Binary Trees

A tree whose element have at most 2 children is called binary trees. Since each element in a binary tree can have only 2 children, we name them left child & right child.

A Binary Tree node contains the following parts :

- Data
- Pointer to left child
- Pointer to right child

A simple binary tree :

```
#include<bits/stdc++.h>
using namespace std;

struct Node{
    int data;
    struct Node* left;
    struct Node* right;

    Node(int x){
        data=x;
        left=NULL;
        right=NULL;
    }
};

int main(){
    Node* root = new Node(1);
    /* The tree after above statement

             1
            / \
          NULL NULL
    */
    root->left = new Node(2);
    root->right = new Node(3);
    /* The tree after above statement

             1
            / \
           2   3
    */
}
```

Properties

1. The maximum number of nodes at level 'l' of a binary tree is 2^l.

A level is the number of parent nodes corresponding to a given node of the tree. Level of the root is zero.

This can be proved by induction.

For root, $l=0$, number of nodes $= 2^0 = 1$

For level, $l=1$, number of nodes $= 2^1 = 2$

Since root is on zeroth level, first level will have at max 2 nodes because ~~it or~~ in binary tree every node has at most 2 children.

For, Level l, no of nodes $= 2^l$

2. The maximum number of nodes in a binary tree of height h is $2^h - 1$

If root node is considered at height 1.

Then,

Max no of nodes in binary tree of height h is

$$1 + 2 + 4 + \ldots\ldots + 2^{h-1}$$

This is geometric series $a + a\cdot r + a\cdot r^2 + \ldots\ldots + a\cdot r^{h-1}$ $\quad \dfrac{a(r^n-1)}{r-1}$

$r=2, n=h, a=1$

$$= \frac{1\cdot(2^h-1)}{2-1} = 2^h - 1$$

If root node is considered at height 0

Then,

Max no of nodes in binary tree of height h is

$r=2, n=h+1, a=1$, $1 + 2 + 4 + \ldots\ldots + 2^h$

$$= \frac{1\cdot(2^{h+1}-1)}{2-1} = 2^{h+1} - 1$$

3) In a Binary Tree with N nodes, minimum possible height or the minimum number of levels is $\log_2(N+1)$

For root node at height 0,

Total no of nodes $= 2^{h+1} - 1$

$$2^{h+1} - 1 = N$$

$$2^{h+1} = N + 1$$

$$(h+1)\log 2 = \log(N+1)$$

$$h = \log_2(N+1) - 1$$

4) In a Binary Tree with 'L' leaves, it has atleast $\lceil \log_2 L \rceil + 1$ levels for root at level 1.

A binary tree with maximum number of leaves & minimum number of levels when all level are fully filled.

No of leaves, L at level $l = 2^{l-1}$

$$L = 2^{l-1}$$

$$(l-1)\log 2 = \log L$$

$$l = \log_2 L + 1$$

5) If a binary tree has 0 or 2 children, then number of leaf nodes are always one more than nodes with 2 children or internal nodes.

$$L = I + 1, \text{ where } L \rightarrow \text{Leaf Nodes},\ I \rightarrow \text{Internal Nodes}$$

Total Nodes = Internal + Leaf

Leaf = Total − Internal

$$2^{h-1} = (2^h - 1) - \text{Internal}$$

$$I = (2^h - 1) - (2^{h-1})$$

$$I = 2^{h-1}(2-1) - 1$$

$$I = 2^{h-1} - 1 \Rightarrow I + 1 = 2^{h-1} (\text{Leaf Node})$$

Leaf Node = I + 1

Types of Binary Tree

Full Binary Tree

A binary tree is a full binary tree if every node has 0 or 2 children. We can also say a binary tree is full it all nodes has two children except leaf nodes. The examples of full binary tree is :

```
          10
        /    \
       11      12
     / \      / \
    13  14    15  16


          10
        /    \
       11      12
     / \
    13  14


          10
        /    \
       11      12
              / \
             15  16
```

Complete Binary Tree

A binary tree is a complete binary tree if all the levels are completely filled or the last unfilled level has all keys on left . The examples of complete binary tree are :-

```
          10
        /    \
       11      12
     /
    13

          10
        /    \
       11      12
     / \      / \
    13  14    15  16
```

Perfect Binary Tree

A binary tree is perfect if all the levels are completely filled or we can say that all internal nodes have two children & leaf nodes are at the same level.

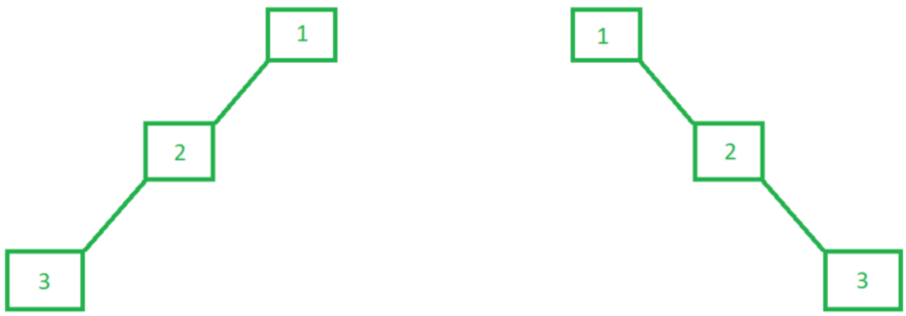
```
               10
             /     \
           11       12
          / \      / \
        13  14    15  16


               10
             /     \
           11       12
```

Skewed Binary Tree

A skewed binary tree is a binary tree in which all nodes have only one child or no child.

- Left Skewed Binary Tree
- Right Skewed Binary Tree



LEFT SKEWED RIGHT SKEWED

Count of Binary Trees

A Binary Tree is labeled if every node is assigned a label and a Binary Tree is unlabelled if nodes are not assigned any label.

```
Below two are considered same unlabelled trees
    o                 o
  /   \             /   \
 o     o           o     o

Below two are considered different labelled trees
    A                C
  /   \             /  \
 B     C           A    B
```

How many different Unlabelled Binary Trees can be there with n nodes?

It is equal to catalan number.

```
T(n) = (2n)! / (n+1)!n!
```

*The number of Binary Search Trees (BST) with n nodes is also the same as the number of unlabelled trees. The reason for this is simple, in BST also we can make any key a root, If the root is i'th key in sorted order, then i-1 keys can go on one side, and (n-i) keys can go on another side. *

How many labeled Binary Trees can be there with n nodes?
To count labeled trees, we can use the above count for unlabeled trees. The idea is simple, every unlabeled tree with n nodes can create n!.

```
Number of Labelled Trees = (Number of unlabelled trees) * n!
                       = [(2n)! / (n+1)!n!]  × n!
```