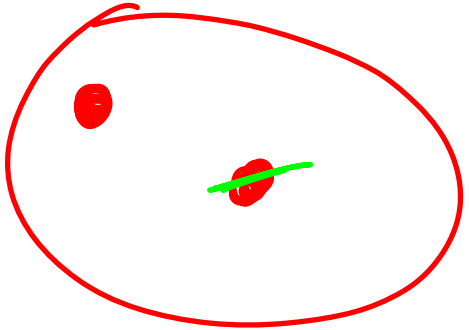
Trees 1

What is a Tree

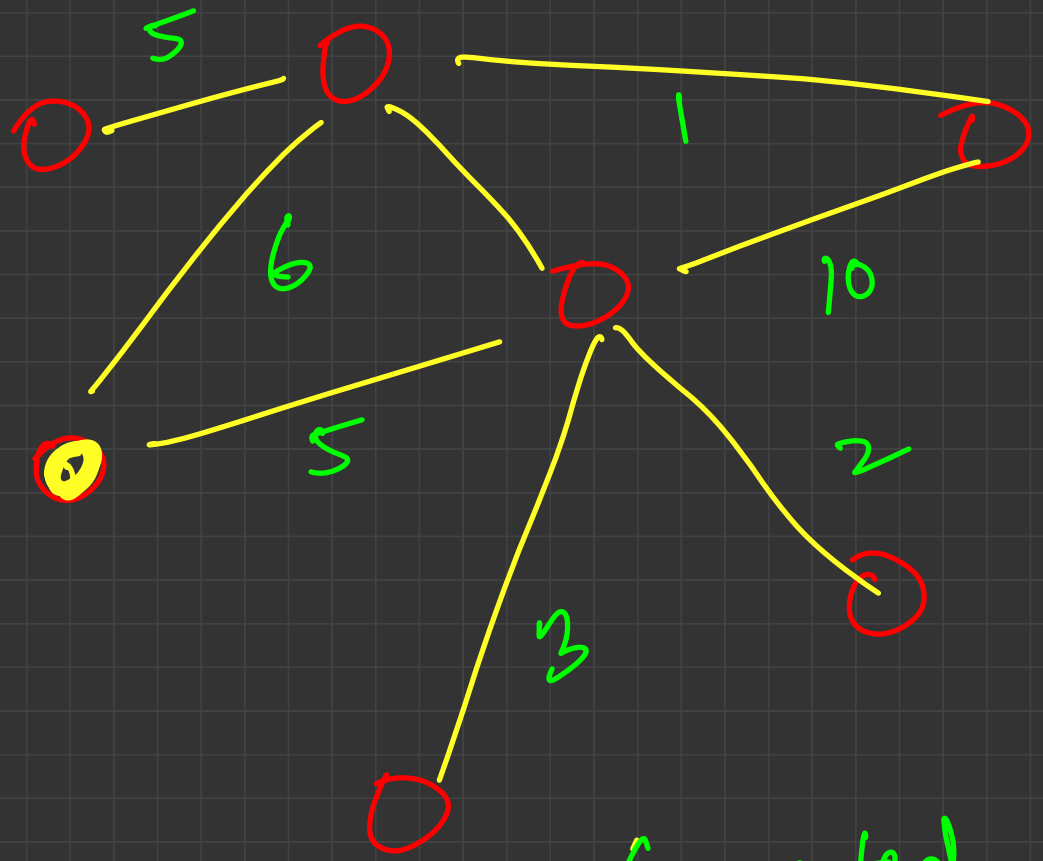
A connected graph of N nodes without any cycles.



What is a graph?

- Imagine it like the Earth
- Contains a bunch of countries connected via roads
- A continent is a group of countries directly or indirectly connected to each other
- Some countries might be in different continents -> disconnected

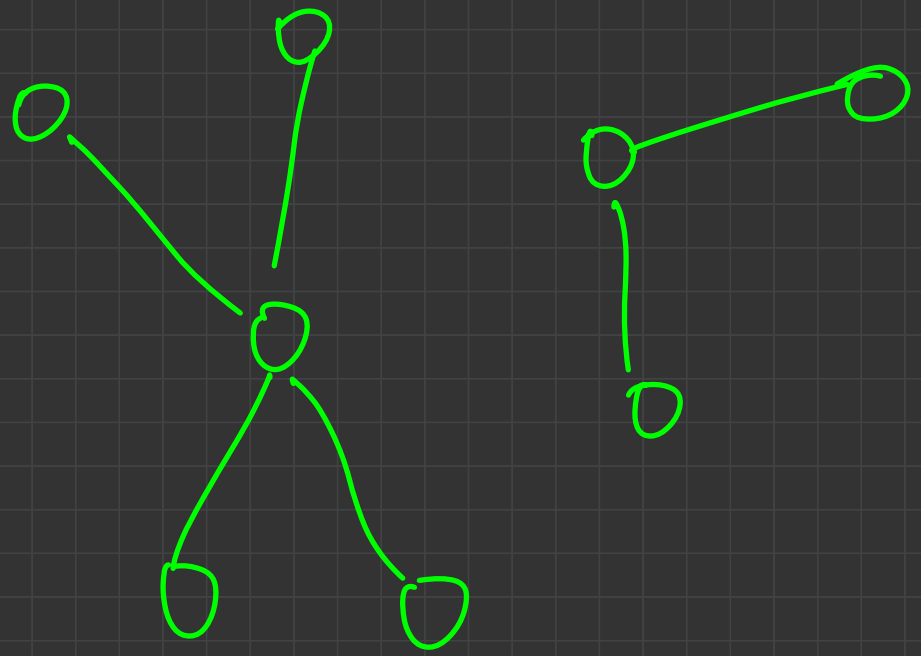
A tree is one such continent with a unique path b/w any 2 countries



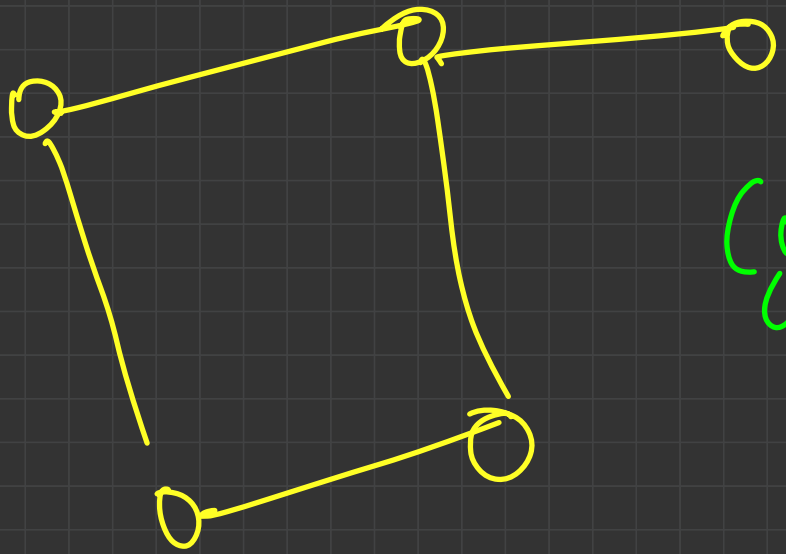
nodes

edges

Connected

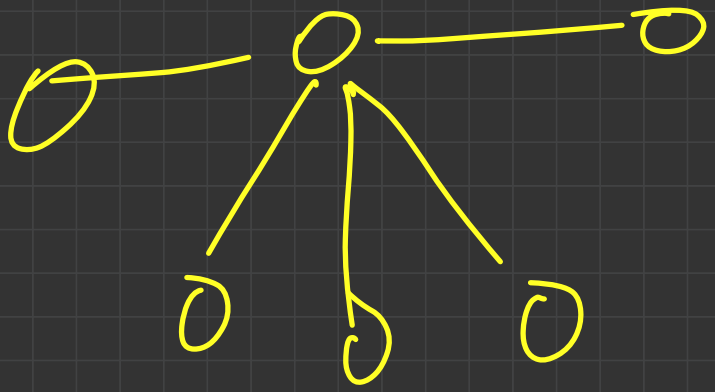


Disconnected Graph

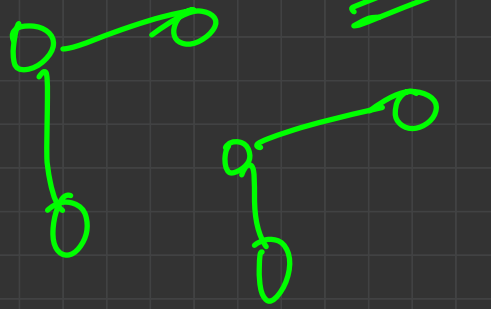


Cyclic Graph

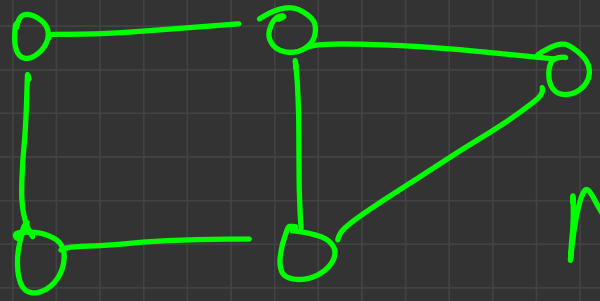
Any 2 nodes have more than 1 path b/w them



Yes

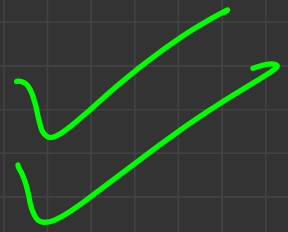


No



No

In a tree b/w any 2 nodes there is just 1 path and it is unique



What is a Tree

Every tree is a graph

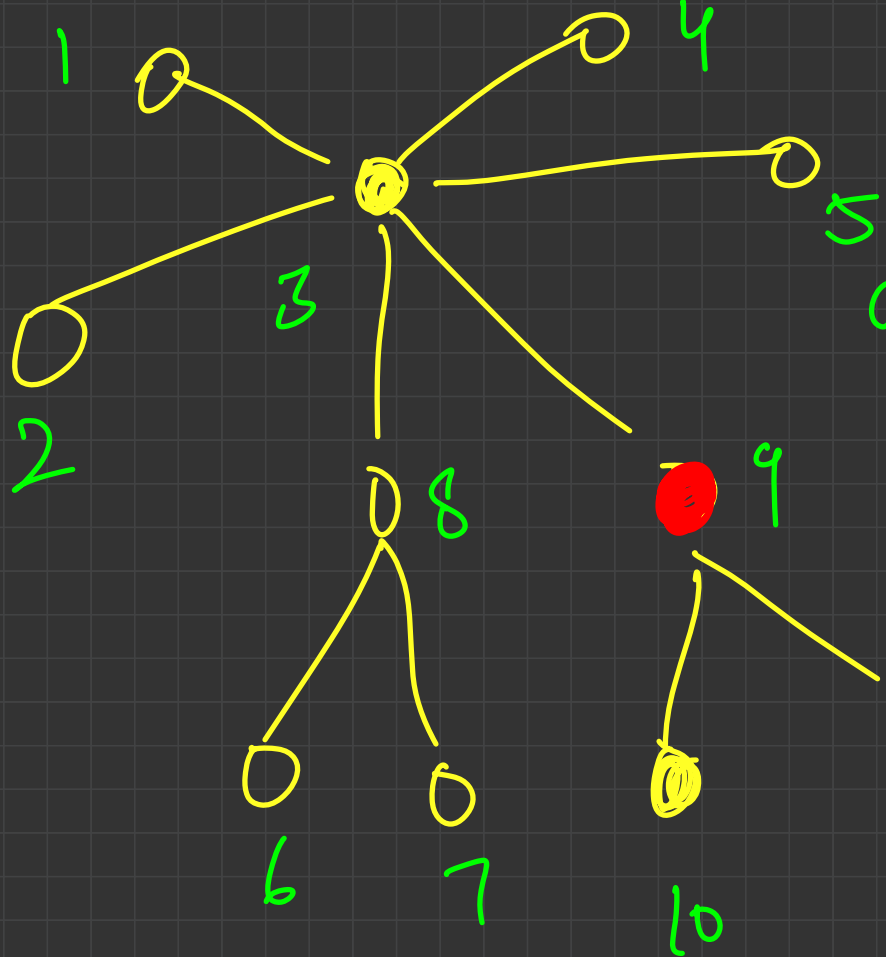
Differentiate b/w a Tree and a Graph from examples

- One Note Illustrations

Some Common Terms

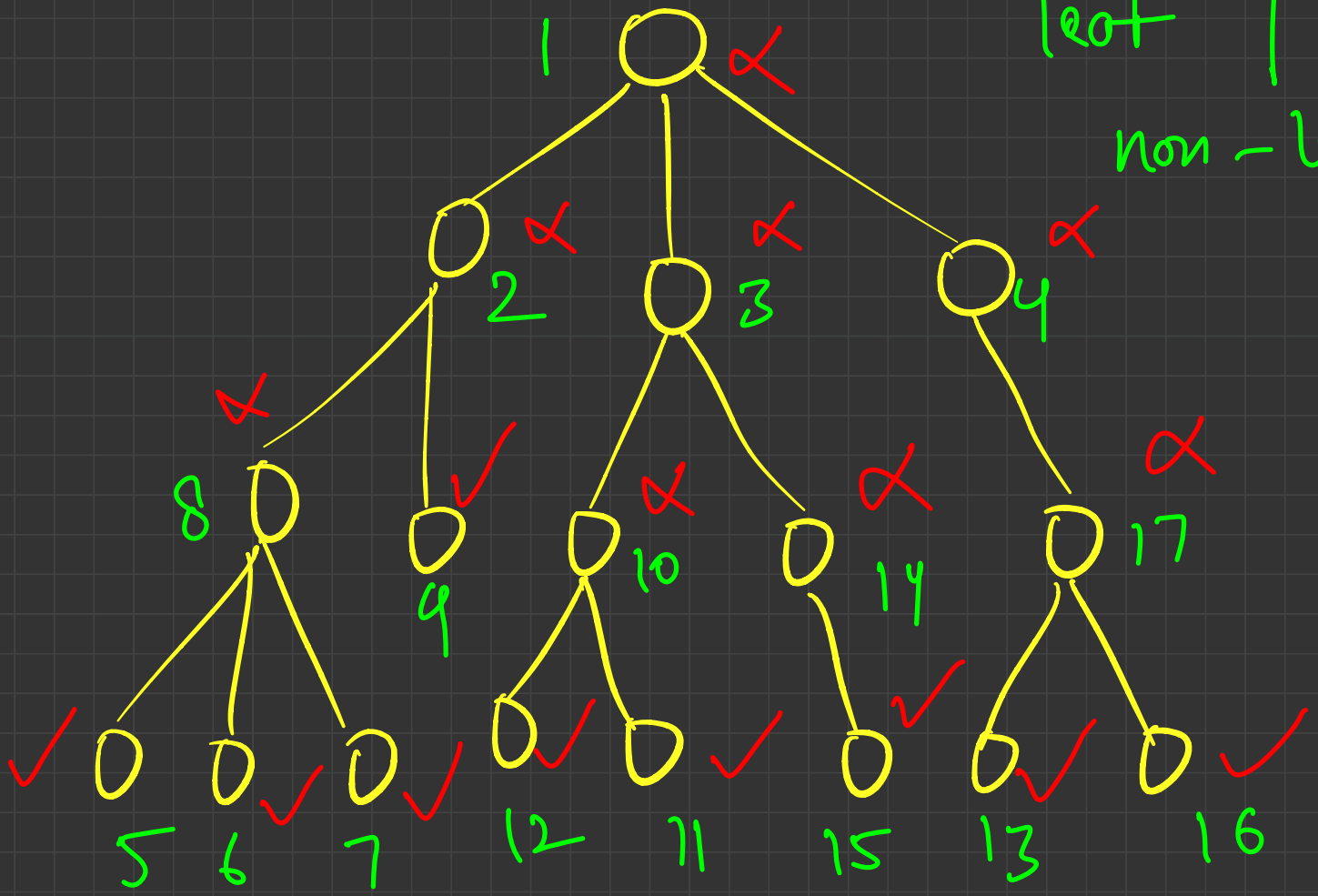
- Neighbour
- Degree → | Neighbours of a node |
- Leaf and non-leaf Nodes (aka Internal Nodes)
- Diameter (can be non-unique)

degree(x) = # of neighbours of x



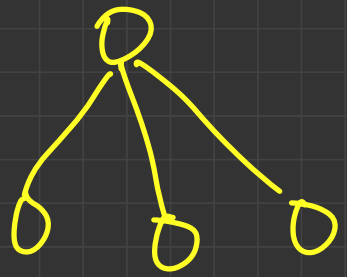
neighbours

any node reachable via just 1 edge

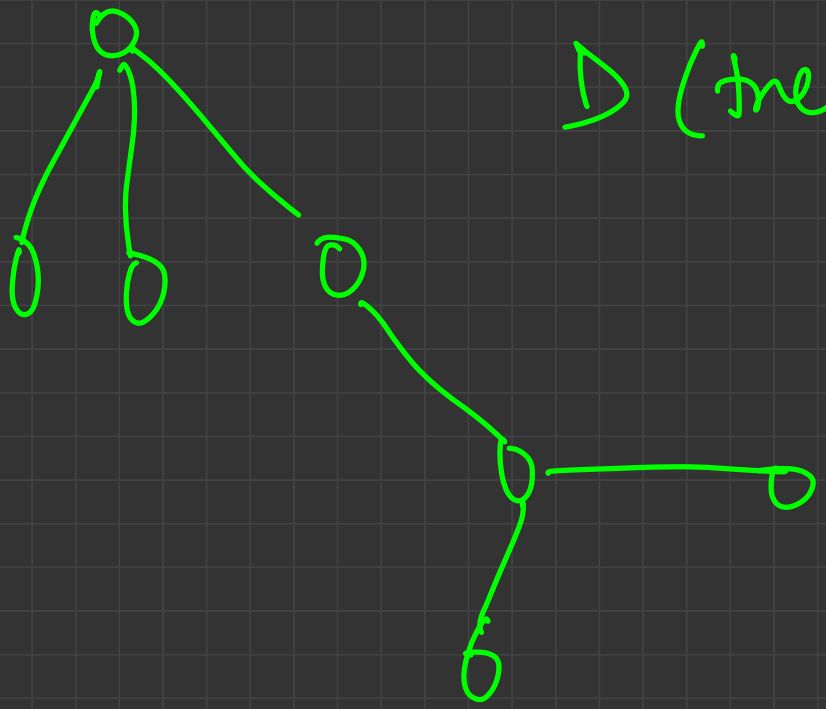

leaf
non-leaf
1
2
3
4
8
9
10
14
17
5
6
7
12
11
15
13
16

Diameter of Tree

⟶ longest distance b/w any 2 nodes in the tree

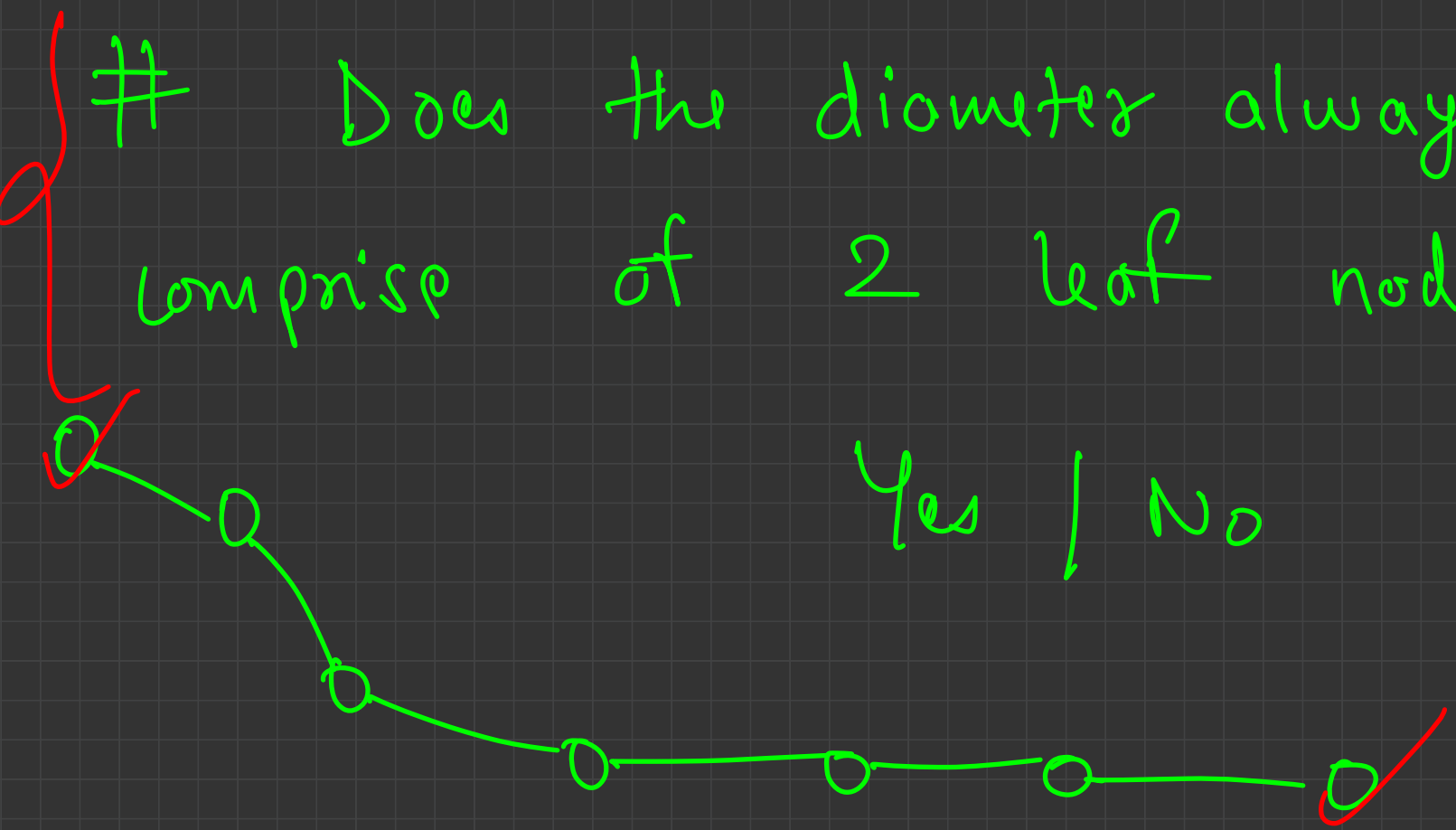


D(tree) = 2



$D(\text{tree}) = 4$

Does the diameter always comprise of 2 leaf nodes



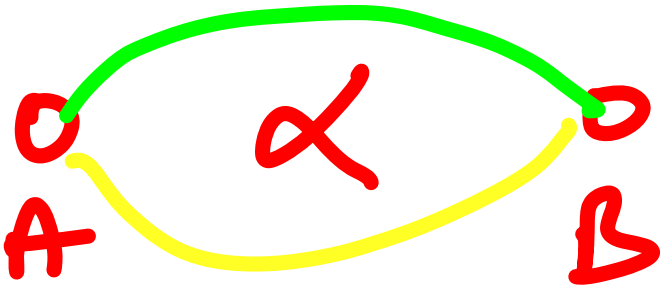
Yes / No


A
B
beat
not a beat

Skewed tree

$D(\text{tree}) = 4$

Properties 1



- Number of Edges in a Tree for N nodes
  - N - 1
- Number of paths between 2 nodes
  - 1
- Sum of Degree of all nodes
  - 2 * (N - 1)

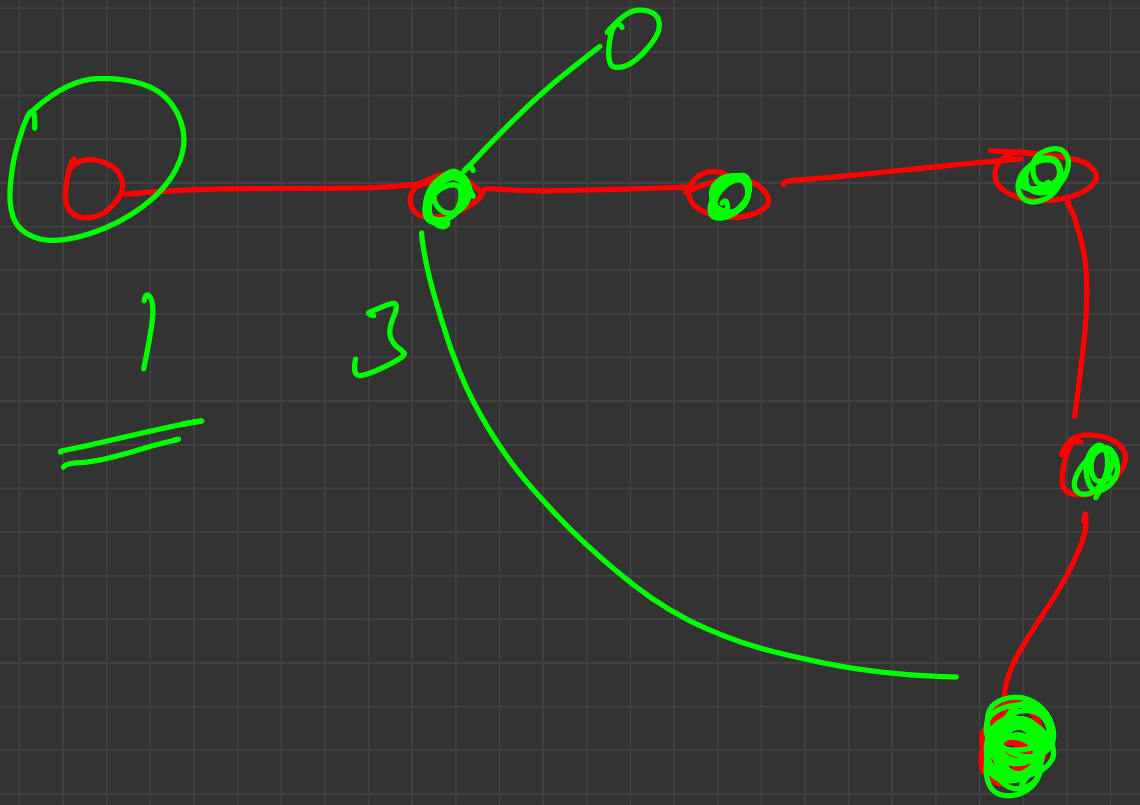
$$= \sum_{i=1}^{n} deg[i]$$

even

- Can there be less than 2 leaf nodes in a Tree
  - No, except for the case when there is just one node in the entire tree

No

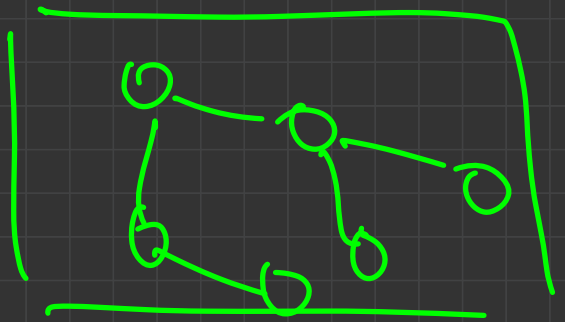
0


1
3

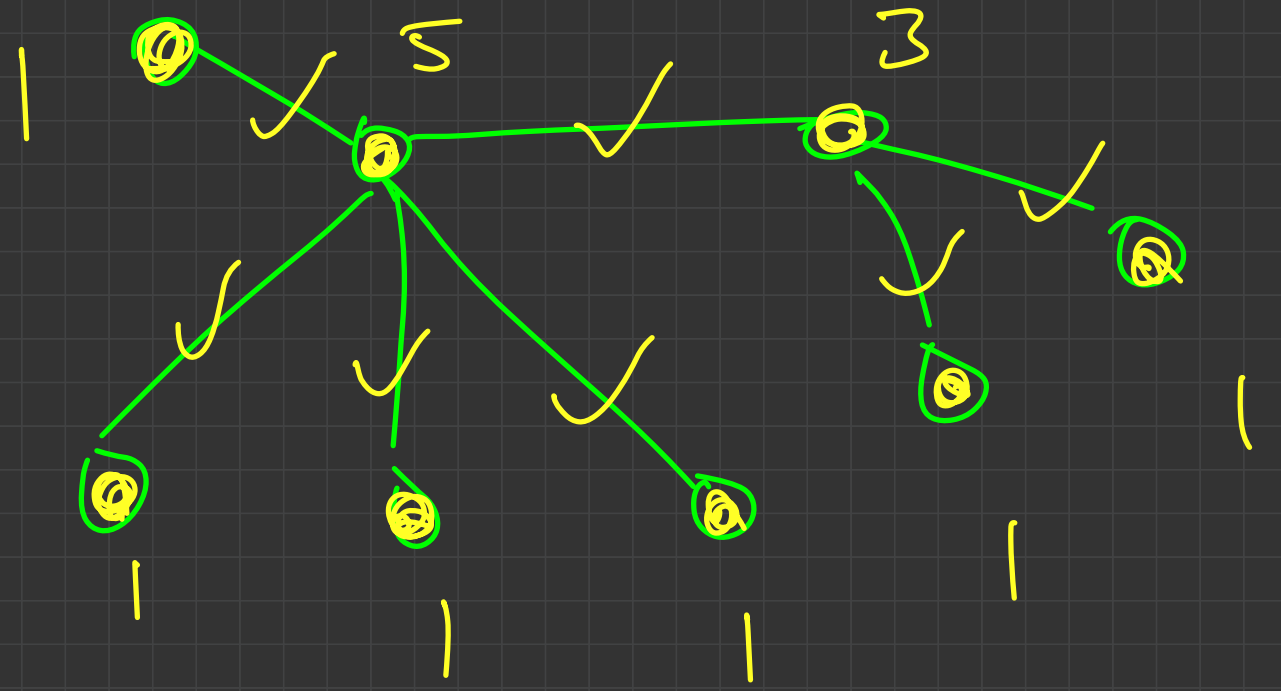
1 → 0 ✓

2 → 1 ✓

for k nodes k-1 edges are required



k-1 edges + 0



7 edges = 14

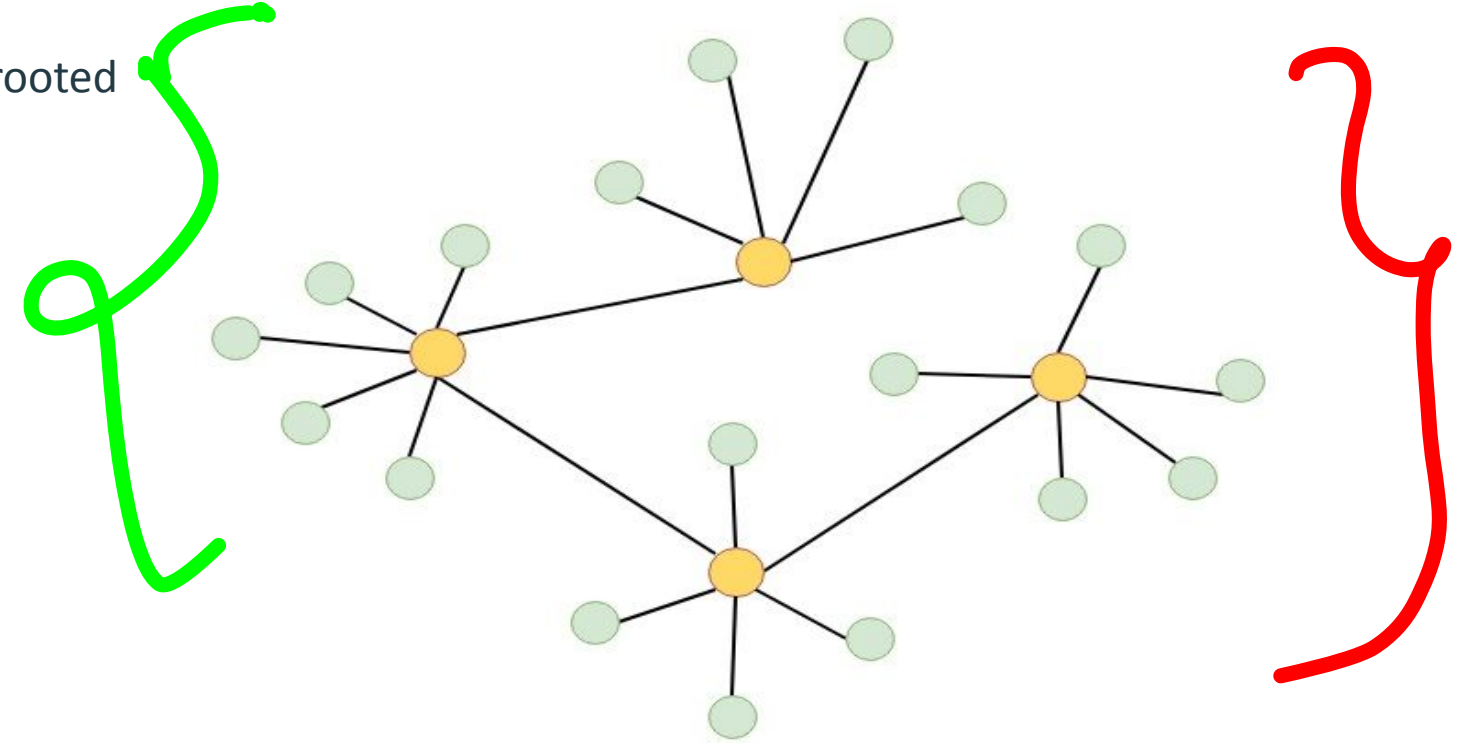
5 nodes

1, 2, 2, 2, 1

Total degree = 2 x (4) = 8

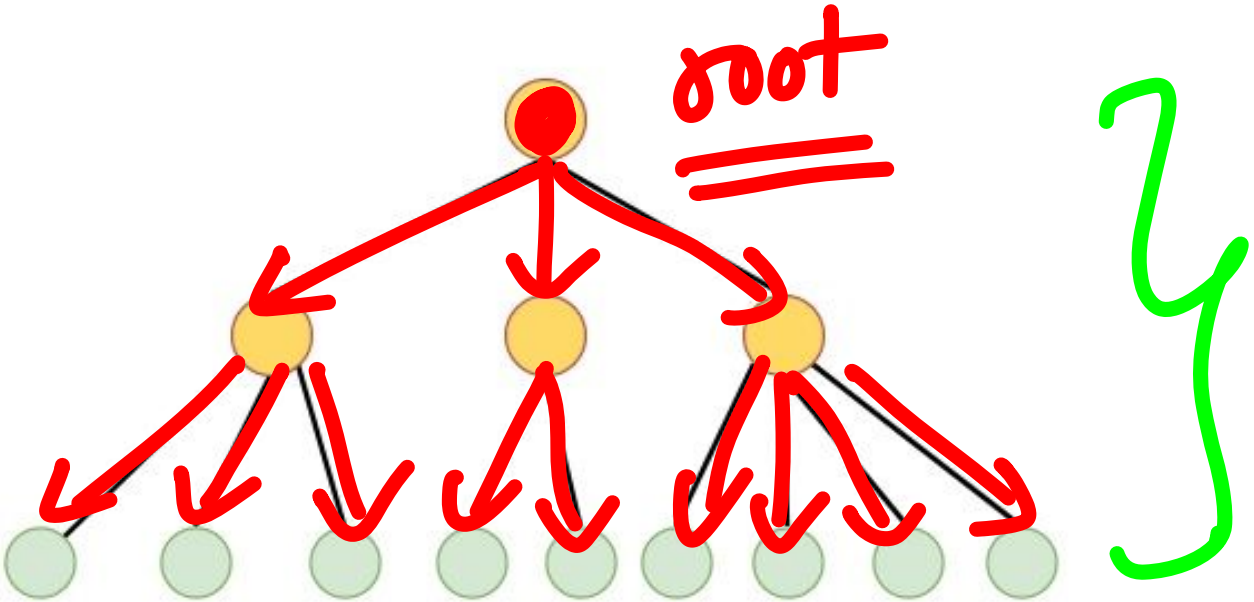
Rooted and Unrooted Trees

Unrooted



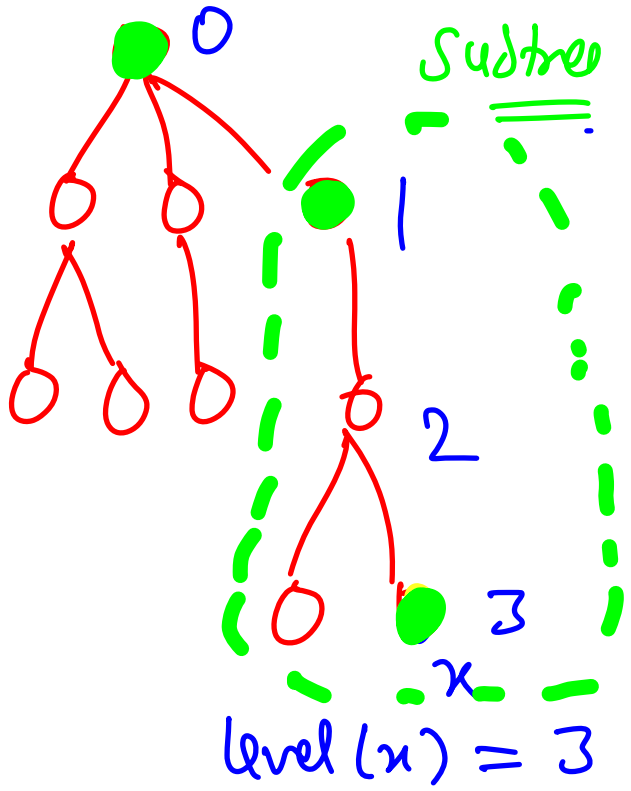
Rooted and Unrooted Trees

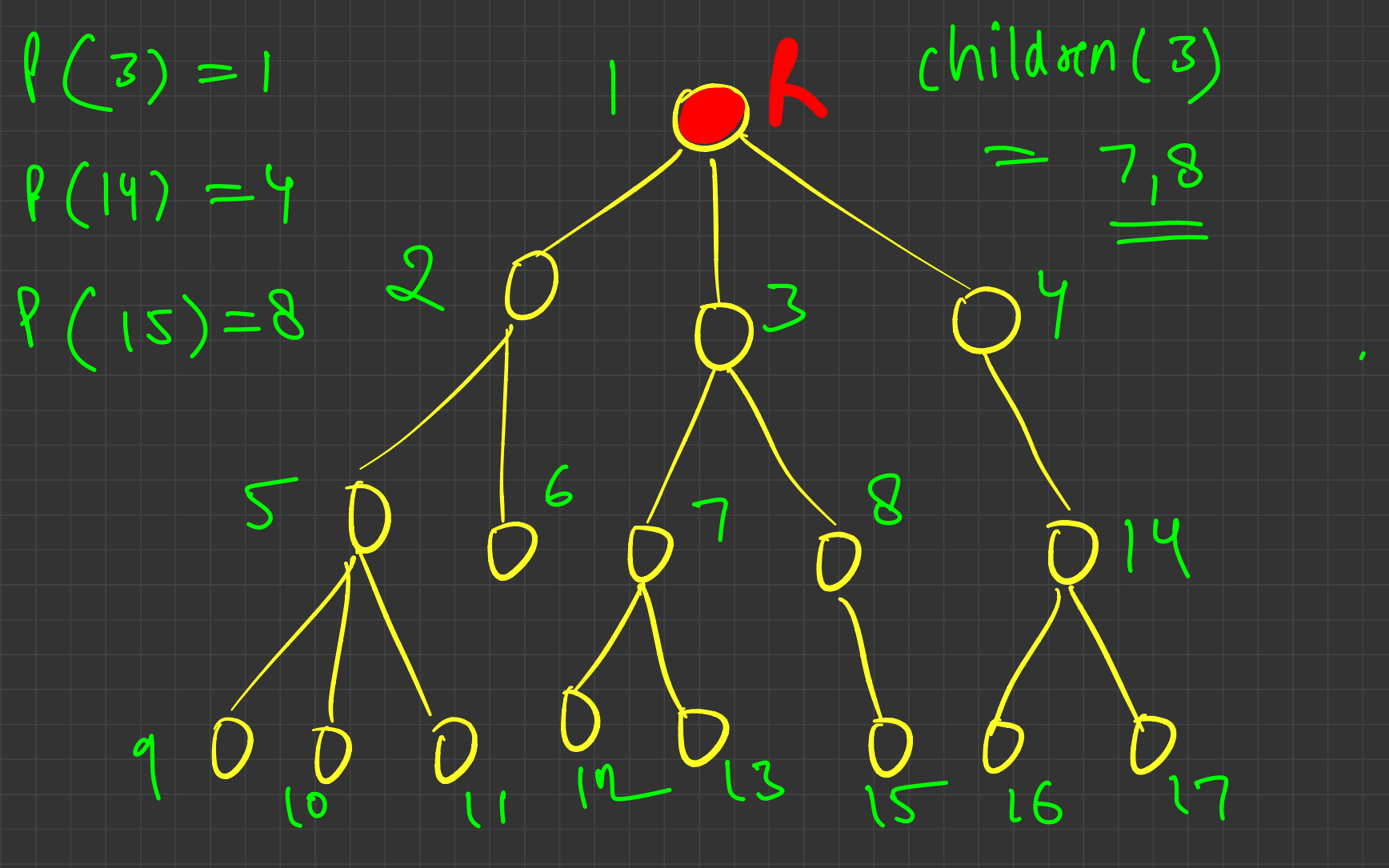
Rooted



Some more terms

- Root
- Parent
- Child
- Ancestor
- Descendant
- Level of Node
- Subtree
- Subtree Size
- Height of Tree
- Lowest Common Ancestor




P(3) = 1
P(14) = 4
P(15) = 8
children(3) = 7, 8
1
R
2
3
4
5
6
7
8
14
9
10
11
12
13
15
16
17

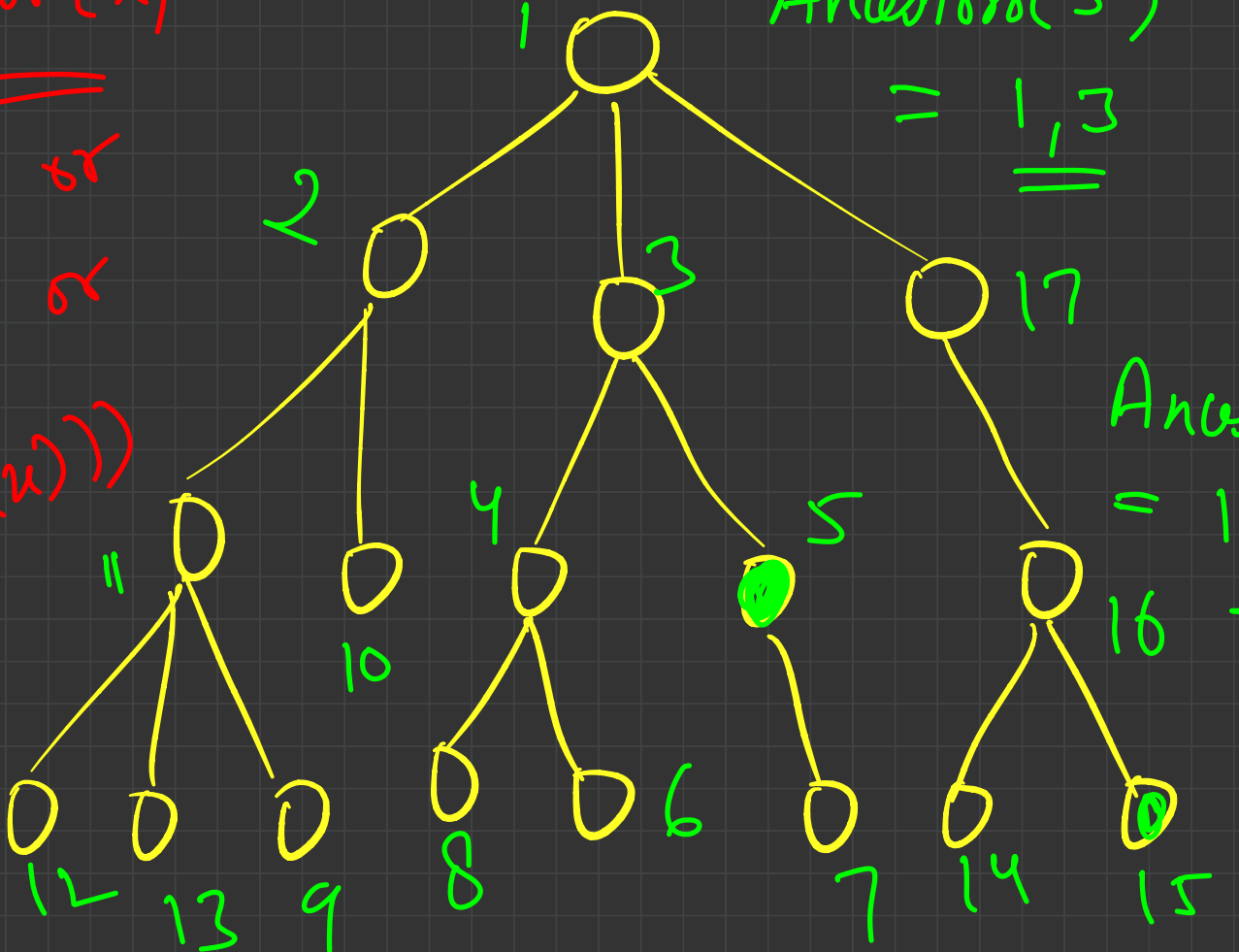
Ancestor(x)

P(x) or

P(P(x)) or

P(P(P(x)))

...



Ancestors(5)

= 1, 3

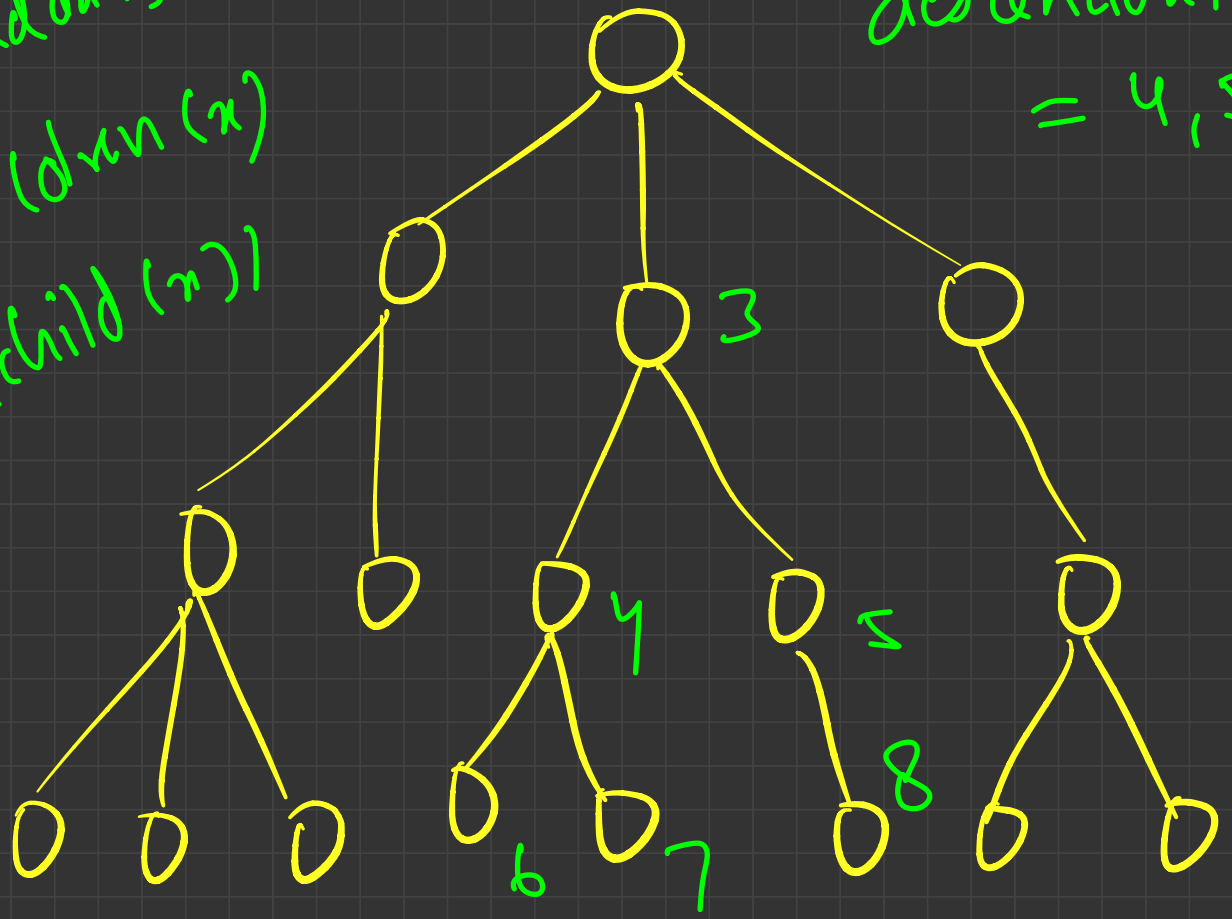
Ancestors(15)

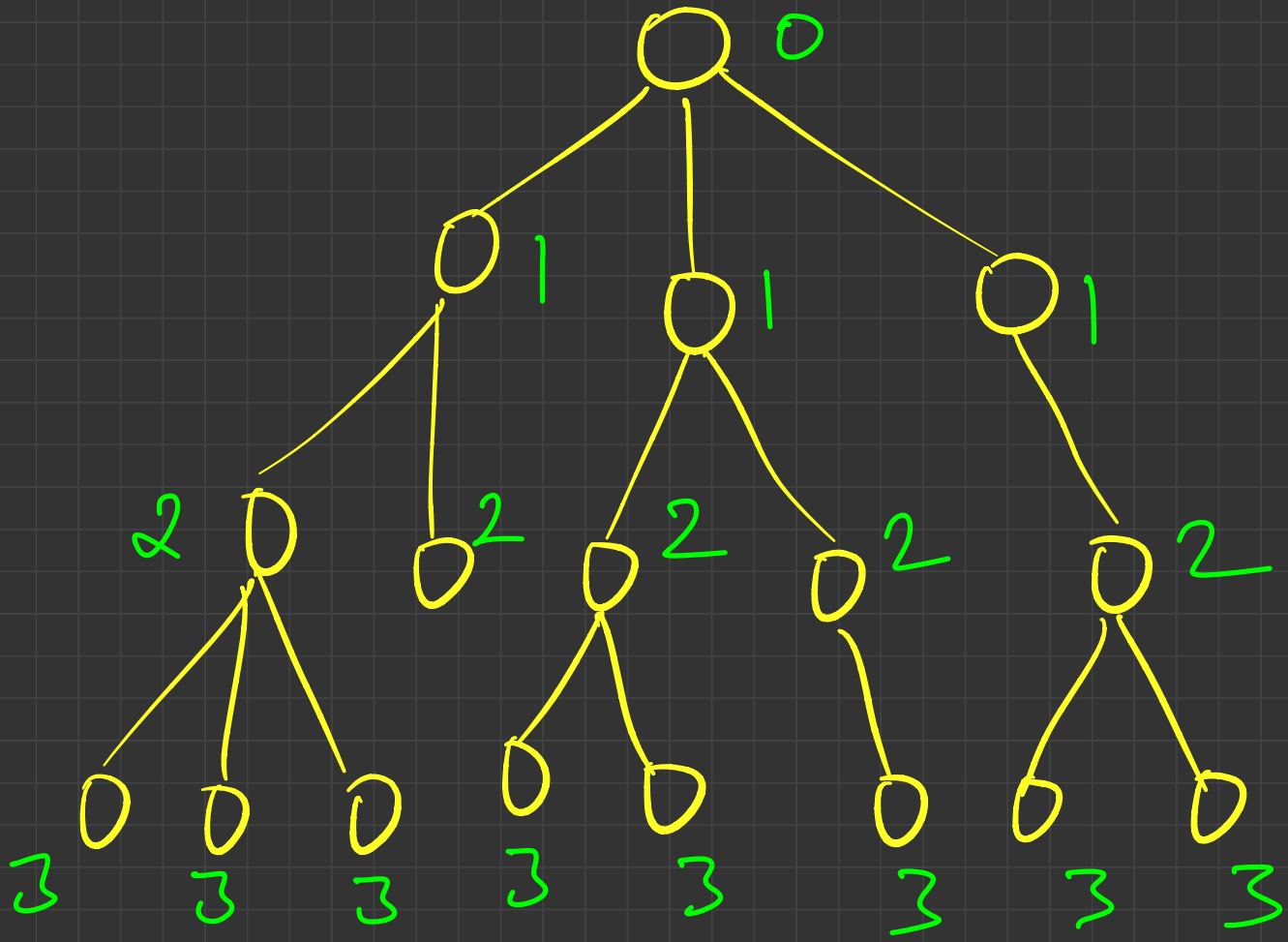
= 16, 17, 1

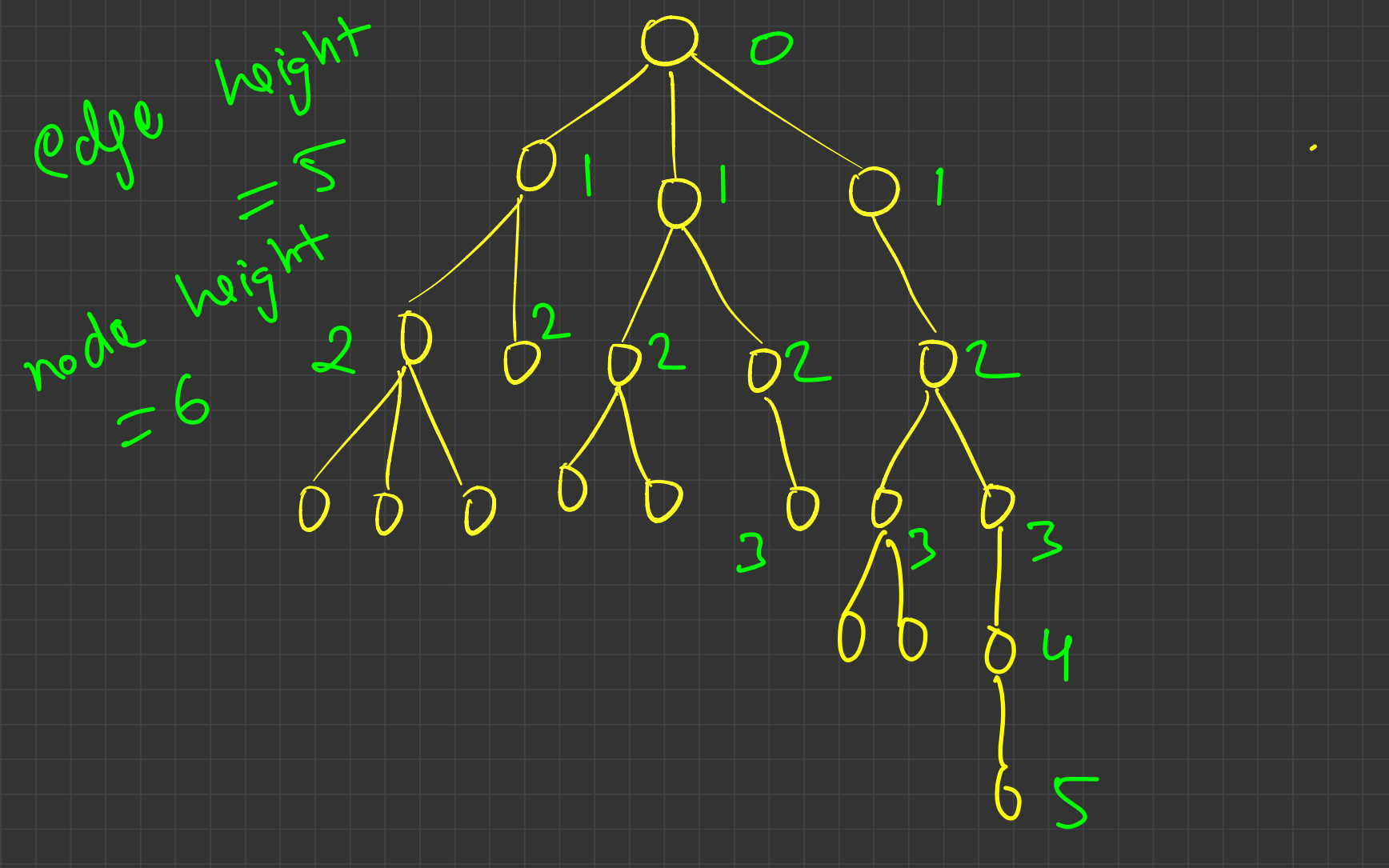
descendants(x)

= children(x) ∪ children(children(x)) ∪ ⋯

descendants(3) = 4, 5, 6, 7, 8




0
1
1
1
2
2
2
2
2
3
3
3
3
3
3
3
3

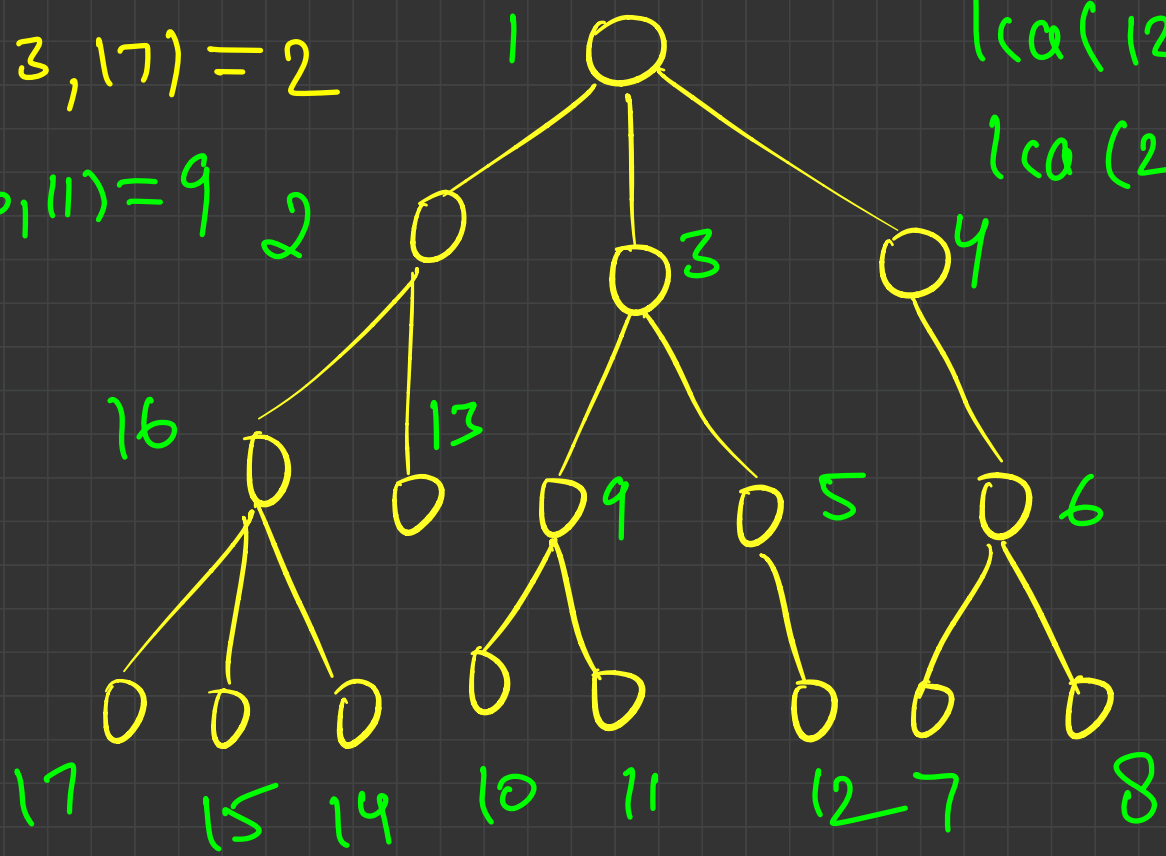

edge height
= 5
node height
= 6

lca(13,17) = 2

lca(10,11) = 9

lca(12,7) = 1

lca(2,13) = 2

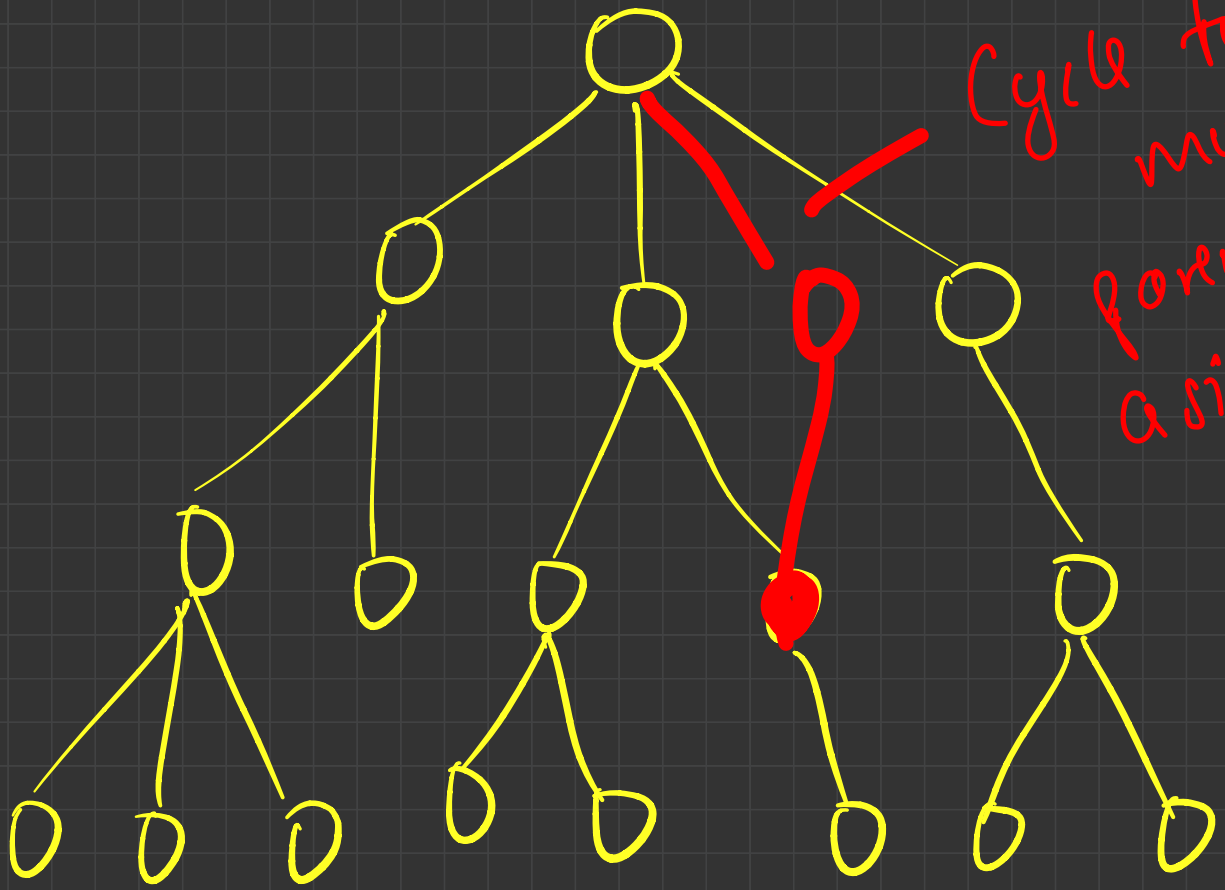


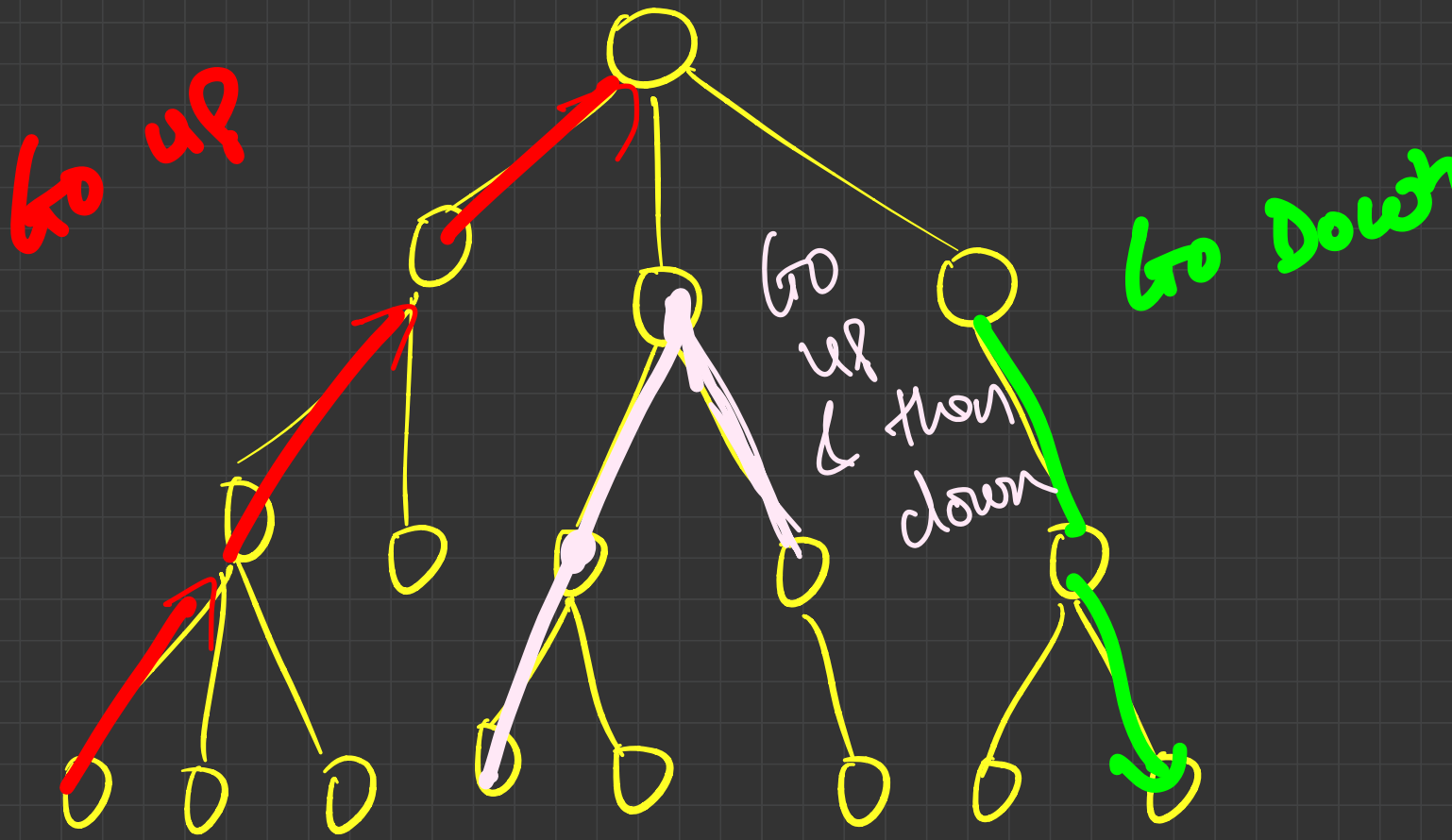
Properties 2

- Can there be more than 2 parents of a single node in a rooted tree
  - No
- Path property. What are the only 3 types of paths possible?
  - 1. **Go Up, Come Down** 2. **Go Up** 3. **Come Down**
- How to color a Tree with just 2 colors such that no two neighbours have the same color
  - Just root the tree and color level wise

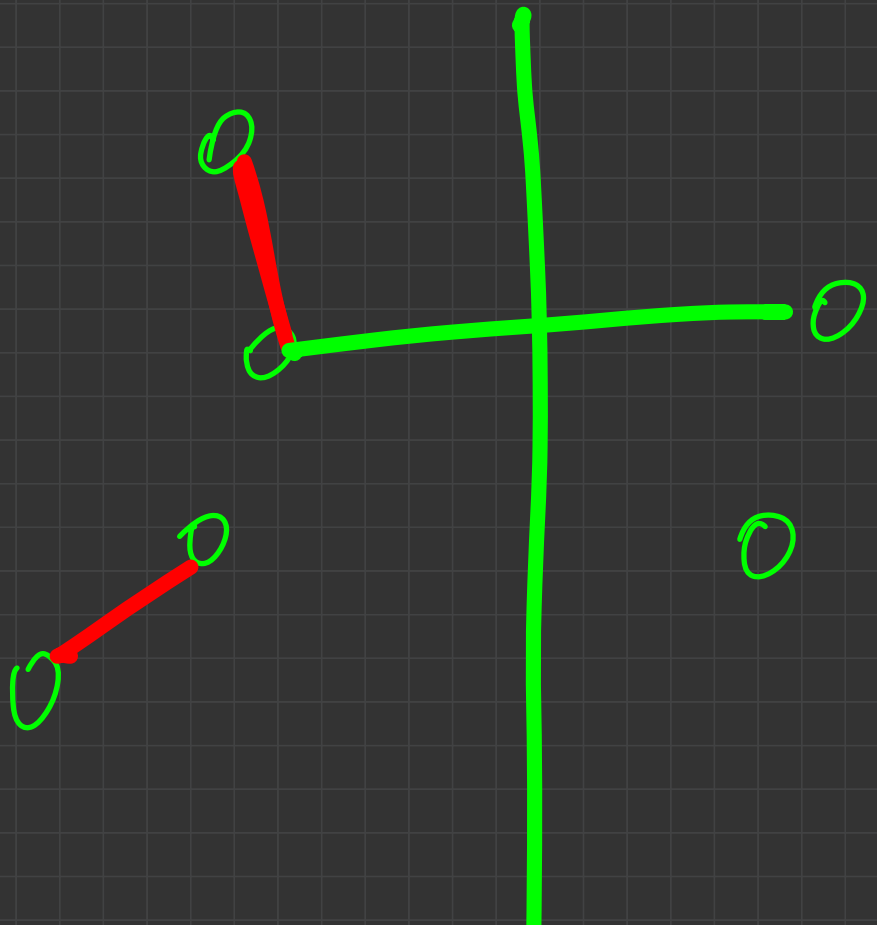
Bonus Tip:

Most Codeforces problems less than 1800 rated on Codeforces can be easily solved if you just remember the basic properties and learn to apply them.

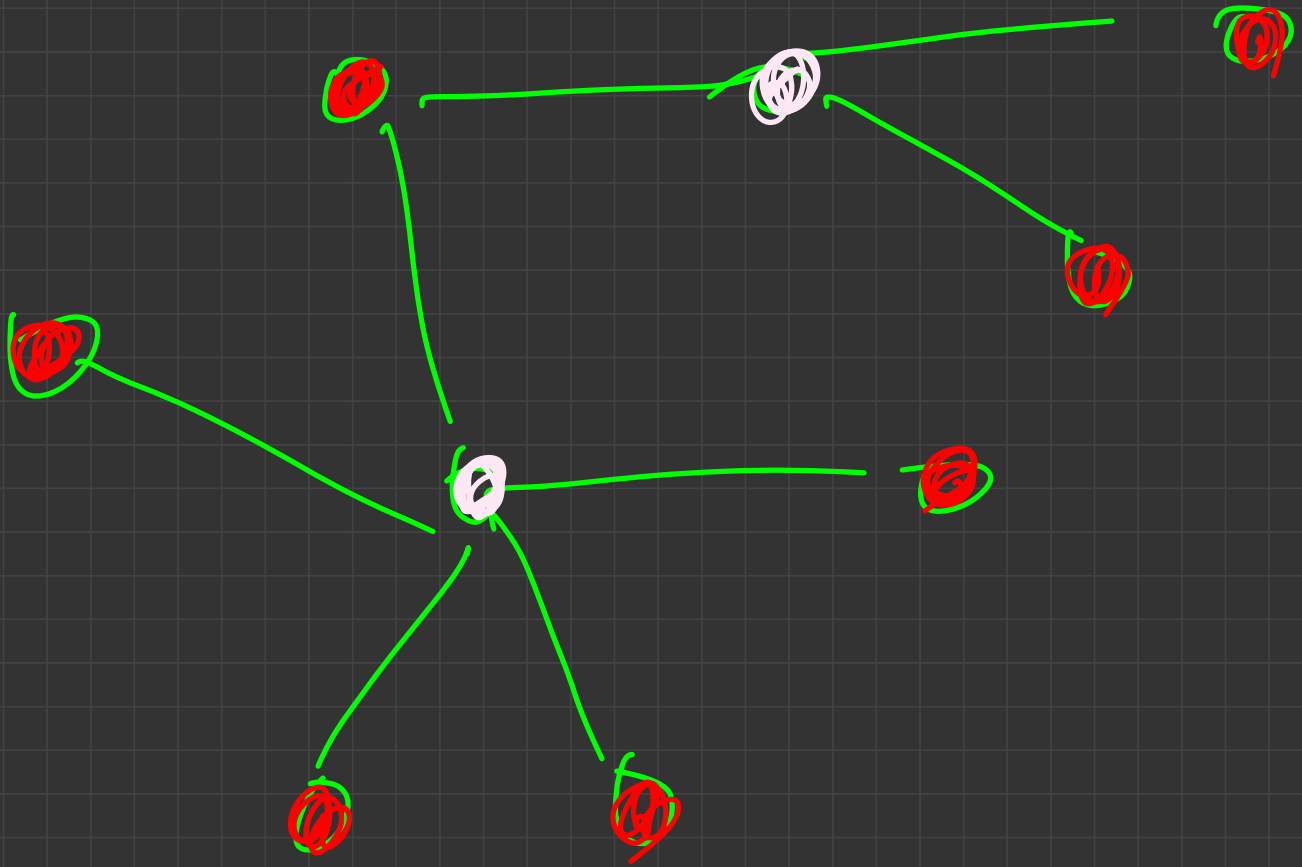

Cycle for multiple parents of a given node

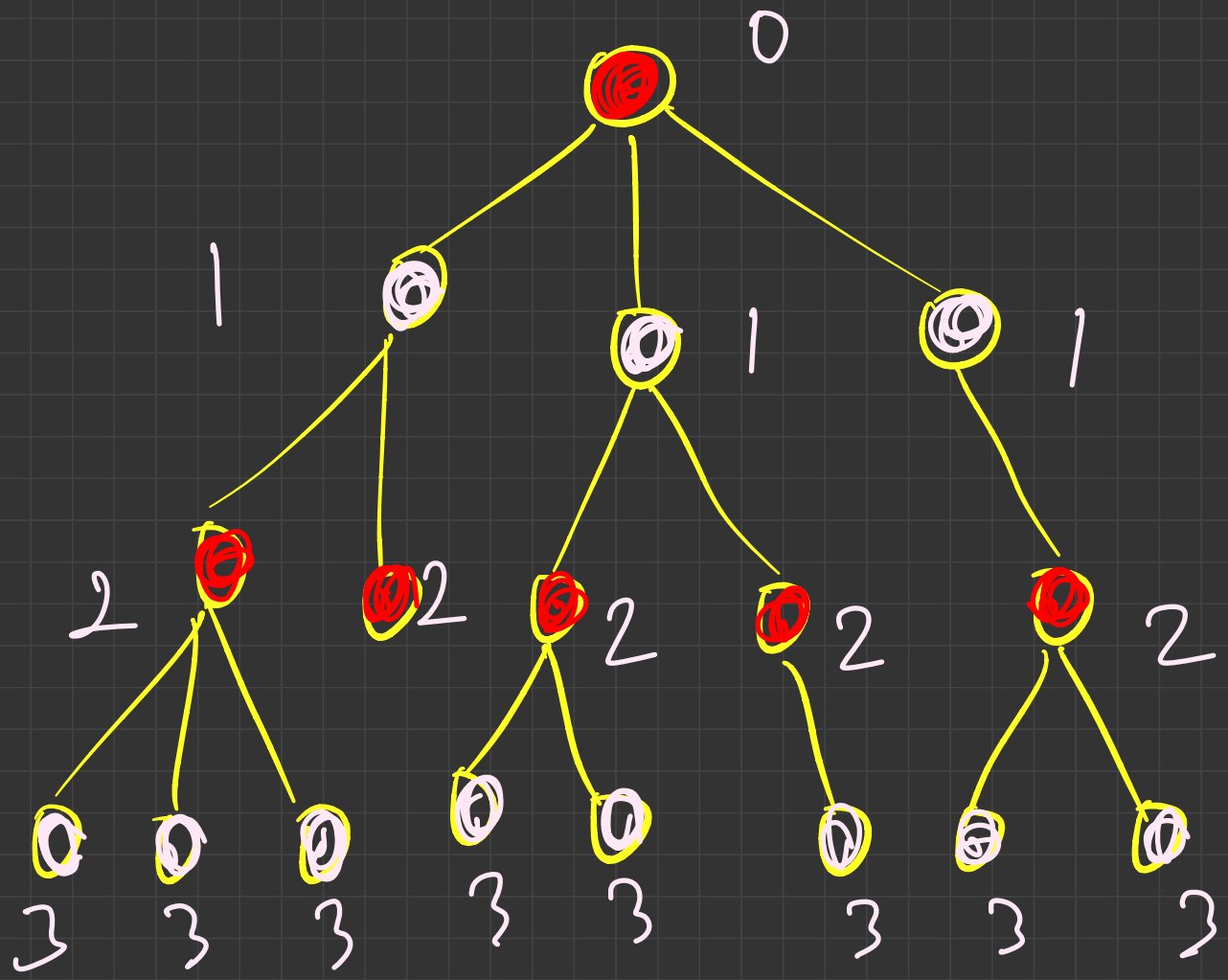

Go up
Go up & then down
Go Down

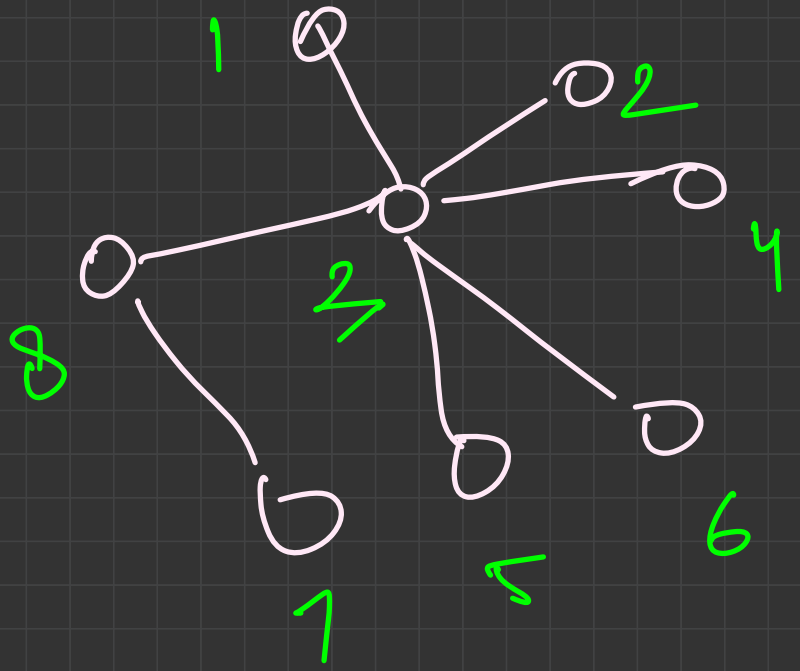
A tree is always bipartite



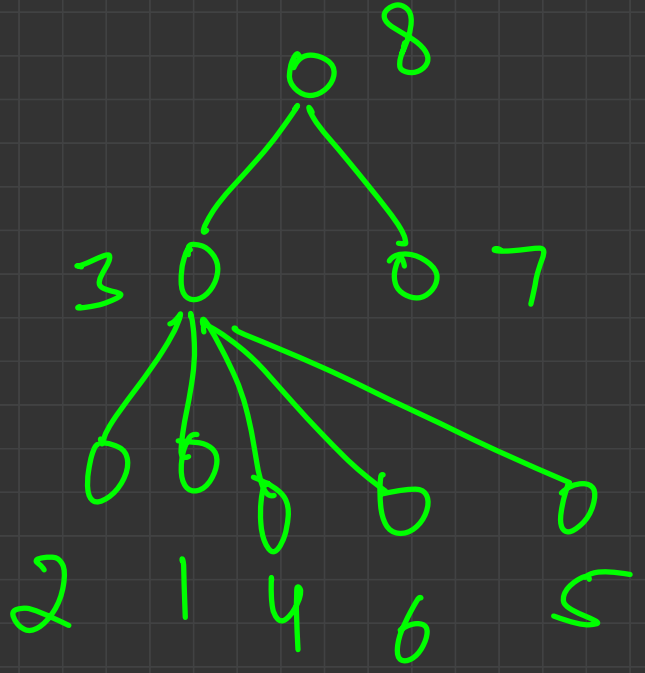
≤ 1800

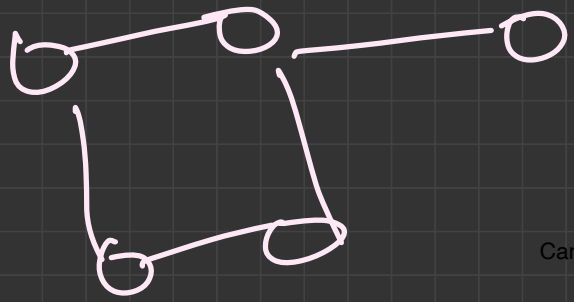



0
1
1
1
2
2
2
2
2
3
3
3
3
3
3
3
3



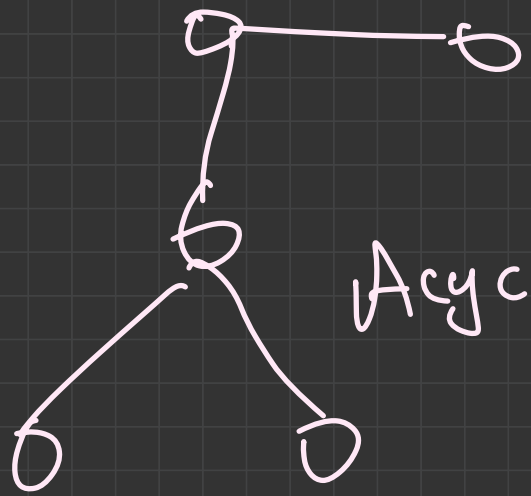
root = 8





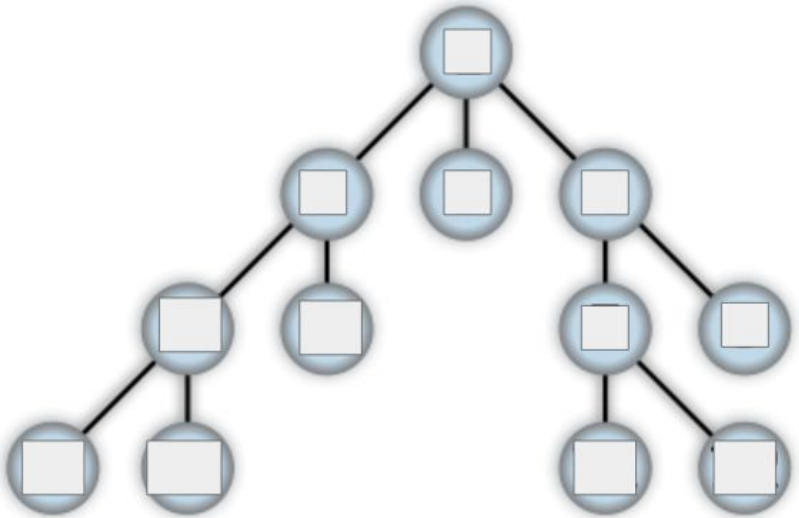
Cyclic

Canthere



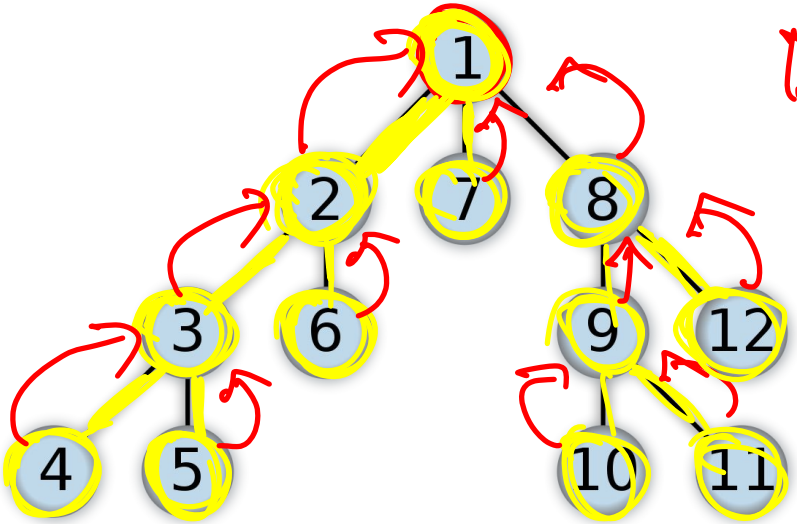
Acyclic

Traversals in a Tree



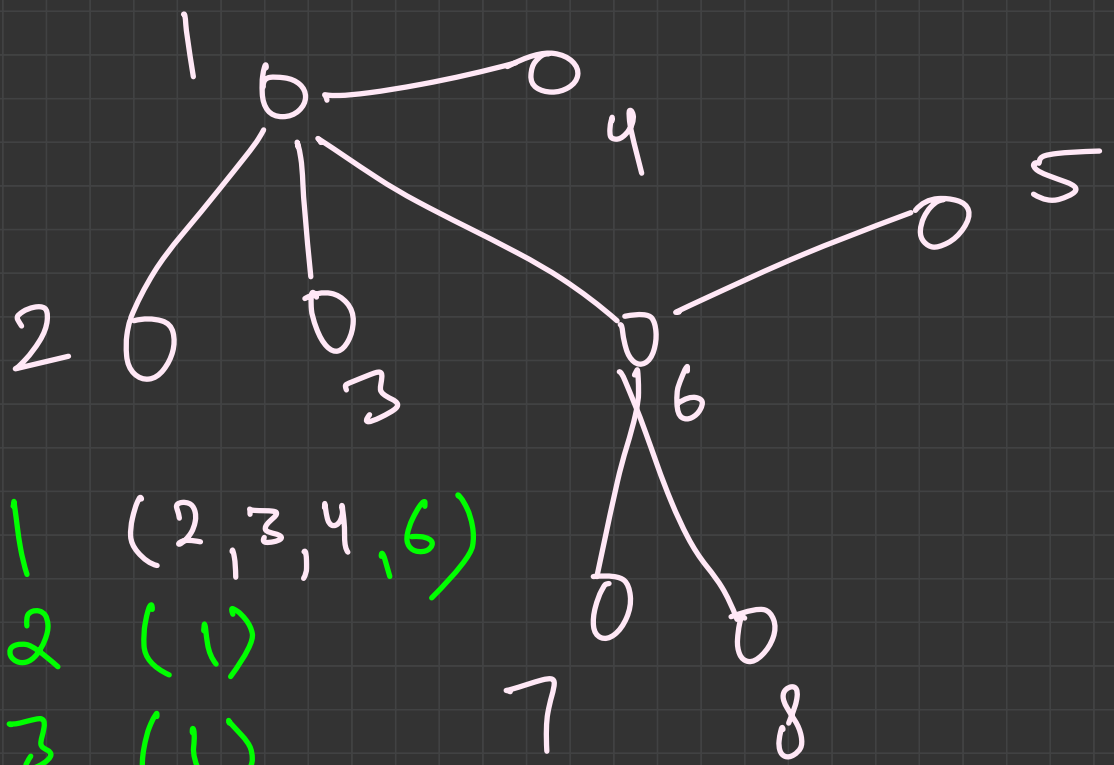
How to traverse the tree in a way such that we visit all nodes

DFS Traversal in a Tree



Depth first search

Nodes are numbered in the order in which they are visited



1 (2, 3, 4, 6)

2 (1)

3 (1)

4 (1)

6 (1, 5, 7, 8)

5 (6)

7 (6)

8 (6)

8 ✓

1 2 ✓

5 6 ✓

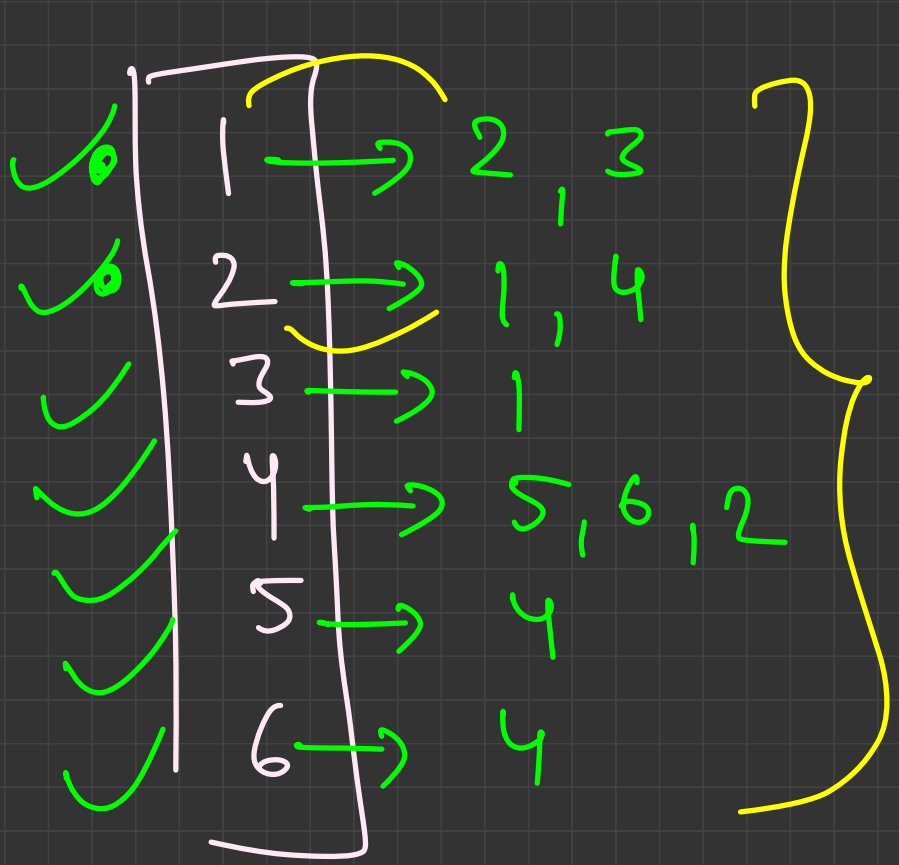
1 4 ✓

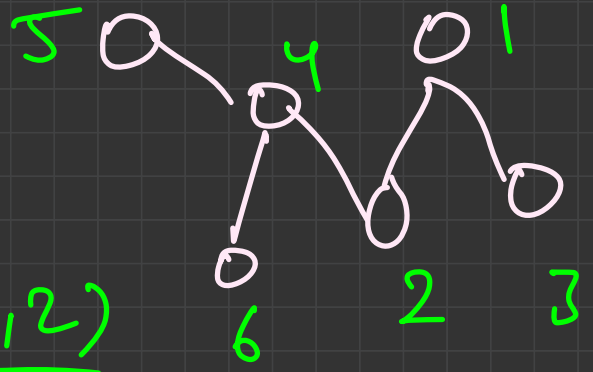
1 3 ✓

6 7 ✓

6 8 ✓

1 6 ✓

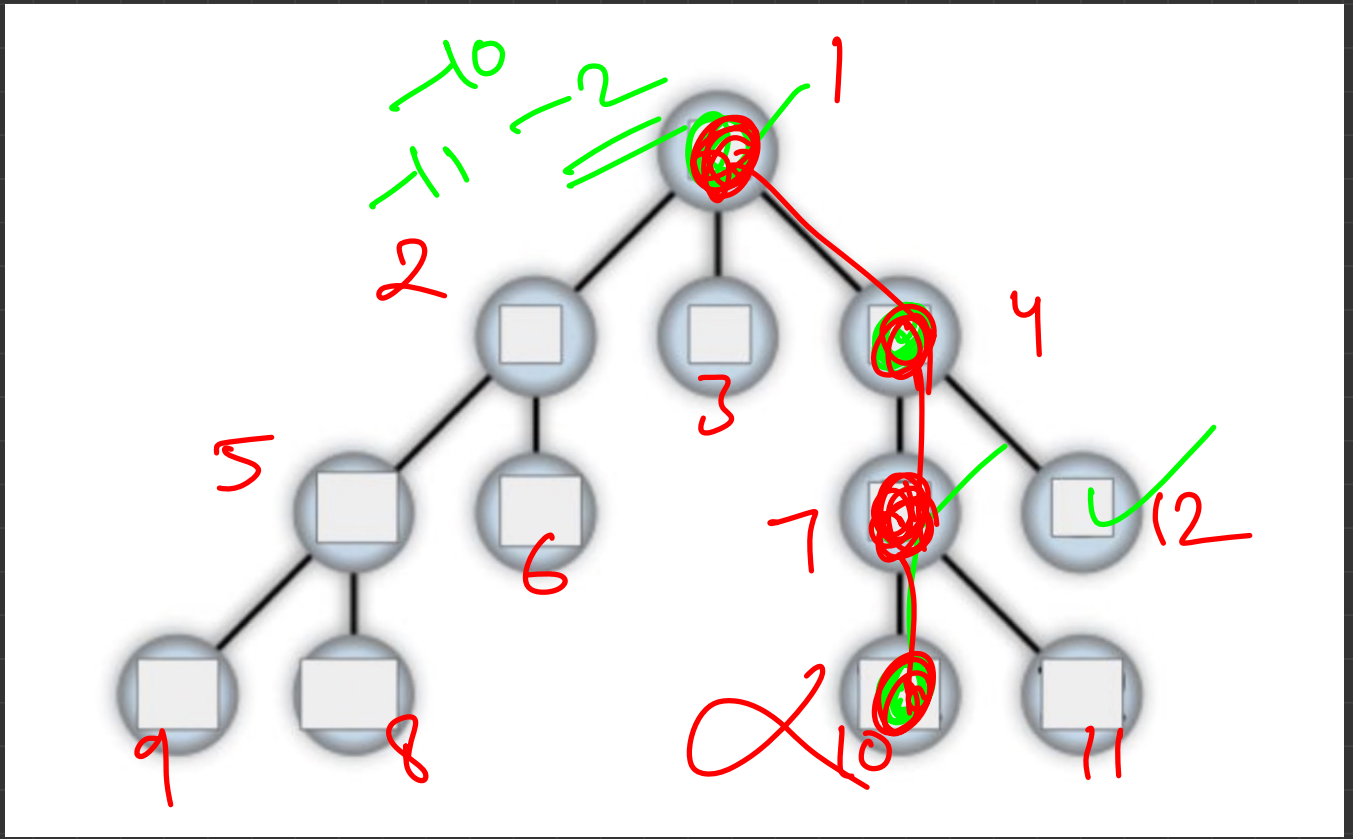




(1,2)

Space complexity
$= O(n)$

$2(n-1) = O(n)$



1, 4, 7, 10, 11, 12, 3, 2, 5, 9, 8, 6

DFS Traversal in a Tree

Implementation:

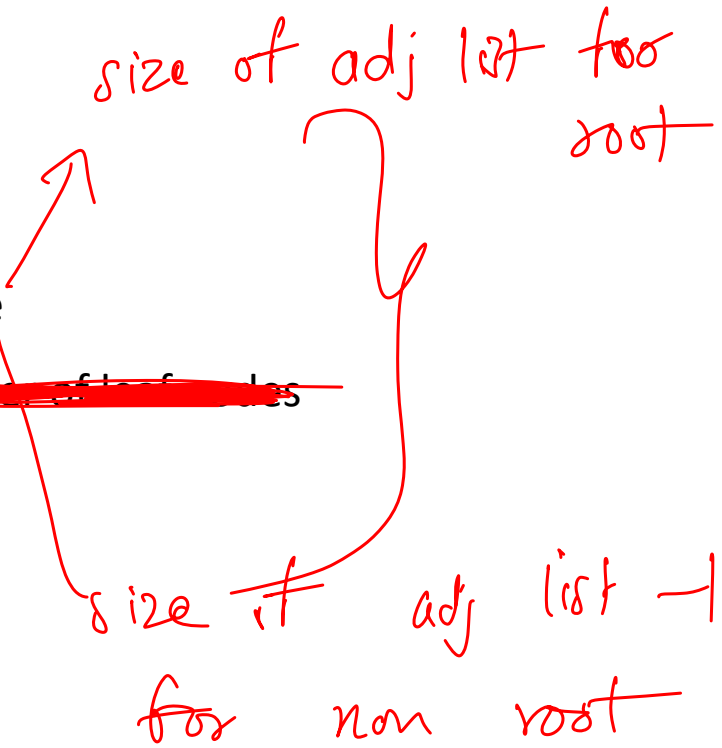
```
void dfs(int currentNode, vector<vector<int>>& adj, int parent, vector<int>& ans){
    ans.push_back(currentNode);
    for(int neighbour : adj[currentNode]) {
        if(neighbour != parent)
            dfs(neighbour, adj, currentNode, ans);
    }
}
void solve(){
    int n;
    vector<vector<int>> adj(n);
    for(int i = 0; i < n - 1; i++){
        int u, v;
        cin >> u >> v;
        u--, v--;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    int root = 0;
    vector<int> dfs_traversal;
    dfs(0, adj, -1, dfs_traversal);
}
```

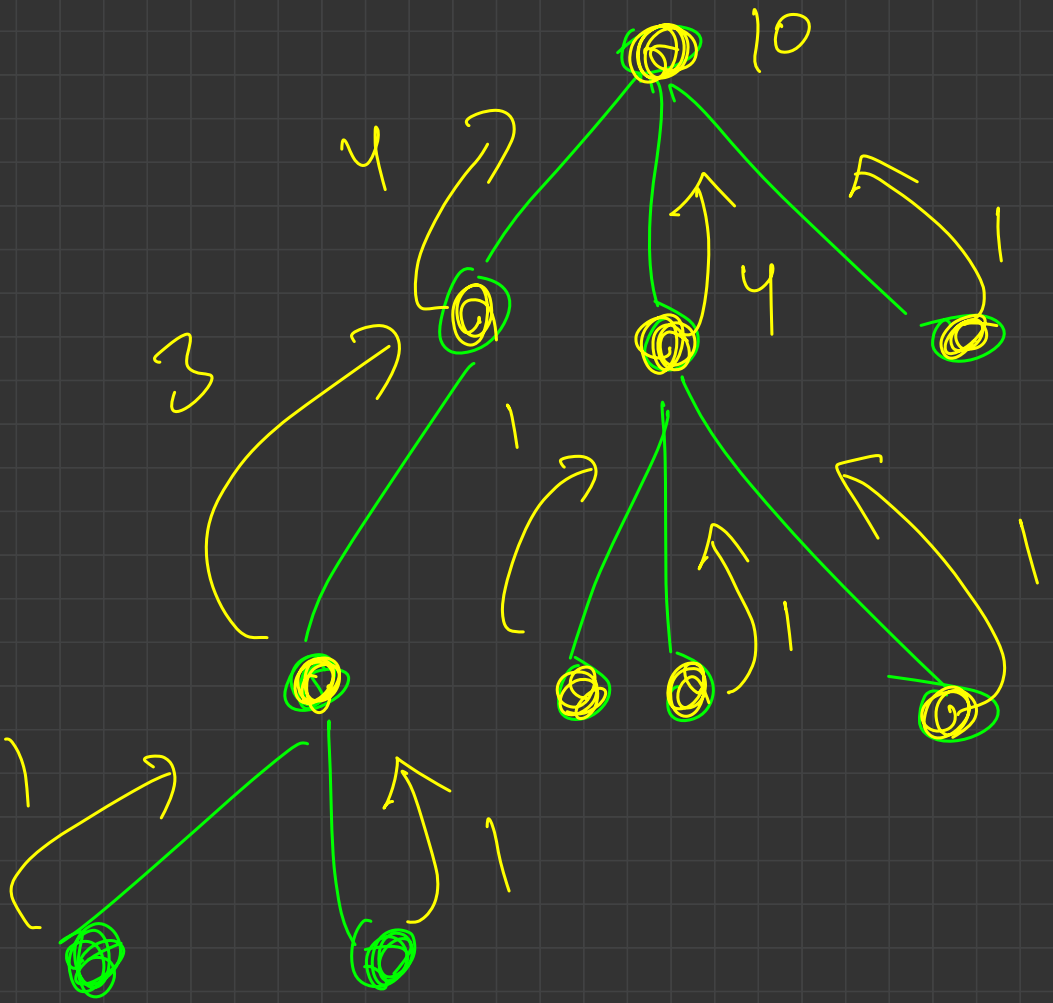
for 1 based indexing

Time Complexity: O(N)

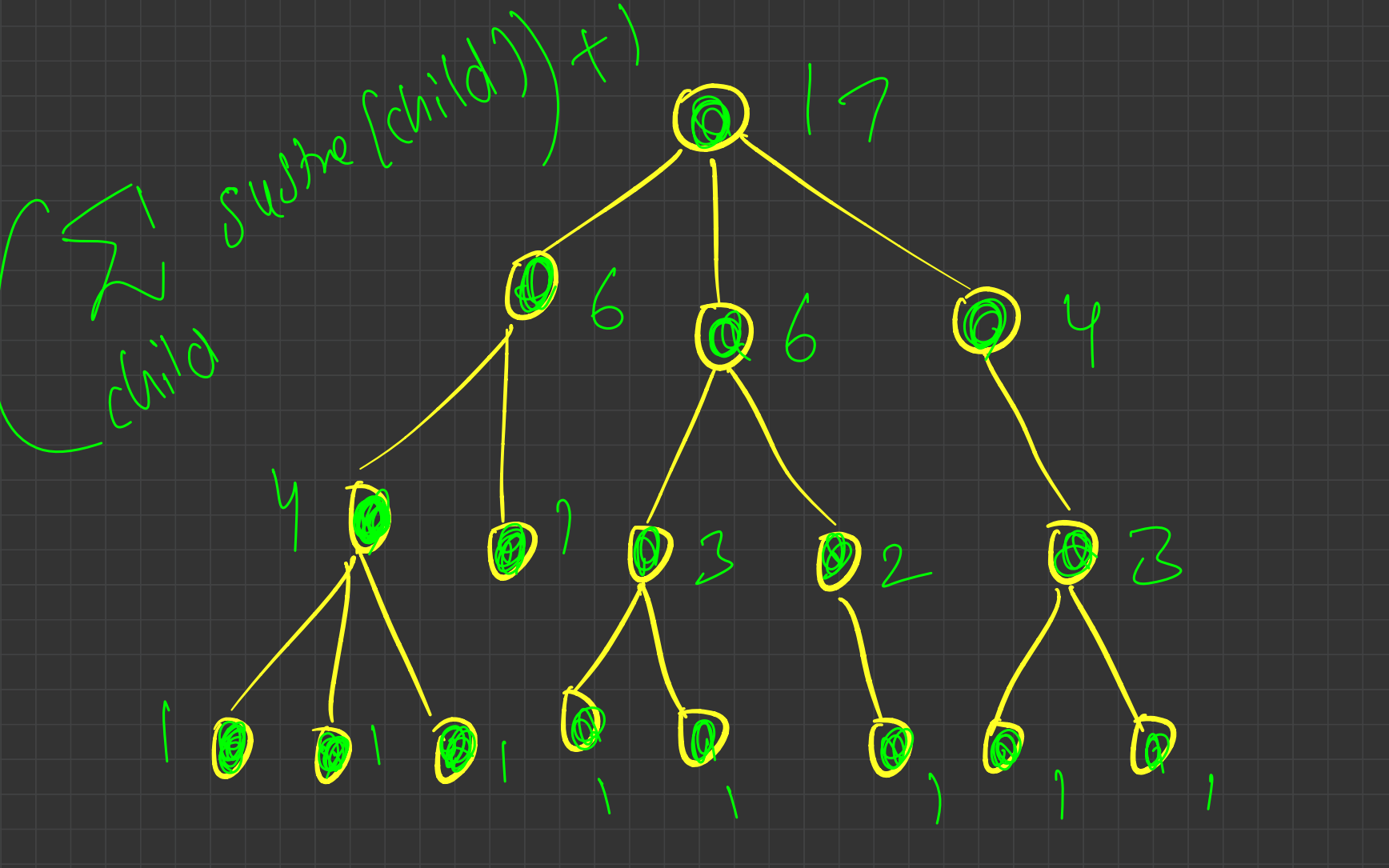
Problems on Traversals

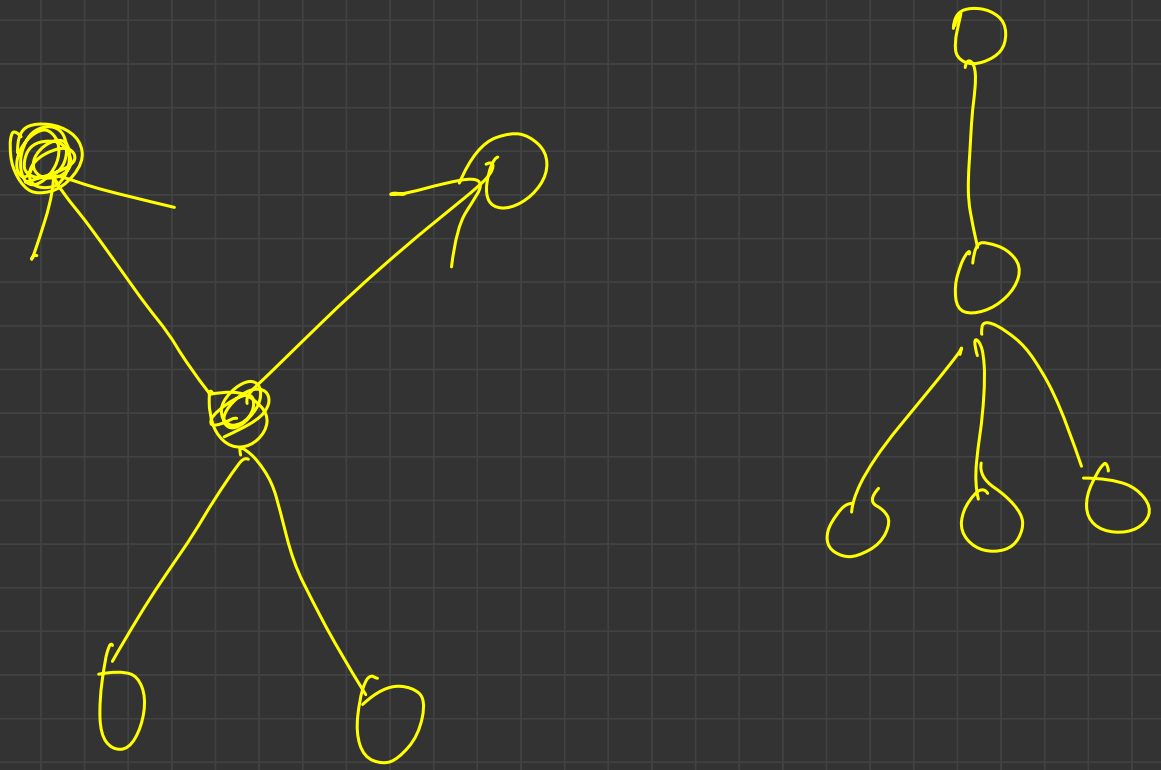
- Level of each node
- Storing the parent of each node
- Finding the number of children of each node
- Finding the subtree size of each node, [redacted]
- Finding the diameter

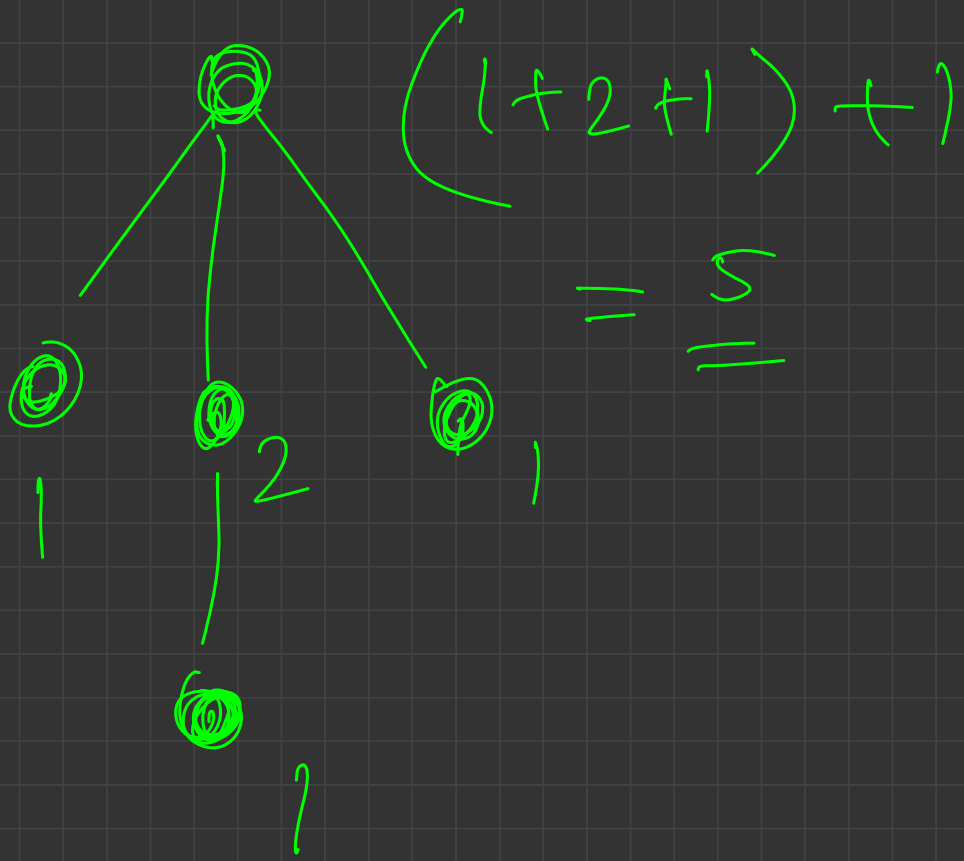


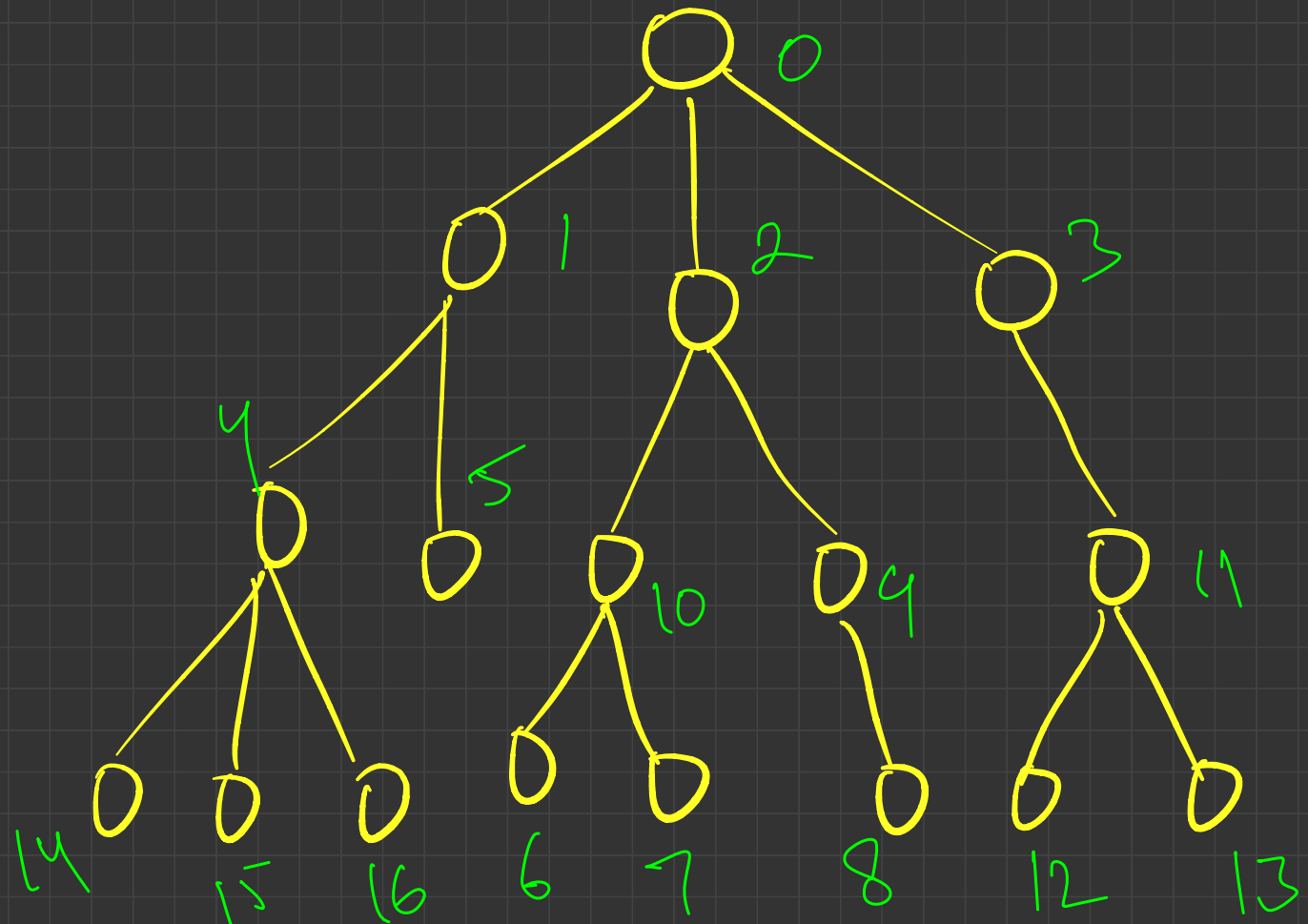

10
4
4
1
3
1
1
1
1
1

$$\left(\sum_{child} size[child]\right) + 1$$






(1+2+1)+1
= 5
1
2
1
1


0
1
2
3
4
5
10
9
11
14
15
16
6
7
8
12
13