Maximum Width of BT

662. Maximum Width of Binary Tree

Solved

Medium | Topics | Companies

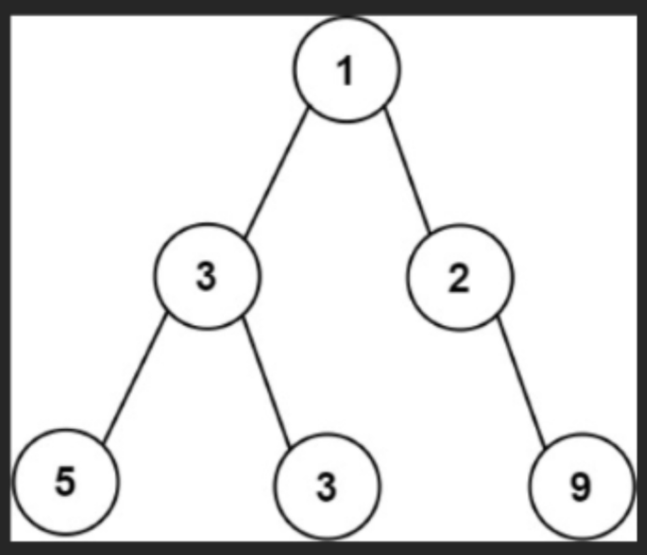
Given the `root` of a binary tree, return *the **maximum width** of the given tree*.

The **maximum width** of a tree is the maximum **width** among all levels.

The **width** of one level is defined as the length between the end-nodes (the leftmost and rightmost non-null nodes), where the null nodes between the end-nodes that would be present in a complete binary tree extending down to that level are also counted into the length calculation.

It is **guaranteed** that the answer will in the range of a **32-bit** signed integer.

Example 1:




```
Input: root = [1,3,2,5,3,null,9]
Output: 4
```

let's suppose.... we have an array

[1, 3, 3, 2, 5, 9]

How many no.of nodes present in between 1 to 5 (inclusive

[1, 3, 3, 2, 5, 9]

How to get the count → ?

if somehow → we know the indexes, we can get right?

[1, 3, 3, 2, 5, 9]
0 1 2 3 4 5

Nodes in between 1 & 5 (inclusive) are → (4-0)+1 = 5

5 nodes

* if we transition above problem into ↓

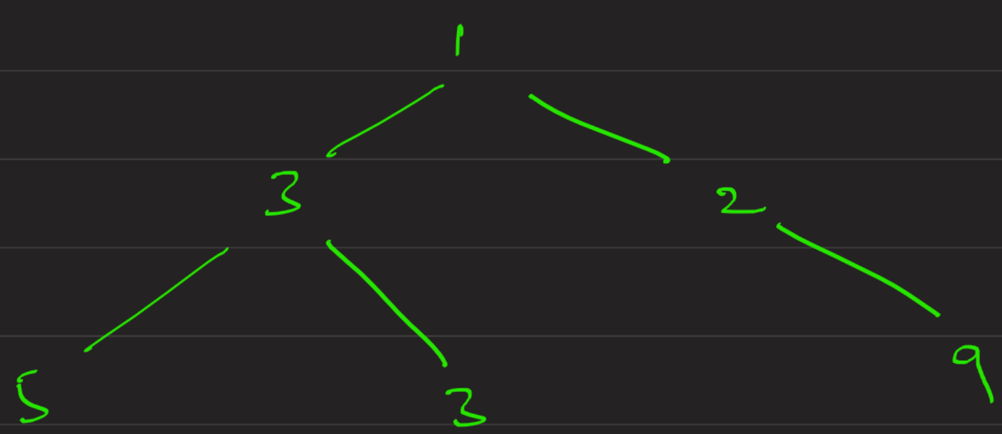
Calculate width between (1) & (5)

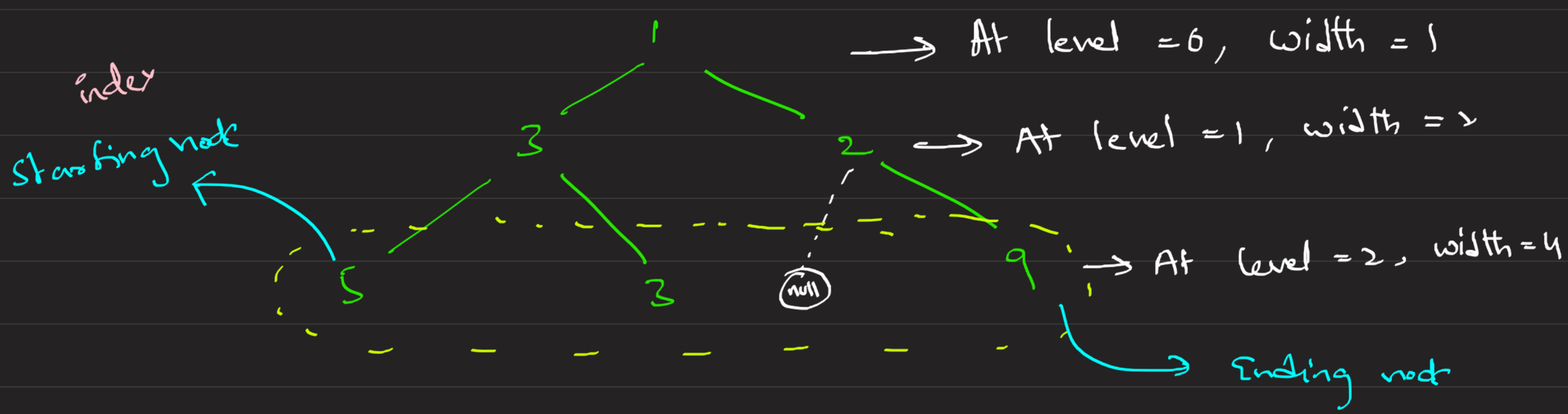
then also ... we do the same

[1, 7, 3, 2, 5, 9]

this is width

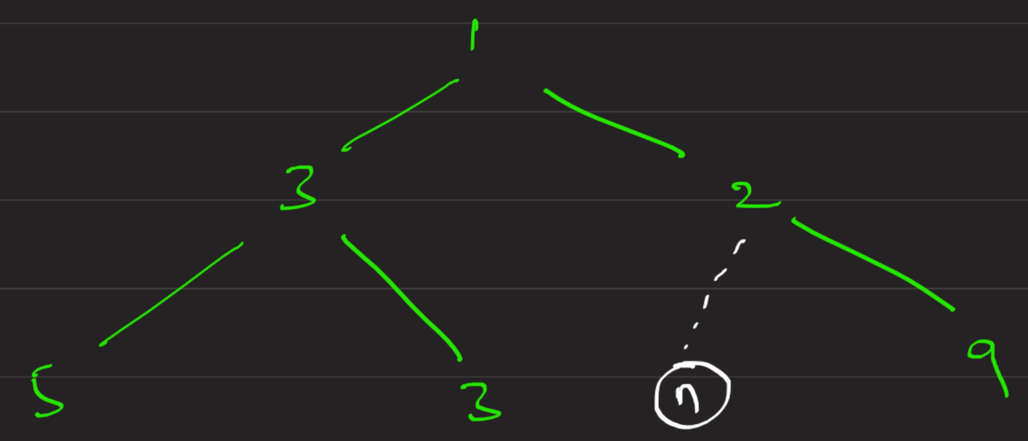
Similarly, in a Binary Tree



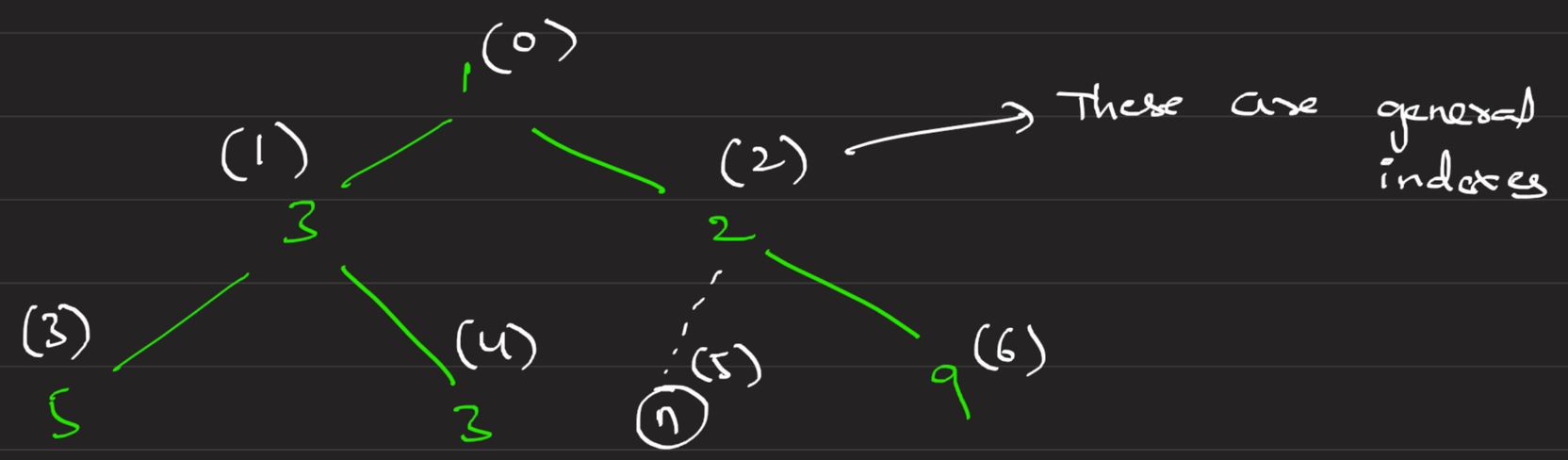


if We have a Starting node & Ending node

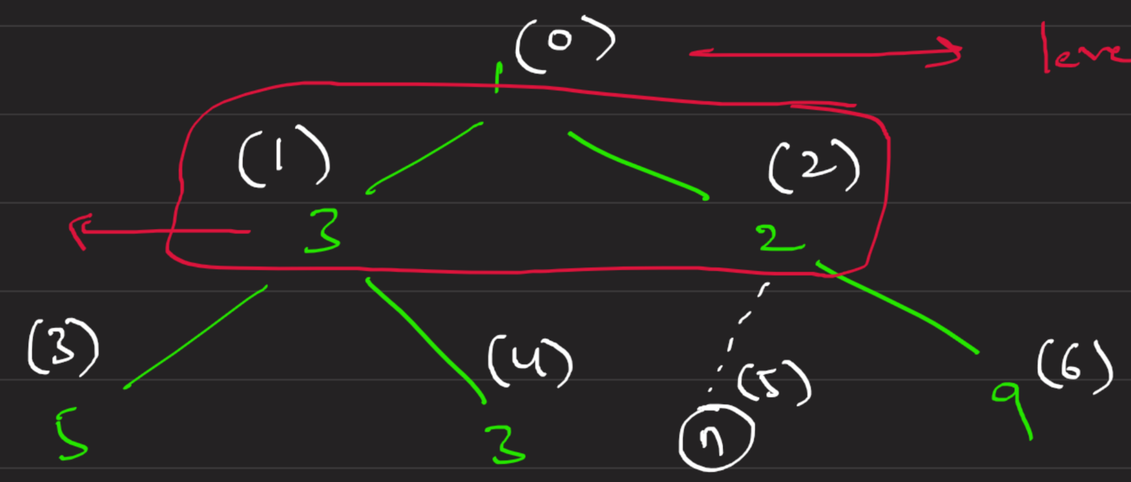
Ans ⇒ Ending node index − Starting node index + 1



If you somehow Convert this problem into array indexing → job is done



Now, Calculating width will be easy...



level 0 → Starting index = Ending index

⇒ 0 - 0 + 1 = 1

level 1

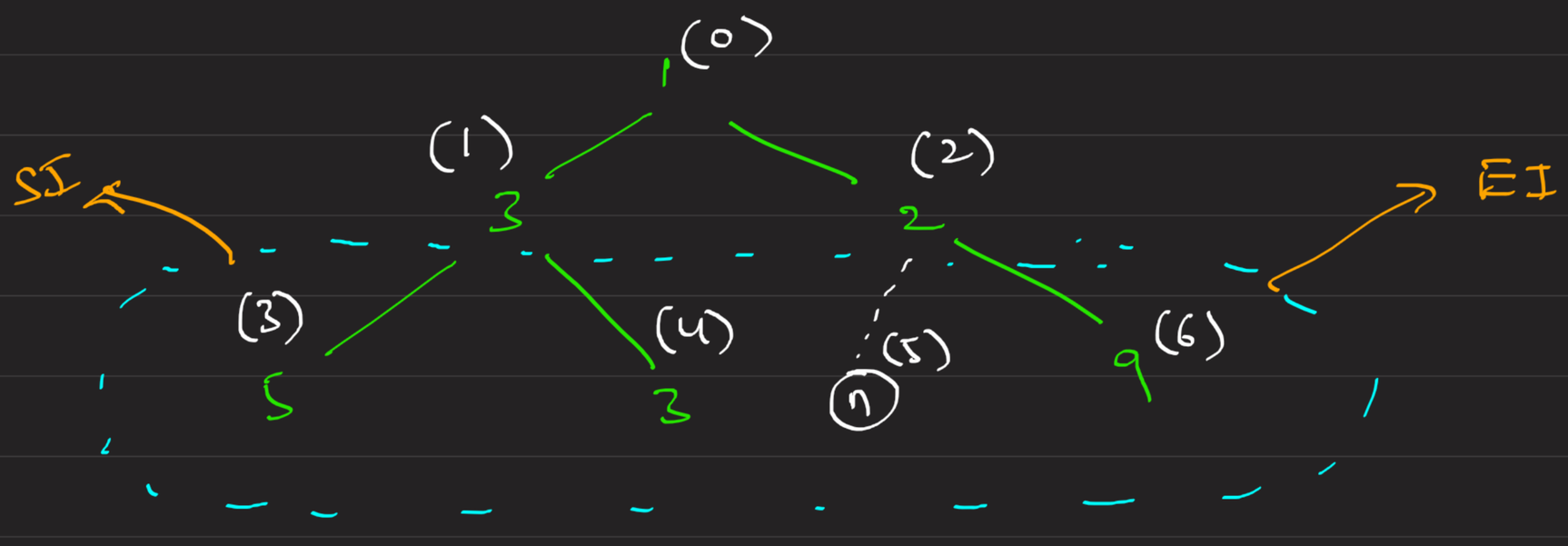
⇒ Starting index = 1

Ending index = 2

∴ Width = EI - SI + 1

= 2 - 1 + 1

= (2)

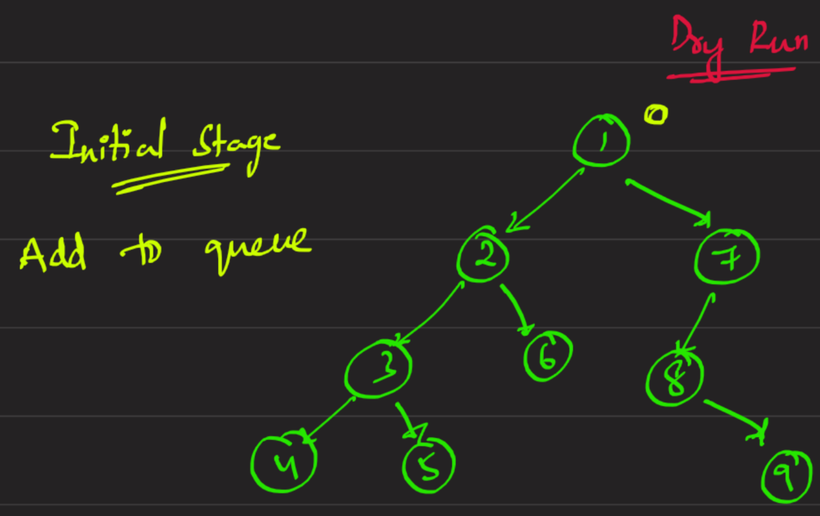


Width = EI - SI + 1 = 6 - 3 + 1 = (4)

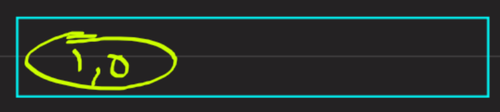
So, while traversing in BFS fashion

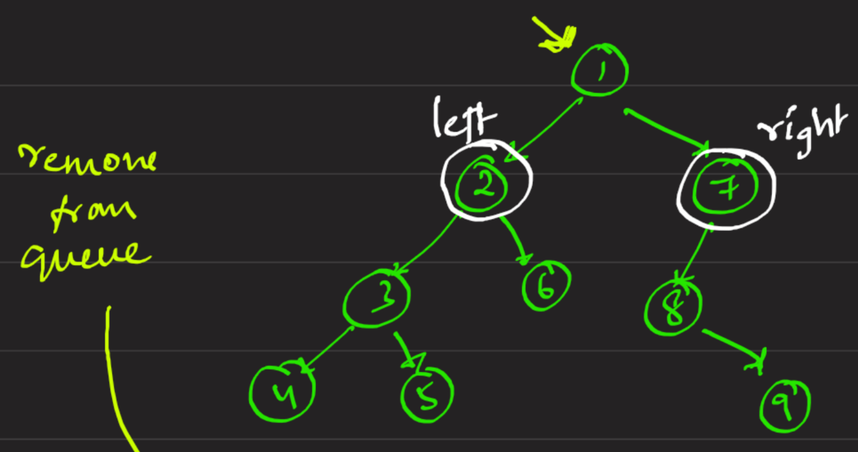
Assigns indexes as well ... the Ques 'ds will be very easy

Dry Run

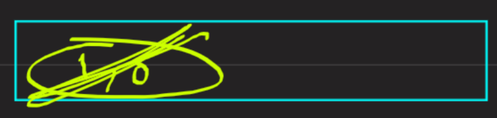


queue





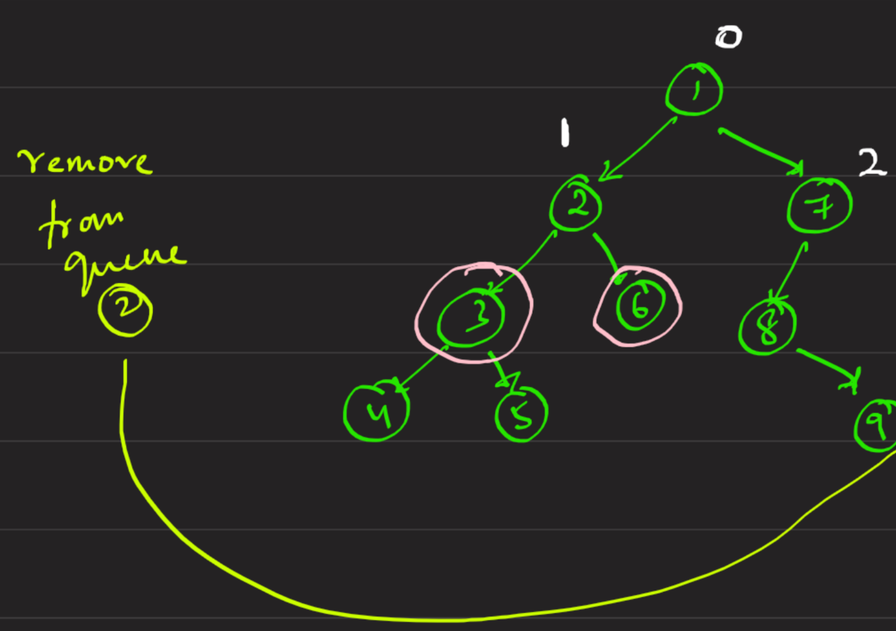
queue



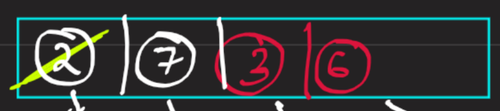
→ then, iterate till is not Empty

* if node (1) has left, add to queue

* if node (1) has right, add to queue



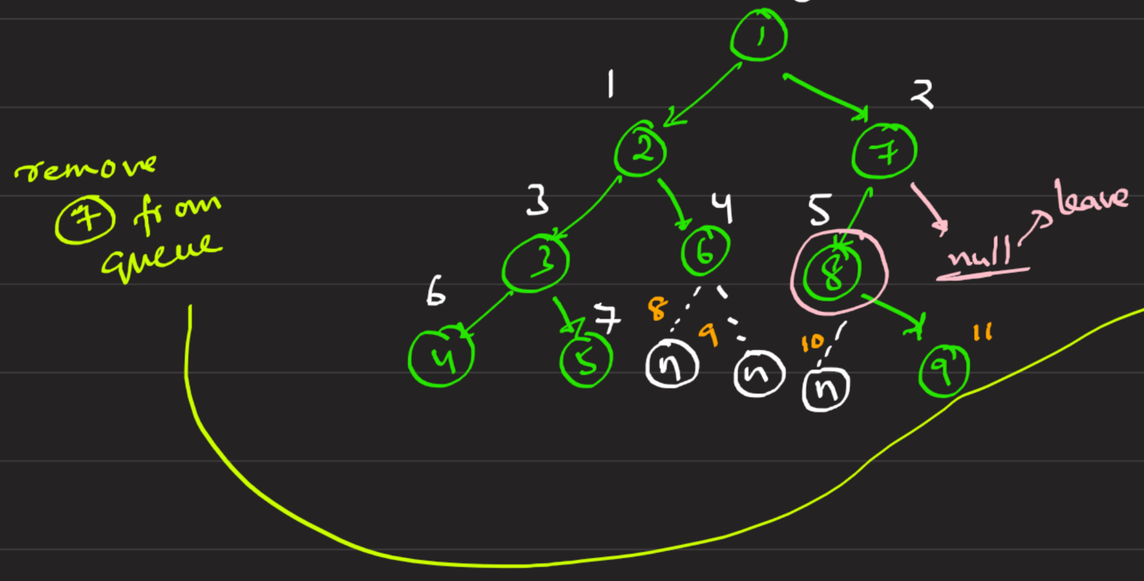
queue



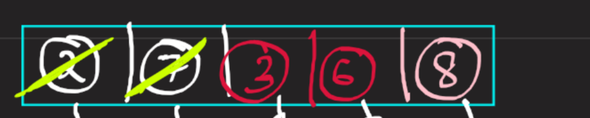
(2,1) (7,2) (3,3) (6,4) → with indexes add it to queue

Add left of (2), if contains

Add right of (2), if contains



queue



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

Add left of (7), if contains

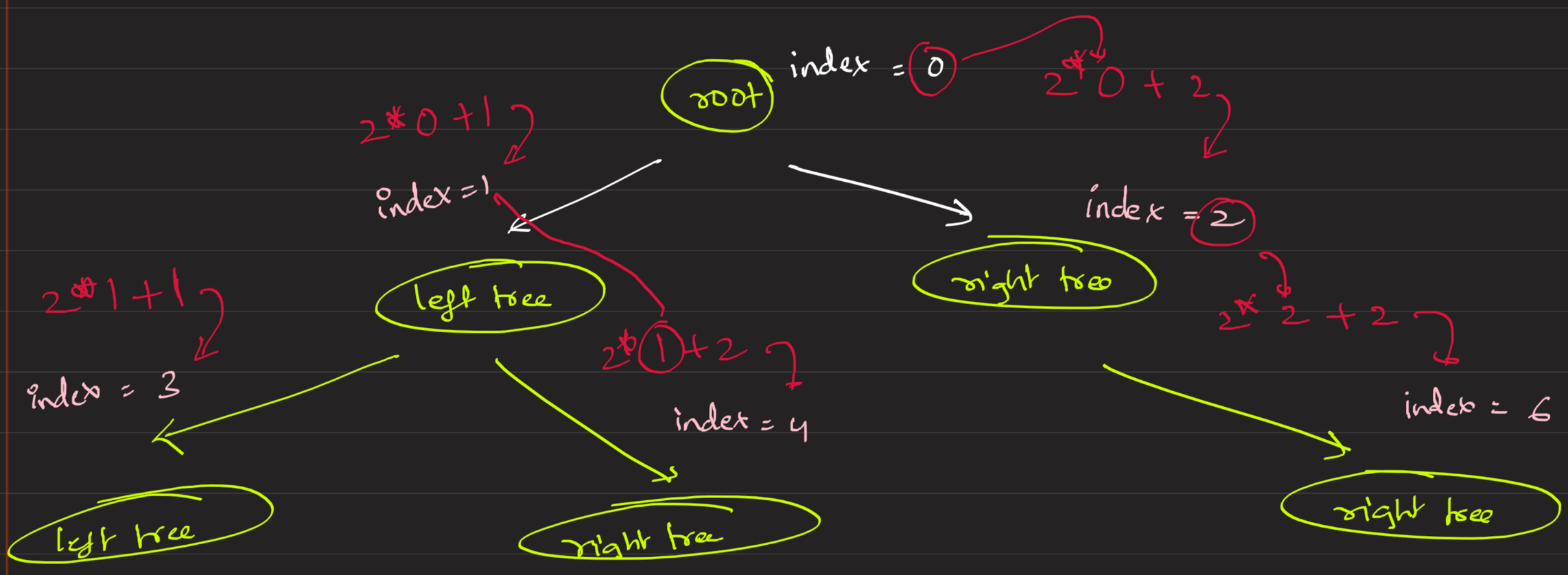
Add right of (7), if contains

∴ The basic idea is

How these indices are generated

if index of root = i

root.left's index = (2*i + 1)



if index of root = i

root.right's index = (2*i + 2)

∴ Do BFS, with index additionally, & Extract max width

in pair, we are having (Node, index)

```java
public int widthOfBinaryTree(TreeNode root) {
    Queue<Pair> q = new LinkedList<>();
    q.add(new Pair(root,0));
    int cnt = 0;
    while(q.size()>0){
        int n = q.size();
        List<Integer> ans = new ArrayList<>();
        for(int p = 0;p<n;p++){
            Pair rem1 = q.remove();
            TreeNode rem = rem1.n;
            int k = rem1.i;
            ans.add(k);
            if(rem.left!=null){
                q.add(new Pair(rem.left,2*k+1));
            }

            if(rem.right!=null){
                q.add(new Pair(rem.right,2*k+2));
            }
        }
        cnt = Math.max(cnt,ans.get(ans.size()-1)-ans.get(0)+1);
    }
    return cnt;
}
```

0th index

2 for loops for level by level

for every level store indices in ans

if in level 2 the, ans[] becomes [2,3,4,5,6,7] ∴ width =

last - first +1

We pick max out of all levels