Am I audible ?

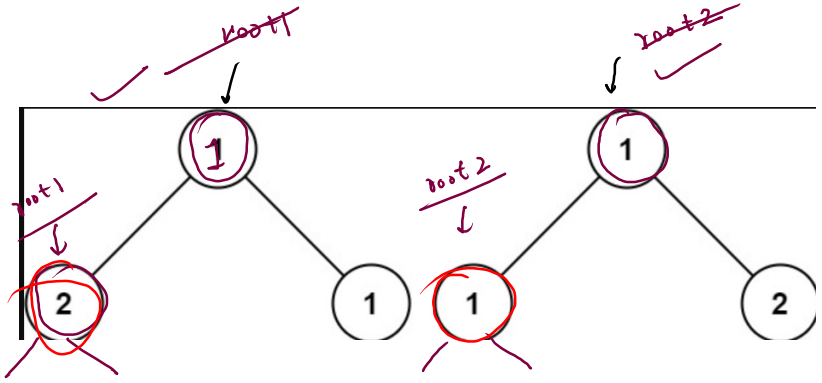
# . Same Tree (Day 43)

Problem | Submissions | Leaderboard | Discussions

Given preorder of two binary trees first you need to construct binary tree you have roots of two binary trees, write a function to check if they are the same or not.

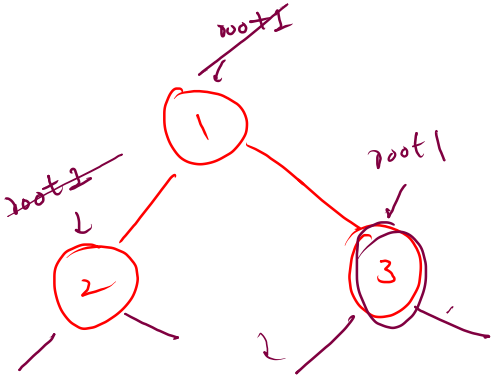
Two binary trees are considered the same if they are structurally identical, and the nodes have the same value.Same Binary tree tree are not same

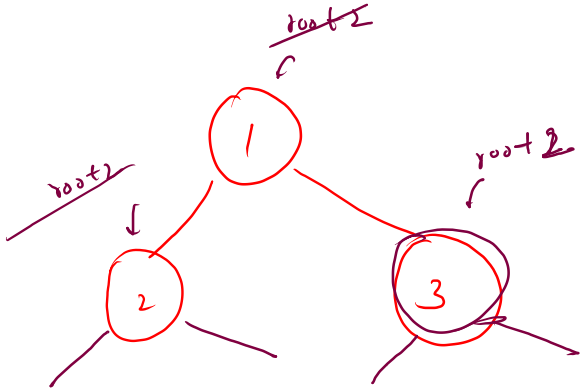
⇒ same
↳ structurally
↳ identical.
↳ data should be same.




⇒ false

⇒ Solve




⇒ true

⇒ root1 → null

root2 → null ⇒ true

⇒ root1 → null

root2 → 1 ⇒ false


⇒ Can I check data of both trees node.

⇒ Can I call to its left child.

⇒ Can I call to its right child.

↳ return true.

```java
public static boolean isSameTree(Node root1, Node root2){
    if(root1 == null && root2 == null){
        return true;
    }

    if(root1 == null || root2 == null){
        return false;
    }

    boolean cond1 = root1.data == root2.data;
    boolean cond2 = isSameTree(root1.left, root2.left);
    boolean cond3 = isSameTree(root1.right,root2.right);

    return cond1 && cond2 && cond3;

/       if(cond1 == true && cond2 == true && cond3 == true){
/           return true;
/       }

/       return false;
}
```

Code
+
do dry run
5 - 6 mins

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.

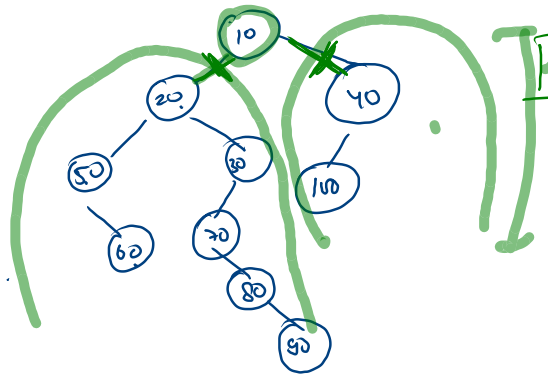
The length of a path between two nodes is represented by the number of edges between them.

Sample Input 0

```
21
10 20 50 n 60 n n 30 70 n 80 n 90 n n 40 100 n n n n
```

Sample Output 0

```
6
```


$h_2$

⇒ find diameter in the left child. → (x)

⇒ ——— × ——— right child. → (y)

⇒ myDiameter = $\underbrace{h_1}_{\text{left}} + \underbrace{h_2}_{\text{right}} + 2$ : → (z)

return max(max(x, y), z))

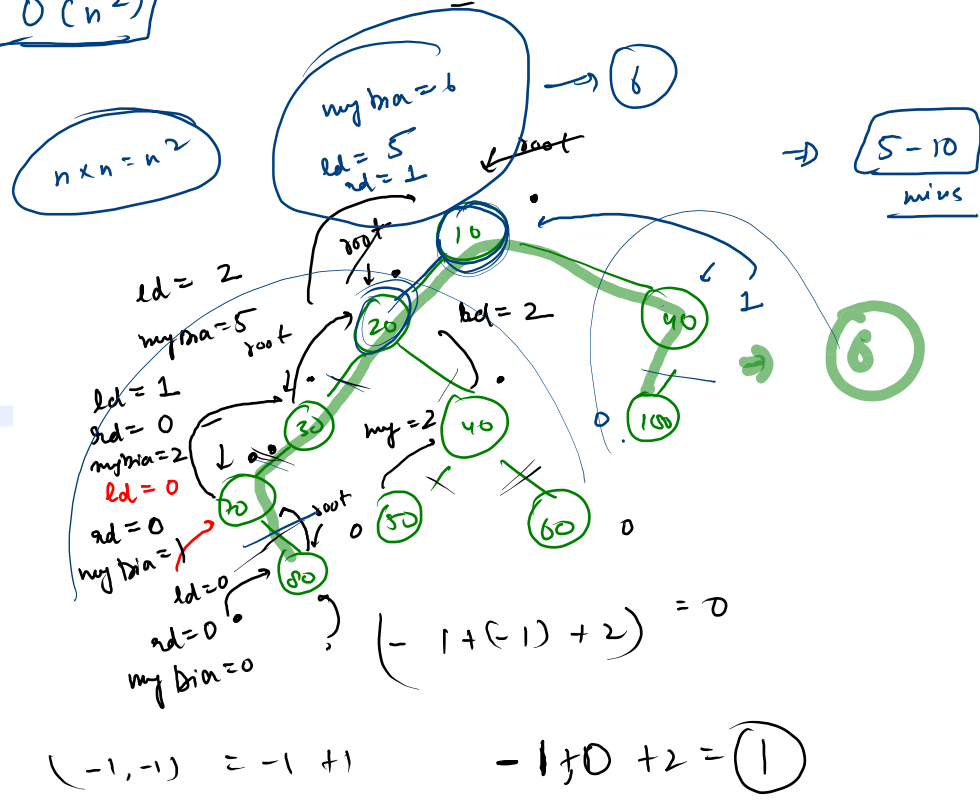
max ($h_1$, $h_2$) + 2

```java
public static int height(Node root){
    if(root == null) return -1;

    int lh = height(root.left);
    int rh = height(root.right);

    return Math.max(lh,rh)+1;
}

public static int diameter(Node root){
    if(root == null){
        return 0;
    }

    int ld = diameter(root.left);
    int rd = diameter(root.right);

    int h1 = height(root.left);
    int h2 = height(root.right);

    int myDiameter = h1+h2+2;


    return Math.max(Math.max(ld,rd),myDiameter);
}
```

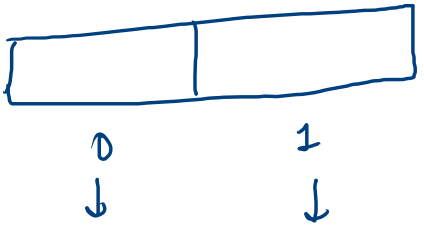
T.C. → $O(n^2)$

$n \times n = n^2$

h → n → n → n



arr 1D




diameter  height

↳ return arr[0];

Jump

A-01

```
//     public static int height(Node root){
//         if(root == null) return -1;
//
//         int lh = height(root.left);
//         int rh = height(root.right);
//
//         return Math.max(lh,rh)+1;
//     }
//
//     public static int diameter01(Node root){
//         if(root == null){
//             return 0;
//         }
//
//         int ld = diameter(root.left);
//         int rd = diameter(root.right);
//
//         int h1 = height(root.left);
//         int h2 = height(root.right);
//
//         int myDiameter = h1+h2+2;
//
//
//         return Math.max(Math.max(ld,rd),myDiameter);
//     }
```

T.C → O(n²)

S.C → O(1)

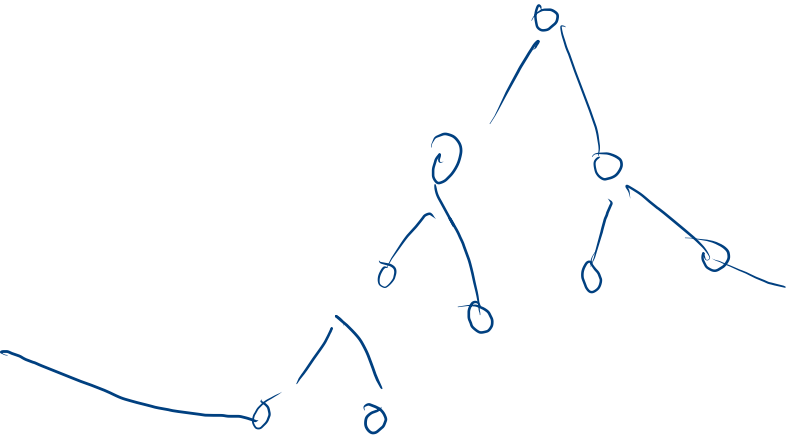
A-02

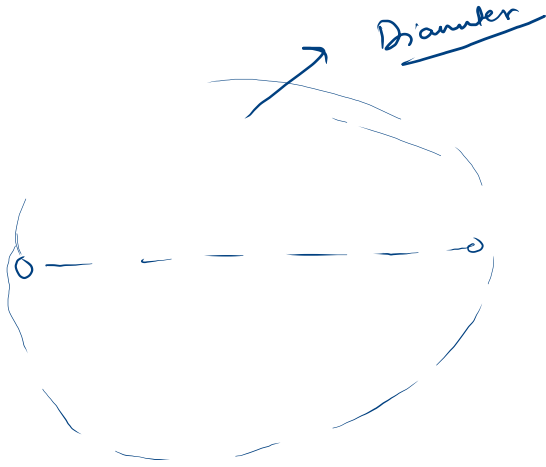
8 int

```
public static int [] helper(Node root){
    if(root == null){
        return new int []{0,-1};
    }

    int [] ld = helper(root.left);
    int [] rd = helper(root.right);

    int [] myAns = new int[2];

    myAns[0] = Math.max(ld[0],Math.max(rd[0],(ld[1]+rd[1]+2)));
    myAns[1] = Math.max(ld[1],rd[1])+1;

    return myAns;

}

public static int diamter02(Node root){
    int arr[] = helper(root);
    // 0 - diameter, 1-> height
    return arr[0];
}
```

T.C → O(n)

S.C → O(1)




D'amuler