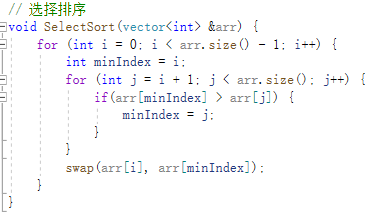
一、**简单排序算法**

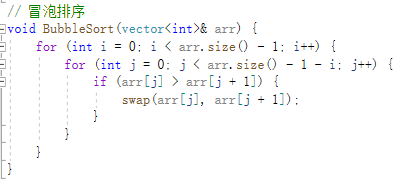
**选择排序**：时间复杂度O（n2）,空间复杂度O（1）

选择最小的与i位置进行交换。

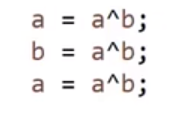


**冒泡排序**：时间复杂度O（n2）,空间复杂度O（1）

两两比较，大的放在后面。



**不使用额外变量情况下，如何交换两个数？**

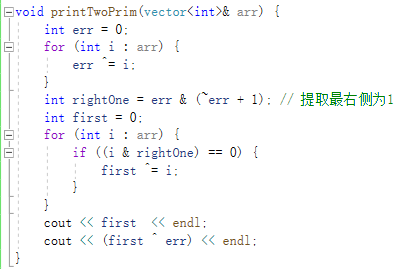


**在一个数组中有一种数是奇数个，其余数都是偶数个，如何获得这个奇数个的数字？**

将所有数进行异或运算得到的结果为输出。

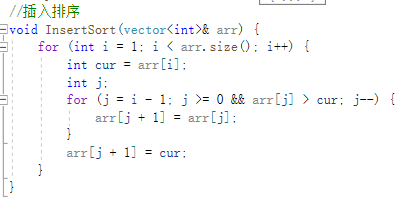
**在一个数组中有两种数是奇数个，其余数都是偶数个，如何获得这两个奇数个的数字？**

思路：第一次将所有数字异或得到err = a ^ b，a和b为想要找到的数字。取err中某一位为1的位，将数组中该位为1的所有数进行异或运算，得到a或b中的某一个，记为err’。最终两个数为err’和err^err’。



**插入排序**：时间复杂度O（n2）,空间复杂度O（1），数组有序时时间复杂度为O（N）

0-1，0-2，0到n-1逐渐有序。与前一个数进行比较，小的话进行交换，直到左边没数了或者比左边数字大。



二分法适用

有序数组是否存在某个数字、有序数组中小于等于某个数中最左面的位置、局部最小值

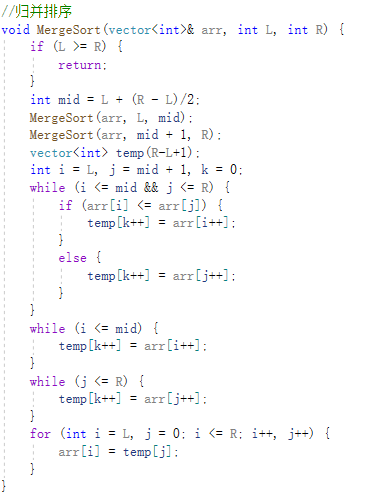
1. **NLogN排序算法**

中点位置的不溢出求法：mid = L + ((R-L) >> 1);

Master公式：T（N）= a\*T(N/b) + o(Nd)。a为子问题数，N/b为子问题规模。满足Master公式需要子问题规模一致。

**归并排序**：时间复杂度O（NlogN），空间复杂度O（N）

思路：中点的左右两侧有序，再合并。合并时需要辅助空间。



小和问题：在一个数组中，每一个数左边比当前数小的数累加起来，叫做这个数组的小和。求一个数组的小和。

利用归并排序过程，分为左右两个数组。在归并过程中，每个数组中有一个指针，当左侧指针的值小于右侧指针时，产生小和，小和为左侧指针的值×右侧指针后面的元素个数。

归并的时候，左右侧都是有序的，才能在O（1）时间内知道右侧有多少个数比左侧指针指向的数字大，同时当左右两侧数字相等时，先拷贝右侧的数字。（力扣315）

给定一个数组arr，和一个数num，请把小于等于num的数放在数组的左边，大于num的数放在数组的右边。要求额外空间复杂度O（1），时间复杂度O（N）。

思路：指针i从左向右开始进行遍历arr，当arr[i]<=num时，arr[i]与【小于等于】区间的下一个数进行交换，并且i++，【小于等于】区间右扩一个；否则，i++。

升级：荷兰国旗问题，给定数组arr和整数num，把小于、等于、大于num的数分为三个区域：小于区，等于区，大于区。

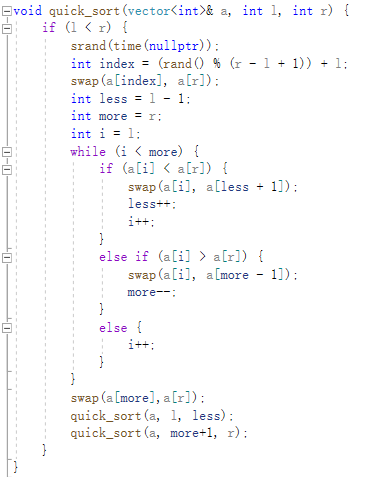
①arr[i]<num，arr[i]与【小于】区间的下一个数交换，【小于】区间右扩，i++。

②arr[i]=num，i++。

③arr[i]>num，arr[i]与【大于】区间的上一个数交换，【大于】区间左扩，i不动。

**快速排序**： 时间复杂度O（NlogN），空间复杂度O（logN），有序数组时间最差情况为O（N2），空间最差为O（N）。划分值选取最后一个数字。

1.0版本、2.0版本为上一题的两种递归。3.0版本：随机选取一个数字，与最后一个数字做交换，其他不变。



1. 桶排序以及排序内容大总结

堆结构

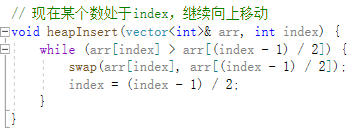
数组中连续的从0开始的一段可以对应到一棵完全二叉树上。节点i的左孩子为2\*i+1，右孩子为2\*i+2，父节点为（i-1）/2。

堆是完全二叉树，堆又分为大根堆和小根堆。

大根堆：每个子树的最大值就是头结点的值。小根堆：每个子树的最小值都是头结点的值。

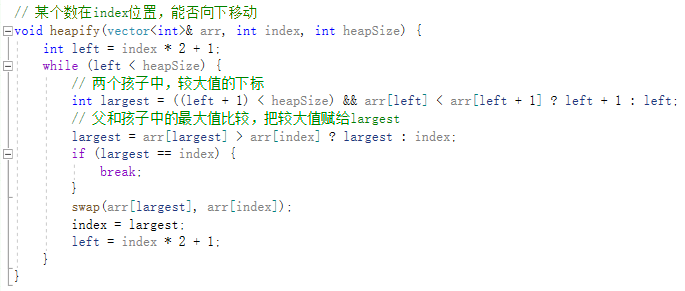
给定一个数组，根据数组初始化堆，使数组与大根堆相对应。（heapInsert，向上调整）

与父节点进行比较，比父节点大的话就进行交换。



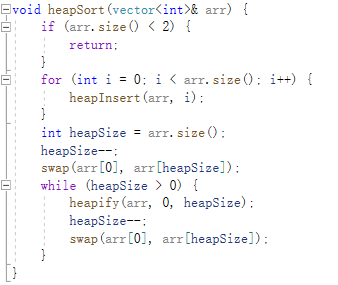
存在一个大根堆，要求返回最大值，并且将最大值在堆中删除，同时调整剩余仍为大根堆。（heapify，向下调整）

思路：删除根节点后，把堆中最后一个节点调整到根节点位置，针对该位置，执行heapify。



**堆排序**：时间复杂度O（NlogN）,空间复杂度O（1）。

思路：设置heapSize = 0，以插入数字的形式对数组进行调整，使其在0~n上是一个堆。然后将堆中的最大值（index = 0）与堆的最后一个值进行交换，然后heapsize--。从0位置做heapify，使得剩下的数字构成大根堆。继续把堆中0位置的数与最后一个数字做交换。重复上述过程，直到heapSize == 1。



直接给出一棵完全二叉树，如何调整为大根堆？（不采用heapInsert的方法）

index从后往前，对每一个节点执行heapify，heapSize 始终为数组长度。

堆排序扩展题目：已知一个有序的数组，几乎有序是指，如果把数组排好顺序的话，每个元素的移动的距离可以不超过k，并且相对于数组来说比较小。请选择一个合适的排序算法针对这个数据进行排序。

思路：举个例子，设k=6。维护一个小根堆heapSize=7 ，最开始时小根堆的范围是0-6，此时最小的元素一定会放置在0位置处。然后将小根堆移除第一个元素，把7位置的元素加入到小根堆中，重新建堆，得到的新堆中最小的元素一定会放在1位置上。

可以采用优先队列实现堆。

以上的排序都是基于比较的排序，后续的排序方法都不是基于比较的。计数排序、基数排序。

**~~基数排序~~**~~：~~

思路：设置标有0-9数字的桶，然后根据数组中的个位数、十位数、...的顺序，把数组中这一位上的数字与对应桶上的数字匹配，放入相应的桶中，全部处理后，将数字从0-9号桶中依次取出放回数组中，直到遍历完最高位。

1. 求数组的最大值有多少个十进制位。
2. 有多少位就需要进出桶多少次。
3. 需要一个长度为10的数组count，去记录本次遍历过程中，这一位（0-9）上的每个数字出现的频数，根据频数，将原数组的数字从右向左添加到辅助数组中。
4. count数组需要变为前缀和的形式，count[2]=25代表在该次遍历中，数组中有25次包含了2，此时从右向左遍历，当遇到该位正好为2的数字时，把该数字放在辅助数组24的位置，然后count[2]--。

排序算法的稳定性

不具备稳定性的排序：选择排序、快速排序、堆排序

具备稳定性的排序：冒泡排序、插入排序、归并排序、一切桶排序思想下的排序。

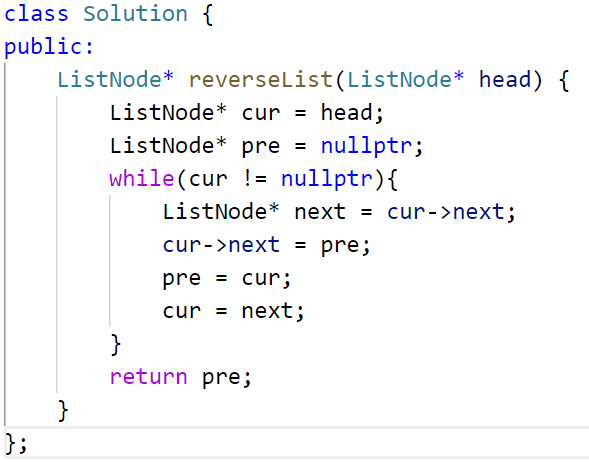
综合排序：范围比较少的时候采用插入排序，整体上采用快速排序。

系统上提供的排序算法，如果发现排序的对象是基本数据类型则会使用快排，如果是自定义的数据类型，则会使用归并排序。原因：稳定性

1. 链表

哈希表或者哈希映射中，插入元素时，如果key是基础数据类型，那么key会拷贝一份，如果key是自定义类型，则会按照引用传递。？

反转单向链表和双向链表，如果链表长度为N，时间复杂度要求为O（N），额外空间复杂度为O（1）。

（单向链表反转）

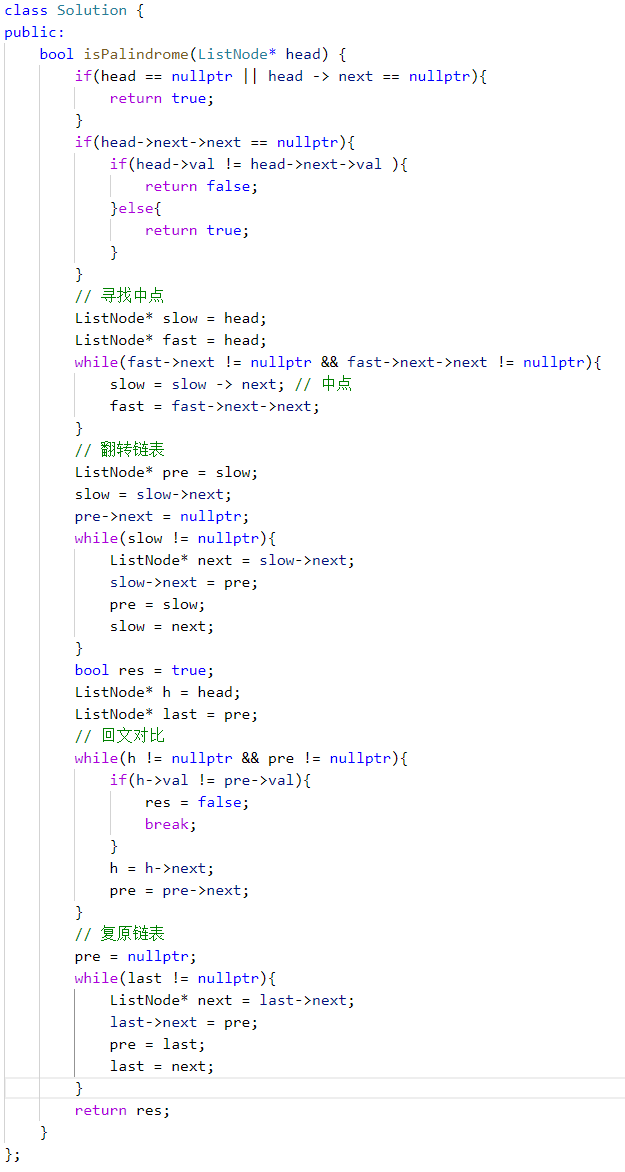
给定两个有序链表的头指针head1和head2，打印两个链表的公共部分。

思路：两个指针分别指向head1和head2，指向的节点谁小谁进行移动，当相等的时候打印，然后共同移动。

**题目**：判断一个链表是否为回文结构

方法一：采用栈对链表进行遍历，可以得到逆序的链表，但是需要额外O（N）的空间复杂度。

方法二：采用快慢指针获取链表的中间位置，然后将后部分的链表进行反转，用两个指针分别从两端向中间遍历，比较是否一致。额外空间复杂度O（1）。



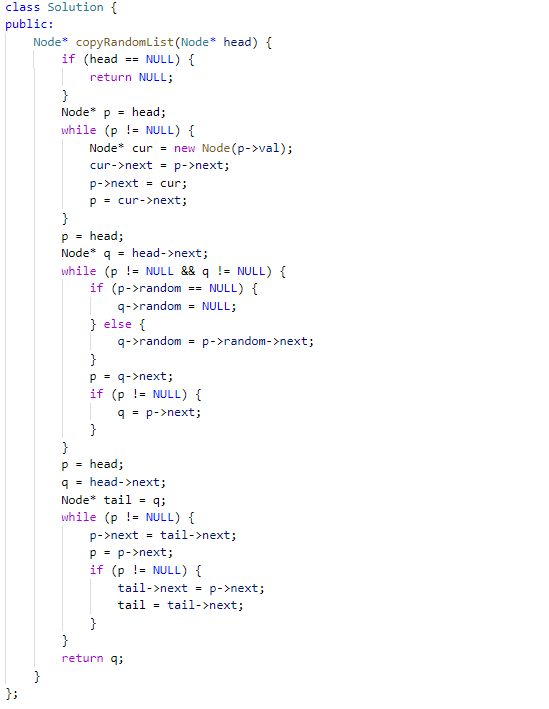
**题目**：给定一个单链表的头结点head，节点的值类型是整型，再给定一个整数pivot。实现一个调整链表的函数，将链表调整为左部分都是值小于pivot的节点，中间部分是等于pivot的节点，右部分都是大于pivot的节点。

简单思路：把链表存在数组中，然后再数组中进行快速排序的partition。

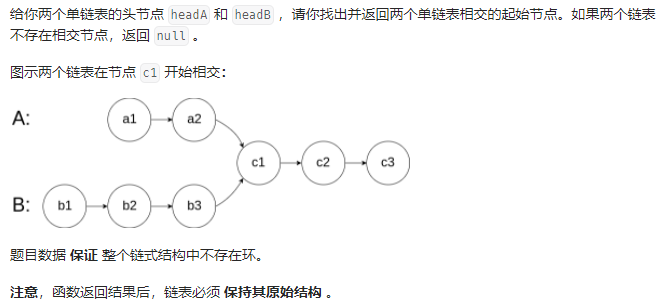
O（1）额外空间复杂的做法：需要6个指针，分别为指向大于区域的头尾指针，指向等于区域的头尾指针，指向小于区域的头尾指针。然后遍历整个链表，当遇到一个大于pivot的数字时，先判断大于区域的头尾指针是否为空，如果为空，则将头尾指针指向这个节点；如果不为空，将尾指针的next指向这个节点，然后将尾指针赋值为这个节点。

**题目**：随机链表的复制



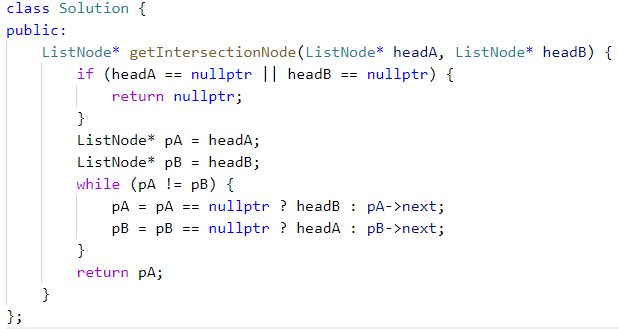


题目：两个单链表相交问题



需要O（1）空间复杂度。

思路：先遍历两个链表，两个链表如果有交点的话，最后一个节点的地址是相同的。然后计算两个链表的长度，如果一个是80，一个是100，第二次遍历的时候，先让100的那个走20步。写法上可以优化如下：



**在上述题目中，当两个链表可能存在环的时候**。如果一个链表有环，一个链表无环，不可能相交，因此只能是两个链表都有环。假设两个链表的入环点为loop1和loop2。

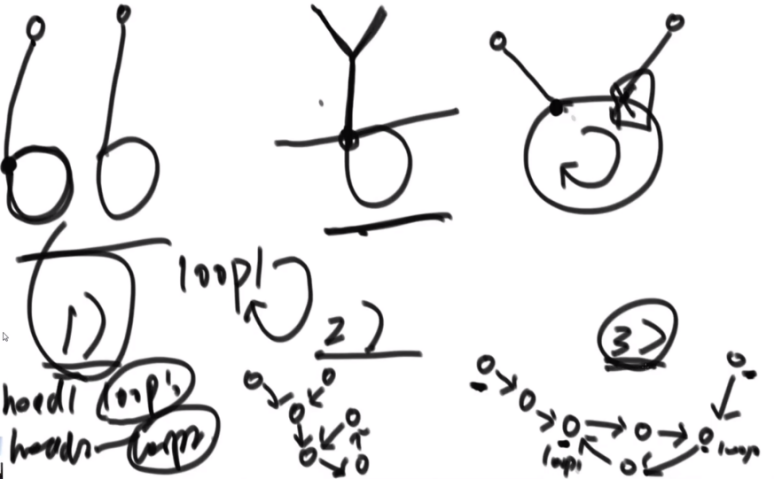
①两个有环的链表不相交

②两个有环链表的入环点是同一个

求两个环的入环地址，相同则为同一个入环点。然后将loop1设置为尾节点，然后根据无环链表相交节点的方式进行计算。

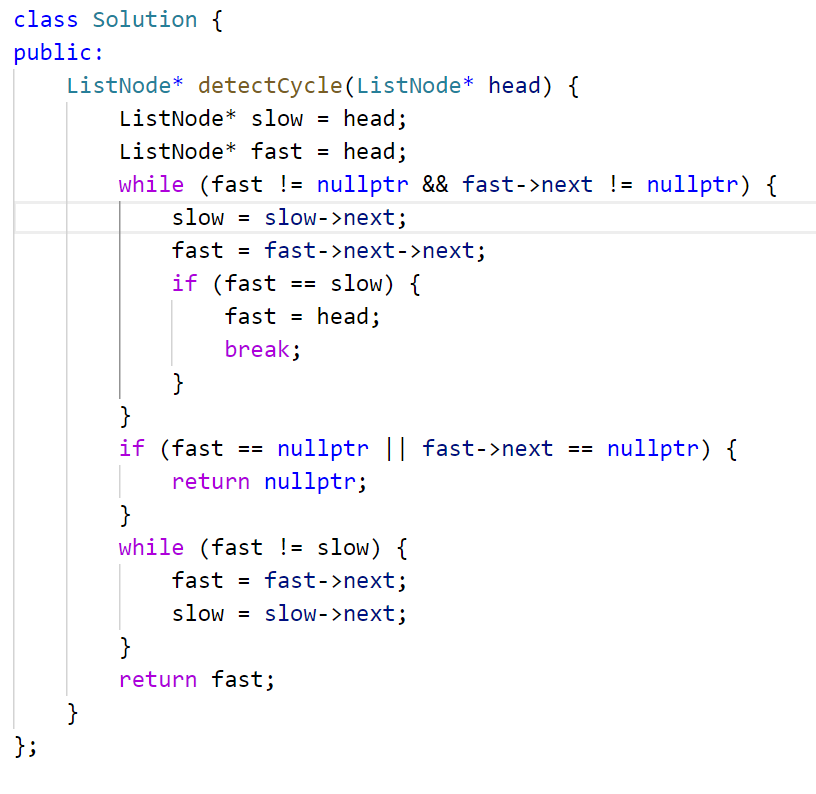
③两个有环链表的入环点不是同一个

对于第一种和第三种的区别，需要将loop1继续向下走，如果能遇到loop2，则为情况三，否则为情况一。情况三的返回值为loop1或者loop2。



题目：判断链表是否有环，并返回环形链表的第一个入环节点。

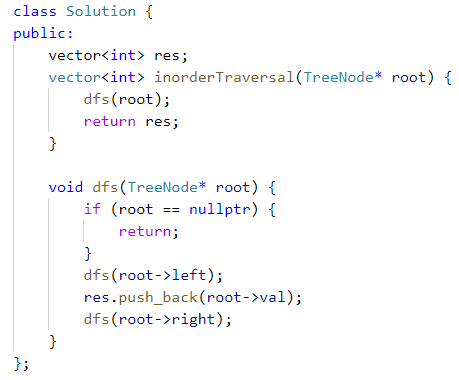
快慢指针，当快慢指针第一次相遇的时候，快指针回到头结点，快指针变为一次走一步，最终快慢指针的第二次相遇即为第一个入环节点。



1. 二叉树

二叉树遍历

①递归遍历。在递归遍历的基础上，可以衍生出先序遍历、中序遍历、后序遍历。



②非递归结构实现

先序遍历：先把头结点压入，每次1）从栈中弹出一个节点，2）打印/处理，3）先压入右节点再压入左节点，4）直到栈为空。（深度优先遍历）

中序遍历：先把树的左边界压入栈，然后弹出栈中一个节点，对该节点的右子树重复上述操作直到栈为空。



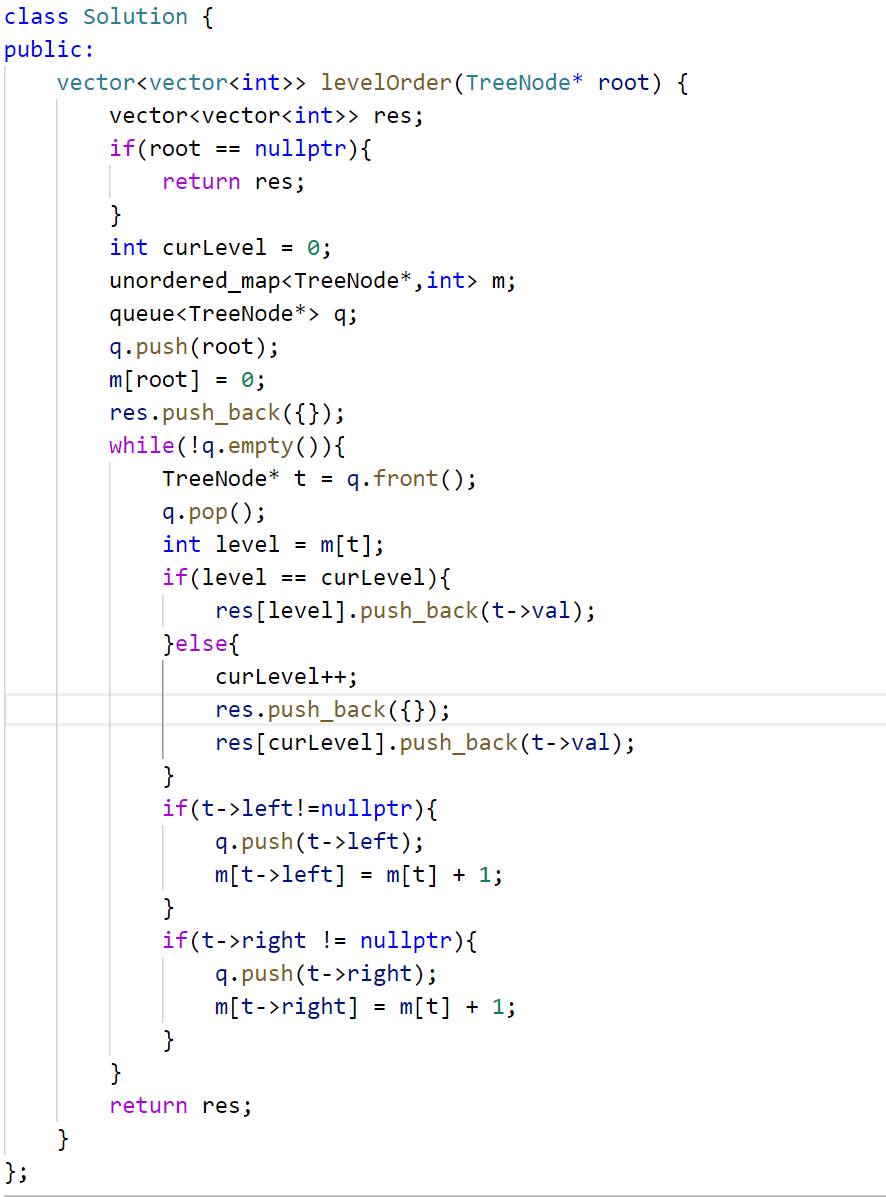
后序遍历：使用前序遍历的方法，但是先压入左节点再压入右节点，把遍历到的节点压入栈中，出栈顺序为后序遍历顺序，这需要两个栈。

**如何完成二叉树的宽度优先遍历？**

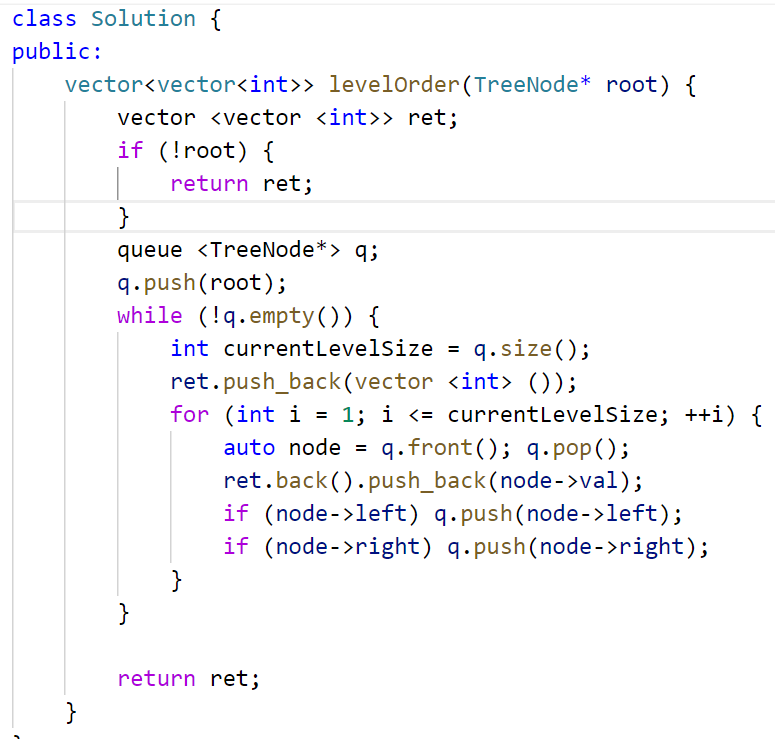
设置一个队列，先把头结点放到队列中。弹出队头的节点并输出，然后把该节点的左节点放入，再放入右节点到队列中，直到队列为空。

常见题目：求一棵二叉树的包含节点个数最多的一层的节点数。思路：设置一个map记录每一个节点处在哪一层中，出队的节点需要与当前层数进行比较。

类型题：层序遍历，按层输出。按照上述思路的代码：



另一种思路：

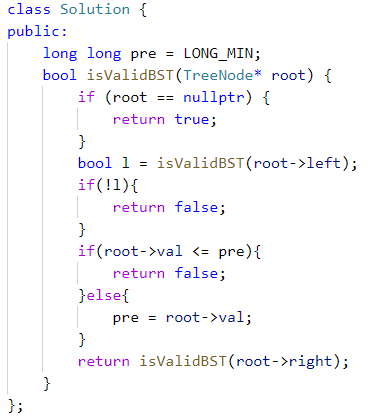


如何判断一棵树是搜索二叉树？

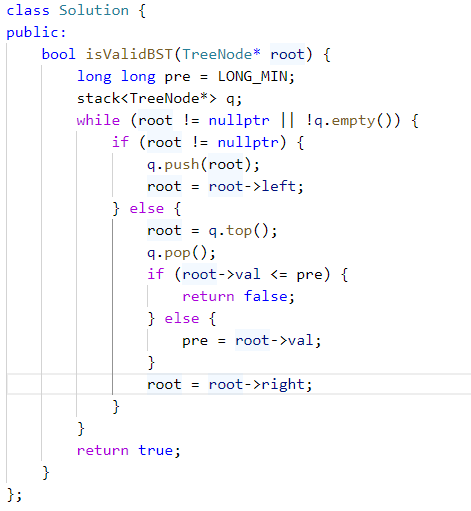
对于二叉树的任意一个节点，它的左子树节点都是比他小的，右子树节点都是比他大的。

判断方法：中序遍历为升序则为搜索二叉树。

递归方式判断：



非递归的方式



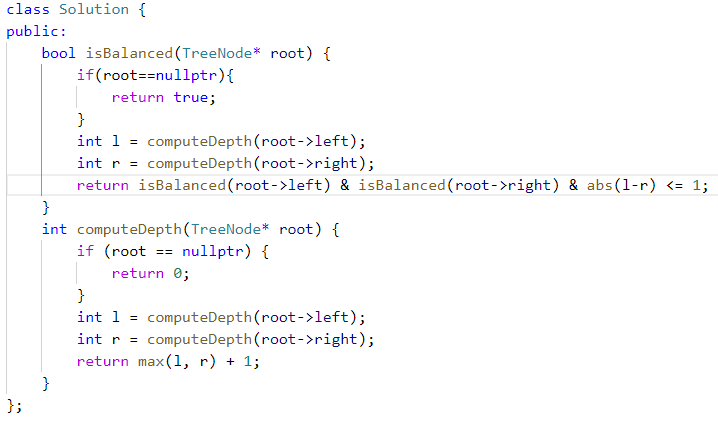
如何判断一棵树是完全二叉树？

采用层序遍历的方法，遍历过程中判断：1）任意节点如果只有右节点不为空，则返回false；2）在第一条不违规的情况下，如果遇到了第一个左右子节点不全，则后续节点全部为叶子节点。

如何判断一棵树是平衡二叉树？（每一颗子树的左右子树的高度差不超过1）

（二叉树题目套路）：节点可以从左子树和右子树中获取信息，需要考虑的只有头结点如何得到这些信息。树形DP。

左子树为平衡二叉树，右子树为平衡二叉树，|左子树高-右子树高|<=1。



改进：设计一个struct，在递归的时候同时计算深度和判断是否为平衡树。

根据上述套路，如何判断一棵树是否为搜索二叉树？



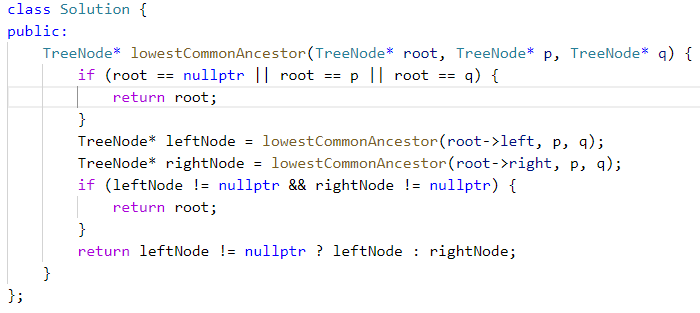
如何判断一棵树是满二叉树？

思路可以同上利用树形DP思想。求左边的高度和左边的节点数，求右边的高度和右边的节点数，最后得到根节点的高度和节点数。再判断h = 2n-1。

题目：给定两个二叉树的节点node1和node2，找到他们的最低公共祖先节点。

思路：遍历二叉树，用map存储每个节点的父节点。然后对node1，set存储所有的node1的全部祖先节点。再用node2依次向上找到第一个在上述集合中的节点。

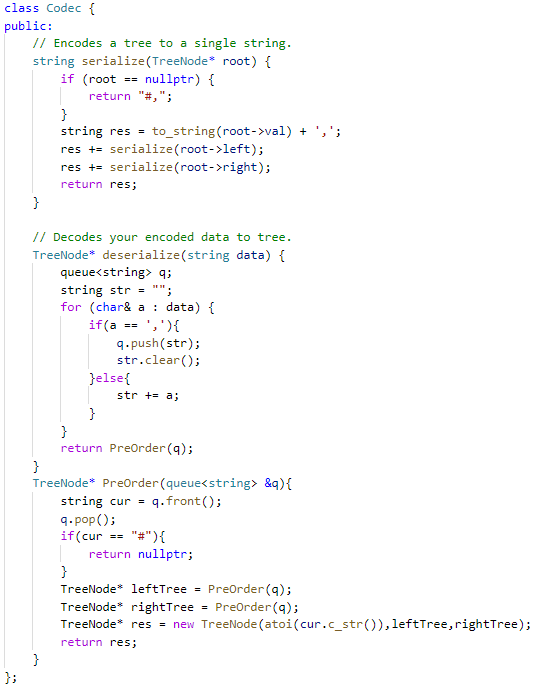
另一稍难的解法：node1和node2的结构关系只有两类：1和2中某个为另一个的祖先；1和2不互为公共祖先。



题目：带有指向父亲节点指针的二叉树中序遍历后继结点。

思路：1）如果x有右子树，那么右子树的最左节点为后继。2）x无右子树，需要向上走，判断遍历到的节点是不是父亲的左孩子，直到走到根节点。

题目：二叉树的序列化与反序列化



此为删减版

完整版联系电话/wx 13144140706