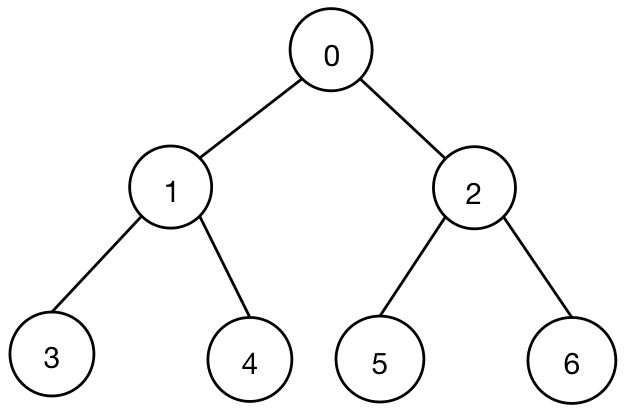
Balanced Binary Tree

*Description:*

Given a Binary Tree, determine if it is height - balanced.

For this problem, a height - balanced binary tree is defined as a binary tree in which the depth of two sub - trees of each node would never differ by more than 1.



*Analysis:*

1. *Leaf Node* - 3, 4, 5, and 6 have no sub - trees, therefore these nodes are height - balanced binary tree nodes.
   1. *The Height of Node 3, 4, 5, and 6 equal to 0.*
2. *Internal Node* - 1, and 2 have sub - trees. The height difference of leaf node 1 equals to 0, therefore node 1 is the height - balanced binary tree node. The height difference of leaf node 2 equals to 0, therefore node 2 is the height - balanced binary tree node.
   1. *The Height of Node 1 equals to max ( Node 3 + 1, Node 4 + 1 ) = max ( 1, 1 ) = 1.*
   2. *The Height of Node 2 equals to max ( Node 5 + 1, Node 6 + 1 ) = 1.*
3. *Root Node* - 0 has 1, and 2 sub - trees. Since internal node 1 and 2 are all height - balanced binary sub - tree. Therefore root node 0 is the height - balanced binary tree.
   1. *The Height of Node 0 equals to max ( Node 1 + 1, Node 2 + 1 ) = max ( 2, 2 ) = 2.*

By thinking recursively, we need to consider each node by using *Pre - Order Display Sequence* to check whether each node satisfies the condition that the depth of two sub - trees of each node would never differ by more than 1.

*Original Prototype - Pre - Order Display:*

void preOrderDisplay ( BinaryTreeNode \* root )

{

if ( root == NULL )

return;

print ( root-> value );

preOrderDisplay ( root -> left );

preOrderDisplay ( root -> right );

return;

}

*Extension - CalculateBalancedBinaryTree:*

int calculateBalancedBinaryTree ( BinaryTreeNode \* root )

{

if ( root == NULL )

return 0;

int left\_height = calculateBalancedBinaryTree ( root -> left );

int right\_height = calculateBalancedBinaryTree ( root -> right );

//

1. Check whether the absolute height difference of left and right sub-trees is more than 1.

1. Check whether the Left - Child Tree is BalancedBinaryTree.
2. Check whether the Right - Child Tree is BalancedBinaryTree.

if ( left\_height == -1 || right\_height == -1 || abs ( left\_height - right\_height ) > 1 ) return -1;

return max ( left\_height, right\_height ) + 1;

}