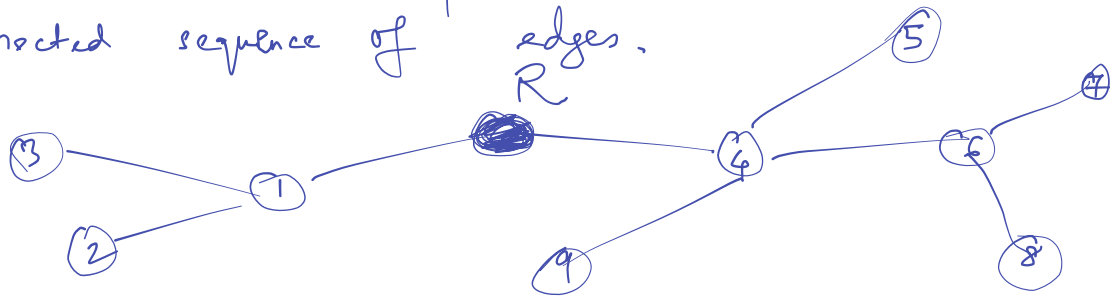


# TREES

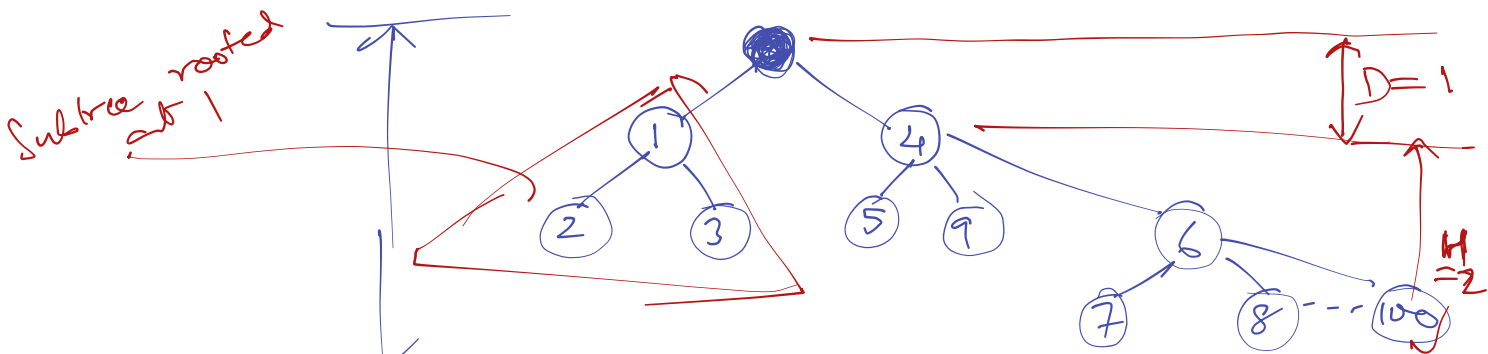
## Rooted Trees

Tree: Set of nodes and edges that connect them.  
→ Exactly one path between two nodes.

Path: Connected sequence of edges.



Rooted tree: One distinguished node is called the Root



Every node  $n$ , except Root, has ONE parent,  $p$ ,  
the first node on the path from  $n$  to the root.

$n$  is  $p$ 's child.

Root has NO parent.

A node can have ANY number of children.

Leaf: Node with NO children.

Siblings: Nodes with the same parent.

Ancestors: of a node  $n$  are the nodes on the path from  $n$  to the root (including  $n$  itself and the root)

If  $a$  is an ancestor of  $n$ ,  $n$  is a descendant

of a.

Length of a path : Number of edges on the path.  
Node to itself  $\rightarrow$  Path length = 0

Depth of node  $n$  : Length of the path from  $n$  to the Root. Depth of the Root = 0.

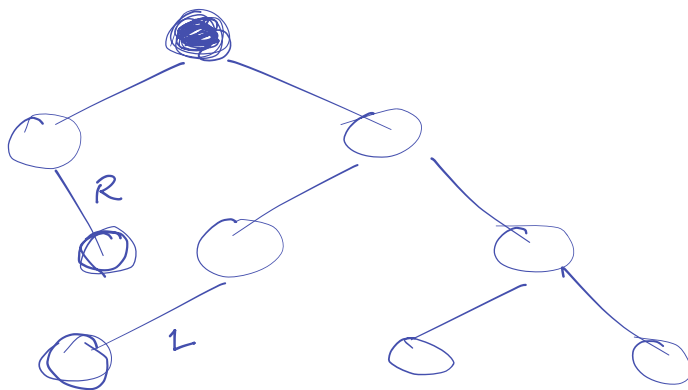
Height of node  $n$  : Length of path from  $n$  to its deepest descendent. Height of a leaf node = 0.

Height of a tree = Height of the Root

Subtree rooted at  $n$  : Tree formed by  $n$  and its descendants.

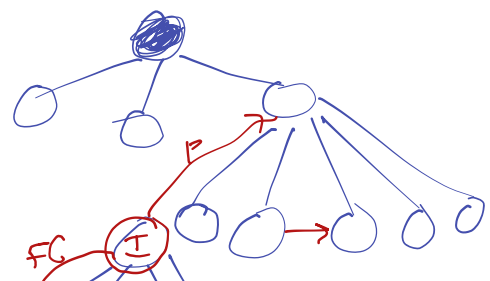
Binary tree : No node has  $> 2$  children  
 $\hookrightarrow \leq 2$  children

Every child is either the left or the right child  
(even if it is the ONLY child)



Representation of Rooted trees

- Item
- Parent
- Siblings are directly linked.



class TreeNode {

Object item;

TreeNode \* parent;

TreeNode \* firstChild;

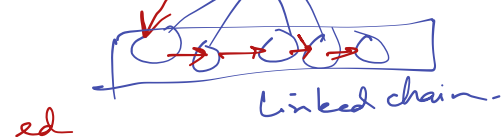
TreeNode \* nextSibling;

}

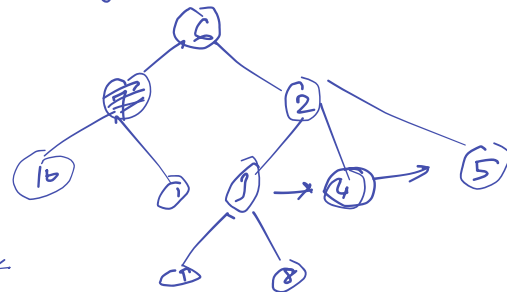
class RootedTree {

TreeNode \* Root;

}



eg. File system



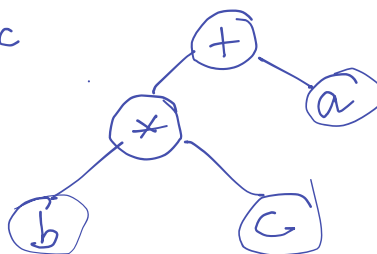
Traversal: A manner of visiting  
a node in a tree ONCE.

Operation: Print(name, level...),  
sum

Eg2.

$a + b * c$

Expression trees.



Node:

Structure/class

→ Collection of data

eg.3.

Attributes of a student

Key

List (Values)

→ Name, ID, year, department

key: used to locate a node.

Node

└ Key

└ Value

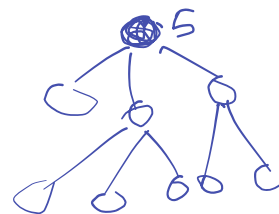
Traversals:

① Preorder traversal: Visit each node before recursively visiting its children, left to right. Root node is visited first.

class TreeNode {

void preorder() {

→ visit(); // Any action (eg. print) & cout << key()



if (firstChild != NULL)      << endl;

firstChild → preorder();

if (nextSibling != NULL)  
nextSibling → preorder();

} # of Nodes → N

Time: Each node is visited exactly once  
O(N)

Post order → visit the current Node LAST.

Inorder traversal → Undefined for trees  
where a node can have more than  
2 children.

Level-order traversal → Breadth first search.