BITTIGER

CS102 Top100高频算法设计课

第三节 二叉树

左程云

版权声明

BITTIGER

Copyright Policy

All content included on the Site or third-party platforms as part of the class, such as text, graphics, logos, button icons, images, audio clips, video clips, live streams, digital downloads, data compilations, and software, is the property of BitTiger or its content suppliers and protected by copyright laws.

Any attempt to redistribute or resell BitTiger content will result in the appropriate legal action being taken.

We thank you in advance for respecting our copyrighted content.For more info see https://www.bittiger.io/termsofuse and https://www.bittiger.io/termsofservice

预备内容 用递归和非递归实现二叉树先、中、后序遍历

【题目一】
用递归和非递归方式，
分别按照二叉树先序、中序和后序打印所有的节点。
我们约定：
先序遍历顺序为根、左、右
中序遍历顺序为左、根、右
后序遍历顺序为左、右、根

预备内容　用递归和非递归实现二叉树先、中、后序遍历

【解题点】
用非递归的方式实现二叉树的先序遍历，具体过程如下：
1．申请一个新的栈，记为stack。然后将头节点head压入stack中。
2．从stack中弹出栈顶节点，记为cur，然后打印cur节点的值，再将节点cur的右孩子（不为空的话）先压入stack中，最后将cur的左孩子（不为空的话）压入stack中。
3．不断重复步骤2，直到stack为空，全部过程结束。

预备内容　用递归和非递归实现二叉树先、中、后序遍历

【解题点】
用非递归的方式实现二叉树的中序遍历，具体过程如下：
1．申请一个新的栈，记为stack。初始时，令变量cur=head。
2．先把cur节点压入栈中，对以cur节点为头的整棵子树来说，依次把左边界压入栈中，即不停地令cur=cur.left，然后重复步骤2。
3．不断重复步骤2，直到发现cur为空，此时从stack中弹出一个节点，记为node。打印node的值，并且让cur=node.right，然后继续重复步骤2。
4．当stack为空且cur为空时，整个过程停止。

预备内容　用递归和非递归实现二叉树先、中、后序遍历

【解题点】

先介绍用两个栈实现后序遍历的过程，具体过程如下：

1．申请一个栈，记为s1，然后将头节点head压入s1中。

2．从s1中弹出的节点记为cur，然后依次将cur的左孩子和右孩子压入s1中。

3．在整个过程中，每一个从s1中弹出的节点都放进s2中。

4．不断重复步骤2和步骤3，直到s1为空，过程停止。

5．从s2中依次弹出节点并打印，打印的顺序就是后序遍历的顺序。

预备内容　用递归和非递归实现二叉树先、中、后序遍历

【解题点】
介绍只用一个栈实现后序遍历的过程，具体过程如下：
1．申请一个栈，记为stack，将头节点压入stack，同时设置两个变量h和c。在整个流程中，h代表最近一次弹出并打印的节点，c代表stack的栈顶节点，初始时h为头节点，c为null。
2．每次令c等于当前stack的栈顶节点，但是不从stack中弹出，此时分以下三种情况。
① 如果c的左孩子不为null，并且h不等于c的左孩子，也不等于c的右孩子，则把c的左孩子压入stack中。
② 如果条件①不成立，并且c的右孩子不为null，h不等于c的右孩子，则把c的右孩子压入stack中。含义是如果h等于c的右孩子，说明c的右子树已经打印完毕，此时不应该再将c的右孩子放入stack中。否则，说明右子树还没处理过，此时将c的右孩子压入stack中。
③ 如果条件①和条件②都不成立，说明c的左子树和右子树都已经打印完毕，那么从stack中弹出c并打印，然后令h=c。
3．一直重复步骤2，直到stack为空，过程停止。

预备内容　如何较为直观地打印二叉树

【题目二】
二叉树可以用常规的三种遍历结果来描述其结构，但是不够直观，尤其是二叉树中有重复值的时候，仅通过三种遍历的结果来构造二叉树的真实结构更是难上加难，有时则根本不可能。给定一棵二叉树的头节点head，已知二叉树节点值的类型为32位整型，请实现一个打印二叉树的函数，可以直观地展示树的形状，也便于画出真实的结构。

预备内容　如何较为直观地打印二叉树

【解题点】
1，自己看code，写出更好的告诉我
2，便于同学们刷二叉树题目后验证树调整之后的样子

二叉树的序列化和反序列化

【题目三】
二叉树被记录成文件的过程叫作二叉树的序列化，通过文件内容重建原来二叉树的过程叫作二叉树的反序列化。给定一棵二叉树的头节点head，并已知二叉树节点值的类型为32位整型。请设计一种二叉树序列化和反序列化的方案，并用代码实现。

二叉树的序列化和反序列化

【解题点】
方法一：按照先序遍历的方式序列化
1，先序遍历二叉树，递归方法即可
2，使用特殊字符表示一个值的结束
3，使用另一个特殊字符表示空节点
4，怎么用递归实现的序列化的就怎么反序列化

方法二：按照层遍历的方式序列化
1，层遍历用队列实现
2，使用特殊字符表示一个值的结束
3，使用另一个特殊字符表示空节点
4，怎么用迭代实现的序列化的就怎么反序列化

遍历二叉树的神级方法 - - Morris遍历

【题目四】
给定一棵二叉树的头节点head，完成二叉树的先序、中序和后序遍历。
如果二叉树的节点数为*N*，
要求：时间复杂度为*O*(*N*)，额外空间复杂度为*O*(1)。

遍历二叉树的神级方法 - - Morris遍历

【解题点】

Morris实现二叉树的先序打印

1，当前节点为h，初始时h为头节点，

2，如果h为空时，返回，结束。否则继续

3，如果h有左子树并且左子树上最右节点的right指针不指向h时，让其
指向h，打印h节点的值，然后让当前节点变为h.left
即令h=h.left

4，如果h没有左子树或者左子树上最右节点的right指针指向h时，让其
指向null，再让当前节点变为h.right
即令h=h.right

5，回到步骤2重复执行

遍历二叉树的神级方法 - - Morris遍历

【解题点】

Morris实现二叉树的中序打印

1，当前节点为h，初始时h为头节点，

2，如果h为空时，返回，结束。否则继续

3，如果h有左子树并且左子树上最右节点的right指针不指向h时，让其
指向h，然后让当前节点变为h.left
即令h=h.left

4，如果h没有左子树或者左子树上最右节点的right指针指向h时，让其
指向null，打印h节点的值，再让当前节点变为h.right
即令h=h.right

5，回到步骤2重复执行

遍历二叉树的神级方法 - - Morris遍历

【解题点】
Morris实现二叉树的后序打印
1，中序打印和先序打印的过程是一样的（？），只是打印时机不一样
2，后序把打印时机放在第二次回到某个节点的时候，这与中序打印的时机相似，然后逆序打印这个节点左子树的右边界
逆序打印用的是调整树的做法，而不是栈！
3，最后打印整棵树的右边界

【关键点】
理解Morris遍历到底是如何做到的？

找到二叉树中的最大搜索二叉子树

【题目五】
给定一棵二叉树的头节点head，已知其中所有节点的值都不一样，找到含有节点最多的搜索二叉子树，并返回这棵子树的头节点。

找到二叉树中的最大搜索二叉子树

【解题点】

最大的搜索二叉子树只可能来自以下两种情况。

第一种：如果来自node左子树上的最大搜索二叉子树是以node.left为头的；来自node右子树上的最大搜索二叉子树是以node.right为头的；node左子树上的最大搜索二叉子树的最大值小于node.value；node右子树上的最大搜索二叉子树的最小值大于node.value，那么以节点node为头的整棵树都是搜索二叉树。

第二种：如果不满足第一种情况，说明以节点node为头的树整体不能连成搜索二叉树。这种情况下，以node为头的树上的最大搜索二叉子树是来自node的左子树上的最大搜索二叉子树和来自node的右子树上的最大搜索二叉子树之间，节点数较多的那个。

找到二叉树中的最大搜索二叉子树

【解题点】

1．整体过程是二叉树的后序遍历。

2．遍历到当前节点记为cur时，先遍历cur的左子树收集4个信息，分别是左子树上最大搜索二叉子树的头节点（lBST）、节点数（lSize）、最小值（lMin）和最大值（lMax）。再遍历cur的右子树收集4个信息，分别是右子树上最大搜索二叉子树的头节点（rBST）、节点数（rSize）、最小值（rMin）和最大值（rMax）。

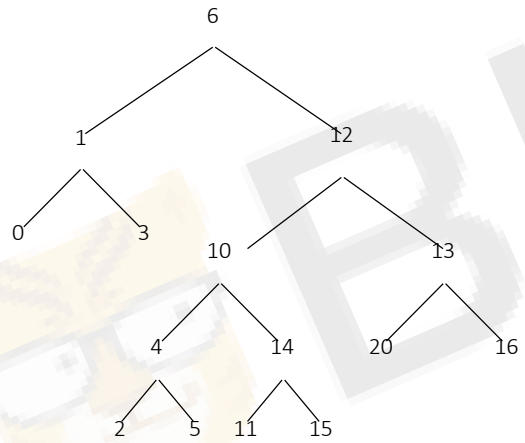
3．根据步骤2所收集的信息，判断是否满足第一种情况，如果满足第一种情况，就返回cur节点，如果满足第二种情况，就返回lBST和rBST中较大的一个。

4．可以使用全局变量的方式实现步骤2中收集节点数、最小值和最大值的问题。

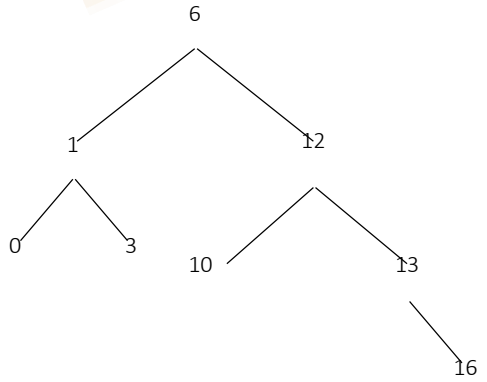
找到二叉树中符合搜索二叉树条件的最大拓扑结构

【题目六】

给定一棵二叉树的头节点head，已知所有节点的值都不一样，返回其中最大的且符合搜索二叉树条件的最大拓扑结构的大小。



二叉树



树中符合搜索二叉树条件的最大拓扑结构

找到二叉树中符合搜索二叉树条件的最大拓扑结构

【解题点】

方法一：时间复杂度O(N^2)的方法

方法二：时间复杂度O(N*logN)的方法 -> 拓扑贡献纪录法

详细解答，我已经把所有解释文字截图放在了代码的包里。

调整搜索二叉树中两个错误的节点

【题目七】
一棵二叉树原本是搜索二叉树，但是其中有两个节点调换了位置，使得这棵二叉树不再是搜索二叉树，请找到这两个错误节点并返回。已知二叉树中所有节点的值都不一样，给定二叉树的头节点head，返回一个长度为2的二叉树节点类型的数组errs，errs[0]表示一个错误节点，errs[1]表示另一个错误节点。
进阶：如果在原问题中得到了这两个错误节点，我们当然可以通过交换两个节点的节点值的方式让整棵二叉树重新成为搜索二叉树。但现在要求你不能这么做，而是在结构上完全交换两个节点的位置，请实现调整的函数。

调整搜索二叉树中两个错误的节点

【原问题解题点】
1，中序遍历二叉树，得到中序遍历序列
2，第一个错误节点为第一次降序时前一个节点，第二个错误节点为最后一次降序时后一个节点。

调整搜索二叉树中两个错误的节点

【进阶问题解题点】
1，第一个错误节点记为e1，e1的父节点记为e1P，e1的左孩子记为e1L，e1的右孩子记为e1R。第二个错误节点记为e2，e2的父节点记为e2P，e2的左孩子记为e2L，e2的右孩子记为e2R。 假设我们都已经找到
2，根据步骤一找到的节点们，把两个节点互换环境
3，这是一个纯练coding技巧的工作

调整搜索二叉树中两个错误的节点

【进阶问题解题点】
1，第一个错误节点记为e1，e1的父节点记为e1P，e1的左孩子记为e1L，e1的右孩子记为e1R。第二个错误节点记为e2，e2的父节点记为e2P，e2的左孩子记为e2L，e2的右孩子记为e2R。 假设我们都已经找到
2，根据步骤一找到的节点们，把两个节点互换环境
3，这是一个纯练coding技巧的工作

调整搜索二叉树中两个错误的节点

【进阶问题解题点】
1，第一个错误节点记为e1，e1的父节点记为e1P，e1的左孩子记为e1L，e1的右孩子记为e1R。第二个错误节点记为e2，e2的父节点记为e2P，e2的左孩子记为e2L，e2的右孩子记为e2R。 假设我们都已经找到
2，根据步骤一找到的节点们，把两个节点互换环境
3，这是一个纯练coding技巧的工作

调整搜索二叉树中两个错误的节点

【进阶问题解题点】
问题一：e1和e2是否有一个是头节点？如果有，谁是头？
问题二：e1和e2是否相邻？如果相邻，谁是谁的父节点？
问题三：e1和e2分别是各自父节点的左孩子还是右孩子？
特别注意：因为是在中序遍历时先找到e1，后找到e2，所以e1一定不是e2的右孩子，e2也一定不是e1的左孩子。

调整搜索二叉树中两个错误的节点

【进阶问题解题点】

1．e1是头，e1是e2的父，此时e2只可能是e1的右孩子。
2．e1是头，e1不是e2的父，e2是e2P的左孩子。
3．e1是头，e1不是e2的父，e2是e2P的右孩子。
4．e2是头，e2是e1的父，此时e1只可能是e2的左孩子。
5．e2是头，e2不是e1的父，e1是e1P的左孩子。
6．e2是头，e2不是e1的父，e1是e1P的右孩子。

调整搜索二叉树中两个错误的节点

【进阶问题解题点】

7．e1和e2都不是头，e1是e2的父，此时e2只可能是e1的右孩子，e1是e1P的左孩子。

8．e1和e2都不是头，e1是e2的父，此时e2只可能是e1的右孩子，e1是e1P的右孩子。

9．e1和e2都不是头，e2是e1的父，此时e1只可能是e2的左孩子，e2是e2P的左孩子。

10．e1和e2都不是头，e2是e1的父，此时e1只可能是e2的左孩子，e2是e2P的右孩子。

调整搜索二叉树中两个错误的节点

【进阶问题解题点】

11．e1和e2都不是头，谁也不是谁的父节点，e1是e1P的左孩子，e2是e2P的左孩子。

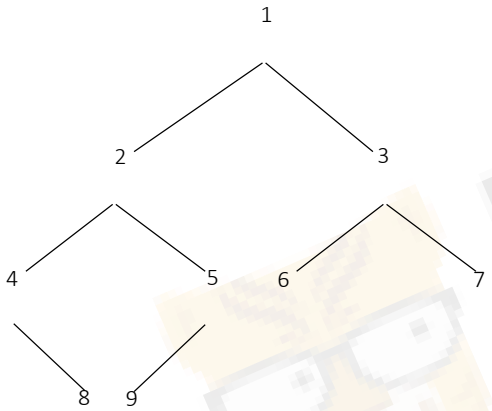
12．e1和e2都不是头，谁也不是谁的父节点，e1是e1P的左孩子，e2是e2P的右孩子。

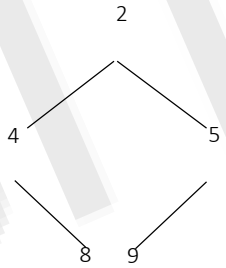
13．e1和e2都不是头，谁也不是谁的父节点，e1是e1P的右孩子，e2是e2P的左孩子。

14．e1和e2都不是头，谁也不是谁的父节点，e1是e1P的右孩子，e2是e2P的右孩子。

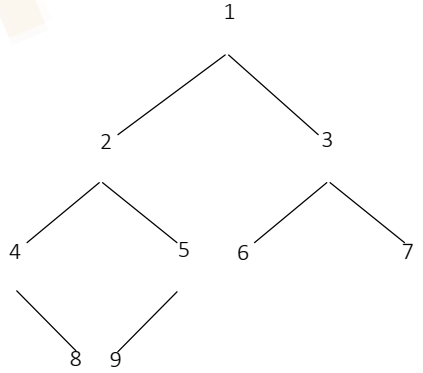
判断t1树中是否有与t2树拓扑结构完全相同的子树

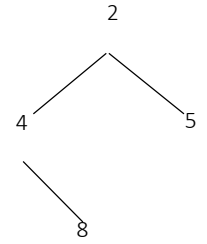
【题目八】
给定彼此独立的两棵树头节点分别为t1和t2，判断t1中是否有与t2树值的拓扑结构完全相同的子树。





TRUE





FALSE

判断t1树中是否有与t2树拓扑结构完全相同的子树

【解题点】

1，将t1和t2先序序列化成字符串s1和s2

2，检查s2是否是s1的子串即可

3，检查的过程用字符串的经典算法，KMP

4，如果t1节点数N， t2节点数M，时间复杂度O(N)

判断二叉树是否为平衡二叉树

【题目九】
平衡二叉树的性质为：要么是一棵空树，要么任何一个节点的左右子树高度差的绝对值不超过1。给定一棵二叉树的头节点head，判断这棵二叉树是否为平衡二叉树。

判断二叉树是否为平衡二叉树

【解题点】

1，对任何一个节点node来说，先遍历node的左子树，遍历过程中收集两个信息，node的左子树是否为平衡二叉树，node的左子树最深到哪一层记为lH。如果发现node的左子树不是平衡二叉树，无须进行任何后续过程，此时返回什么已不重要，因为已经发现整棵树不是平衡二叉树，退出遍历过程；

判断二叉树是否为平衡二叉树

【解题点】

2，如果node的左子树是平衡二叉树，再遍历node的右子树，遍历过程中再收集两个信息，node的右子树是否为平衡二叉树，node的右子树最深到哪一层记为rH。如果发现node的右子树不是平衡二叉树，无须进行任何后续过程，返回什么也不重要，因为已经发现整棵树不是平衡二叉树，退出遍历过程；

判断二叉树是否为平衡二叉树

【解题点】
3，如果node的右子树也是平衡二叉树，就看lH和rH差的绝对值是否大于1，如果大于1，说明已经发现整棵树不是平衡二叉树，如果不大于1，则返回lH和rH较大的一个。

判断一棵二叉树是否为搜索二叉树和完全二叉树

【题目十】
给定一个二叉树的头节点head，已知其中没有重复值的节点，实现两个函数分别判断这棵二叉树是否是搜索二叉树和完全二叉树。

判断一棵二叉树是否为搜索二叉树和完全二叉树

【解题点】
1，判断一棵二叉树是否是搜索二叉树，只要改写一个二叉树中序遍历，在遍历的过程中看节点值是否都是递增的即可。
2，改写Morris，面试官就震惊了呀！
3，但如果用Morris，千万别忘了调回来的过程！

判断一棵二叉树是否为搜索二叉树和完全二叉树

【解题点】
判断一棵二叉树是否是完全二叉树，依据以下标准会使判断过程变得简单且易实现：
1．从每层的左边向右边依次遍历所有的节点。
2．如果当前节点有右孩子无左孩子，直接返回false。
3．如果当前节点并不是左右孩子全有，那之后的节点必须都为叶节点，否则返回false。
4．遍历过程中如果不返回false，遍历结束后返回true。

在二叉树中找到一个节点的后继节点

【题目十一】
如果一个二叉树节点的结构比普通二叉树节点结构多了一个指向父节点的parent指针。假设有一棵新类型的节点组成的二叉树，树中每个节点的parent指针都正确地指向自己的父节点，头节点的parent指向null。只给一个在二叉树中的某个节点node，请实现返回node的后继节点的函数。在二叉树的中序遍历的序列中，node的下一个节点叫作node的后继节点。

在二叉树中找到一个节点的后继节点

【解题点】

情况1：如果node有右子树，那么后继节点就是右子树上最左边的节点。

在二叉树中找到一个节点的后继节点

【解题点】
情况2：如果node没有右子树，那么先看node是不是node父节点的左孩子，如果是左孩子，那么此时node的父节点就是node的后继节点；如果是右孩子，就向上寻找node的后继节点，假设向上移动到的节点记为s，s的父节点记为p，如果发现s是p的左孩子，那么节点p就是node节点的后继节点，否则就一直向上移动。

在二叉树中找到一个节点的后继节点

【解题点】
情况3：如果在情况2中一直向上寻找，都移动到空节点时还是没有发现node的后继节点，说明node根本不存在后继节点。

在二叉树中找到两个节点的最近公共祖先

【题目十二】
给定一棵二叉树的头节点head，以及这棵树中的两个节点o1和o2，请返回o1和o2的最近公共祖先节点。

在二叉树中找到两个节点的最近公共祖先

【解题点】
1，后序遍历二叉树，假设遍历到的当前节点为cur。因为是后序遍历，所以先处理cur的两棵子树。假设处理cur左子树时返回节点为left，处理右子树时返回right。
2．如果发现cur等于null，或者o1、o2，则返回cur。
3．如果left和right都为空，说明cur整棵子树上没有发现过o1或o2，返回null。

在二叉树中找到两个节点的最近公共祖先

【解题点】

4．如果left和right都不为空，说明左子树上发现过o1或o2，右子树上也发现过o2或o1，说明o1向上与o2向上的过程中，首次在cur相遇，返回cur。

5．如果left和right有一个为空，另一个不为空，假设不为空的那个记为node，此时node到底是什么？有两种可能，要么node是o1或o2中的一个，要么node已经是o1和o2的最近公共祖先。不管是哪种情况，直接返回node即可。

二叉树节点间的最大距离问题

【题目十三】
从二叉树的节点A出发，可以向上或者向下走，但沿途的节点只能经过一次，当到达节点B时，路径上的节点数叫作A到B的距离。给定一棵二叉树的头节点head，求整棵树上节点间的最大距离 。

二叉树节点间的最大距离问题

【解题点】
一个以h为头的树上，最大距离只可能来自以下三种情况：
h的左子树上的最大距离。
h的右子树上的最大距离。
h左子树上离h.left最远的距离 + 1(h) + h右子树上离h.right最远的距离。
三个值中最大的那个就是整棵h树中最远的距离。

二叉树节点间的最大距离问题

【解题点】

1，整个过程为后序遍历，在二叉树的每棵子树上执行步骤2。

2，假设子树头为h，处理h左子树，得到两个信息，左子树上的最大距离记为lMax，左子树上距离h左孩子的最远距离记为maxfromLeft。同理，处理h右子树得到右子树上的最大距离记为rMax和距离h右孩子的最远距离记为maxFromRight。那么maxfromLeft + 1 + maxFromRight就是跨h节点情况下的最大距离，再与lMax和rMax比较，把三者中的最值作为h树上的最大距离返回，maxfromLeft+1就是h左子树上离h最远的点到h的距离，maxFromRight+1就是h右子树上离h最远的点到h的距离，选两者中最大的一个作为h树上距离h最远的距离返回。

3，如何返回两个值？一个正常返回，另一个用全局变量表示。

统计所有不同的二叉树的数量

【题目十四】
给定一个整数*N*，如果*N*<1，代表空树结构，否则代表中序遍历的结果为{1,2,3,…，*N*}。请返回可能的二叉树结构有多少。

统计所有不同的二叉树的数量

【解题点】

如果中序遍历有序且无重复值，则二叉树必为搜索二叉树。假设num(a)代表*a*个节点的搜索二叉树有多少种可能，再假设序列为{1, …, *i*, …, *N*}，如果以1作为头节点，1不可能有左子树，故以1作为头节点有多少种可能的结构，完全取决于1的右子树有多少种可能结构，1的右子树有*N*-1个节点，所以有num(N-1)种可能。

统计所有不同的二叉树的数量

【解题点】

如果以*i*作为头节点，*i*的左子树有*i*-1个节点，所以可能的结构有num(i-1)种，右子树有*N*-*i*个节点，所以有num(N-i)种可能。故以*i*为头节点的可能结构有num(i-1)×num(N-i)种。

统计所有不同的二叉树的数量

【解题点】
如果以*N*作为头节点，*N*不可能有右子树，故以*N*作为头节点有多少种可能，完全取决于*N*的左子树有多少种可能，*N*的左子树有*N*-1个节点，所以有num(N-1)种。

把从1到*N*分别作为头节点时，所有可能的结构加起来就是答案，可以利用动态规划来加速计算的过程，从而做到$O(N^2)$的时间复杂度。

统计完全二叉树的节点数

【题目十五】
给定一棵完全二叉树的头节点head，返回这棵树的节点个数

统计完全二叉树的节点数

【解题点】

如果完全二叉树的层数为h，可以做到时间复杂度为$O(h^2)$，具体过程如下：

1．如果head==null，说明是空树，直接返回0。

2．如果不是空树，就求树的高度，求法是找到树的最左节点看能到哪一层，层数记为h。

统计完全二叉树的节点数

【解题点】

3．一个递归过程记为bs(node,l,h)，node表示当前节点，*l*表示node所在的层数，*h*表示整棵树的层数是始终不变的。bs(node,l,h)的返回值表示以node为头的完全二叉树的节点数是多少。

初始时node为头节点head，*l*为1，因为head在第1层，一共有*h*层始终不变。

统计完全二叉树的节点数

【解题点】

4．找到node右子树的最左节点，如果发现能到达最后一层。此时说明node的整棵左子树都是满二叉树，并且层数为*h*-l层，一棵层数为*h*-l的满二叉树，其节点数为2^{h-1}-1个。如果加上node节点自己，那么节点数为2^(h-1)-1+1==2^(h-1)个。此时如果再知道node右子树的节点数，那么以node为头的完全二叉树上到底有多少个节点就求出来了，就是bs(node.right,l+1,h)的结果，递归求即可最后返回2^(h-1)+bs(node.right,l+1,h)。

统计完全二叉树的节点数

【解题点】

5．找到node右子树的最左节点，如果发现它没有到达最后一层，说明node的整棵右子树都是满二叉树，并且层数为*h*-*l*-1层，一棵层数为*h*-*l*-1的满二叉树，其节点数为2^{h-l-1}-1个。如果加上node节点自己，那么节点数为2^(h-l-1)-1+1==2^(h-l-1)个。此时如果再知道node左子树的节点数，那么以node为头的完全二叉树上到底有多少个节点就求出来了，就是bs(node.left,l+1,h)的结果，递归去求即可，最后整体返回2^(h-l-1)+bs(node.left,l+1,h)。

统计完全二叉树的节点数

【解题点】

每一层只会选择一个节点node进行bs的递归过程，所以调用bs函数的次数为$O(h)$。每次调用bs函数时，都会查看node右子树的最左节点，所以会遍历$O(h)$个节点，整个过程的时间复杂度为$O(h^2)$。

课程项目负责人：Catherine

邮件：weiyi@bittiger.io

左程云答疑邮箱：chengyunzuo@gmail.com

微信二维码：

关注微信，获得太阁最新信息

微信: bit_tiger

官网: BitTiger.io