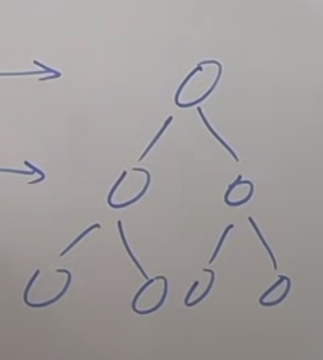
TREES Data Structures

## A Binary Tree is a hierarchy tree which atmax contains 2 nodes at their beneath children

1. Root node
2. Children
3. Last node of the tree is called Leaf NODE
4. Predecessors of node are called as ancestors

Types of Binary Tree:

1. Full Binary Tree -> Either hava 0 or 2 childrens
2. Complete Binary Tree -> all levels are completely filled except the last level !  The last level node as left as possible
3. Perfect BT -> all leaf node are at the same level
4. Balanced BT -> height of tree at max of log(n)
5. Degenrate Tree -> Skew tree of a staright line (Every node has a single children)

Structure of BT in JAVA

Class Node {  
Node left;

Node right;

Int data;

Public Node(int data){

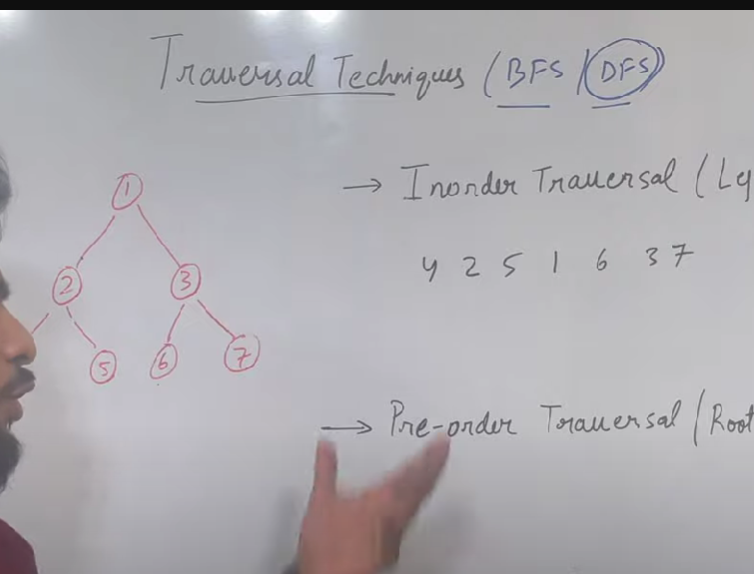
This.data = data;

Left = null;

Right = null;

}

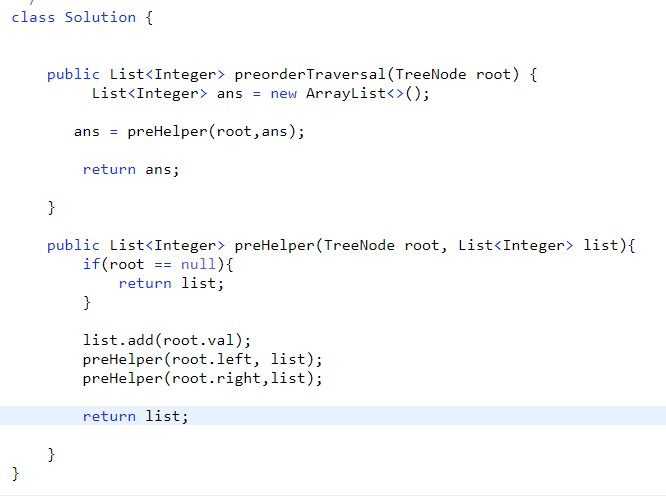
Traversal Of Trees

1. In order -> (left root right)
2. Pre order -> (root, left,right)
3. Post order -> (left, right , root)
4. 

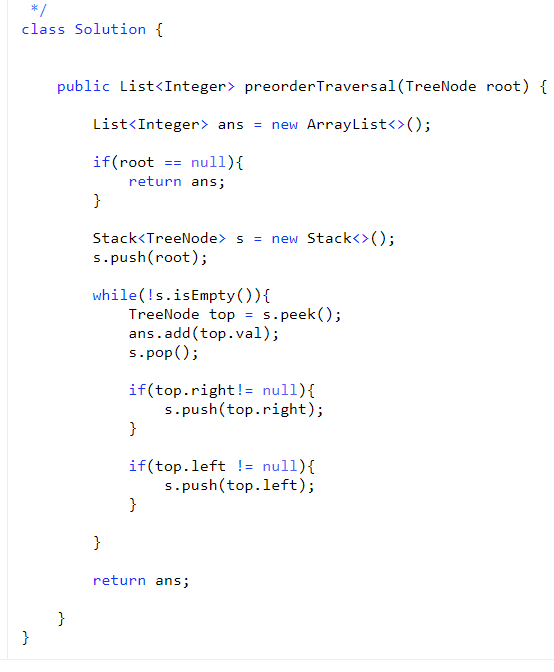
inorder -> 4251637 pre -> 1245367 post -> 4526731

-

1. PreOrder Traversal Code

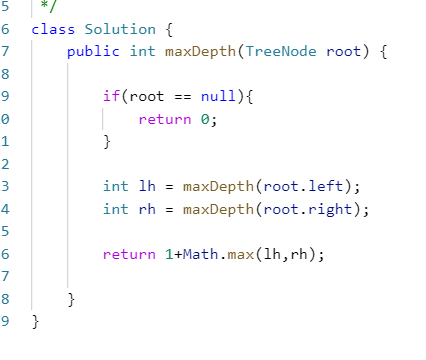


Iterative Solution of Pre-Order traversal:-

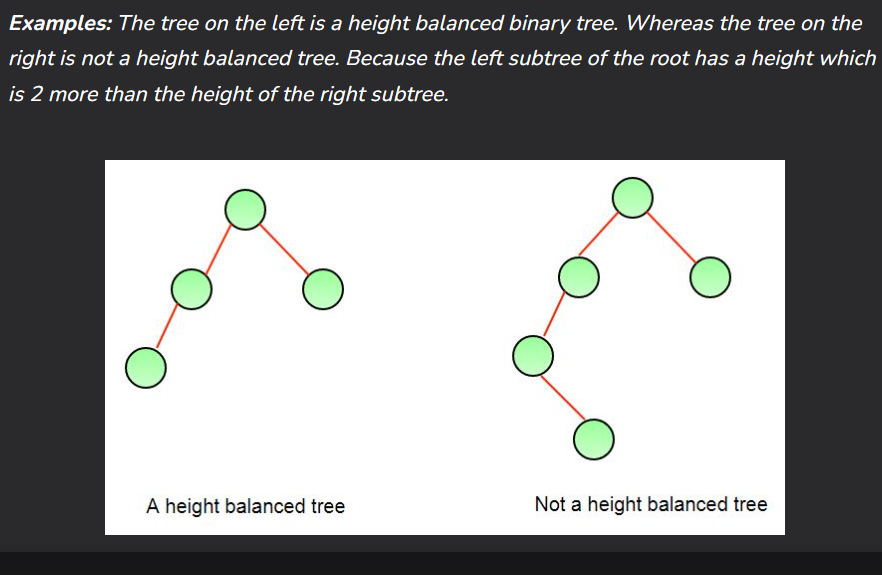


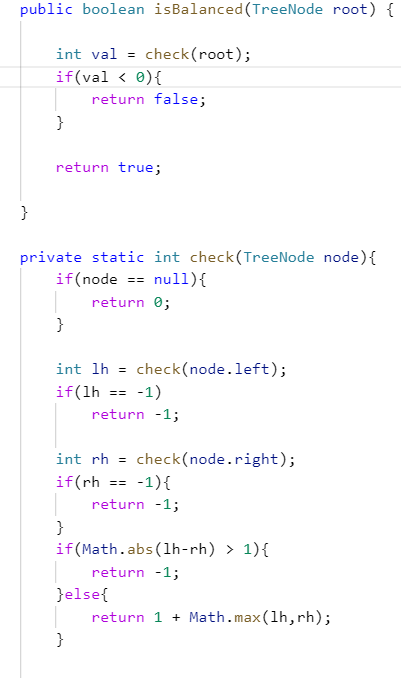
Question: Maximum Depth of Binary Trees

It’s nothing but we need to find the (root +max\_height\_of(Left,right));



Question : Find the height of the balanced Binary Tree





The approach is like we will check for the height of the BT if at any point of time we will get -1 we will return it and tell them it’s not valid Tree!!!

We will check for left height and right one and count the max height and check if absolute sum is Greater THAN > 1 then definitely it’s not balanced return -1 ;

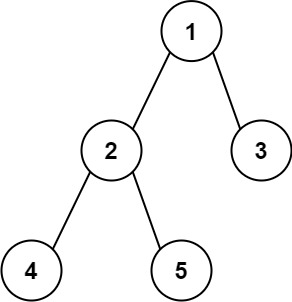
Questions : Diameter of Binary TREE

Given the root of a binary tree, return *the length of the****diameter****of the tree*.

The **diameter** of a binary tree is the **length** of the longest path between any two nodes in a tree. This path may or may not pass through the root.

The **length** of a path between two nodes is represented by the number of edges between them.

**Example 1:**



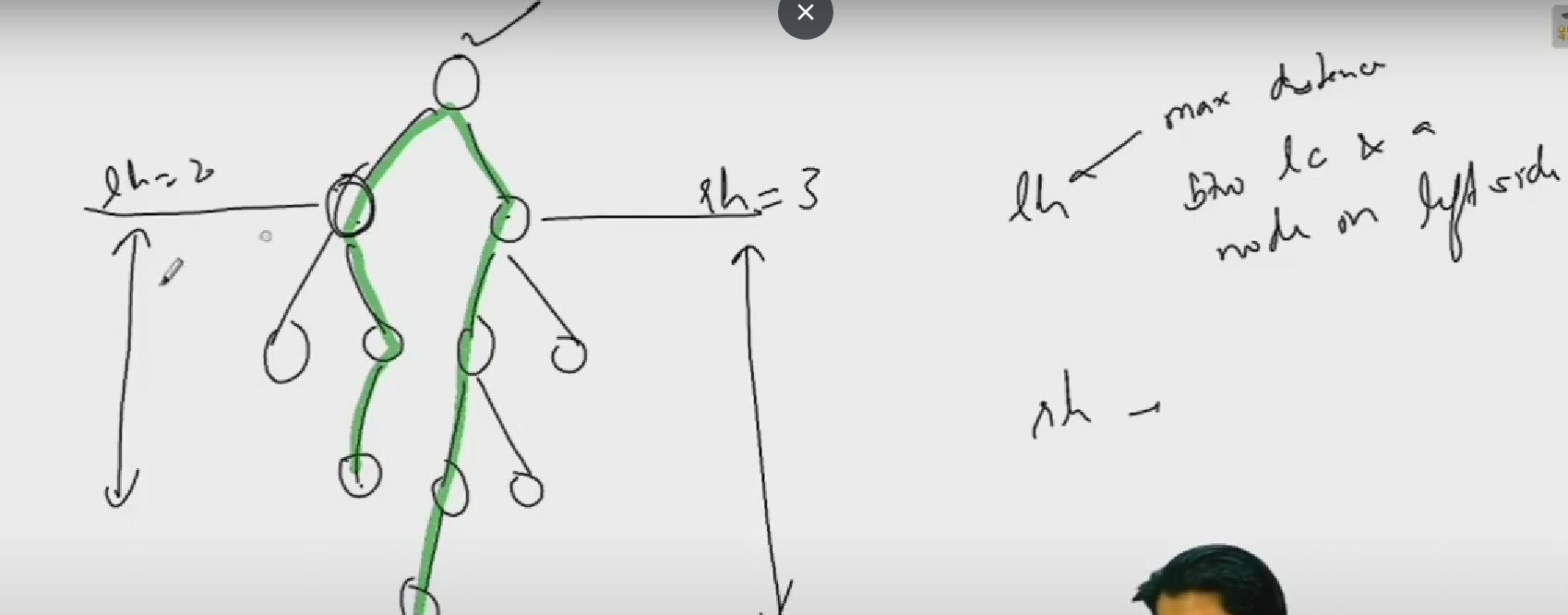
**Input:** root = [1,2,3,4,5]

**Output:** 3

**Explanation:** 3 is the length of the path [4,2,1,3] or [5,2,1,3].

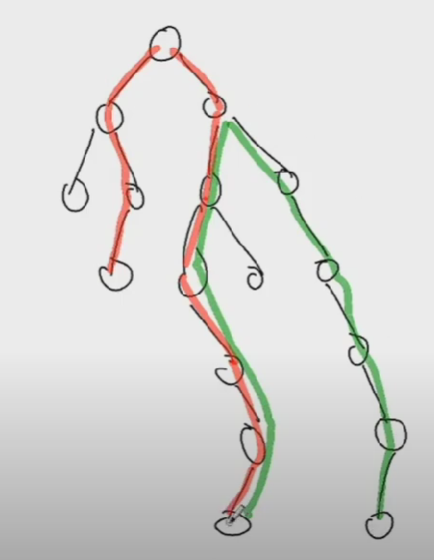
NOTE: *The****diameter/width of a tree****is defined as the number of nodes on the longest path between two end nodes.*

Find the height of the right path + left path

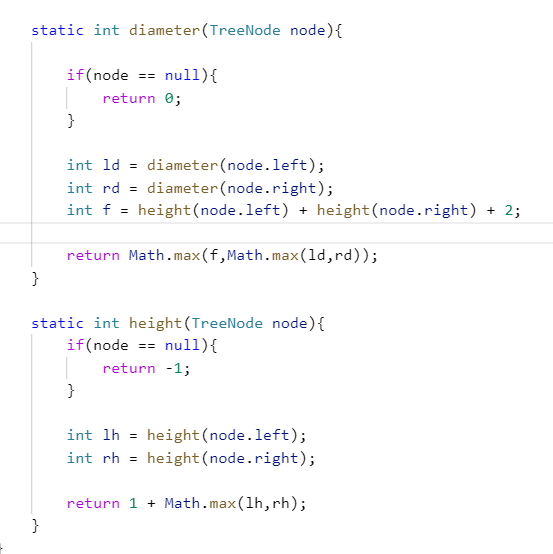


Left height represent the maximum dist btw left child to a node on left Side !

It’s not necessary to represent diameter on the basis of the height from root to node



As in the give example it’s not compulsory to find the diameter from the root element as we can say that in green color line the diameter is of length 10 so that’s how for every height along with diameter of every node we need to find



BruteForce solution is above with TC – O(n)

#OPTIMAL SOLUTIONS

**Solution 2: Post Order Traversal**

**Intuition :**

Is it possible to optimize the above solution further? Which operation do you think is very repetitive in nature in the above solution?

 Height of the subtrees.

Can we use postorder traversal to calculate everything in a single traversal of the tree?

Yes, as in post-order traversal, we have to completely traverse the left and right subtree before visiting the root node.

So, the idea is to use post-order traversal and keep calculating the height of the left and right subtrees. Once we have the heights at the current node, we can easily calculate both the diameter and height of the current node.

**Approach :**

* Start traversing the tree recursively and do work in Post Order.
* In the Post Order of every node , calculate diameter and height of the current node.
* If current diameter is maximum then update the variable used to store the maximum diameter.
* Return height of current node to the previous recursive call.