Trees Demux

18 October 2021   22:03

Topic	Question	Difficulty	Link	STATUS	Similar pblms
Trees	96. Unique BST	med	https://leetcode.com/problems/unique-binary-search-trees/		
	94. Inorder traversal binary tree	easy			https://leetcode.com/problems/binary-tree-level-order-traversal-ii/ || https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal/ || https://leetcode.com/problems/n-ary-tree-level-order-traversal/
	102. level order traversal				
	105. Construct binary tree from preorder and inorder 106. Construct binary tree from postorder and inorder		https://leetcode.com/problems/construct-binary-tree-from-preorder-and-inorder-traversal/	Must To do	https://leetcode.com/problems/construct-binary-tree-from-inorder-and-postorder-traversal/
	101. Symmetric tree	easy	https://leetcode.com/problems/symmetric-tree/	To do	
	226. Invert binary tree			To do	
	104. Find max depth				https://leetcode.com/problems/minimum-depth-of-binary-tree/ https://leetcode.com/problems/time-needed-to-inform-all-employees/ https://leetcode.com/problems/balanced-binary-tree/
	1038. Binary search tree to greater sum tree				
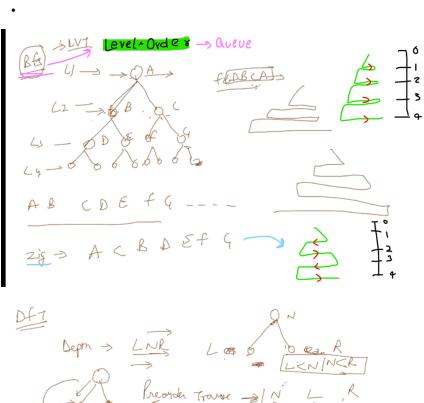
Depth of tree: # of edges between node and root of tree
Height of node(/tree) : # of edges between node(/root) and farthest leaf node
Examples : Height (3) = 3 , Height(1) =4 , Depth(3) = 1 , Depth(1)=0
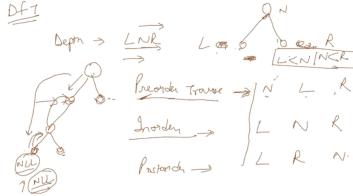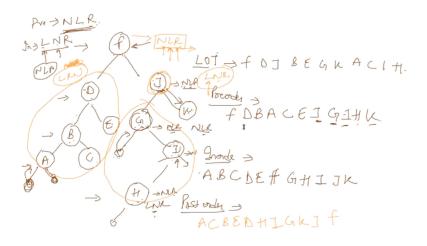
Diameter / Width of tree
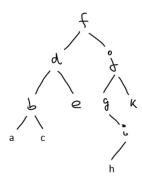
- BFS (in graphs) = Level Order Traversal (in trees)
- DFS (in graphs) = Preorder/Inorder/Postorder Traversal (in trees)

→ LV1   **Level-Order** → Queue

B E →

4 →   → A   f E D B C A

L2 → B   C

L3 → D E f G

L4 →

A B   C D E f G - - - -

Zig →   A C B D E f G →

0
1
2
3
4

0
1
2
3
4

DFT

Depth → LNR →   L   N   R

Preorder Traverse → N   L , R

Inorder →   L   N   R

Postorder →   L   R   N

Pre → NLR.
In → LNR →   f   → NLR
NLR LRN

LOT → f D J B E G K A C I H.

D   J → NLR

W   Preorder →
f D B A C E J G I H K

E   G → NLR NLR

B   I J → Inorder →
A   C   A B C D E f G H I J K

H → NLR
LNR Postorder →
A C B E D H I G K J f
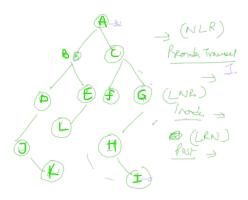
LOT(Level Order Traversal) : f,d,j,b,e,g,k,a,c,i,h
Inorder Traversal : a,b,c,d,e,f,g,h,i,j,k
Postorder Traversal : a,c,b,e,d,h,i,g,k,j,f
Preoder Traversal: f,d,b,a,c,e,j,g,i,h,k

Last element in Pre-order : **i**
Last element in In-order : **g** (Another way to look at the solution: Since g is the greatest element in the BST)
Last element in Post-order : **a**

----

96. Unique Binary Search Trees

Given n, how many structurally unique **BST's** (binary search trees) that store values 1 ... n?

Example:

```
Input: 3
Output: 5
Explanation:
Given n = 3, there are a total of 5
unique BST's:

   1         3     3      2      1
    \       /     /      / \      \
     3     2     1      1   3      2
    /     /       \                 \
   2     1         2                 3
```

Accepted 303.5K   Submissions 578K

```cpp
class Solution {
public:
    int numTrees(int n) {

    }
};
```



```
1 → 1
2 → 2
3 → 5
4 → 14
5 → 42
6 → 132
7 →
```

```cpp
class Solution {
public:
    int numTrees(int n) {
        int dp[n+1];
        dp[0]=1;
        dp[1]=1;
        // dp[n] -> answer for n number of nodes
        for(int nodes=1;nodes<=n;nodes++){
            dp[nodes]=0;
            for(int root=1;root<=nodes;root++){
                dp[nodes]+=(dp[nodes-root]*dp[root-1]);
            }
        }
        return dp[n];
    }
};
```

Testcase   Run Code Result   Debugger 🐞

Time Submitted	Status	Runtime	Men
a few seconds ago	Accepted	0 ms	6.1 l
19 days ago	Accepted	0 ms	6.2 l

```cpp
class Solution {
public:
    int numTrees(int n) {
        vector<int> dp(n+5,-1);int x,mid;
        dp[0]=0,dp[1]=1,dp[2]=2;
        for(int i=3;i<=n;i++){
            x=0;
            for(int j=1;j<=i;j++){ // root as j : 1...i/2-1
                x=x+(max(1,dp[j-1]) * max(1,dp[i-j]));
            }
            // for(int j=1;j<ceil((floor)(i+1)/2);j++){ // root as j : 1....i/2-1
            //     x=x+(max(1,dp[j-1]) * max(1,dp[i-j]));
            // }x*=2;
            // //for middle element
            // mid=dp[i/2];
            // if(i%2==0) mid*=2;
            // x+=mid;
            dp[i]=x;
        }
        // for(auto i:dp) cout<<i<<" ";
        return dp[n];
    }
};
```

Time Submitted	Status	Runtime	Memory	Language
10/19/2021 01:57	Accepted	4 ms	6.1 MB	cpp
10/19/2021 01:56	Runtime Error	N/A	N/A	cpp
10/19/2021 01:31	Wrong Answer	N/A	N/A	cpp

----

94. Binary Tree Inorder Traversal

Easy   👍 5949   👎 252   ♡ Add to List   ⎘ Share

Given the `root` of a binary tree, return *the inorder traversal of its nodes' values*.

Example 1:


```
Input: root = [1,null,2,3]
Output: [1,3,2]
```

Example 2:

📄 Description   △ Solution   🗪 Discuss (999+)   ⊙ Submissions   i C++ ▾   ✴ Autocomplete

```cpp
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    void inorder(TreeNode *node,vector<int> &ans){
        //base case
        if(node==NULL) return;

        //recursive traversal
        // left Root right
        inorder(node->left,ans);
        ans.push_back(node->val);
        inorder(node->right,ans);
    }
    vector<int> inorderTraversal(TreeNode* root) {
        vector<int> ans;
        inorder(root,ans);
        return ans;
    }
};
```

Time Submitted	Status	Runtime	Memory	Language
10/19/2021 13:31	Accepted	4 ms	8.4 MB	cpp

//try iterative approach of the 3 traversals: inorder,preorder,postorder

----

102. Binary Tree Level Order Traversal

Medium   👍 6113   👎 127   ♡ Add to List   🔗 Share

Given the `root` of a binary tree, return *the level order traversal of its nodes' values.* (i.e., from left to right, level by level).

Example 1:


```
Input: root = [3,9,20,null,null,15,7]
Output: [[3],[9,20],[15,7]]
```
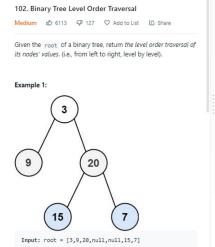
---

102. Binary Tree Level Order Traversal

Medium   👍 6113   👎 127   ♡ Add to List   🔗 Share

Given the `root` of a binary tree, return *the level order traversal of its nodes' values.* (i.e., from left to right, level by level).

Example 1:


```
Input: root = [3,9,20,null,null,15,7]
```

```cpp
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    void levelTraversal(TreeNode* node,vector<vector<int>> &ans,int level){
        if(node==NULL) return;
        // TRICK: to overcome the need for size knowing of 2D ans vector in advance for initialization
        if(level==ans.size()) ans.push_back({});
        //above is step is important/makes the job easy, otherwise ,we had 2 options:
        //1. we first needed to find the height of tree (an extra time to find it)
        //2. or,we had to initialize: vector<vector<int>> ans(2005,vector<int> (0) ); and then trim off the empty lists at the end
        // [[3],[9,20],[15,7],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],...

        //recursive call
        ans[level].push_back(node->val);
        levelTraversal(node->left,ans,level+1);
        levelTraversal(node->right,ans,level+1);
    }
    vector<vector<int>> levelOrder(TreeNode* root) {
        vector<vector<int>> ans;
        // vector<vector<int>> ans(2005,vector<int> (0) );
        levelTraversal(root,ans,0);
        return ans;
    }
};
```

Q) To construct a binary tree from 2 of the following : preorder,inorder,postorder , Which of the combinations will work and why?

Ans)
- Works: Inorder + Preorder
- Works: Inorder + Postorder
- NOT Works : Postorder + Preorder

Reason :
- we can get the **current root node** from either the preorder or postorder
- Now we need to differentiate between the left and right subtree elements for the current node ,which can be done only with help of inorder (left **node** right) ,where the **current_node** acts as a divider , We can then recurse the above 2 steps to build the binary tree

----

Homework after lecture 1:

1. Iterative approach for : pre,post,in -order traversals
2. Pblms : 105,106