Introduction to Trees 1

- Priyansh Agarwal

What is a Tree?

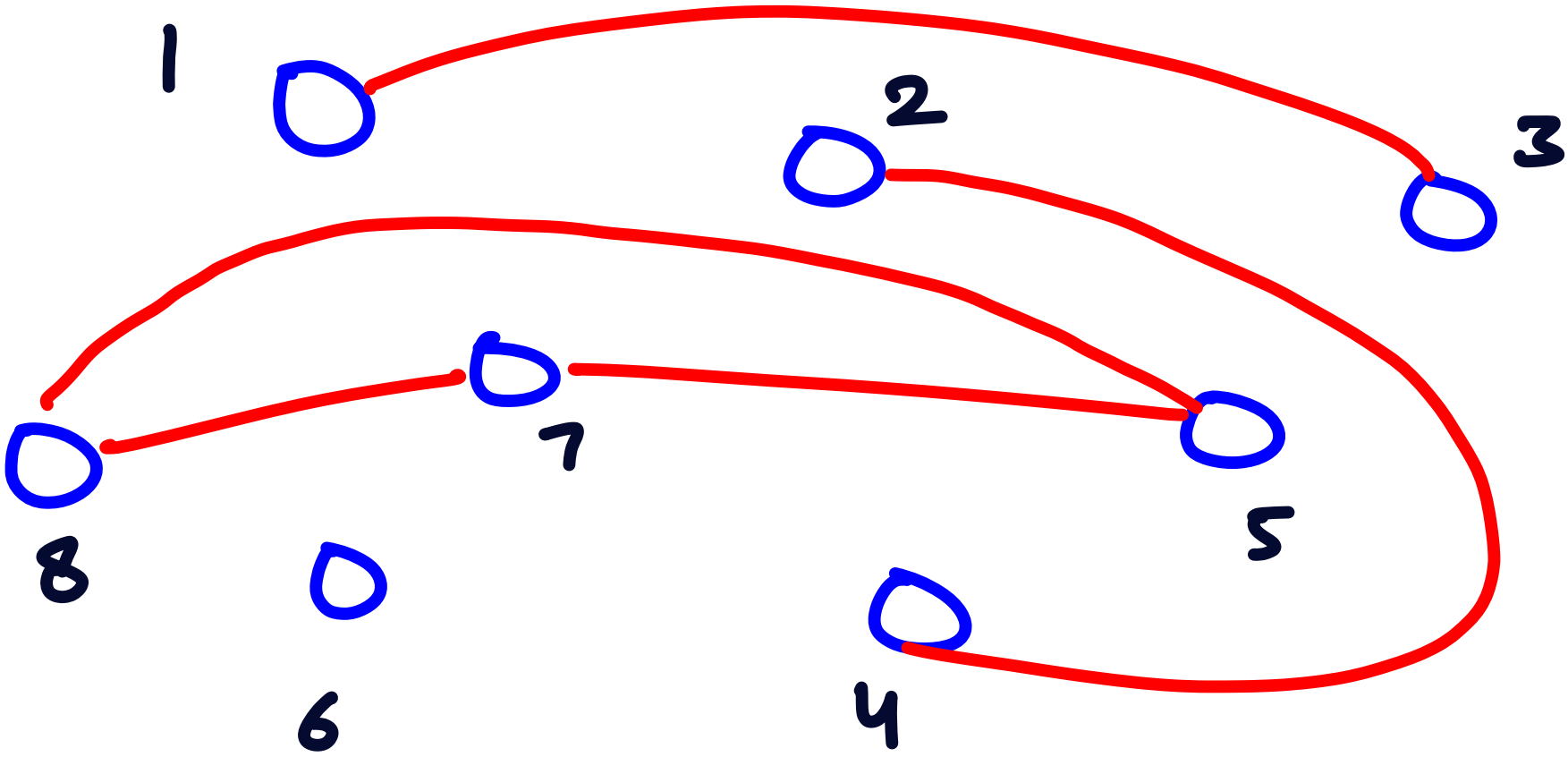
Every Tree is a Graph but every graph is not a tree

A connected **graph** of nodes without any cycles.

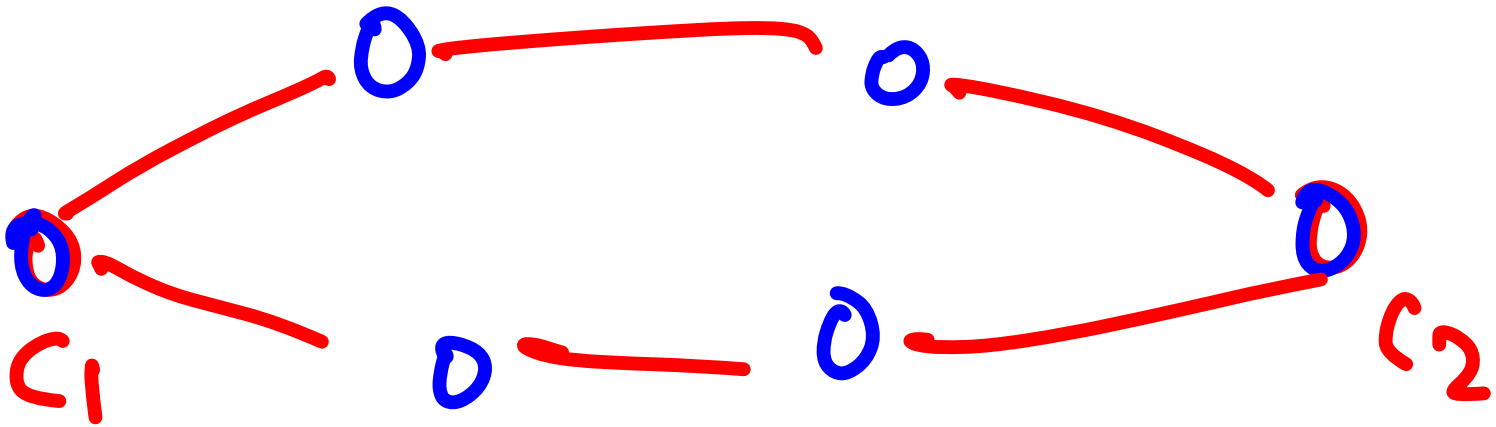
What is a graph?

Imagine it like the Earth. 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 so we consider them as **disconnected**.

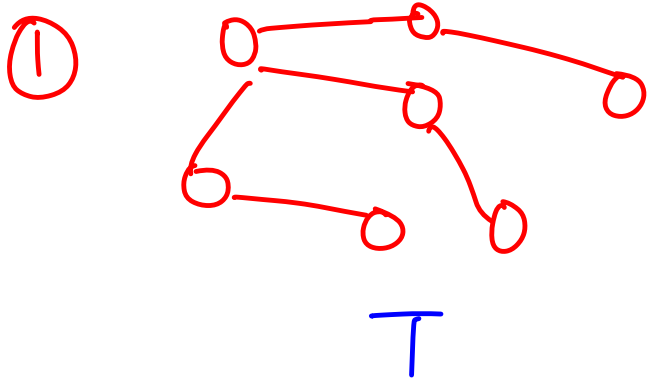
A tree is one such continent with a unique path b/w any 2 countries

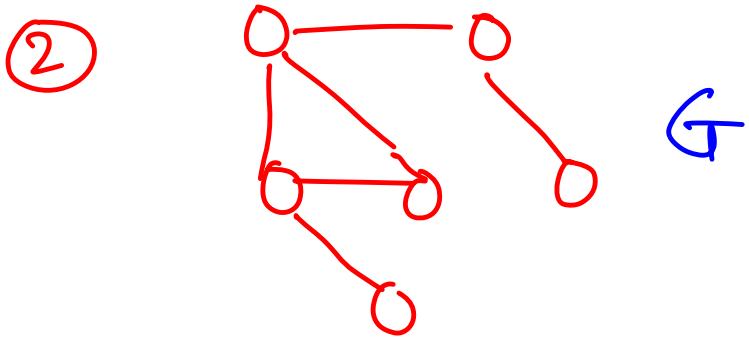


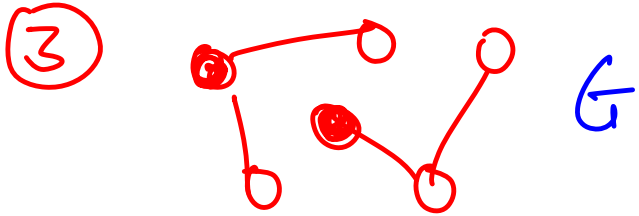
Graph

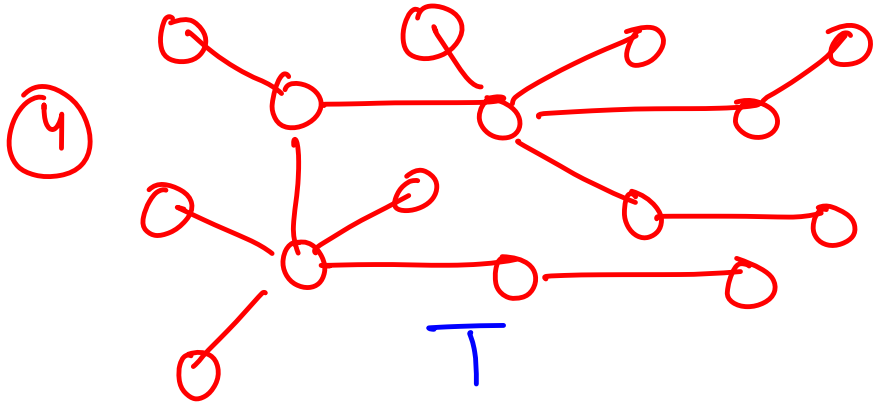


Difference b/w Tree & Graph - illustrations



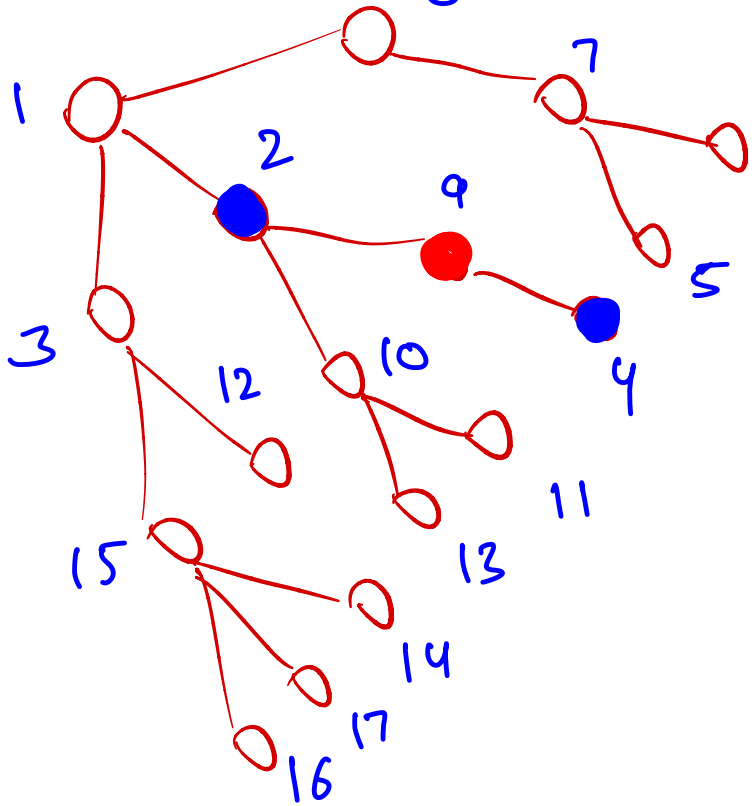






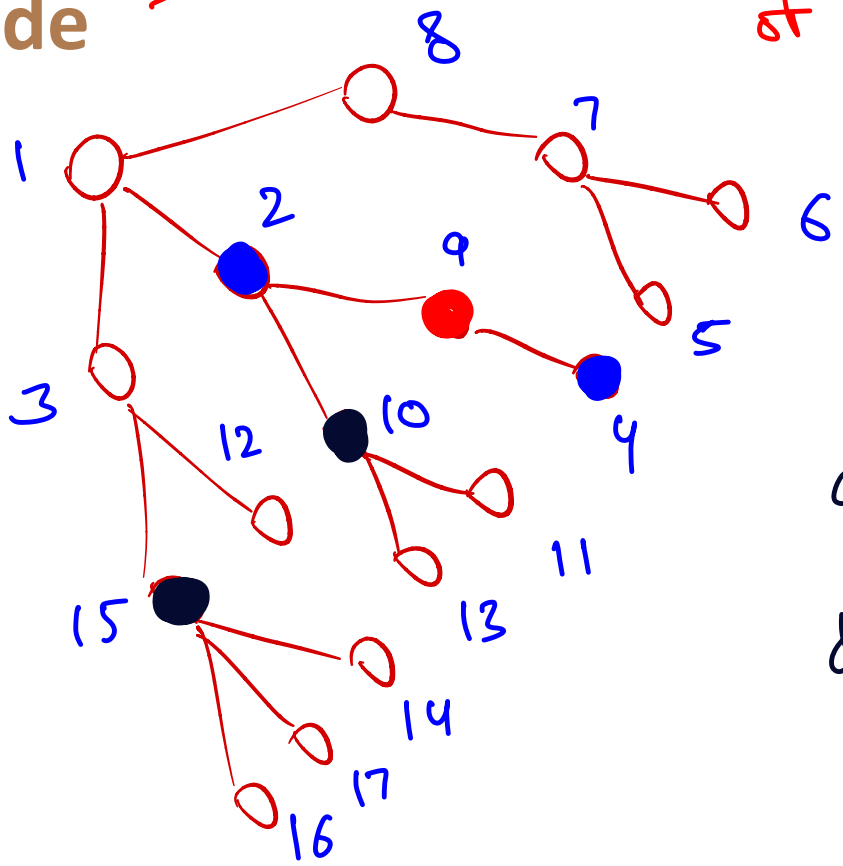
Neighbour of Node (x)

all those nodes which have an edge directly connecting them to x



Degree of Node

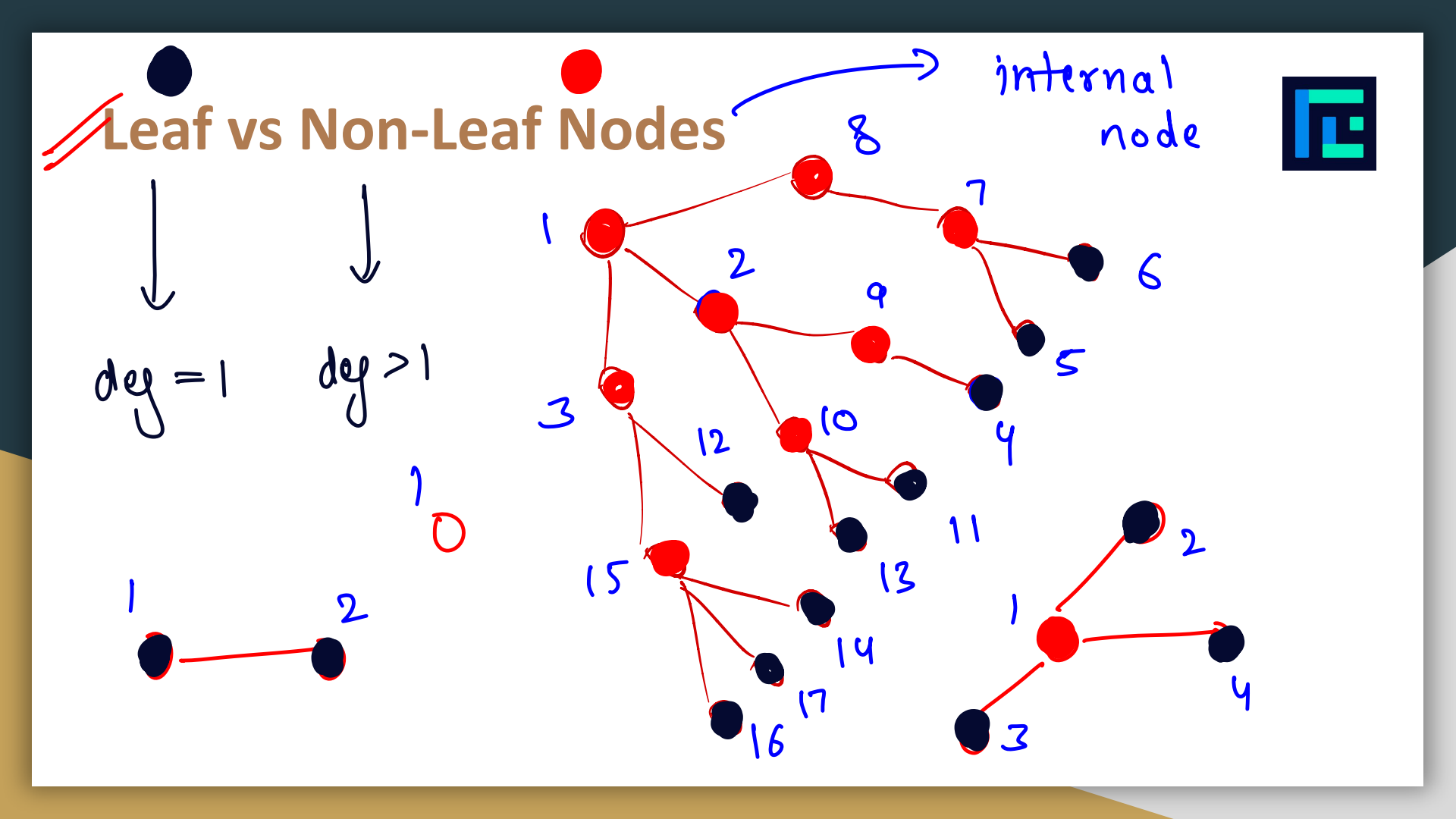
no. of neighbours of the node



deg(10) = 3

deg(15) = 4

Leaf vs Non-Leaf Nodes

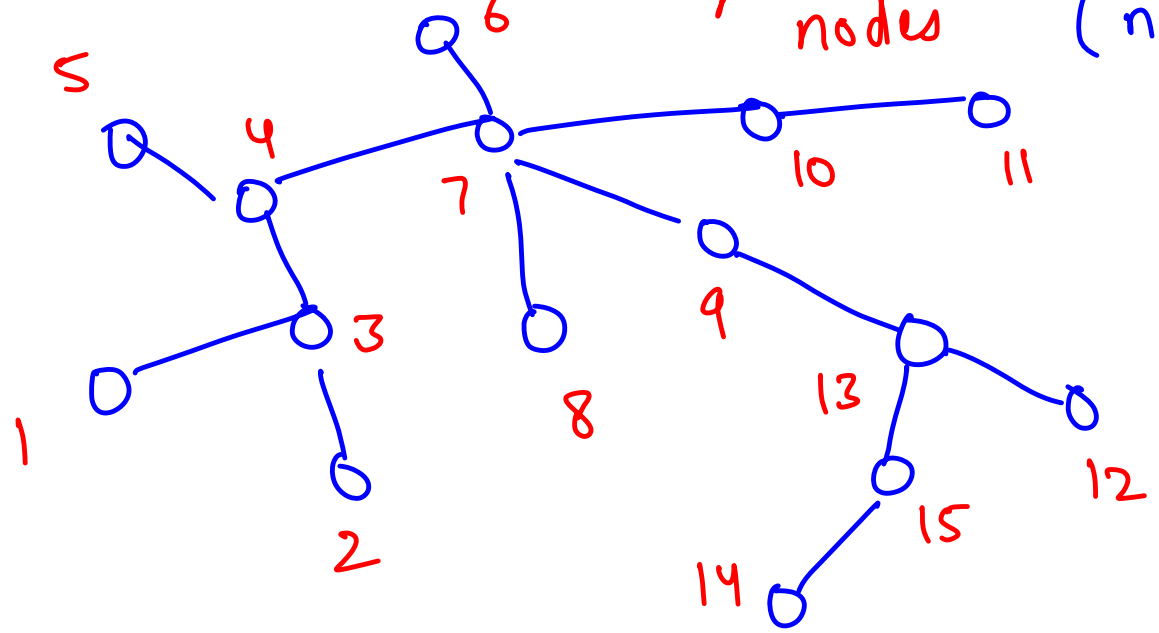


Diameter of Tree

→ maximum distance b/w any 2 nodes in the tree

endpoints of any diameter are always leaf nodes

(non-unique)

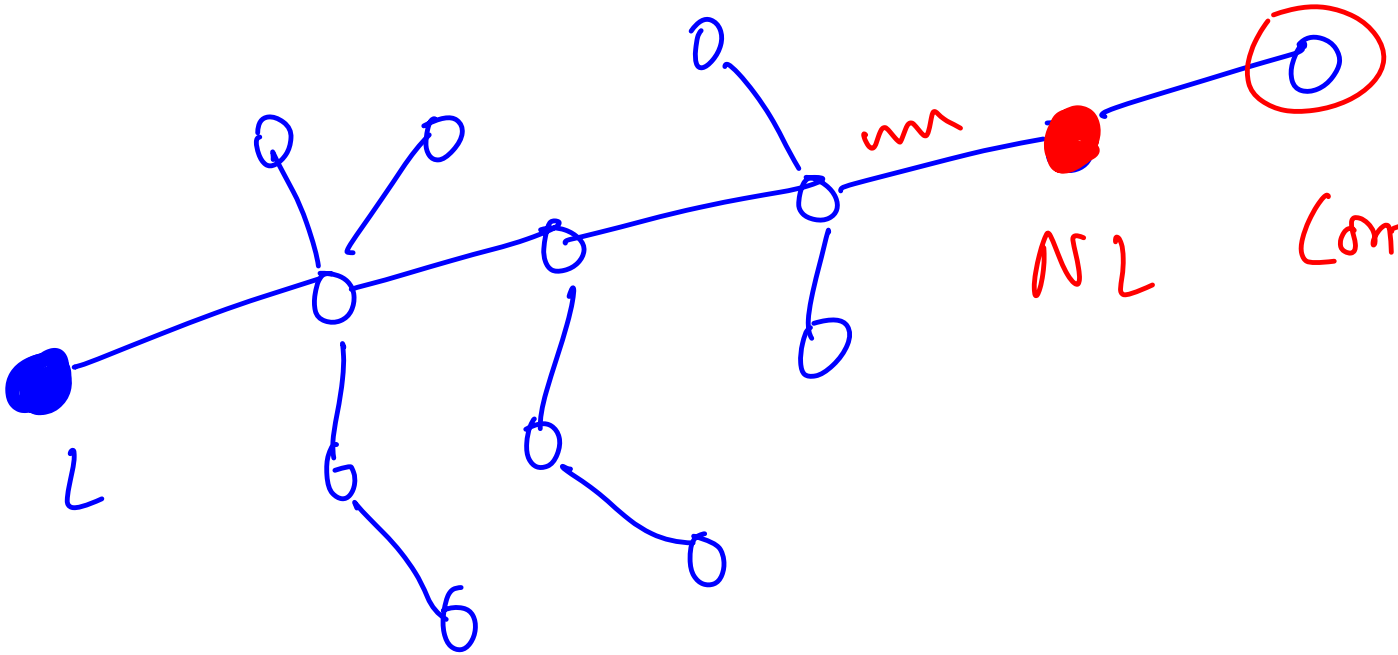


(2, 14) – 7

(1, 14) – 7

Both endpoints of a diameter are always leaf nodes.

Assume that one of those endpoints is a non-leaf



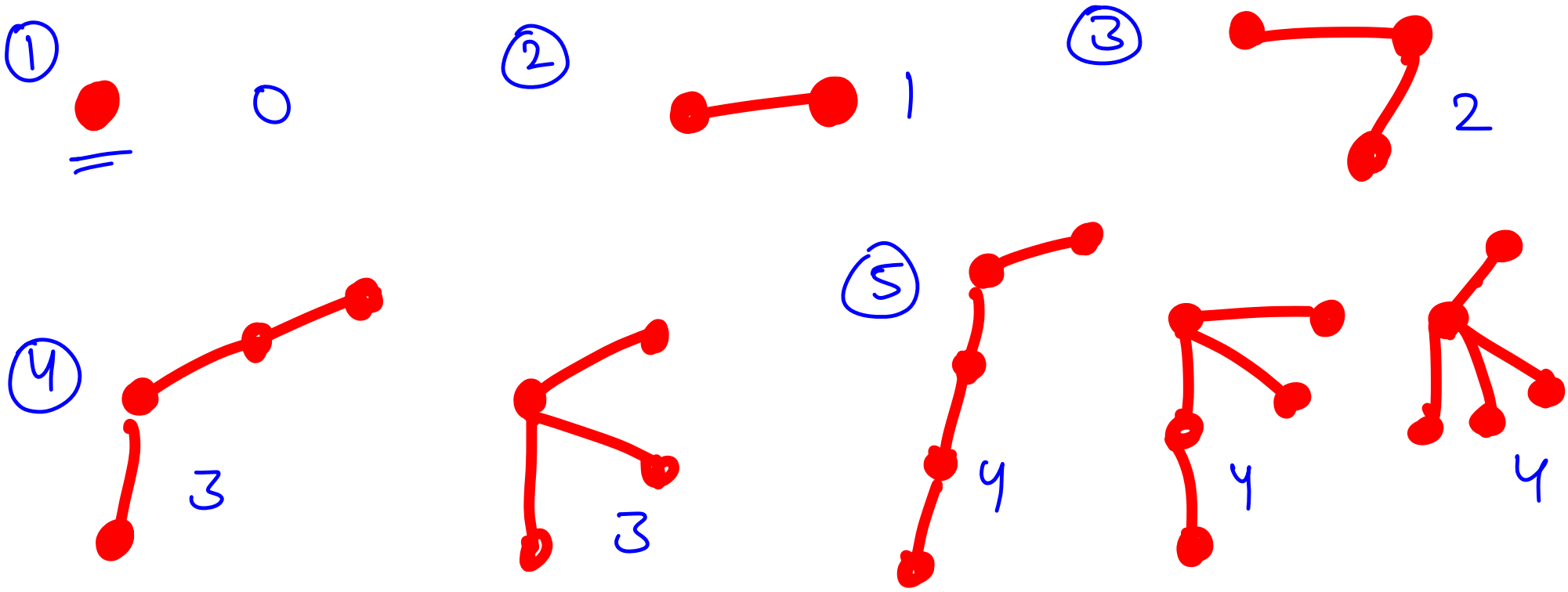
Properties 1

- There exists a unique path b/w every 2 nodes in a Tree
- Number of edges in a Tree for N nodes = N - 1
- Sum of degree of all nodes = 2 * (N - 1) hence it is always even
- A Tree cannot have less than 2 leaf nodes unless the Tree only contains a single node.

A tree's diameter is both end points are leaf nodes and every tree must have a diameter

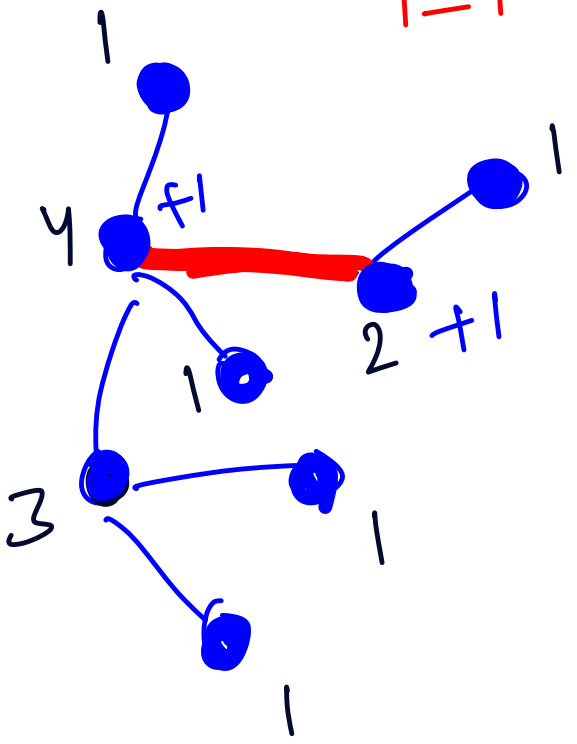
no. of edges in a tree of n nodes

$= n - 1$



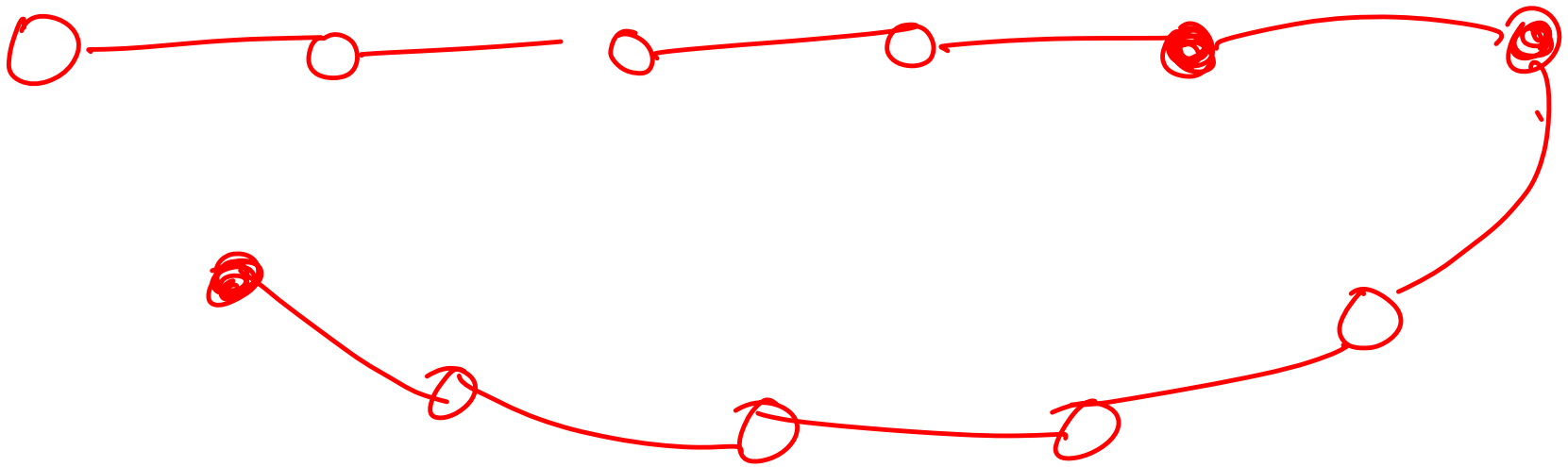
sum of degree of all nodes in a Tree of N nodes = 2(N-1)

$$\sum_{i=1}^{N} deg(i) = 2 \cdot (N-1)$$

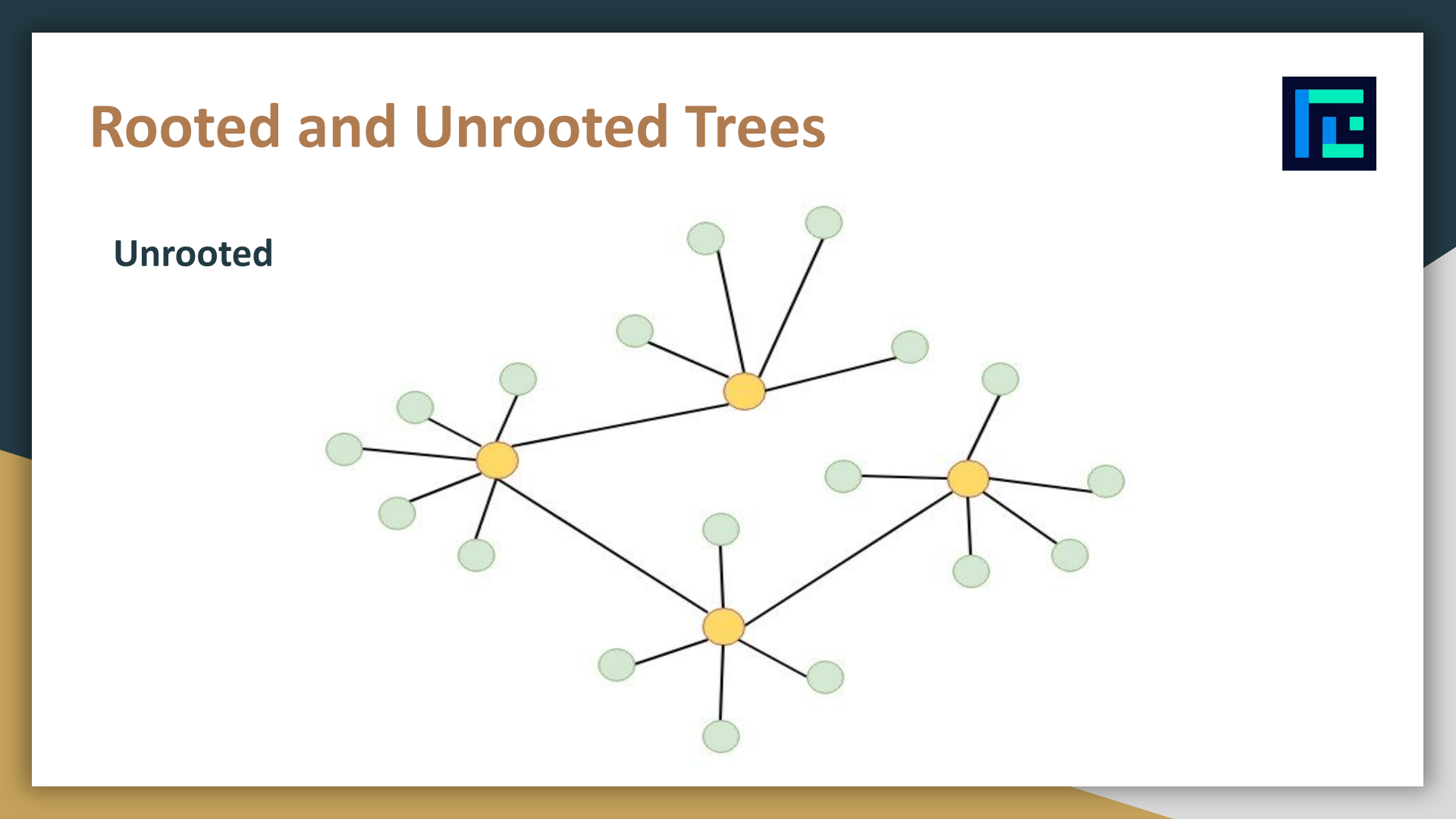


——> every edge contributes +2 to the total degree sum

A tree with more than 1 node is bound to have atleast 2 leaf nodes

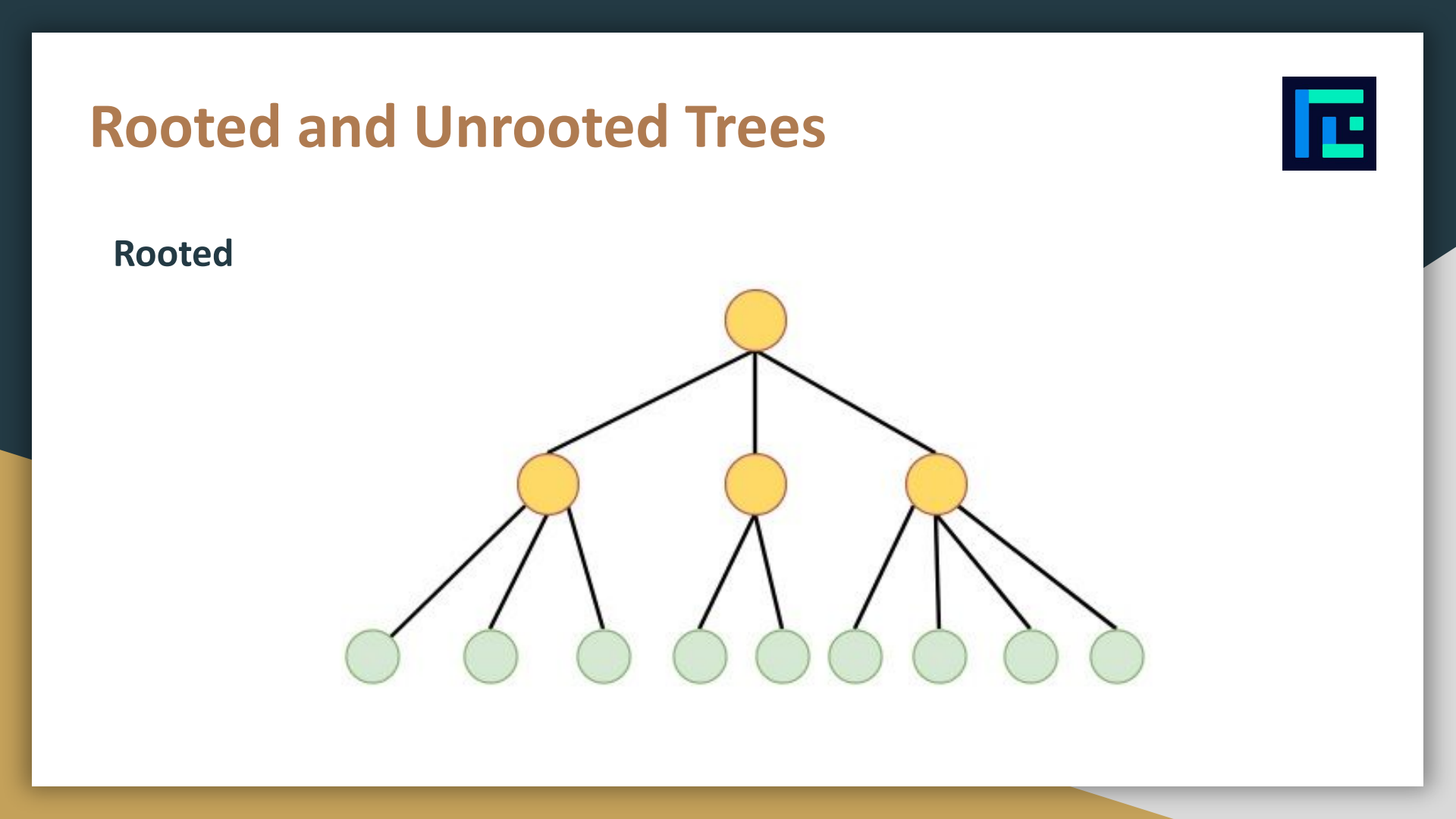


Rooted and Unrooted Trees



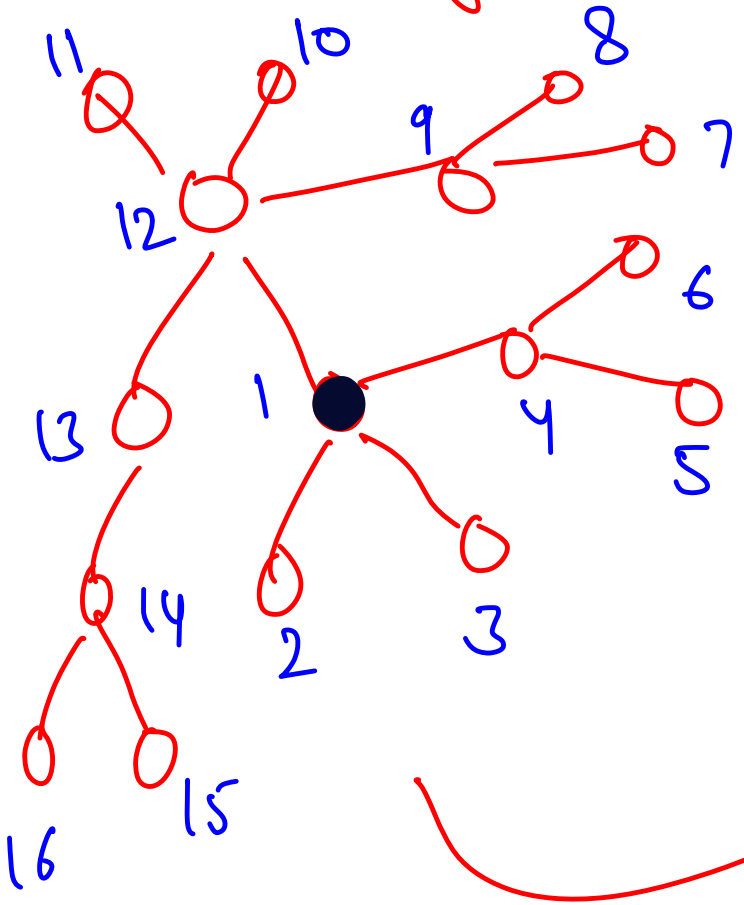
Unrooted

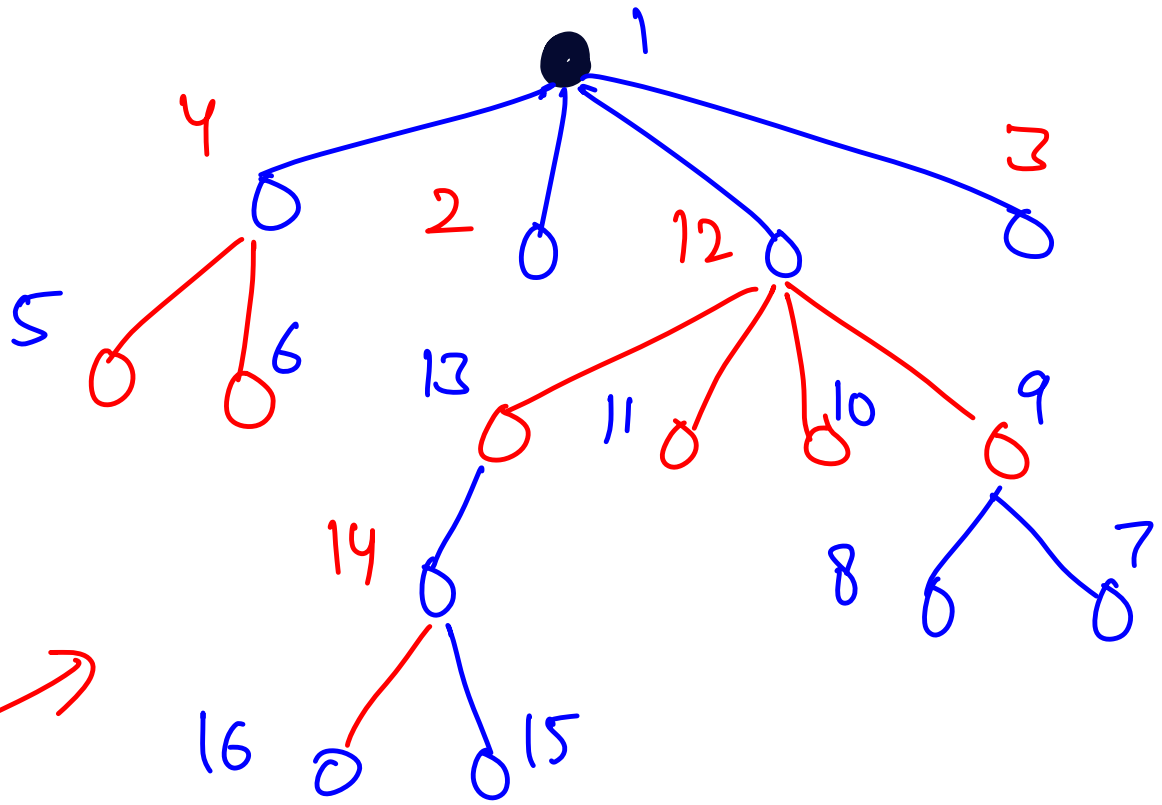
Rooted and Unrooted Trees



Rooted

Converting an unrooted tree into a rooted tree





Important Terms in Rooted Tree

- Root
- Parent → $p(x)$ = 2nd node in the path from x to root
- Child
- Ancestor
- Descendant
- Level of Node
- Subtree
- Subtree Size
- Height of Tree
- Lowest Common Ancestor

multiple parents? No

children(x) = all the nodes y such that parent(y) = x

# of children(leaf) = 0

Important Terms in Rooted Tree

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

ancestor(x) = p(x), p(p(x)), p(p(p(x))) ..... root

→ all nodes in the path from x to root (excluding x)

descendant(x) = all nodes y such that ancestor(y) includes x

level(x) = dist of x from root

Important Terms in Rooted Tree

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

N = N = T

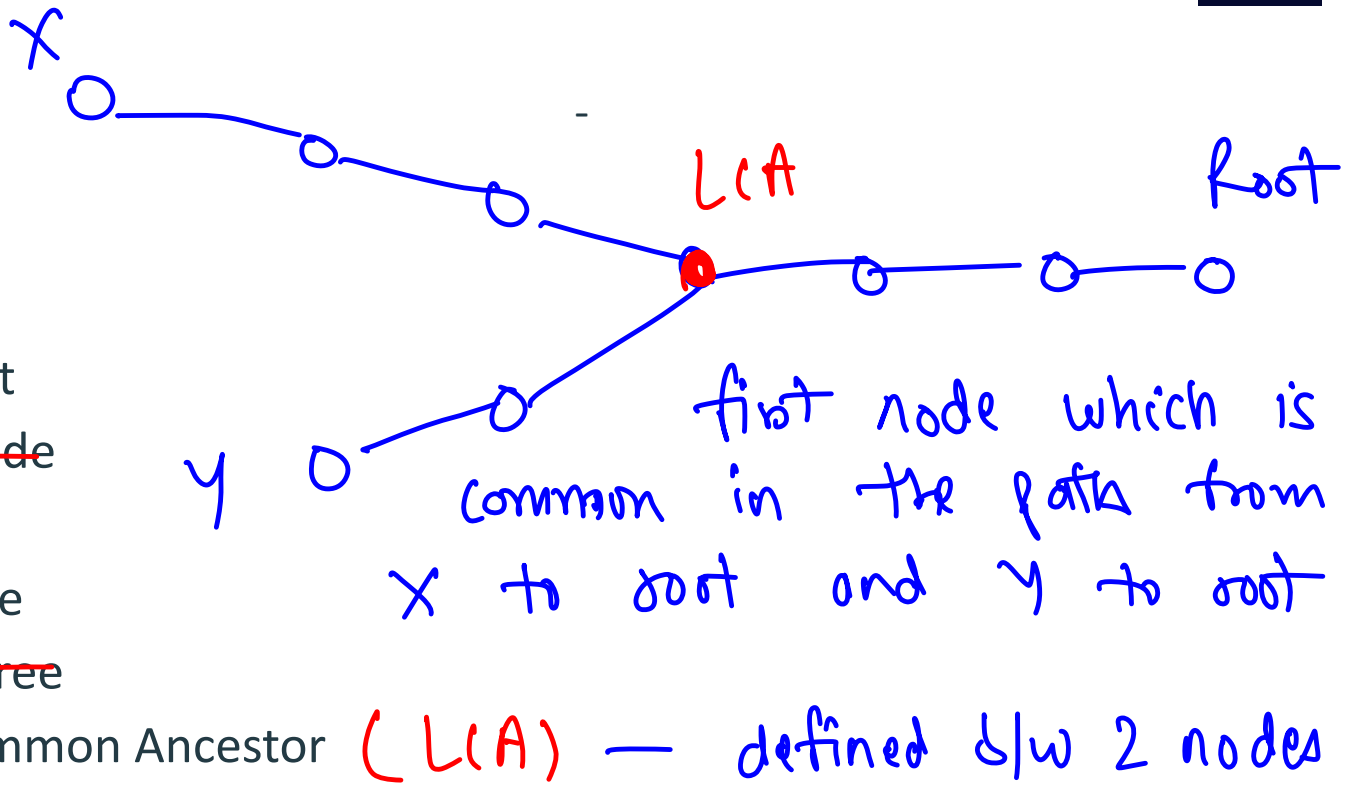
height(T) = maximum level of a node in the tree

Subtree(x) = set of all descendants of x and x

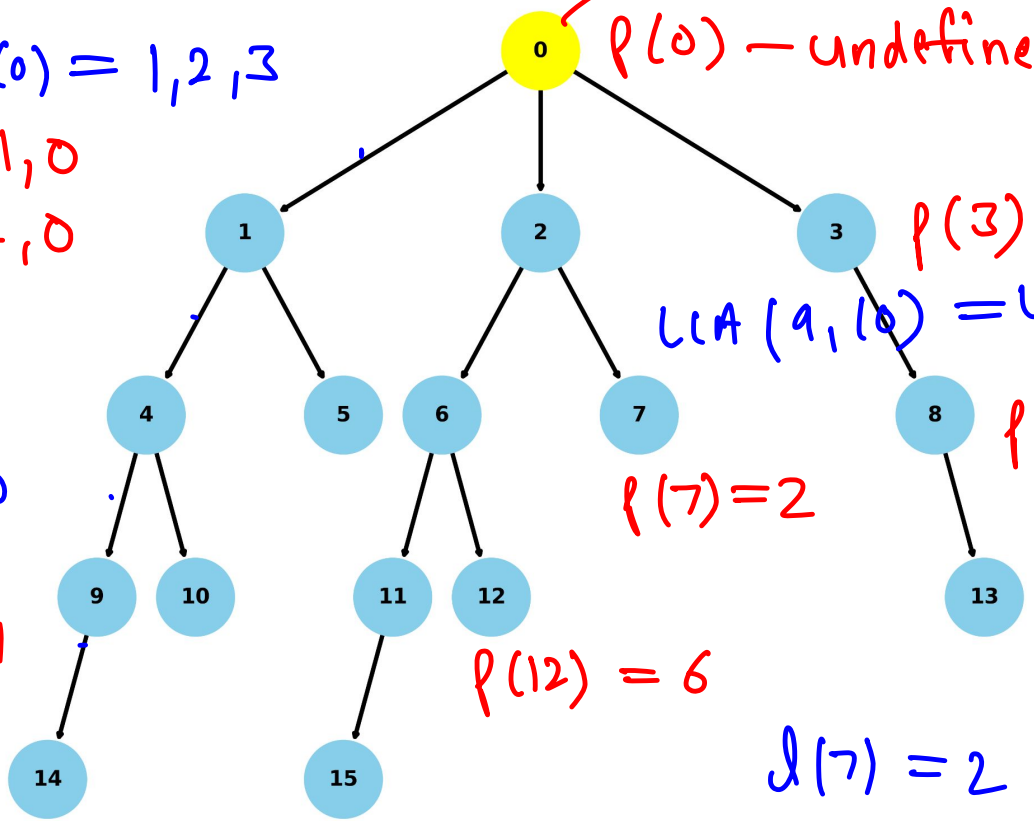
subtree(root) = all nodes in tree

Important Terms in Rooted Tree

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



Let's Visualise



s(6) = 6, 11, 12, 15

LCA(7, 11) = 2

Root

p(0) – undefined = -1

C(0) = 1, 2, 3

a(14) = 9, 4, 1, 0

a(12) = 6, 2, 0

a(1) = 0

C(4) = 9, 10

d(2) = 6, 7, 11, 12, 14

l(14) = 4

p(3) = 0

LCA(9, 10) = 4

p(8) = 3

p(7) = 2

p(12) = 6

d(0) = 0

d(7) = 2

Properties 2 (Rooted)

- There can only be 1 unique parent of a node in a rooted tree.
- A Path in a rooted can only be of 3 forms:
  - 1) **Go Up, Come Down** 2) **Go Up** 3) **Come Down**

Can be split into 2 paths

- Every node of a tree can be colored with just 2 colors so that no two neighbours have the same color (Bipartite coloring)

2 and 3

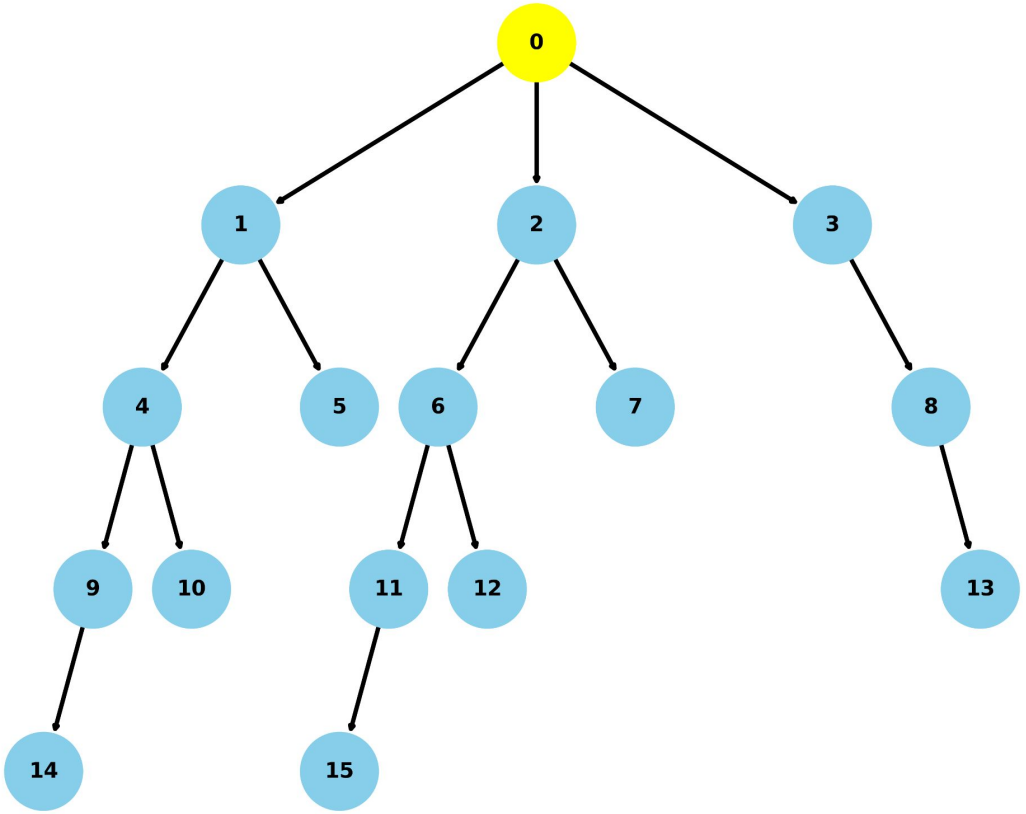
Bonus Tip: Most Codeforces problems less than 1800 rated on Codeforces can be easily solved if you just remember the basic properties and learn to apply them.

Let's Visualise

(10, 0) — 2

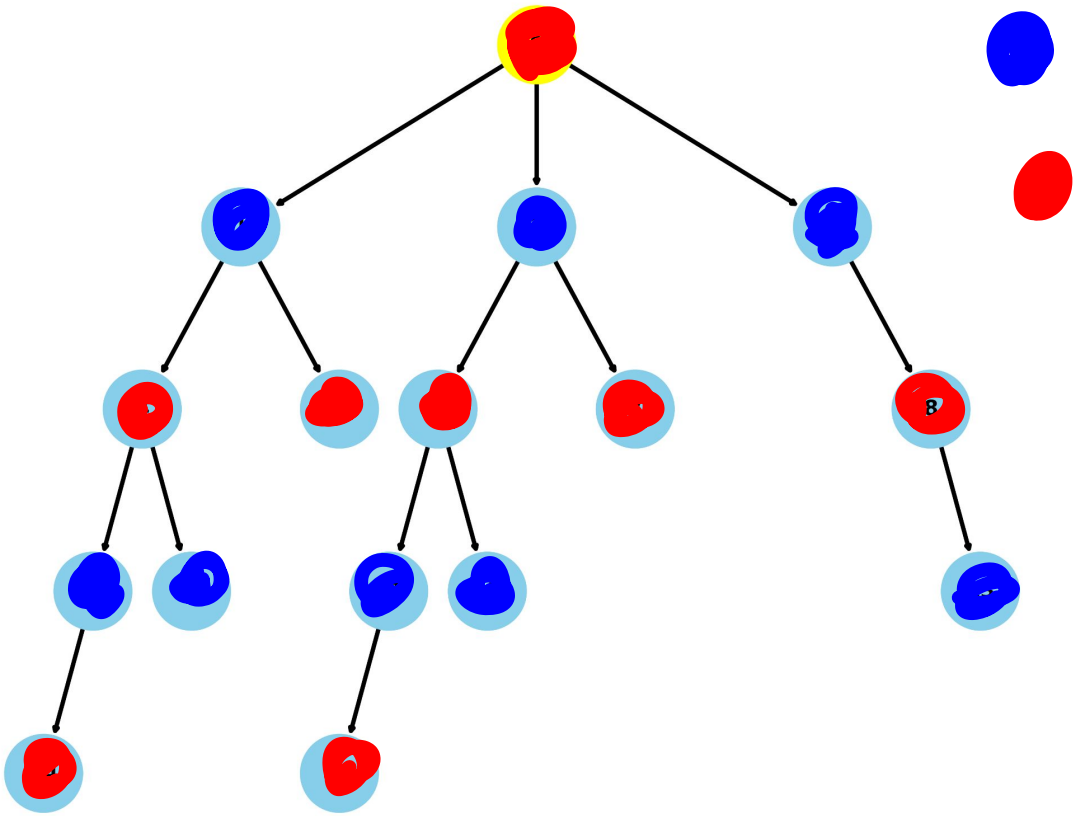
(14, 13) — 1

(0, 15) — 3



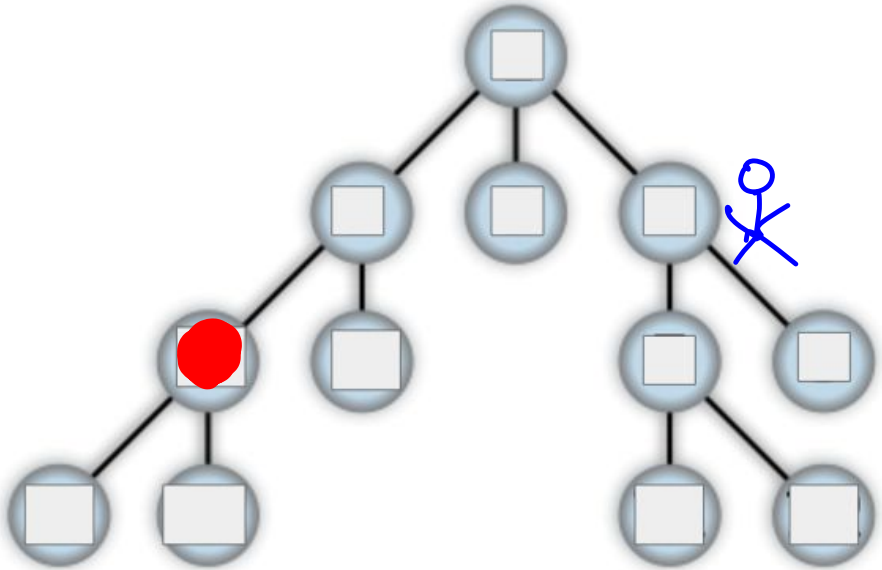
Let's Visualise

(Bipartite coloring)



Traversals in a Tree

Depth first search
Breadth first search



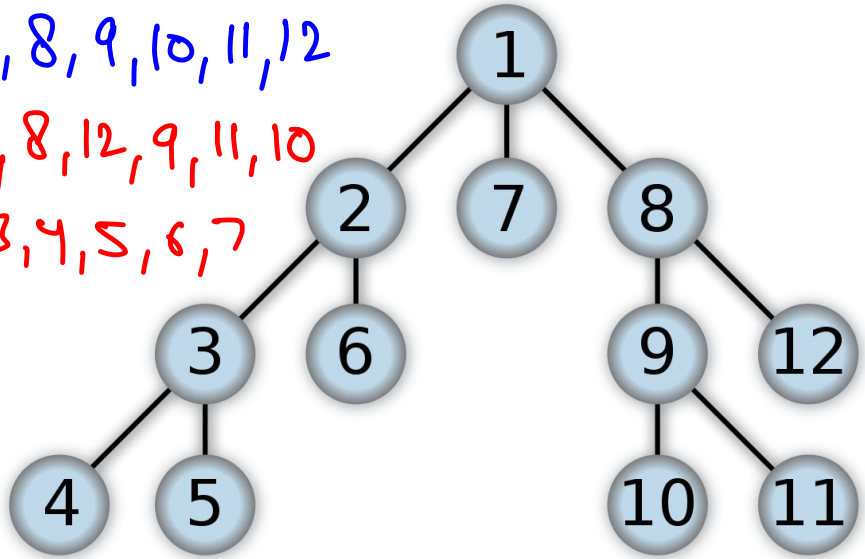
How to traverse the tree in a way such that we visit all nodes

DFS Traversal in a Tree

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

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

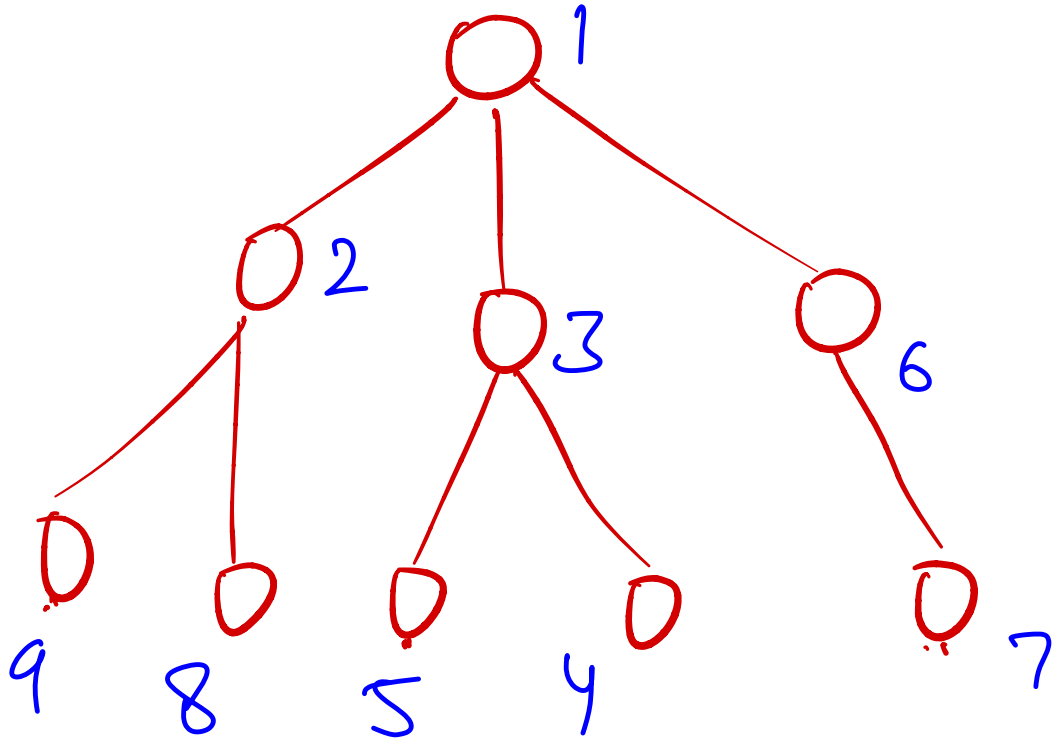
1, 8, 9, 10, 11, 12, 2, 3, 4, 5, 6, 7



Nodes are numbered in the order in which they are visited

Storing a Tree

① store the parent for every node (rooted tree)



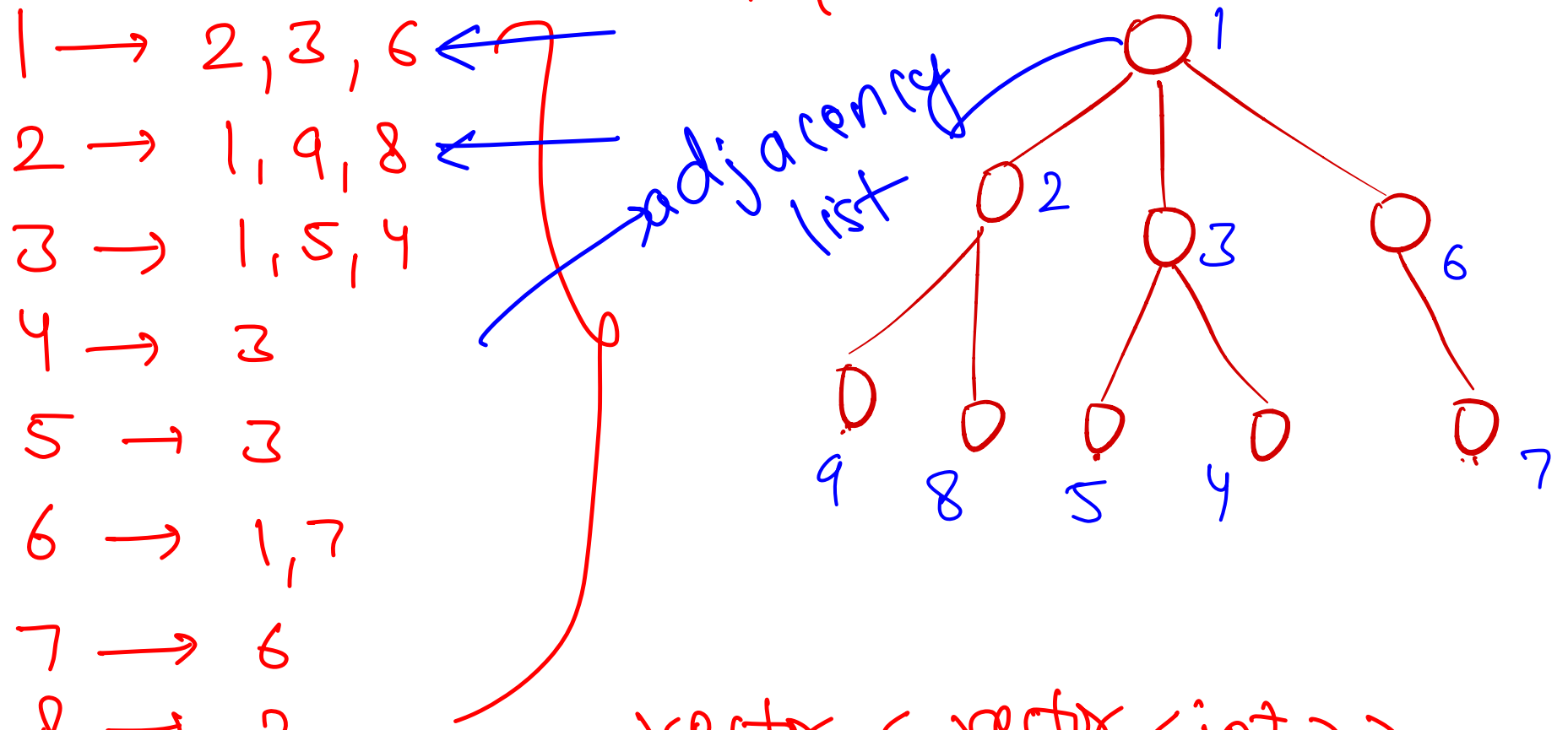
O(N) space

-1	1	1	3	3	1	6	2	2
1	2	3	4	5	6	7	8	9

parent array

(2) Store the neighbours for each node in the tree

map $\{ int \rightarrow vector \}$



vector < vector <int> >

Space complexity $\longrightarrow$ $2(N-1) \longrightarrow O(N)$

set of edges

find out Dfs traversal of the entire tree from some node X

Dfs → 1, 2, 9, 8, 3, 5, 4, 6, 7

1 → 2, 3, 6

2 → 1, 9, 8

3 → 1, 5, 4

4 → 3

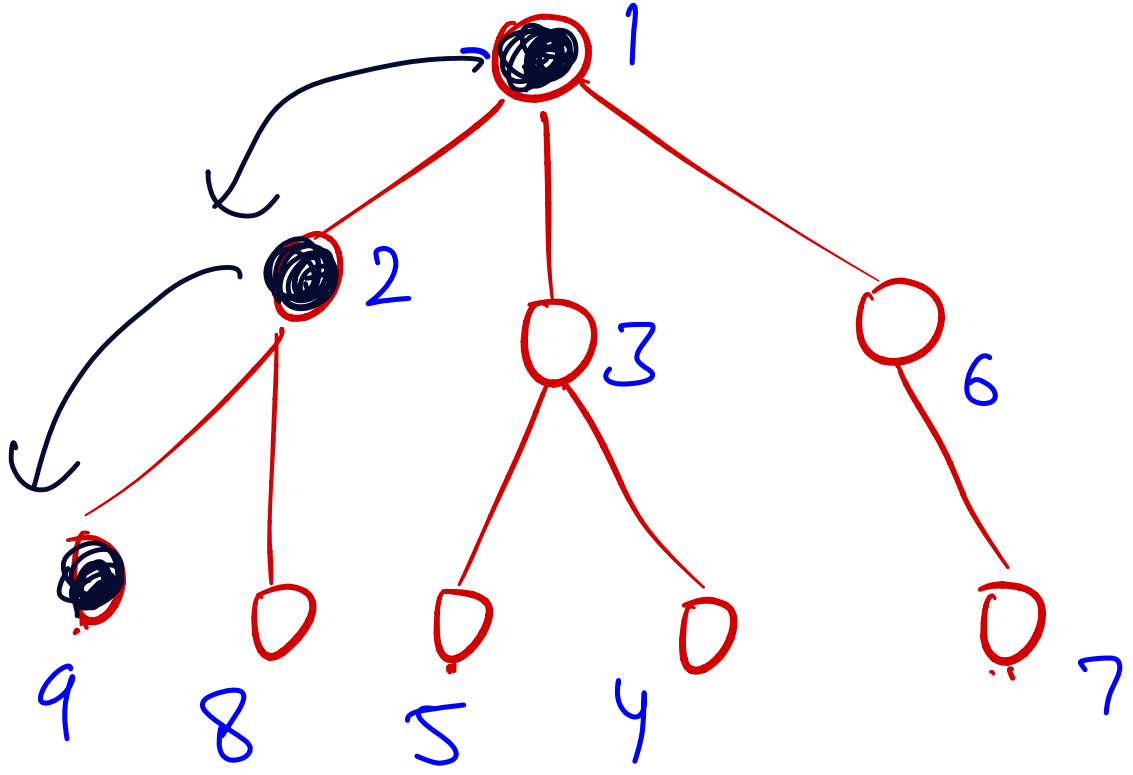
5 → 3

6 → 1, 7

7 → 6

8 → 2

9 → 2



DFS Traversal in a Tree - Code O(N)

```
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);
}
```

→ $2 \cdot (N-1)$

→ N

cin>>n;

edge list

u, v

input is 1 based index

cin>>root;

root