王争的算法训练营

习题课：二叉树

二叉树相关题型/套路

王争的算法训练营

重点中的重点，面试常考，是个硬骨头，一定要拿下

1）类似链表操作，二叉树上的操作一般都会涉及到树的遍历。

2）因为树跟子树结构、操作很相似，所以，绝大部分题目都使用递归实现。

二叉树的8类小题型：

题型1：二叉树前中后序遍历

题型2：二叉树按层遍历

题型3：二叉树上的递归

题型4：二叉查找树

题型5：LCA最近公共祖先

题型6：二叉树转单、双、循环链表

题型7：按照遍历结果反向构建二叉树

题型8：二叉树上的最长路径和（树形DP）

习题课：二叉树

王争的算法训练营

题型1：二叉树前中后序遍历

144. 二叉树的前序遍历（简单）
94. 二叉树的中序遍历（简单）
145. 二叉树的后序遍历（简单）
589. N 叉树的前序遍历（简单）
590. N 叉树的后序遍历（简单）

题型2：二叉树按层遍历

剑指 Offer 32 - I. 从上到下打印二叉树（中等）
102. 二叉树的层序遍历（中等）
剑指 Offer 32 - III. 从上到下打印二叉树 III（中等）
429. N 叉树的层序遍历（中等）
513. 找树左下角的值（中等）

题型3：二叉树上的递归

104. 二叉树的最大深度（简单）
559. N 叉树的最大深度（简单）
剑指 Offer 55 - II. 平衡二叉树（中等）
617. 合并二叉树（简单）
226. 翻转二叉树（简单）
101. 对称二叉树（中等）
98. 验证二叉搜索树（中等）

题型4：二叉查找树

剑指 Offer 54. 二叉搜索树的第k大节点（中等）
538. 把二叉搜索树转换为累加树（中等）
面试题 04.06. 后继者（中等）

题型5：LCA最近公共祖先

236. 二叉树的最近公共祖先（中等）
剑指 Offer 68 - I. 二叉搜索树的最近公共祖先（中等）

题型6：二叉树转单、双、循环链表

114. 二叉树展开为链表（中等）
面试题 17.12. BiNode（中等）
剑指 Offer 36. 二叉搜索树与双向链表（中等）
面试题 04.03. 特定深度节点链表（中等）

题型7：按照遍历结果反向构建二叉树

105. 从前序与中序遍历序列构造二叉树（中等）
106. 从中序与后序遍历序列构造二叉树（中等）
889. 根据前序和后序遍历构造二叉树（中等）
剑指 Offer 33. 二叉搜索树的后序遍历序列（中等）

题型8：二叉树上的最长路径和

543. 二叉树的直径（简单）
剑指 Offer 34. 二叉树中和为某一值的路径（中等）
124. 二叉树中的最大路径和（困难）
437. 路径总和 III（困难）

习题课：二叉树

王争的算法训练营

题型5：LCA最近公共祖先

236. 二叉树的最近公共祖先（中等） 已讲

剑指 Offer 68 - I. 二叉搜索树的最近公共祖先（中等）

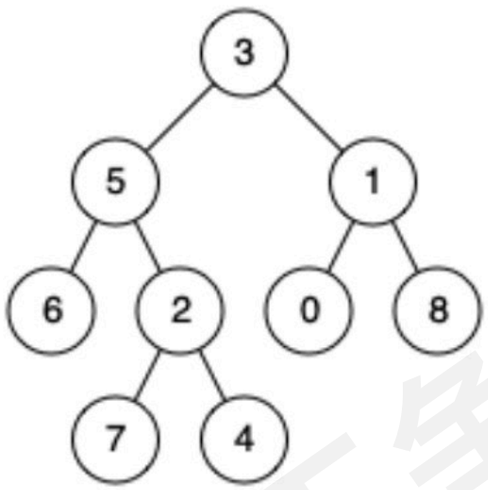
二叉树相关题型/套路

题型5：LCA最近公共祖先 236. 二叉树的最近公共祖先（中等）

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 的两个节点 p、q，最近公共祖先表示为一个节点 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**）。”

示例 1：




```
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 。
```

```java
class Solution {
    private TreeNode lca = null;

    public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
        traverse(root, p, q);
        return lca;
    }

    private int traverse(TreeNode root, TreeNode p, TreeNode q) {
        if (root == null) return 0;
        int leftContains = traverse(root.left, p, q);
        if (lca != null) { // 提前退出
            return 2;
        }
        int rightContains = traverse(root.right, p, q);
        if (lca != null) { // 提前退出
            return 2;
        }
        int rootContains = 0;
        if (root==p || root==q) {
            rootContains = 1;
        }
        if (rootContains == 0 && leftContains == 1 && rightContains == 1) {
            lca = root;
        }
        if (rootContains == 1 && (leftContains == 1 || rightContains == 1)) {
            lca = root;
        }
        return leftContains + rightContains + rootContains;
    }
}
```

习题课：二叉树

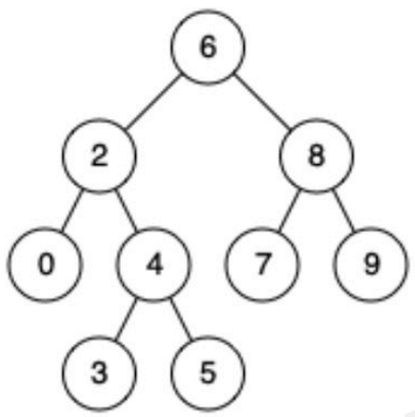
题型5：LCA最近公共祖先

剑指 Offer 68 - I. 二叉搜索树的最近公共祖先（中等）

给定一个二叉搜索树, 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**）。”

例如，给定如下二叉搜索树：root = [6,2,8,0,4,7,9,null,null,3,5]



示例 1:

```
输入: root = [6,2,8,0,4,7,9,null,null,3,5], p = 2, q = 8
输出: 6
解释: 节点 2 和节点 8 的最近公共祖先是 6。
```

题型5：LCA最近公共祖先

剑指 Offer 68 - I. 二叉搜索树的最近公共祖先（中等）**非递归实现**

```
class Solution {
    public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
        TreeNode x = root;
        while (true) {
            if (p.val < x.val && q.val < x.val) {
                x = x.left;
            } else if (p.val > x.val && q.val > x.val) {
                x = x.right;
            } else { //包含各种情况
                return x;
            }
        }
    }
}
```

题型5：LCA最近公共祖先

剑指 Offer 68 - I. 二叉搜索树的最近公共祖先（中等） **递归实现**

```
class Solution {
    public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
        if (p == root || q == root || (p.val < root.val && root.val < q.val)
            || (q.val < root.val && root.val < p.val)) {
                return root;
        }
        if (p.val < root.val && q.val < root.val) {
            return lowestCommonAncestor(root.left, p, q);
        } else {
            return lowestCommonAncestor(root.right, p, q);
        }
    }
}
```

习题课：二叉树

题型6：二叉树转单、双、循环链表

114. 二叉树展开为链表（中等）

面试题 17.12. BiNode（中等）

剑指 Offer 36. 二叉搜索树与双向链表（中等）

面试题 04.03. 特定深度节点链表（中等）

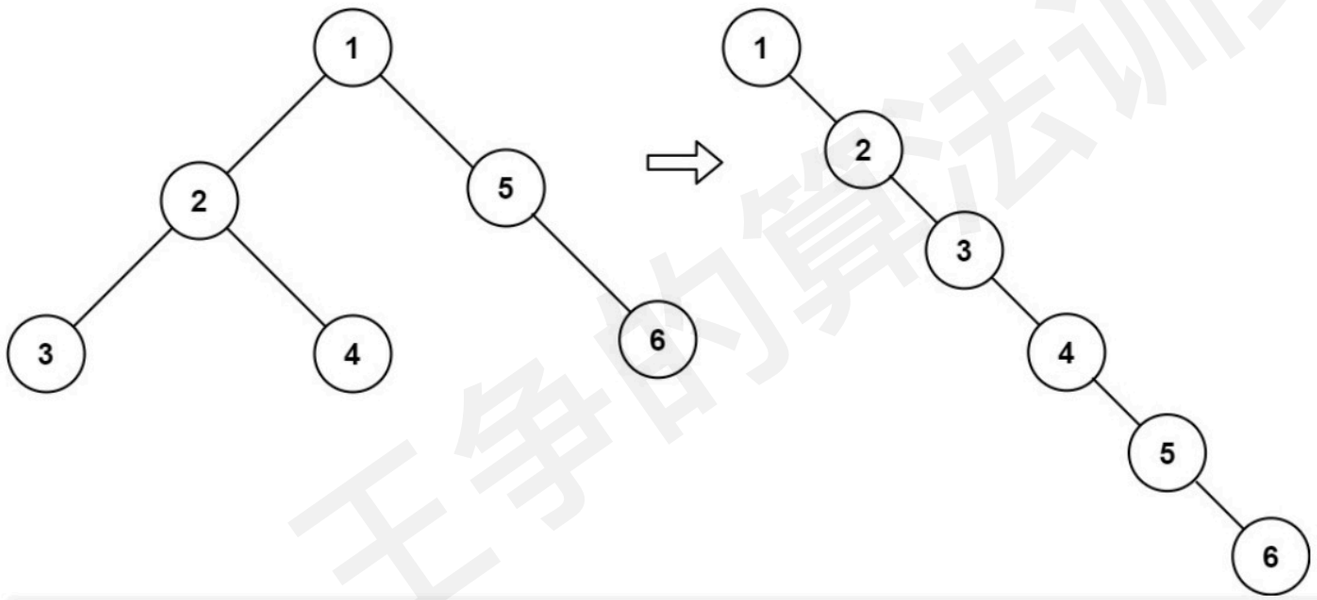
习题课：二叉树

题型6：二叉树转单、双、循环链表 114. 二叉树展开为链表（中等）

给你二叉树的根结点 `root` ，请你将它展开为一个单链表：

- 展开后的单链表应该同样使用 `TreeNode` ，其中 `right` 子指针指向链表中下一个结点，而左子指针始终为 `null` 。
- 展开后的单链表应该与二叉树 先序遍历 顺序相同。中序遍历，后序遍历

示例 1：




```
输入：root = [1,2,5,3,4,null,6]
输出：[1,null,2,null,3,null,4,null,5,null,6]
```

```java
class Solution {
    private TreeNode dummyHead = new TreeNode();
    private TreeNode tail = dummyHead;

    public void flatten(TreeNode root) {
      preorder(root);
    }

    private void preorder(TreeNode root) {
      if (root == null) return;

      TreeNode left = root.left;
      TreeNode right = root.right;
      // 把遍历到的节点放到结果链表中
      tail.right = root;
      tail = root;
      root.left = null;

      // 左子树
      preorder(left);
      // 右子树
      preorder(right);
    }
}
```

习题课：二叉树

题型6：二叉树转单、双、循环链表 面试题 17.12. BiNode（中等）

二叉树数据结构 `TreeNode` 可用来表示单向链表（其中 `left` 置空， `right` 为下一个链表节点）。实现一个方法，把二叉搜索树转换为单向链表，要求依然符合二叉搜索树的性质，转换操作应是原址的，也就是在原始的二叉搜索树上直接修改。

返回转换后的单向链表的头节点。

注意：本题相对原题稍作改动

示例：

```
输入：  [4,2,5,1,3,null,6,0]
输出：  [0,null,1,null,2,null,3,null,4,null,5,null,6]
```

提示：

- 节点数量不会超过 100000。

习题课：二叉树

王争的算法训练营

```
class Solution {
    private TreeNode dummyHead = new TreeNode();
    private TreeNode tail = dummyHead;

    public TreeNode convertBiNode(TreeNode root) {
      inorder(root);
      return dummyHead.right;
    }

    private void inorder(TreeNode root) {
      if (root == null) return;

      // 左子树
      inorder(root.left);
      // 把遍历到的节点放到结果链表中
      tail.right = root;
      tail = root;
      root.left = null;
      // 右子树
      inorder(root.right);
    }
}
```

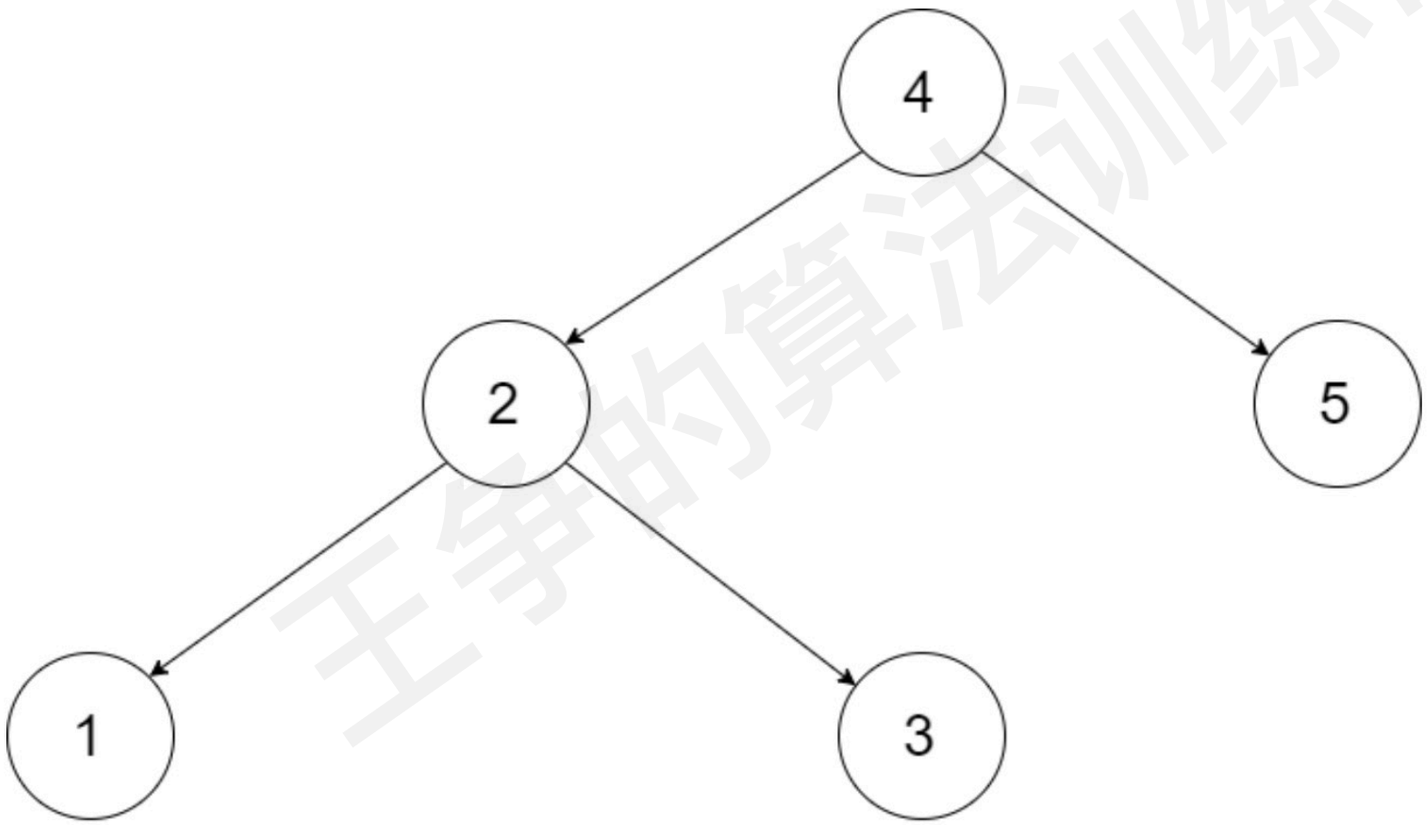
习题课：二叉树

王争的算法训练营

题型6：二叉树转单、双、循环链表 剑指 Offer 36. 二叉搜索树与双向链表（中等）

输入一棵二叉搜索树，将该二叉搜索树转换成一个排序的循环双向链表。要求不能创建任何新的节点，只能调整树中节点指针的指向。

为了让您更好地理解问题，以下面的二叉搜索树为例：



我们希望将这个二叉搜索树转化为双向循环链表。链表中的每个节点都有一个前驱和后继指针。对于双向循环链表，第一个节点的前驱是最后一个节点，最后一个节点的后继是第一个节点。

下图展示了上面的二叉搜索树转化成的链表。“head” 表示指向链表中有最小元素的节点。

```
class Solution {
    private Node dummyHead = new Node();
    private Node tail = dummyHead;

    public Node treeToDoublyList(Node root) {
      if (root == null) return null;
      inorder(root);
      tail.right = dummyHead.right;
      dummyHead.right.left = tail;
      return dummyHead.right;
    }

    private void inorder(Node root) {
      if (root == null) return;

      // 左子树
      inorder(root.left);
      // 把遍历到的节点放到结果链表中
      root.left = tail;
      tail.right = root;
      tail = root;
      // 右子树
      inorder(root.right);
    }
}
```

习题课：二叉树

题型6：二叉树转单、双、循环链表 面试题 04.03. 特定深度节点链表（中等）

给定一棵二叉树，设计一个算法，创建含有某一深度上所有节点的链表（比如，若一棵树的深度为 `D`，则会创建出 `D` 个链表）。返回一个包含所有深度的链表的数组。

示例：

输入： `[1,2,3,4,5,null,7,8]`

输出： `[[1],[2,3],[4,5,7],[8]]`

```java
class Solution {
    public ListNode[] listOfDepth(TreeNode tree) {
      if (tree == null) return new ListNode[0];
      List<ListNode> result = new ArrayList<ListNode>();
      Queue<TreeNode> queue = new LinkedList<TreeNode>();
      queue.add(tree);
      while (!queue.isEmpty()) {
        ListNode dummyHead = new ListNode();//虚拟头节点
        ListNode tail = dummyHead;
        int curLevelNum = queue.size();
        for (int i = 0; i < curLevelNum; ++i) {
          TreeNode treeNode = queue.poll();
          tail.next = new ListNode(treeNode.val);
          tail = tail.next;
          if (treeNode.left != null) {
            queue.add(treeNode.left);
          }
          if (treeNode.right != null) {
            queue.add(treeNode.right);
          }
        }
        result.add(dummyHead.next);
      }
      ListNode[] resultArr = new ListNode[result.size()];
      for (int i = 0; i < result.size(); ++i) {
          resultArr[i] = result.get(i);
      }
      return resultArr;
    }
}
```

习题课：二叉树

题型7：按照遍历结果反向构建二叉树

105. 从前序与中序遍历序列构造二叉树（中等）

106. 从中序与后序遍历序列构造二叉树（中等）

889. 根据前序和后序遍历构造二叉树（中等）

剑指 Offer 33. 二叉搜索树的后序遍历序列（中等）

习题课：二叉树

题型7：按照遍历结果反向构建二叉树 105. 从前序与中序遍历序列构造二叉树（中等）

根据一棵树的前序遍历与中序遍历构造二叉树。

注意:
你可以假设树中没有重复的元素。

例如，给出

```
前序遍历 preorder = [3,9,20,15,7]
中序遍历 inorder = [9,3,15,20,7]
```

返回如下的二叉树：

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

```java
class Solution {
    public TreeNode buildTree(int[] preorder, int[] inorder) {
        return myBuildTree(preorder, 0, preorder.length-1, inorder, 0, inorder.length-1);
    }

    // preorder下标i,j；inorder下标p,q
    private TreeNode myBuildTree(int[] preorder, int i, int j, int[] inorder, int p, int r) {
        if (i>j) return null;

        TreeNode root = new TreeNode(preorder[i]);
        // 在中序遍历结果inorder中，查询preorder[i]所在的位置[p, q-1] q [q+1, r]
        int q = p;
        while (inorder[q] != preorder[i]) {
            q++;
        }
        int leftTreeSize = q-p; //左右子树大小

        // 构建左子树
        TreeNode leftNode = myBuildTree(preorder, i+1, i+leftTreeSize, inorder, p, q-1);
        // 构建右子树
        TreeNode rightNode = myBuildTree(preorder, i+leftTreeSize+1, j, inorder, q+1, r);
        // 根据root、左子树、右子树构建树
        root.left = leftNode;
        root.right = rightNode;
        return root;
    }
}
```

习题课：二叉树

题型7：按照遍历结果反向构建二叉树 106. 从中序与后序遍历序列构造二叉树（中等）

根据一棵树的中序遍历与后序遍历构造二叉树。

注意:
你可以假设树中没有重复的元素。

例如，给出

```
中序遍历 inorder = [9,3,15,20,7]
后序遍历 postorder = [9,15,7,20,3]
```

返回如下的二叉树：

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

```java
class Solution {
    public TreeNode buildTree(int[] inorder, int[] postorder) {
        return myBuildTree(postorder, inorder, 0, postorder.length-1, 0, inorder.length-1);
    }

    // postorder下标i,j；inorder下标p,q
    private TreeNode myBuildTree(int[] postorder, int i, int j, int[] inorder, int p, int r) {
        if (i>j) return null;

        TreeNode root = new TreeNode(postorder[j]);
        // 在中序遍历结果inorder中，查询postorder[j]所在的位置q
        // [p, q-1] q [q+1, r]
        int q = p;
        while (inorder[q] != postorder[j]) {
            q++;
        }
        //左右子树大小
        int leftTreeSize = q-p;

        // 构建左子树
        TreeNode leftNode = myBuildTree(postorder, i, i+leftTreeSize-1, inorder, p, q-1);
        // 构建右子树
        TreeNode rightNode = myBuildTree(postorder, i+leftTreeSize, j-1, inorder, q+1, r);
        root.left = leftNode;
        root.right = rightNode;
        return root;
    }
}
```

习题课：二叉树

题型7：按照遍历结果反向构建二叉树 889. 根据前序和后序遍历构造二叉树（中等）

返回与给定的前序和后序遍历匹配的任何二叉树。

`pre` 和 `post` 遍历中的值是不同的正整数。

示例：

```
输入：pre = [1,2,4,5,3,6,7], post = [4,5,2,6,7,3,1]
输出：[1,2,3,4,5,6,7]
```

提示：

- `1 <= pre.length == post.length <= 30`
- `pre[]` 和 `post[]` 都是 `1, 2, ..., pre.length` 的排列
- 每个输入保证至少有一个答案。如果有多个答案，可以返回其中一个。

```java
class Solution {
    public TreeNode constructFromPrePost(int[] pre, int[] post) {
        return myBuildTree(pre, post, 0, pre.length-1, 0, post.length-1);
    }

    // pre下标i,j; post下标p,r
    private TreeNode myBuildTree(int[] pre, int i, int j, int[] post, int p, int r) {
        if (i>j) return null;

        TreeNode root = new TreeNode(pre[i]);
        if (i == j) return root;//注意这一行跟前面几题不一样

        // 在post中, 查询pre[i+1]所在的位置q, [p, q] [q+1, r-1] r(root)
        int q = p;
        while (post[q] != pre[i+1]) {
            q++;
        }
        //左子树大小
        int leftTreeSize = q-p+1;

        // 构建左子树
        TreeNode leftNode = myBuildTree(pre, i+1, i+leftTreeSize, post, p, q);
        // 构建右子树
        TreeNode rightNode = myBuildTree(pre, i+leftTreeSize+1, j, post, q+1, r-1);
        root.left = leftNode;
        root.right = rightNode;
        return root;
    }
}
```

习题课：二叉树

王争的算法训练营

题型7：按照遍历结果反向构建二叉树 剑指 Offer 33. 二叉搜索树的后序遍历序列（中等）

输入一个整数数组，判断该数组是不是某二叉搜索树的后序遍历结果。如果是则返回 `true`，否则返回 `false`。假设输入的数组的任意两个数字都互不相同。

参考以下这颗二叉搜索树：

```
     5
    / \
   2   6
  / \
 1   3
```

左子树是BST
右子树是BST
root节点值都大于左子树节点值，小于右子树节点值
=》
二叉树是BST

示例 1：

```
输入: [1,6,3,2,5]
输出: false
```

示例 2：

```
输入: [1,3,2,6,5]
输出: true
```

```java
class Solution {
    public boolean verifyPostorder(int[] postorder) {
        return myVerify(postorder, 0, postorder.length-1);
    }

    private boolean myVerify(int[] postorder, int i, int j) {
        if (i >= j) return true;

        // postorder[j]是根节点,先分离出左子树[i, k-1]
        int k = i;
        while (k < j && postorder[k] < postorder[j]) {
            k++;
        }
        // 验证[k, j-1]满足有子树的要求，都大于postorder[j]
        int p = k;
        while (p < j) {
            if (postorder[p] < postorder[j]) {
                return false;
            }
            p++;
        }

        // 递归验证左右子树是否满足BST的要求
        boolean leftValid = myVerify(postorder, i, k-1);
        if (leftValid == false) return false;
        boolean rightValid = myVerify(postorder, k, j-1);
        return rightValid;
    }
}
```

习题课：二叉树

题型8：二叉树上的最长路径和

543. 二叉树的直径（简单）

剑指 Offer 34. 二叉树中和为某一值的路径（中等）

124. 二叉树中的最大路径和 （困难）

437. 路径总和 III （困难）

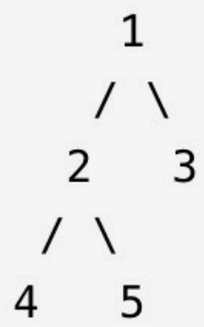
习题课：二叉树

王争的算法训练营

题型8：二叉树上的最长路径和 543. 二叉树的直径（简单）

给定一棵二叉树，你需要计算它的直径长度。一棵二叉树的直径长度是任意两个结点路径长度中的最大值。这条路径可能穿过也可能不穿过根结点。

示例：
给定二叉树



努力转化为用递归来实现

- **子问题：左子树的直径**
- **子问题：右子树的直径**
- **问题：二叉树的直径?**

子问题的解无法推导出问题的解

将问题转化成：求树的最大高度

返回 **3**, 它的长度是路径 [4,2,1,3] 或者 [5,2,1,3]。

注意：两结点之间的路径长度是以它们之间边的数目表示。

习题课：二叉树

王争的算法训练营

题型8：二叉树上的最长路径和 543. 二叉树的直径（简单）

```
class Solution {
    private int result = 0;

    public int diameterOfBinaryTree(TreeNode root) {
        calMaxHeight(root);
        return result;
    }

    public int calMaxHeight(TreeNode root) {
        if (root == null) return 0;
        int maxLeftHeight = calMaxHeight(root.left);
        int maxRightHeight = calMaxHeight(root.right);
        int diameter = maxLeftHeight + maxRightHeight;
        if (diameter > result) result = diameter;
        return Math.max(maxLeftHeight, maxRightHeight) + 1;
    }
}
```

习题课：二叉树

王争的算法训练营

题型8：二叉树上的最长路径和 124. 二叉树中的最大路径和 （困难）

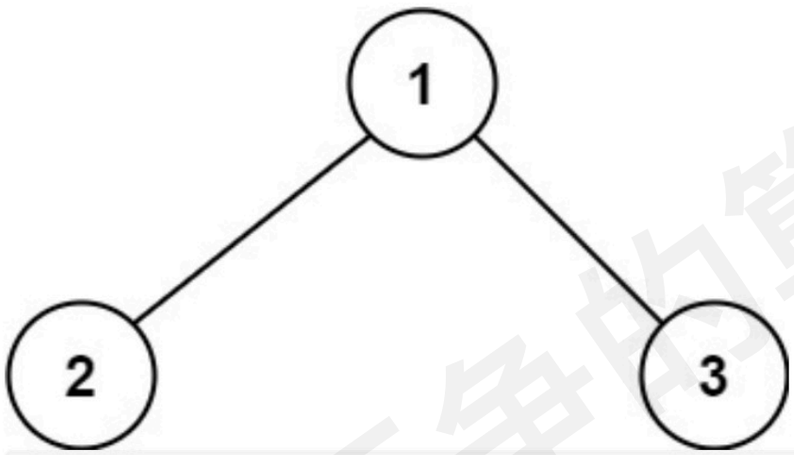
路径 被定义为一条从树中任意节点出发，沿父节点-子节点连接，达到任意节点的序列。同一个节点在一条路径序列中 **至多出现一次** 。该路径 **至少包含一个** 节点，且不一定经过根节点。

路径和 是路径中各节点值的总和。

给你一个二叉树的根节点 `root` ，返回其 **最大路径和** 。

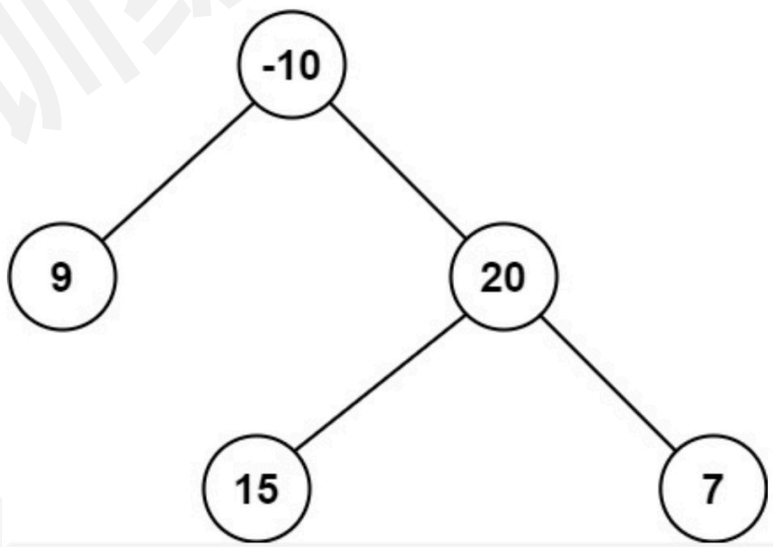
- **转化为：以任意节点为转折点的最大路径和**
- **怎么计算某个节点为转折点的最大路径和呢?**

示例 1：



输入： root = [1,2,3]
输出： 6
解释： 最优路径是 2 -> 1 -> 3 ，路径和为 2 + 1 + 3 = 6

示例 2：



输入： root = [-10,9,20,null,null,15,7]
输出： 42
解释： 最优路径是 15 -> 20 -> 7 ，路径和为 15 + 20 + 7 = 42

提示：

- 树中节点数目范围是 $[1, 3 * 10^4]$
- $-1000 <= Node.val <= 1000$

```java
class Solution {
    private int result = -1001;

    public int maxPathSum(TreeNode root) {
        dfs(root);
        return result;
    }

    public int dfs(TreeNode root) {
        if (root == null) return 0;

        int leftMaxPath = dfs(root.left);
        int rightMaxPath = dfs(root.right);

        int max = root.val;
        if (leftMaxPath > 0) max += leftMaxPath;
        if (rightMaxPath > 0) max += rightMaxPath;
        if (max > result) result = max;

        int ret = root.val;
        if (ret < leftMaxPath+root.val) ret = leftMaxPath+root.val;
        if (ret < rightMaxPath+root.val) ret = rightMaxPath+root.val;
        return ret;
    }
}
```

习题课：二叉树

王争的算法训练营

题型8：二叉树上的最长路径和 剑指 Offer 34. 二叉树中和为某一值的路径（中等）

输入一棵二叉树和一个整数，打印出二叉树中节点值的和为输入整数的所有路径。从树的根节点开始往下一直到叶节点所经过的节点形成一条路径。

- 穷举每条路径
- 穷举->回溯
- 图上的回溯->DFS

示例:
给定如下二叉树，以及目标和 `target = 22`，

```
              5
             / \
            4   8
           /   / \
          11  13  4
         /  \    / \
        7    2  5   1
```

返回:

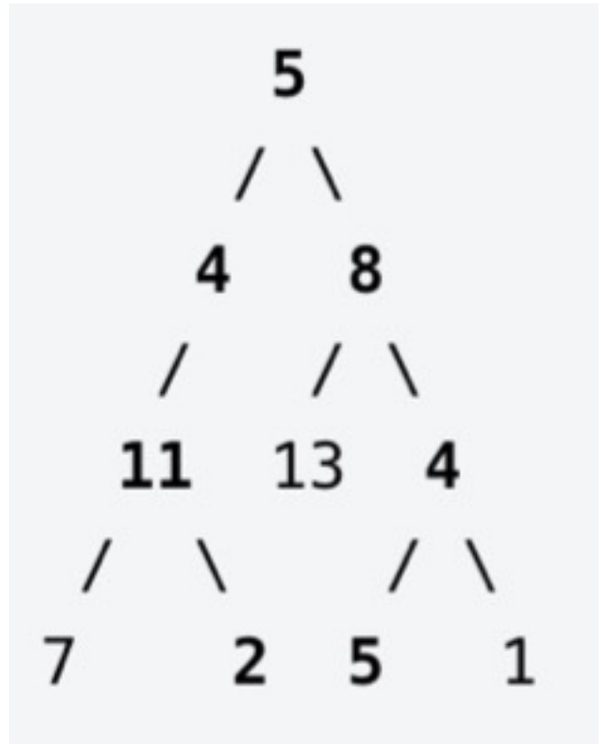
```
[
   [5,4,11,2],
   [5,8,4,5]
]
```

提示：

1. 节点总数 <= 10000

```java
class Solution {
    private List<List<Integer>> result = new ArrayList<>();
    public List<List<Integer>> pathSum(TreeNode root, int sum) {
        if (root == null) return result;
        dfs(root, sum, new ArrayList<>(), 0);
        return result;
    }

    private void dfs(TreeNode root, int sum, List<Integer> path, int pathSum) {
        path.add(root.val);
        pathSum += root.val;
        if (root.left == null && root.right == null) {
            if (pathSum == sum) {
                List<Integer> pathSnapshot = new ArrayList<>();
                pathSnapshot.addAll(path);
                result.add(pathSnapshot);
            }
            path.remove(path.size()-1);
            return;
        }
        if (root.left != null) {
            dfs(root.left, sum, path, pathSum);
        }
        if (root.right != null) {
            dfs(root.right, sum, path, pathSum);
        }
        path.remove(path.size()-1);
        // pathSum == root.val 不需要这一句
    }
}
```



习题课：二叉树

王争的算法训练营

题型8：二叉树上的最长路径和 437. 路径总和 III （困难）

给定一个二叉树，它的每个结点都存放着一个整数值。

找出路径和等于给定数值的路径总数。

路径不需要从根节点开始，也不需要在叶子节点结束，但是路径方向必须是向下的（只能从父节点到子节点）。

二叉树不超过1000个节点，且节点数值范围是 [-1000000,1000000] 的整数。

示例：

```
root = [10,5,-3,3,2,null,11,3,-2,null,1], sum = 8

      10
     /  \
    5   -3
   / \    \
  3   2   11
 / \   \
3  -2   1

返回 3。和等于 8 的路径有:

1.  5 -> 3
2.  5 -> 2 -> 1
3.  -3 -> 11
```

```java
class Solution {
    private int count = 0;
    public int pathSum(TreeNode root, int sum) {
        dfs(root, sum);
        return count;
    }

    private Map<Integer, Integer> dfs(TreeNode root, int sum) {
        if (root == null) return new HashMap<>();
        Map<Integer, Integer> leftValues = dfs(root.left, sum);
        Map<Integer, Integer> rightValues = dfs(root.right, sum);
        Map<Integer, Integer> rootValues = new HashMap<>();
        rootValues.put(root.val, 1);
        for (Map.Entry<Integer, Integer> entry : leftValues.entrySet()) {
            int newKey = entry.getKey()+root.val;
            int newValue = entry.getValue();
            if (rootValues.containsKey(newKey)) {
                newValue += rootValues.get(newKey);
            }
            rootValues.put(newKey, newValue);
        }
        for (Map.Entry<Integer, Integer> entry : rightValues.entrySet()) {
            int newKey = entry.getKey() + root.val;
            int newValue = entry.getValue();
            if (rootValues.containsKey(newKey)) {
                newValue += rootValues.get(newKey);
            }
            rootValues.put(newKey, newValue);
        }
        for (Map.Entry<Integer, Integer> entry : rootValues.entrySet()) {
            if (entry.getKey() == sum) {
                count += entry.getValue();
            }
        }
        return rootValues;
    }
}
```

提问环节

关注微信公众号“**小争哥**”，

后台回复“**PDF**”获取独家算法资料