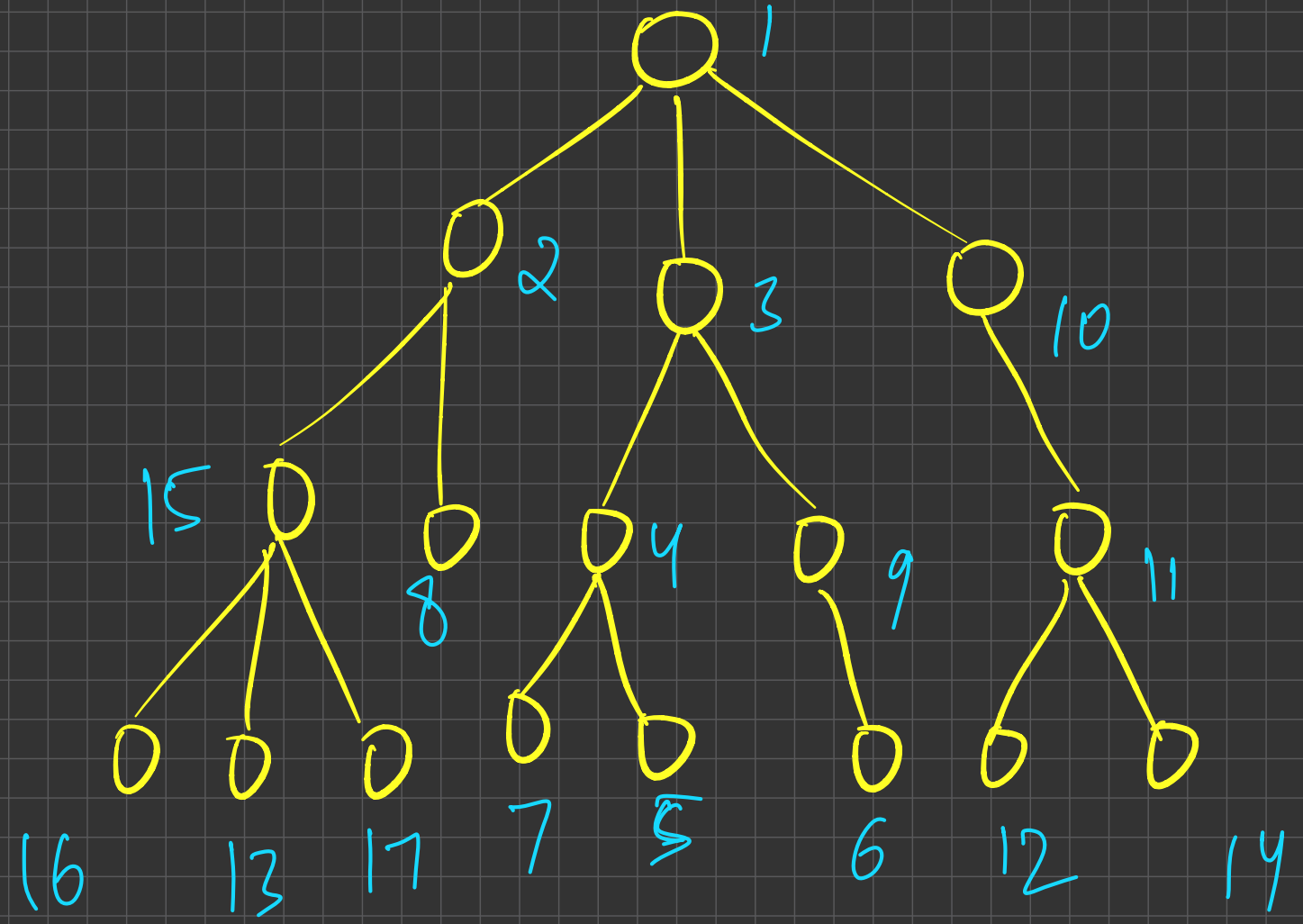
Trees 2

— Priyansh Agarwal

In Trees so far

- Properties
- Terms
- Traversals
  - DFS
  - BFS
- Some applications
  - Level of nodes
  - Parent " "
  - Children of nodes

Subtree size of each node

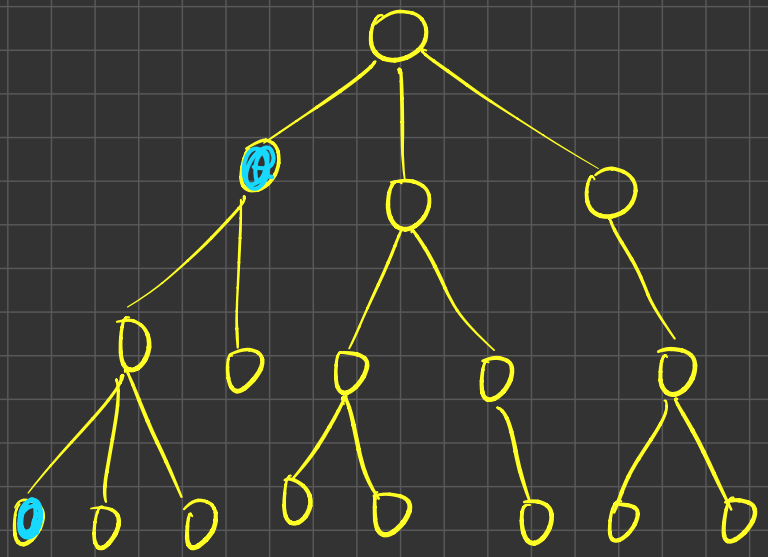


$$sub(x) = \left[\sum_{child} sub(child) \right] + 1$$

```
void dfs(int curr, vector<vector<int>> adj
         , int parent, vector<int> sub)
{
    for(int neighbour : adj[curr]) {
        if neighbour != parent {
            dfs(neighbour, adj, curr, sub)
        }
        sub[curr] += sub[neighbour]
```

```
}
sub[cur]++;
}
```

leaf in each subtree



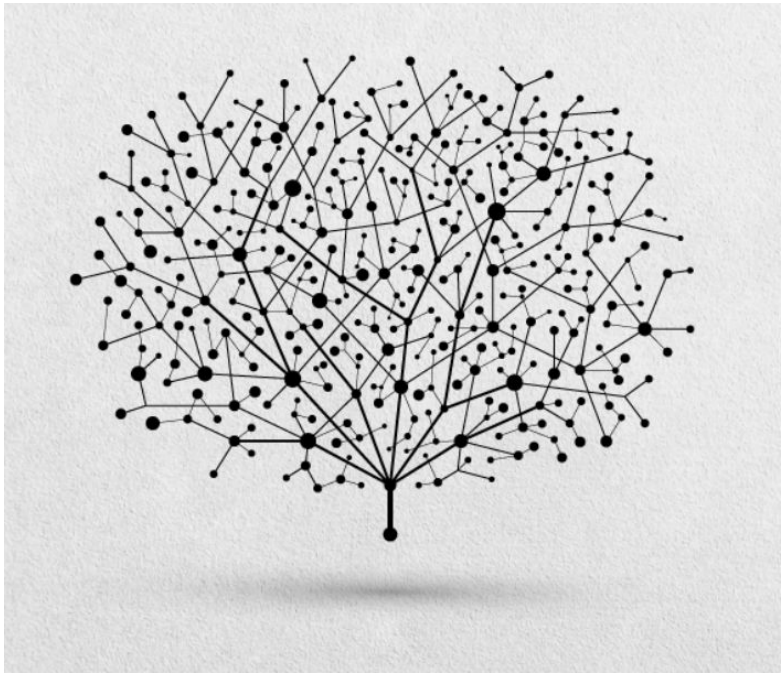
if current = leaf node

$$leafs[current] = 1$$

else

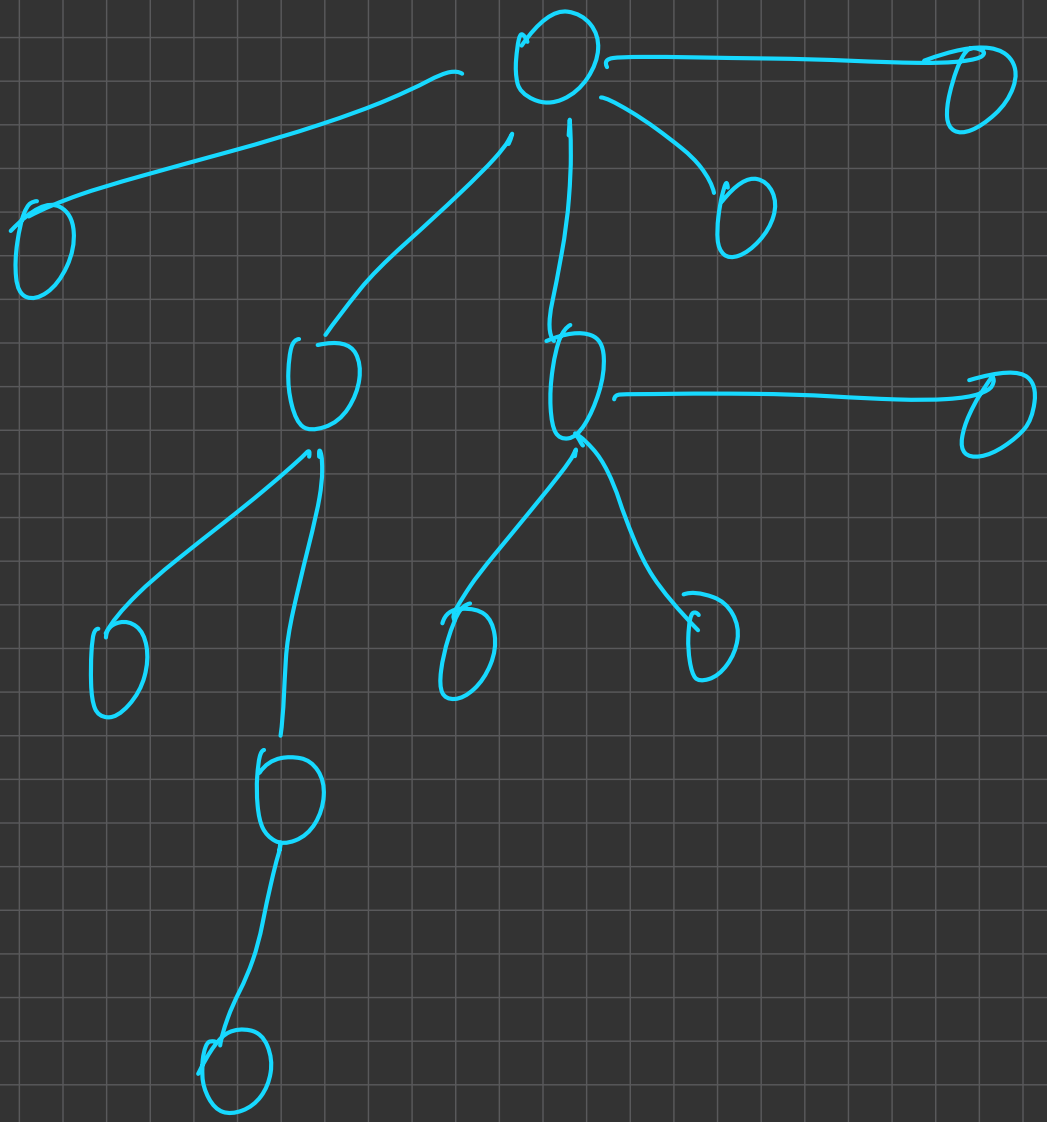
$$leafs[current] = \sum_{child} leafs(child)$$

Diameter of a Tree



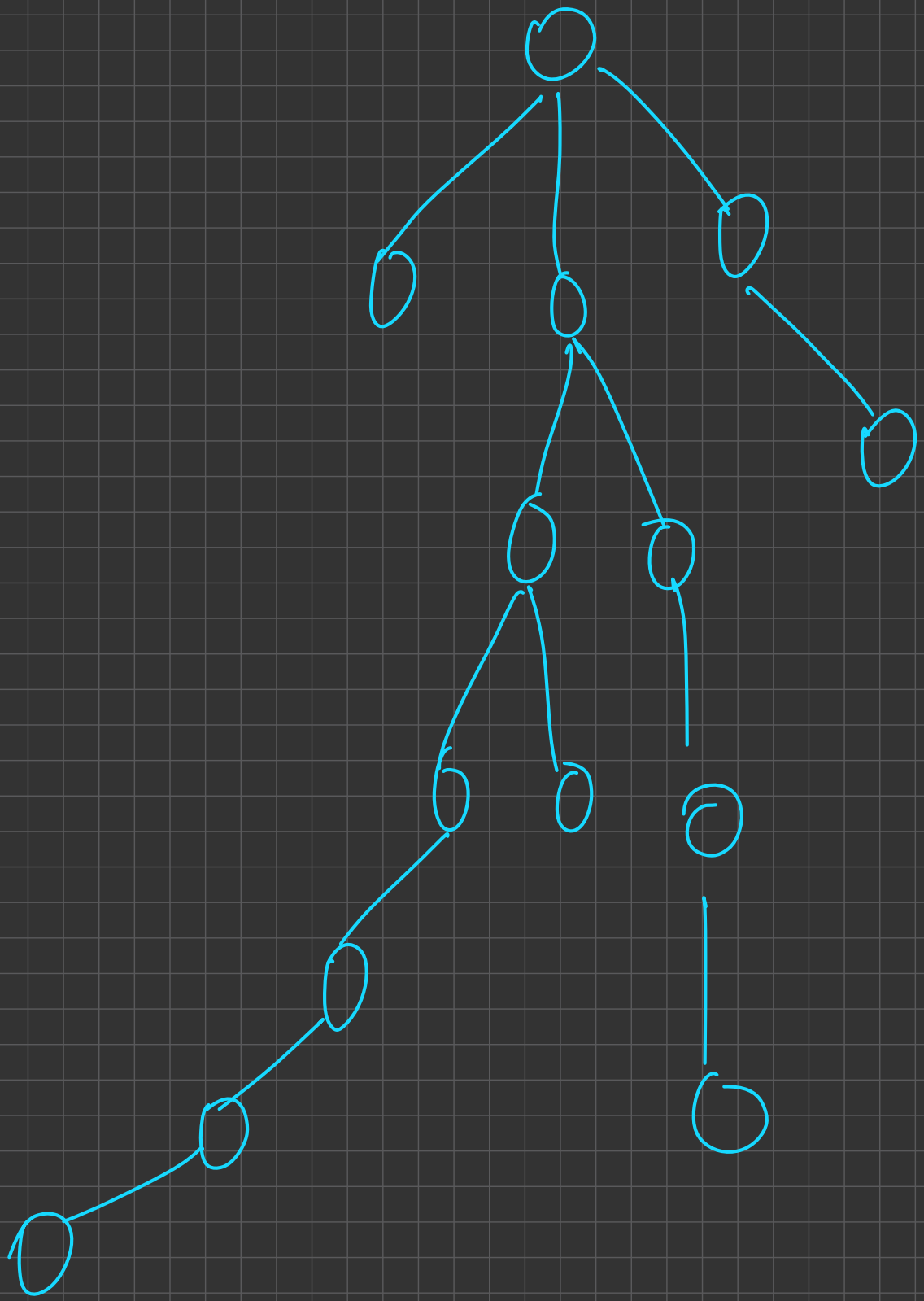
Diameter of a tree = Maximum distance between any 2 nodes in the tree

Problem: Link

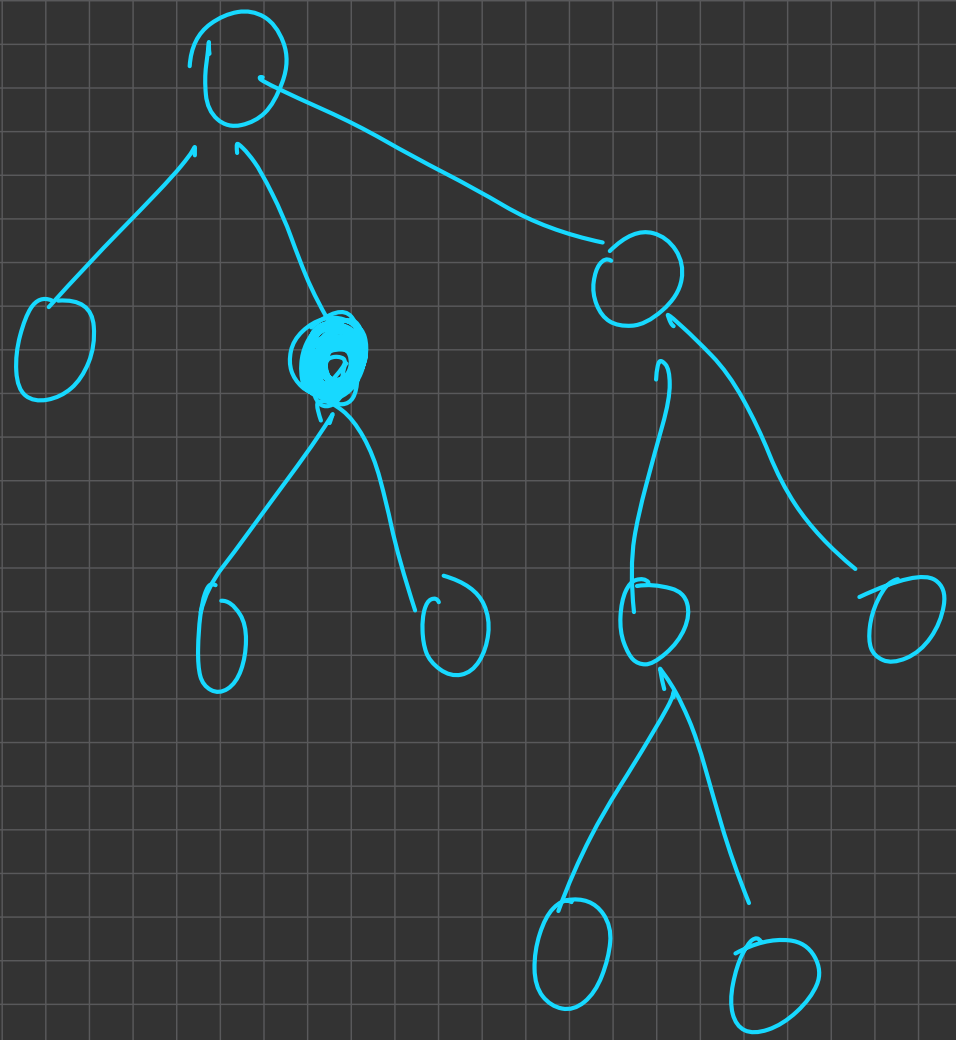


Brute force. Root the tree at every node and find out the height of the tree,

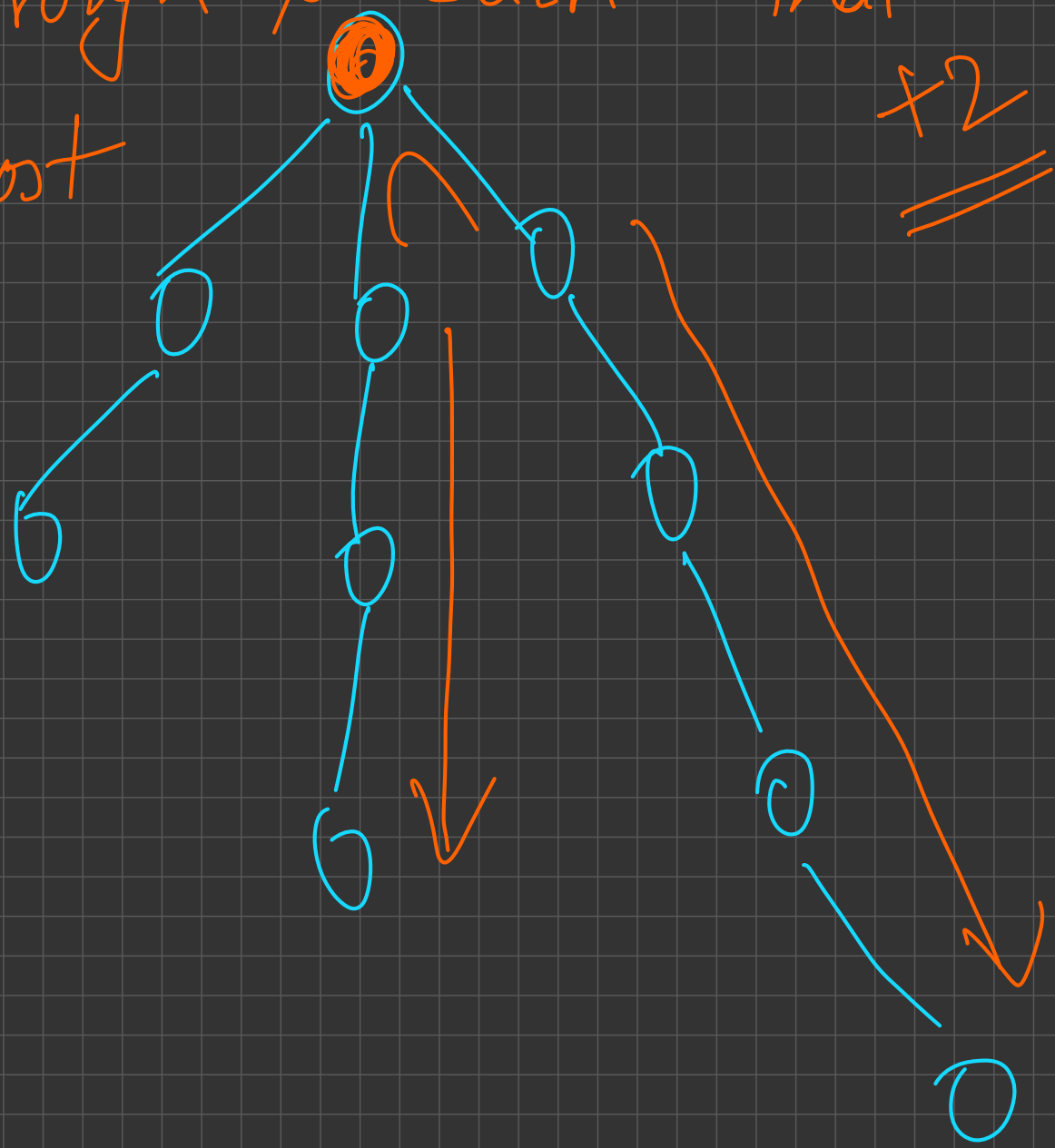
Maximum height encountered = Diameter

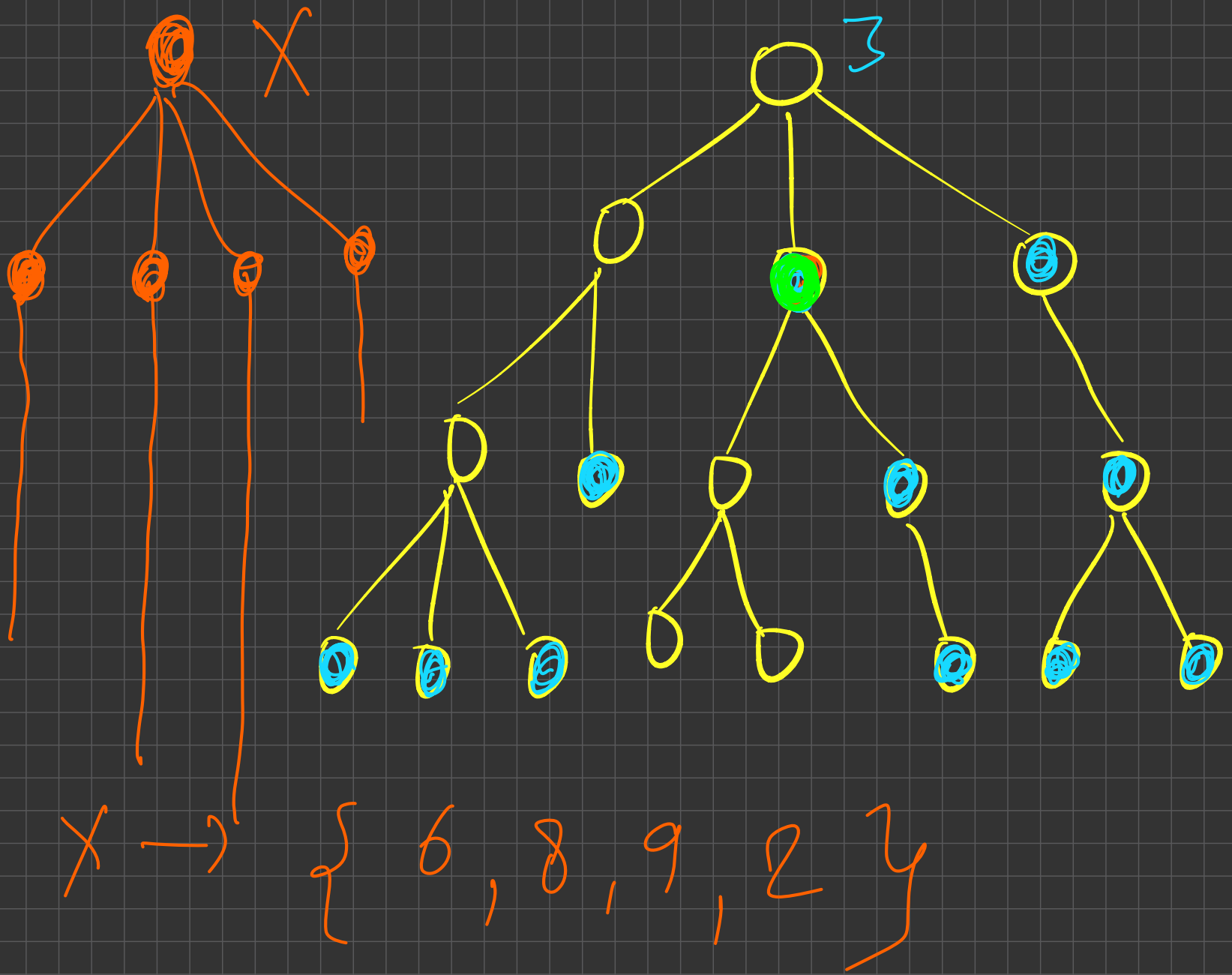


Path Property



for a node x, find the longest path passing through x such that x is the topmost node in that path



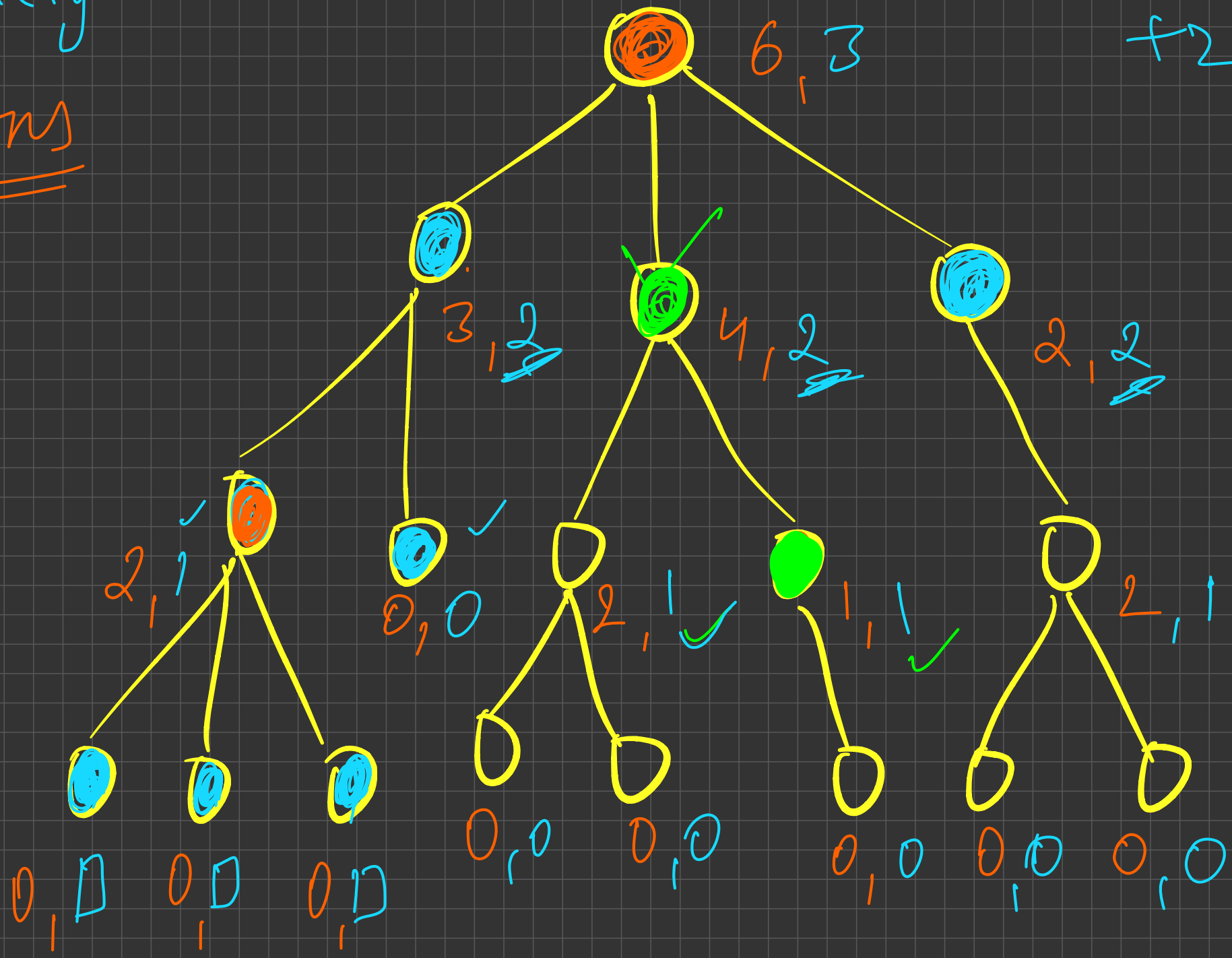


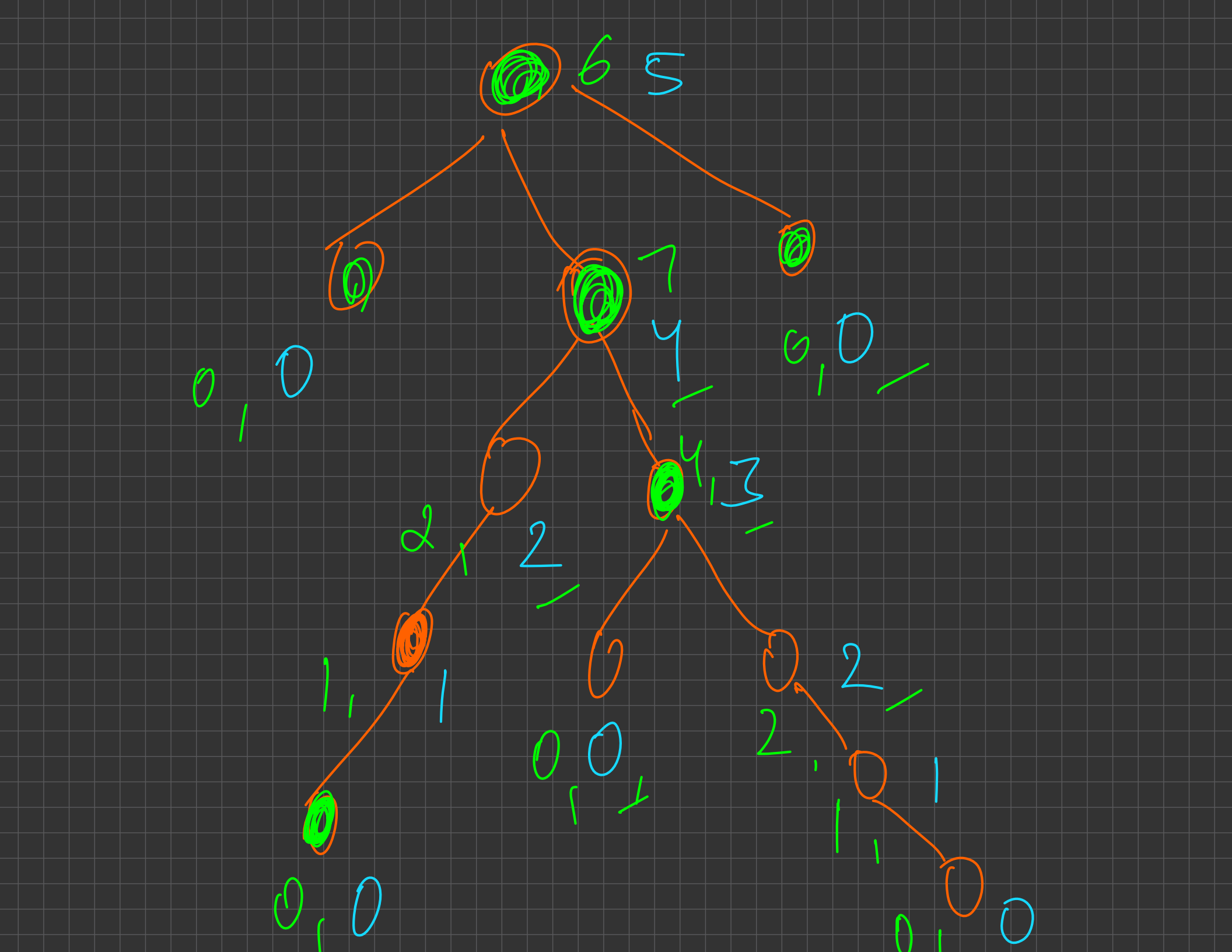
X → { 6, 8, 9, 2 }

Height

Ans

max heigh + s.max



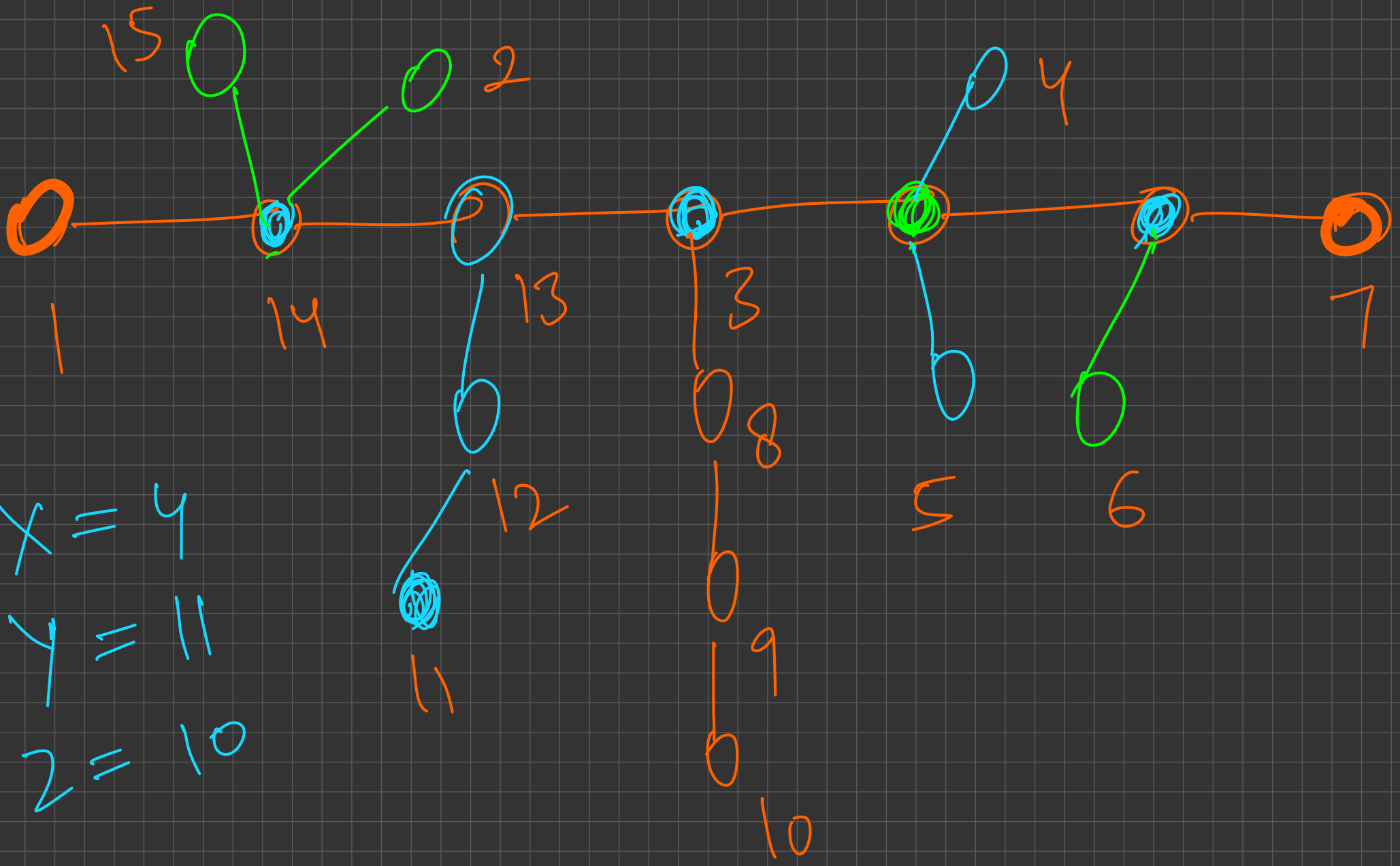


Diameter Approach 2

① Pick a random node and find the farthest node from X (call it Y) (X) O(n)

② find the farthest node from Y and call it Z O(n)

[then dist(Y, Z) = diameter]


15
2
4
1
14
13
3
7
12
8
5
6
11
9
10
X = 4
Y = 11
2 = 10

Pick node X, do dfs from X

find Y which is farthest

do dfs from Y, find Z

which is farthest. Level of

Z = diameter

Ancestor - Descendant Problem

Given a rooted tree with N nodes and Q queries.

For each query of the form X, Y check whether X is an ancestor of Y or not

N <= 1e5, Q <= 1e5

$N \leq 10^5$

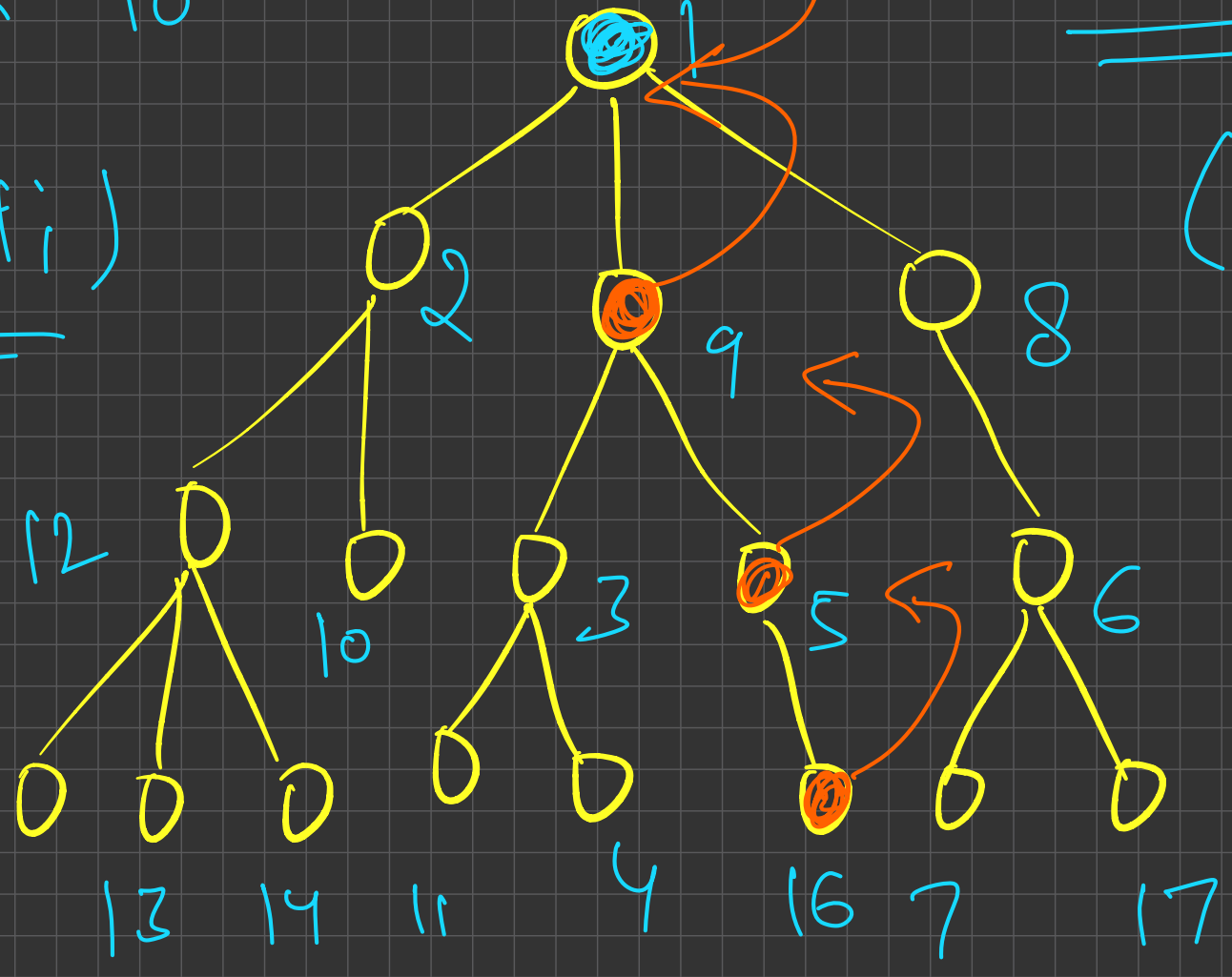
$0 \leq 10^5$

rooted at node 1

$(8, 17) \rightarrow$ Yes

(x_i, y_i)

$(3, 16) \rightarrow$ No



$(9, 16)$

if level(x) > level(y) ==

Can x be the ancestor of y

—> no

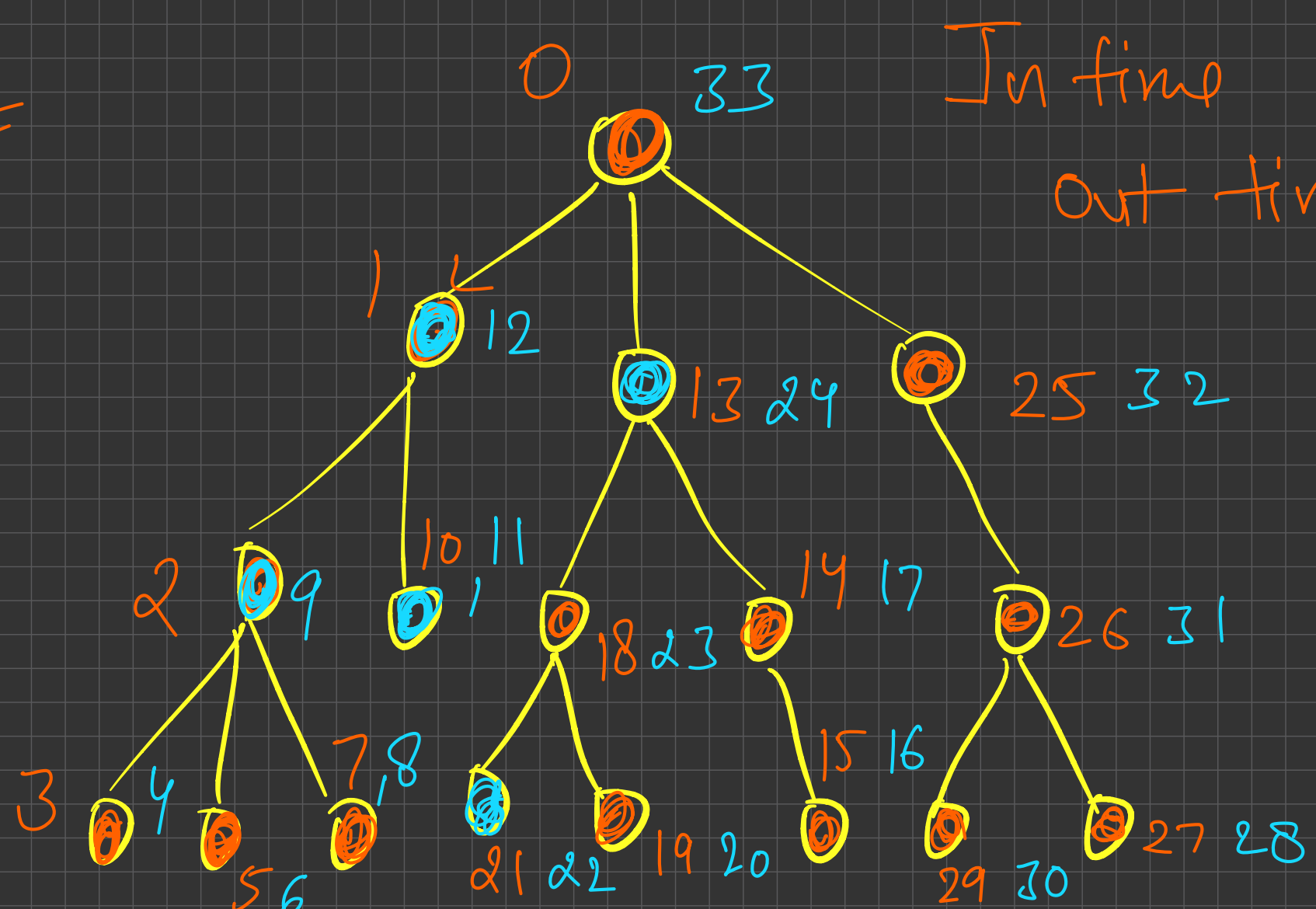
if level(x) < level(y)

Store two parent of every node

$y \rightarrow p(y) \rightarrow p[p(y)]$

$\longrightarrow$ root, X

$O(n)$



In time(x) = first time when you reach x

Out time(x) = time when you have explored the subtree and you backtrack

if X is ancestor of Y

intime(X) < intime(Y) < outtime(Y)

< outtime(X)

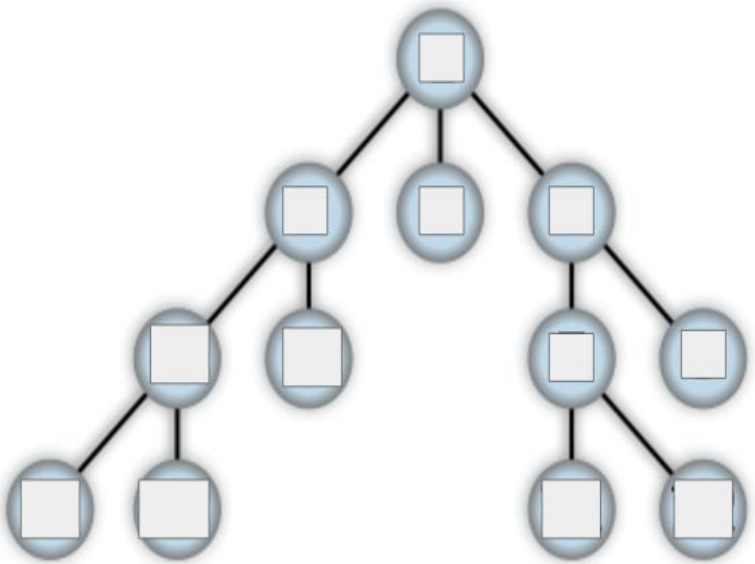
In - Out Time trick

Do a DFS traversal.

Store the following information for each node:

First visited time = In time

Last visited time = out time



Can you solve the ancestor descendant problem now?

In - Out Time trick

Solving the ancestor - descendant problem:

If X is an ancestor of Y

$$X_{\text{in time}} < Y_{\text{in time}} < Y_{\text{out time}} < X_{\text{out time}}$$