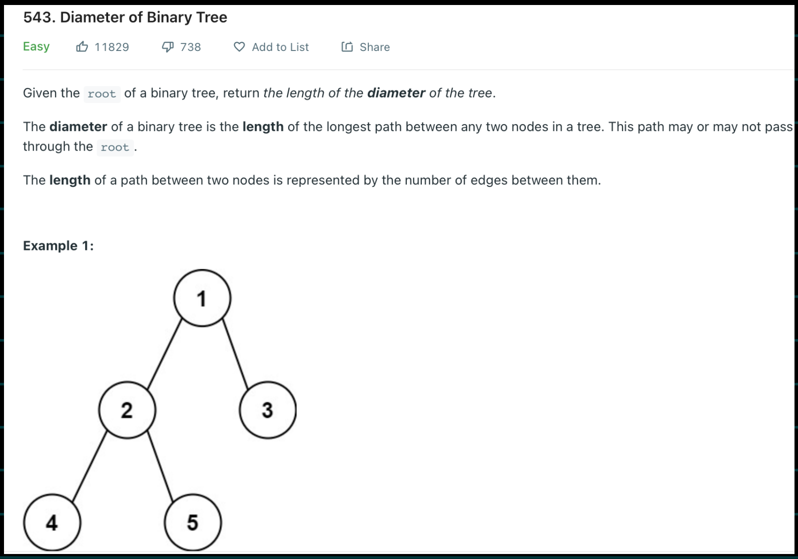
Diameter of Binary Tree



543. Diameter of Binary Tree

Easy 11829 738 Add to List Share

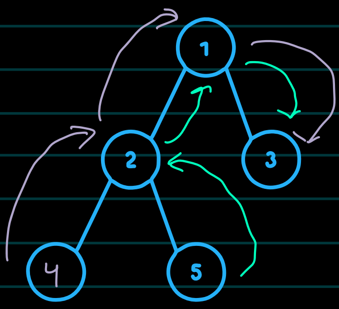
Given the `root` of a binary tree, return *the length of the* ***diameter*** *of the tree*.

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:

Diameter of Binary is the maximum distance between 2 nodes



Distance = 4

" = 4

Brute force :-

height will give max height

Diameter can be found

$$\begin{tikzcd}
 & \text{Diameter can be found} \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{Left subtree} \arrow[d] & \text{Right subtree} \arrow[d] & \text{combination of LST / RST} \arrow[d] \\
\text{op1= Diameter(root} \rightarrow \text{left)} & \text{op2= Diameter(root} \rightarrow \text{right)} & \text{op3 = height(root} \rightarrow \text{left)+1+height(root} \rightarrow \text{Right)}
\end{tikzcd}$$

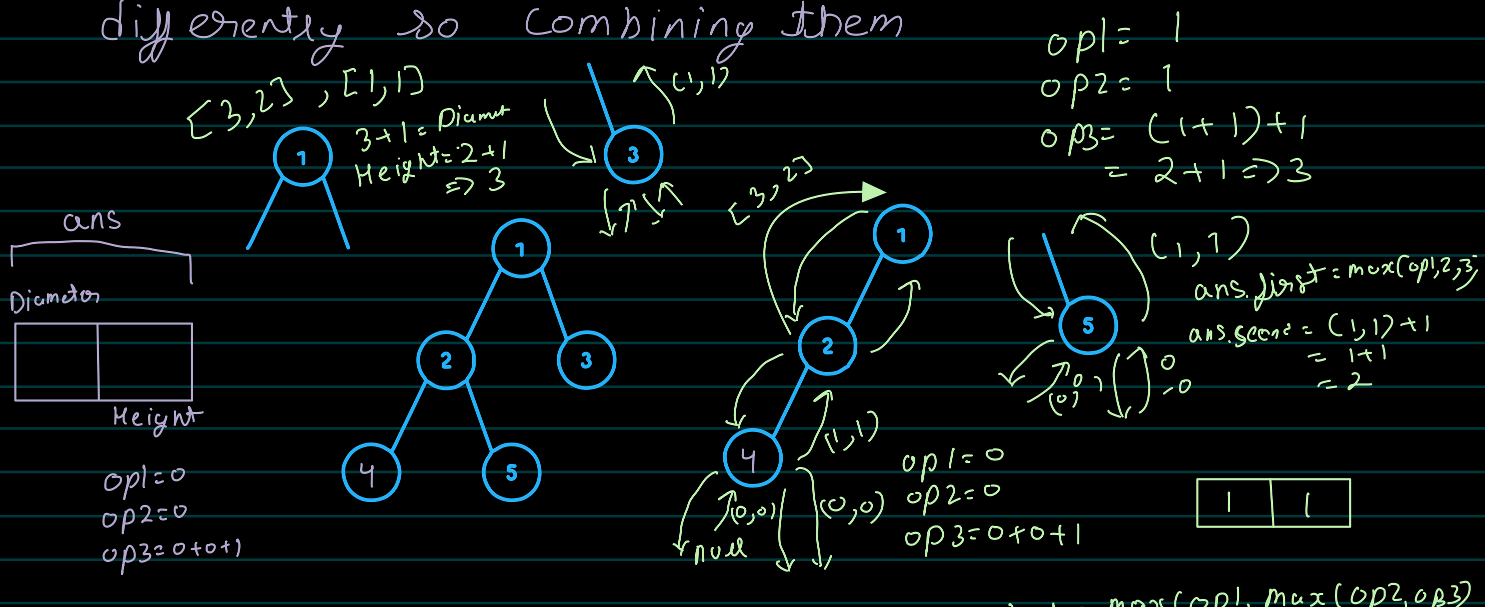
ans = max(op1, max(op2, op3)

T.C − O(n x n)

$\swarrow$ Diameter $\downarrow$ Height

Optimal :

we can return pair instead of returning Single int Since we are calculating max height and Diameter differently so combining them

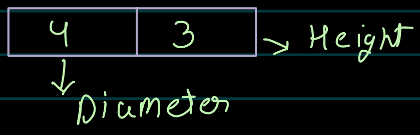


ans.first = max(op1, max(op2,op3)
= (0, (0,1)
= 1
ans.second = (0,0)+1
= 0+1
= 1

ans.first = max(op1, max(op2, op3))

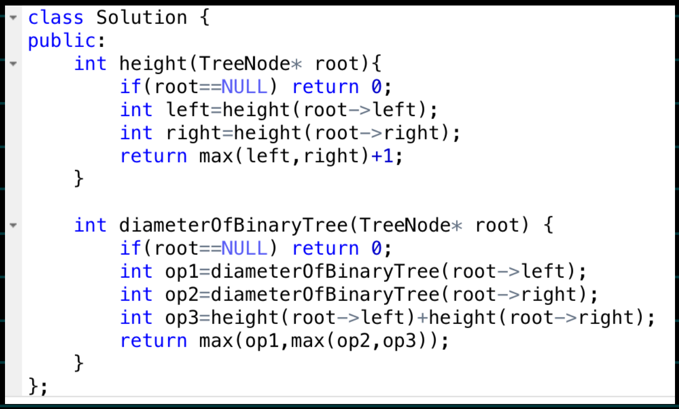
ans.Second = max(left.second , Right.second)+1 ;

return ans



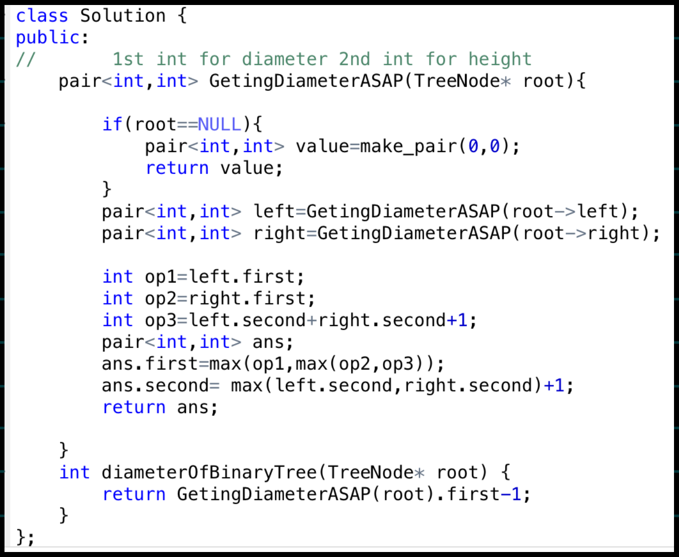
Optimal

Bruteforce



```
class Solution {
public:
    int height(TreeNode* root){
        if(root==NULL) return 0;
        int left=height(root->left);
        int right=height(root->right);
        return max(left,right)+1;
    }

    int diameterOfBinaryTree(TreeNode* root) {
        if(root==NULL) return 0;
        int op1=diameterOfBinaryTree(root->left);
        int op2=diameterOfBinaryTree(root->right);
        int op3=height(root->left)+height(root->right);
        return max(op1,max(op2,op3));
    }
};
```



```
class Solution {
public:
//       1st int for diameter 2nd int for height
    pair<int,int> GetingDiameterASAP(TreeNode* root){

        if(root==NULL){
            pair<int,int> value=make_pair(0,0);
            return value;
        }
        pair<int,int> left=GetingDiameterASAP(root->left);
        pair<int,int> right=GetingDiameterASAP(root->right);

        int op1=left.first;
        int op2=right.first;
        int op3=left.second+right.second+1;
        pair<int,int> ans;
        ans.first=max(op1,max(op2,op3));
        ans.second= max(left.second,right.second)+1;
        return ans;

    }
    int diameterOfBinaryTree(TreeNode* root) {
        return GetingDiameterASAP(root).first-1;
    }
};
```