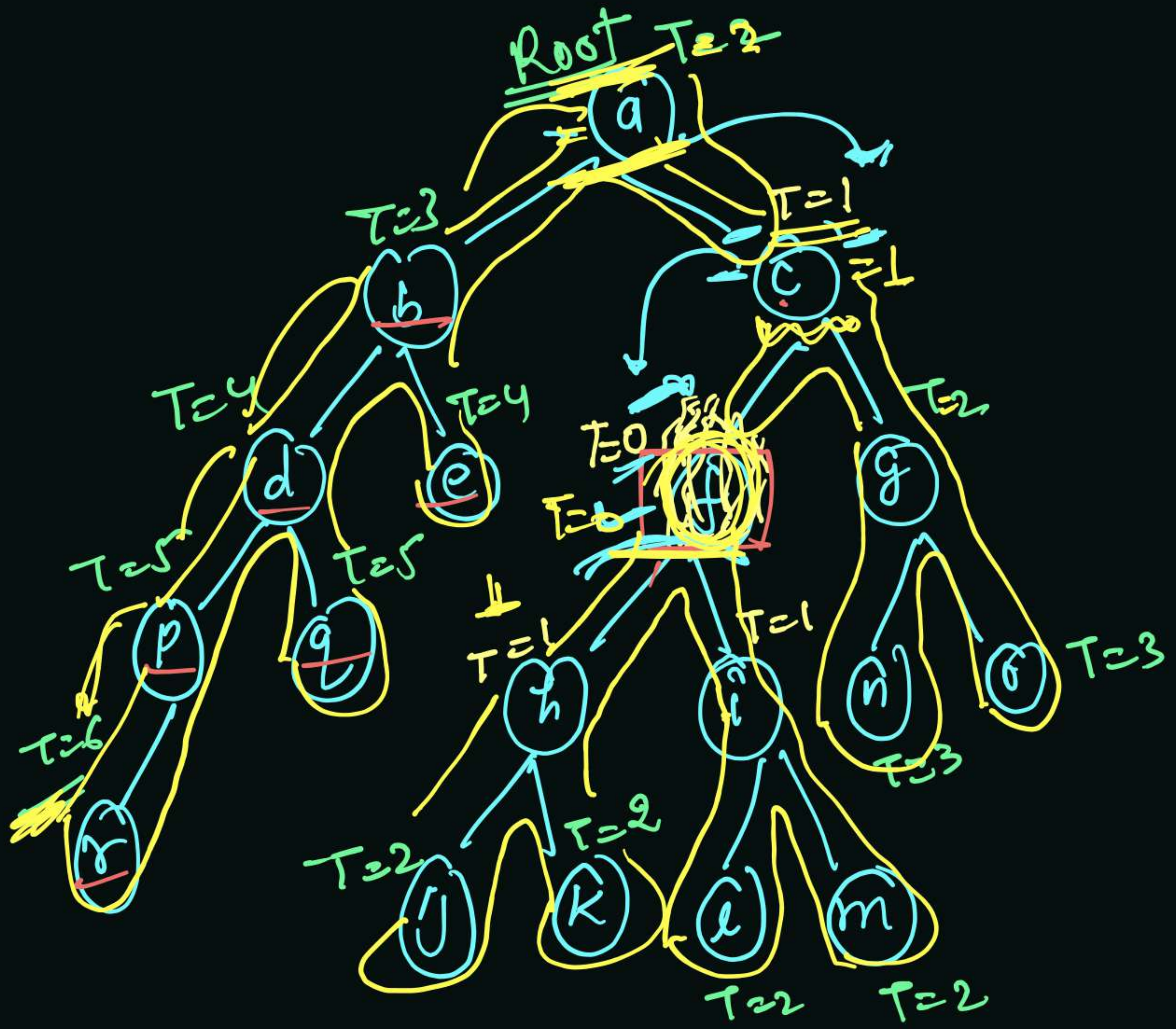
Burning Tree

Saturday, 28 August 2021 7:16 PM

Min time Required to burn all Node of a tree

Behaviour of fire → on every Second adjacent node will catch fire.

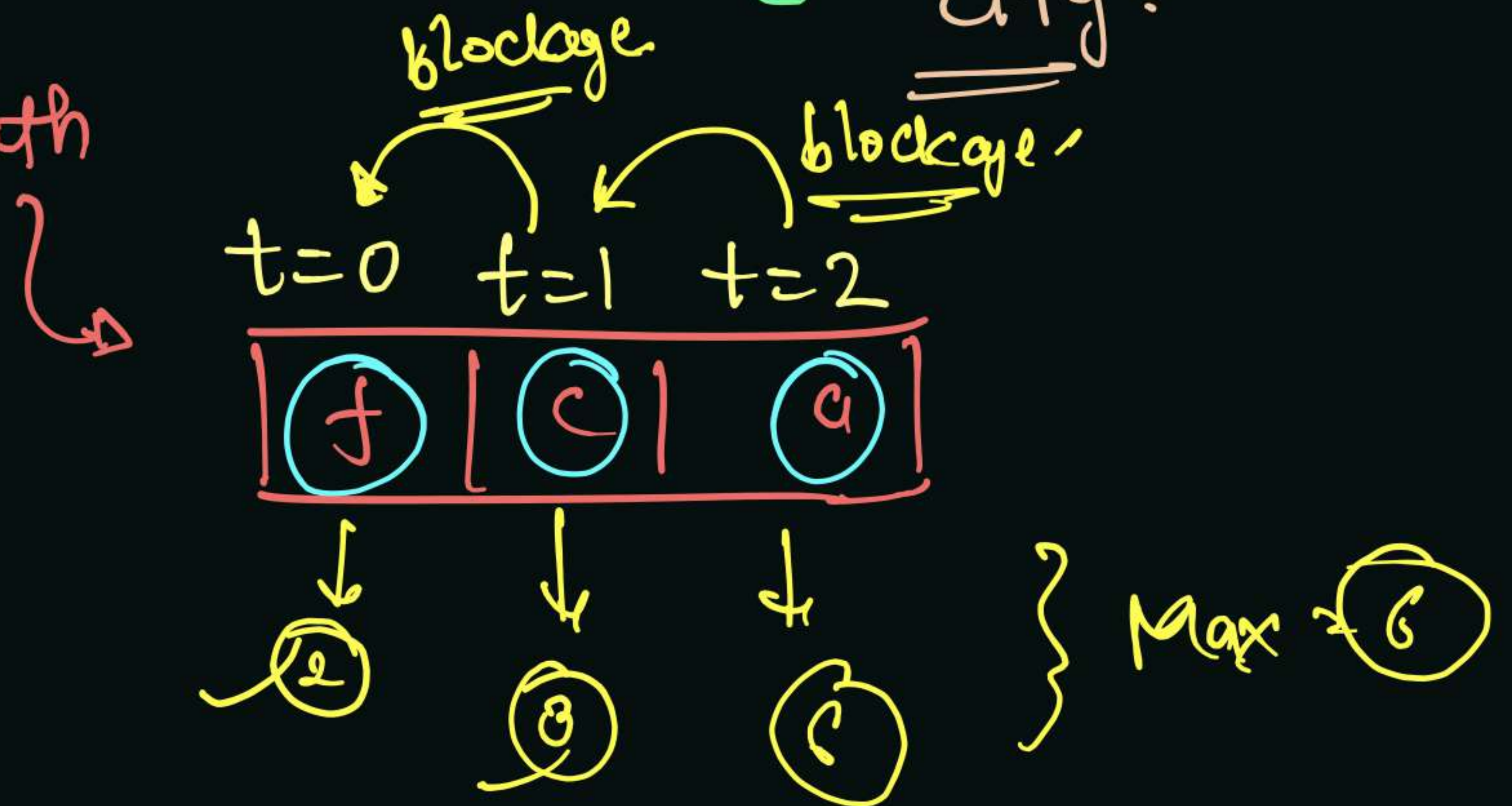
Initial node with fire ⟶ (f)



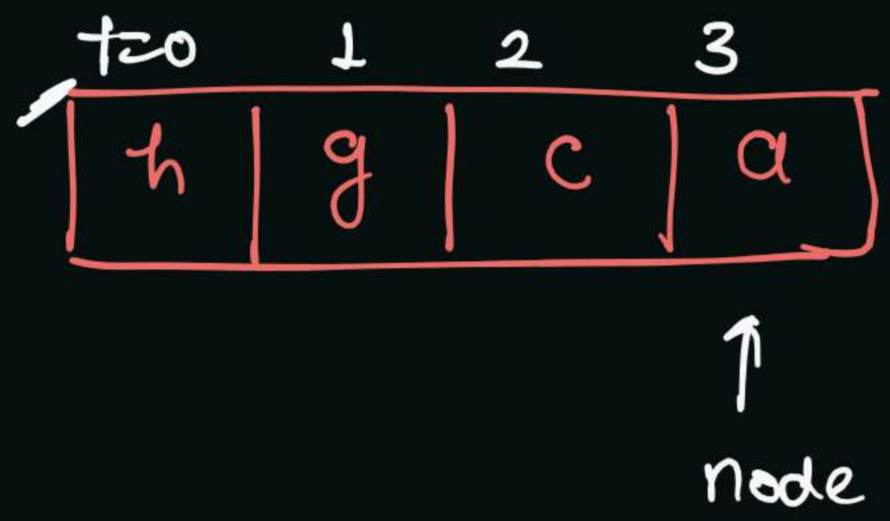
T=6 Seconds Required.

Graph. (Back Edge)
- → Spread of infection
- → fire in the city.

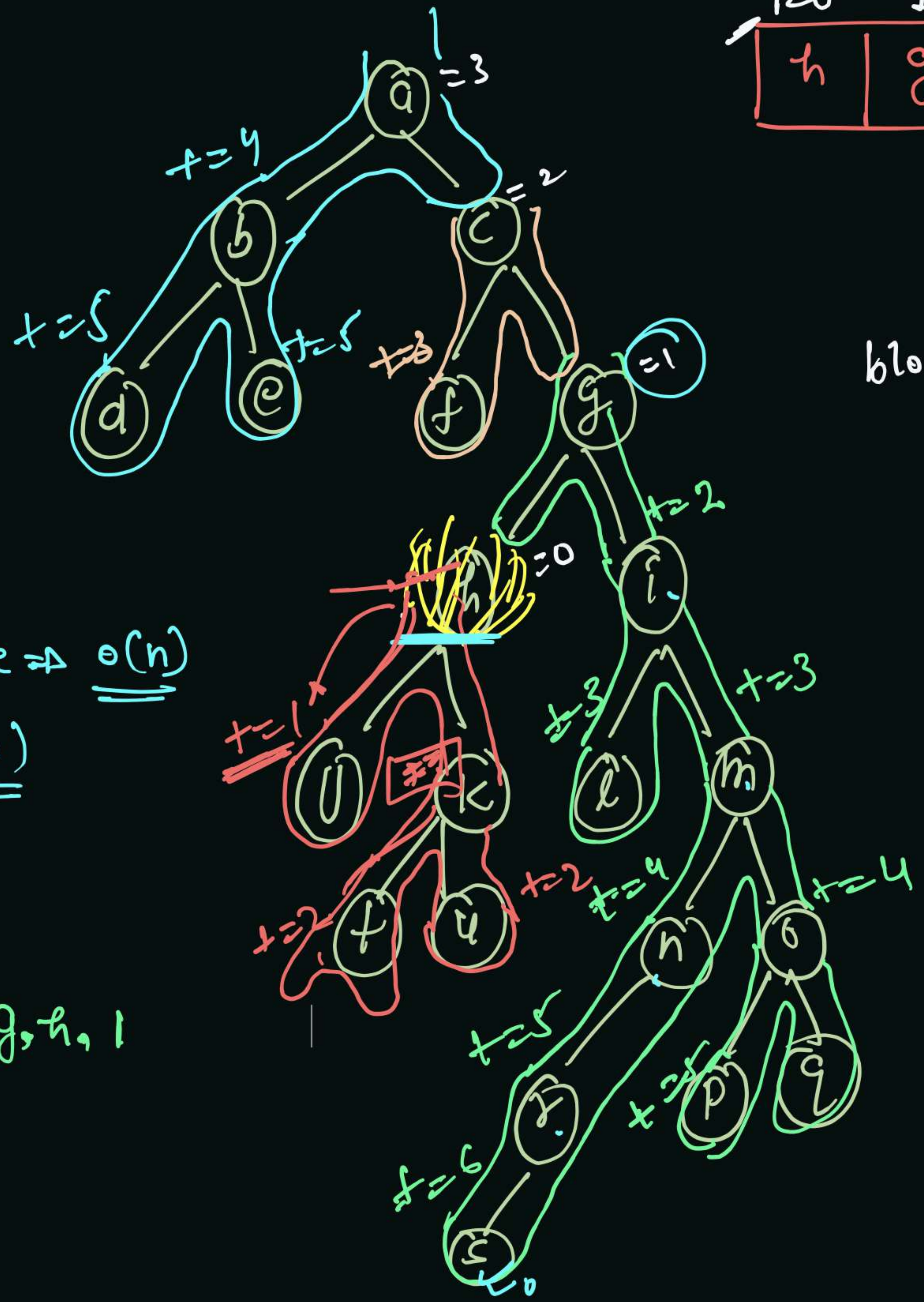
node to Root path



static int maxtime = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6 maxTime = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6



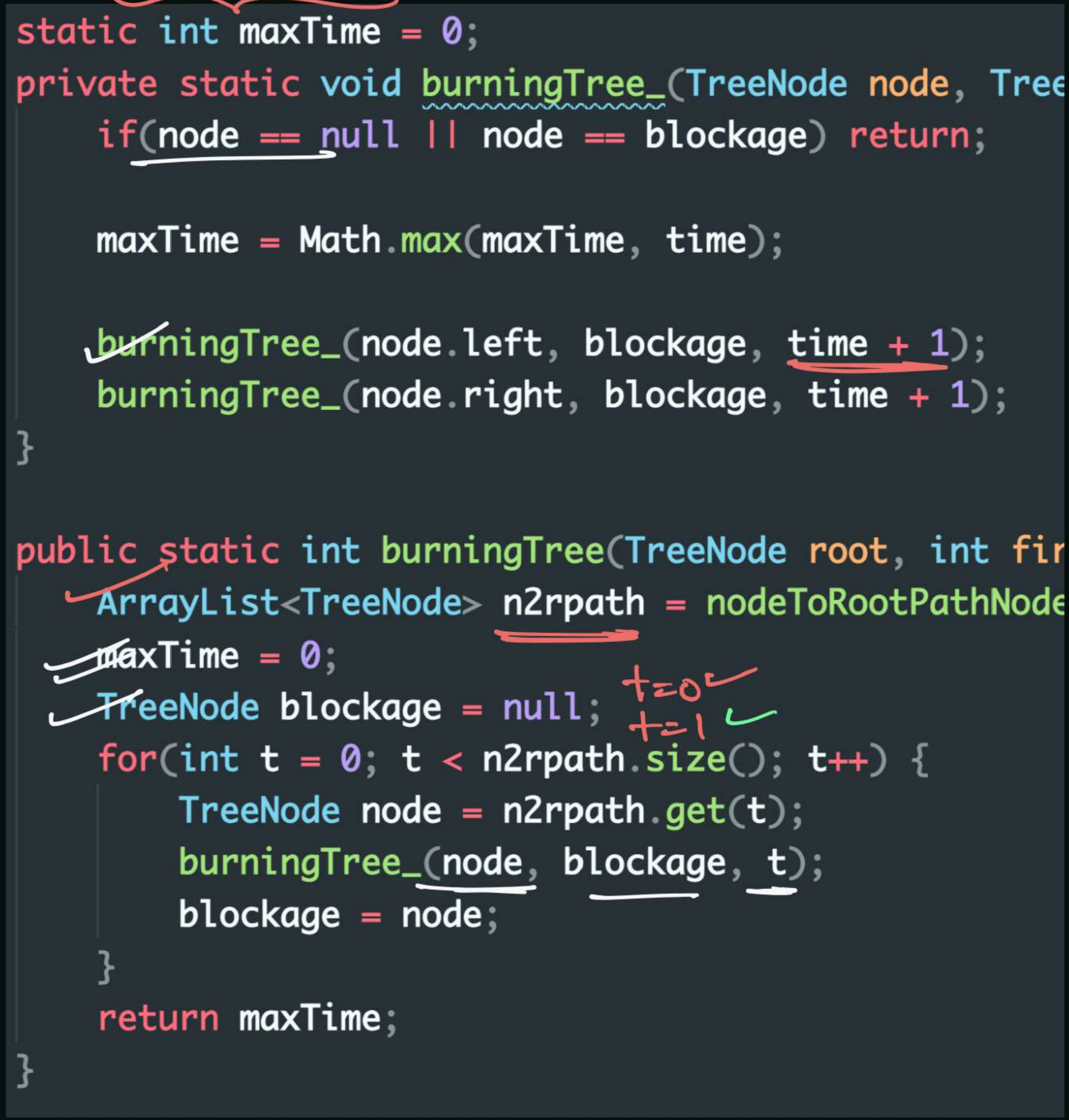
blockage = ~~null~~
~~h~~
~~g~~
C



time => O(n)

O(h)

g, h, l

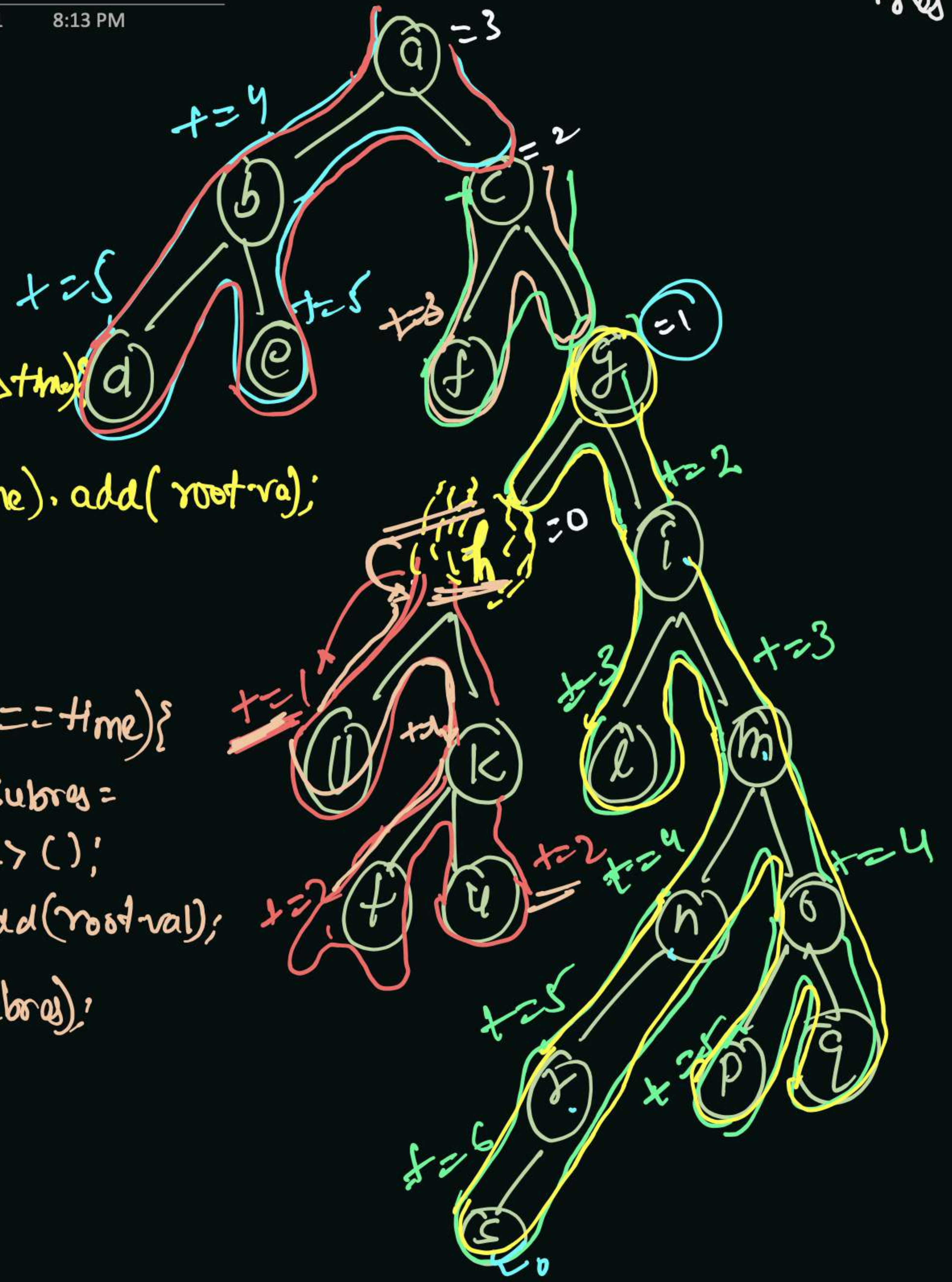


```
static int maxTime = 0;
private static void burningTree_(TreeNode node, Tree
    if(node == null || node == blockage) return;

    maxTime = Math.max(maxTime, time);

    burningTree_(node.left, blockage, time + 1);
    burningTree_(node.right, blockage, time + 1);
}

public static int burningTree(TreeNode root, int fir
    ArrayList<TreeNode> n2rpath = nodeToRootPathNode
    maxTime = 0;
    TreeNode blockage = null;
    for(int t = 0; t < n2rpath.size(); t++) {
        TreeNode node = n2rpath.get(t);
        burningTree_(node, blockage, t);
        blockage = node;
    }
    return maxTime;
}
```

t=0 ✓
t=1 ✓

Burning Tree 2

Saturday, 28 August 2021 8:13 PM



```
if(res.size()>time){
    res.get(time).add(root.val);
} else
if(res.size()==time){
    AL<Int> subres =
        new AL<>();
    Subres.add(root.val);
    res.add(subres);
}
```

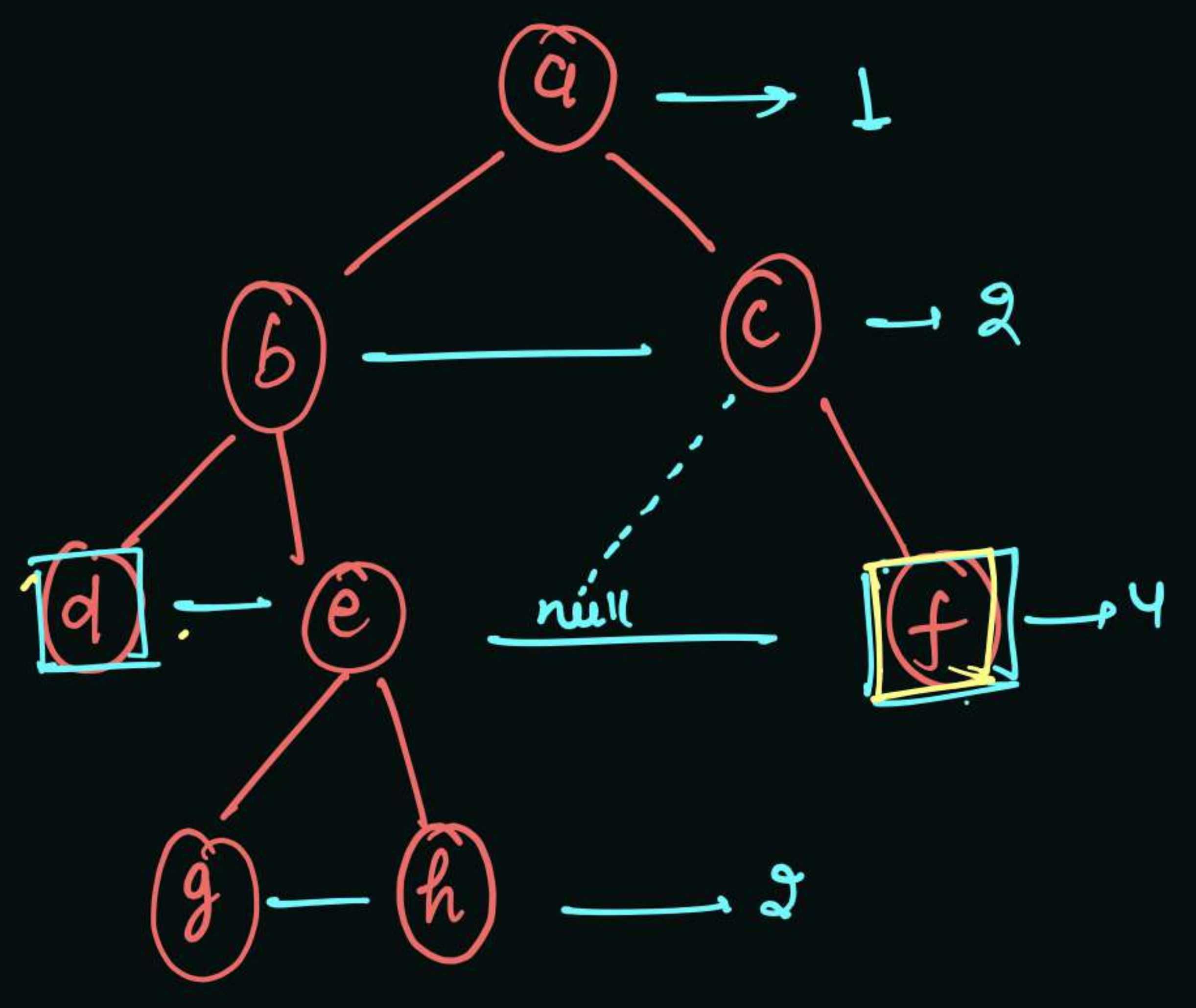
res → ArrayList<ArrayList<Integer>> res =

res → [
0 → [h],
1 → [j, k, g],
2 → [t, u, i, c],
3 → [l, m, f, a],
4 → [n, o, b],
5 → [r, p, q, d, e],
6 → [s]
]

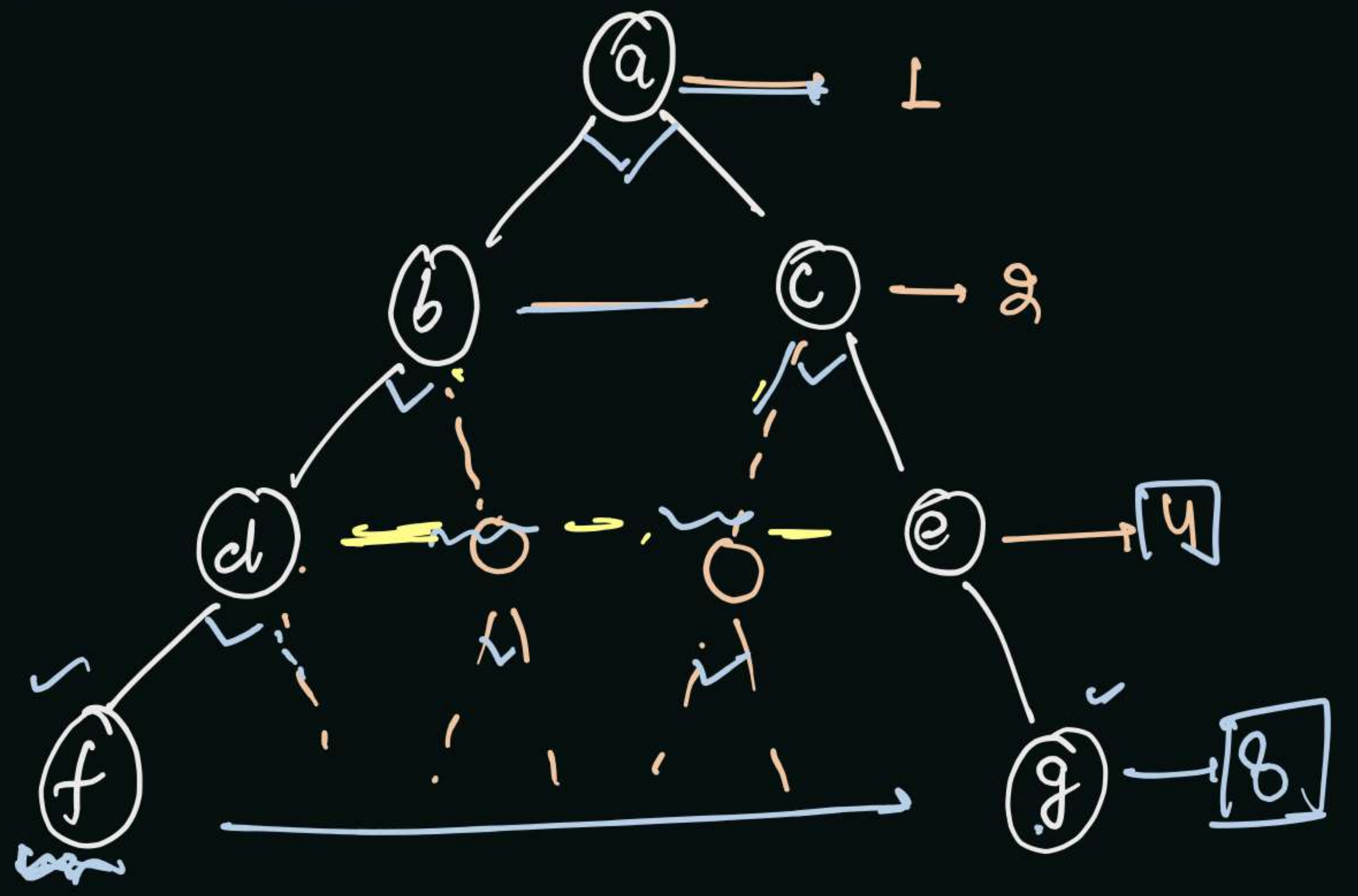
Maximum Width of Binary Tree

Saturday, 28 August 2021 8:49 PM

distance on the basis of node. Not on Edge.

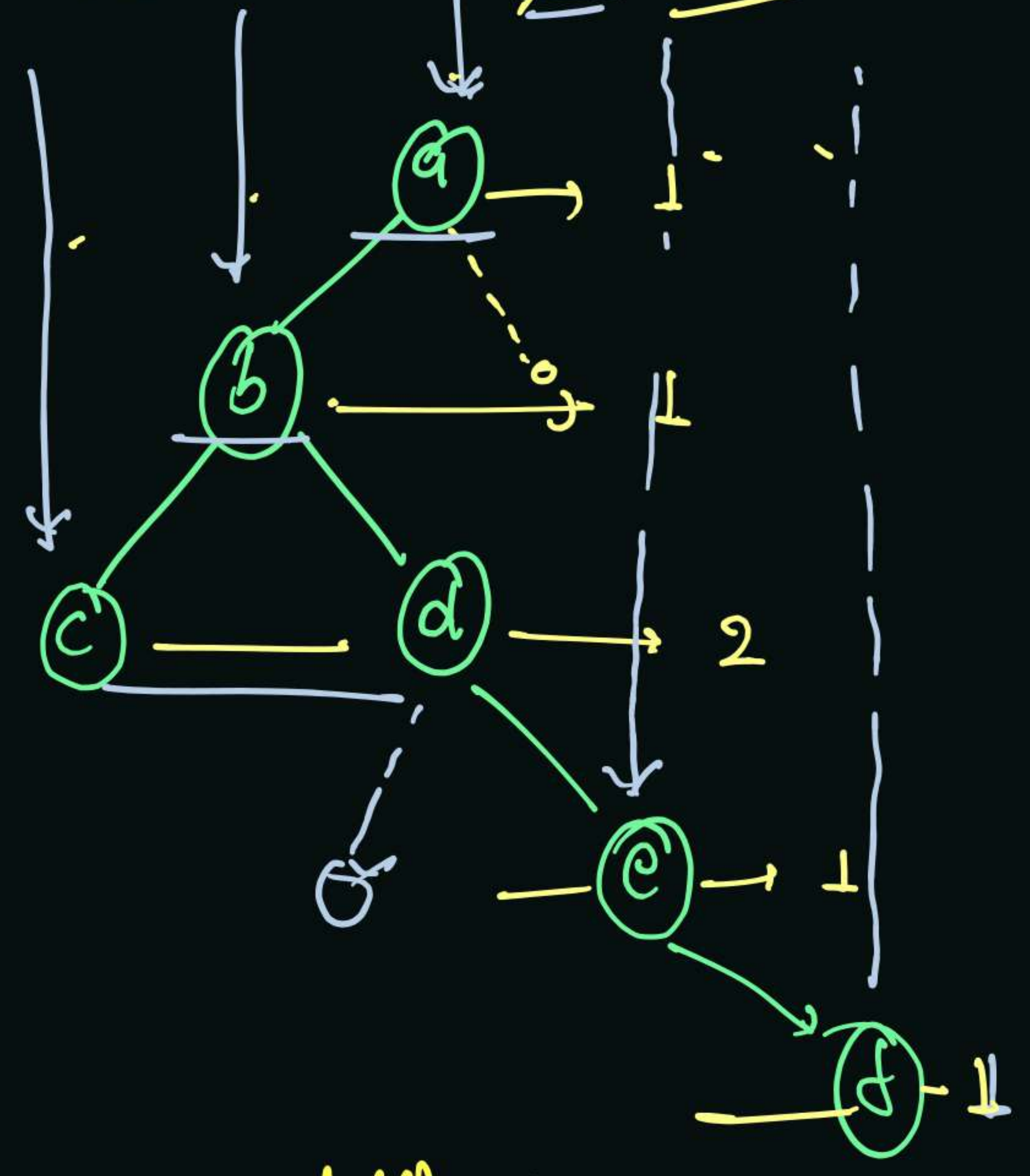


Max width = 4.



Max width = 6

Shadow width = ~~3~~ 4



max width = 2

NOTE: there is a diff b/w Shadow width of tree and max width of binary Tree.

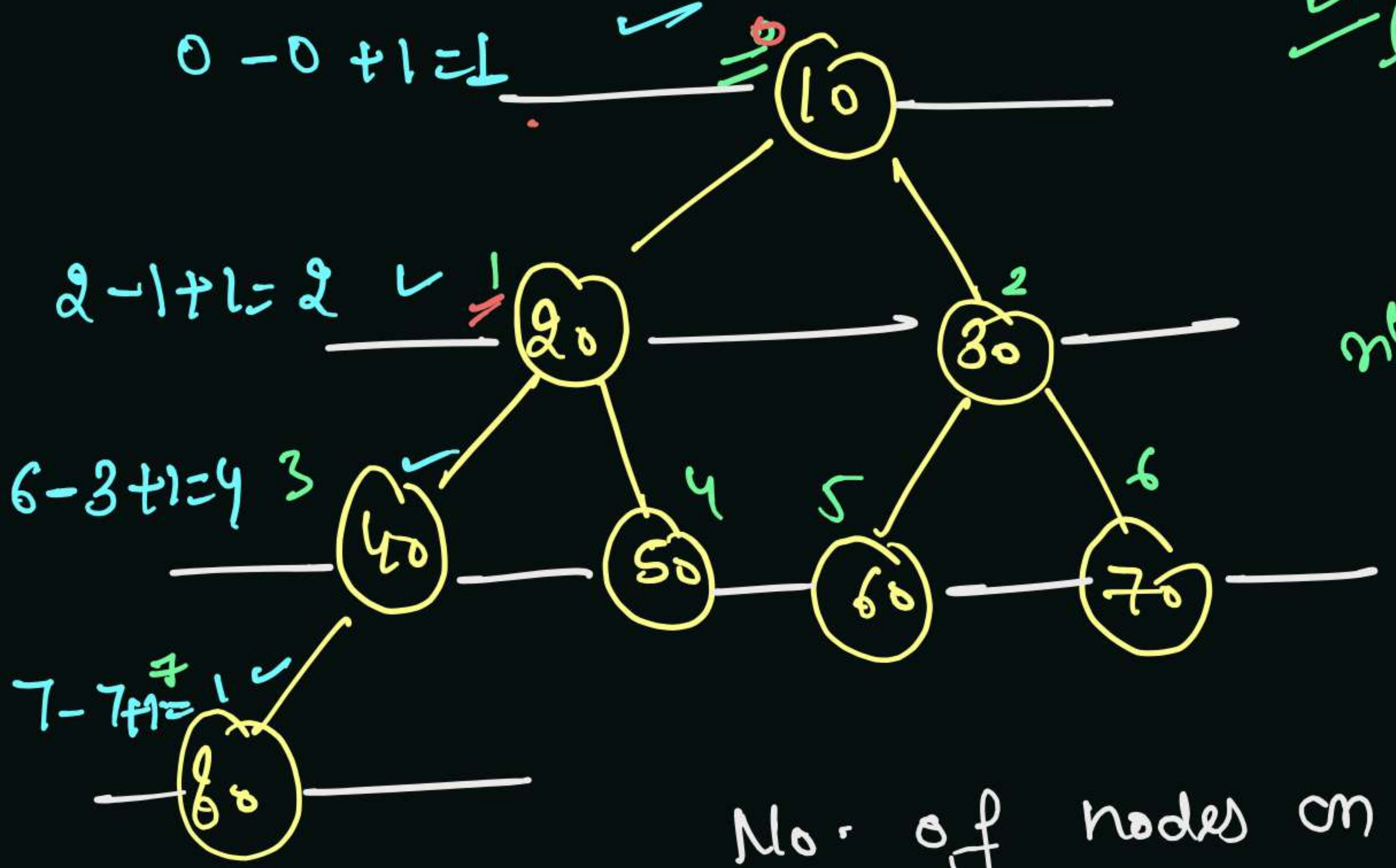
Priority Queue → creation →

level=0 level=1 level=2 level=3

10 20 30 40 50 60 70 80

0 1 2 3 4 5 6 7

this logic hold middle null element

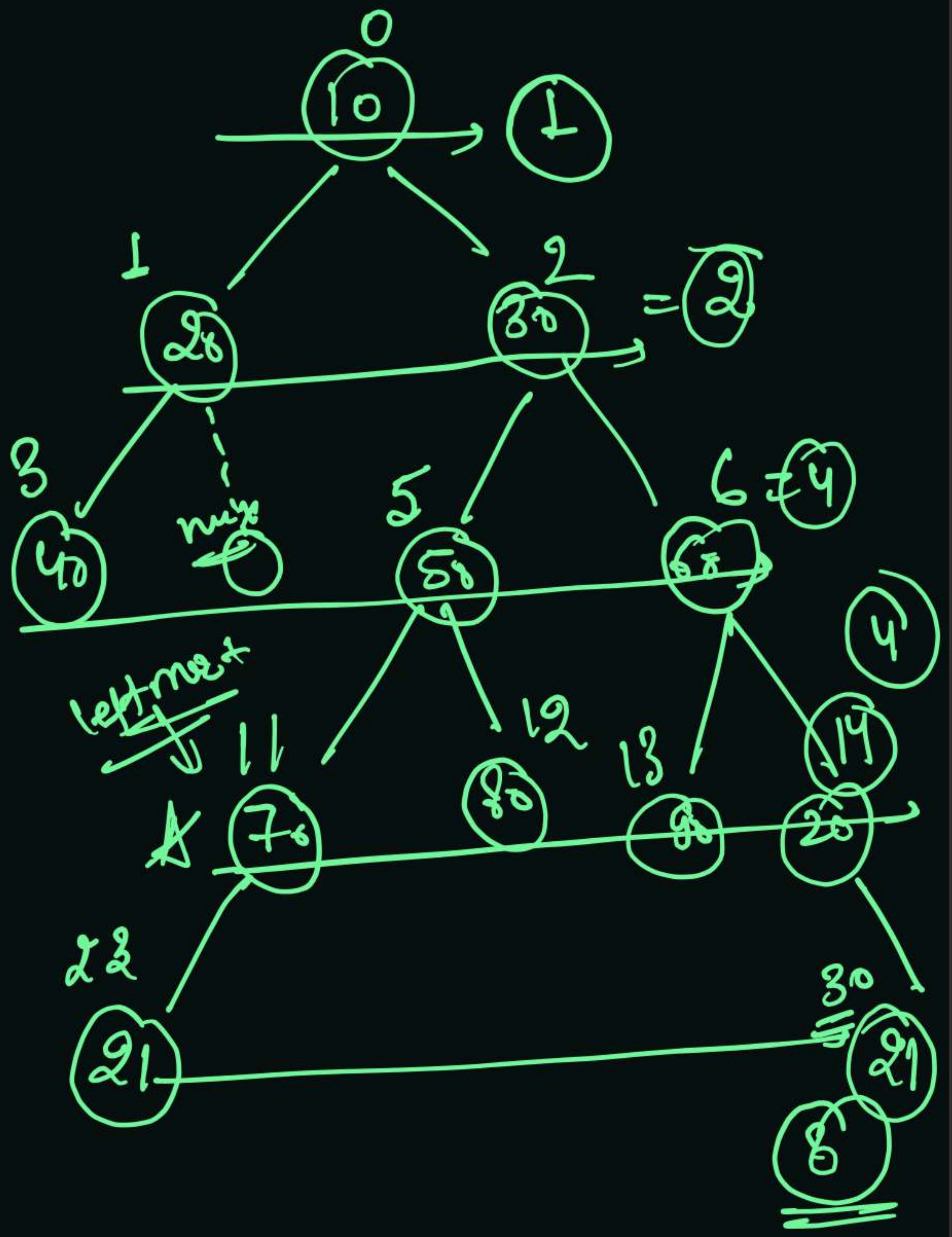


left child index = 2 * parent index + 1

right child index = 2 * parent index + 2

No. of nodes on Every level

= rightmost node index − leftmost node index + 1

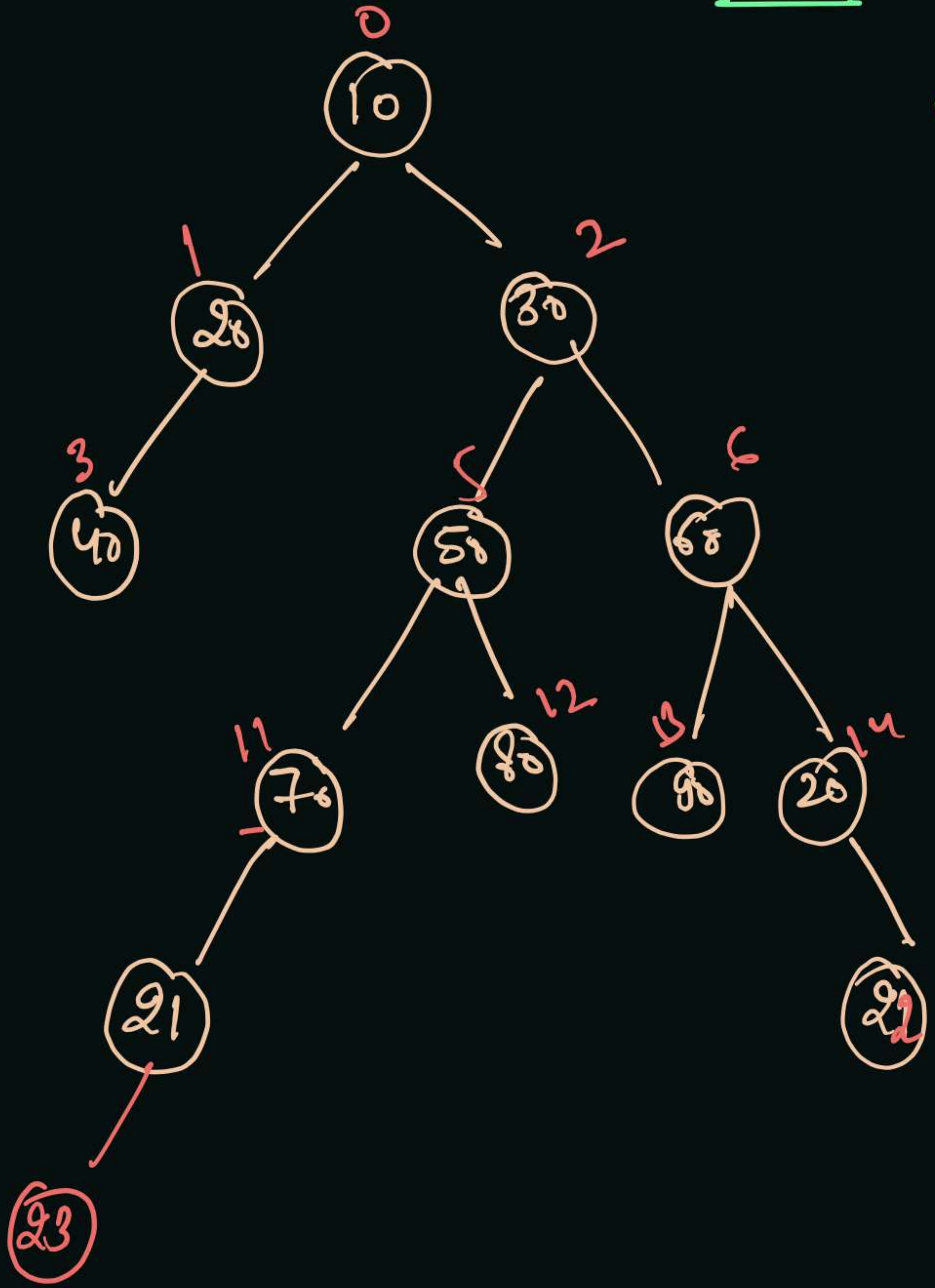


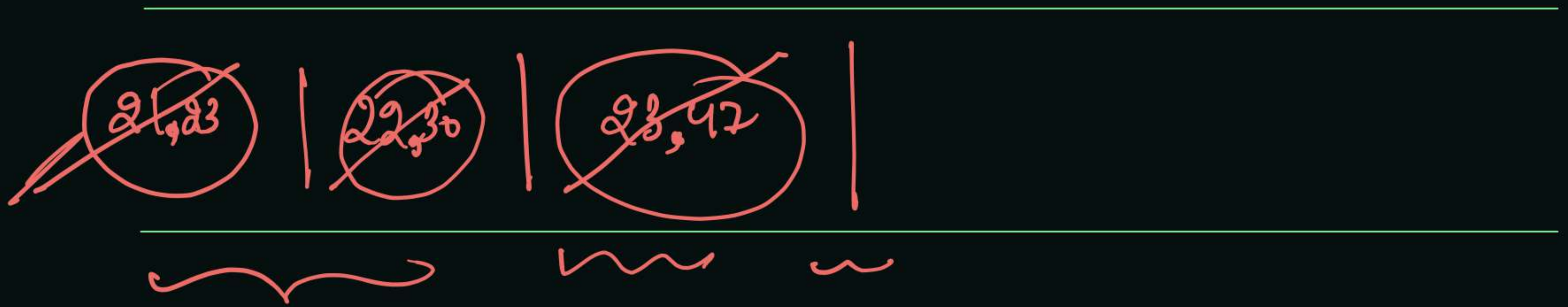
rm − lm + 1 = width on Every level

max width = $\boxed{8}$

BFS → wrapper class → node + Index

queue< wrapper class> qu.





leftmost Index = ~~0~~ ~~1~~ ~~3~~ ~~11~~ ~~23~~ 47

rightmost Index = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~5~~ ~~6~~ ~~11~~ ~~12~~ ~~13~~ ~~28~~ ~~30~~ 47

width = rm - lm+1 = ~~1~~ ~~2~~ ~~4~~ ~~4~~ ~~8~~ 1

max width = ~~0~~ ~~1~~ ~~2~~ ~~4~~ 8 final Result

Convert BST into DLL

Saturday, 28 August 2021 10:00 PM

1. convert BST into Sorted DLL using DFS Method — ~~Inorder in array~~
2. H.W — convert BST into Sorted DLL using STACK
3. H.W — convert BST into Sorted DLL using Morris Traversal (O(1) Space)

Iterative Inorder → 2

Inorder Traversal in Morris → 3

Inplace →

O/p → null ← 5 ⇄ 10 ⇄ 12 ⇄ 20 ⇄ 22 ⇄ 25 ⇄ 30 ⇄ 36 ⇄ 38 ⇄ 40 ⇄ 48 → null

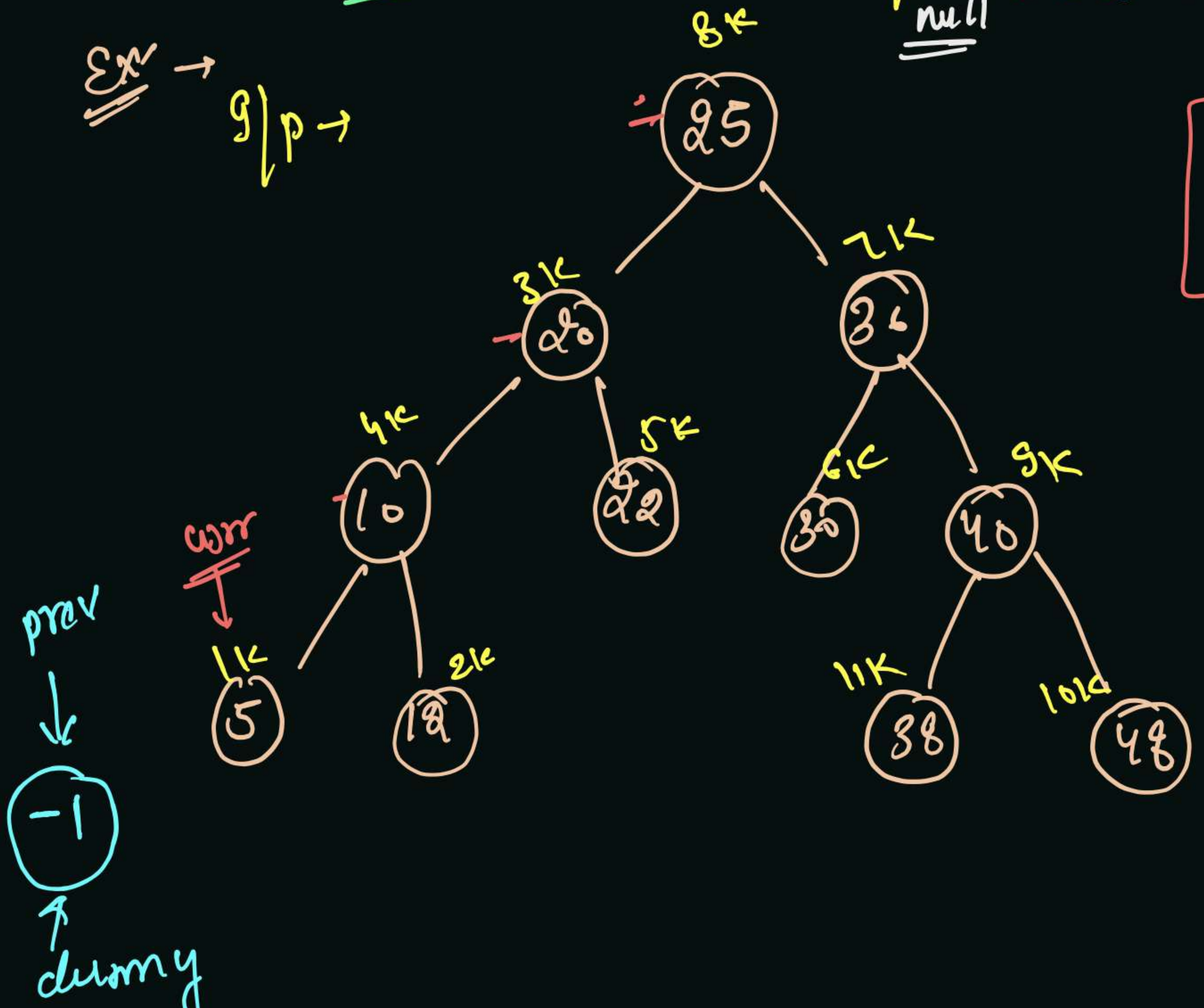
(1K, 4K, 2K, 8K, 5K, 8K, 6K, 7K, 11K, 9K, 10K)

in TreeNode class
- prev ptr = left
- next ptr = Right

Sorted DLL → circular DLL

Ex →

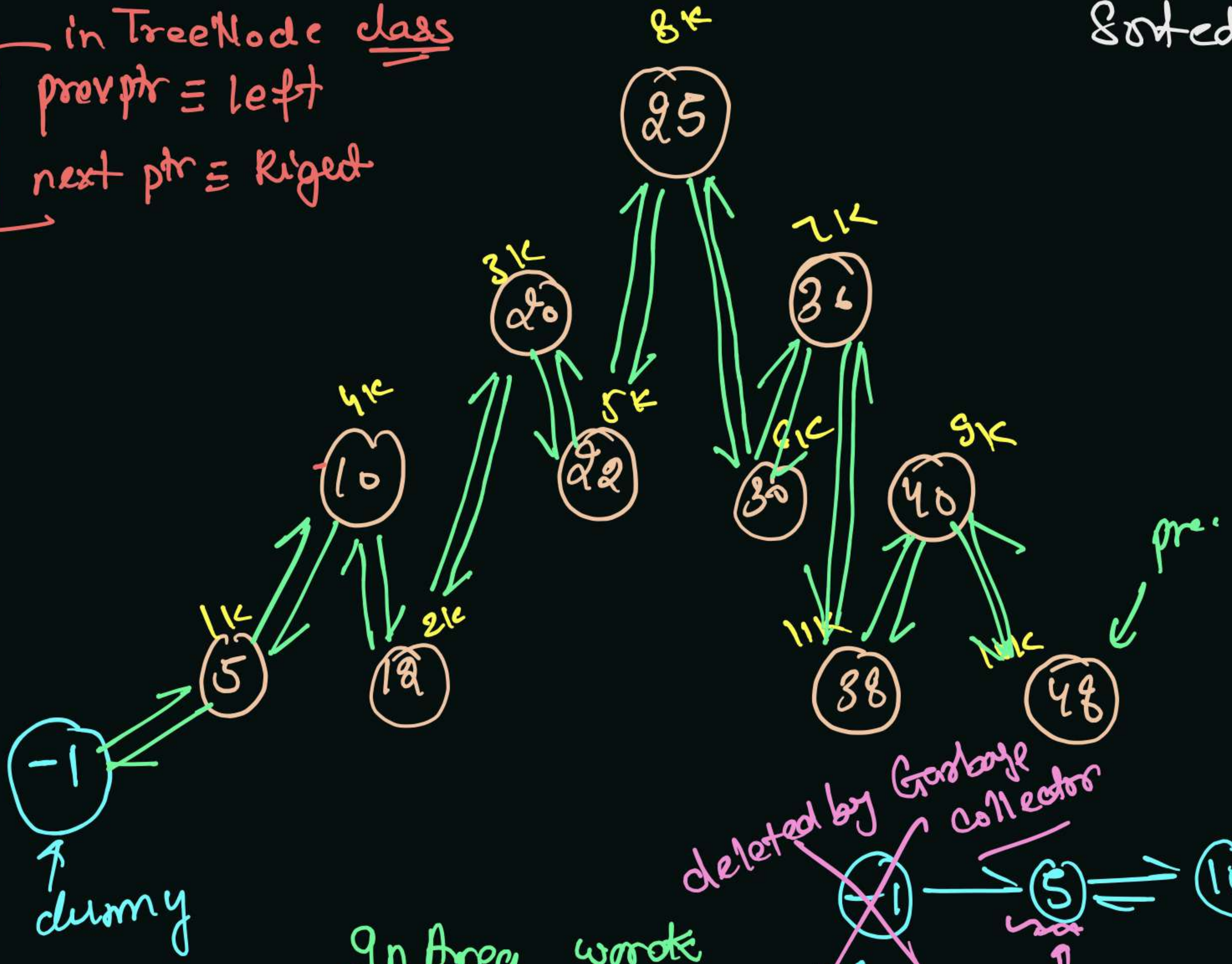
I/p →



prev.right = curr
curr.left = prev;
prev = curr

in TreeNode class
prevptr ≡ left
next ptr ≡ Right

Sorted DLL → circular DLL



After making DLL →

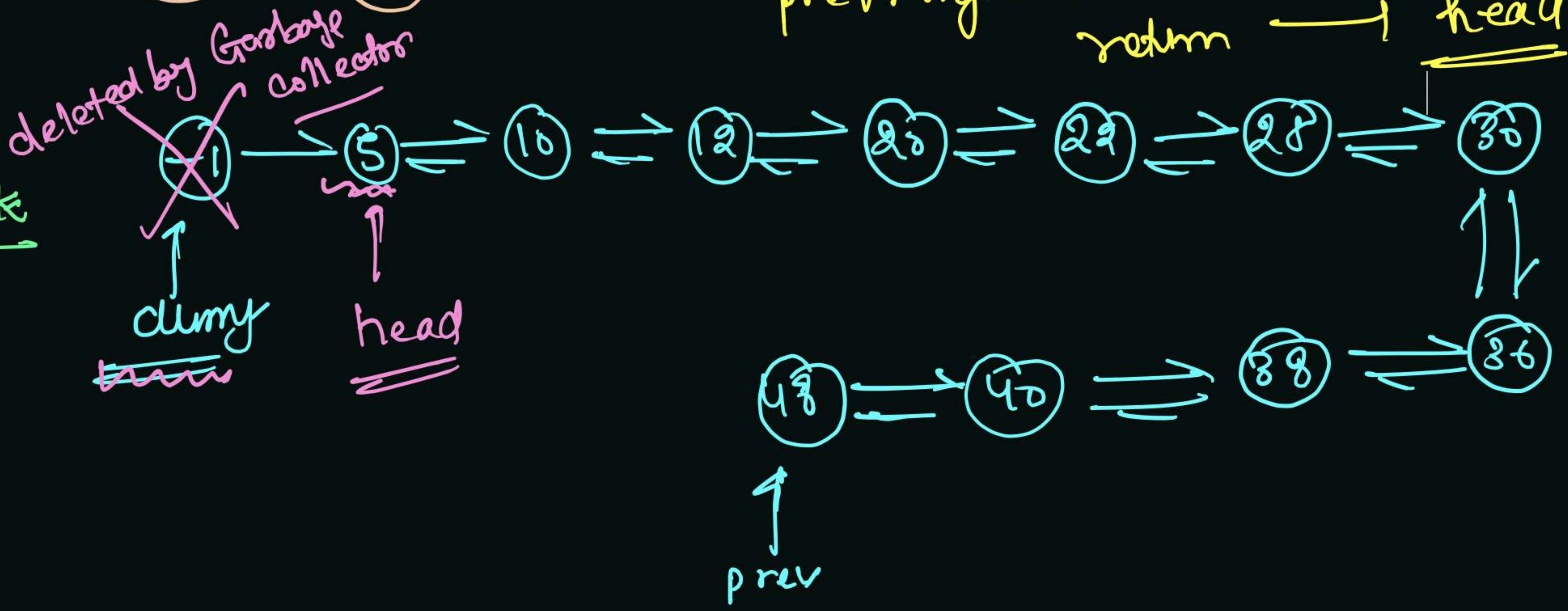
head = dummy.right
head.left = null

convert into circular →

head.left = prev.
prev.right = head.
return → head

In Area work

✓ prev.right = curr
curr.left = prev;
prev = curr


deleted by Garbage collector
-1
5
10
12
20
22
25
30
36
38
40
48
dummy
head
prev