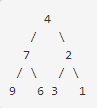
# 226. Invert Binary Tree

Invert a binary tree. (难度：easy)

Example:

**Input: Output：**

递归算法实现：

/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode(int x) { val = x; }

\* }

\*/

class Solution {

public TreeNode invertTree(TreeNode root) {

if(root == null){

return root;

}

swap(root);

return root;

}

private void swap(TreeNode root){

if(root == null || root.left == null&& root.right == null){

return;

}

*TreeNode temp = root.left;*

*root.left = root.right;*

*root.right = temp;*

**swap(root.left);**

**swap(root.right);**

}

}

# 222. Count Complete Tree Nodes

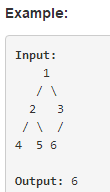
计算完全二叉树的节点数目(不懂)？？？？？

Given a complete binary tree, count the number of nodes.

Note:

Definition of a complete binary tree from **Wikipedia**:

In a complete binary tree every level, except possibly the last, is completely filled, and all nodes in the last level are as far left as possible. It can have between 1 and 2h nodes inclusive at the last level h.



算法：利用递归遍历和非递归遍历方法，都出现超时的提醒。

去掉那个-100的判断也是超时提醒。不知道-100的判断是什么作用，设置-10的判断也可以。

/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode(int x) { val = x; }

\* }

\*/

public class Solution {

public int countNodes(TreeNode root) {

if(root==null){

return 0;

}

Queue<TreeNode> q = new LinkedList<TreeNode>();

q.add(root);

int count=1;

while(!q.isEmpty()){

TreeNode temp = q.poll();

**if(temp.val!=-100){//不懂，为什么加上这句话，就不超时了？？？**

**temp.val=-100;** **//不懂为甚么加上这个就不会超时呢？？？？？？难道temp从q出来，还会再次进去？？？？？？**

if(temp.left!=null){

q.offer(temp.left);

count++;

}

if(temp.right!=null){

q.offer(temp.right);

count++;

}

}

}

return count;

}

}