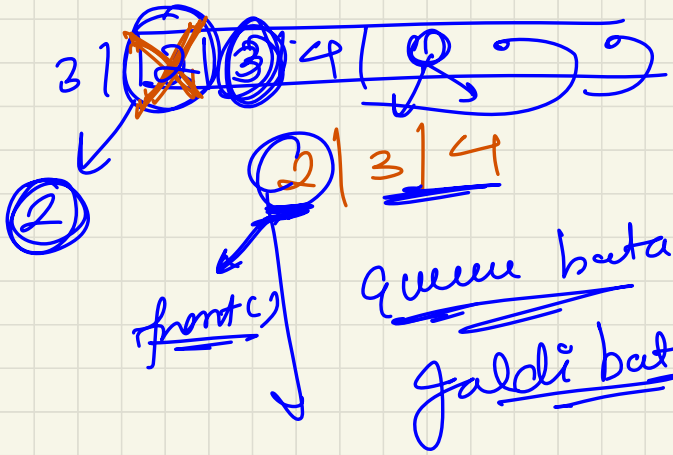
q.push() , q.pop() , q.front() , q.size();

eg:- queue <int> q.

q.push(2);

q.push(3);

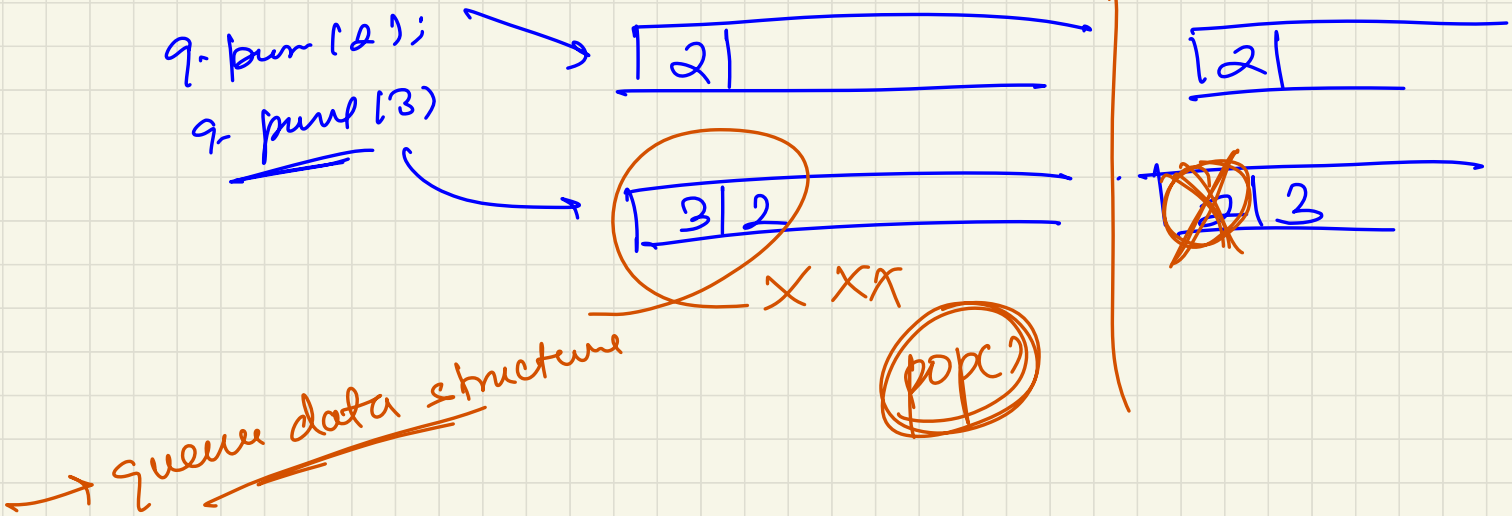
q.push(4)



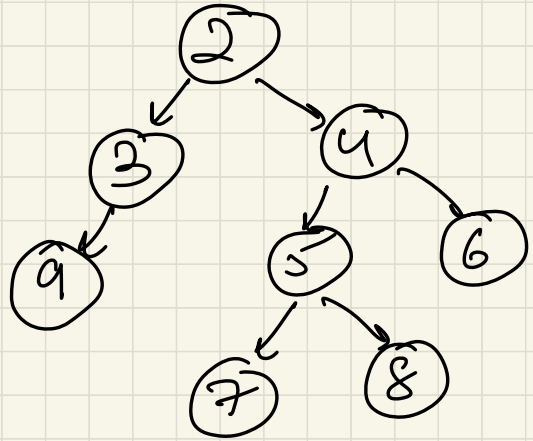
cout<<q.front();

⟶ q.pop()

cout<<q.front()<<endl;


q.push(2);
q.push(3)
2
3 2
2
2 3
X X
pop()
queue data structure

Trees — Binary — level wise kese draw karte



input nai

2 3 9 -1 -1 -1 4 5 7

-1 -1 8 -1 -1 6 -1 -1

2

3 4

9 5 6

7 8

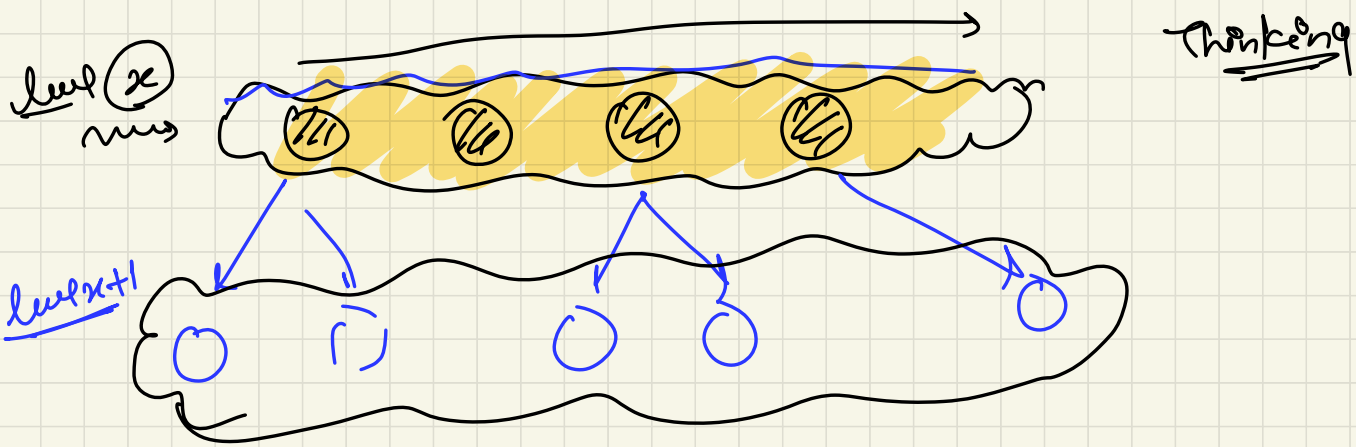
observe karo

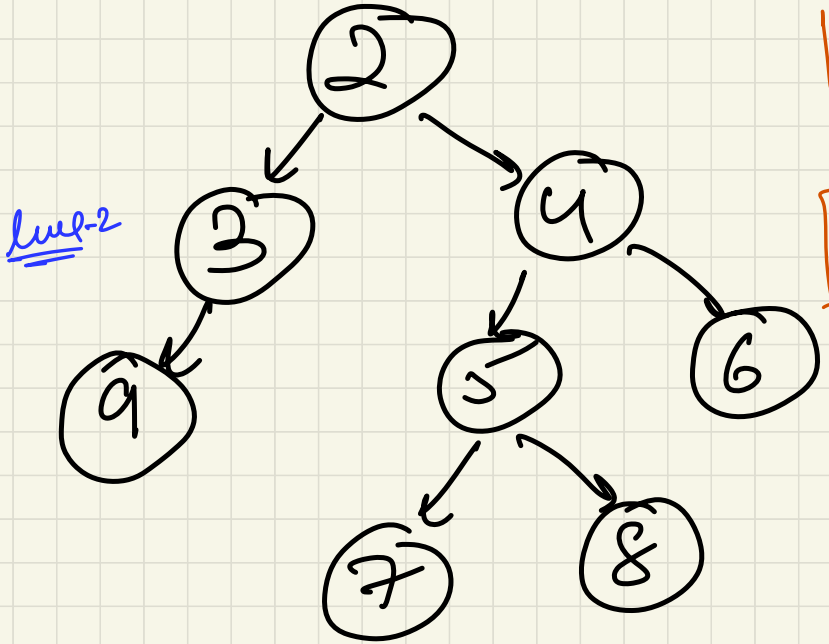
2 | 3 4 | 9 5 6 | 7 8

level 1 level 2 level 3 level 4

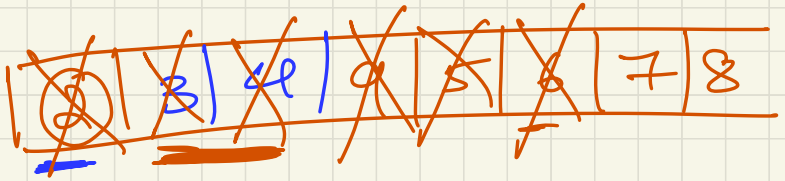
queue

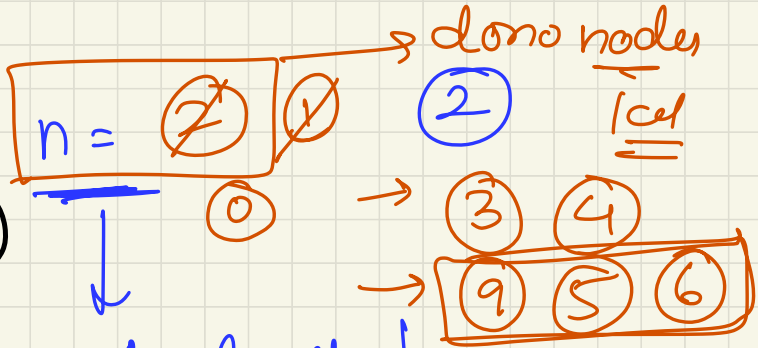
kon sa data structure


level x
level x+1
Thinking

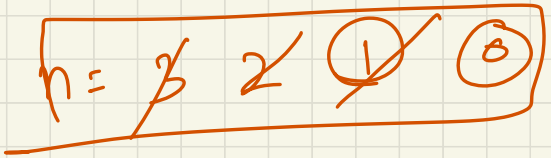


level-2

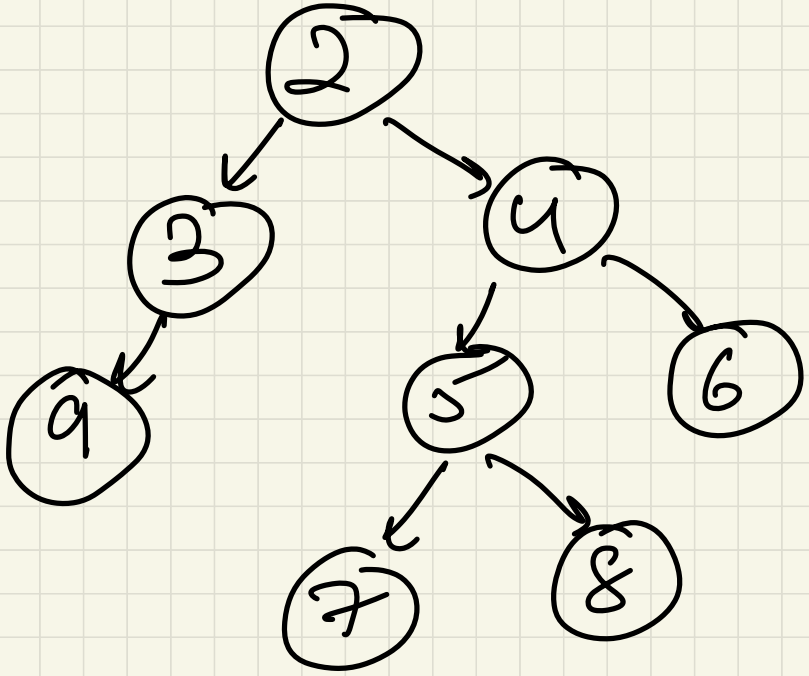




is level par kitne nodes present hai



Print Zig-Zag traversal



output

2

4 3

9 5 6

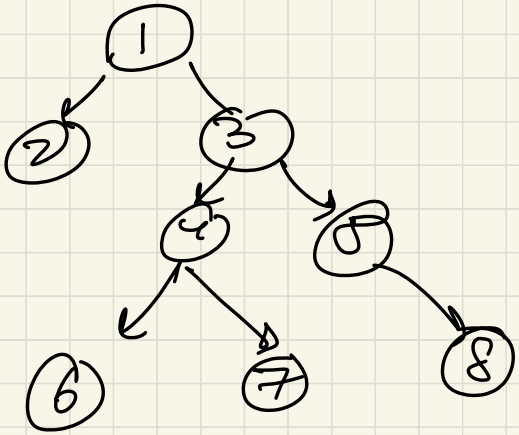
8 7

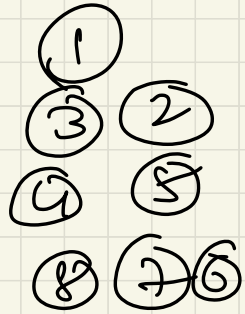
odd

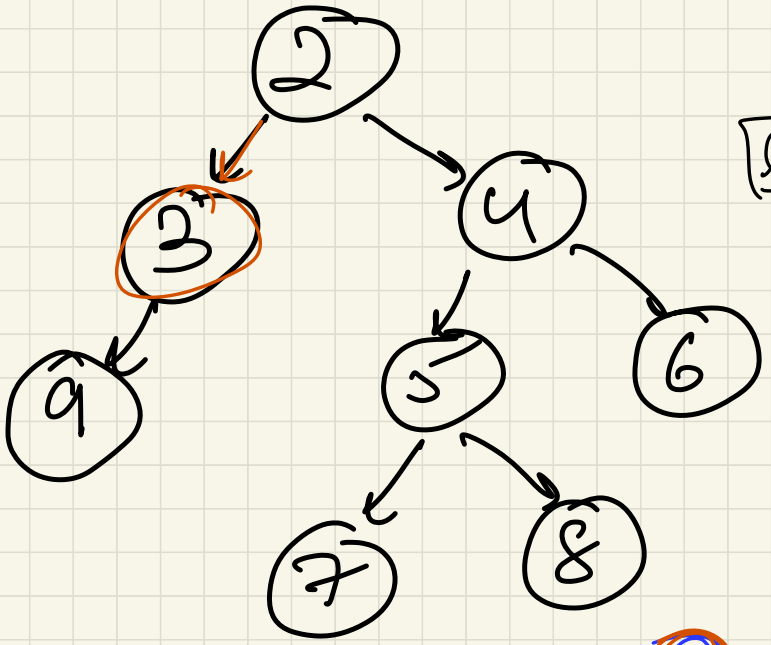
→ front to back

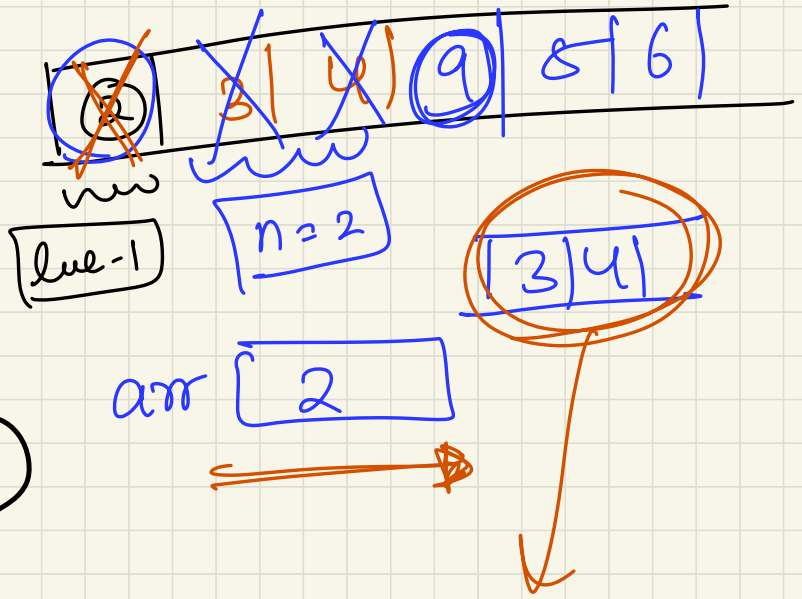
even

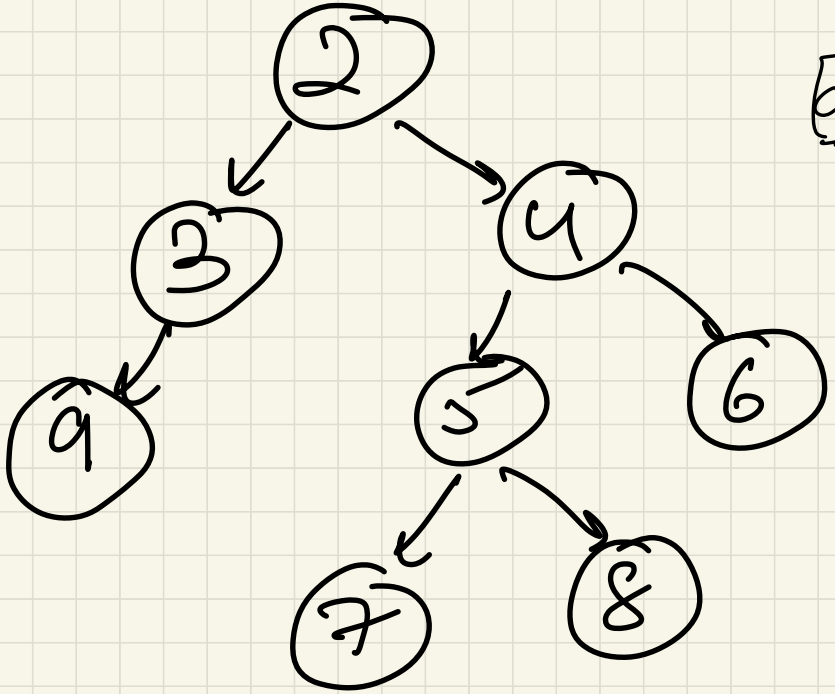
back to front


1
2
3
4
5
6
7
8


1
3
2
4
5
8
7
6


2
3
4
9
5
6
7
8


2
3
4
9
5
6
level-1
n=2
3
4
arr
2
level = 1
2
even

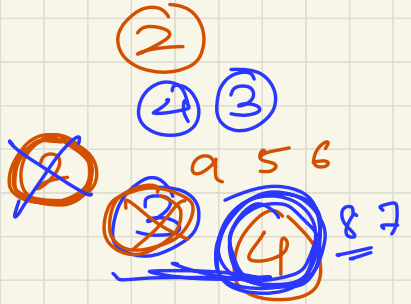


9 | 5 | 6 | 7 | 8

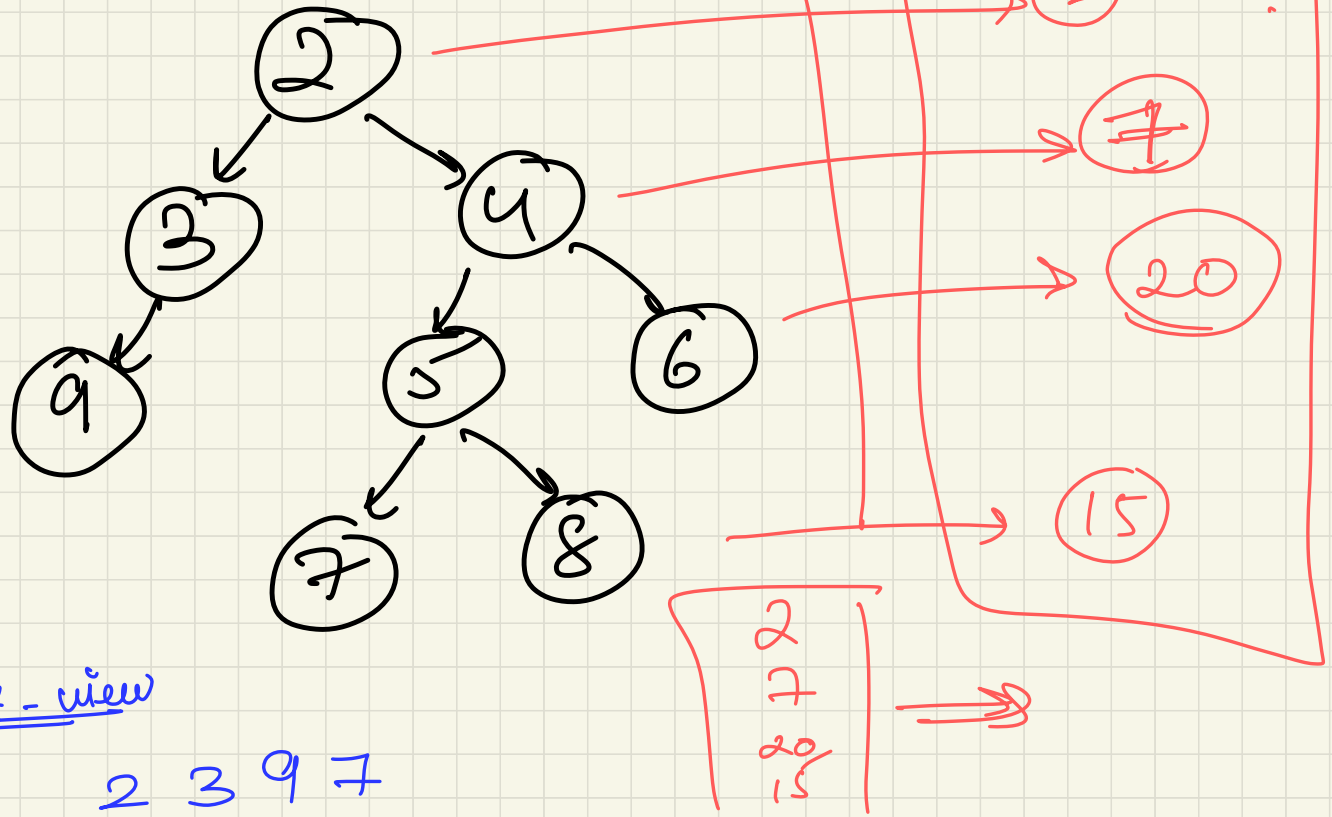
9 | 10 | 5 | 6

7 | 8

8, 7



(maximum sum of a level)

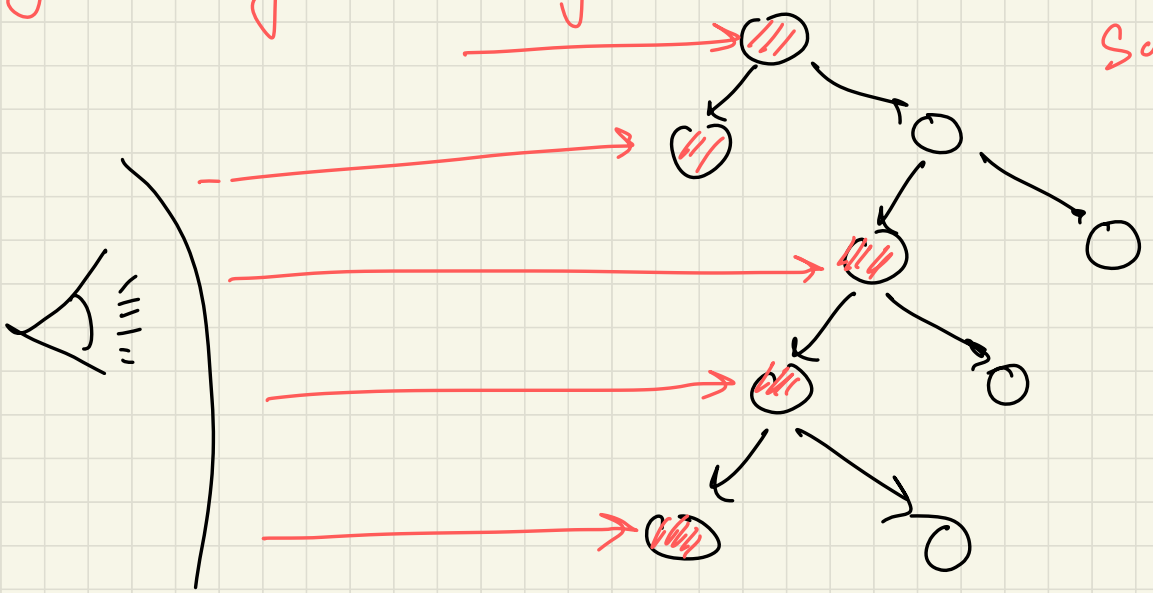


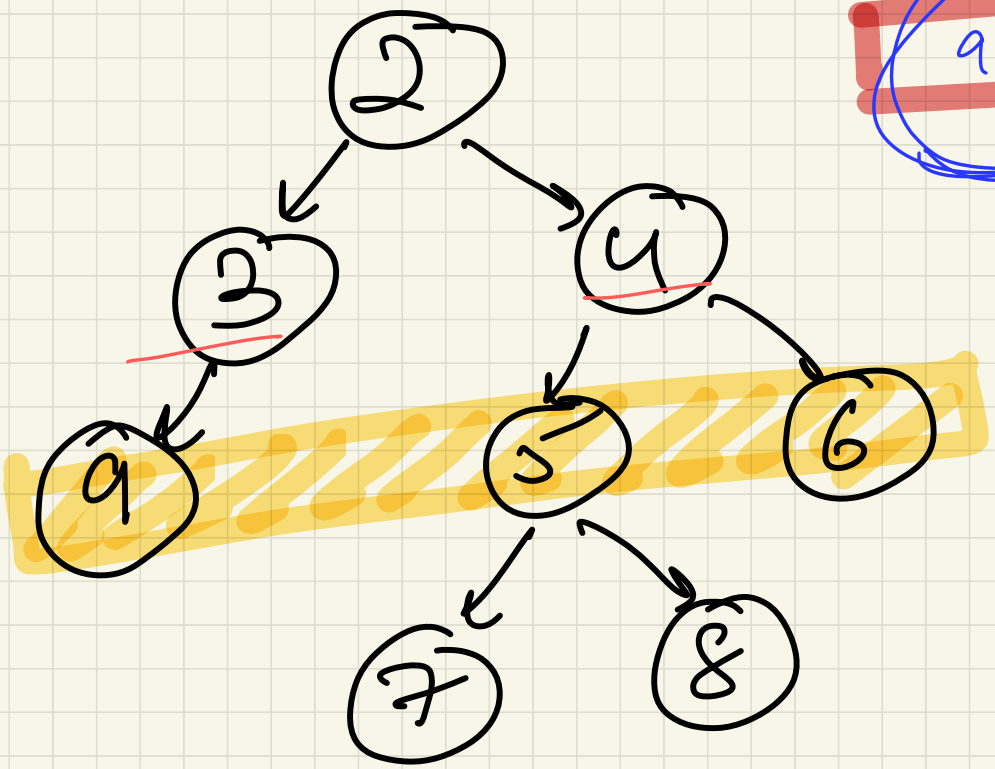
left-view

→ 2 3 9 7

left view of a Binary Tree

Sabse - left

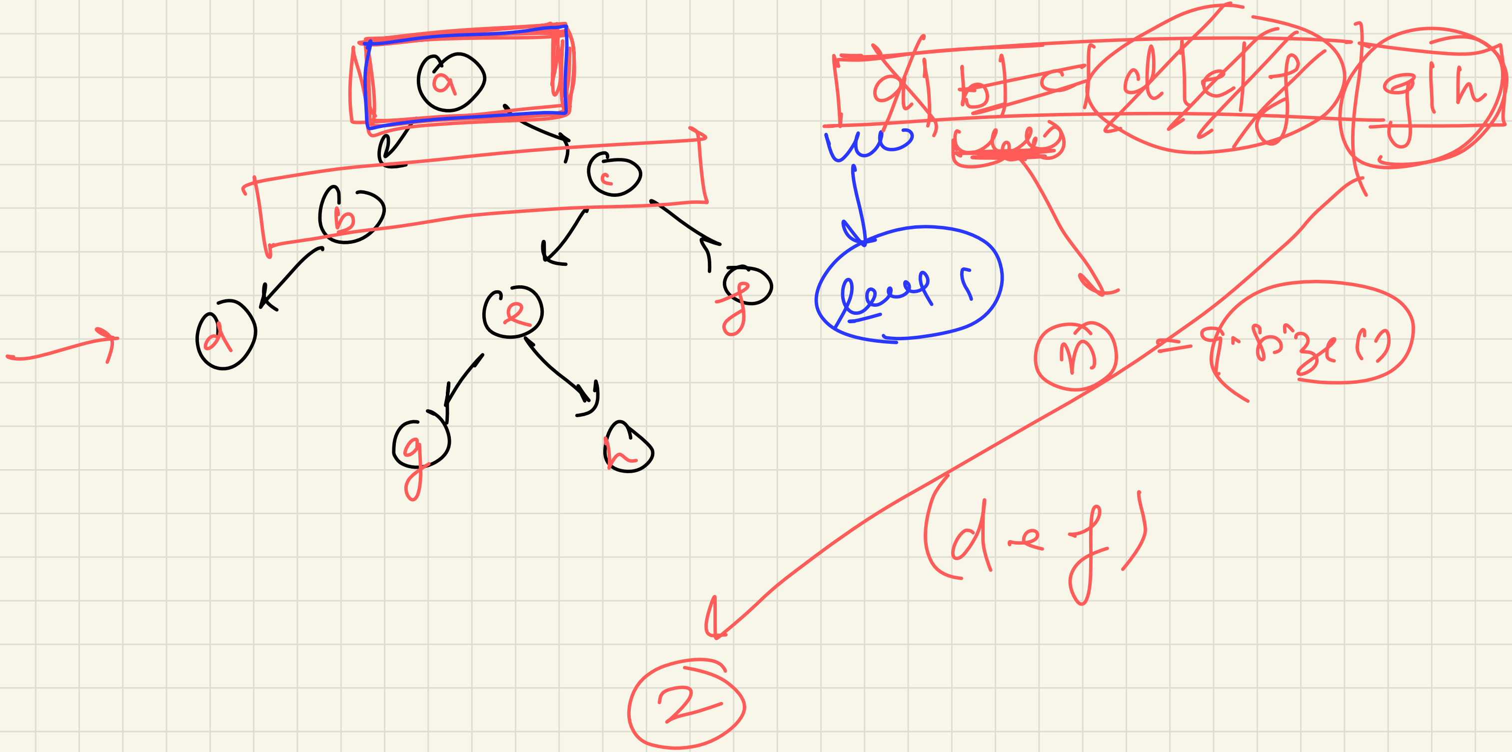


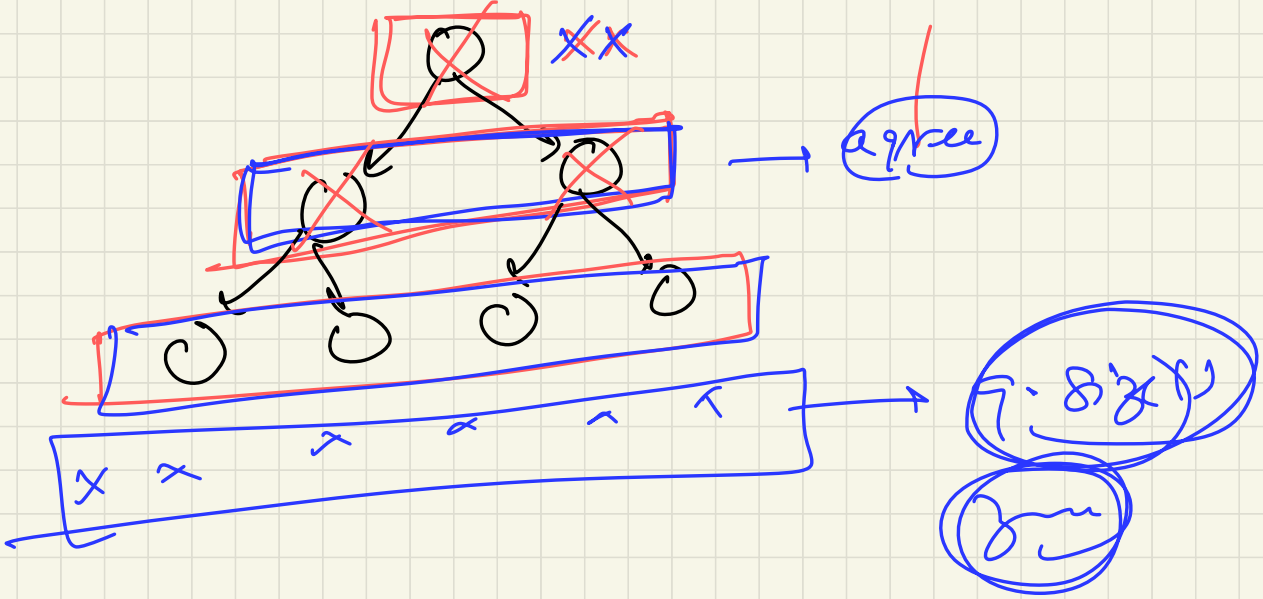


9 | 5 | 6

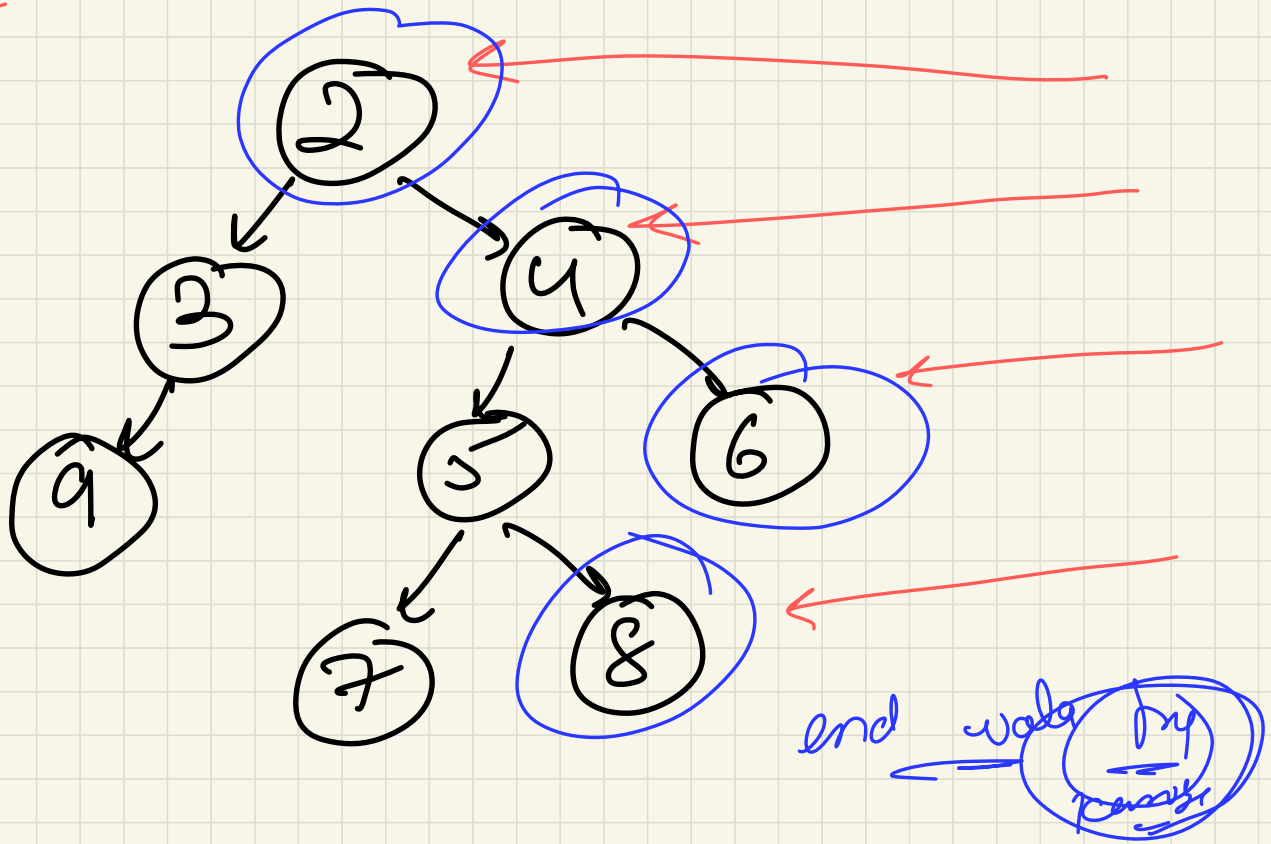
q.size()

level ke iske elements


a
b
c
d
e
f
g
h
level 1
n
(d e f)
2



right view

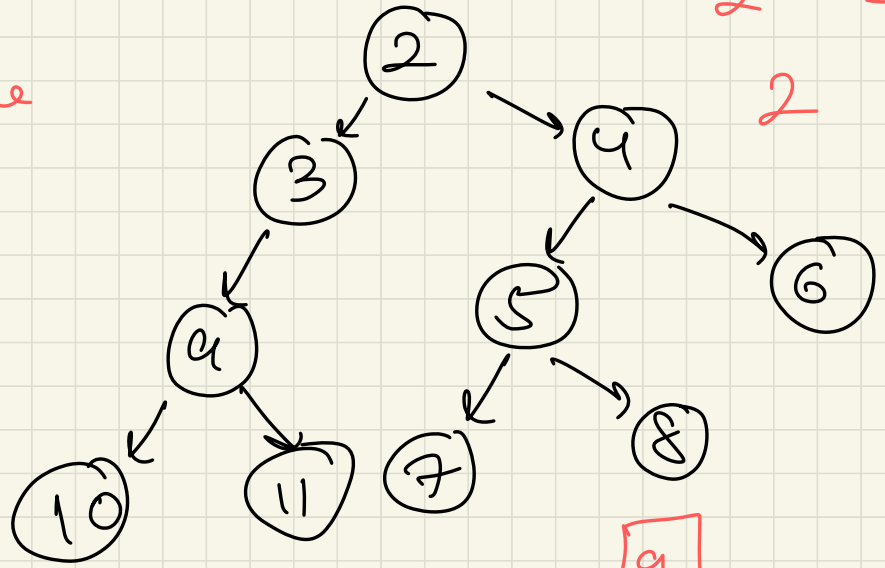


longest common Ancestor

Home-work

root → node

print



2 3 9 10

2 3 9

Recursion

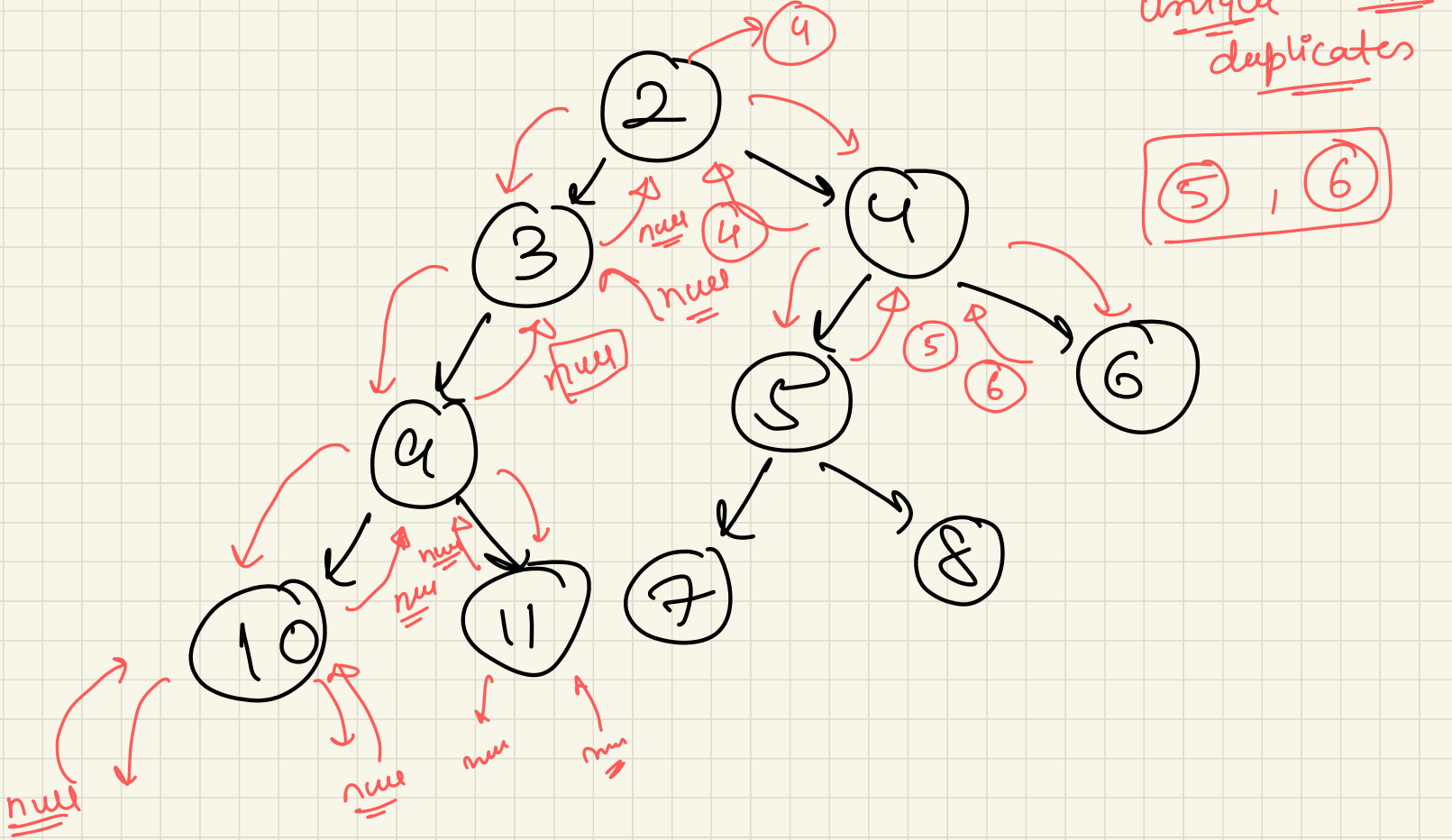
path

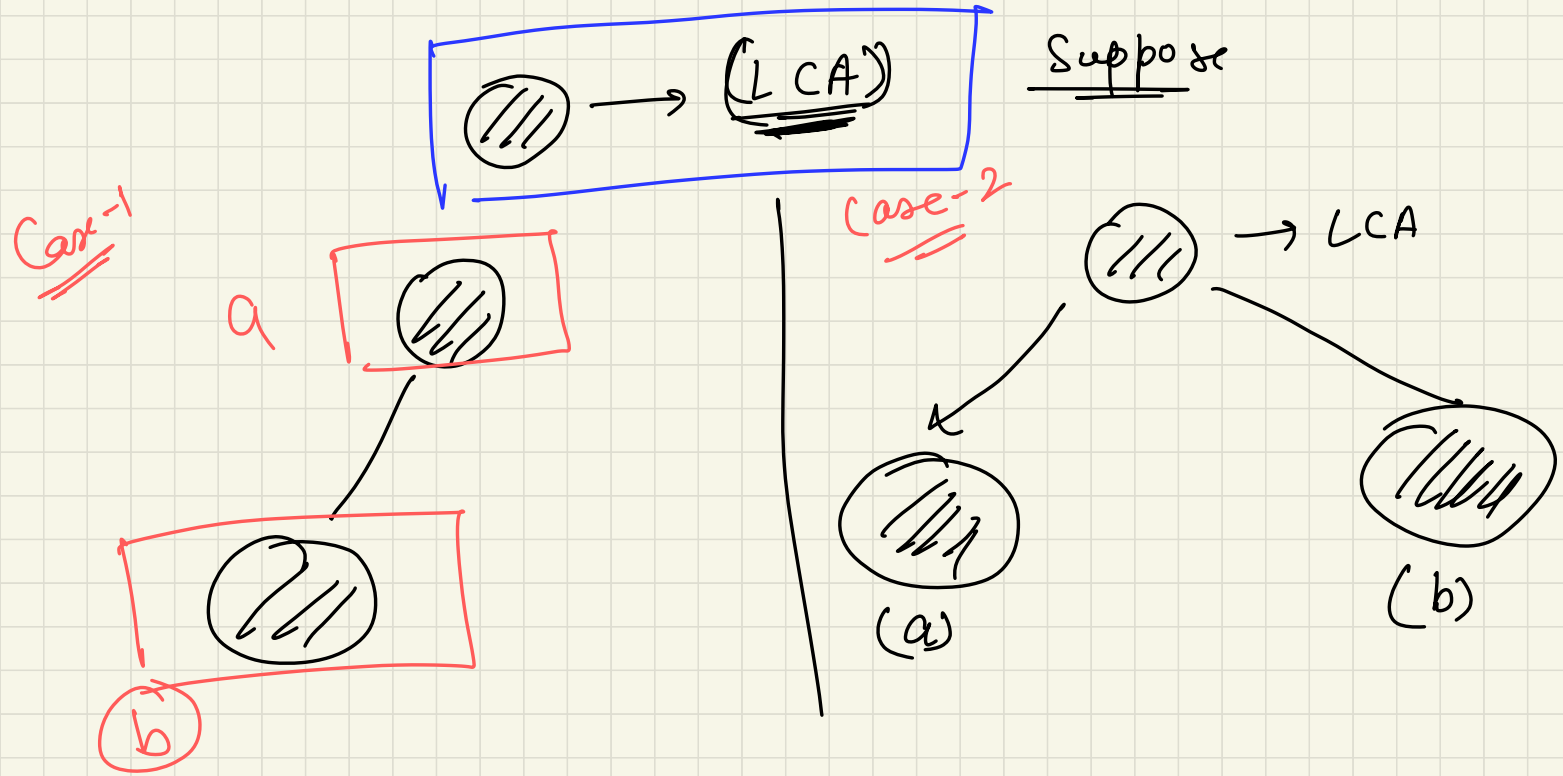
→ [2 4 5 7]

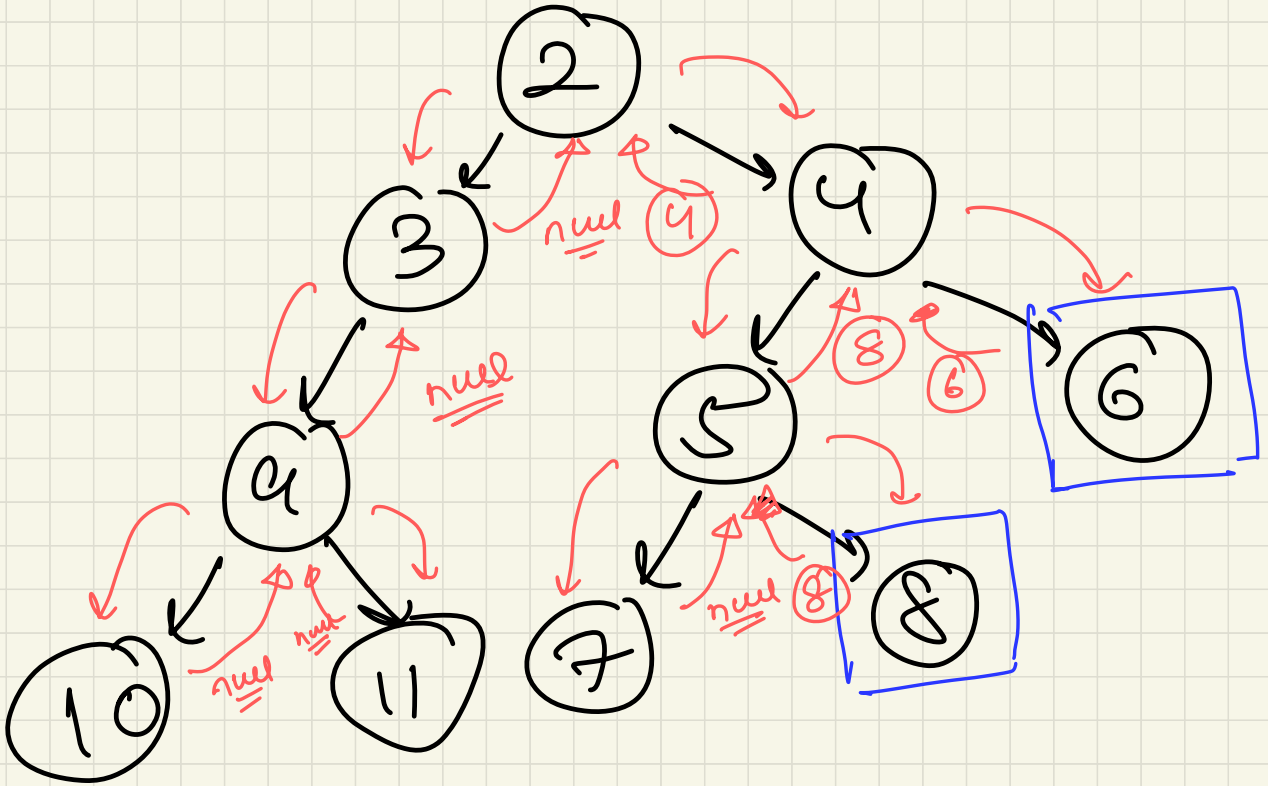
9

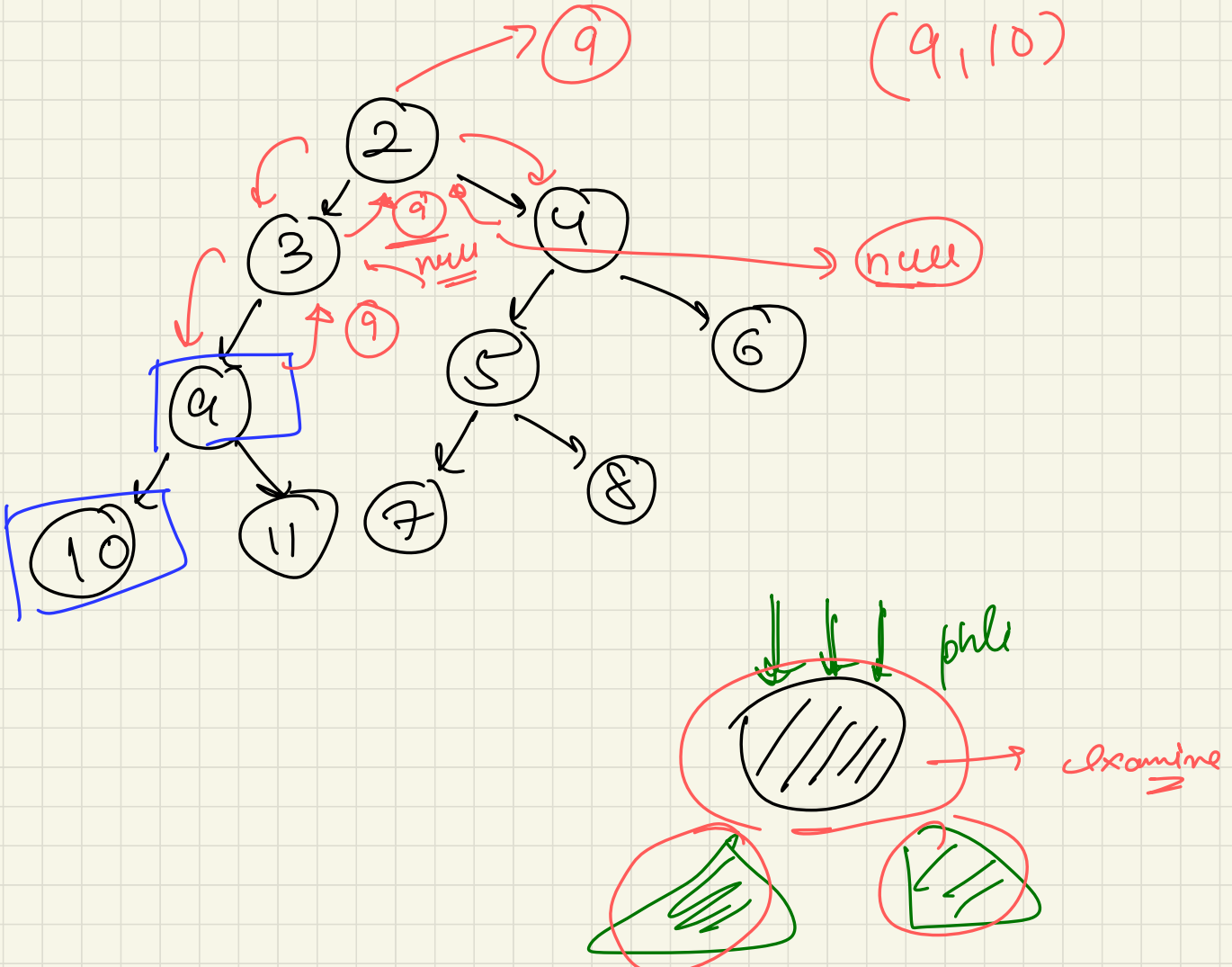
[2 3 9]

path :-


unique Constraint
duplicates
5 , 6
2
4
3
4
4
null
null
null
4
5
6
5
6
7
8
10
11
null
null
null
null
null
null


LCA
Suppose
Case-1
a
b
Case-2
LCA
(a)
(b)

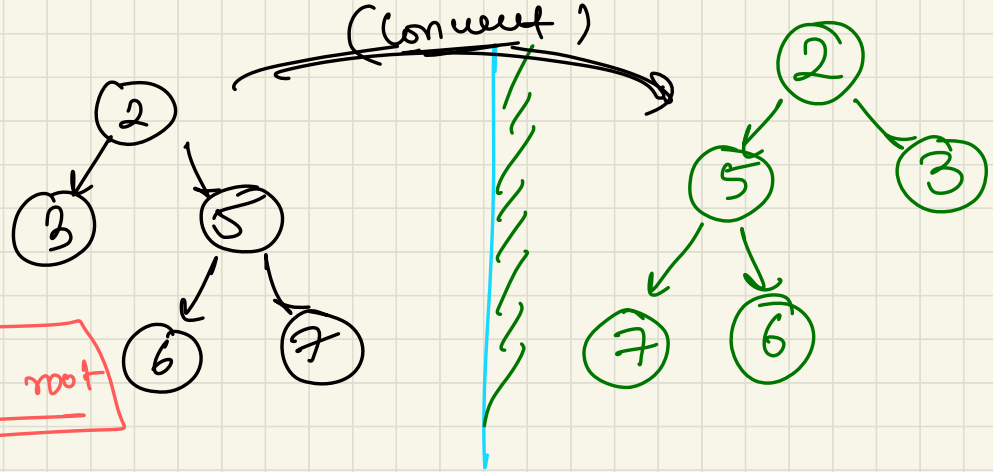

2
3
4
9
5
6
8
6
4
null
null
10
null
null
11
7
null
8
8


9
(9,10)
2
3
9
9
null
null
9
4
5
6
10
11
7
8
phle
examine

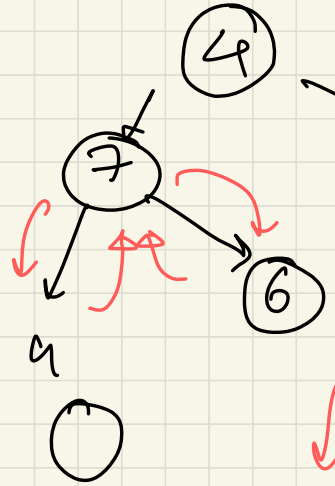
post-order traversal

important

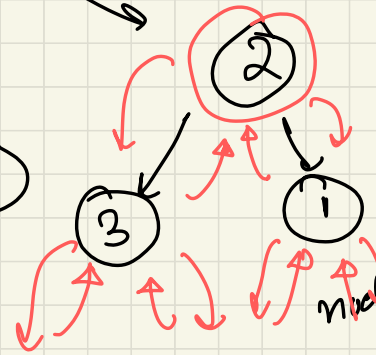
left, right, root



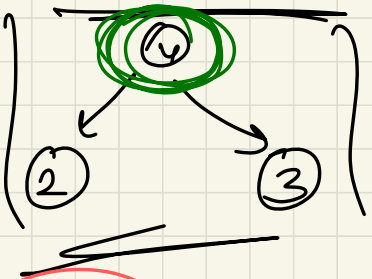
post order



right



Smallest - problem

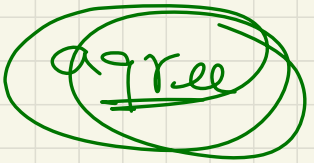


clear hai

smj

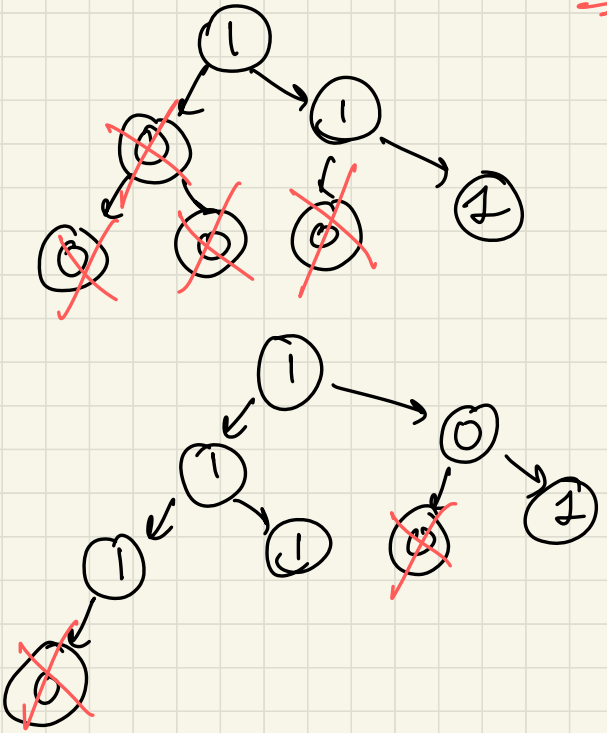
(111)

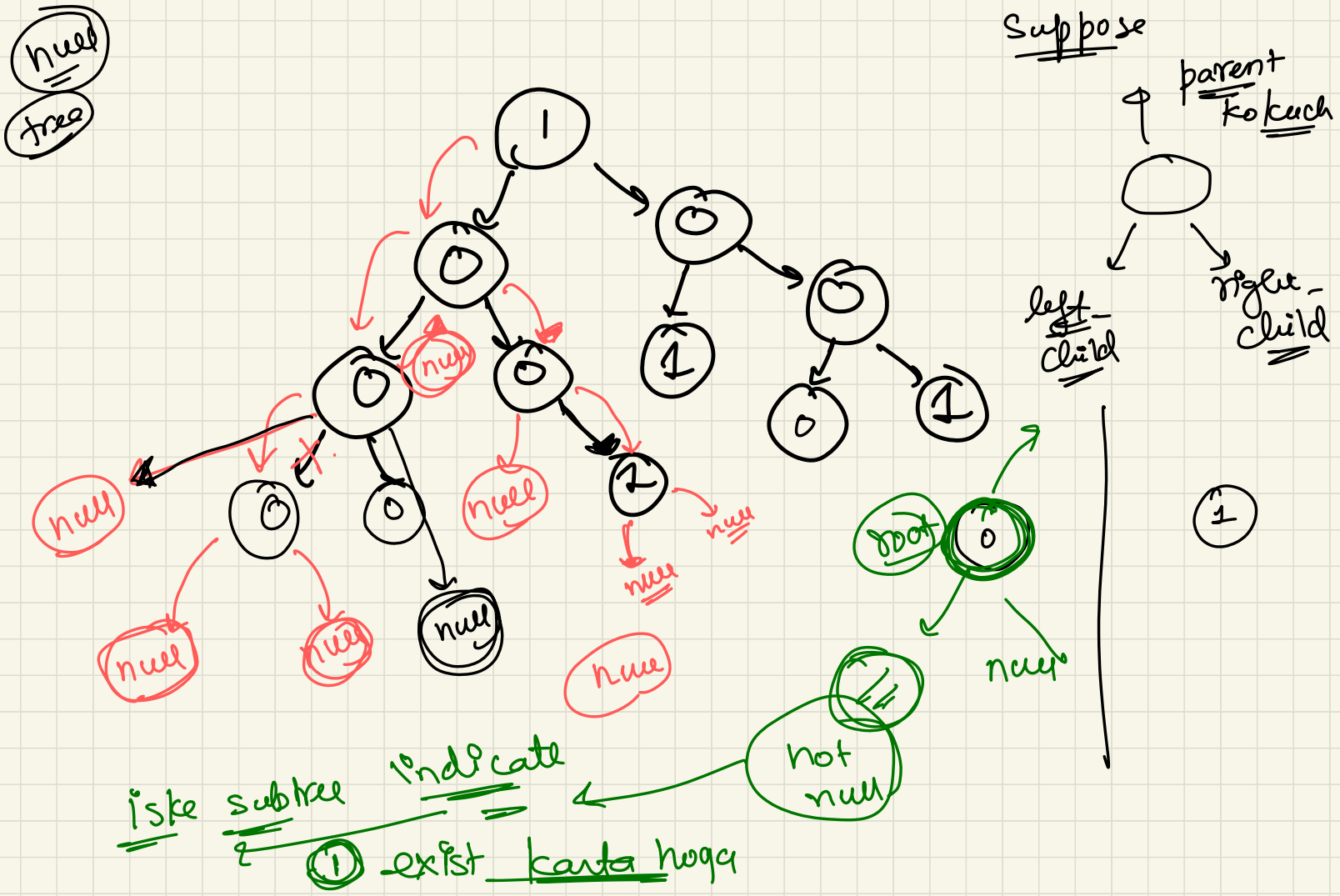
Swap (root → left, root → right);

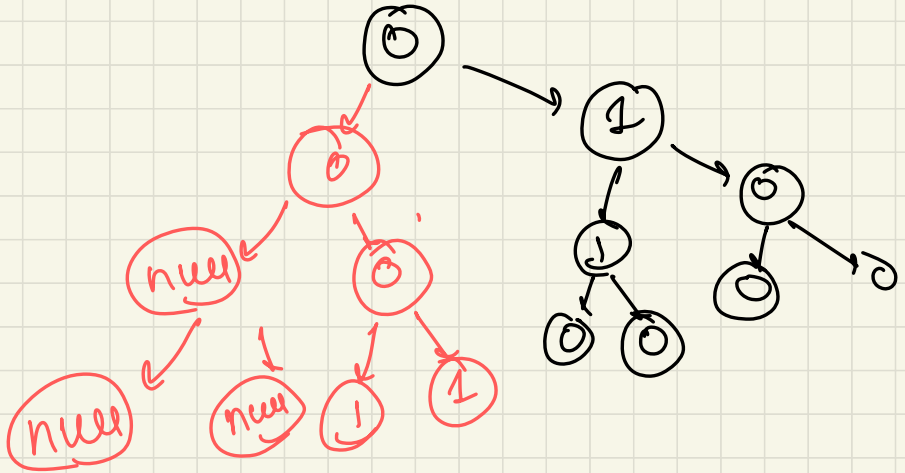


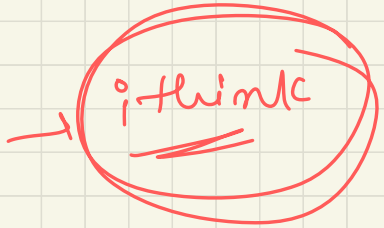
Binary Tree pruning

example



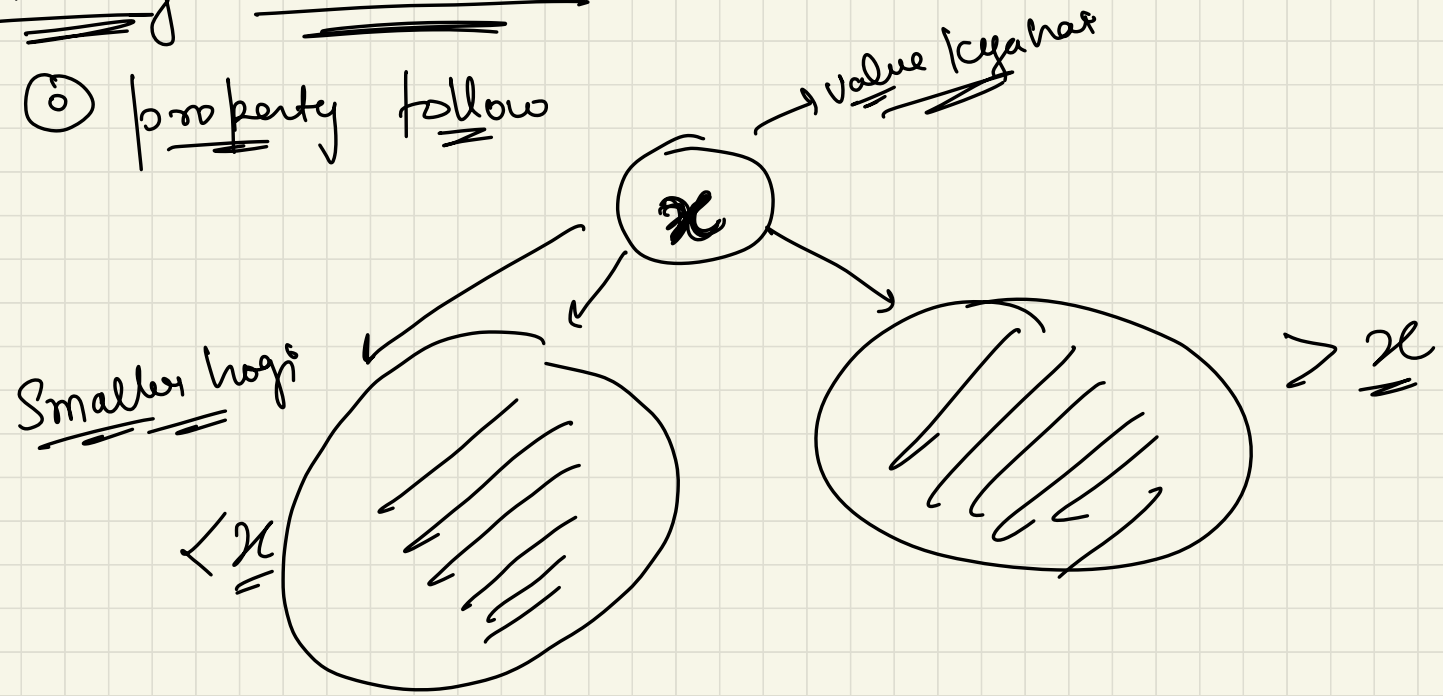

null
tree
Suppose
parent ko kuch
left_child
right_child
null
root
1
not null
null
jiske subtree
indicate
1 exist karta hoga


0
0
1
null
0
1
0
null
null
1
1
0
0
0
0

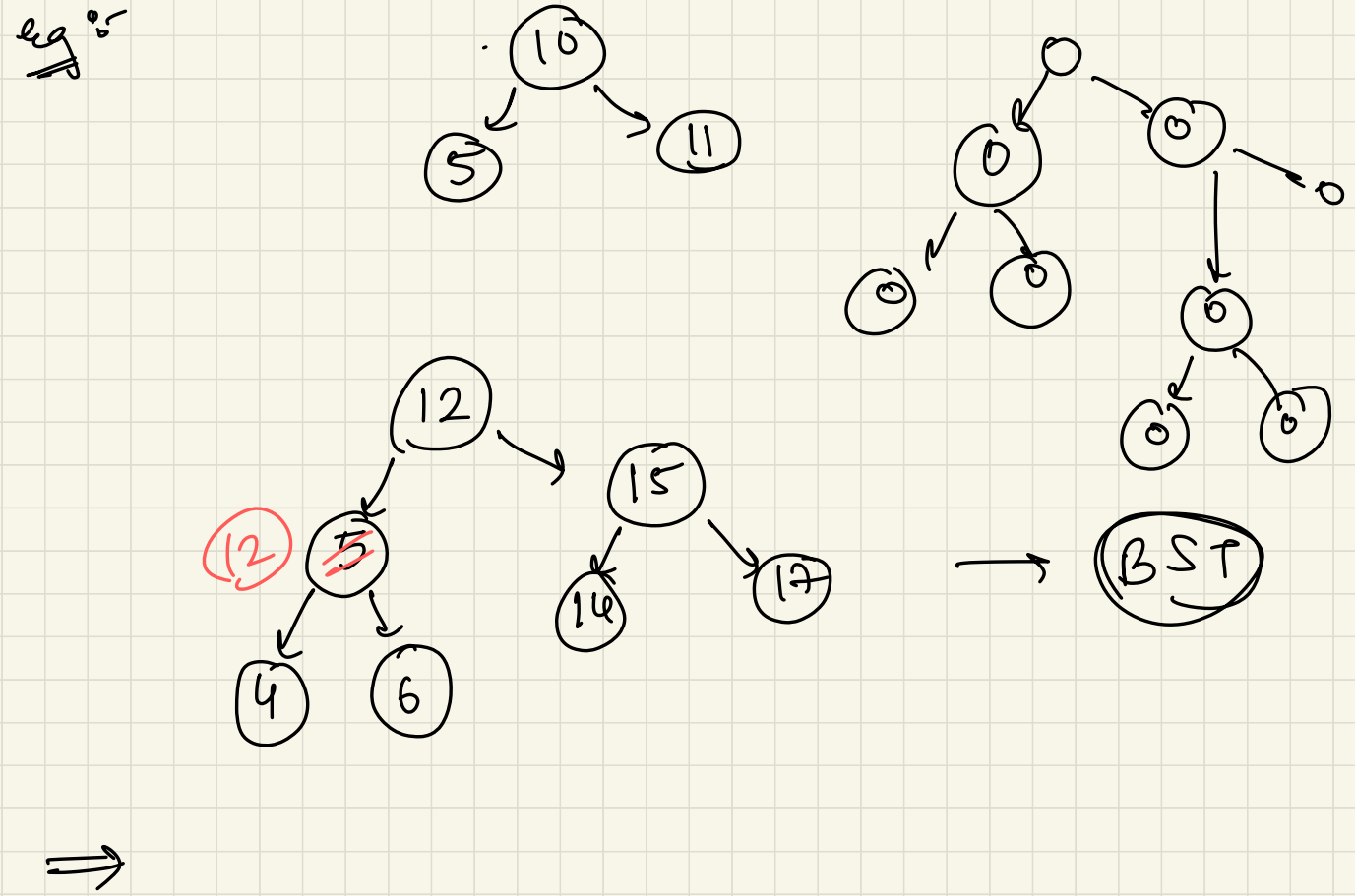

jthink

Binary Search Tree

① property follow

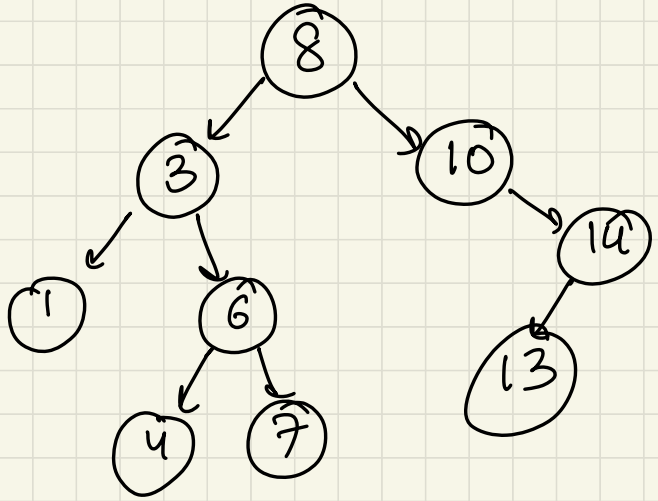


eg :-



⇒

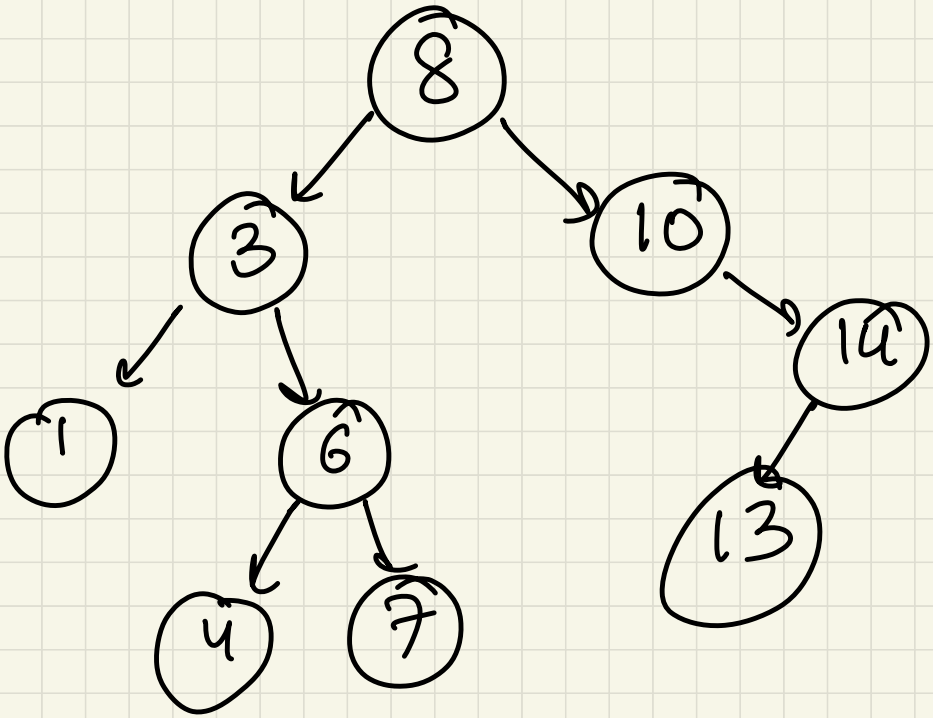
BST



(inorder) traversal

inorder :- Sorted hoti

important



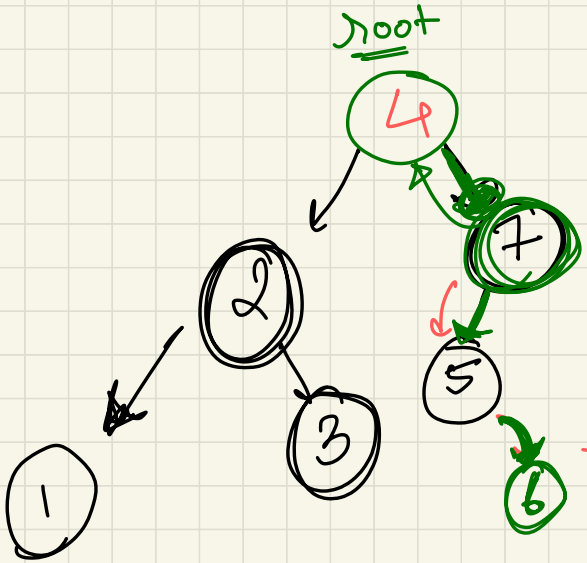
searching $\log(n)$

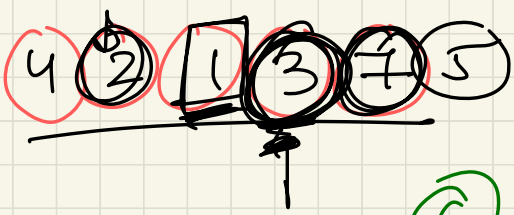
6

no of nodes in tree

4

5





6

Smj aaya

6

Smj aaya

4 right

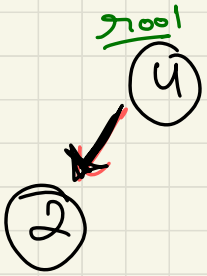
(7, 6)

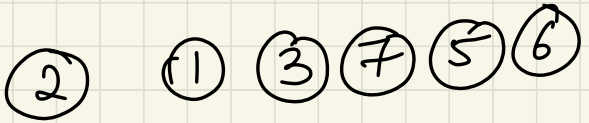
7 left 5

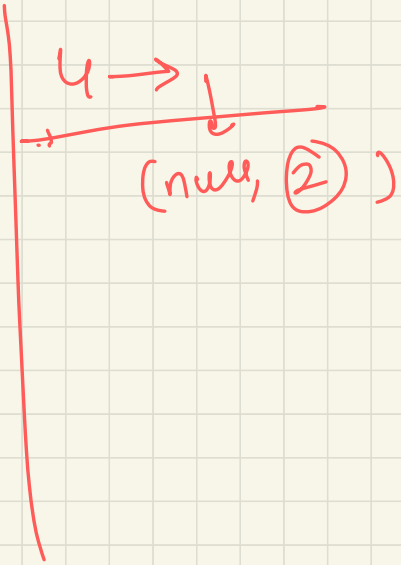
(5, 6)

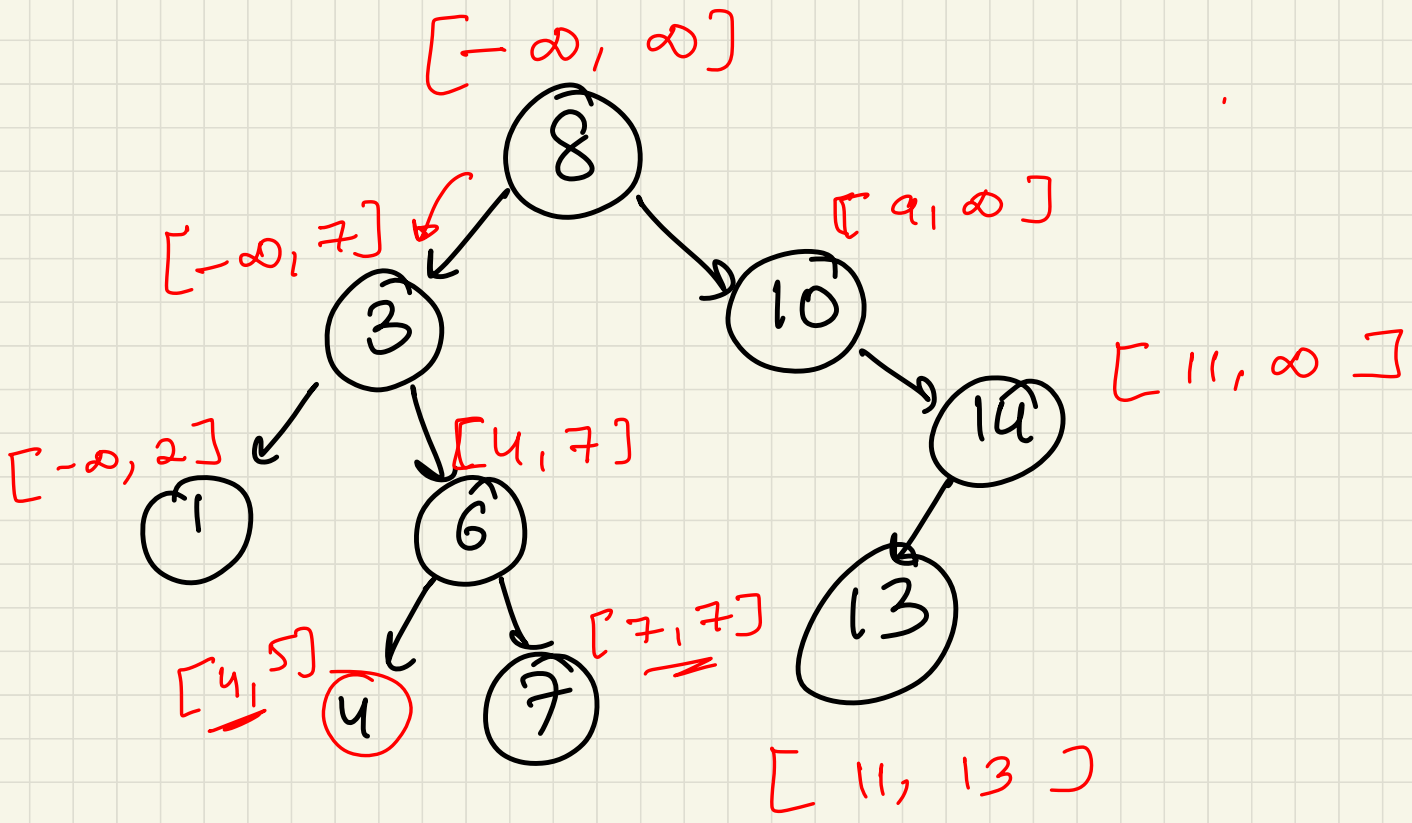
5 right

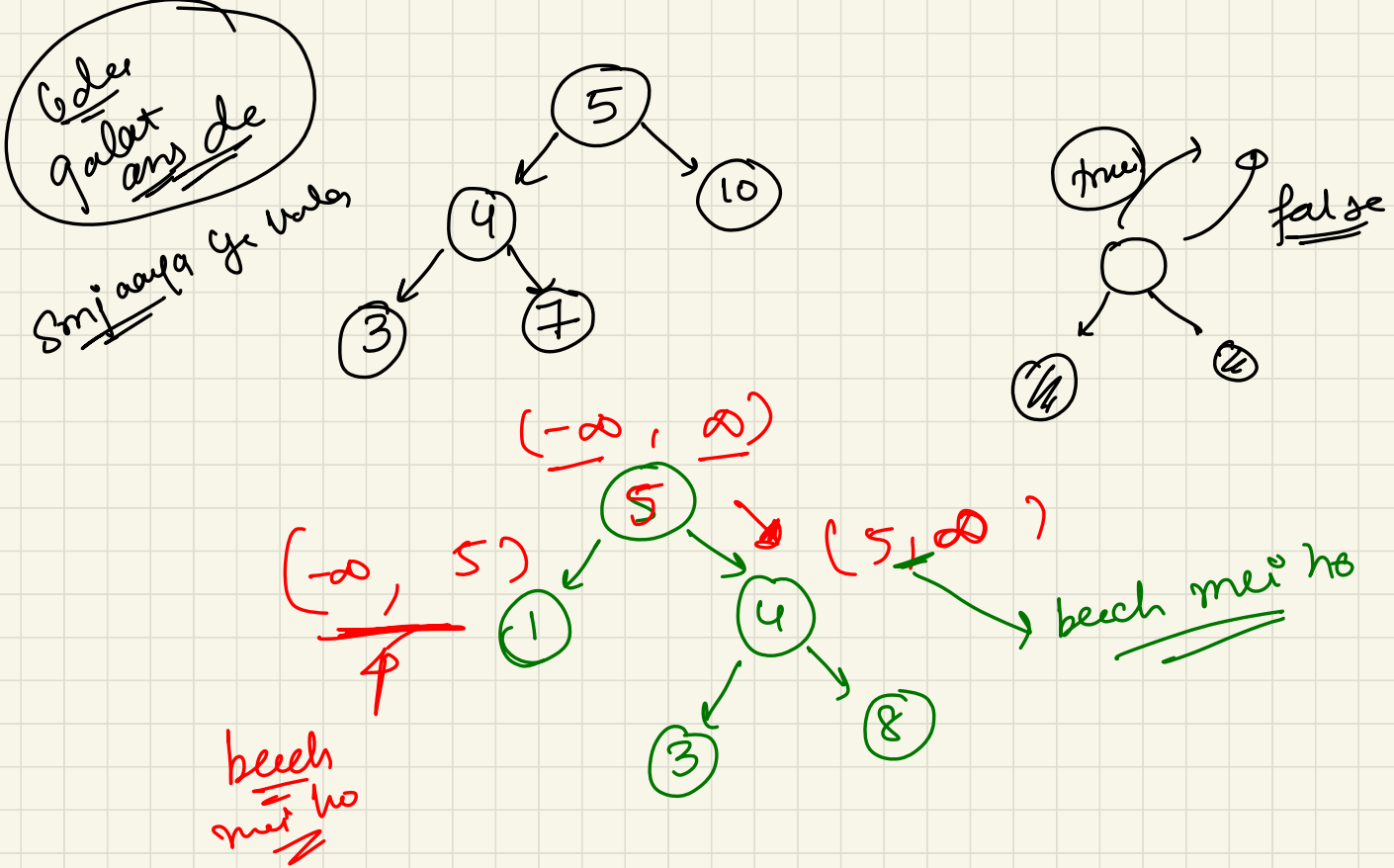
(null, 6)

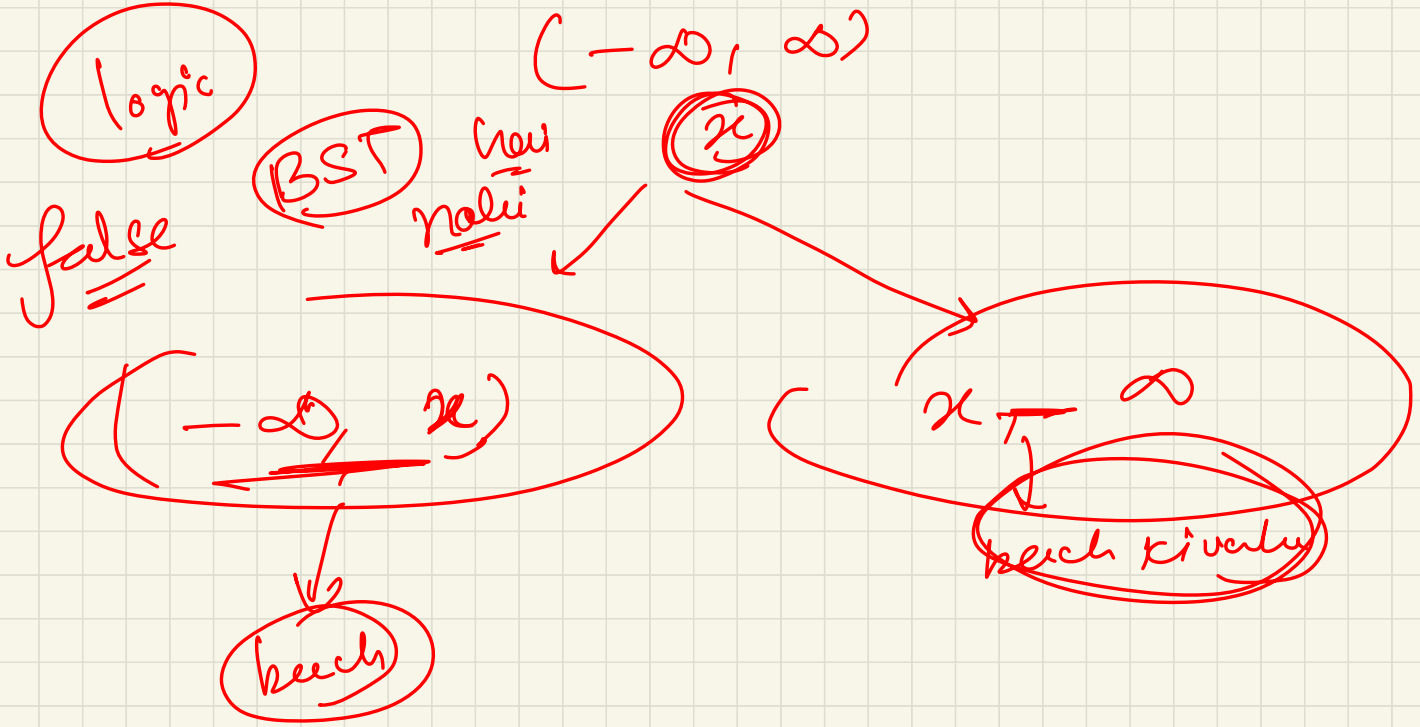

root
4
2

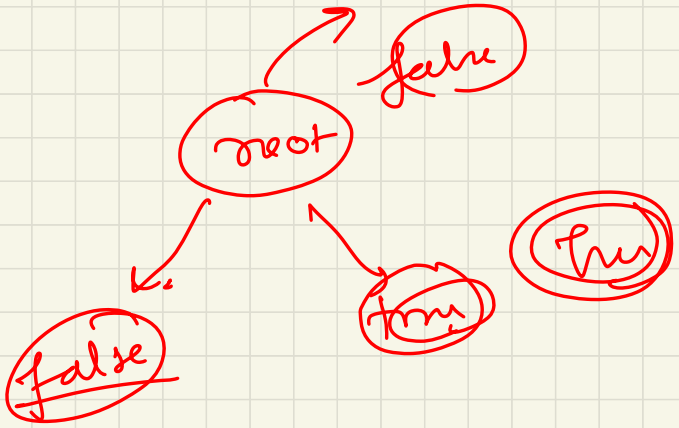

2
1
3
7
5
6


4
(null, 2)

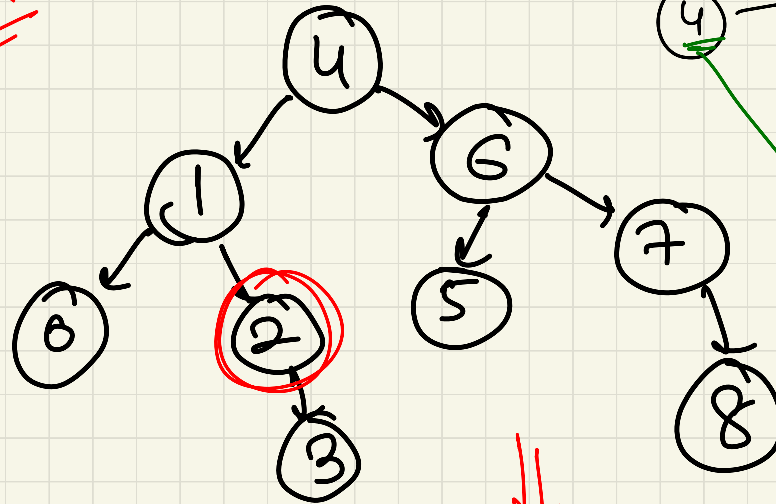

[-∞, ∞]
8
[-∞, 7]
3
[9, ∞]
10
[11, ∞]
14
[-∞, 2]
1
[4, 7]
6
13
[7, 7]
[4, 5]
4
7
[11, 13]

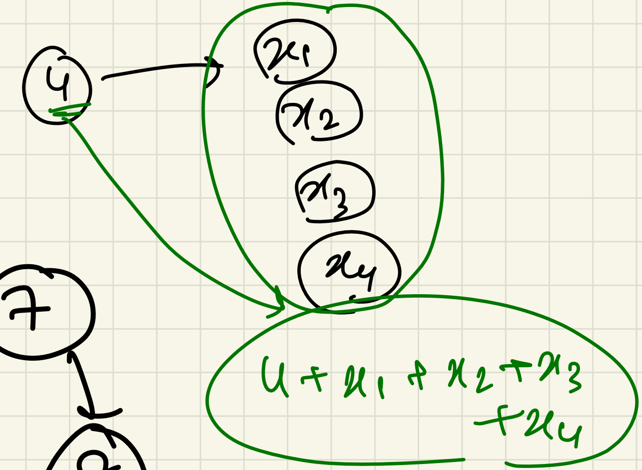

Coder
galat
ans de
Samjaaya ya nahi
5
4
10
3
7
true
false
(-∞ , ∞)
5
(-∞ , 5)
(5, ∞)
1
4
beech mei ho
3
8
beech
mei ho


logic
BST
false
$(-\infty, \infty)$
x
$(-\infty, x)$
(x, ∞)
recursively
recursively

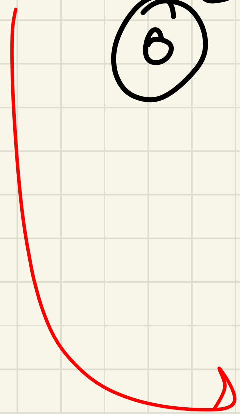

false
root
false
true
true

given a tree

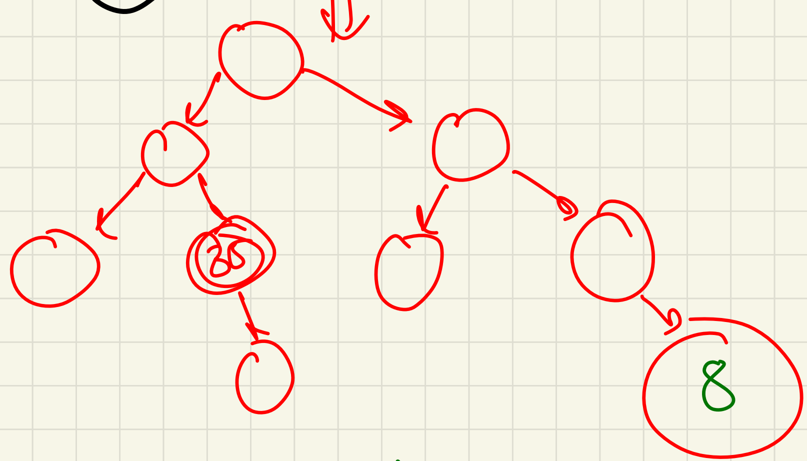




Replace

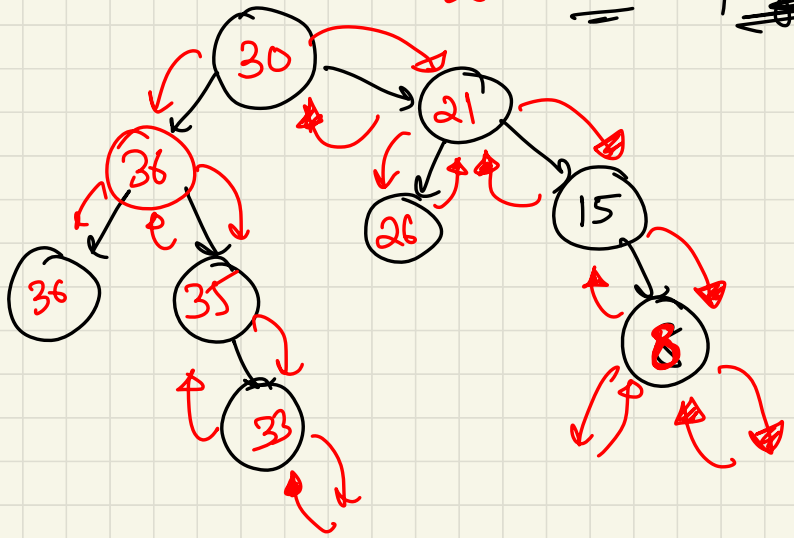


sum



inorder-wise

Sum = ~~0~~ ~~8~~ ~~15~~ ~~21~~ ~~26~~ ~~30~~ ~~33~~ ~~35~~ 36 Value ° Right-most node hai



agr

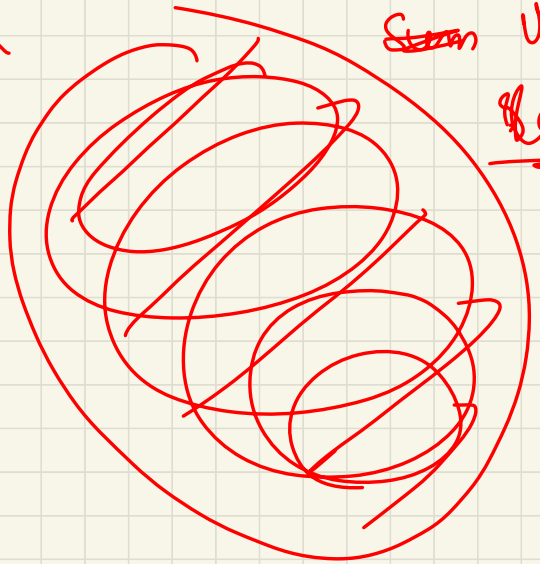
Value = 7

Sm° aaya

Binary - Search - Tree

Intuition

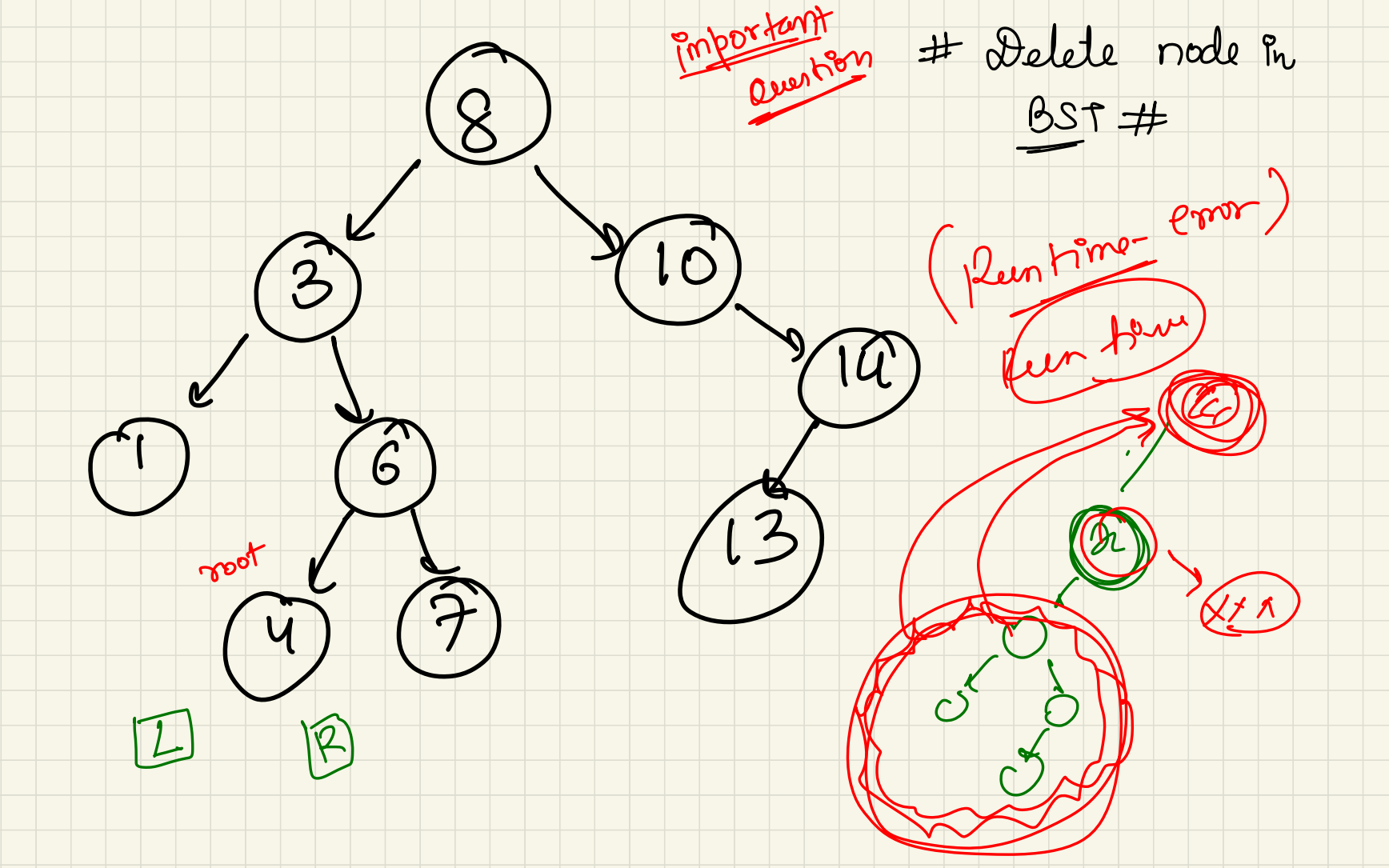
2ℓ



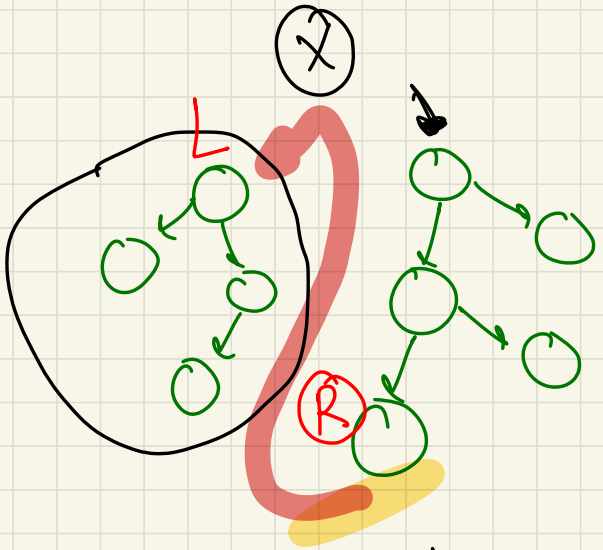
right( )

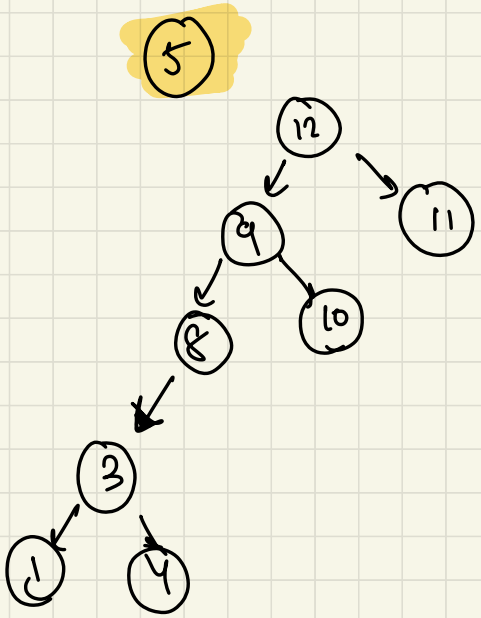
~~Sum~~ Val = Val + Sum;

Eff (1)

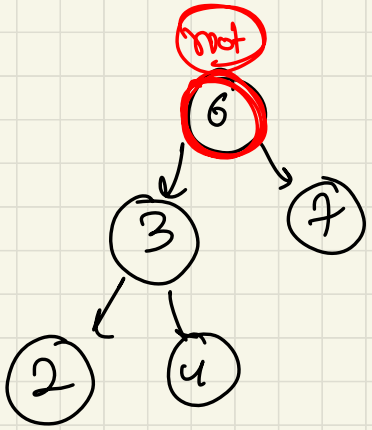

important question
Delete node in BST
8
3
10
1
6
14
4
7
13
root
L
R
(Runtime- error)
Recursion
cc
dr
1+1

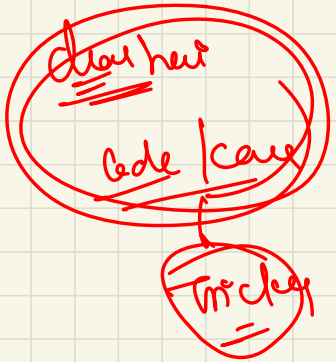
Core-work

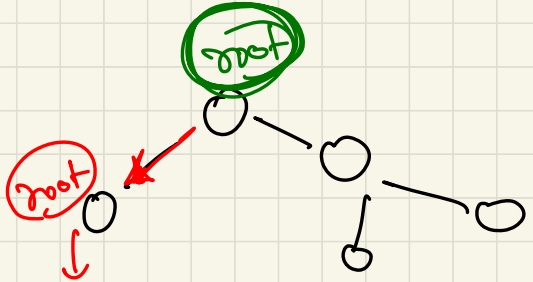




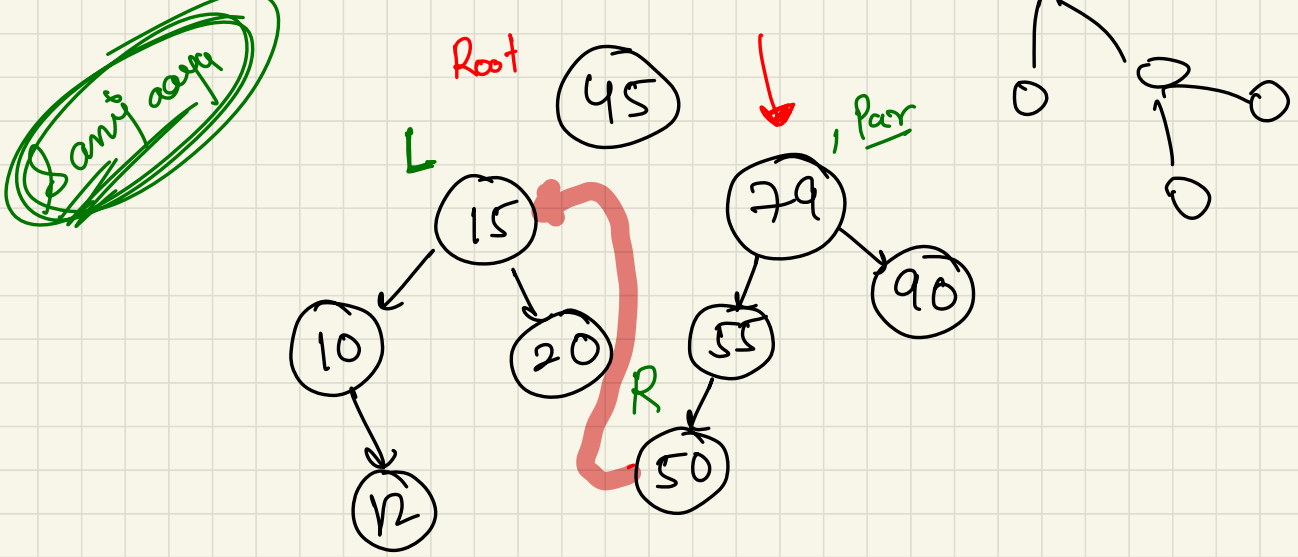
① right - Subtree ka extreme left wali node par jaya ko jaya left - Subtree attach kar denge.







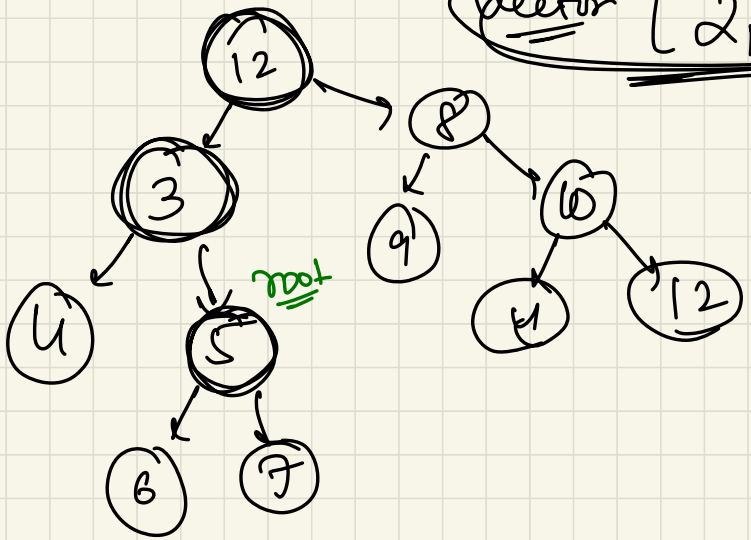
45


Sanjaay
Root
45
L
15
79
Par
10
20
55
90
R
50
12

Question:-

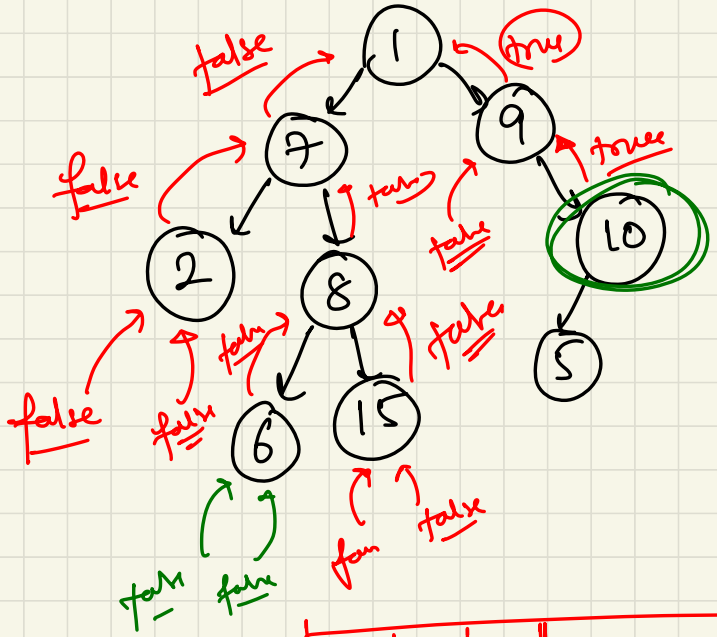
vector [2, 3, 5]

Path



root

Back-tracking



path kese nikalenge

clear hai

10

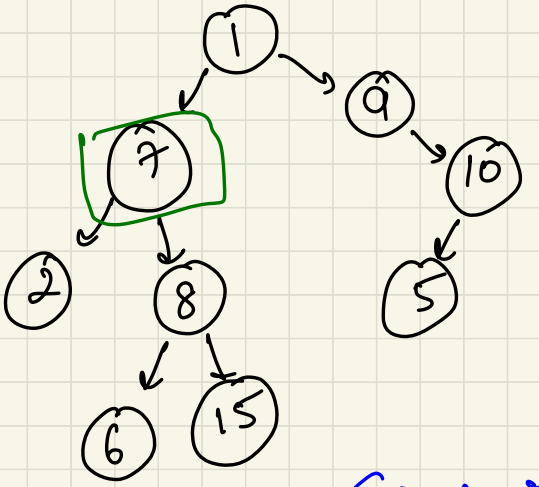
path | 1 | 9 | 10 |

(parent → child)

child → parent

nahi

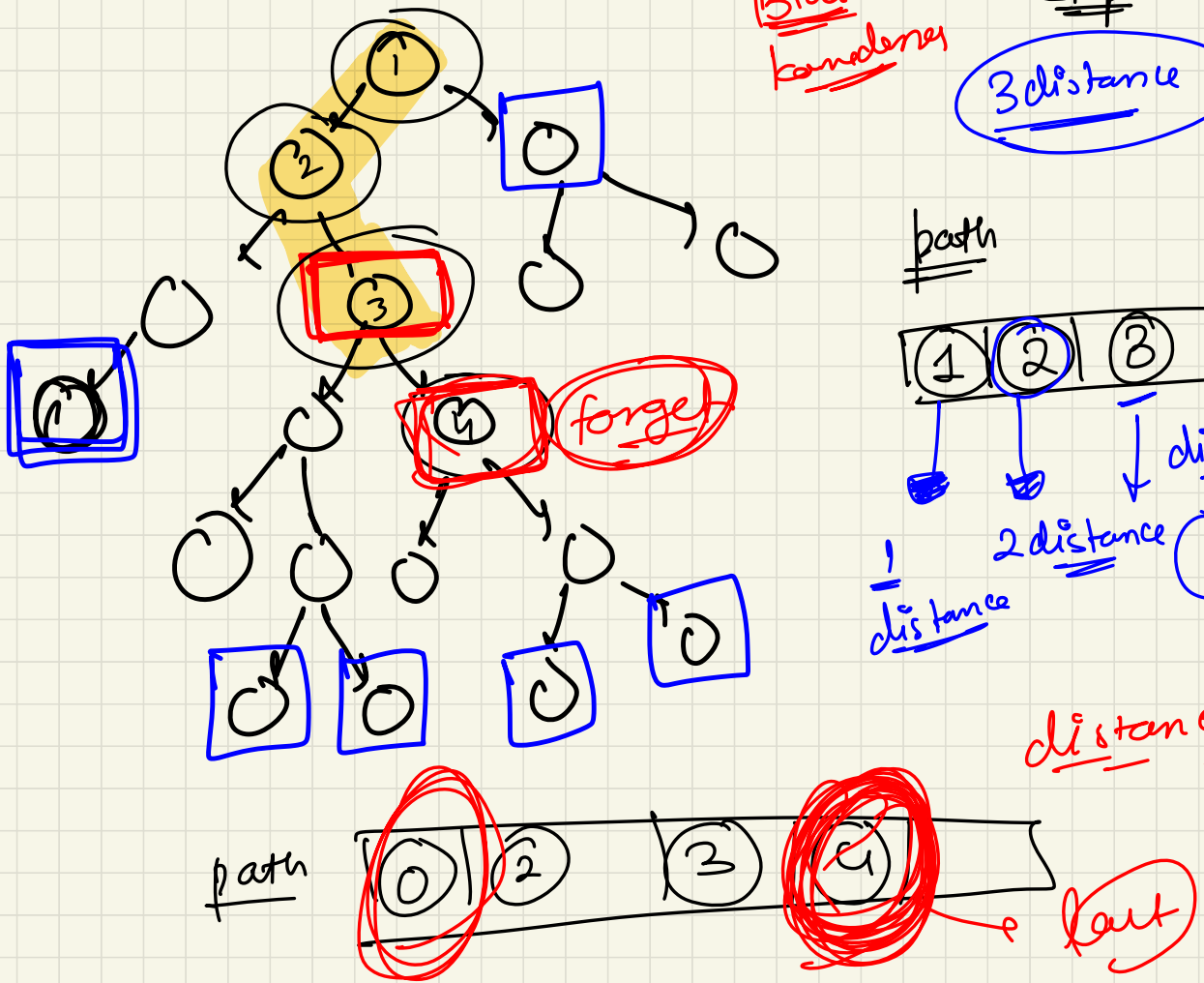
(Amazon)

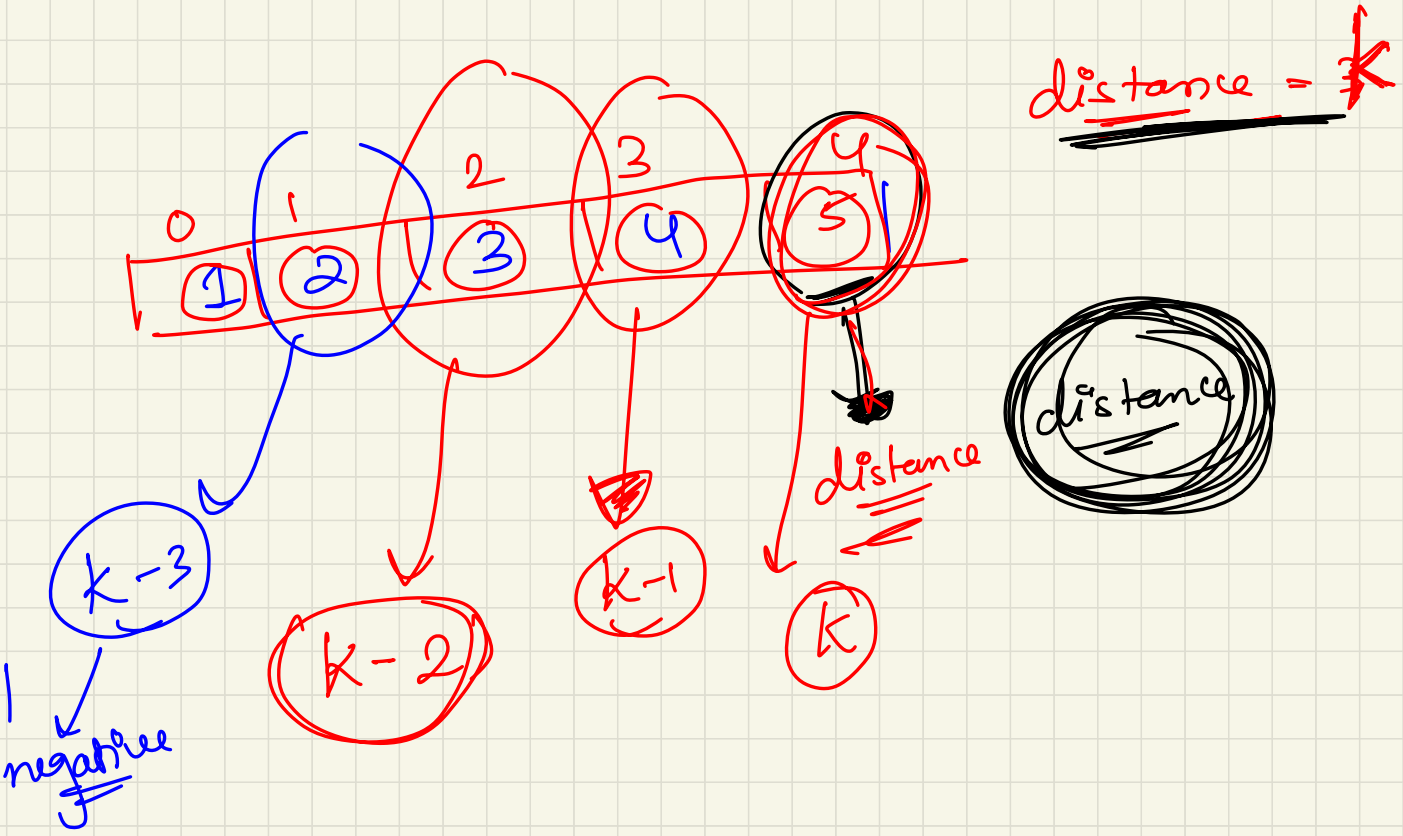


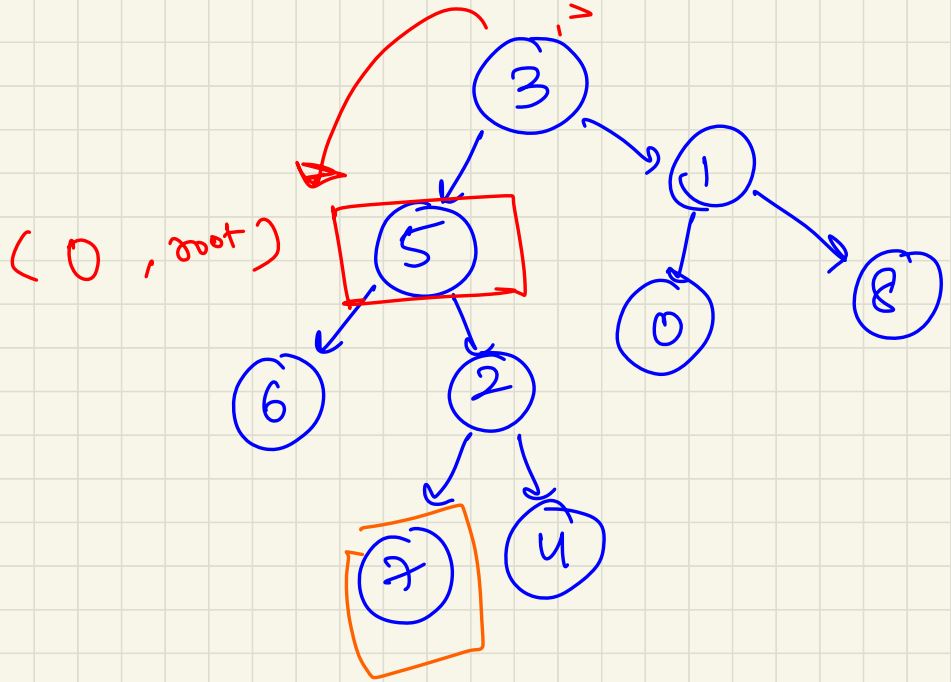
2 distance

[6, 15, 9]

Blocking node → (Technique)


Block kemudian
Steps !
3 distance
path
1
2
3
distance
2 distance
3
1 distance
forget
distance ??
path
0
2
3
4
fault


distance = k
0
1
2
3
4
5
1
2
3
4
5
distance
k-3
k-2
k-1
k
distance
1
negative

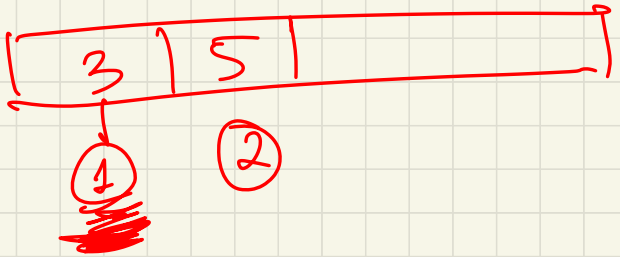


(0, root)

Ans :- 3 4

k = 2

path =



Start karde

Shortest subarray

no of ele

(2 4 3 8 7 9)

1 2 3 4 5 5

R

O(log

2 4 4 8 8 9

2 3 3 7 7 9

start