小码哥教育 SEEMYGO

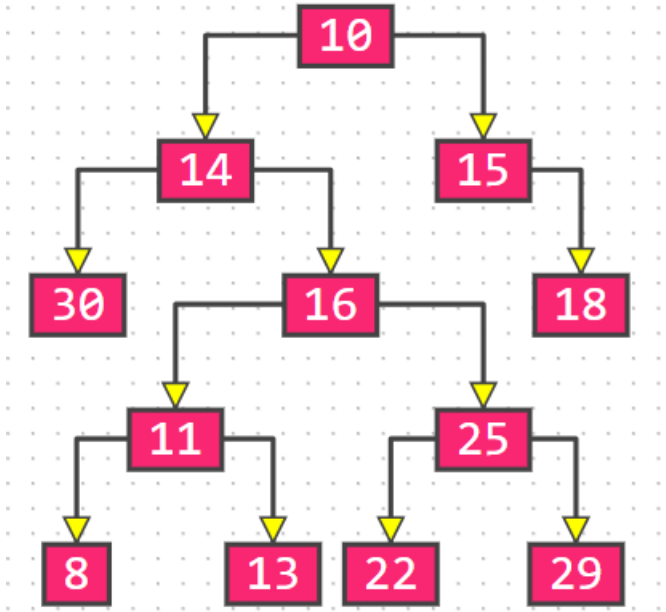
333. 最大BST子树

给定一个二叉树，找到其中最大的二叉搜索树（BST）子树，其中最大指的是子树节点数最多的。

注意:
子树必须包含其所有后代。

输入: [10,5,15,1,8,null,7]

输出: 3
解释: 高亮部分为最大的 BST 子树。
返回值 3 在这个样例中为子树大小。

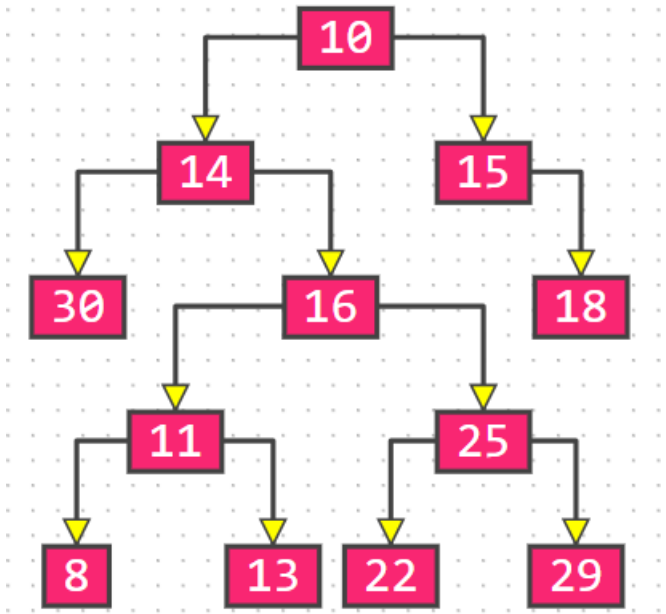


■ 输出：7（以16为根节点的子树）

你能想出用 O(n) 的时间复杂度解决这个问题吗?

小码哥教育 SEEMYGO

自顶向下

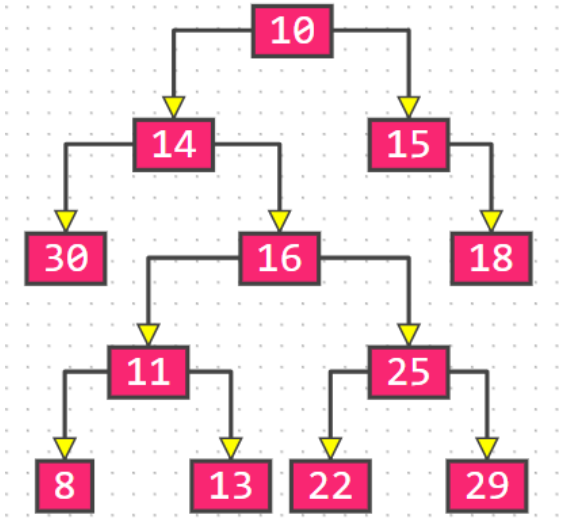



```
public int func(TreeNode root);
```

- func的作用：返回以root为根节点的二叉树的最大BST子树的节点数量
- func的实现
  - 如果以root为根节点的二叉树S是BST，就返回S的节点数量
  - 否则，就返回func(root.left)、func(root.right)中的最大值
- 时间复杂度分析
  - func使用了前序遍历，时间复杂度是O(n)
  - 判断一棵树是否为BST，时间复杂度是O(n)
  - 所以，总体时间复杂度是O(n^2)
- 如何优化?
  - 由于是自顶向下的遍历方式，所以在判断一棵树是否为BST方面，存在重复的遍历判断
  - 可以考虑改为自底向上的遍历方式：后序遍历

小码哥教育 SEEMYGO

自底向上




```
/** 最大BST子树的信息 */
private static class Info {
    /** 根节点 */
    public TreeNode root;
    /** 节点总数 */
    public int size = 1;
    /** 最大值 */
    public int max;
    /** 最小值 */
    public int min;
}
```

```
private Info getInfo(TreeNode root);
```

- getInfo的作用：返回以root为根节点的二叉树的最大BST子树的信息
- getInfo的实现
  - 计算li = getInfo(root.left)，ri = getInfo(root.right)
  - 如果下面的条件成立，说明以root为根节点的二叉树就是最大BST子树
    - ✓ li == null || (li.root == root.left && li.max < root.val)
    - ✓ ri == null || (ri.root == root.right && li.min > root.val)
  - 如果li != null && ri != null
    - ✓ 如果li.size > ri.size，返回li；否则返回ri
  - 如果li != null，返回li；否则返回ri