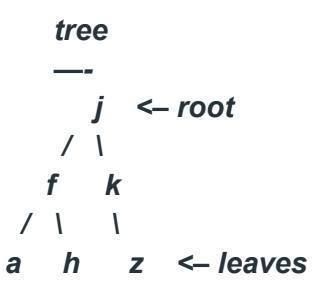
What is a tree data structure?

There are two major types of data structures:

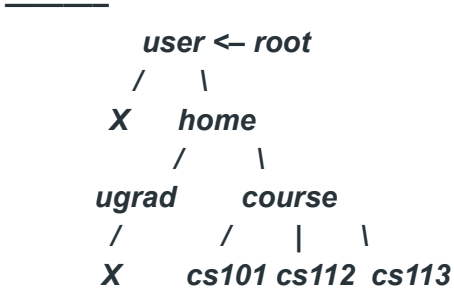
- **Linear**
- **Non-Linear**

A tree is a popular data structure that is non-linear in nature. Unlike other data structures like array, stack, queue, and linked list which are linear in nature, a tree represents a hierarchical structure. The ordering information of a tree is not important. A tree contains nodes and 2 pointers. These two pointers are the left child and the right child of the parent node. Let us understand the terms of the tree in detail.

- **Root: The root of a tree is the topmost node of the tree that has no parent node. There is only one root node in every tree.**
- **Edge: Edge acts as a link between the parent node and the child node.**
- **Leaf: A node that has no child is known as the leaf node. It is the last node of the tree. There can be multiple leaf nodes in a tree.**
- **Depth: The depth of the node is the distance from the root node to that particular node.**
- **Height: The height of the node is the distance from that node to the deepest node of the tree.**
- **Height of tree: The Height of the tree is the maximum height of any node.**
-



Why Use Trees?
1. One reason to use trees might be because you want to store information that naturally forms a hierarchy. For example, the file system on a computer:



2. Trees (with some ordering e.g., BST) provide moderate access/search (quicker than Linked List and slower than arrays).
3. Trees provide moderate insertion/deletion (quicker than Arrays and slower than Unordered Linked Lists).
4. Like Linked Lists and unlike Arrays, Trees don't have an upper limit on the number of nodes as nodes are linked using pointers.

Main applications of trees include:

1. **Manipulate hierarchical data.**
2. **Make information easy to search (see tree traversal).**
3. **Manipulate sorted lists of data.**
4. **As a workflow for compositing digital images for visual effects.**
5. **Router algorithms**
6. **Form of multi-stage decision-making (see business chess).**

Binary Tree: A tree whose elements have at most 2 children is called a binary tree. Since each element in a binary tree can have only 2 children, we typically name them the left and right child.

Binary Tree Representation: A tree is represented by a pointer to the topmost node of the tree. If the tree is empty, then the value of the root is NULL.
A Tree node contains the following parts.

1. **Data**
2. **Pointer to the left child**
3. **Pointer to the right child**

In C, we can represent a tree node using structures. In other languages, we can use classes as part of their OOP feature. Below is an example of a tree node with integer data.

```
// Structure of each node of the tree
struct node
{
    int data;
    struct node* left;
    struct node* right;
};
```

Basic Operation On Binary Tree:

- **Inserting an element.**
- **Removing an element.**
- **Searching for an element.**
- **Traversing an element. There are three types of traversals in a binary tree which will be discussed ahead.**

Auxiliary Operation On Binary Tree:

- **Finding the height of the tree**
- **Find the level of the tree**
- **Finding the size of the entire tree.**

Applications of Binary Tree:

- **In compilers, Expression Trees are used which is an application of binary tree.**
- **Huffman coding trees are used in data compression algorithms.**
- **Priority Queue is another application of binary tree that is used for searching maximum or minimum in O(logN) time complexity.**

Binary Tree Traversals:

- **PreOrder Traversal: Here, the traversal is: root – left child – right child. It means that the root node is traversed first then its left child and finally the right child.**
- **InOrder Traversal: Here, the traversal is: left child – root – right child. It means that the left child is traversed first then its root node and finally the right child.**
- **PostOrder Traversal: Here, the traversal is: left child – right child – root. It means that the left child is traversed first then the right child and finally its root node.**

Let us traverse the following tree with all the three traversal methods:

```
Tree
_____________________

                1 //Root Node
               / \
              2    3
             / \  / \
            4  5  6  7 //Leaf Nodes
```

PreOrder Traversal of the above tree: 1-2-4-5-3-6-7
InOrder Traversal of the above tree: 4-2-5-1-6-3-7
PostOrder Traversal of the above tree: 4-5-2-6-7-3-1

Properties of Binary Tree

1) The maximum number of nodes at level 'l' of a binary tree is 2^l.
Here level is the number of nodes on the path from the root to the node (including root and node). Level of the root is 0.
This can be proved by induction.
For root, l = 0, number of nodes = $2^0 = 1$
Assume that the maximum number of nodes on level 'l' is 2^l
Since in Binary tree every node has at most 2 children, next level would have twice nodes, i.e. $2 * 2^l$

2) The Maximum number of nodes in a binary tree of height 'h' is $2^h – 1$.
Here the height of a tree is the maximum number of nodes on the root to leaf path. Height of a tree with a single node is considered as 1.
This result can be derived from point 2 above. A tree has maximum nodes if all levels have maximum nodes. So maximum number of nodes in a binary tree of height h is $1 + 2 + 4 + .. + 2^{h-1}$. This is a simple geometric series with h terms and sum of this series is $2^h–1$.
In some books, the height of the root is considered as 0. In this convention, the above formula becomes $2^{h+1} – 1$

3) In a Binary Tree with N nodes, minimum possible height or* the *minimum number of levels is $Log_2(N+1)$.

There should be at least one element on each level, so the height cannot be more than N.
A binary tree of height 'h' can have maximum $2^h – 1$ nodes (previous property). So the number of nodes will be less than or equal to this maximum value.

$N <= 2^h - 1$
$2^h >= N+1$
$\log_2(2^h) >= \log_2(N+1)$ (Taking log both sides)
$h \log_2 2 >= \log_2(N+1)$ (h is an integer)
$h >= \lceil \log_2(N+1) \rceil$

So the minimum height possible is $\lceil \log_2(N+1) \rceil$

4) A Binary Tree with L leaves has at least $\lceil Log_2 L \rceil + 1$ levels.
A Binary tree has the maximum number of leaves (and a minimum number of levels) when all levels are fully filled. Let all leaves be at level l, then below is true for the number of leaves L.

$L <= 2^{l-1}$ [From Point 1]
$l = \lceil \log_2 L \rceil + 1$
where l is the minimum number of levels.

5) In Binary tree where every node has 0 or 2 children, the number of leaf nodes is always one more than nodes with two children.

L = T + 1
Where L = Number of leaf nodes
T = Number of internal nodes with two children
Proof:
No. of leaf nodes (L) i.e. total elements present at the bottom of tree = 2^{h-1} (h is height of tree)
No. of internal nodes = {total no. of nodes} - {leaf nodes} = { 2^h - 1 } - {2^{h-1}} = 2^{h-1} (2-1) - 1 = 2^{h-1} - 1
So , L = 2^{h-1}
T = 2^{h-1} - 1
Therefore L = T + 1
Hence proved

6) In a non empty binary tree, if n is the total number of nodes and e is the total number of edges, then e = n-1

Every node in a binary tree has exactly one parent with the exception of root node. So if n is the total number of nodes then n-1 nodes have exactly one parent. There is only one edge between any child and its parent. So the total number of edges is n-1

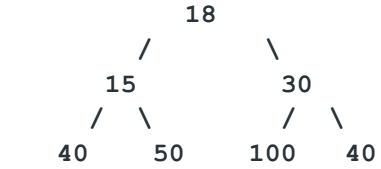
Types of Binary Trees

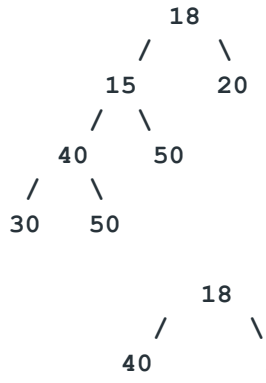
The following are common types of Binary Trees.

Full Binary Tree:

A Binary Tree is a full binary tree if every node has 0 or 2 children. The following are the examples of a full binary tree. We can also say a full binary tree is a binary tree in which all nodes except leaf nodes have two children.

A full Binary tree is a special type of binary tree in which every parent node/internal node has either two or no children. It is also known as a proper binary tree.





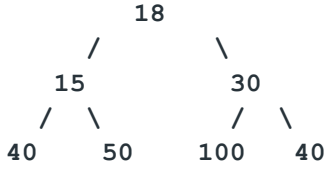
Complete Binary Tree:-

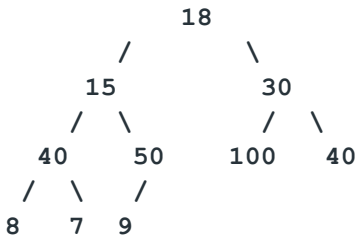
A Binary Tree is a Complete Binary Tree if all the levels are completely filled except possibly the last level and the last level has all keys as left as possible.

A complete binary tree is just like a full binary tree, but with two major differences:

- **Every level must be completely filled**
- **All the leaf elements must lean towards the left.**
- **The last leaf element might not have a right sibling i.e. a complete binary tree doesn't have to be a full binary tree.**

The following are examples of Complete Binary Trees.





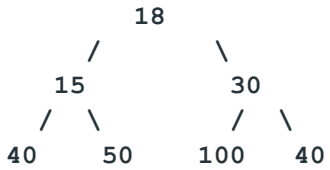
Practical example of Complete Binary Tree is [Binary Heap](#).

Perfect Binary Tree:-

A Binary tree is a Perfect Binary Tree in which all the internal nodes have two children and all leaf nodes are at the same level.

The following are the examples of Perfect Binary Trees.

A perfect binary tree is a type of binary tree in which every internal node has exactly two child nodes and all the leaf nodes are at the same level.

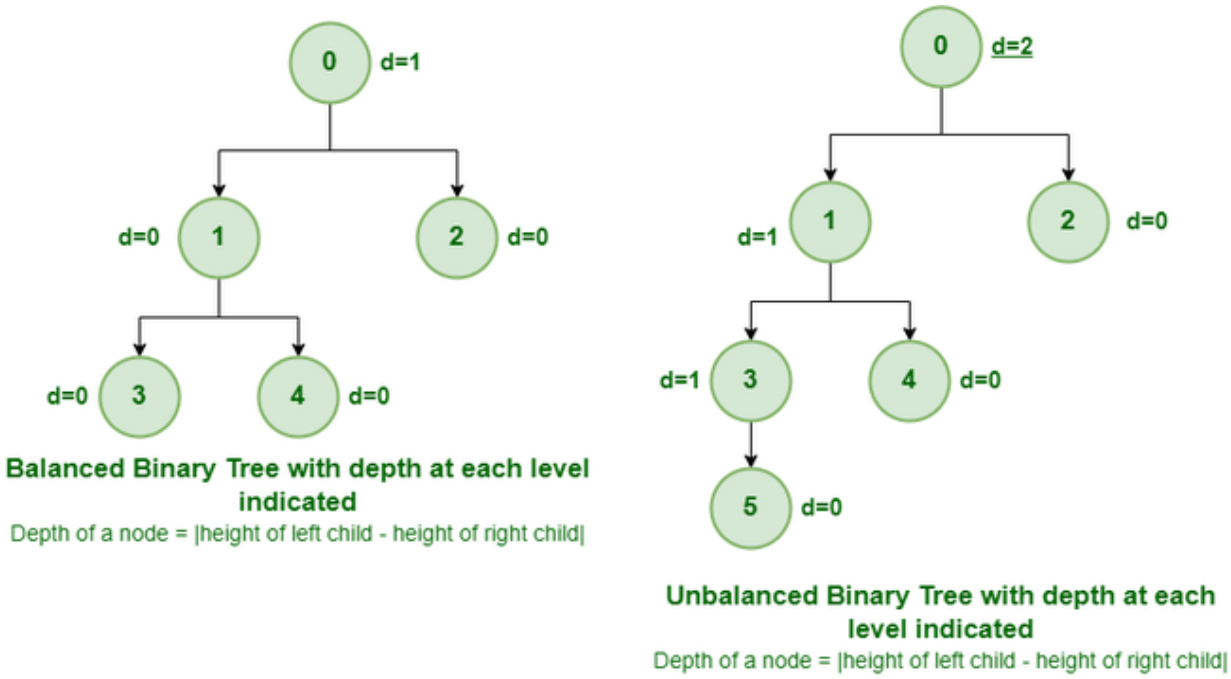


In a Perfect Binary Tree, the number of leaf nodes is the number of internal nodes plus 1

L = I + 1 Where L = Number of leaf nodes, I = Number of internal nodes.

A Perfect Binary Tree of height h (where the height of the binary tree is the number of edges in the longest path from the root node to any leaf node in the tree, height of root node is 0) has $2^{h+1} - 1$ node.

An example of a Perfect binary tree is ancestors in the family. Keep a person at root, parents as children, parents of parents as their children.
Balanced Binary Tree:-
A binary tree is balanced if the height of the tree is O(Log n) where n is the number of nodes. For Example, the AVL tree maintains O(Log n) height by making sure that the difference between the heights of the left and right subtrees is at most 1. Red-Black trees maintain O(Log n) height by making sure that the number of Black nodes on every root to leaf paths is the same and there are no adjacent red nodes. Balanced Binary Search trees are performance-wise good as they provide O(log n) time for search, insert and delete.



Example of Balanced and Unbalanced Binary Tree

It is a type of binary tree in which the difference between the height of the left and the right subtree for each node is either 0 or 1. In the figure above, the root node having a value 0 is unbalanced with a depth of 2 units.

A degenerate (or pathological) tree:-
A Tree where every internal node has one child. Such trees are performance-wise same as linked list.
A degenerate or pathological tree is the tree having a single child either left or right.

```
    10
     /
   20
    \
    30
     \
     40
```

Skewed Binary Tree:-

A skewed binary tree is a pathological/degenerate tree in which the tree is either dominated by the left nodes or the right nodes. Thus, there are two types of skewed binary tree: left-skewed binary tree and right-skewed binary tree.