Introduction to Binary Tree
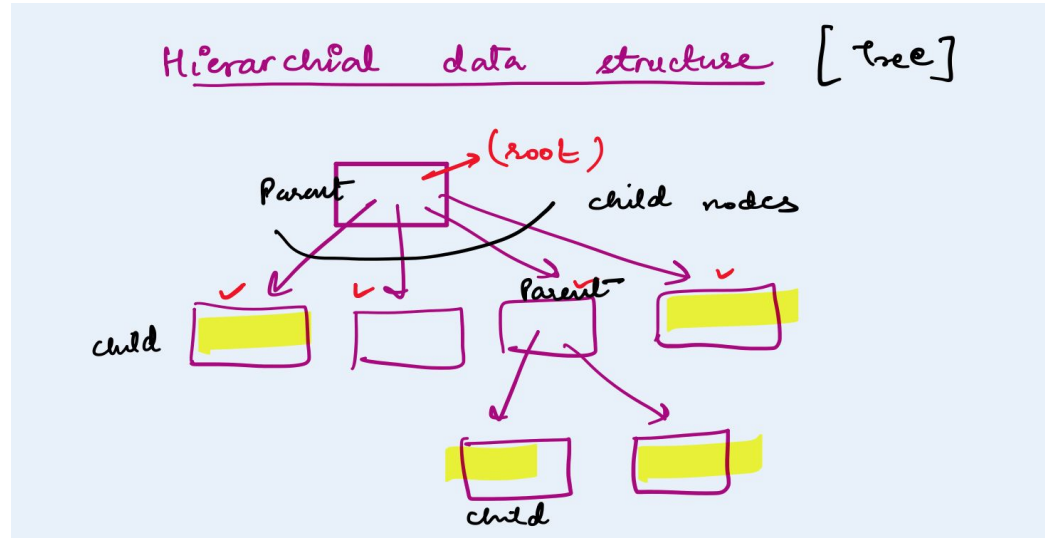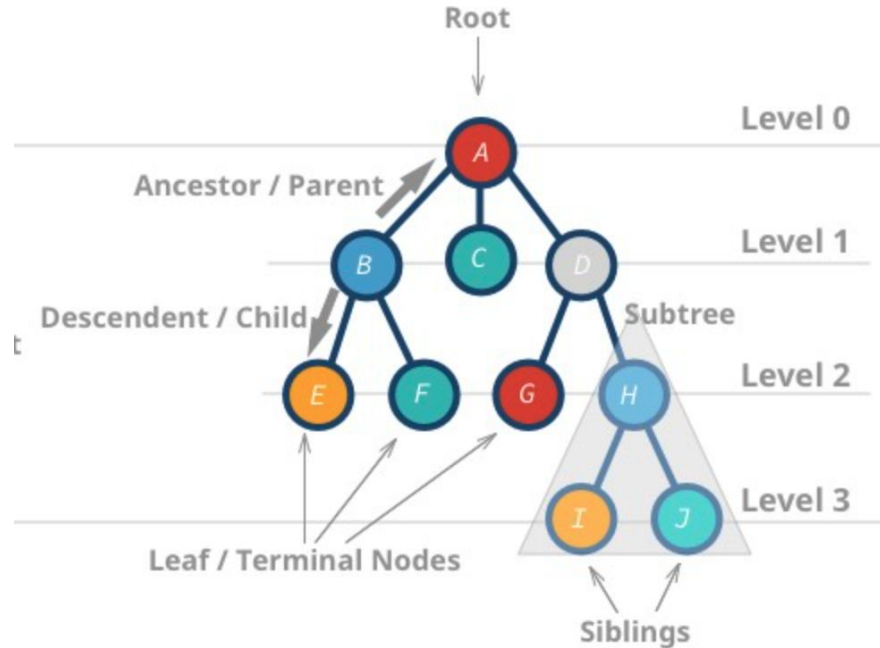
Hierarchical Data Structure

So far we have been studying linear data structures like Arrays, LinkedList and Queues. But the data that we need to store is not always going to be linear in nature, sometimes it is hierarchical depicting a parent child relationship. In Those situations, we use a hierarchical data structure like Tree.
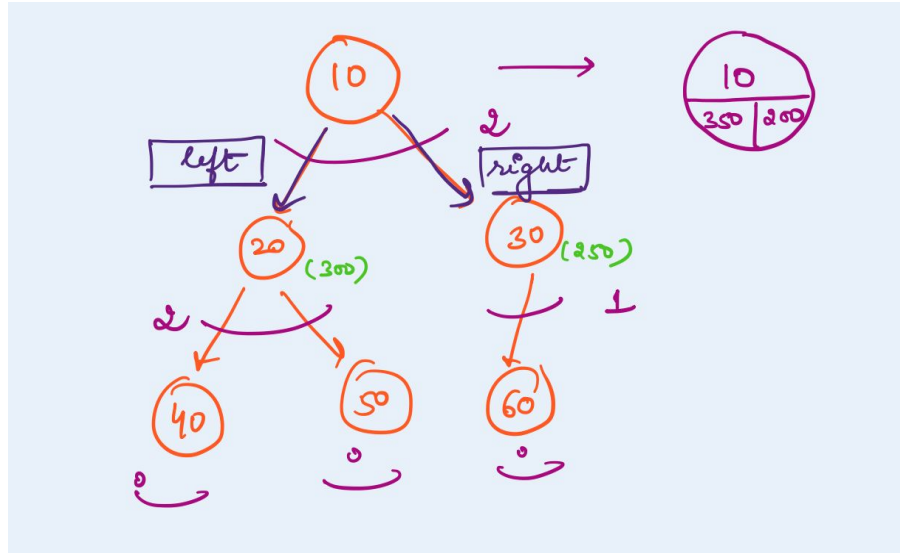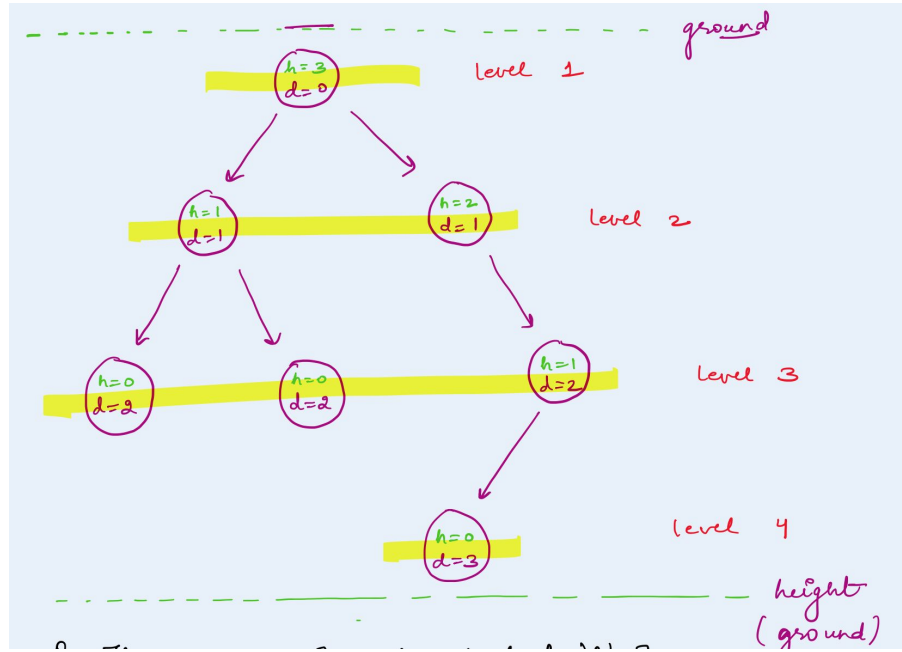
Hierarchical Data Structure

Binary Tree

The binary tree is the kind of tree in which most two children can be found for each parent. The kids are known as the left kid and right kid.
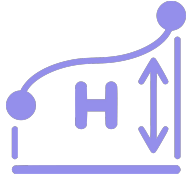
Binary Tree

Common Terminologies to use in Trees

Binary Tree

Height:

Height of node – The height of a node is the number of edges on the longest downward path between that node and a leaf.

Depth:

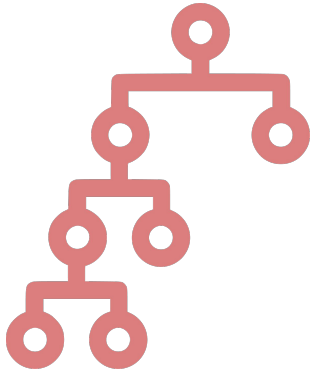Depth –The depth of a node is the number of edges from the node to the tree's root node.

Level:

Level – The level of a node is defined by 1 + the number of connections between the node and the root.

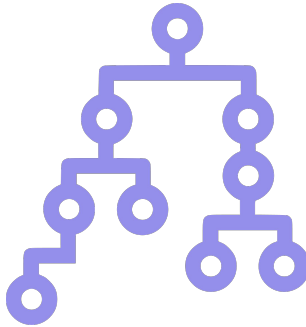Now that we know some terminologies, let's answer a few questions.

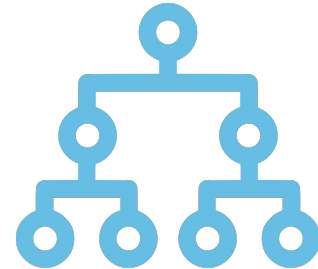# Binary Tree

Types of Binary Tree
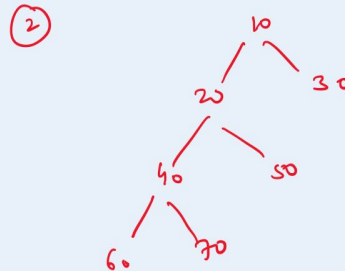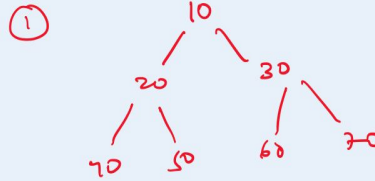
Full Binary Tree

Complete Binary Tree

Perfect Binary Tree
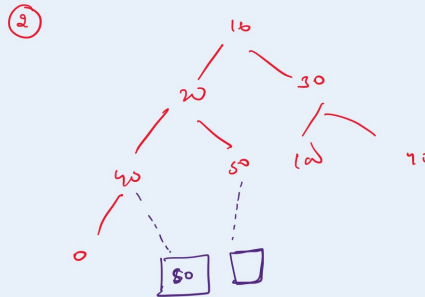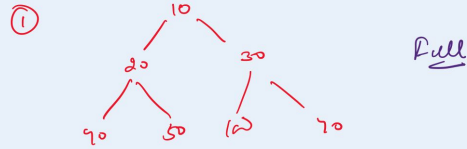
Binary Tree

Full Binary Tree

→ Every node in a FBT, 0 children or 2 children.

① 10
  20   30
 40 50 60 70

② 10
  20   30
 40   50
 60 70

Binary Tree

Binary Tree

Perfect **Binary Tree**

→ all internal nodes have two children
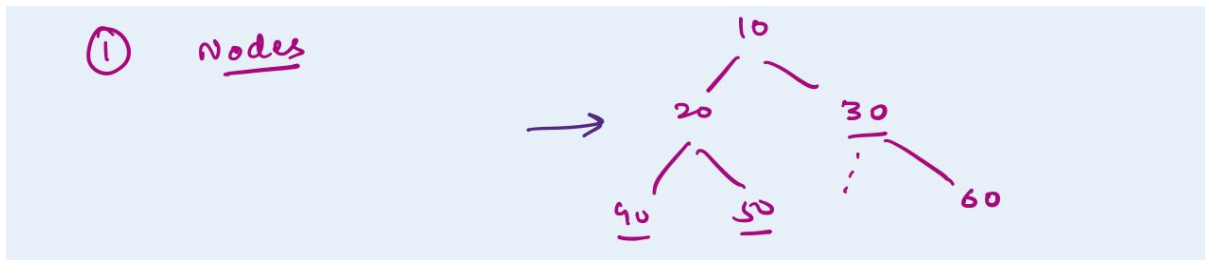and all leaves are at same level
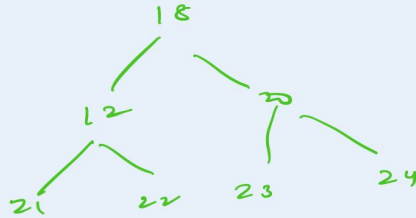
Implementation of Binary Tree

Now that we have a fair idea about what Binary Tree is, let's see how we can implement it programmatically.

1.   Using Nodes

We can implement trees using nodes which store the information about the current node as well as the information about the child nodes.
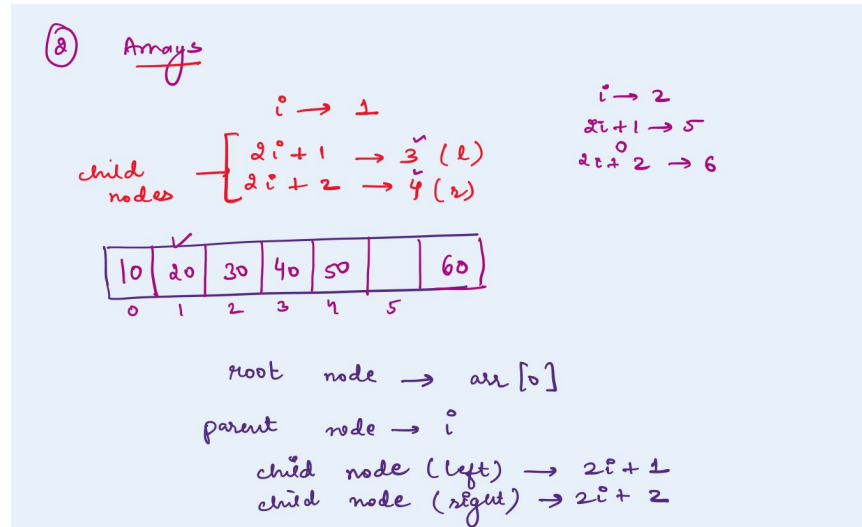
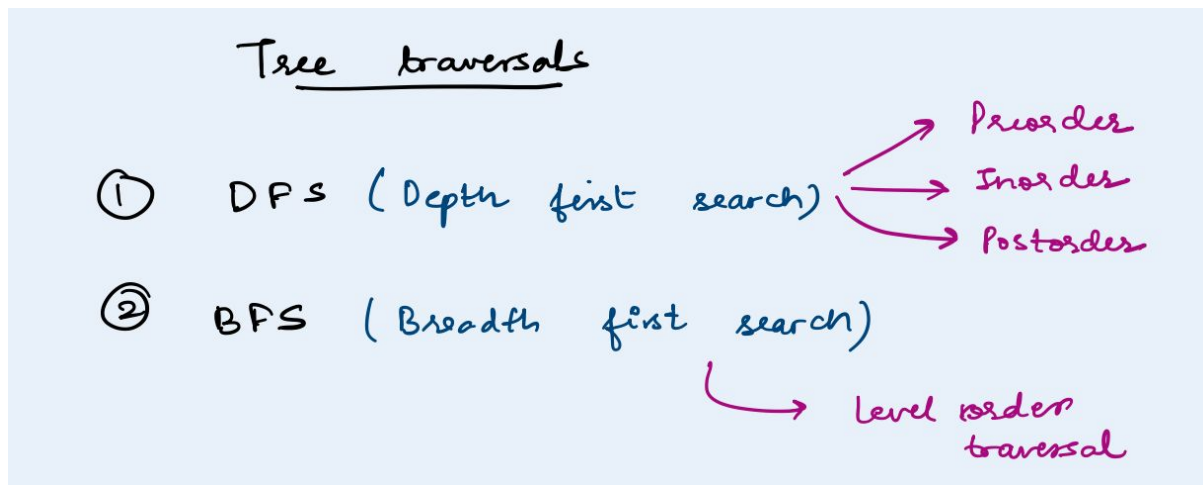# Implementation of Binary Tree

2. Using Arrays

We can also store the information about Binary Tree using Arrays.

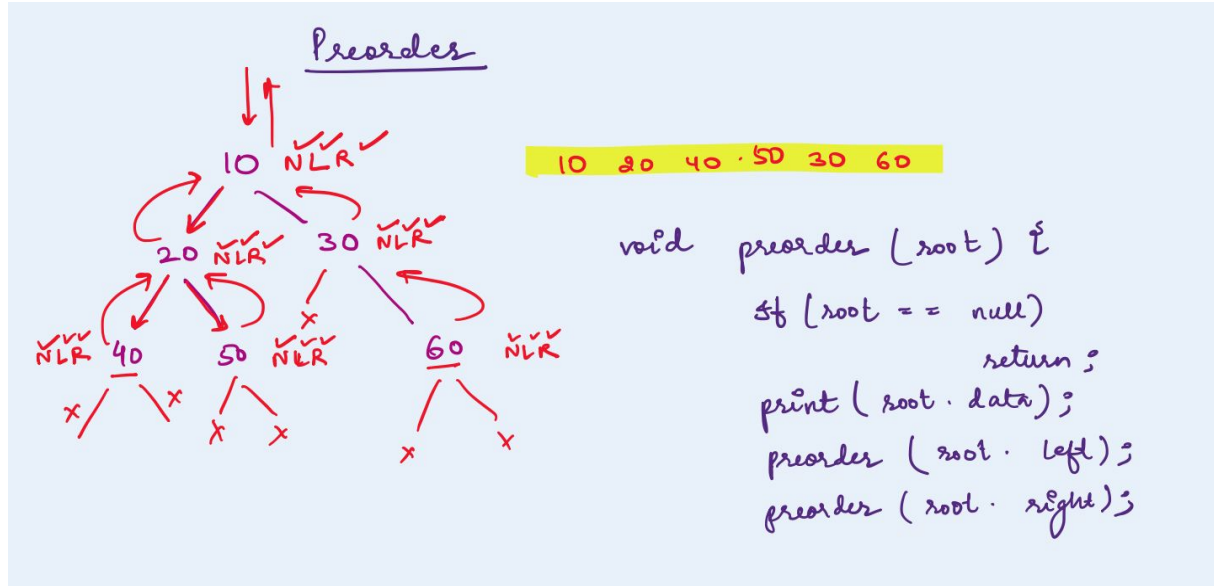If a node is stored at an index i, it's child is stored at index 2*i +1 and 2*i + 2.

Tree Traversals

Now that we have covered what a Binary Tree looks like, we'll talk about how we can traverse a tree. Traversing a tree means visiting every node in the tree. There are two ways of tree traversal.

Tree traversals

① DFS (Depth first search) → Preorder
→ Inorder
→ Postorder

② BFS (Breadth first search)
→ Level order traversal

Tree Traversals

Preorder

10   20   40 · 50   30   60

```
void preorder (root) {
    if (root == null)
                    return;
    print (root. data);
    preorder (root. left);
    preorder (root. right);
}
```

Tree Traversals

Inorder (LNR)

50 20 10 40 30 80 70

```
void inorder (root) {

    If (root == null)
              return;

    inorder (root.left);
    print (root.val);
    inorder (root.right);
```

Tree Traversals

Postorder   (↓↓↓ LRN)

LRN

10

20   70

30   80   70

40

50   60

50 60 40 30 20 80

90 70 10
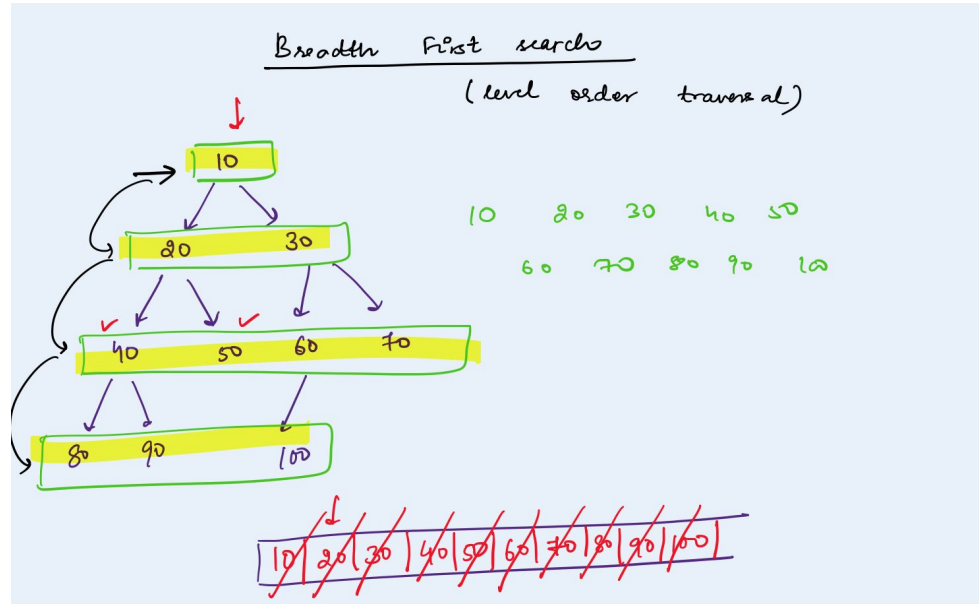
Breadth First Traversal

Level Order traversal is also known as Breadth-First Traversal since it traverses all the nodes at each level before going to the next level (depth).

Thank you!