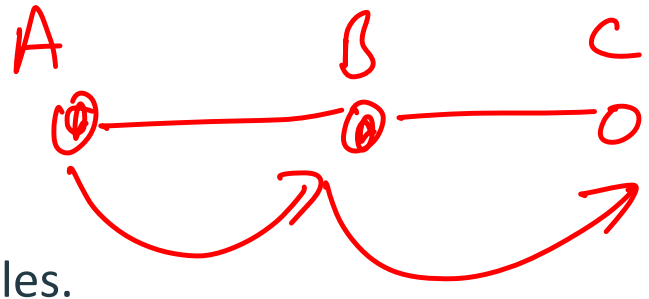
Trees 1

What is a Tree



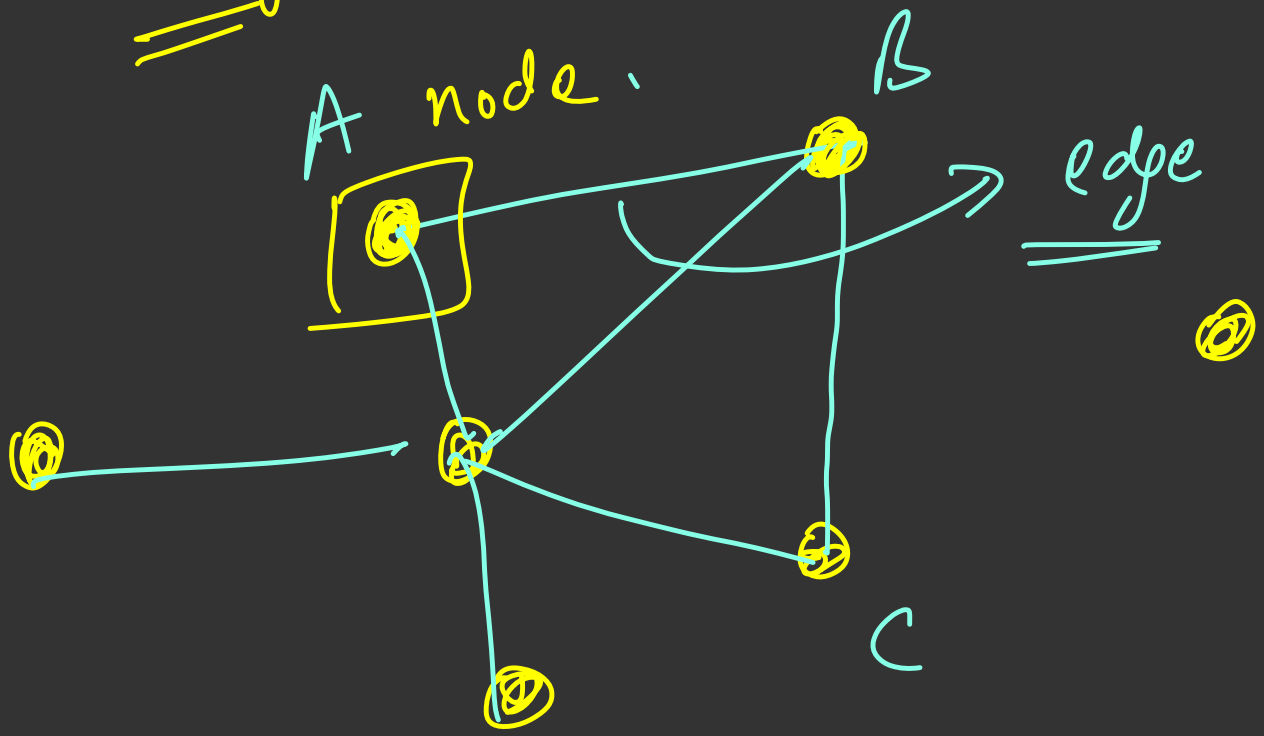
A connected graph of N nodes without any cycles.

What is a graph?

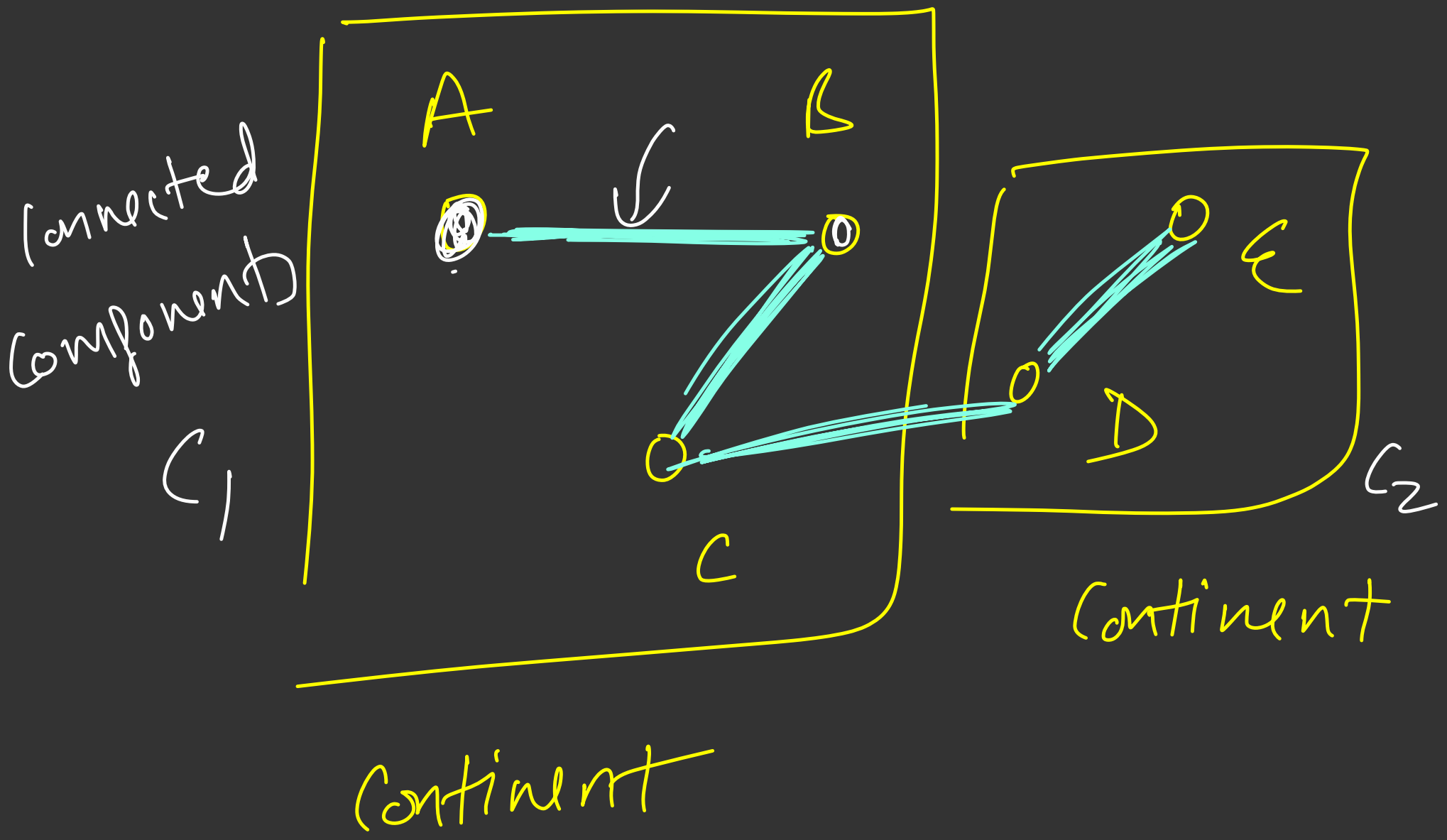
- Imagine it like the Earth
- Contains a bunch of countries connected via roads
- A continent is a group of countries directly or indirectly connected to each other
- Some countries might be in different continents -> disconnected

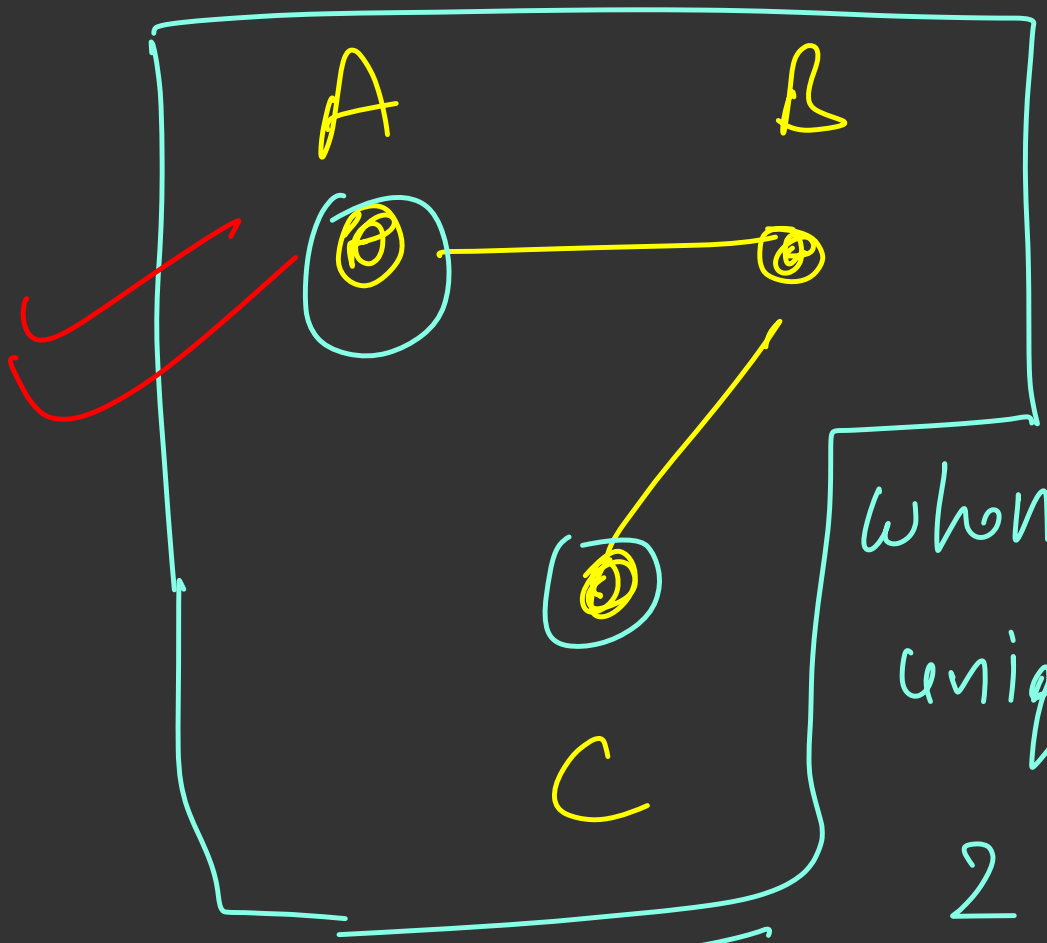
A tree is one such continent with a unique path b/w any 2 countries

Node, Edge



edge connects 2 nodes


Connected Components
C_1
A
B
C
D
E
C_2
Continent
Continent



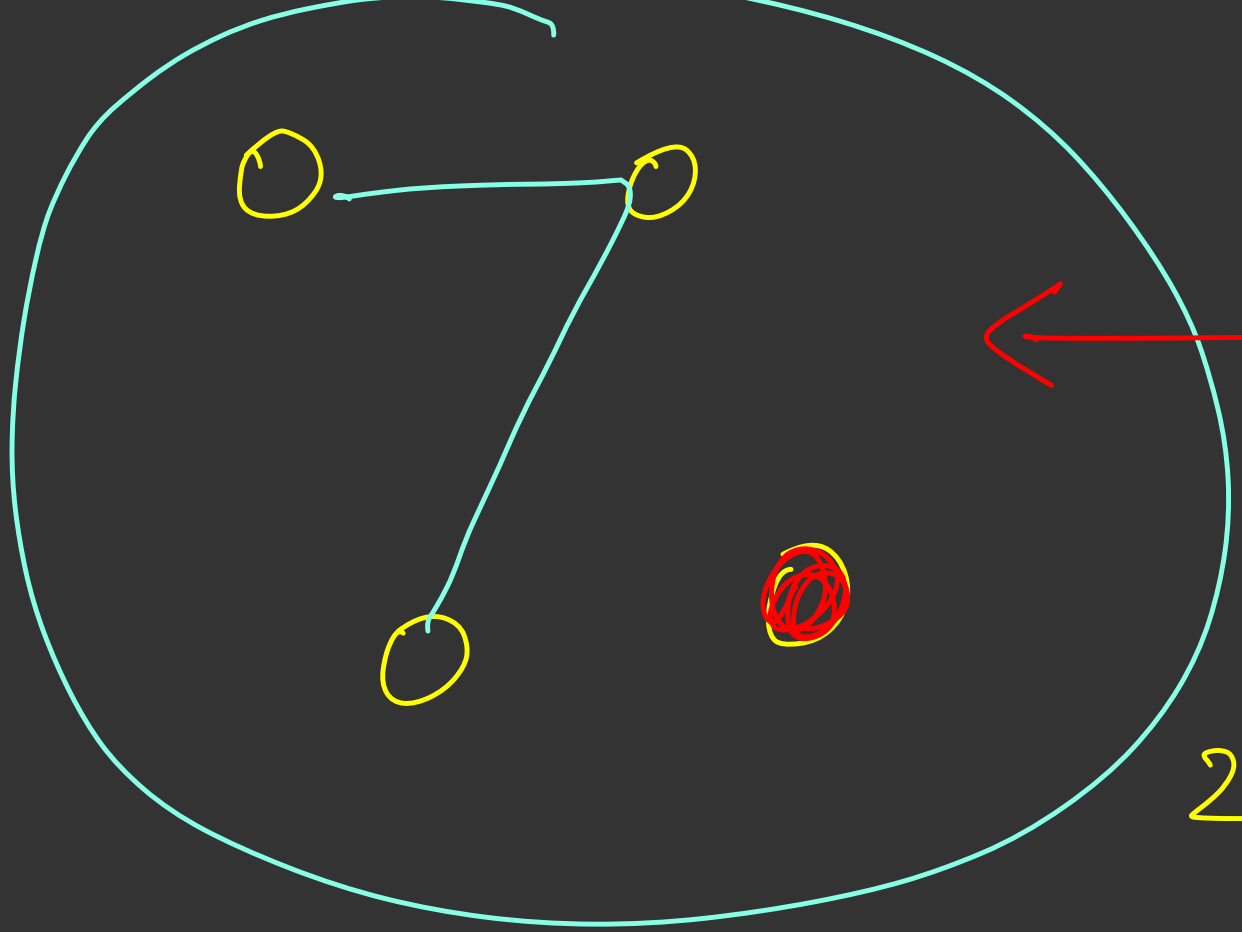
○ nodes

— edges

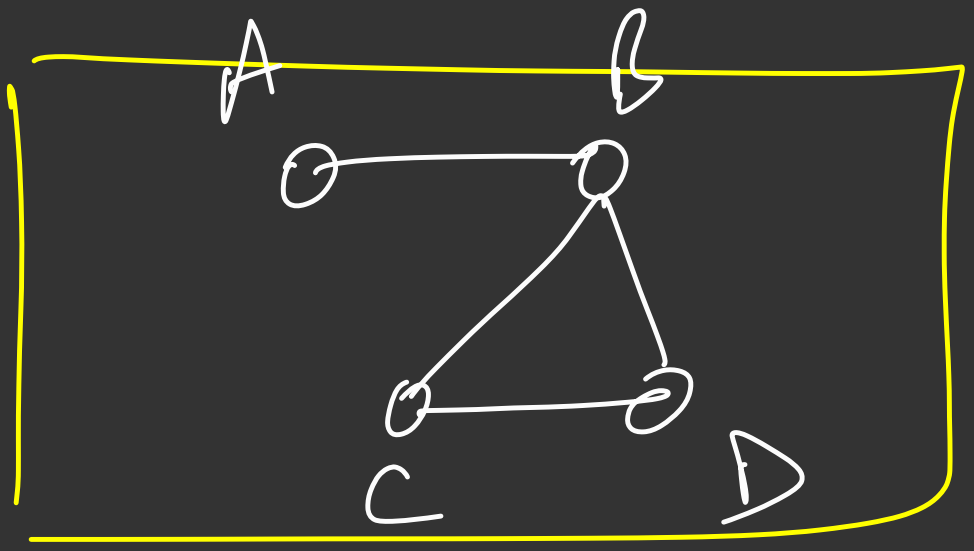
when there is a unique path b/w any 2 nodes

↳ Tree

↳ there must exist exactly 1 path b/w any 2 nodes



connected & it must have exactly 1 path b/w any 2 nodes



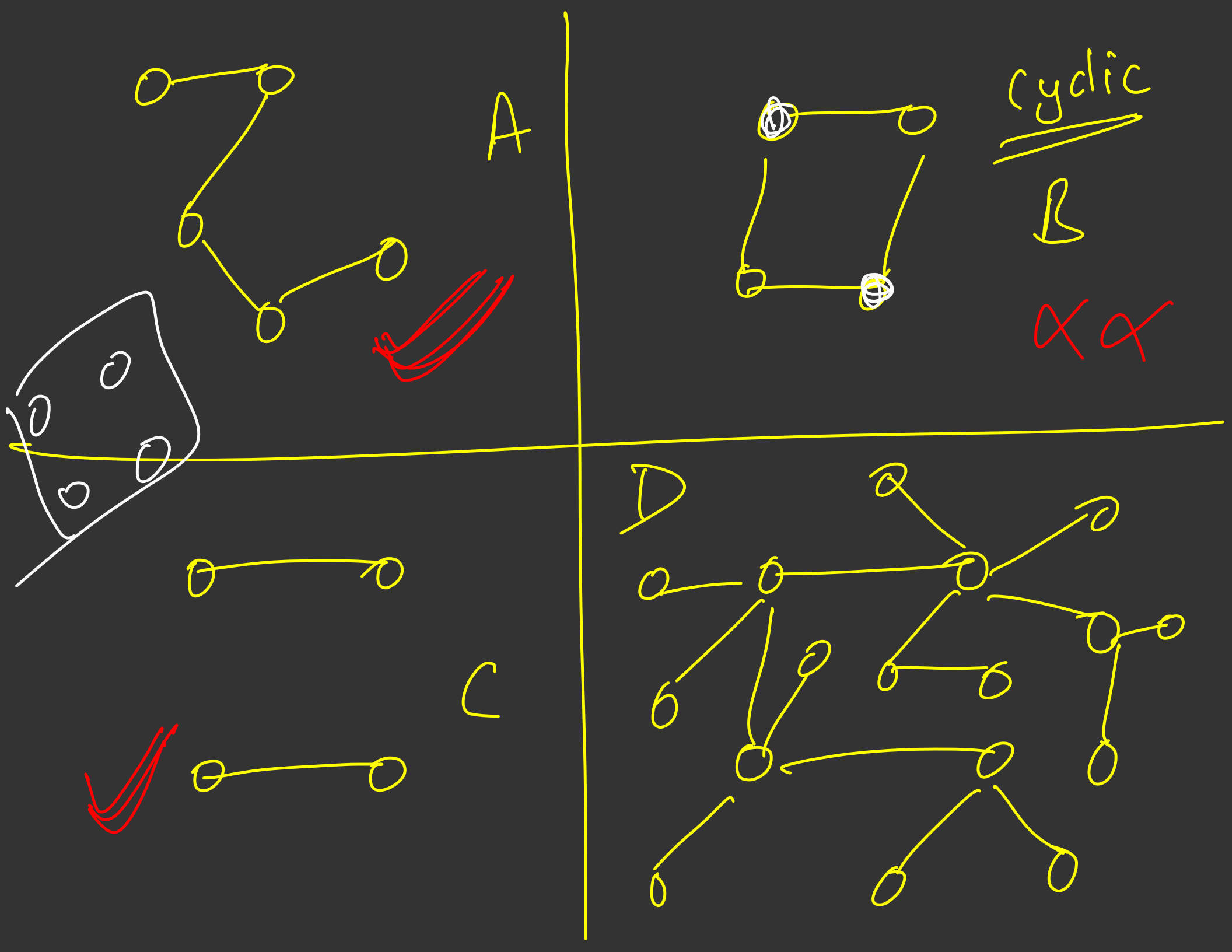
What is a Tree

Differentiate b/w a Tree and a Graph from examples

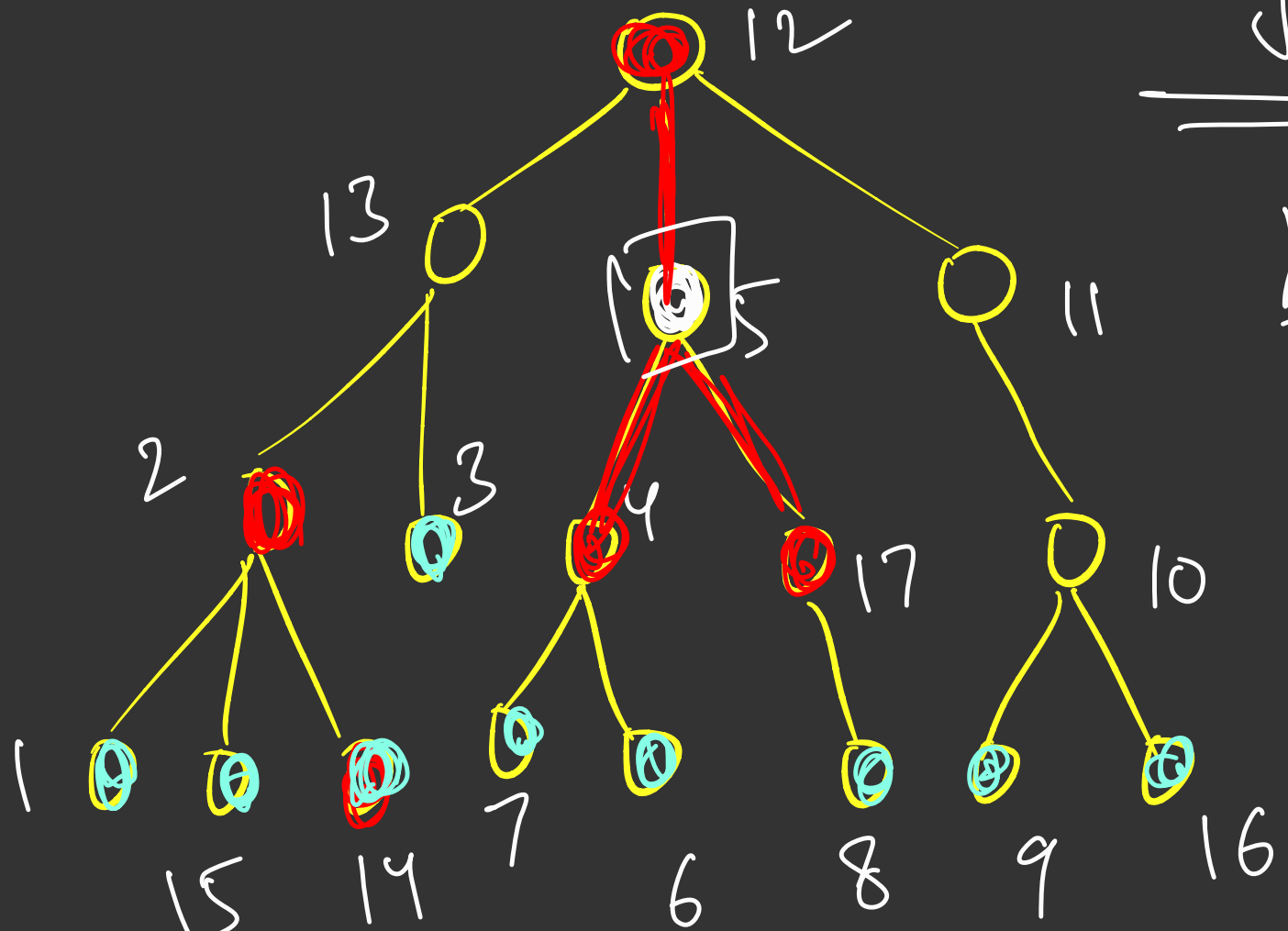
- One Note Illustrations

Some Common Terms

- Neighbour
- Degree
- Leaf and non-leaf Nodes (aka Internal Nodes)
- Diameter (can be non-unique)


A
Cyclic
B
C
D

Neighbour



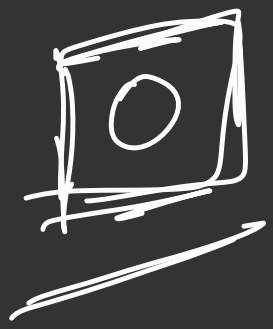
Degree

Neighbour of some node X

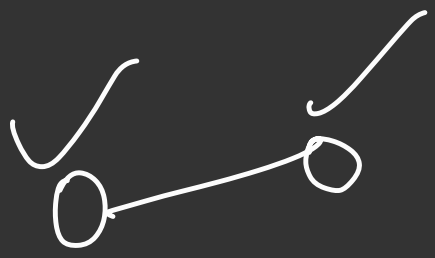
Degree of node X = |No. of neigh of X|

✓ leaf node → any node with degree = 1

✓ Internal node → any node which is a not a leaf



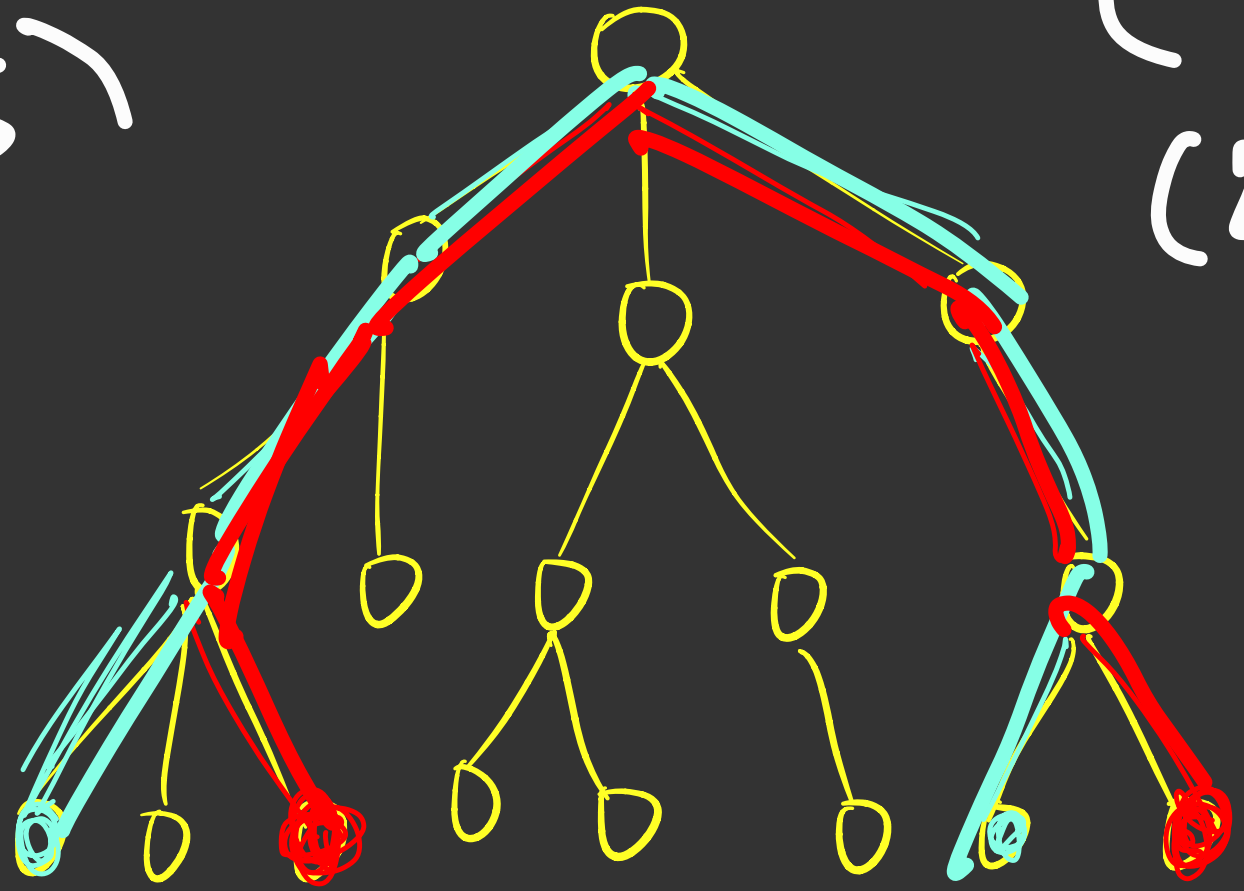
Connected acyclic



Diameter of tree = the maximum distance b/w any 2 nodes

(1,7)
(1,5)

(2,6)
(2,5)



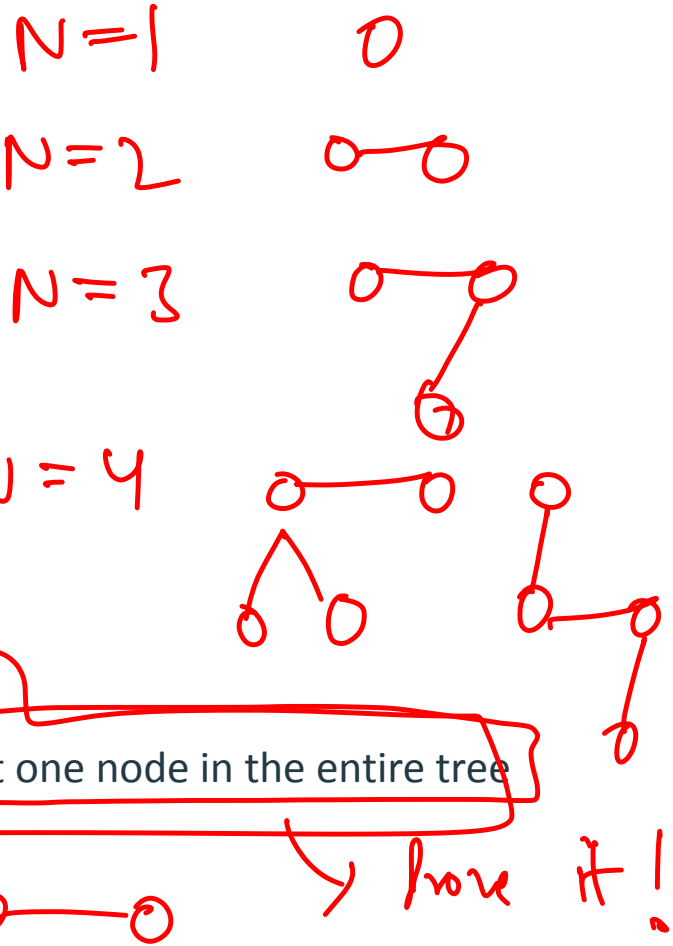
1 2 3 7 6 4 5

Properties 1

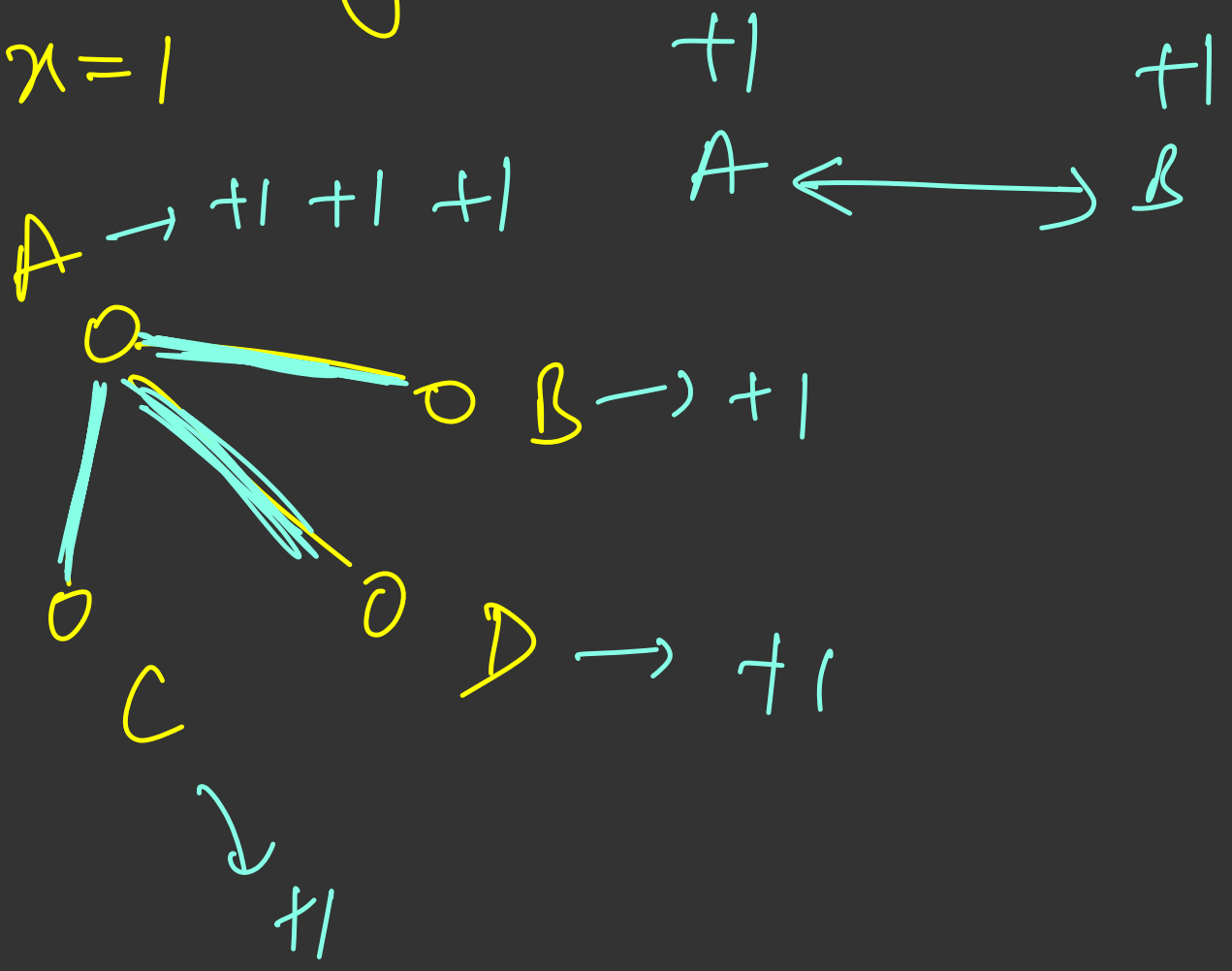
- Number of Edges in a Tree for N nodes
  - N - 1
- Number of paths between 2 nodes
  - 1
- Sum of Degree of all nodes
  - 2 * (N - 1)
- Can there be less than 2 leaf nodes in a Tree
  - No, except for the case when there is just one node in the entire tree

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) $= 2E$
- 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

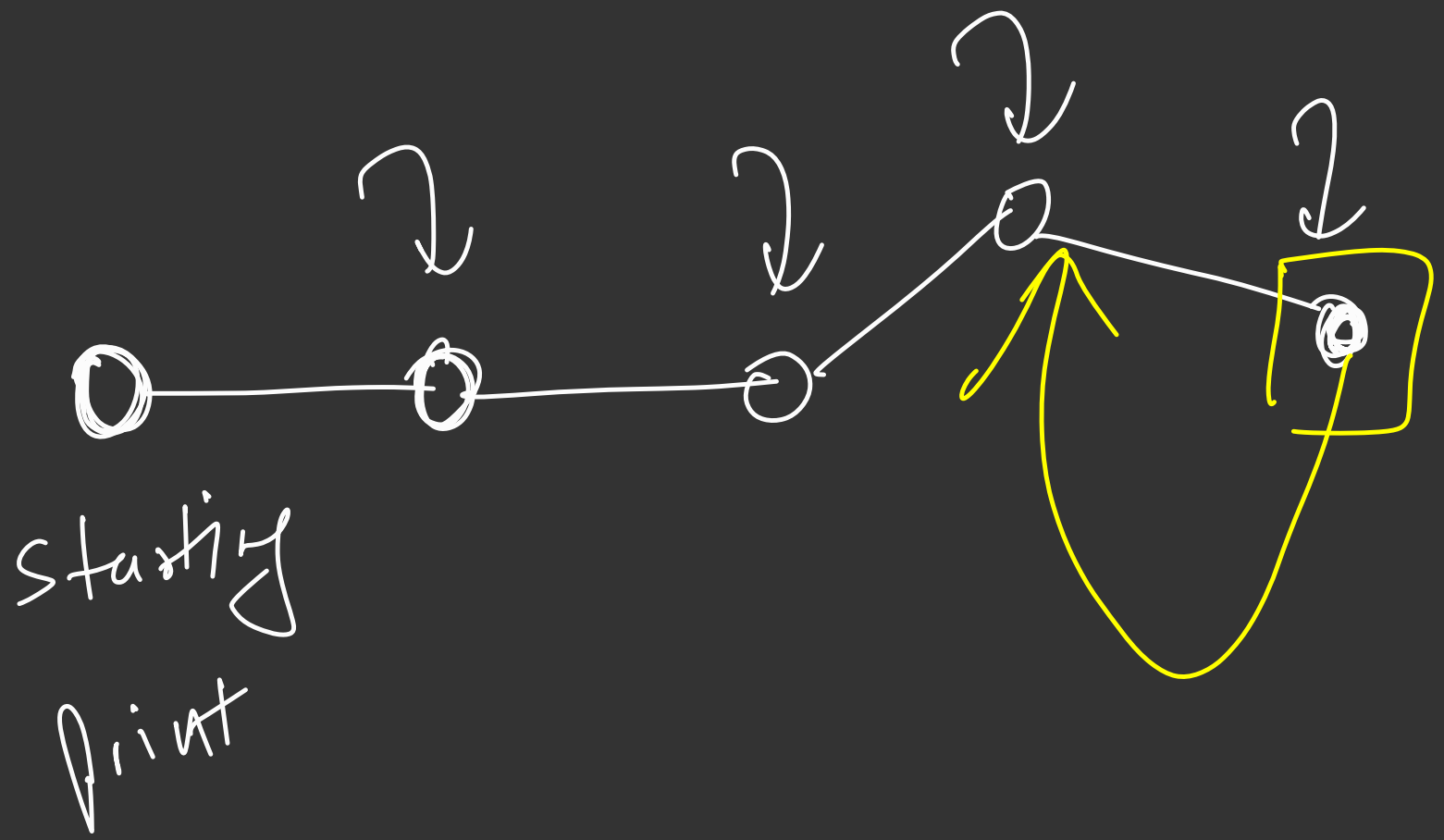


$$\sum_{x=1}^{n} \deg(x) = 2 \times (n-1)$$



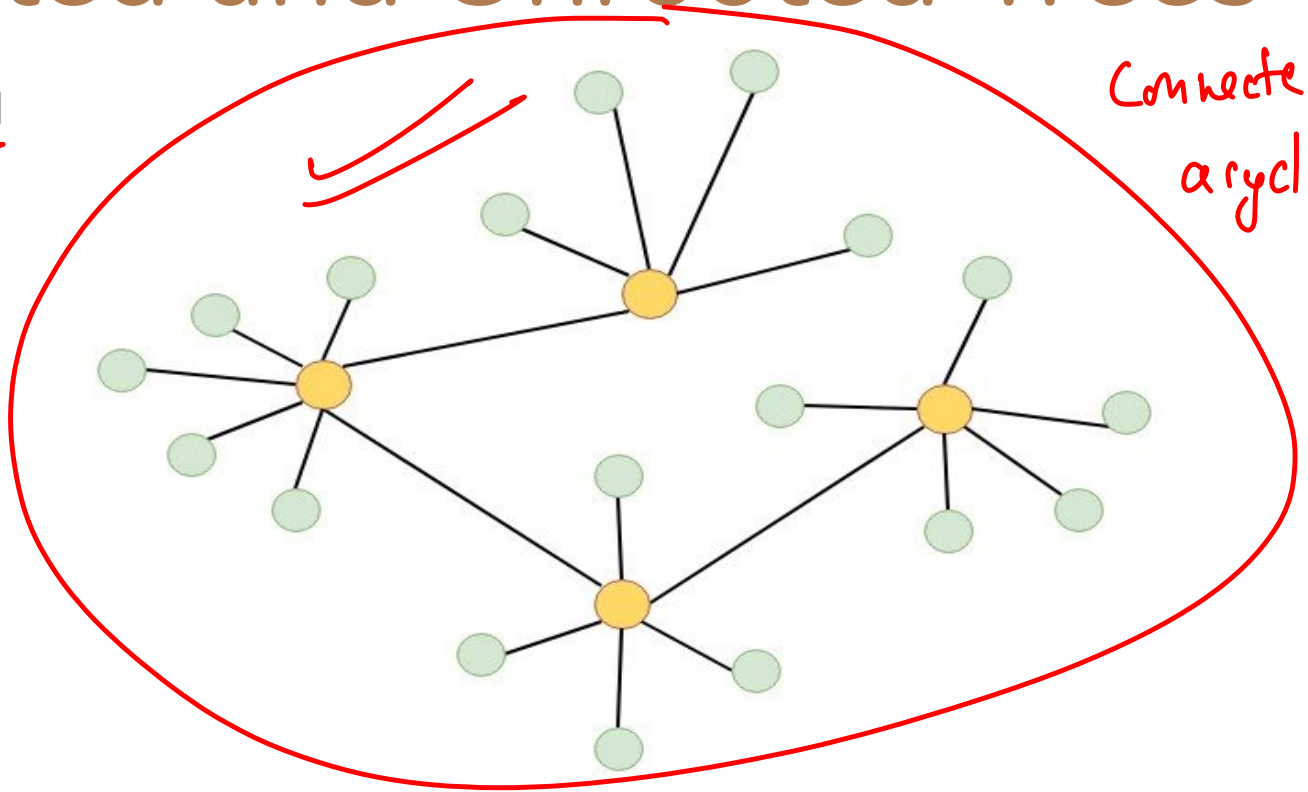
Can a Tree with $N \geq 2$ have less than 2 leaf nodes

$\longrightarrow$ No $N = 1$



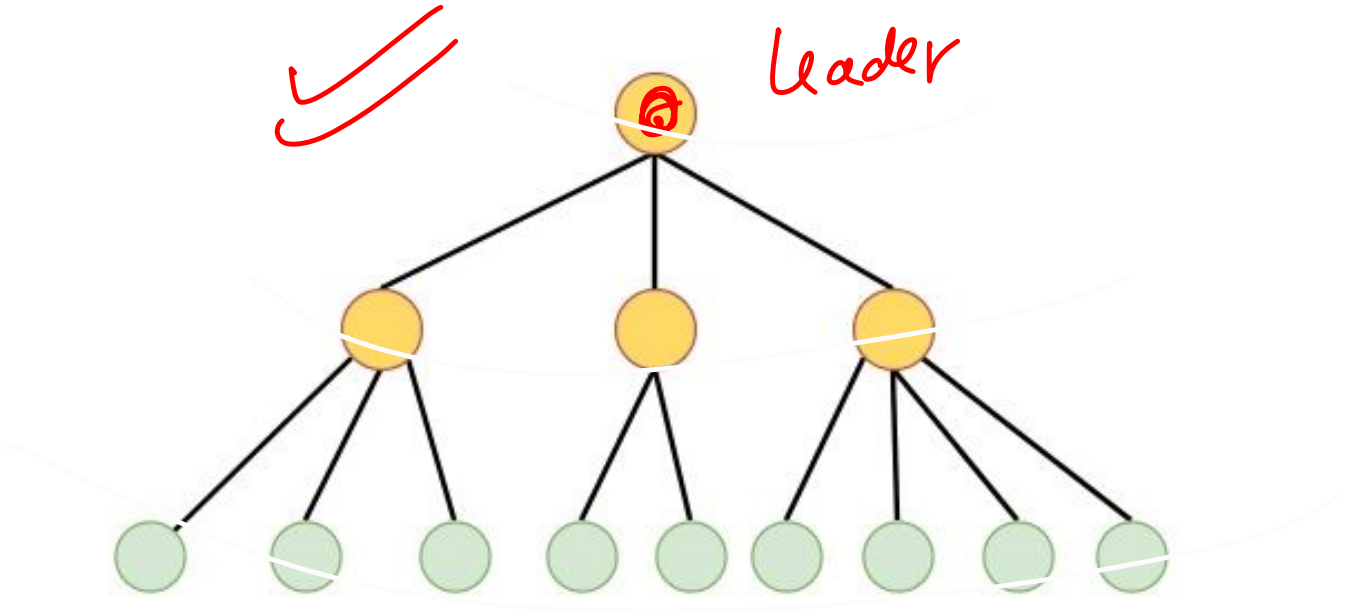
Rooted and Unrooted Trees

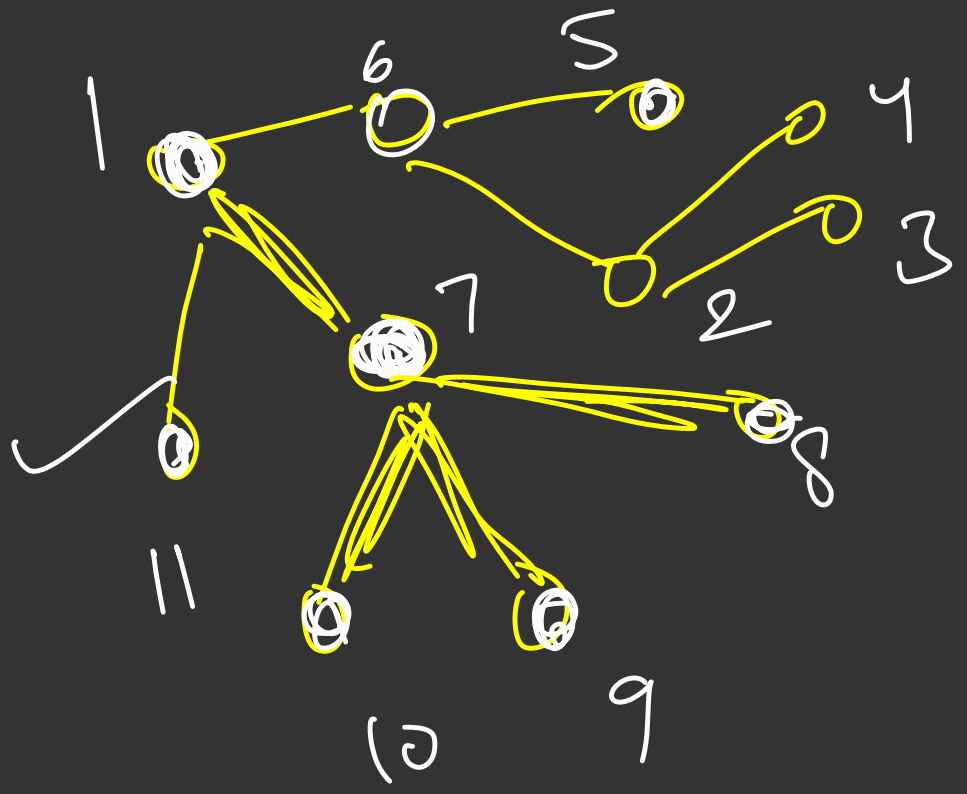
Unrooted

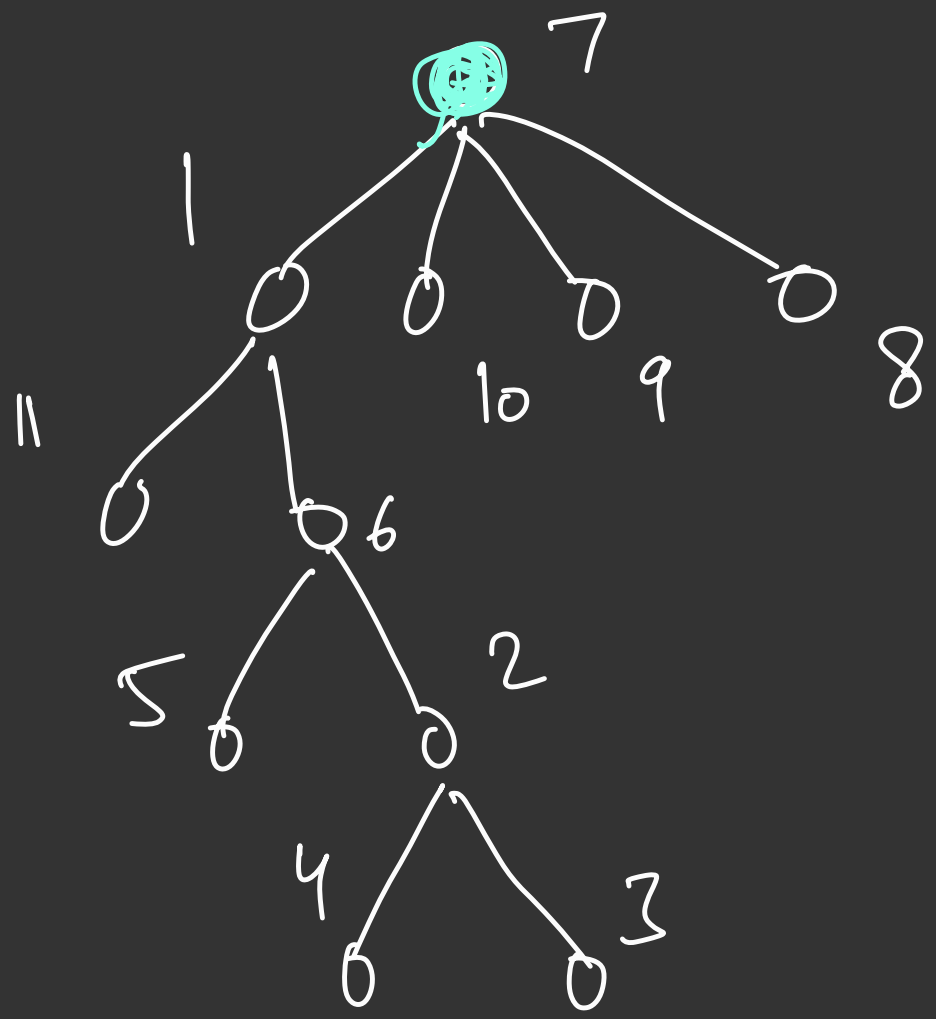


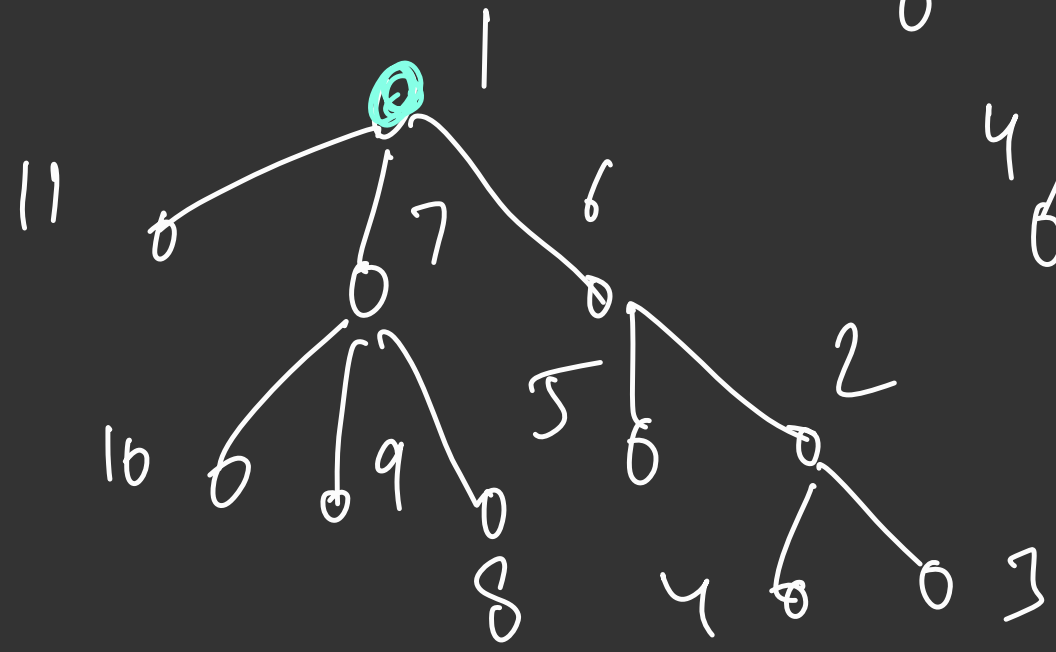
Rooted and Unrooted Trees

Rooted




1
6
5
4
3
2
7
8
11
10
9


7
1
11
10
9
8
6
5
2
4
3

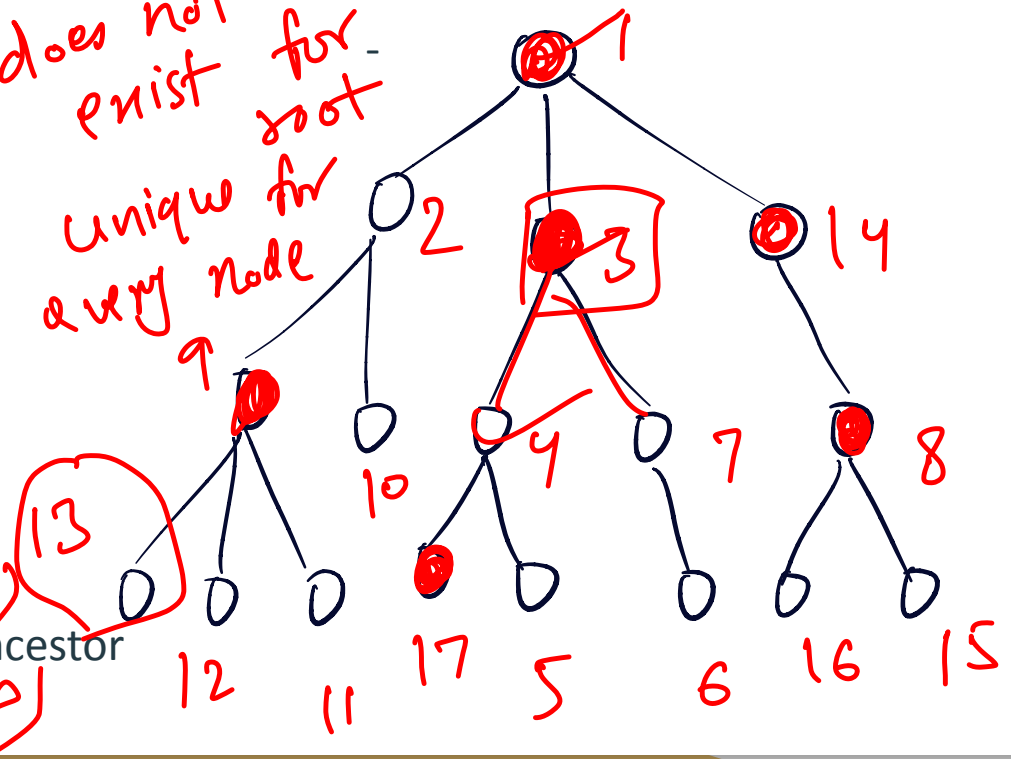

1
11
7
6
10
9
8
5
2
4
3

Some more terms (for rooted tree)

- Root (T)
- Parent (N)
- Child (N)
- Ancestor (N)
- Descendant (N)
- Level of Node (N)
- Subtree (N)
- Subtree Size (N)
- Height of Tree (T)
- Lowest Common Ancestor (2 nodes)

does not exist for root

unique for every node



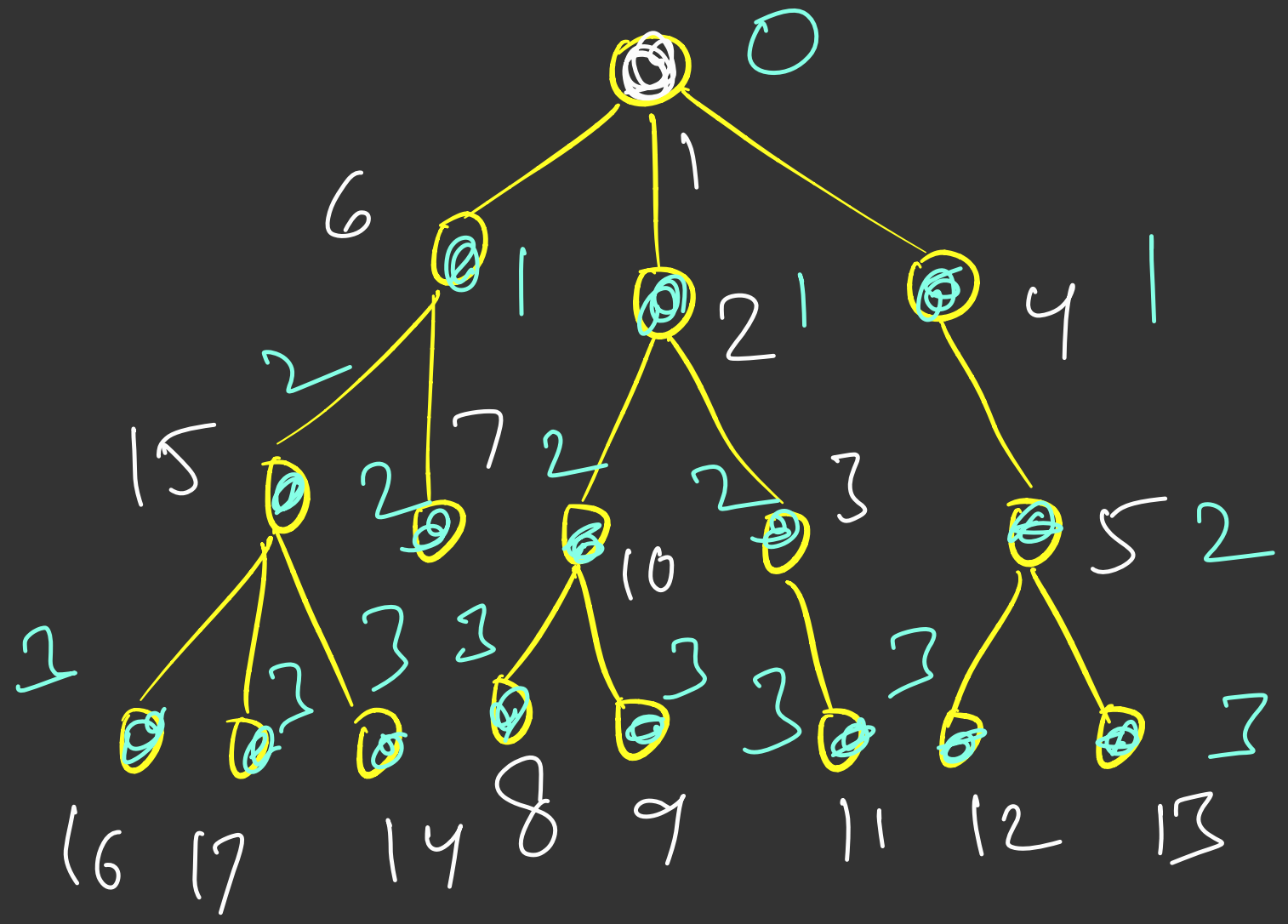
children of x are all those nodes Y such parent of $Y = x$

$x \rightarrow f(x), f(f(x)), f(f(f(x)))$

.... root

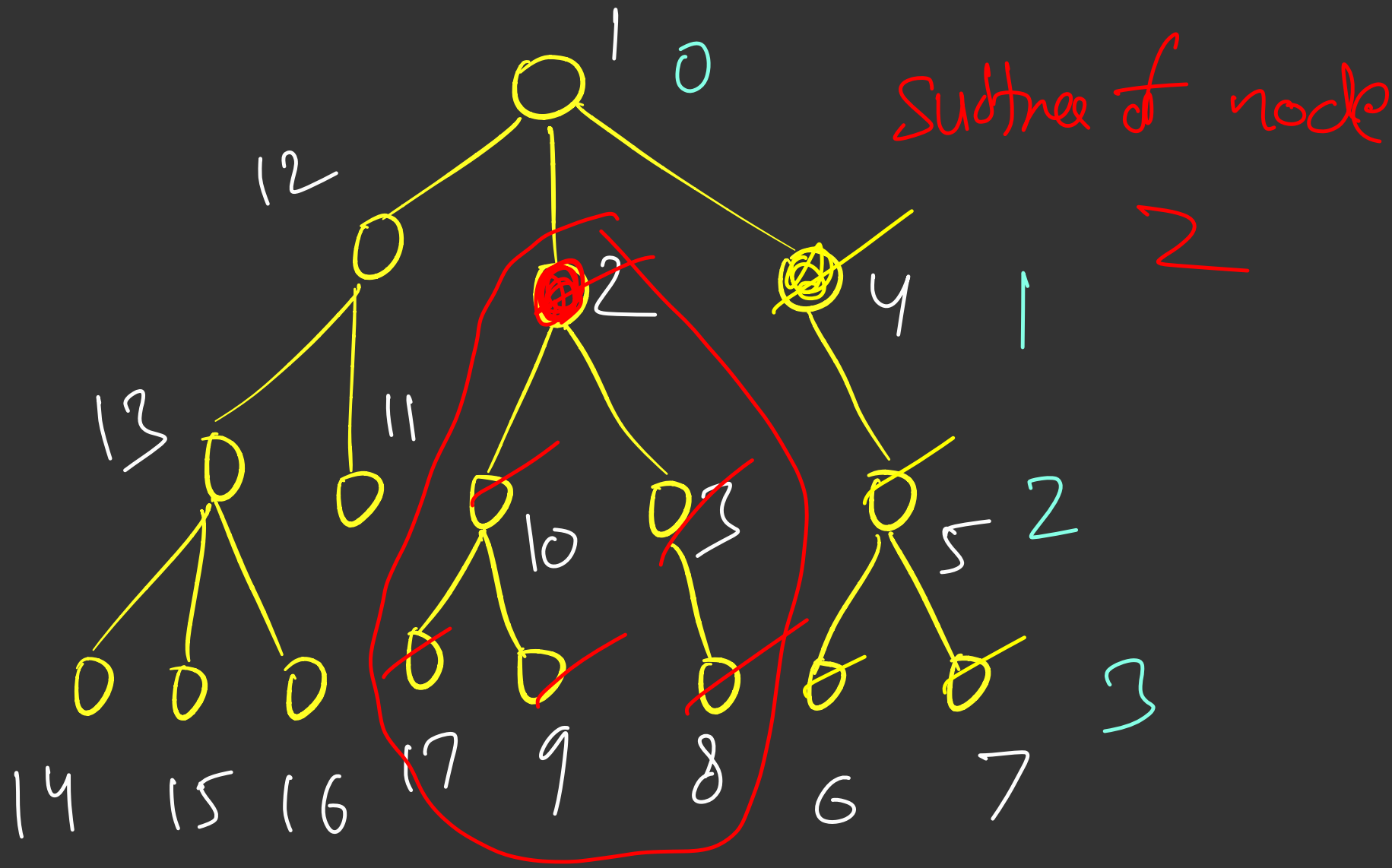
all these are ancestors

level(x) = distance of x from root



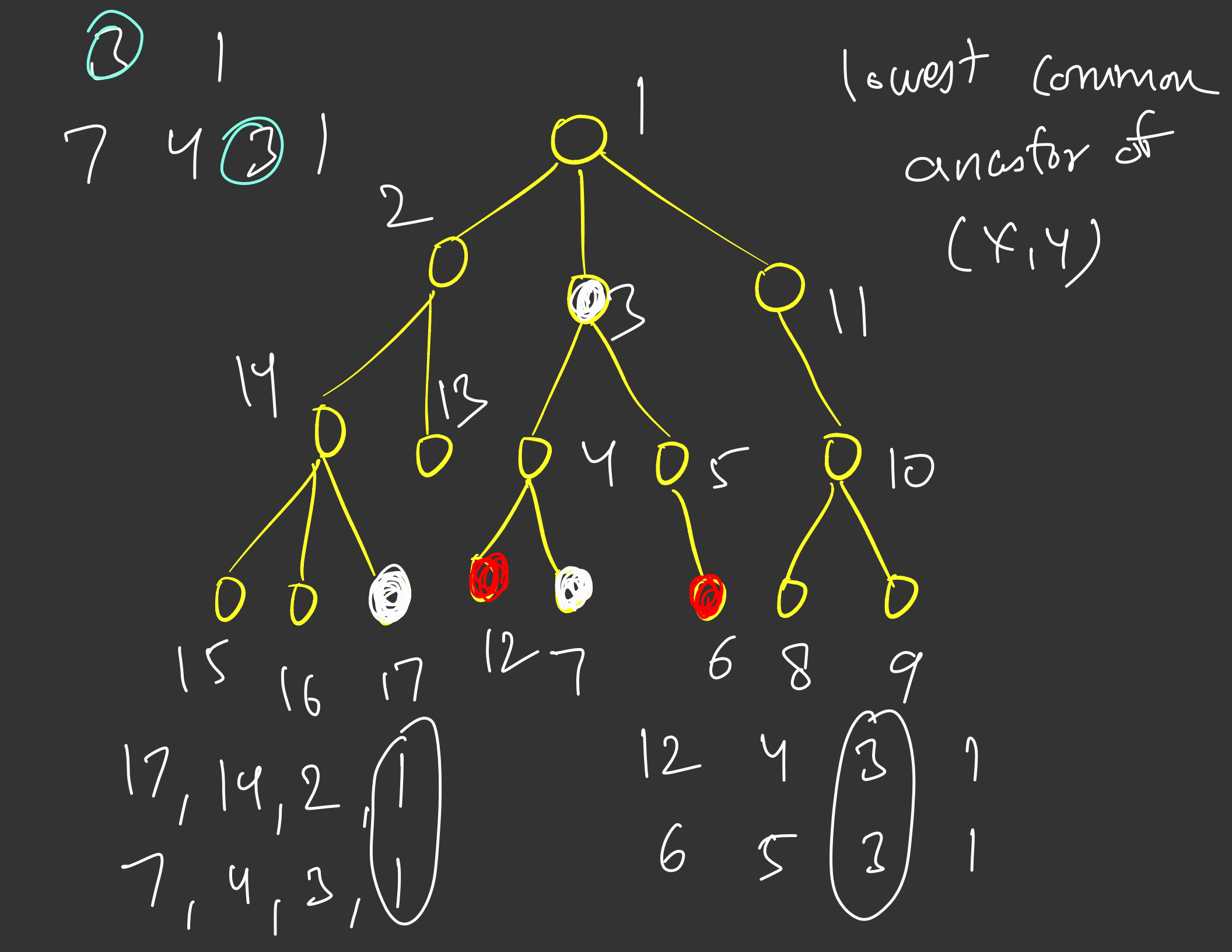
$level(root) = 0$

$level(x) = level(p(x)) + 1$


1
0
Subtree of node
12
2
4
1
2
13
11
10
3
5
2
14
15
16
17
9
8
6
7
3

Height of Tree =

max Level of any node

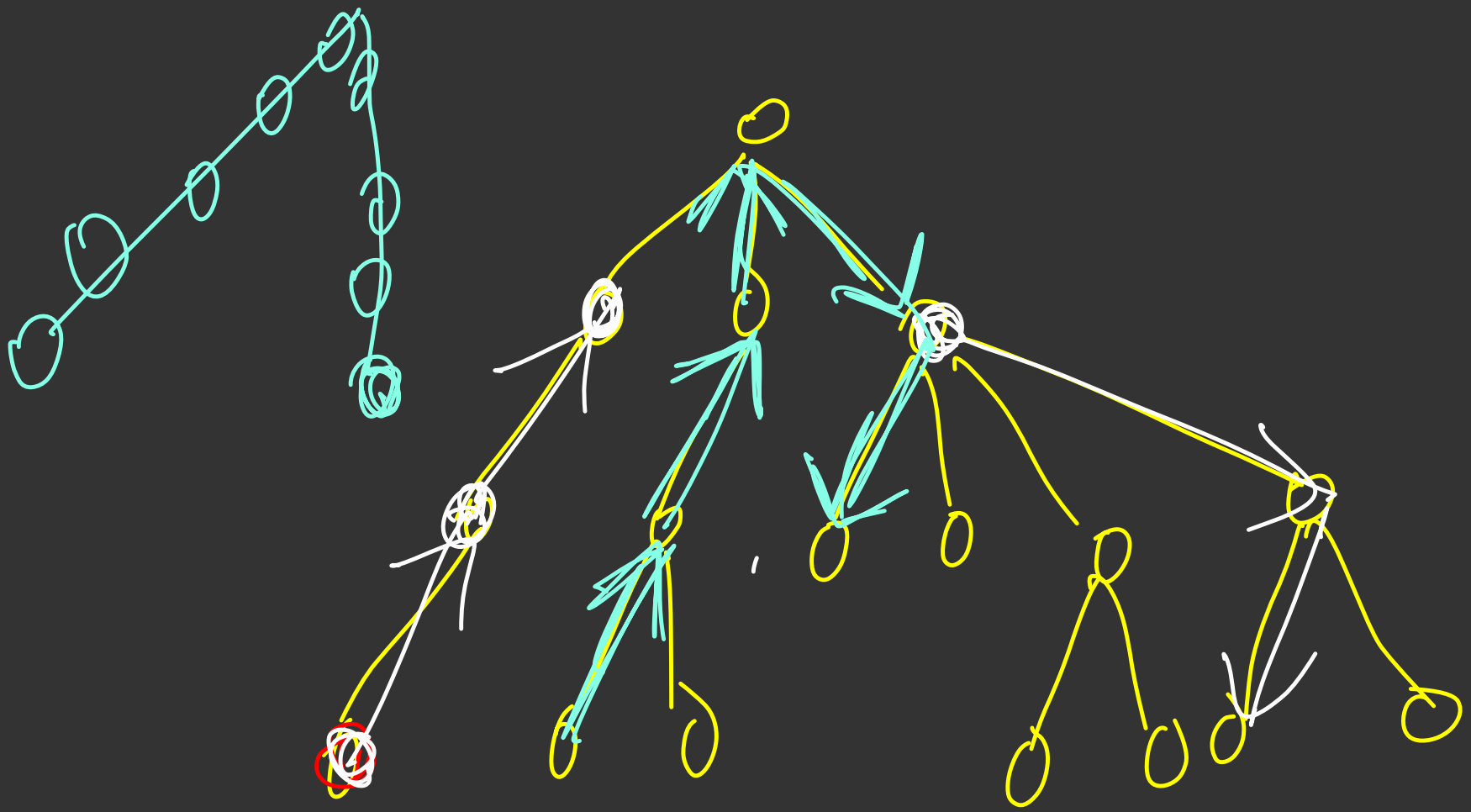

lowest common ancestor of (x,y)
1
2
3
11
14
13
4
5
10
15
16
17
12
7
6
8
9
17, 14, 2, 1
7, 4, 3, 1
12 4 3 1
6 5 3 1

Properties 2

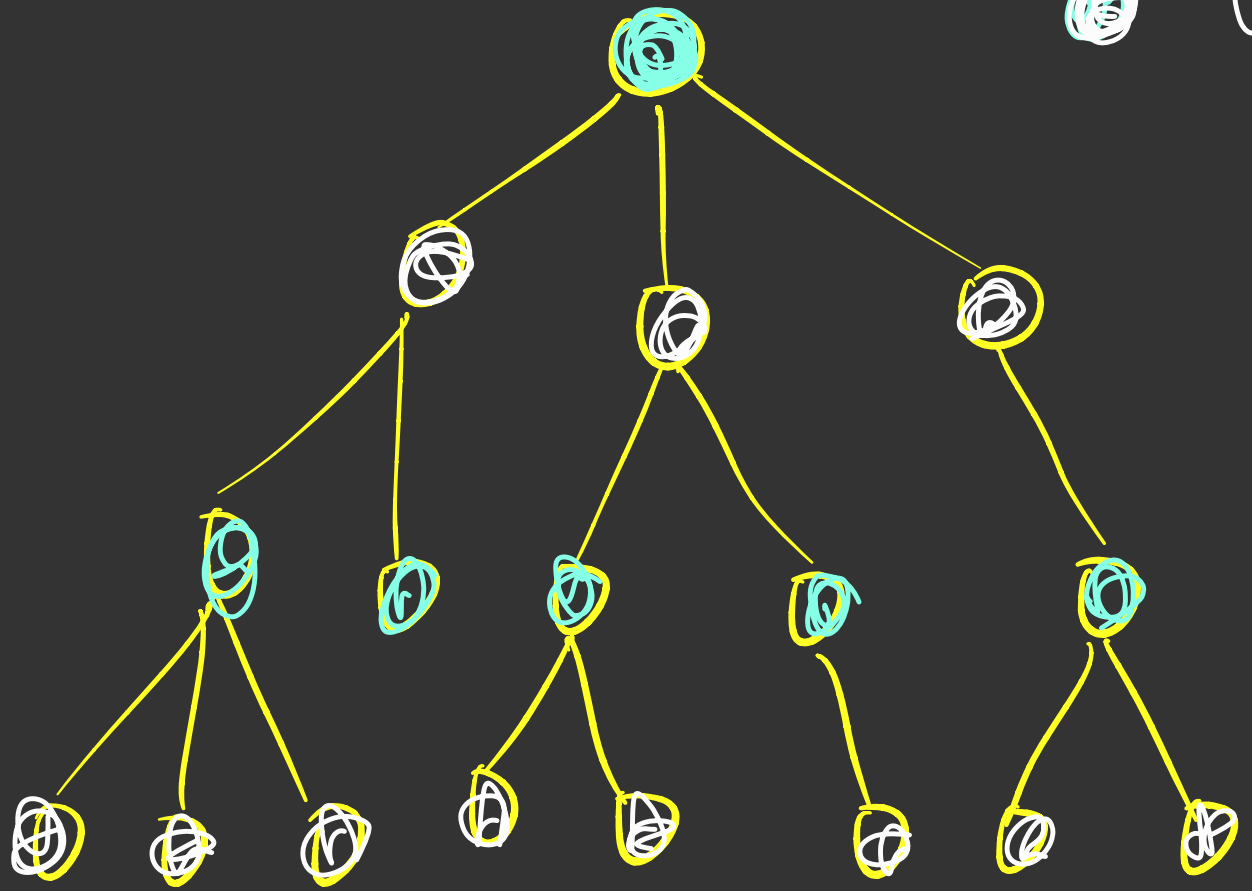
- Can there be more than 2 parents of a single node in a rooted tree
  - No
- Path property. What are the only 3 types of paths possible?
  - 1. **Go Up, Come Down** 2. **Go Up** 3. **Come Down**
- How to color a Tree with just 2 colors such that no two neighbours have the same color
  - Just root the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

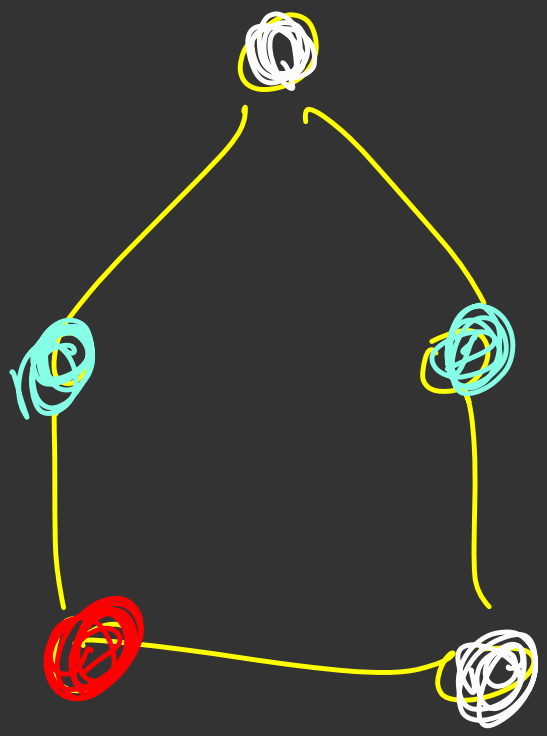
Bonus Tip:

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.



Bipartite Coloring





Neon

White

A tree can always be colored bipartitely

Traversals in a Tree

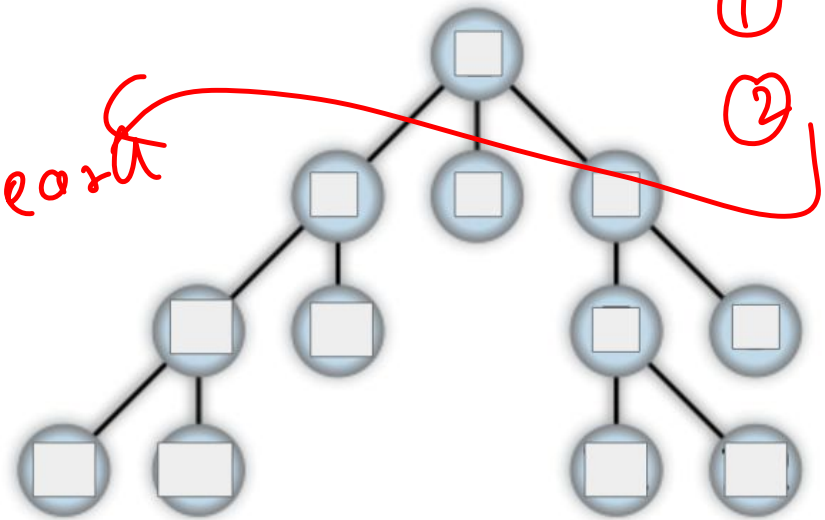
2 types

① DFS

② BFS

Depth first search

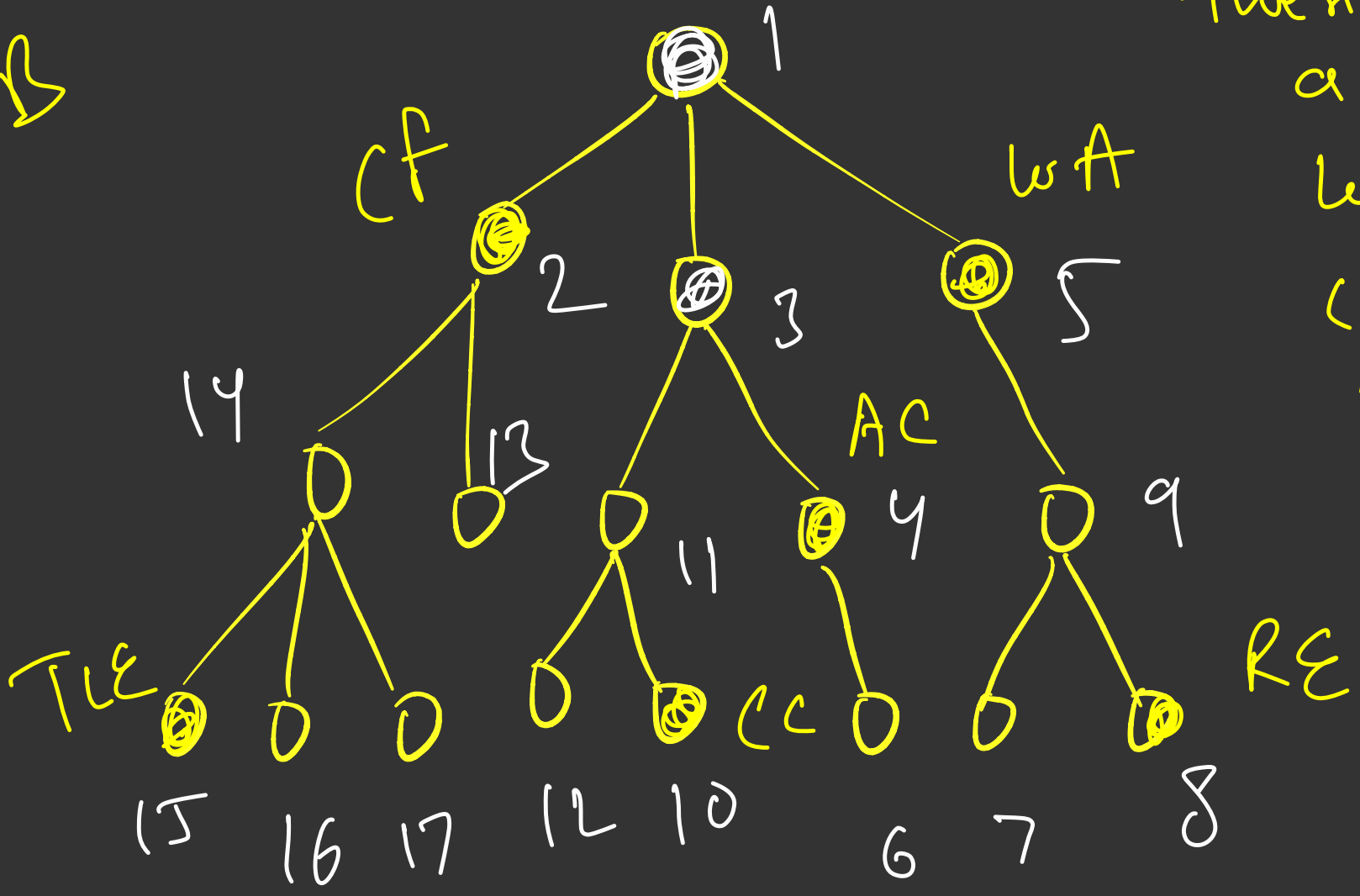
Breadth first search



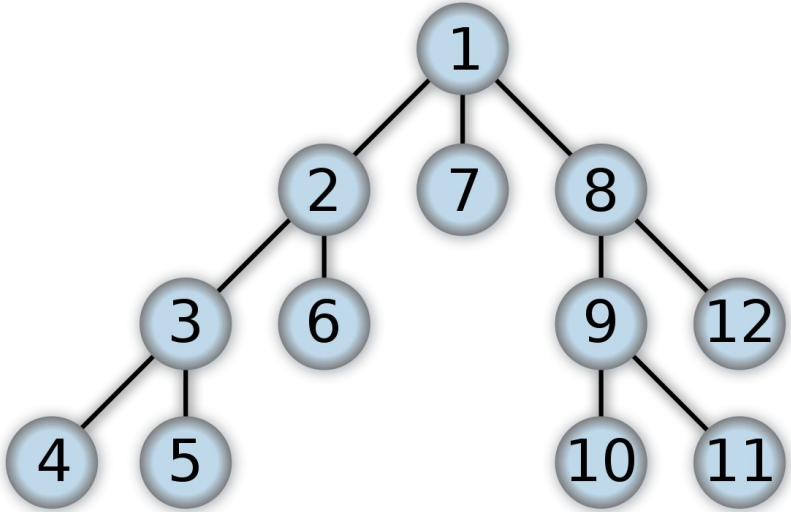
How to traverse the tree in a way such that we visit all nodes

DFS

Tell me whether there is a node which contains "RE"



DFS Traversal in a Tree



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

Nodes are numbered in the order in which they are visited

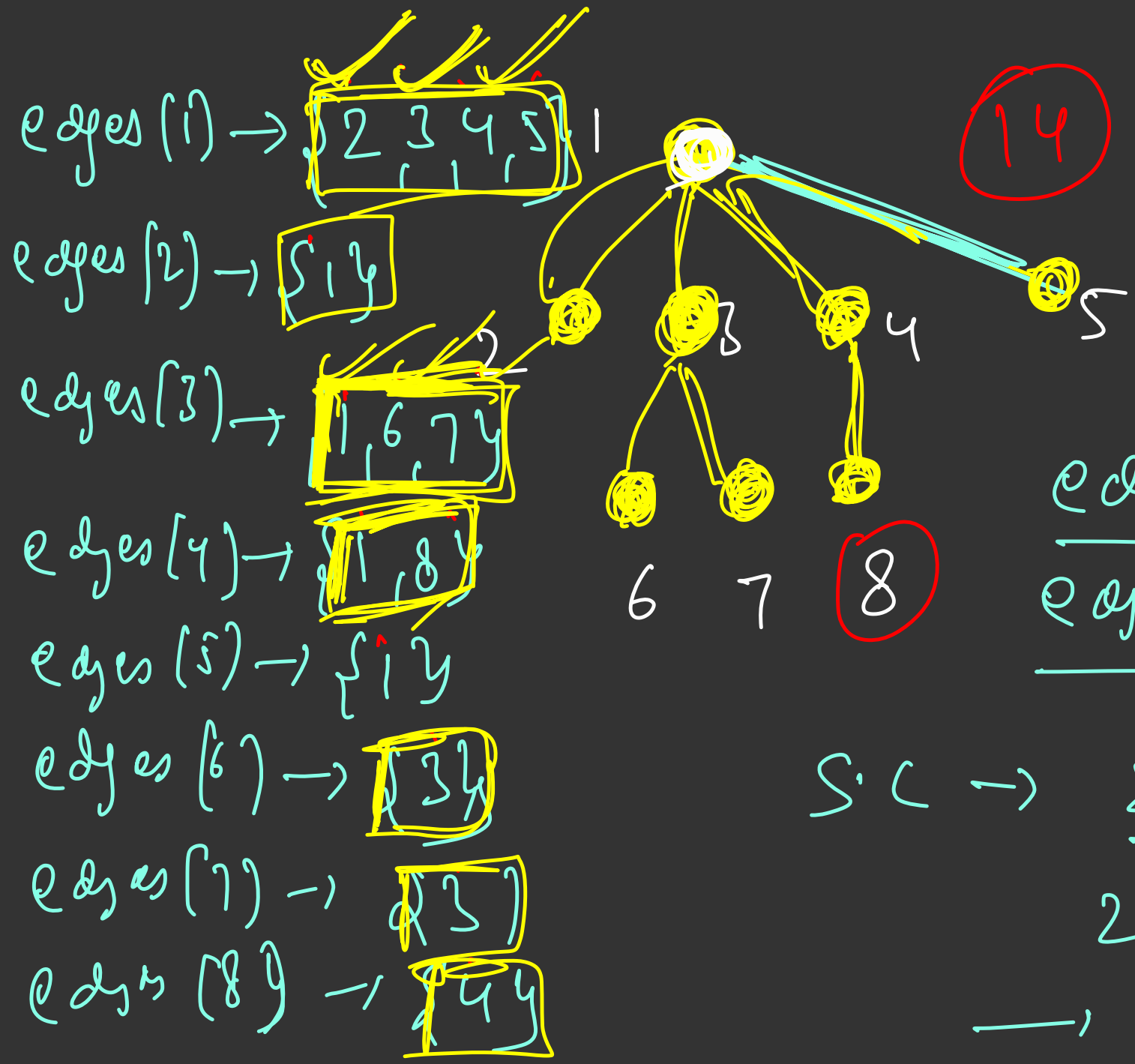
1 7 8 9 11 10 12 2 6 3 4 5

for every node just store the other nodes to which it is connected

vector<vector<int>> edges(n)

edges[i] -> { all nodes to which node i is connected }

adjacency list



14

N nodes in a tree

edges[5] → 1

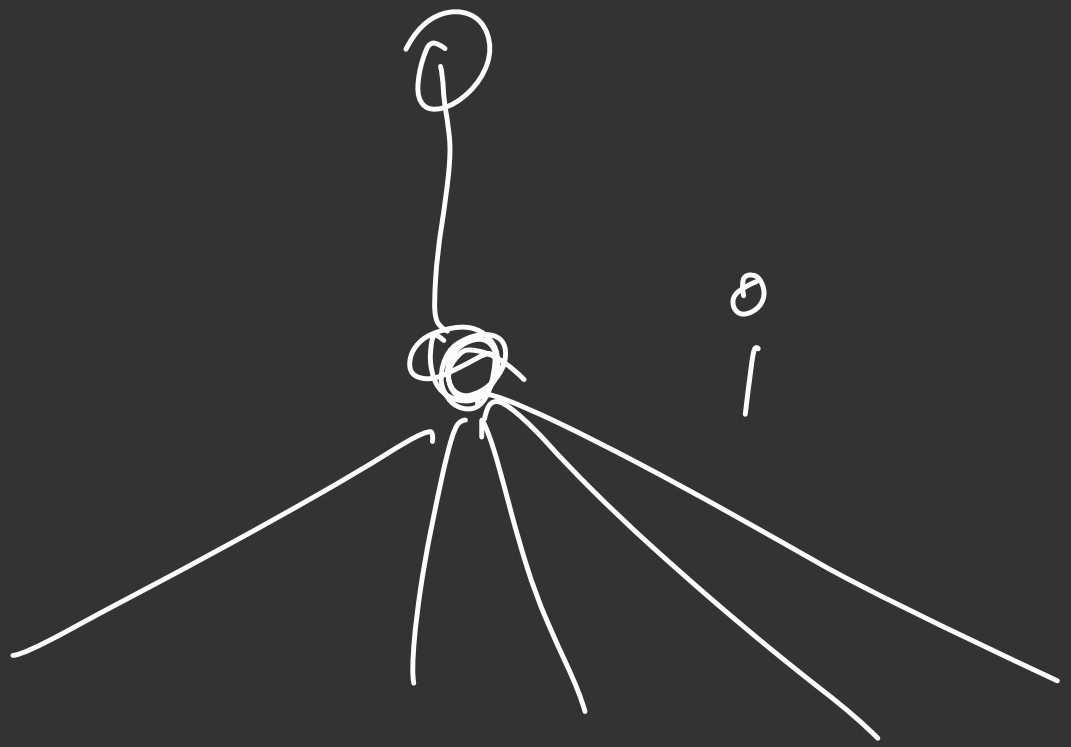
edges[1] → 5

S.C → 2E

2(n-1)

→ O(n)

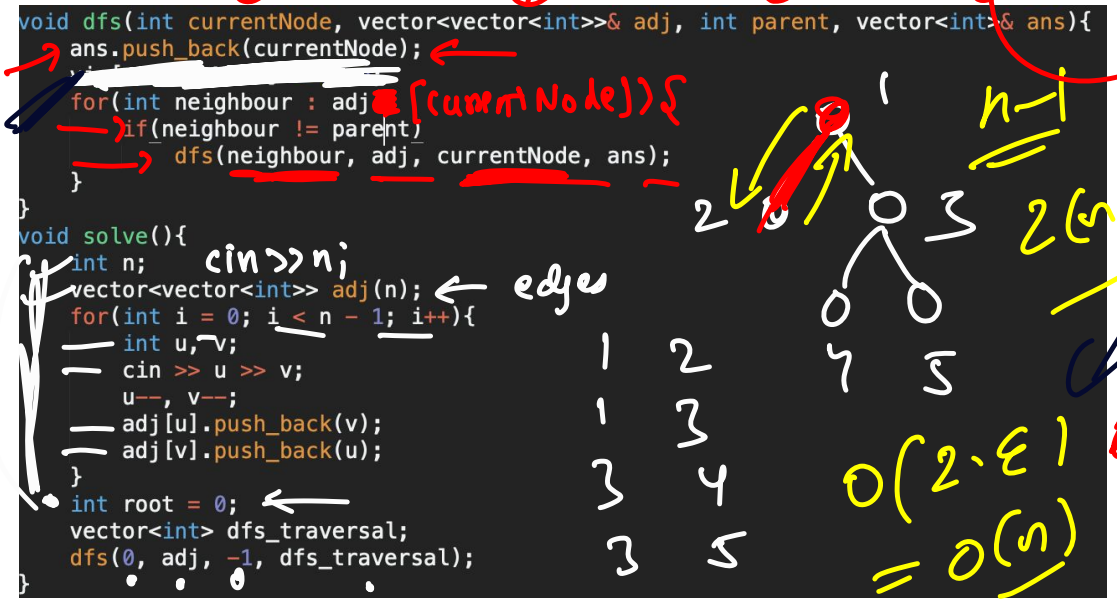
edges(i) → {p(i), children(i)}



```
vector<vector<int>> edges(n);
```

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);
}
```

Time Complexity: O(N)

DFS Traversal in a Tree

Implementation:

```
void dfs(int currentNode, vector<vector<int>>& adj, int parent, vector<int>& ans){
    ans.push_back(currentNode);
    vis[currentNode] = true;
    for(int neighbour : adj){
        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);
}
```

T.C — total iterations of this for loop

$$T.C \Rightarrow \sum_{n=1}^{n} \text{iterations}(n)$$

$$T.C = \sum_{n=1}^{n} |\text{neighbours}(n)|$$

Time Complexity: O(N)

$$T.C = \sum_{x=1}^{n} \text{degree}(x)$$