**tree traversal** : refers to the process of visiting (checking and/or updating) each node in a [tree data structure](https://en.wikipedia.org/wiki/Tree_(data_structure)), **exactly once**

***level-order***: where we visit every node on a level before going to a lower level.

Leaf node height is 0, hence if there is no tree then it has to return -1

Tree with one node height is 0, with no nodes it is -1

Empty tree: number of nodes- return 0

Height –return -1

There is only one kind of breadth-first traversal--the level order traversal

There are three different types of depth-first traversals, : Pre, post and in

Unlike the other traversal methods, a recursive version does not exist for traversing the tree, for height and cout recursive exist

a stack is replaced with a FIFO queue.

**height** of a tree as the length of the longest path (each line counts as one step) from a root to one of its descendants. Alternatively, we can define the **depth** of a node as the number of ancestors it has, and then define the height of a tree as the largest depth of any of its nodes

HEIGHT

if (h == 0) return -1;

**if (t->left == null && t->right == null) //leaf check**

**return 0;**

**else**

**{**

int u = height(h->l);

int v = height(h->r);

if (u > v) return u+1;

else return v+1;

}

PRINT

void printLevelOrder(struct node\* root)

{

int h = height(root);

int i;

for (i=1; i<=h; i++)

printGivenLevel(root, i);

}

void printGivenLevel(struct node\* root, int level)

{

if (root == NULL)

return 0;

if (level == 1)

printf("%d ", root->data);

else if (level > 1)

{

printGivenLevel(root->left, level-1);

printGivenLevel(root->right, level-1);

}

}

COUNT

int count(link h)

{

if (h == 0) return 0;

return count(h->l) + count(h->r) + 1;

}