

基本思路

类似 96. 不同的二叉搜索树，这题的思路也是类似的，想要构造出所有合法 BST，分以下三步：

- 1、穷举 `root` 节点的所有可能。
- 2、递归构造出左右子树的所有合法 BST。
- 3、给 `root` 节点穷举所有左右子树的组合。

- 详细题解： [东哥带你刷二叉搜索树（构造篇）](#)

解法代码

```
class Solution {
    /* 主函数 */
    public List<TreeNode> generateTrees(int n) {
        if (n == 0) return new LinkedList<>();
        // 构造闭区间 [1, n] 组成的 BST
        return build(1, n);
    }

    /* 构造闭区间 [lo, hi] 组成的 BST */
    List<TreeNode> build(int lo, int hi) {
        List<TreeNode> res = new LinkedList<>();
        // base case
        if (lo > hi) {
            res.add(null);
            return res;
        }

        // 1、穷举 root 节点的所有可能。
        for (int i = lo; i <= hi; i++) {
            // 2、递归构造出左右子树的所有合法 BST。
            List<TreeNode> leftTree = build(lo, i - 1);
            List<TreeNode> rightTree = build(i + 1, hi);
            // 3、给 root 节点穷举所有左右子树的组合。
            for (TreeNode left : leftTree) {
                for (TreeNode right : rightTree) {
                    // i 作为根节点 root 的值
                    TreeNode root = new TreeNode(i);
                    root.left = left;
                    root.right = right;
                    res.add(root);
                }
            }
        }
    }
}
```

```

    }
  }
  return res;
}

```

- 类似题目：
 - 96. 不同的二叉搜索树 🟡

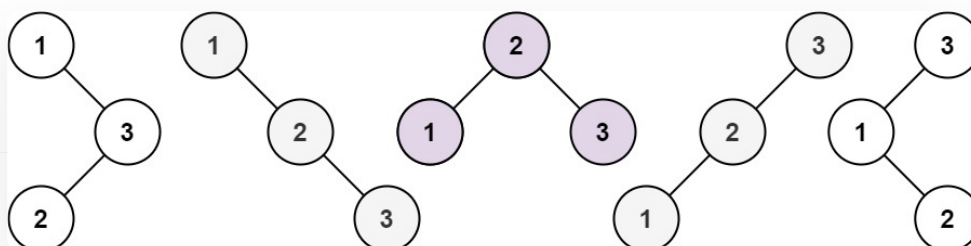
96. 不同的二叉搜索树

LeetCode	力扣	难度
2096. Unique Binary Search Trees	2096. 不同的二叉搜索树	🟡

- 标签：二叉搜索树，数据结构

给你一个整数 n ，求恰由 n 个节点组成且节点值从 1 到 n 互不相同的**二叉搜索树**有多少种？返回满足题意的二叉搜索树的种数。

示例 1：

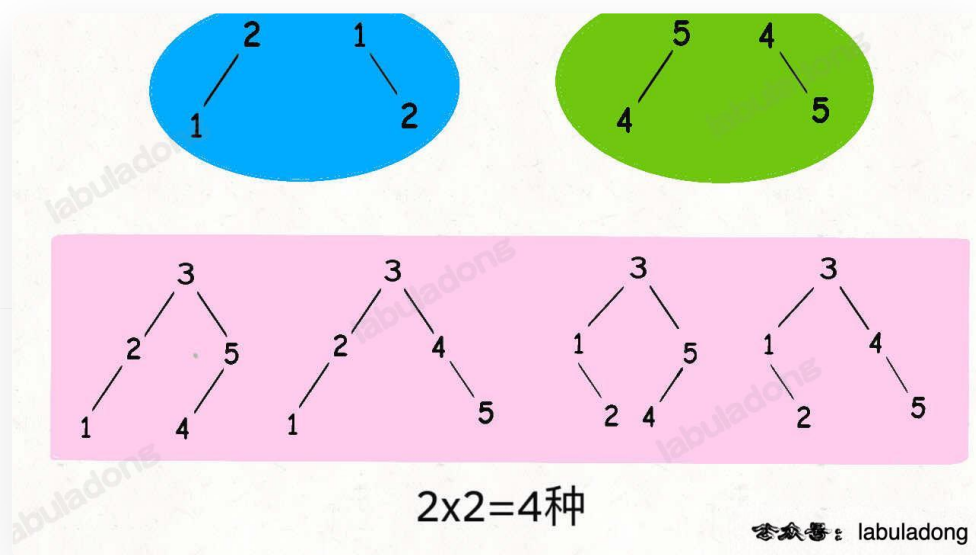


输入： $n = 3$
输出：5

基本思路

假设给算法输入 $n = 5$ ，也就是说用 $\{1, 2, 3, 4, 5\}$ 这些数字去构造 BST。

如果固定 3 作为根节点，左子树节点就是 $\{1, 2\}$ 的组合，右子树就是 $\{4, 5\}$ 的组合：



那么 $\{1, 2\}$ 和 $\{4, 5\}$ 的组合有多少种呢？只要合理定义递归函数，这些可以交给递归函数去做。

另外，这题存在重叠子问题，可以通过备忘录的方式消除冗余计算。

- 详细题解： [东哥带你刷二叉搜索树（构造篇）](#)

解法代码

```
class Solution {
    // 备忘录
    int[][] memo;

    int numTrees(int n) {
        // 备忘录的值初始化为 0
        memo = new int[n + 1][n + 1];
        return count(1, n);
    }

    int count(int lo, int hi) {
        if (lo > hi) return 1;
        // 查备忘录
```

```

    if (memo[lo][hi] != 0) {
        return memo[lo][hi];
    }

    int res = 0;
    for (int mid = lo; mid <= hi; mid++) {
        int left = count(lo, mid - 1);
        int right = count(mid + 1, hi);
        res += left * right;
    }
    // 将结果存入备忘录
    memo[lo][hi] = res;

    return res;
}
}

```

• 类似题目：

- 95. 不同的二叉搜索树 II ●

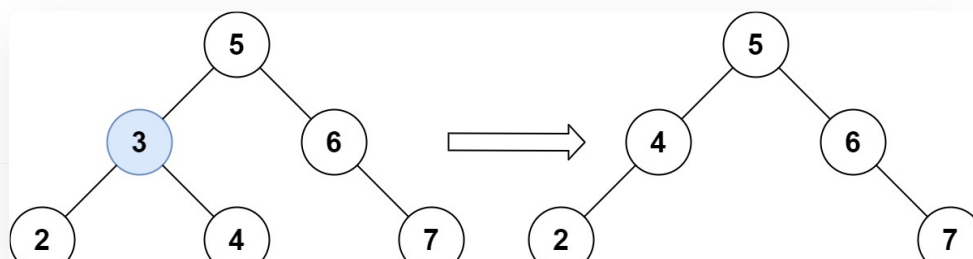
450. 删除二叉搜索树中的节点

LeetCode	力扣	难度
2096. Delete Node in a BST	2096. 删除二叉搜索树中的节点	●

• 标签：二叉搜索树，数据结构

给定一个二叉搜索树的根节点 `root` 和一个值 `key`，删除二叉搜索树中的 `key` 对应的节点，并保证二叉搜索树的性质不变，返回删除后的二叉搜索树的根节点（有可能被更新）。

示例 1:



输入: root = [5,3,6,2,4,null,7], key = 3

输出: [5,4,6,2,null,null,7]

解释: 给定需要删除的节点值是 3, 所以我们首先找到 3 这个节点, 然后删除它。

一个正确的答案是 [5,4,6,2,null,null,7], 如下图所示。

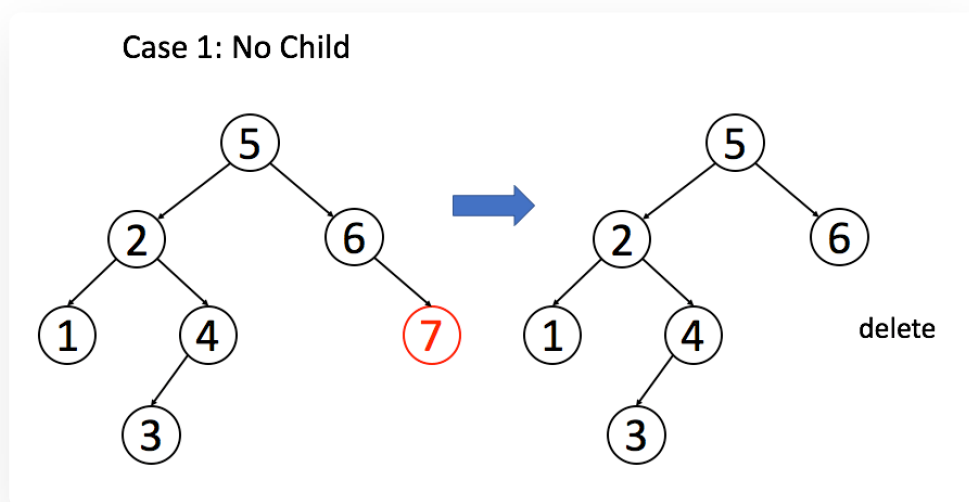
另一个正确答案是 [5,2,6,null,4,null,7]。

基本思路

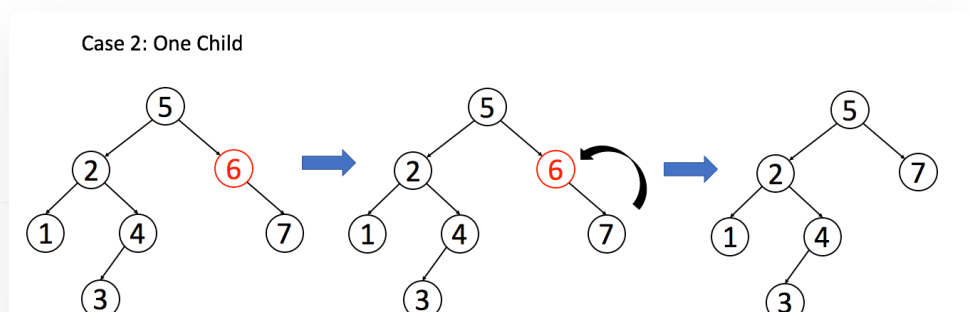
PS: 这道题在 《算法小抄》 的第 235 页。

删除比插入和搜索都要复杂一些, 分三种情况:

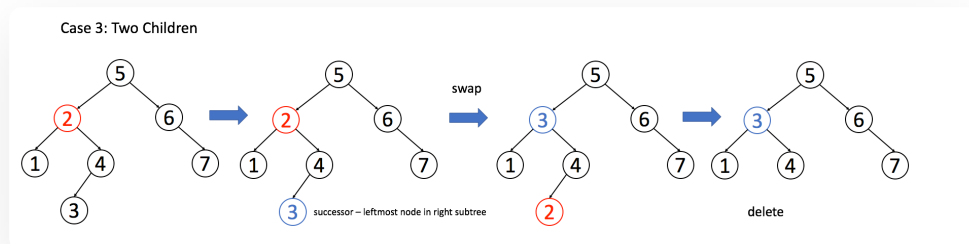
情况 1: A 恰好是末端节点, 两个子节点都为空, 那么它可以当场去世了:



情况 2: A 只有一个非空子节点, 那么它要让这个孩子接替自己的位置:



情况 3: A 有两个子节点，麻烦了，为了不破坏 BST 的性质，A 必须找到左子树中最大的那个节点或者右子树中最小的那个节点来接替自己，我的解法是用右子树中最小节点来替换：






- 详细题解： [东哥带你刷二叉搜索树（基操篇）](#)

解法代码


```
class Solution {
    public TreeNode deleteNode(TreeNode root, int key) {
        if (root == null) return null;
        if (root.val == key) {
            // 这两个 if 把情况 1 和 2 都正确处理了
            if (root.left == null) return root.right;
            if (root.right == null) return root.left;
            // 处理情况 3
            // 获得右子树最小的节点
            TreeNode minNode = getMin(root.right);
            // 删除右子树最小的节点
            root.right = deleteNode(root.right, minNode.val);
            // 用右子树最小的节点替换 root 节点
            minNode.left = root.left;
            minNode.right = root.right;
            root = minNode;
        } else if (root.val > key) {
            root.left = deleteNode(root.left, key);
        } else if (root.val < key) {
            root.right = deleteNode(root.right, key);
        }
        return root;
    }
}
```

```
TreeNode getMin(TreeNode node) {  
    // BST 最左边的就是最小的  
    while (node.left != null) node = node.left;  
    return node;  
}
```

• 类似题目：

- 98. 验证二叉搜索树 
- 700. 二叉搜索树中的搜索 
- 701. 二叉搜索树中的插入操作 

700. 二叉搜索树中的搜索

LeetCode	力扣	难度
2096. Search in a Binary Search Tree	2096. 二叉搜索树中的搜索	

• 标签：二叉搜索树，数据结构

给定二叉搜索树（BST）的根节点和一个目标值。你需要在 BST 中找到节点值等于目标值的节点并返回，如果节点不存在，则返回 NULL。

例如，

给定二叉搜索树：



目标值：2

你应该返回节点 2。

基本思路




PS：这道题在 《算法小抄》 的第 235 页。

利用 BST 左小右大的特性，可以避免搜索整棵二叉树去寻找元素，从而提升效率。

- 详细题解： [东哥带你刷二叉搜索树（基操篇）](#)

解法代码

```
class Solution {
    public TreeNode searchBST(TreeNode root, int target) {
        if (root == null) {
            return null;
        }
        // 去左子树搜索
        if (root.val > target) {
            return searchBST(root.left, target);
        }
        // 去右子树搜索
        if (root.val < target) {
            return searchBST(root.right, target);
        }
        return root;
    }
}
```

- 类似题目：
 - 98. 验证二叉搜索树 
 - 450. 删除二叉搜索树中的节点 
 - 701. 二叉搜索树中的插入操作 

701. 二叉搜索树中的插入操作

LeetCode

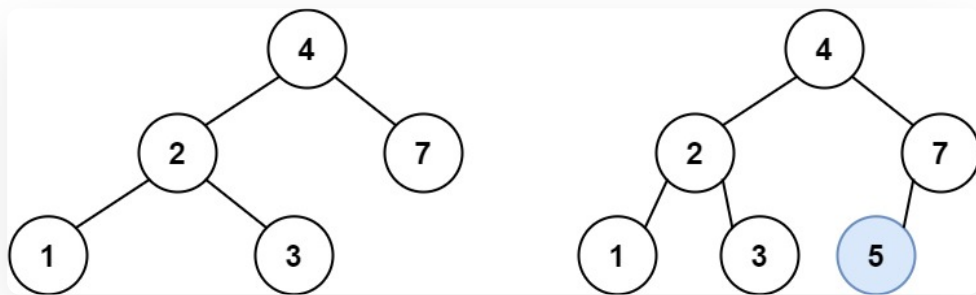
力扣

难度

- 标签：二叉搜索树，数据结构

给定二叉搜索树（BST）的根节点和要插入树中的值（不会插入 BST 已存在的值），将值插入二叉搜索树，返回插入后二叉搜索树的根节点。

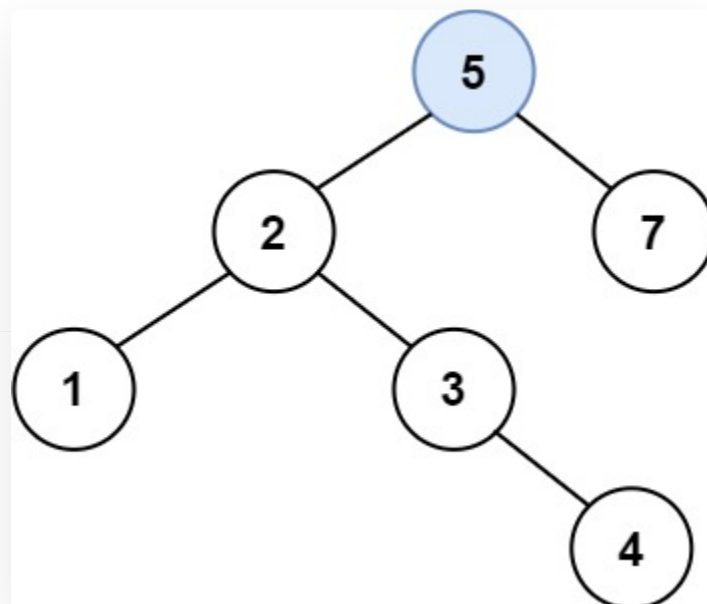
示例 1:



输入: root = [4,2,7,1,3], val = 5

输出: [4,2,7,1,3,5]

解释: 另一个满足题目要求可以通过的树是:



基本思路

PS：这道题在 《算法小抄》 的第 235 页。

如果要递归地插入或者删除二叉树节点，递归函数一定要有返回值，而且返回值要被正确的接收。

插入的过程可以分两部分：




- 1、寻找正确的插入位置，类似 700. 二叉搜索树中的搜索。
- 2、把元素插进去，这就要把新节点以返回值的方式接到父节点上。

- 详细题解： [东哥带你刷二叉搜索树（基操篇）](#)

解法代码

```
class Solution {  
    public TreeNode insertIntoBST(TreeNode root, int val) {  
        // 找到空位置插入新节点  
        if (root == null) return new TreeNode(val);  
        // if (root.val == val)  
        //     BST 中一般不会插入已存在元素  
        if (root.val < val)  
            root.right = insertIntoBST(root.right, val);  
        if (root.val > val)  
            root.left = insertIntoBST(root.left, val);  
        return root;  
    }  
}
```

- 类似题目：

- 98. 验证二叉搜索树 
- 450. 删除二叉搜索树中的节点 
- 700. 二叉搜索树中的搜索 

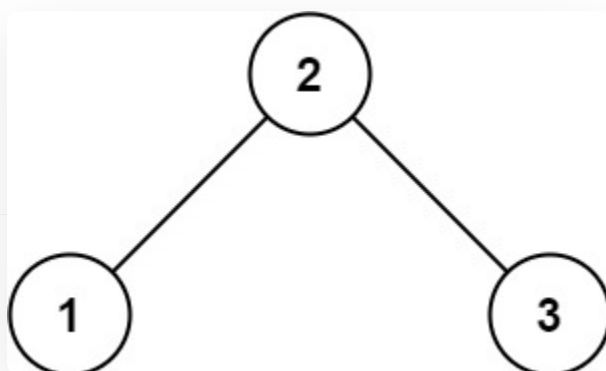
98. 验证二叉搜索树

LeetCode	力扣	难度
2096. Validate Binary Search Tree	2096. 验证二叉搜索树	●

- 标签：二叉搜索树，数据结构

给你一个二叉树的根节点 `root`，判断其是否是一个有效的二叉搜索树。

示例 1:



输入: `root = [2,1,3]`

输出: `true`

基本思路

PS：这道题在《算法小抄》的第 235 页。

初学者做这题很容易有误区：BST 不是左小右大么，那我只要检查 `root.val > root.left.val` 且 `root.val < root.right.val` 不就行了？

这样是不对的，因为 BST 左小右大的特性是指 `root.val` 要比左子树的所有节点都更大，要比右子树的所有节点都小，你只检查左右两个子节点当然是不够的。

正确解法是通过使用辅助函数，增加函数参数列表，在参数中携带额外信息，将这种约束传递给子树的所有节点，这也是二叉搜索树算法的一个小技巧吧。




- 详细题解： [东哥带你刷二叉搜索树（基操篇）](#)

解法代码


```
class Solution {
    public boolean isValidBST(TreeNode root) {
        return isValidBST(root, null, null);
    }

    /* 限定以 root 为根的子树节点必须满足 max.val > root.val > min.val */
    boolean isValidBST(TreeNode root, TreeNode min, TreeNode max) {
        // base case
        if (root == null) return true;
        // 若 root.val 不符合 max 和 min 的限制，说明不是合法 BST
        if (min != null && root.val <= min.val) return false;
        if (max != null && root.val >= max.val) return false;
        // 限定左子树的最大值是 root.val，右子树的最小值是 root.val
        return isValidBST(root.left, min, root)
            && isValidBST(root.right, root, max);
    }
}
```

- 类似题目：

- 450. 删除二叉搜索树中的节点 
- 700. 二叉搜索树中的搜索 
- 701. 二叉搜索树中的插入操作 

1038. 把二叉搜索树转换为累加树

LeetCode	力扣	难度
2096. Binary Search Tree to Greater Sum Tree	2096. 把二叉搜索树转换为累加树	

- 标签： [二叉搜索树](#)

给定一个二叉搜索树，请将它的每个节点的值替换成树中大于或者等于该节点值的所有节点值之和。

示例 1:



输入: [4,1,6,0,2,5,7,null,null,null,3,null,null,null,8]

输出: [30,36,21,36,35,26,15,null,null,null,33,null,null,null,8]

基本思路

和第 538. 把二叉搜索树转换为累加树 一模一样，这里就不多解释了。

- 详细题解: [东哥带你刷二叉搜索树 \(特性篇\)](#)

解法代码

```
class Solution {
    public TreeNode bstToGst(TreeNode root) {
        traverse(root);
        return root;
    }

    // 记录累加和
    int sum = 0;
    void traverse(TreeNode root) {
        if (root == null) {
            return;
        }
        traverse(root.right);
        // 维护累加和
        sum += root.val;
        // 将 BST 转化成累加树
        root.val = sum;
        traverse(root.left);
    }
}
```

```
}  
}
```

• 类似题目：

- 230. 二叉搜索树中第K小的元素 ●
- 538. 把二叉搜索树转换为累加树 ●
- 剑指 Offer II 054. 所有大于等于节点的值之和 ●

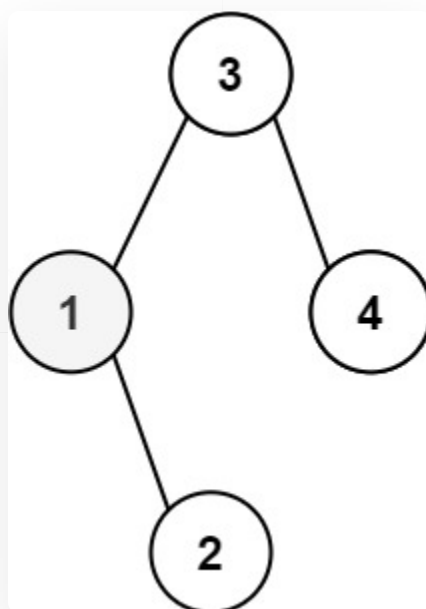
230. 二叉搜索树中第 k 小的元素

LeetCode	力扣	难度
2096. Kth Smallest Element in a BST	2096. 二叉搜索树中第K小的元素	●

• 标签：二叉搜索树，数据结构

给定一个二叉搜索树的根节点 `root`，和一个整数 `k`，请你设计一个算法查找其中第 `k` 个最小元素（从 1 开始计数）。

示例 1：



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

基本思路

BST 的中序遍历结果是有序的（升序），所以用一个外部变量记录中序遍历结果第 `k` 个元素即是第 `k` 小的元素。

- 详细题解: [东哥带你刷二叉搜索树（特性篇）](#)

解法代码


```

class Solution {
    public int kthSmallest(TreeNode root, int k) {
        // 利用 BST 的中序遍历特性
        traverse(root, k);
        return res;
    }

    // 记录结果
    int res = 0;
    // 记录当前元素的排名
    int rank = 0;
    void traverse(TreeNode root, int k) {
        if (root == null) {
            return;
        }
        traverse(root.left, k);
        /* 中序遍历代码位置 */
        rank++;
        if (k == rank) {
            // 找到第 k 小的元素
            res = root.val;
            return;
        }
        /***/
        traverse(root.right, k);
    }
}

```

• 类似题目：

- 538. 把二叉搜索树转换为累加树 🟡
- 1038. 把二叉搜索树转换为累加树 🟡
- 剑指 Offer II 054. 所有大于等于节点的值之和 🟡

538. 把二叉搜索树转换为累加树

LeetCode	力扣	难度
2096. Convert BST to Greater Tree	2096. 把二叉搜索树转换为累加树	🟡

- 标签：二叉搜索树，数据结构

给出二叉搜索树的根节点，该树的节点值各不相同，请你将其转换为累加树（Greater Sum Tree），使每个节点 `node` 的新值等于原树中大于或等于 `node.val` 的值之和。

PS：本题和 1038. 把二叉搜索树转换为累加树 相同。

示例 1：



输入：[4,1,6,0,2,5,7,null,null,null,3,null,null,null,8]

输出：[30,36,21,36,35,26,15,null,null,null,33,null,null,null,8]

基本思路

前文 [手把手刷二叉树总结篇](#) 说过二叉树的递归分为「遍历」和「分解问题」两种思维模式，这道题需要用到「遍历」的思维。

维护一个外部累加变量 `sum`，在遍历 BST 的过程中增加 `sum`，同时把 `sum` 赋值给 BST 中的每一个节点，就将 BST 转化成累加树了。

但是注意顺序，正常的中序遍历顺序是先左子树后右子树，这里需要反过来，先右子树后左子树。

- 详细题解： [东哥带你刷二叉搜索树（特性篇）](#)

解法代码

```
class Solution {
    public TreeNode convertBST(TreeNode root) {
        traverse(root);
        return root;
    }
}
```

```

// 记录累加和
int sum = 0;
void traverse(TreeNode root) {
    if (root == null) {
        return;
    }
    traverse(root.right);
    // 维护累加和
    sum += root.val;
    // 将 BST 转化成累加树
    root.val = sum;
    traverse(root.left);
}
}

```

• 类似题目：

- 230. 二叉搜索树中第K小的元素 ●
- 1038. 把二叉搜索树转换为累加树 ●
- 剑指 Offer II 054. 所有大于等于节点的值之和 ●

501. 二叉搜索树中的众数

LeetCode	力扣	难度
2096. Find Mode in Binary Search Tree	2096. 二叉搜索树中的众数	●

• 标签：二叉搜索树，二叉树

给定一个有相同值的二叉搜索树（BST），找出 BST 中的所有众数（出现频率最高的元素）。

例如给定 BST `[1,null,2,2]`，返回 `[2]`。

```

1
 \
 2
 /
2

```

提示：如果众数超过 1 个，不需考虑输出顺序

基本思路

前文 [手把手刷二叉树总结篇](#) 说过二叉树的递归分为「遍历」和「分解问题」两种思维模式，这道题需要用到「遍历」的思维。

BST 的中序遍历有序，在中序遍历的位置做一些判断逻辑和操作有序数组差不多，很容易找出众数。

解法代码

```
class Solution {
    ArrayList<Integer> mode = new ArrayList<>();
    TreeNode prev = null;
    // 当前元素的重复次数
    int curCount = 0;
    // 全局的最长相同序列长度
    int maxCount = 0;

    public int[] findMode(TreeNode root) {
        // 执行中序遍历
        traverse(root);

        int[] res = new int[mode.size()];
        for (int i = 0; i < res.length; i++) {
            res[i] = mode.get(i);
        }
        return res;
    }

    void traverse(TreeNode root) {
        if (root == null) {
            return;
        }
        traverse(root.left);

        // 中序遍历位置
        if (prev == null) {
            // 初始化
            curCount = 1;
            maxCount = 1;
        }
```

```


        mode.add(root.val);
    } else {
        if (root.val == prev.val) {
            // root.val 重复的情况
            curCount++;
            if (curCount == maxCount) {
                // root.val 是众数
                mode.add(root.val);
            } else if (curCount > maxCount) {
                // 更新众数
                mode.clear();
                maxCount = curCount;
                mode.add(root.val);
            }
        }

        if (root.val != prev.val) {
            // root.val 不重复的情况
            curCount = 1;
            if (curCount == maxCount) {
                mode.add(root.val);
            }
        }
    }
    // 别忘了更新 prev
    prev = root;

    traverse(root.right);
}
}

```

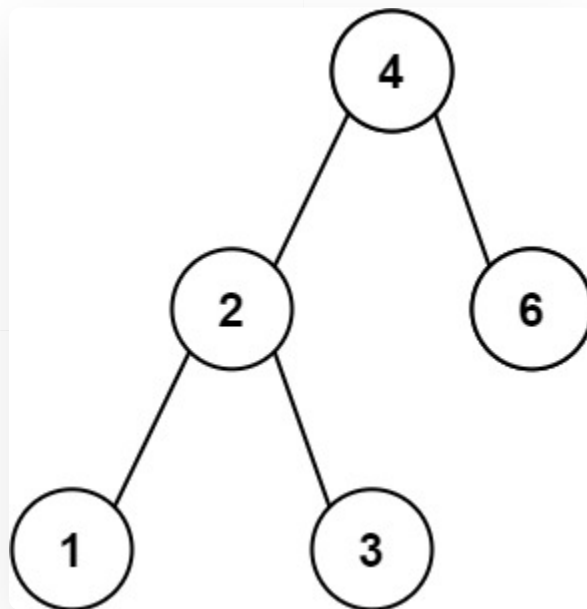
530. 二叉搜索树的最小绝对差

LeetCode	力扣	难度
2096. Minimum Absolute Difference in BST	2096. 二叉搜索树的最小绝对差	

• 标签：二叉搜索树

给你一个二叉搜索树的根节点 `root`，返回树中任意两不同节点值之间的最小差值。差值是一个正数，其数值等于两值之差的绝对值。

示例 1:



输入: root = [4,2,6,1,3]

输出: 1

基本思路

前文 [手把手刷二叉树总结篇](#) 说过二叉树的递归分为「遍历」和「分解问题」两种思维模式，这道题需要用到「遍历」的思维。

中序遍历会有序遍历 BST 的节点，遍历过程中计算最小差值即可。

解法代码

```
class Solution {  
    public int getMinimumDifference(TreeNode root) {  
        traverse(root);  
        return res;  
    }  
}
```

```
TreeNode prev = null;
```

```

int res = Integer.MAX_VALUE;

// 遍历函数
void traverse(TreeNode root) {
    if (root == null) {
        return;
    }
    traverse(root.left);
    // 中序遍历位置
    if (prev != null) {
        res = Math.min(res, root.val - prev.val);
    }
    prev = root;
    traverse(root.right);
}
}

```

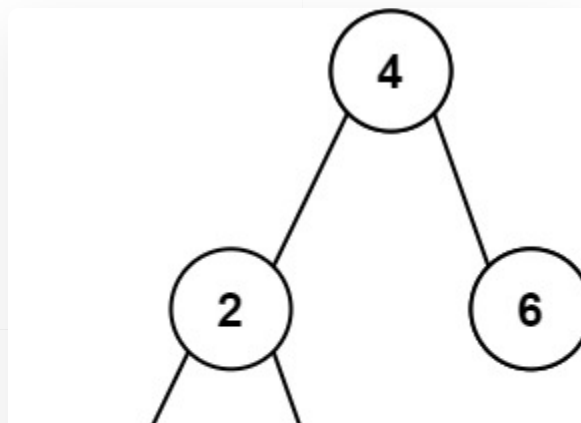
783. 二叉搜索树节点最小距离

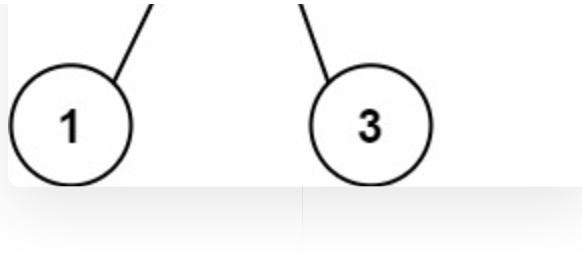
LeetCode	力扣	难度
2096. Minimum Distance Between BST Nodes	2096. 二叉搜索树节点最小距离	●

• 标签：二叉搜索树

给你一个二叉搜索树的根节点 `root`，返回**树中任意两不同节点值之间的最小差值**。差值是一个正数，其数值等于两值之差的绝对值。

示例 1:





输入: root = [4,2,6,1,3]

输出: 1

基本思路

这题和 [530. 二叉搜索树的最小绝对差](#) 完全一样，可去那道题看下思路。

解法代码

```
class Solution {
    public int minDiffInBST(TreeNode root) {
        traverse(root);
        return res;
    }

    TreeNode prev = null;
    int res = Integer.MAX_VALUE;

    // 遍历函数
    void traverse(TreeNode root) {
        if (root == null) {
            return;
        }
        traverse(root.left);
        // 中序遍历位置
        if (prev != null) {
            res = Math.min(res, root.val - prev.val);
        }
        prev = root;
        traverse(root.right);
    }
}
```


1373. 二叉搜索子树的最大键值和

LeetCode	力扣	难度
2096. Maximum Sum BST in Binary Tree	2096. 二叉搜索子树的最大键值和	●

- 标签：二叉搜索树

给你一棵以 `root` 为根的**普通二叉树**，请你返回这棵普通二叉树中所有**二叉搜索子树**的最大节点和。

示例 1:



输入: `root = [1,4,3,2,4,2,5,null,null,null,null,null,null,4,6]`
输出: `20`
解释: 以节点 `3` 为根的子树是二叉搜索树，且节点之和是所有二叉搜索子树中最大的。

基本思路

- 二叉树相关题目最核心的思路是明确当前节点需要做的事情是什么，那么我们想计算子树中 BST 的最大和，站在当前节点的视角，需要做什么呢？
- 1、我肯定得知道左右子树是不是合法的 BST，如果这俩儿子有一个不是 BST，以我为根的这棵树肯定不会是 BST，对吧。
 - 2、如果左右子树都是合法的 BST，我得瞅瞅左右子树加上自己还是不是合法的 BST 了。因为按照 BST 的定义，当前节点的值应该大于左子树的最大值，小于右子树的最小值，否则就破坏了 BST 的性质。
 - 3、因为题目要计算最大的节点之和，如果左右子树加上我自己还是一棵合法的 BST，也就是说以我为根的整棵树是一棵 BST，那我们需要知道我们这棵 BST 的所有节点值之和是多少，方便和别的

BST 争个高下，对吧。

简单说就是要知道以下具体信息：

- 1、左右子树是否是 BST。
- 2、左子树的最大值和右子树的最小值。
- 3、左右子树的节点值之和。

想要获得子树的信息，就要用到前文 [手把手刷二叉树总结篇](#) 说过的后序位置的妙用了。

我们定义一个 `traverse` 函数，`traverse(root)` 返回一个大小为 4 的 int 数组，我们暂且称它为 `res`，其中：

`res[0]` 记录以 `root` 为根的二叉树是否是 BST，若为 1 则说明是 BST，若为 0 则说明不是 BST；

`res[1]` 记录以 `root` 为根的二叉树所有节点中的最小值；

`res[2]` 记录以 `root` 为根的二叉树所有节点中的最大值；

`res[3]` 记录以 `root` 为根的二叉树所有节点值之和。

按照上述思路理解代码。

- 详细题解： [美团面试官：你对后序遍历一无所知](#)

解法代码

```
class Solution {
    // 全局变量，记录 BST 最大节点之和
    int maxSum = 0;

    /* 主函数 */
    public int maxSumBST(TreeNode root) {
        traverse(root);
        return maxSum;
    }

    int[] traverse(TreeNode root) {
        // base case
```

```

    if (root == null) {
        return new int[] {
            1, Integer.MAX_VALUE, Integer.MIN_VALUE, 0
        };
    }

    // 递归计算左右子树
    int[] left = traverse(root.left);
    int[] right = traverse(root.right);

    /*****后序遍历位置*****/
    int[] res = new int[4];
    // 这个 if 在判断以 root 为根的二叉树是不是 BST
    if (left[0] == 1 && right[0] == 1 &&
        root.val > left[2] && root.val < right[1]) {
        // 以 root 为根的二叉树是 BST
        res[0] = 1;
        // 计算以 root 为根的这棵 BST 的最小值
        res[1] = Math.min(left[1], root.val);
        // 计算以 root 为根的这棵 BST 的最大值
        res[2] = Math.max(right[2], root.val);
        // 计算以 root 为根的这棵 BST 所有节点之和
        res[3] = left[3] + right[3] + root.val;
        // 更新全局变量
        maxSum = Math.max(maxSum, res[3]);
    } else {
        // 以 root 为根的二叉树不是 BST
        res[0] = 0;
        // 其他的值都没必要计算了，因为用不到
    }
    /*****/

    return res;
}
}

```

共同维护高质量学习环境，评论礼仪[见这里](#)，违者直接拉黑不解释

6 Comments - powered by utteranc.es

JingweiZuo commented on Mar 15, 2022

如何让每一个节点知道自己的排名？每个节点需要记录，以自己为根的这棵二叉树有多少个节点。有了 size 字段，外加 BST 节点左小右大的性质，对于每个节点 node 就可以通过 node.left 推导出 node 的排名