1485. Clone Binary Tree With Random Pointer

Medium 318 57 Add to List Share

A binary tree is given such that each node contains an additional random pointer which could point to any node in the tree or null.

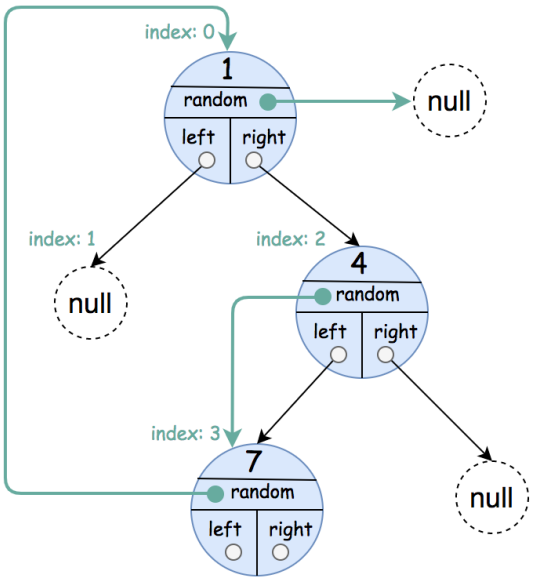
Return a **deep copy** of the tree.

The tree is represented in the same input/output way as normal binary trees where each node is represented as a pair of `[val, random_index]` where:

- `val` : an integer representing `Node.val`
- `random_index` : the index of the node (in the input) where the random pointer points to, or `null` if it does not point to any node.

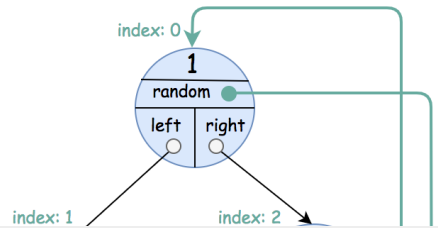
You will be given the tree in class `Node` and you should return the cloned tree in class `NodeCopy` . `NodeCopy` class is just a clone of `Node` class with the same attributes and constructors.

Example 1:




```
Input: root = [[1,null],null,[4,3],[7,0]]
Output: [[1,null],null,[4,3],[7,0]]
Explanation: The original binary tree is [1,null,4,7].
The random pointer of node one is null, so it is represented as [1, null].
The random pointer of node 4 is node 7, so it is represented as [4, 3] where 3 is the
index of node 7 in the array representing the tree.
The random pointer of node 7 is node 1, so it is represented as [7, 0] where 0 is the
index of node 1 in the array representing the tree.
```

Example 2:



C++ Autocomplete

```
/**
 * Definition for a Node.
 * struct Node {
 *     int val;
 *     Node *left;
 *     Node *right;
 *     Node *random;
 *     Node() : val(0), left(nullptr), right(nullptr), random(nullptr) {}
 *     Node(int x) : val(x), left(nullptr), right(nullptr), random(nullptr) {}
 *     Node(int x, Node *left, Node *right, Node *random) : val(x), left(left), right(right), random(random) {}
 * };
 */

class Solution {

 unordered_map<Node*,NodeCopy*> mp;
public:
    NodeCopy* dfs(Node* root)
    {
        if(!root)
            return NULL;

        NodeCopy* curr = new NodeCopy(root->val);
        mp[root] = curr;
        curr->left = dfs(root->left);
        curr->right= dfs(root->right);
        return curr;
    }
    void cloneOne(Node* root,NodeCopy* curr)
    {
        if(!root)
            return;

        curr->random = mp[root->random];
        cloneOne(root->left,curr->left);
        cloneOne(root->right,curr->right);
    }

    NodeCopy* copyRandomBinaryTree(Node* root) {

        NodeCopy* curr =
```

NEW

Testcase Run Code Result Debugger

Accepted Runtime: 0 ms

Your input [[1,6],[2,5],[3,4],[4,3],[5,2],[6,1],

Output [[1,6],[2,5],

Diff

Expected [[1,6],[2,5],[3,4],[4,3],[5,2],[6,1],

Problems Pick One Prev 8/150 Next Example Cases Run Code Submit