九章算法面试大总结之二：Java搞定面试中的二叉树题

About the Tree: full binary tree: A binary tree in which each node has exactly zero or two children. Perfect binary tree: A binary tree with all leaf nodes at the same depth. All internal nodes have degree 2 [1]

满二叉树：每个节点都有0或是2个孩子。

完美二叉树：所有的叶子都拥有同的深度，所有的内部节点拥有 2个孩子

The difference between Full Binary Tree & Complete Binary Tree: (1). a binary tree T is full if each node is either a leaf or possesses exactly two child nodes. (2). a binary tree T with n levels is complete if all levels except possibly the last are completely full, and the last level has all its nodes to the left side. [2]

满二叉树和完全二叉树的区别：

满二叉树是每个节点要不是叶子要不拥有 2个孩子

完全二叉树：每一层都是完全的，除了最后一层，而最后一层所有的节点都在左边。

AVL Trees: AVL trees are self-balancing binary search trees. These trees are named after their two inventors G.M. Adel’son-Vel’skii and E.M. Landis. [3]

The height/depth of a tree: The height of a node is the length of the longest downward path to a leaf from that node. The height of the root is the height of the tree.

The depth of a node is the length of the path to its root (i.e., its root path).

This is commonly needed in the manipulation of the various self-balancing trees, AVL Trees in particular. The root node has depth zero, leaf nodes have height zero, and a tree with only a single node (hence both a root and leaf) has depth and height zero. Conventionally, an empty tree (tree with no nodes, if such are allowed) has depth and height −1.[4]

根的深度为0，叶子高度为0.只有一个节点的树的depth height 都为0. 按照惯例，一个空树的depth, height 都是-1

[GitHub代码链接-Tree大总结](https://github.com/yuzhangcmu/LeetCode_algorithm/blob/master/tree/TreeDemo.java)  
https://github.com/yuzhangcmu/LeetCode\_algorithm/blob/master/tree/TreeDemo.java  
   
1. 求二叉树中的节点个数:  
        getNodeNumRec（递归），getNodeNum（迭代）  
2. 求二叉树的深度:  
        getDepthRec（递归），getDepth   
3. 前序遍历，中序遍历，后序遍历:  
        preorderTraversalRec, preorderTraversal, inorderTraversalRec, postorderTraversalRec  
4. 分层遍历二叉树（按层次从上往下，从左往右）:  
        levelTraversal, levelTraversalRec（递归解法）  
5. 将二叉查找树变为有序的双向链表:  
        convertBST2DLLRec, convertBST2DLL  
6. 求二叉树第K层的节点个数：  
        getNodeNumKthLevelRec, getNodeNumKthLevel  
7. 求二叉树中叶子节点的个数：  
        getNodeNumLeafRec, getNodeNumLeaf  
8. 判断两棵二叉树是否相同的树：  
        isSameRec, isSame  
9. 判断二叉树是不是平衡二叉树：isAVLRec  
10. 求二叉树的镜像（破坏和不破坏原来的树两种情况）：  
     mirrorRec, mirrorCopyRec  
     mirror, mirrorCopy  
10.1 判断两个树是否互相镜像：isMirrorRec isMirror  
11. 求二叉树中两个节点的最低公共祖先节点：  
         LAC        求解最小公共祖先, 使用list来存储path.  
         LCABstRec  递归求解BST树.  
         LCARec     递归算法 .  
12. 求二叉树中节点的最大距离：  
         getMaxDistanceRec  
13. 由前序遍历序列和中序遍历序列重建二叉树：  
         rebuildBinaryTreeRec  
14. 判断二叉树是不是完全二叉树：  
         isCompleteBinaryTree, isCompleteBinaryTreeRec  
15. 找出二叉树中最长连续子串(即全部往左的连续节点，或是全部往右的连续节点）  
         findLongest