剑指 Offer 54. 二叉搜索树的第k大节点

Monday, June 13, 2022 9:29 AM

https://leetcode.cn/problems/er-cha-sou-suo-shu-de-di-kda-jie-dian-lcof/

简单

给定一棵二叉搜索树，请找出其中**第 k 大**的节点的值。

示例 1:

```
输入: root = [3,1,4,null,2], k = 1
   3
  / \
 1   4
  \
   2
输出: 4
```

我的思路：

二叉搜索树左节点小于父节点，右节点大于父节点，中序遍历有序。

故，中序遍历，保存结果至ArryList，再取倒数第k个即可。

code：

```
class Solution {
    public int kthLargest(TreeNode root, int k) {
        // 存中序遍历结果
        ArrayList<Integer> list=new ArrayList<>();
        // 中序遍历
        inOrder(root, list);
        // 取第k大数
        return list.get(list.size()-k);
    }
// 中序遍历：左 中 右；前序遍历： 中 左 右；后序遍历： 左右 中
// 即不同遍历顺序 都是基于 中 而言的，中也指处理节点步骤。
    private void inOrder(TreeNode root, ArrayList<Integer> list){
        // 递归出口
        if(root==null) return;
        // 先左 再中 再右
        inOrder(root.left,list);
        // 处理步骤
        list.add(root.val);
        // 再右
        inOrder(root.right,list);
    }
}
```

优化：

中序遍历时，左 中 右时，遍历的是正序有序；右 中 左时，遍历的是倒序有序。

故增加两个全局变量 count\res，

count 计数当前返回的次数，对应的数为第count大的数。

count==k时，将对应值赋给res。

code：

```
// 中序遍历的代码改为：
// ps：k是主函数局部变量（形参），中序遍历函数访问不到，故每次都传进去。
private void inOrder(TreeNode root,int k){
    // 判空
    if(root==null) return;
    // 先右 再中 再左
    inOrder(root.right,k);
    // 中，处理操作
    count++;
    if(count==k){
        res=root.val;
    }
    // 再右
    inOrder(root.left,k);
}
```

剑指 Offer 55 - I. 二叉树的深度

https://leetcode.cn/problems/er-cha-shu-de-shen-du-lcof/

简单

输入一棵二叉树的根节点，求该树的深度。

从根节点到叶节点依次经过的节点（含根、叶节点）形成树的一条路径，最长路径的长度为树的深度。

例如：

给定二叉树 [3,9,20,null,null,15,7]，

```
  3
 / \
9  20
  /  \
 15   7
```

返回它的最大深度 3 。

解法：

递归遍历左右子树再处理，

最下层空节点返回0，
处理时，选左右子树的最大值，再+1返回，因为最下层返回的是0，每返回上一层需要加1。

code：

```
class Solution {
    public int maxDepth(TreeNode root) {
        // root为空，或是递归最下层的空节点，返回0
        if(root==null) return 0;
        // 递归求左右子树的最大深度
        int left=maxDepth(root.left);
        int right=maxDepth(root.right);
        // 左右都有返回值了，处理当前子树的深度
        // 最底下空节点返回的是0，从下往上返回的，每层深度需要+1。
        int max = Math.max(left,right);
        return max+1;
    }
}
```