Lecture

NOTE: FOR FURTHER DETAILS AND MORE COMPREHENSIVE STUDY, PLEASE SEE RECOMMENDED BOOKS OR INTERNET.

Trees

A tree is a ***nonlinear hierarchical*** data structure that consists of ***nodes*** connected by ***edges***.

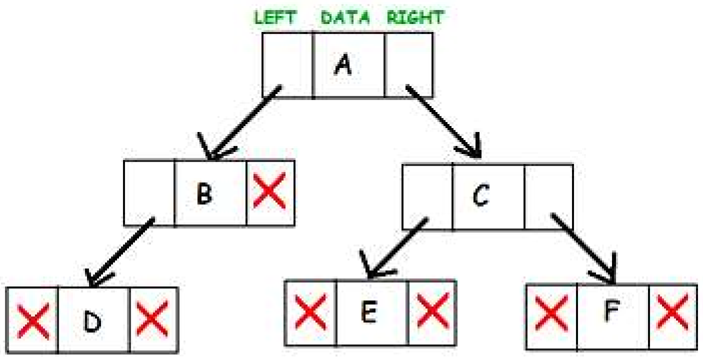
Introduction to Binary Trees

A binary tree is a hierarchical data structure in which each node has at most ***two children*** generally referred as ***left child*** and ***right child***.

Each node contains three components:

- ***Data***
- ***Reference*** to ***left*** subtree
- ***Reference*** to ***right*** subtree

The topmost node in the tree is called the ***root***. An empty tree is represented by ***NULL*** pointer. A representation of binary tree is shown:



Binary Tree: Common Terminologies

Following are the common terminologies used for binary trees.

Root:

A ***top most node*** in a tree.

Parent:

Every node (excluding a root) in a tree is ***connected upward*** to exactly one other node. This node is called a parent.

Child:

A node directly connected to another node when moving down away from the root.

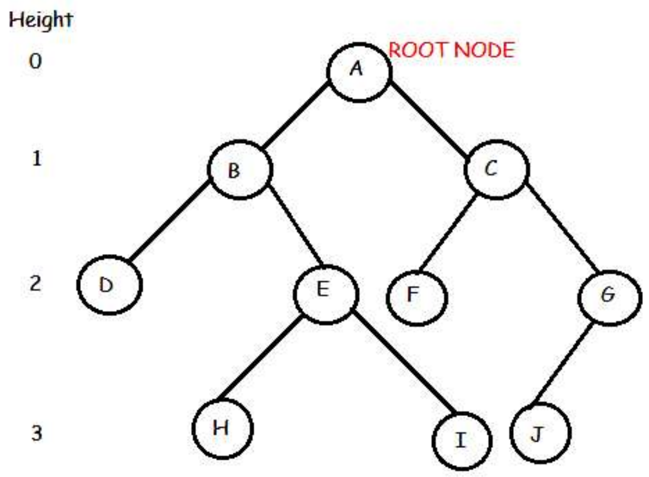
Leaf/External node:

A Node with no children.

Internal node:
A Node with atleast one children.

Depth of a node:
Number of edges from root to the node.

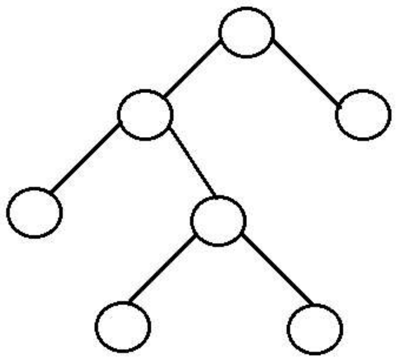
Height of a node:
Number of edges from the node to the deepest leaf. Height of the tree is the height of the root.



In the above binary tree we see that root node is ***A***. The tree has 10 nodes with 5 internal nodes, i.e, ***A,B,C,E,G*** and 5 external nodes (leaf), i.e, ***D,F,H,I,J***. The height of the tree is ***3***. ***B*** is the parent of ***D*** and ***E*** while ***D*** and ***E*** are children of ***B***.

Types of Binary Trees (Based on Structure)
Following are the types of binary trees that are very important to understand. These types provide the knowledge base for most of the tree operations and further implementation of the variants of the binary trees.

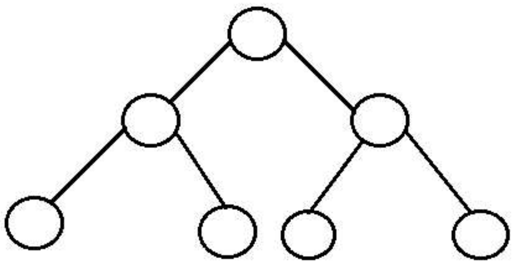
Full binary tree:
It is a tree in which every node in the tree has either 0 or 2 children.



The number of nodes ***n***, in a full binary tree is atleast $n = 2^h - 1$, and atmost $n = 2^{h+1} - 1$, where ***h*** is the height of the tree.
The number of leaf nodes ***l***, in a full binary tree is ***L+1*** where ***L*** is the number of internal nodes, i.e, $l = L+1$.

Perfect binary tree:

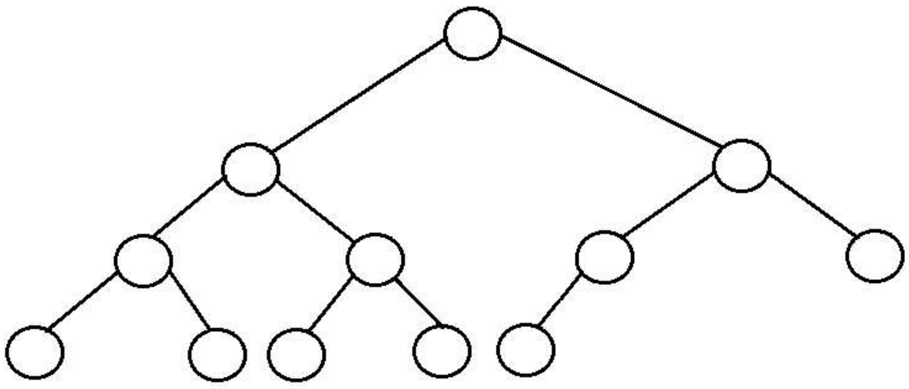
It is a binary tree in which all internal nodes have two children and all leaves have the same depth or same level.



A perfect binary tree with ***l*** leaves has $n = 2l - 1$ nodes.
In perfect full binary tree, $l = 2^h$ and $n = 2^{h+1} - 1$ where, ***n*** is number of nodes, ***h*** is height of tree and ***l*** is number of leaf nodes.

Complete binary tree:

It is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.

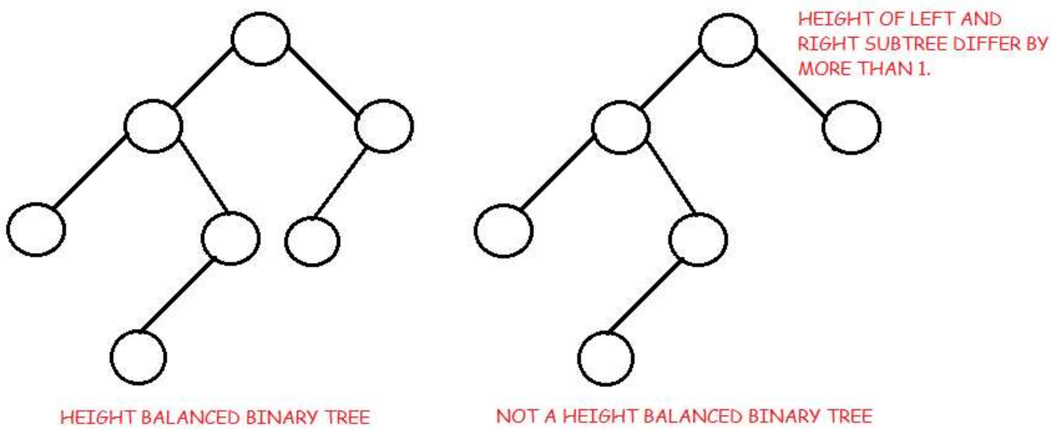


Balanced binary tree:

A binary tree is height balanced if it satisfies the following constraints:

1. The left and right subtrees' heights differ by ***at most one***, AND
2. The left subtree is ***balanced***, AND
3. The right subtree is ***balanced***

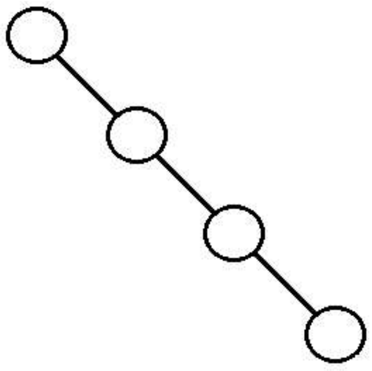
An empty tree is height balanced.


HEIGHT OF LEFT AND RIGHT SUBTREE DIFFER BY MORE THAN 1.
HEIGHT BALANCED BINARY TREE
NOT A HEIGHT BALANCED BINARY TREE

The height of a balanced binary tree is ***O(Log n)*** where ***n*** is number of nodes.

Degenarate tree:

It is a tree is where each parent node has only one child node. It behaves like a linked list.



Array Implementation of Binary Tree

Trees can be represented in two ways:

1. ***Dynamic Node Representation (Linked-List Representation).***
2. ***Array Representation (Sequential Representation).***

We are going to talk about the sequential representation of the trees.

To represent tree using an array, numbering of nodes can start either from ***0–(n-1)*** or ***1–n***.

In this example we will start from ***1–n***.

The ***root*** node is always inserted at ***1*** index in an array. Then the left child and right child are inserted as follows:

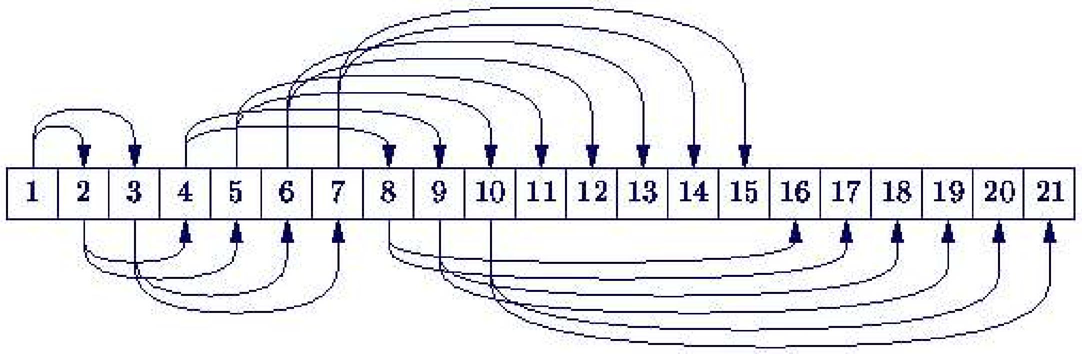
- If the parent is at index ***i*** then left child will be at ***2i***.
- If the parent is at index ***i*** then right child will be at ***2i + 1***.

So we can insert nodes in an array as per following formula:

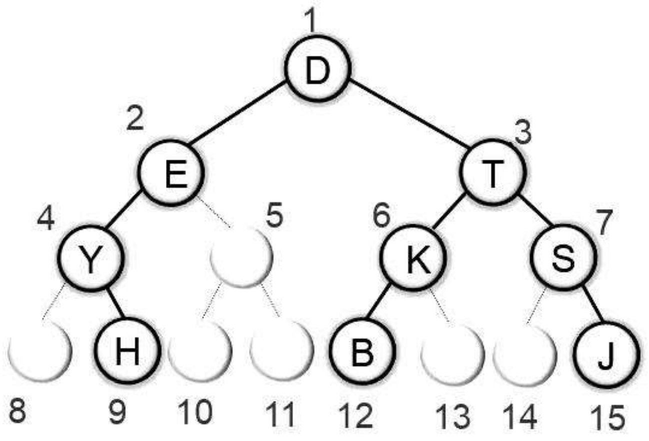
Parent = i

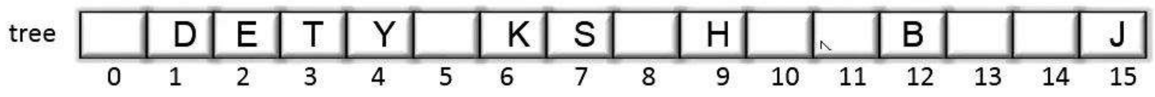
Left child = 2i

Right child = 2i + 1



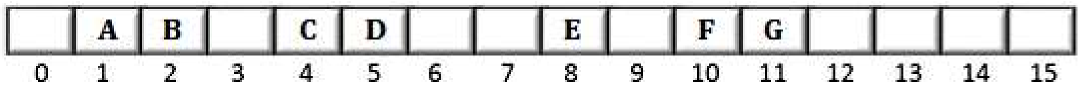
Now, consider we have the following tree what will be the array representation of the following tree?





tree		D	E	T	Y		K	S		H			B			J
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Now let's create the tree from array



	A	B		C	D			E		F	G				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15