#LeetCode285

https://leetcode.com/problems/inorder-successor-in-bst
http://lintcode.com/en/problem/inorder-successor-in-binary-search-tree/

Yifeng Zeng

#题目描述

Inorder Successor in BST

#思路报告

When I got this question, the first idea I have is that we can do an inorder traversal of this tree, when travered to the key node, mark a flag and return the next node that we traversed, O(n) time.

代码如下：

```
public TreeNode inorderSuccessor(TreeNode root, TreeNode p) {
    if (root == null) {
        return root;
    }

    Deque<TreeNode> stack = new ArrayDeque<>();
    TreeNode cur = root;
    TreeNode res = null;
    boolean found = false;

    while (!stack.isEmpty() || cur != null) {
        while (cur != null) {
            stack.push(cur);
            cur = cur.left;
        }
        cur = stack.pop();

        if (found) {
            return cur;
        }
        if (cur == p) {
            found = true;
        }
```

```
            cur = cur.right;
        }

        return res;
    }
```

We can do inorder traversal for any tree, so we didn't leverage the property of the BST. So I can compare the root.val to p.val. If root.val < p.val then we can throw away all the children of root.left so look in root.right (do it recursively). Otherwise if root.val > p.val then we can throw away all the children of root.right, so look in root.left (do it recursively) or it is this root. If root.val == p.val, we just return the smallest node in subtree root.right. In this case, we throw away half of the candidates every time, so this is O(logn) time.
(So from a couple of feedback from my last homeworks, I guess I'm not supposed to discuss the corner cases at begining. So should I discuss the corner case when I write the code?)

Code:

```
public TreeNode inorderSuccessor(TreeNode root, TreeNode p) {
    if (root == null) {
        return null;
    }

    if (root.val > p.val) {
        TreeNode left = inorderSuccessor(root.left, p);
        return left == null ? root : left;
    } else if (root.val < p.val) {
        return inorderSuccessor(root.right, p);
    }
    if (root.right == null) {
        return null;
    }
    root = root.right;
    while (root.left != null) {
        root = root.left;
    }
    return root;
}
```

#套路总结

- I forgot root.val > p.val, root may be the result.

- Split one big problem into a sequence of sub-prolem may do the trick.
- 通过这个题我发现其实有些时候做不出一道题最根本的原因还是思路好象不对，想到了分左边和分右边，但是没有考虑好具体root.val > p.val时其实root本身就是一个candidate。听老师讲了之后有种顿悟的感觉，思路清晰了，代码很容易就码出来了。