1. Count Complete Tree Nodes

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

**Note:**

**Definition of a complete binary tree from** [**Wikipedia**](http://en.wikipedia.org/wiki/Binary_tree#Types_of_binary_trees)**:** 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.

**Example:**

Input:   
 1  
 / \  
 2 3  
 / \ /  
4 5 6  
  
Output: 6

**解法1** 一般的计算节点数目方法，count(root) = 1 + count(root->left) + count(root->left)

class Solution {  
public:  
 int countNodes(TreeNode\* root) {  
 if(root == NULL)return 0;  
 return 1 + countNodes(root->left) + countNodes(root->right);  
 }  
}

**解法2** 先序/中序/后序遍历一次

class Solution {  
public:  
 int countNodes(TreeNode\* root) {  
 int ans = 0;  
 pre(root, ans);  
 return ans;  
 }  
 void pre(TreeNode \*root, int &ans){  
 if(root == NULL)return;  
 ans++;  
 pre(root->left, ans);  
 pre(root->right, ans);  
 }  
};

**解法3** 考虑到完全二叉树的特点：对于每一个节点来说，总有一边是满二叉树，高度为满二叉树共有个节点

class Solution {  
public:  
 int countNodes(TreeNode\* root) {  
 int l\_h = get\_left\_h(root);  
 int r\_h = get\_right\_h(root);  
 if(l\_h == r\_h)return pow(2, l\_h) - 1;  
 return 1 + countNodes(root->left) + countNodes(root->right);  
 }  
 int get\_left\_h(TreeNode \*root){  
 int d = 0;  
 while(root){  
 d++;  
 root = root->left;  
 }  
 return d;  
 }  
 int get\_right\_h(TreeNode \*root){  
 int d = 0;  
 while(root){  
 d++;  
 root = root->right;  
 }  
 return d;  
 }  
};

**解法4** 考虑完全二叉树的特点：只有最后一层是不满的，采用二分查找确定是从哪里分割的，即寻找第一个没有出现在叶子节点中的节点编号

* 如何判断一个叶子节点是否存在
* 将叶子节点编号为，根节点将叶子节点分成了和两部分，查找编号为idx的叶子时：
  + idx <= mid, root = root->left
  + idx > mid, root = root->right
* 如何查找第一个不存在的叶子结点的编号
* 假设第一个不存在的叶子结点在区间中，判断中点
  + exist(mid) == true ：中点及左侧区间被排除
  + exist(mid) == false : 右侧区间被排除

class Solution {  
public:  
 int countNodes(TreeNode\* root) {  
 int l\_h = get\_left\_h(root);  
 int r\_h = get\_right\_h(root);  
 if(l\_h == r\_h)return pow(2, r\_h) - 1;  
 int l = 0, r = pow(2, r\_h) - 1;  
 while(l < r){  
 int mid = (l + r) / 2;  
 if(exist(mid, r\_h, root)){  
 l = mid + 1;  
 }else{  
 r = mid;  
 }  
 }  
 return pow(2, r\_h) + l-1;  
 }  
 bool exist(int val, int d, TreeNode \*root){  
 int l = 0, r = pow(2, d)-1;  
 TreeNode \*cur = root;  
 for(int i = 0; i < d; ++i){  
 int mid = (l + r) / 2;  
 if(val <= mid){  
 r = mid;  
 cur = cur->left;  
 }else{  
 l = mid + 1;  
 cur = cur->right;  
 }  
 }  
 return cur != NULL;  
 }  
 int get\_left\_h(TreeNode \*root){  
 int d = 0;  
 while(root){  
 d++;  
 root = root->left;  
 }  
 return d;  
 }  
 int get\_right\_h(TreeNode \*root){  
 int d = 0;  
 while(root){  
 d++;  
 root = root->right;  
 }  
 return d;  
 }  
};