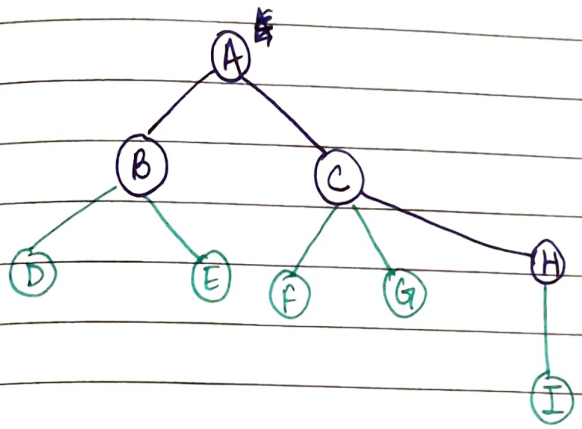
TREES (lecture-13)

```
typedef struct treenode {
    int element;
    struct treenode * firstchild;
    struct treenode * nextsibling;
} rootnode;
```



level 0

level 1

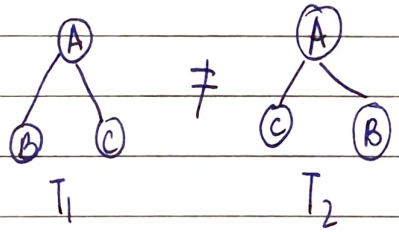
level 2

level 3

- A, B, C, H are internal nodes
- The depth (level) of E is 2
- The height of tree is 3
- The degree of node B (no. of children) is 2.

* Tree Degree is the maximum of node degrees.

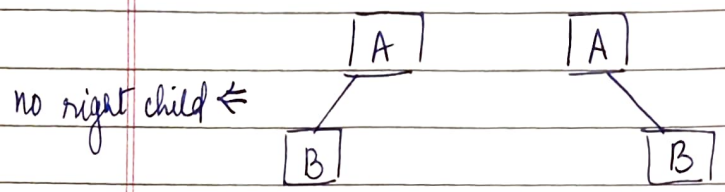
ORDERED TREE ⇒ An oriented tree in which the children of a node are somehow "ordered".



if T_1 & T_2 are ordered trees.

BINARY TREE ⇒ ordered tree with all nodes having at most 2 children.

- 0 or more nodes
- Each node - value (data)
  - ptr to left child (maybe NULL)
  - ptr to right child (maybe NULL)
- Maybe empty (NO NODES)
- Node with no ~~leaf ch~~ left child & no right child is called a leaf.

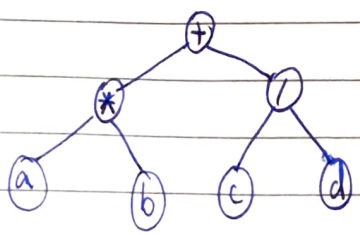


These 2 trees are different

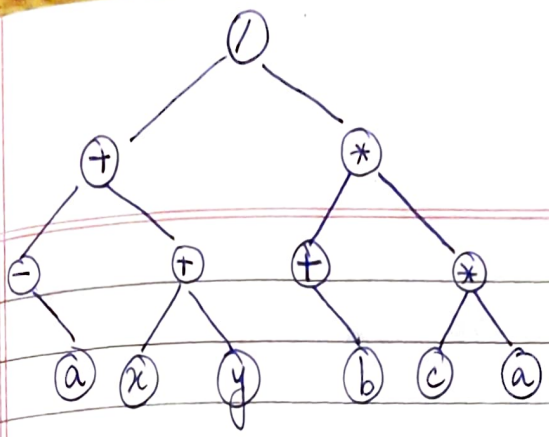
BINARY TREE	TREE
$(\text{degree of node})_{max.} = 2$	No limit on degree of node
subtrees are ordered	subtrees are NOT ordered.

Recursive Definition:- Binary tree is either a
- leaf
- an internal node (root) & 1/2 binary trees

For Expressions:-



⇒ (a*b) + (c/d)

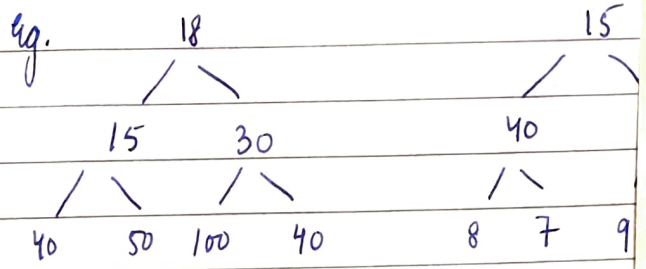


$((-a)+(x+y))/((+b)*(c*a))$

(1) Full Binary Tree - If every node has either 0 or 2 children.

(2) Complete Binary Tree

- if each level of the tree, except the last level have 2^l nodes, where 'l' is the level.
- if the last level doesn't have 2^l nodes, then all nodes in the last level should be left aligned.



Tree of height = h :-
No. of leaves = 2^h
No. of internal nodes = $2^h - 1$
Total nodes = $2^{h+1} - 1 \geq n$

Tree of 'n' nodes, :-
No. of leaves = $(n+1)/2$
height $\geq \log_2$ (no. of leaves)
Max^m height = $n-1$
(every node except leaf has 1 child)

(3) Perfect Binary Tree - All internal nodes ha
leaves are at the same level.

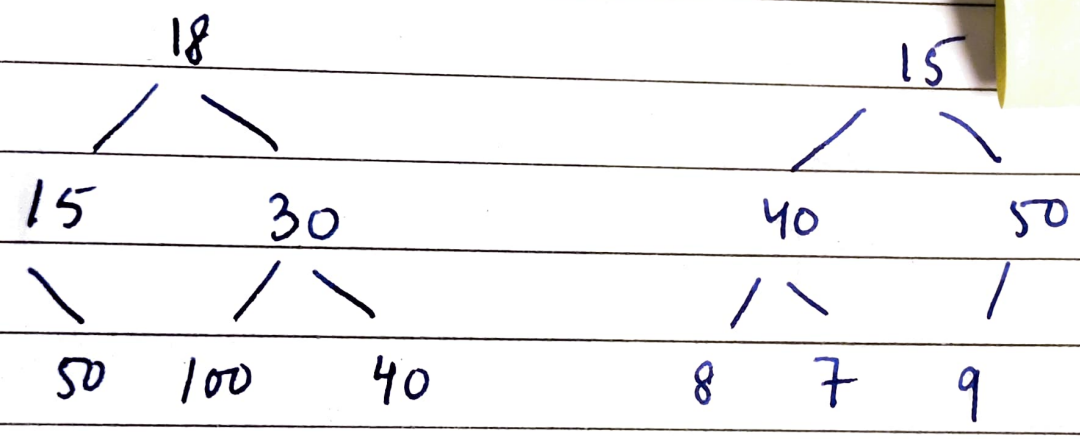
OPERATIONS ON TREES :-

1. Insert
2. Delete
3. Search
4. Traverse

inary Tree - If every node has either 0 or 2 children.

Binary Tree

each level of the tree, except the l
les, where 'l' is the level.
the last level doesn't have 2^l nod
level should be left aligned.



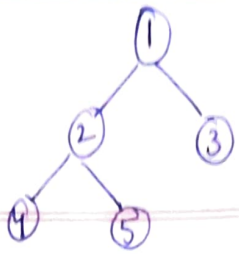
ary Tree - All internal nodes have 2 children & all leaves are at the same level.

S ON TREES:-

t

e

l.



Trees can be traversed in diff. ways:-

Depth First Traversal

(a) Inorder (Left, Root, Right)	4	2	5	1	3
(b) Preorder (Root, Left, Right)	1	2	4	5	3
(c) Postorder (Left, Right, Root)	4	5	2	3	1

Breadth First or level order Traversal : 1 2 3 4 5

↓

For each node, first the node is visited, then it's child nodes are put in a FIFO queue.

Level Order Traversal

Algorithm:- 1) Create an empty queue q
2) temp_node = root
3) While temp_node is NOT NULL
   a) print temp_node -> data
   b) Enqueue temp_node's children (first left then right) to q.
   c) Dequeue a node from q & assign it's value to temp_node

```
void level_order (node* root) {
    node* temp = root;
    node * queue [Q_SIZE] = {NULL};
    int front = rear = 0;
    while (temp != NULL) {
        printf ("%d", temp -> data);
        if (temp -> left != NULL) {
            queue[rear] = temp -> left;
            rear++;
        }
        if (temp -> right != NULL) {
            queue [rear] = temp -> right;
            rear++;
    }
```

```
        temp = queue[front++];
    }
}
```

Inorder Traversal (Left, Root, Right)

Algorithm :- if (N = NULL) N → current node
return
else
Traverse left subtree i.e. call Inorder (N → left child)
Visit root node (N)
Traverse right subtree i.e. call Inorder (N → right child)

Similarly for Preorder & Postorder!

⇒ CONSTRUCTING A TREE

Inorder : D B F E (A) G C L J H K
Left Root Right

Preorder : A B D E F C G H J L K
Root Root of Left Sub-Tree Root of Right Sub-Tree