

# 二叉树 (Binary Tree)

# 二叉树 (Binary Tree)

- 二叉树的绝大部分题目都可以直接通过递归+遍历解决

- 前序遍历

- 中序遍历

- 后序遍历

- 层序遍历

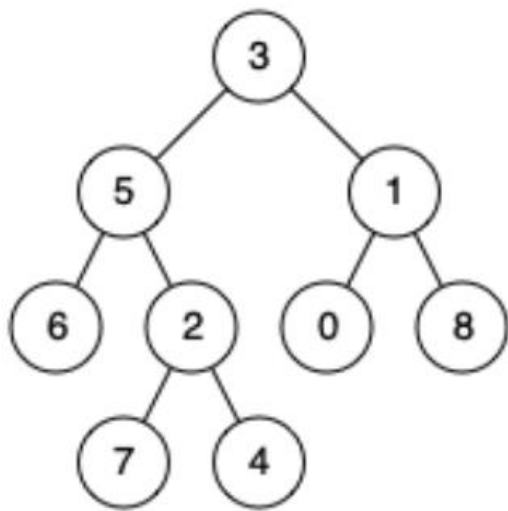
## 236. 二叉树的最近公共祖先

给定一个二叉树, 找到该树中两个指定节点的最近公共祖先。

百度百科中最近公共祖先的定义为：“对于有根树  $T$  的两个结点  $p$ 、 $q$ ，最近公共祖先表示为一个结点  $x$ ，满足  $x$  是  $p$ 、 $q$  的祖先且  $x$  的深度尽可能大（一个节点也可以是它自己的祖先）。”

- 所有节点的值都是唯一的。
- $p$ 、 $q$  为不同节点且均存在于给定的二叉树中。

例如，给定如下二叉树:  $root = [3,5,1,6,2,0,8,null,null,7,4]$



输入:  $root = [3,5,1,6,2,0,8,null,null,7,4]$ ,  $p = 5$ ,  $q = 1$

输出: 3

解释: 节点 5 和节点 1 的最近公共祖先是节点 3。

输入:  $root = [3,5,1,6,2,0,8,null,null,7,4]$ ,  $p = 5$ ,  $q = 4$

输出: 5

解释: 节点 5 和节点 4 的最近公共祖先是节点 5。因为根据定义最近公共祖先节点可以为节点本身。

■ 一样的题目: [面试题68 - II. 二叉树的最近公共祖先](#)

## 99. 恢复二叉搜索树

二叉搜索树中的两个节点被错误地交换。

请在不改变其结构的情况下，恢复这棵树。

- 使用  $O(n)$  空间复杂度的解法很容易实现。
- 你能想出一个只使用常数空间的解决方案吗？

输入: [1,3,null,null,2]

```
  1
 /
3
 \
  2
```

输出: [3,1,null,null,2]

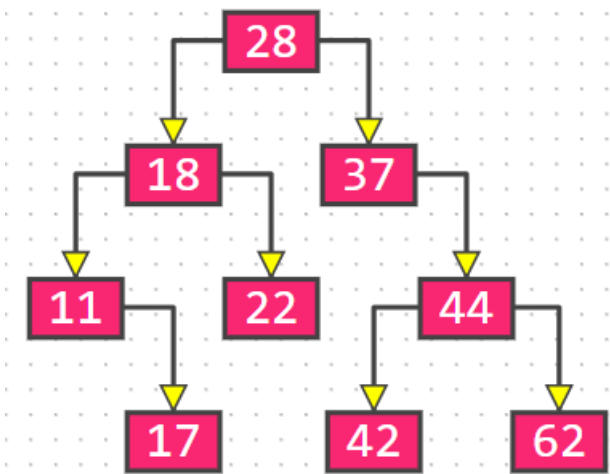
```
  3
 /
1
 \
  2
```

输入: [3,1,4,null,null,2]

```
  3
 / \
1   4
  /
  2
```

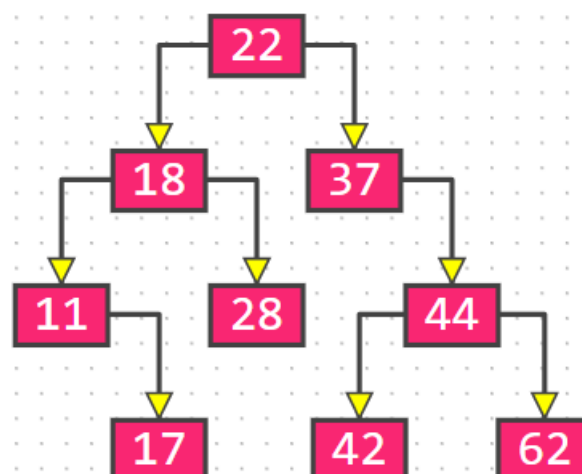
输出: [2,1,4,null,null,3]

```
  2
 / \
1   4
  /
  3
```



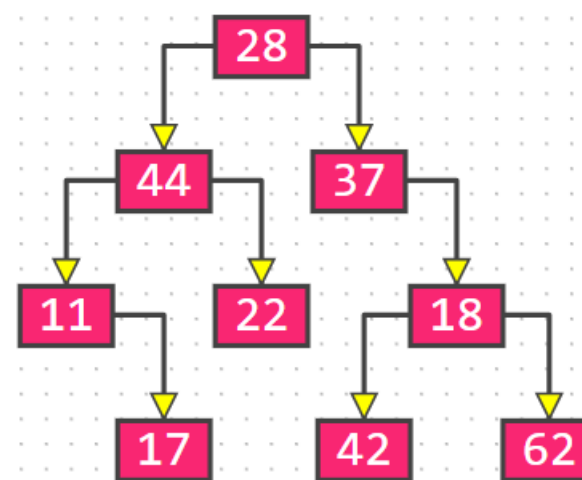
■ 中序遍历的结果是升序的

□ 11, 17, 18, 22, 28, 37, 42, 44, 62



■ 中序遍历的结果

□ 11, 17, 18, 28, 22, 37, 42, 44, 62



■ 中序遍历的结果

□ 11, 17, 44, 22, 28, 37, 42, 18, 62

■ 第1个错误节点: 第1个逆序对中的较大节点

■ 第2个错误节点: 最后1个逆序对中的较小节点

# 二叉树的Morris遍历

- 使用Morris方法遍历二叉树，可以实现时间复杂度 $O(n)$ 、空间复杂度 $O(1)$
- 这里只演示二叉树的Morris中序遍历。前序遍历、后序遍历在此基础上做一些调整即可

■ 执行步骤（假设遍历到当前节点是N）

① 如果 $N.left \neq null$ ，找到N的前驱节点P

□ 如果 $P.right == null$

✓  $P.right = N$

✓  $N = N.left$

✓ 回到①

□ 如果 $P.right == N$

✓  $P.right = null$

✓ 打印N

✓  $N = N.right$

✓ 回到①

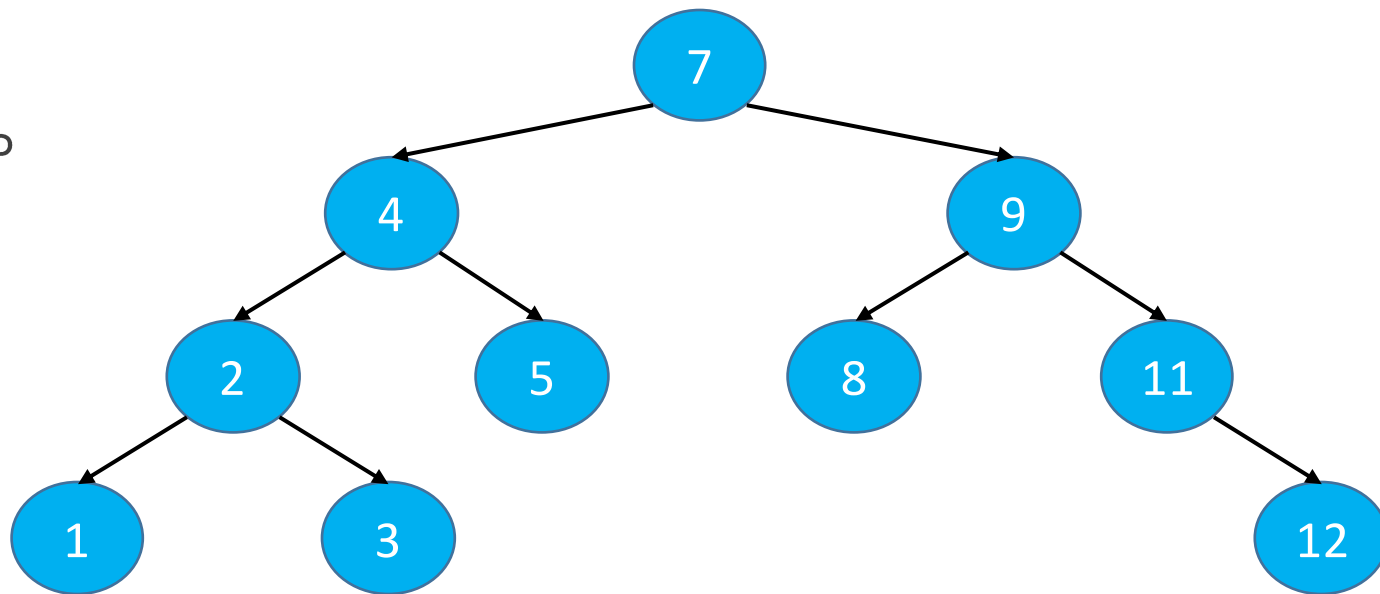
② 如果 $N.left == null$

□ 打印N

□  $N = N.right$

□ 回到①

③ 重复①、②直到 $N == null$



## 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。

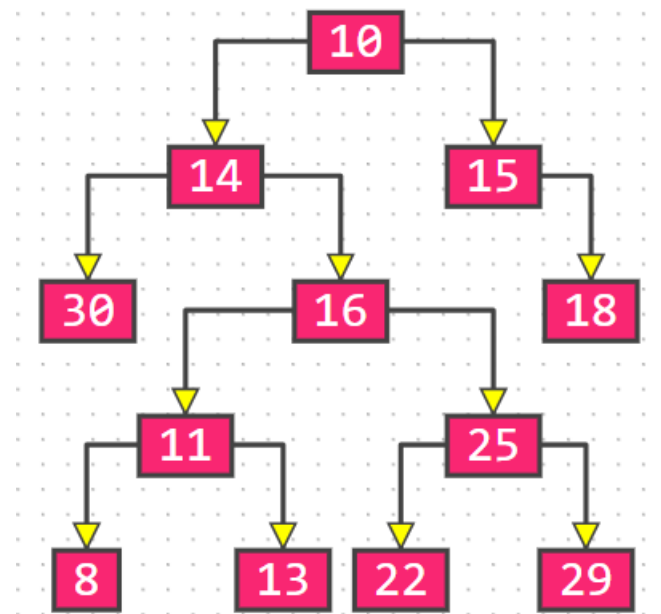
输入：[10,5,15,1,8,null,7]

```
    10
   /  \
  5    15
 /  \   \
1    8   7
```

输出：3

解释：高亮部分为最大的 BST 子树。

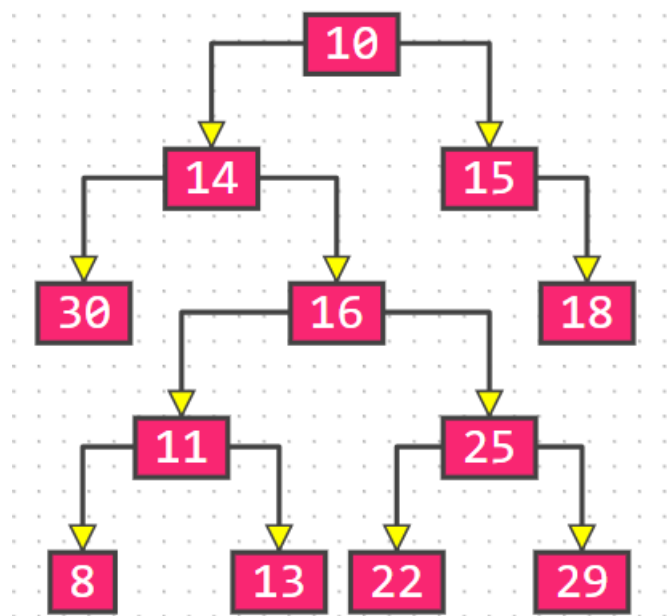
返回值 3 在这个样例中为子树大小。



■ 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？

# 自顶向下



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

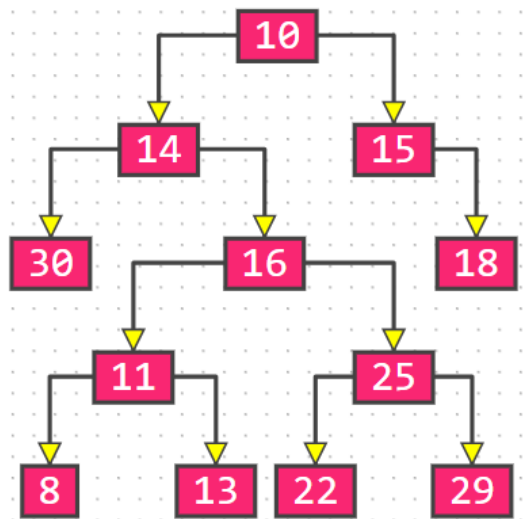
- func的作用：返回以root为根节点的二叉树的最大BST子树的节点数量
- func的实现
  - 如果以root为根节点的二叉树S是BST，就返回S的节点数量
  - 否则，就返回func(root.left)、func(root.right)中的最大值
- 时间复杂度分析
  - func使用了前序遍历，时间复杂度是 $O(n)$
  - 判断一棵树是否为BST，时间复杂度是 $O(n)$
  - 所以，总体时间复杂度是 $O(n^2)$

## ■ 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



# 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

# 作业

- [94. 二叉树的中序遍历](#) ([第一季](#)中讲过)
- [98. 验证二叉搜索树](#) ([第一季](#)中讲过)
- [230. 二叉搜索树中第K小的元素](#)
- [101. 对称二叉树](#)
- [108. 将有序数组转换为二叉搜索树](#)
- [102. 二叉树的层次遍历](#) ([第一季](#)中讲过)
- [104. 二叉树的最大深度](#) ([第一季](#)中讲过)

# 思考题

- [105. 从前序与中序遍历序列构造二叉树](#)
- [106. 从中序与后序遍历序列构造二叉树](#)
- [297. 二叉树的序列化与反序列化](#)
- [449. 序列化和反序列化二叉搜索树](#)