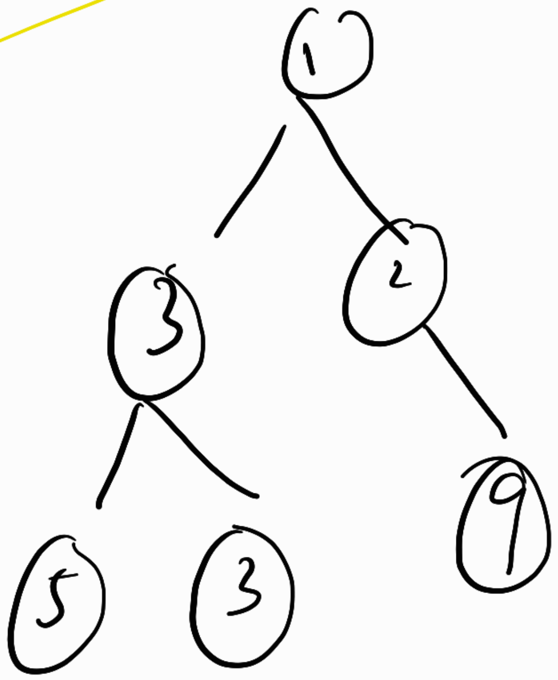
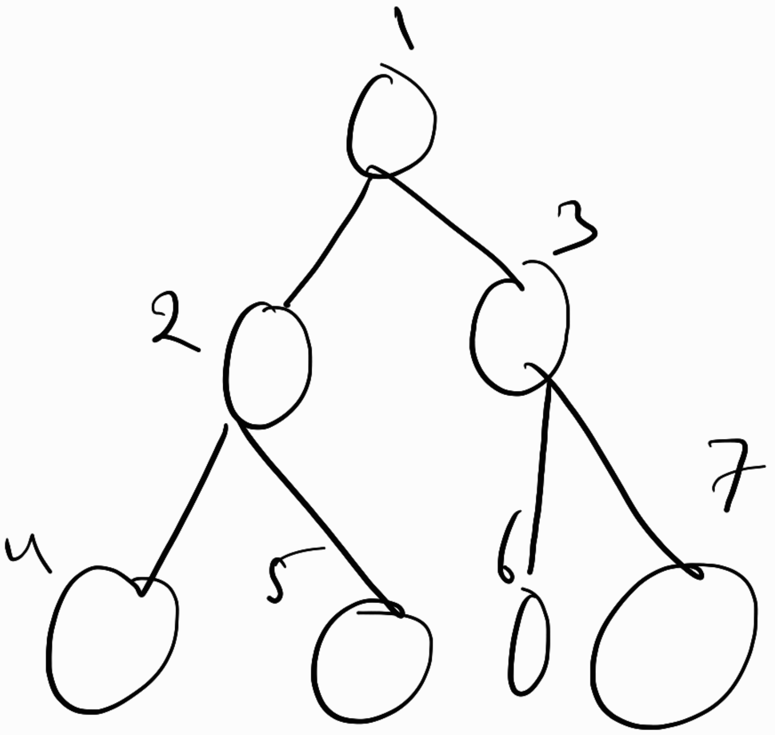
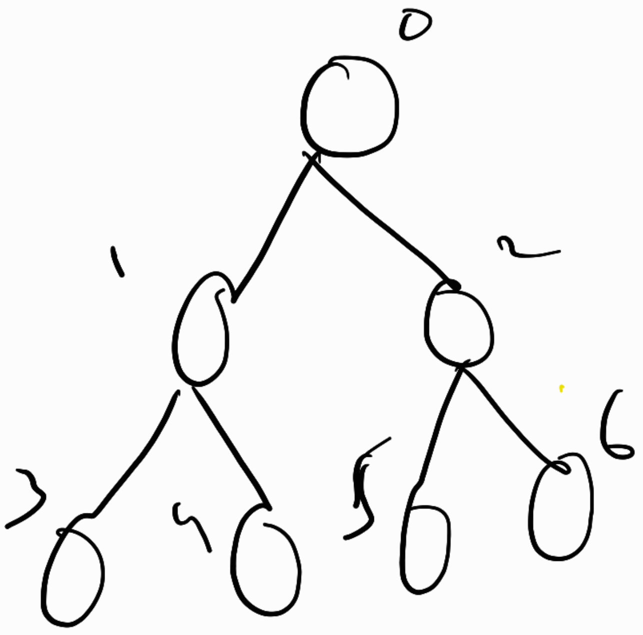
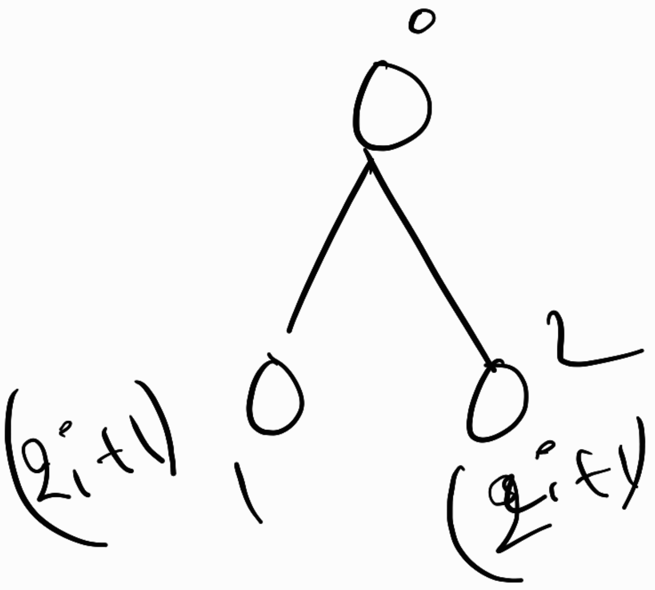
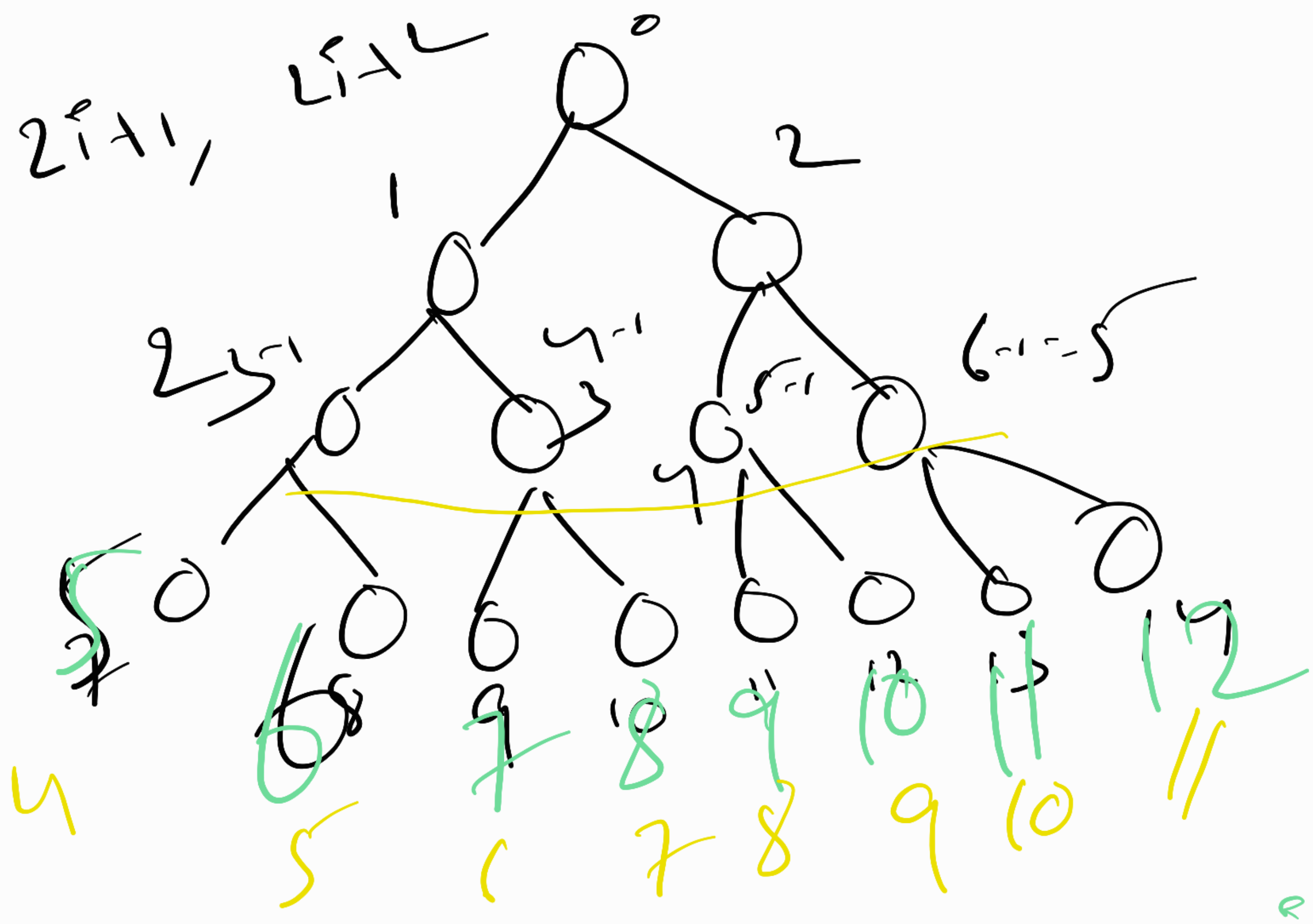
11/19/24 maximum width:





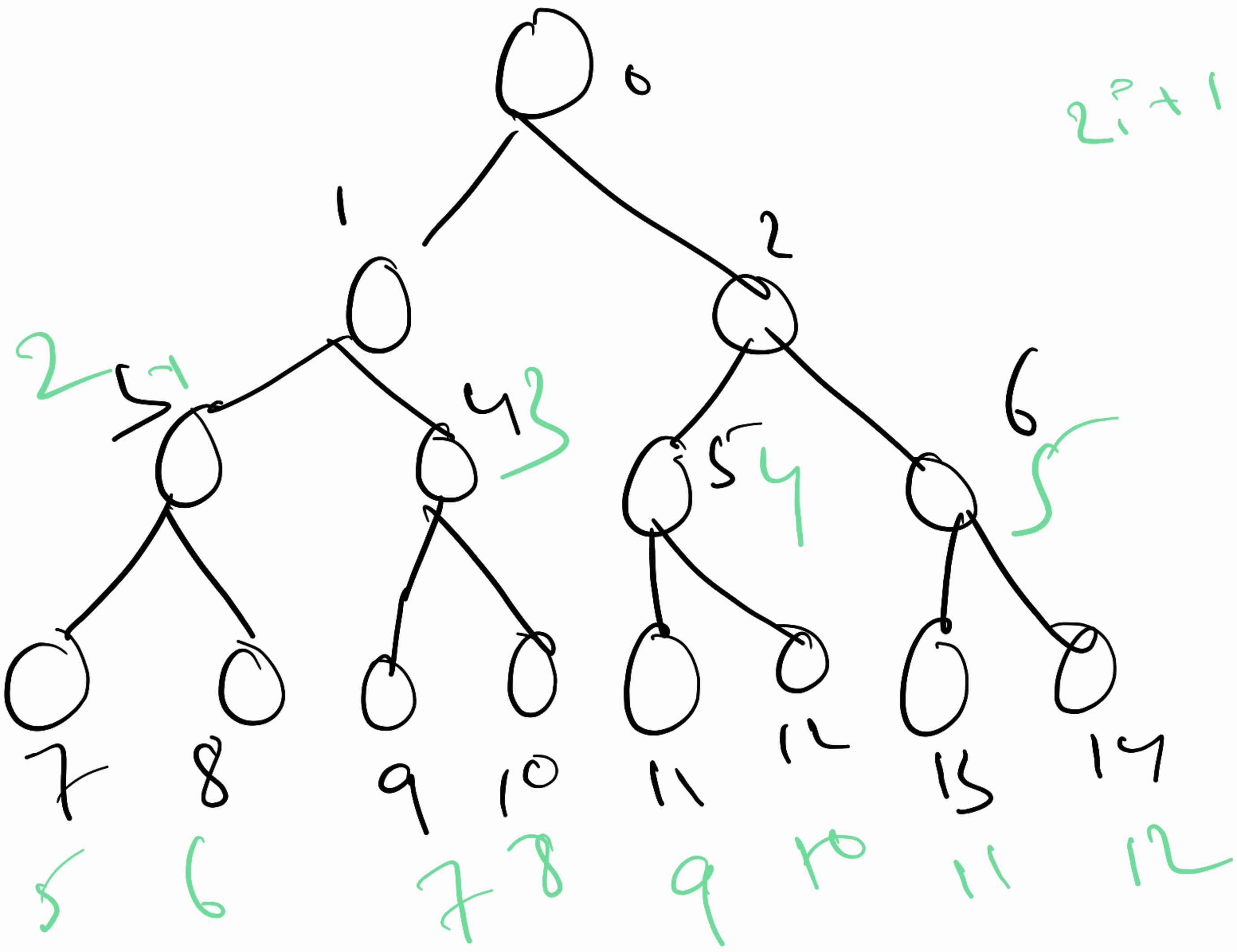


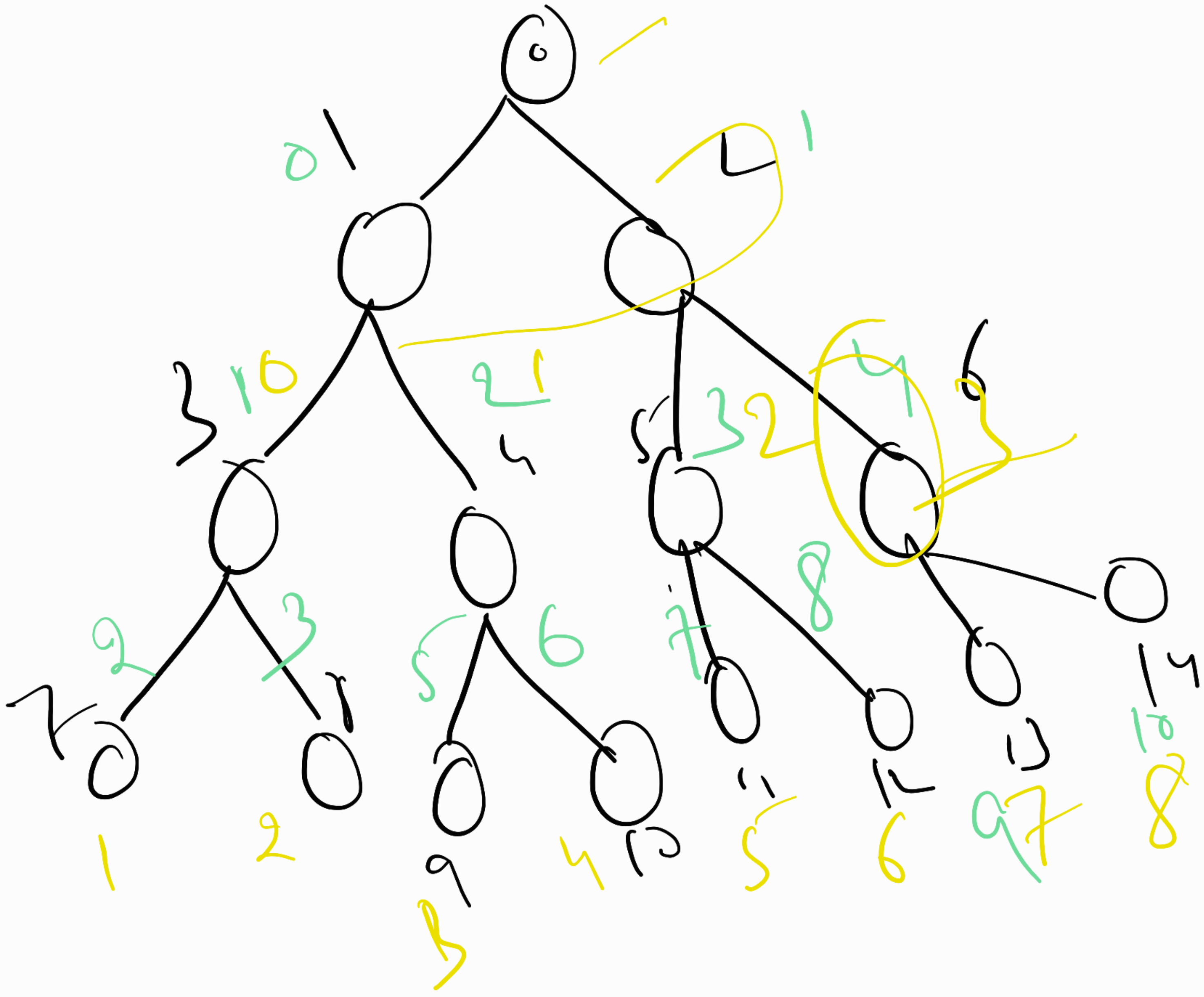


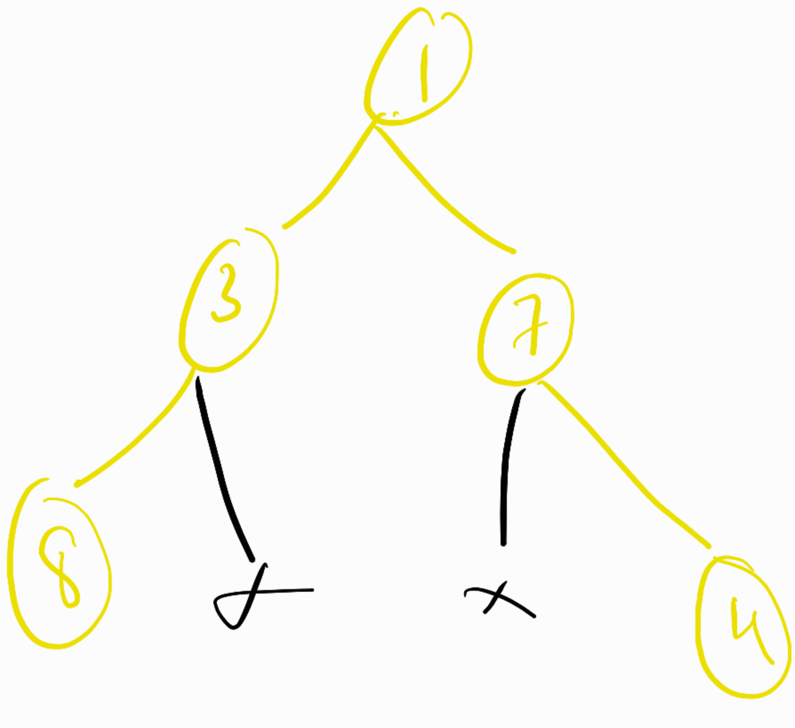

2i+1, 2i+2
0
1
2
2 3-1
4-1 3
5-1 4
6-1=5
7 8 9 10 11 12 13 14
5 6 7 8 9 10 11 12
4 5 6 7 8 9 10 11

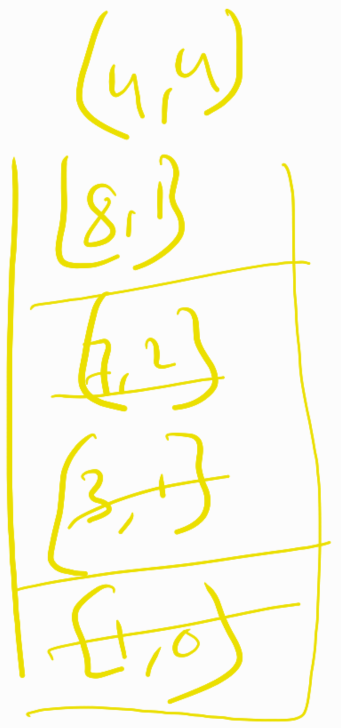
i==0

2i+1


0
1
2
2 3-1
4 3
5 4
6 5
7 8 9 10 11 12 13 14
5 6 7 8 9 10 11 12


0
01
1
2
310
3
21
4
5
32
41
6
2
3
5
6
7
8
7
8
9
10
11
12
13
14
10
1
2
3
4
5
6
97
8


1
3
7
8
4


(4,4)
[8,1]
~~[7,2]~~
~~[3,1]~~
~~[1,0]~~

width = last guy index - firstGuy Index +1)

width

$\boxed{x}$

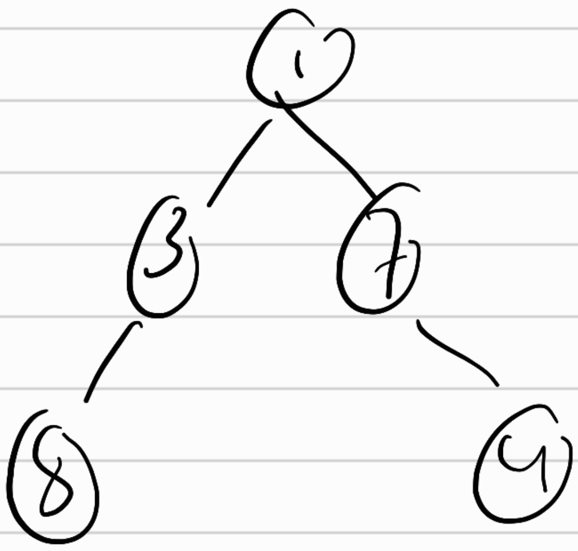
~~2~~ 4

{8, 1} {4, 4}

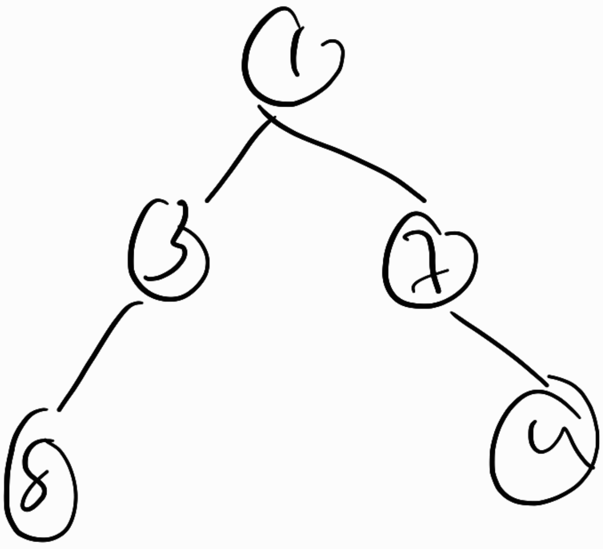
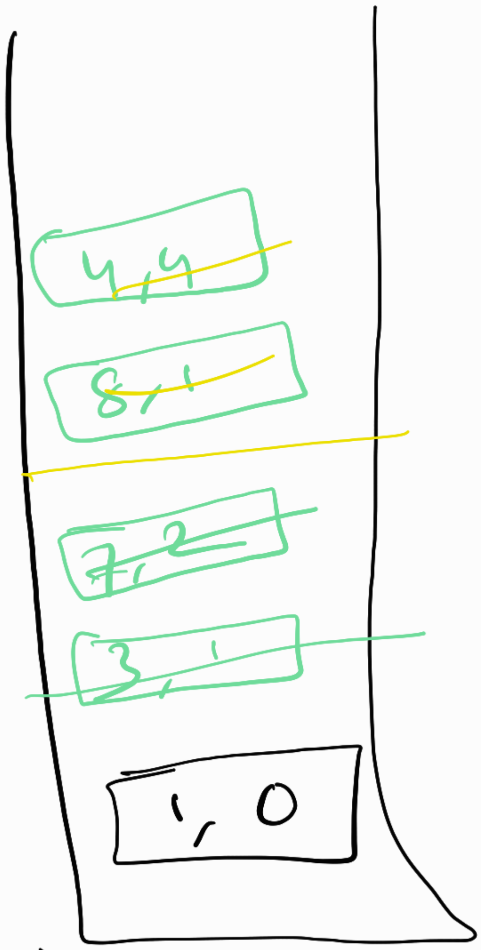
Queue.

1) offer (push)

2) poll (pop)



```
Class Pair{
      TreeNode   node;
        int      num;
    Pair (TreeNode node, int num){
          this.node = node;
          this.num = num;
      }
}
class Solution{
                                                    root
   public int  widthOfBinaryTree(TreeN r{
   if(root == null) return 0;
     Int  ans = 0;
 Queue < Pair> q = new LinkedList<>();
     q.push(new Pair (root, 0))
       offer
```

```
while(!q.isEmpty()){
int size = q.size();
           2
int mmin = q.peek().num;
int first =0, last =0;
                            2
for(int i=0; i < size; i++){
    int cur_id = q.peek().num - mmin;
    TreeNode node = q.peek().node;
     q.pop(); poll
if(i==0) first = cur_id;
if (i==size-1) last= cur_id;
If (node.left != null)
  q.offer(new Pair(node.left, 2*(curr_id + 1)))
if (node.right != null)
 q.offer(new Pair(node.right,
                 2*(curr_id + 2)))
}(break)
   ans = Math.max(ans, last - first + 1)
}
ret ans;
};
```

4

2 1   4 1 1