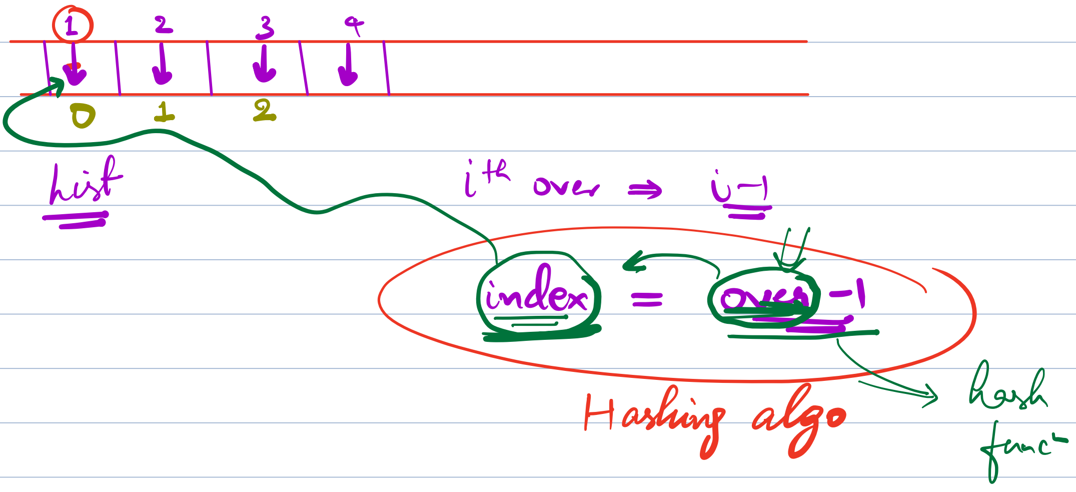
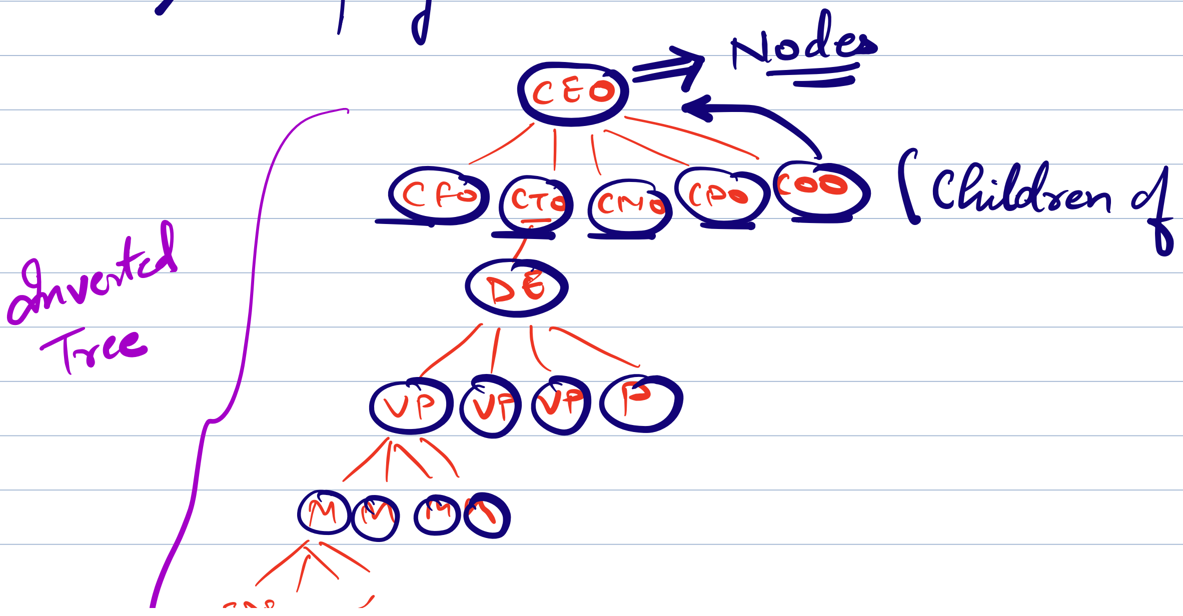
Arrays
Linked List
Stacks
Queues. } Linear Data

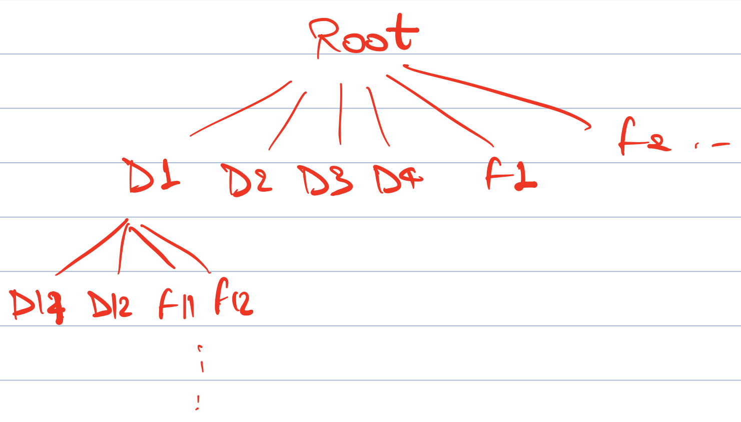


Q What if there is a hierarchy in the data ??

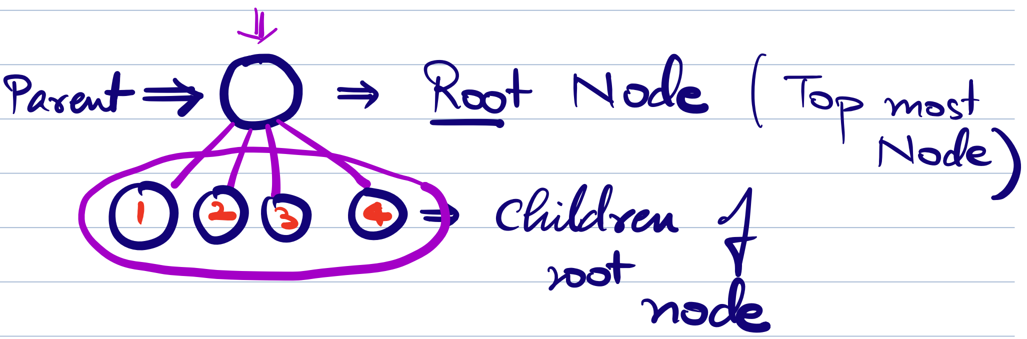
1) Company structure



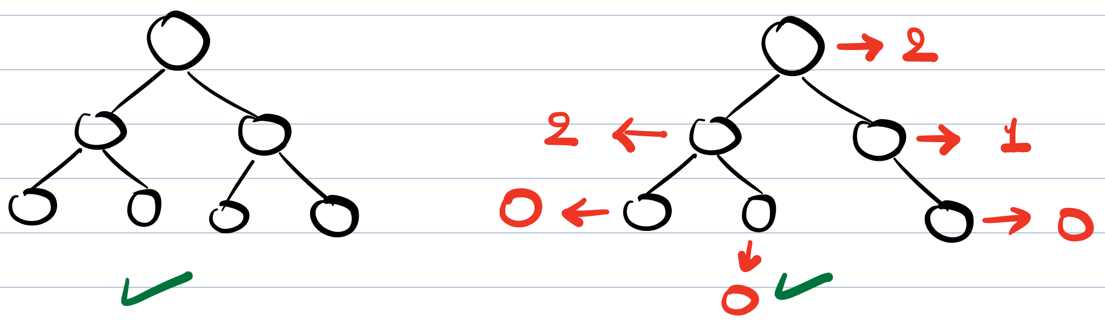
2) File System

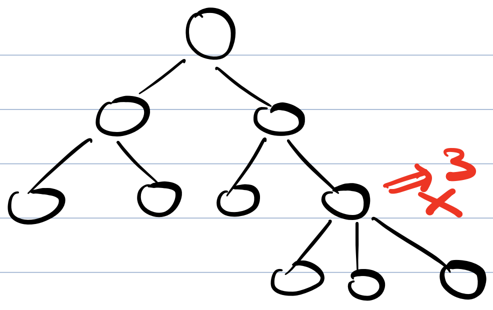


⇒ So, to represent hierarchy, we use Tree Data Structure.



A node can Max of ② children ⇒ Binary Tree



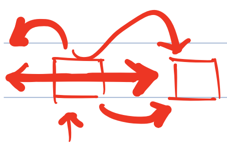


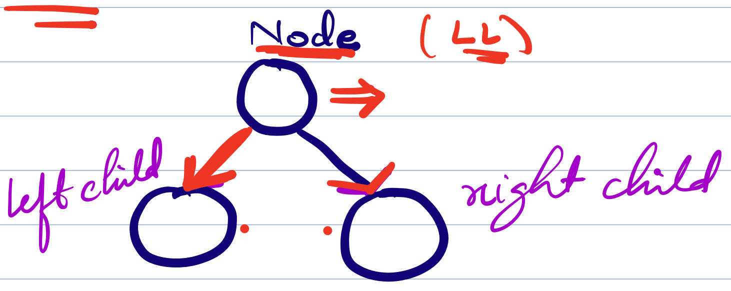
max children
3 ⇒ 3ary tree or ternary tree

4 ⇒ 4 ary
⋮

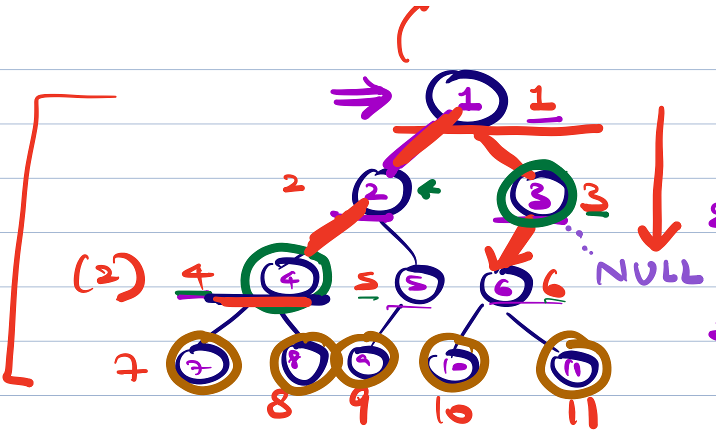
N ⇒ N ary tree

Binary Tree





```
Node {
    int data;
    Node leftChild;       Structure of
    Node RightChild;      a treeNode
}
```

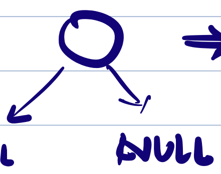


1 { data = 1;
LC = 2
RC = 3 }

2 { data = 2
LC = 4, RC = 5 }

3 { LC = 6, RC = NULL }

nodes = 11



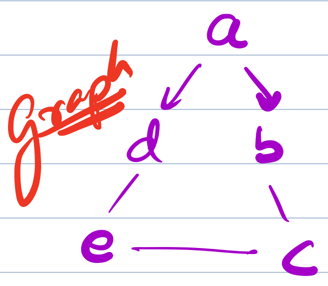
→ leaf nodes

1) Depth ⇒ Distance from the root node

Depth (root) = 0

Depth (2, 3) = 1

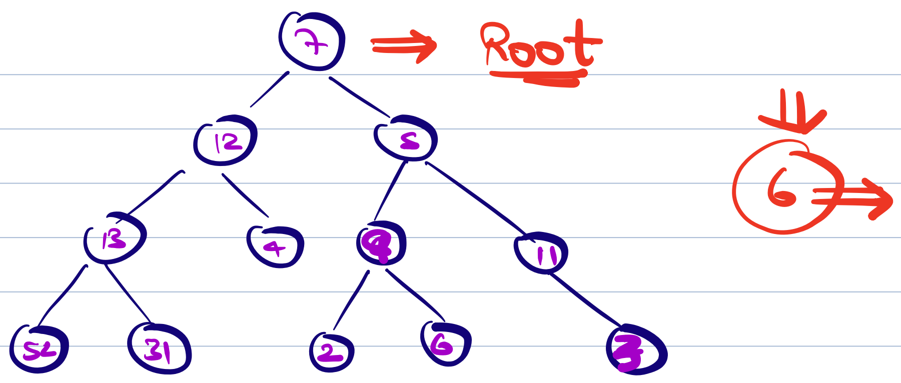
Depth (4, 5, 6) = 2



(Advanced DSA)

No longer a tree

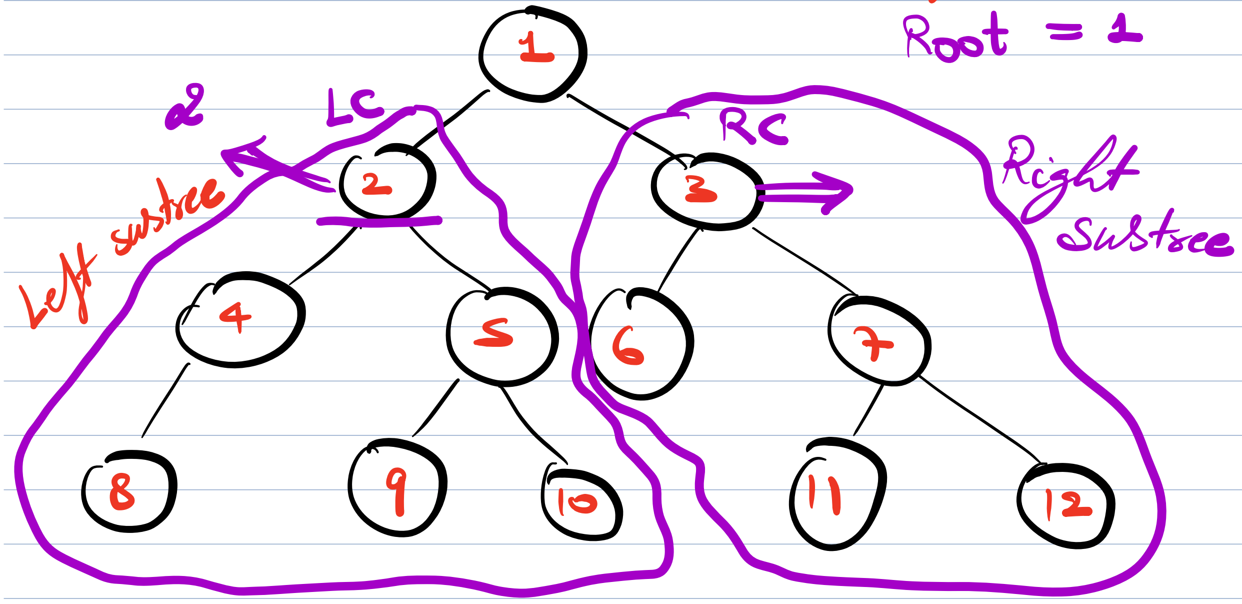
Q
=



⇒ To search for a node, we will have to go to each node of the tree.

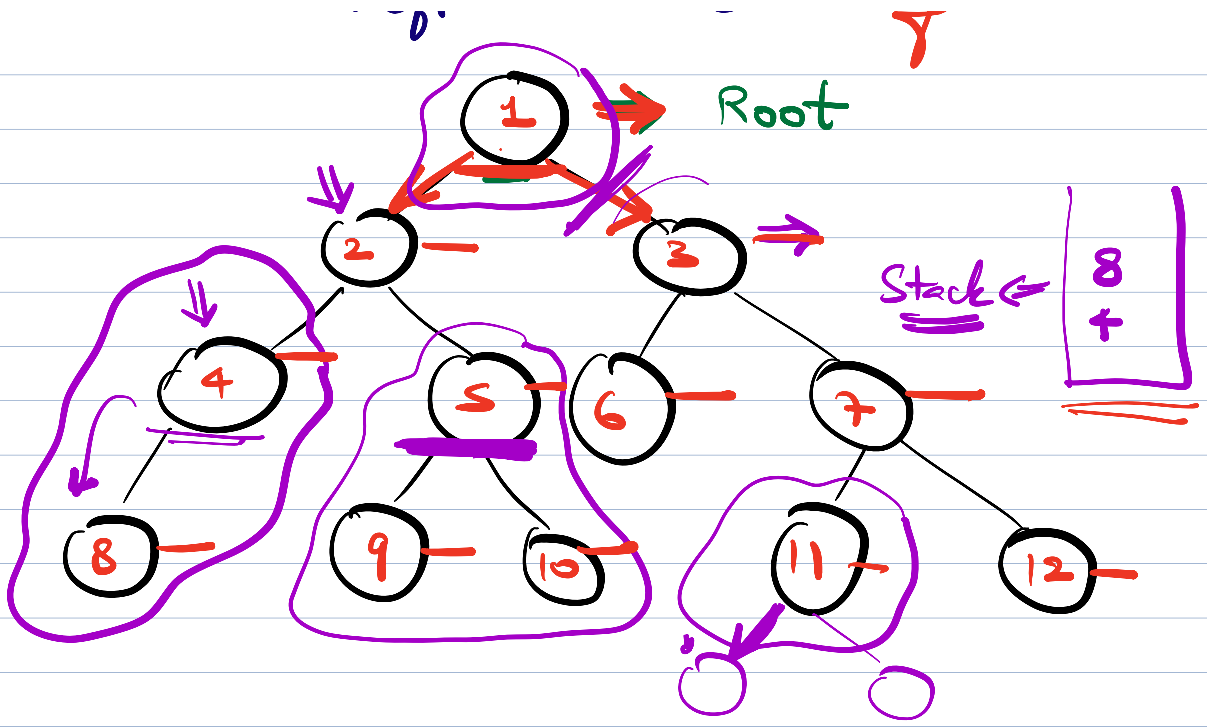
⇓

Traversal of tree

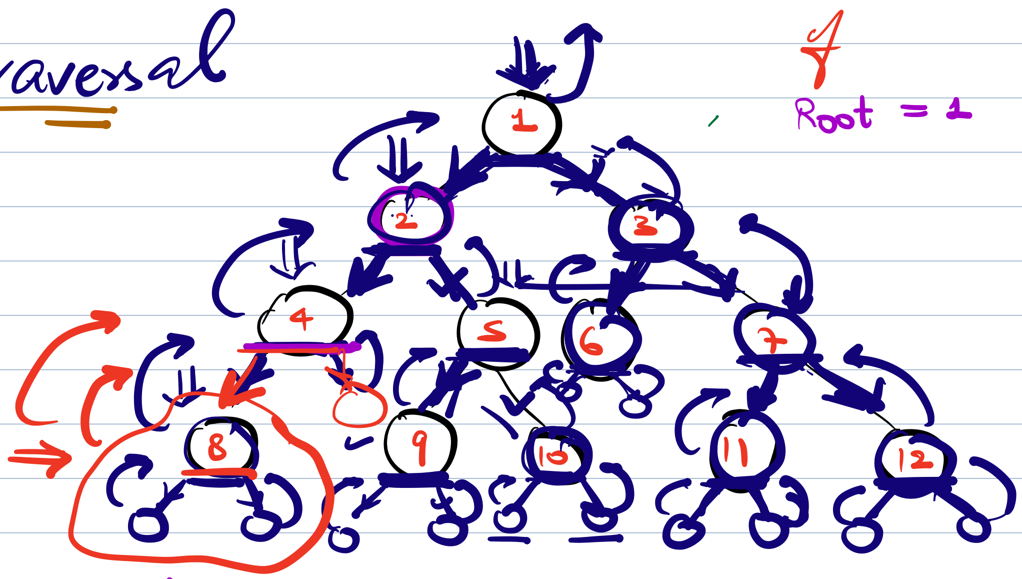


⇒ The given tree is rooted at Node 1

⇒ The tree rooted at the LC of a given node is known as the Left Subtree of



Traversal



Pre Order	In Order	Post Order
Root, LST, RST.	LST, Root, RST	LST, RST, Root
1, 2, 4, 8, 5, 9, 10, 3, 6, 7, 11, 12	8, 4, 2, 9, 5, 10, 1, 6, 3, 11, 7, 12,	H.W: Dry Run

void preorder (root) {

```
  if (root == NULL) {
        return;
  }
1) print (root. data);
LST preorder (root. left
                    child);
RST preorder (root. right
                    child);
  return;
}
```

```
inorder (root) {
  if (root == NULL) {
        return;
  }
inorder (root. leftchild);
print (root. data);
inorder (root. RC);
return;
}
```

⇓

Root (Pre)   LST   Root (In)   RST   Root (Post)

H.W.

Q Can we do tree traversal w/o recursion ??

Use stack & traverse tree iteratively w/o recursion