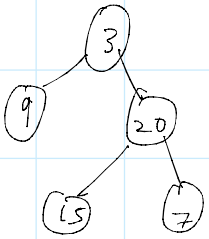
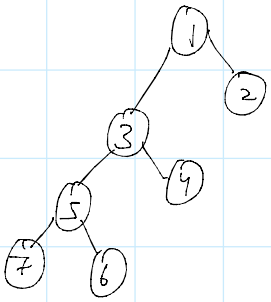
110. Balanced Binary Tree

18 February 2022 07:47 PM

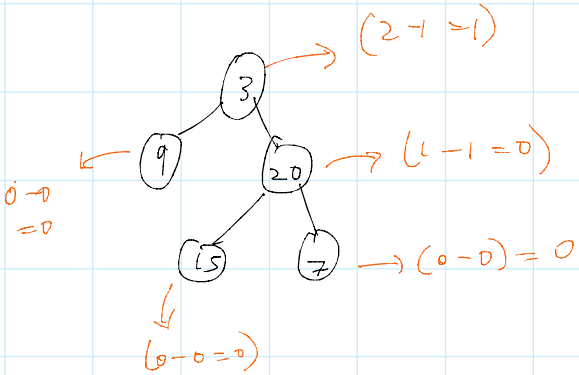


(1)

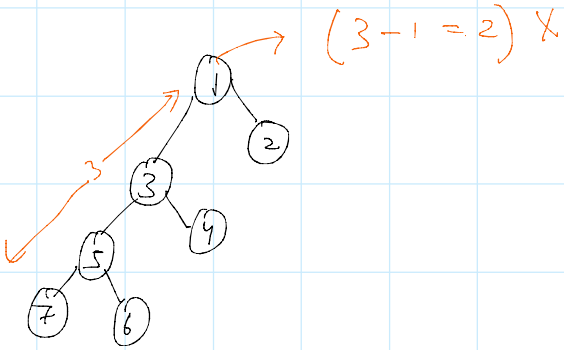


(2)

Balanced B.T → for every node, height of (left) − height of (right) <= 1



It's balanced B.T



Brute force

```
bool check(node)
if node == null
return true
```

```
lh = findHleft(node.left)
rh = findHright(node.right)
```
] O(N)

```
if (abs (rh - lh) > 1 return false;

Bool left = check (node.left)
Bool right = check (node.right)
if ( ! left || ! right) return false
return true.
```

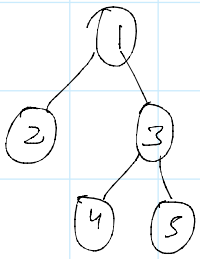
T.C ⇒ O(N) × O(N) = O(N²)

↳ for finding height

↳ for traversing

Better Approach

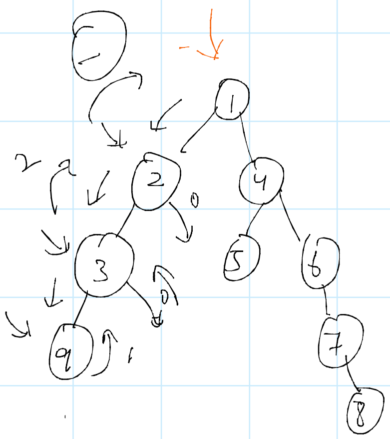
find height of tree




```
int height (node)
{
    if (node == null)
        return 0
    lh = height (node.left)
    rh = height (node.right)
if (lh == -1 || rh == -1) return -1;
if (abs (lh - rh > 1) return -1; // not a Balanced B-T
return max (lh, rh) + 1
```

}.

Dry Run



```
int height (node)
{
  if (node == null)
     return 0

  lh = height (node.left)
  rh = height (node.right)
  if (lh == -1 || rh == -1) return -1;
  if (abs (lh - rh > 1) return -1; // not a Balanced B.T

  return max (lh, rh) + 1
}
```

At (2)

lh - 2
rh -

a (lh - >1) turn -1 ✓

l omen't node (1) saw -1 it became false so no need to

isit other tree.

```java
class Solution {
    public boolean isBalanced(TreeNode root) {
        // if it does not return -1 then its balanced tree
        return dfsHeight (root) != -1;
    }

    int dfsHeight(TreeNode root){
        if(root == null) return 0;

        int leftHeight = dfsHeight(root.left);
        if(leftHeight == -1) return -1;

        int rightHeight = dfsHeight(root.right);
        if(rightHeight == -1) return -1;

        if(Math.abs(leftHeight - rightHeight) > 1) return -1;
        return Math.max(leftHeight, rightHeight) + 1;    // height
    }

}
```

T.C ⇒ O(N)   S.C ⇒ O(N)