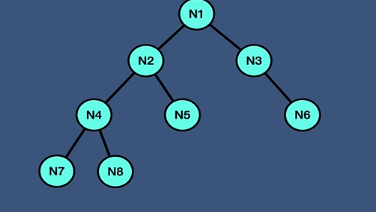
What is a Tree?

A tree is a nonlinear data structure with hierarchical relationships between its elements having any cycle, it is basically revered from a real-life tree.

What are the properties?

1. Represent hierarchical data.
2. Each node has two components, data, and a link to its subcategory.
3. Base category and subcategories under it.



Each data has a reference to its parent. And the root node has subcategory.

Why Tree?

Other data structure such as arrays, linked list, stack, queue are linear data structures that store data sequentially, in order to perform an operation in a linear data structure the time complexity increases with data size.

But in Tree;

1. Quicker and easier access to the data
2. Store hierarchical data, like folder structure, organization structure, XNL/HTML data.
3. There are many different types of data structure which performs better in various situations :
4. Binary Search tree, AVL, red black tree, trie

What is Tree Terminology?

Root: is node which has no parent

Edge: is a link between the parent and child.

Leaf: is node which does not have any children.

Siblings: the children of same parent

Ancestor: parent or grandparent, great grandparents of a nod.

Depth of node: a length of the path from root to node.

Height of a node: a length of the node to the deepest node

Depth of tree: depth of root node

Height of the tree: height of root node

What is a Binary Tree?

Binary tree is the data structure in which each node has at most two children, often referred to as the left and right children.

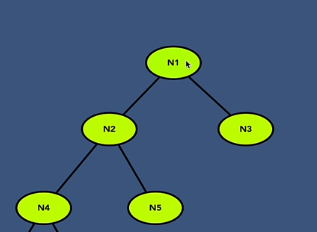
Binary tree is a family of data structure (BST, Heap tree, red black tree, syntax tree)

Huffman coding problem, heap priority problem and expression parsing problems can be solved efficiently using binary trees.

Types of Binary Tree:

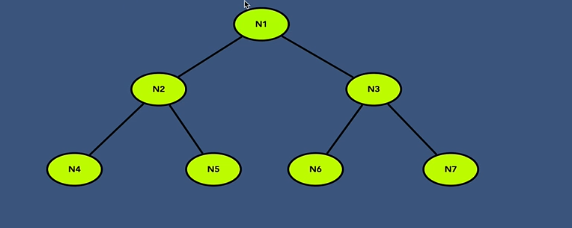
1. Full binary tree

Each node of binary tree has 0 or 2 children not 1



1. Perfect binary tree

All non – leaf nodes have two children

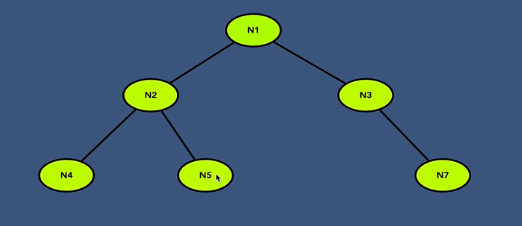


1. Complete binary tree

All levels are completely full except the last level.

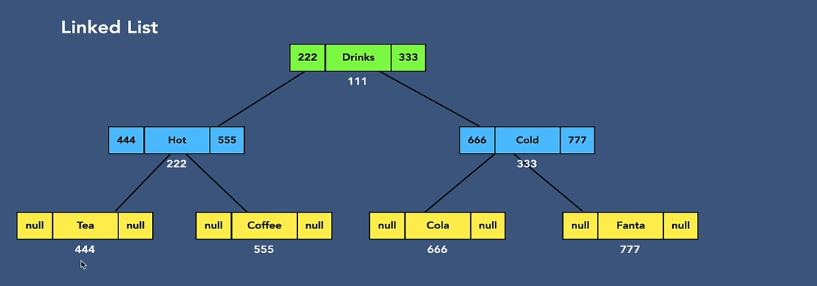
1. Balanced binary tree

Each leaf is not more than a certain distance from the root node than any other leaf.

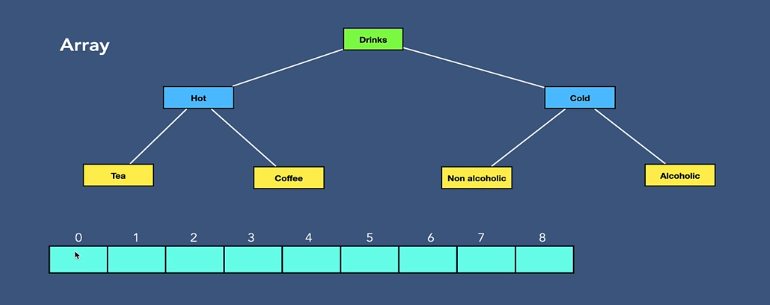


Binary tree can be presented by:

1. Linked List



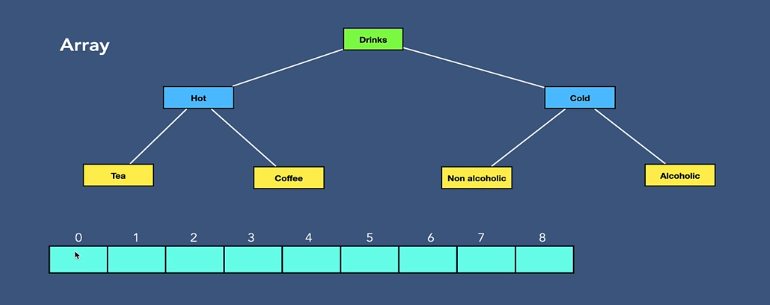
1. Array



The first cell will be skipped and we use formula

For left child = cell[2x] gives us the index number

Right child = cell[2x+1]

Root node stays at cell 1 

Traversal over the binary tree?

1. Depth first search
2. Preorder traversal

First, we visit the root node, then we visit the left subtree, And then we visit the right subtree. Time and space complexity :O(N)

1. In order traversal

First start from the left subtree and then the root node and continue in right subtree.

1. Post order traversal

Start from left subtree and the right subtree and then the root.

1. Breadth first search
2. Level order traversal

We do the search level by level.

Insert in Binary tree:

A root node is full.

The tree exists and we must look for a first vacant place. We are using level order traversal and checking to find an empty place and son one till we find a place.

In deleting:

1. Find the node.
2. Find deepest node.
3. Set deepest node value to current node.
4. Delete deepest node

Binary tree (Array vs linked list).

