剑指offer(python版) 37-40

37数字在排序数组中出现的次数

题目：统计一个数字在排序数组中出现的次数

思路一：看到排序数组，首先想到的是二分法查找，先找到数字重复出现的左边位置，再找到右边位置，再相减即得到目标值

代码：

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. **def** GetNumberOfK(self, data, k):
4. # write code here
5. **if** len(data) < 1:
6. **return** 0
7. mid = len(data)//2
8. **if** data[mid] == k:
9. count,mida, midz = 1,mid-1, mid+1
10. **while** mida >= 0 **and** data[mida] == k:
11. count += 1
12. mida -= 1
13. **while** midz < len(data) **and** data[midz] == k:
14. count += 1
15. midz += 1
16. **return** count
17. **elif** data[mid]>k:
18. **return** self.GetNumberOfK(data[:mid], k)
19. **else**:
20. **return** self.GetNumberOfK(data[mid+1:],k)

思路二：直接遍历

代码：

1. **class** Solution:
2. **def** GetNumberOfK(self, data, k):
3. num=0
4. **for** i **in** data:
5. **if** i==k:
6. num+=1
7. **return** num

还有利用python的count函数，但是在算法题中，没有意义

代码：

1. **class** Solution:
2. **def** GetNumberOfK(self, data, k):
3. **return** data.count(k)

38二叉树的深度

题目：输入一棵二叉树，求该树的深度。从根结点到叶结点依次经过的结点（含根、叶结点）形成树的一条路径，最长路径的长度为树的深度

思路一：要求树的深度，可以用层次遍历二叉树，每遍历完一层，则加1

list A为当前层的结点，遍历A中的每个节点，然后用temp一一保存当前每个节点可能有的左结点和右结点

代码：

1. # -\*- coding:utf-8 -\*-
2. # class TreeNode:
3. #     def \_\_init\_\_(self, x):
4. #         self.val = x
5. #         self.left = None
6. #         self.right = None
7. **class** Solution:
8. **def** TreeDepth(self, pRoot):
9. # write code here
10. **if** **not** pRoot:
11. **return** 0
12. a=[]
13. depth=0
14. a.append(pRoot)
15. **while** a:
16. temp=[]
17. **for** node **in** a:
18. **if** node.left:
19. temp.append(node.left)
20. **if** node.right:
21. temp.append(node.right)
22. a=temp
23. depth+=1
24. **return** depth

思路二：递归遍历

代码：

1. # -\*- coding:utf-8 -\*-
2. # class TreeNode:
3. #     def \_\_init\_\_(self, x):
4. #         self.val = x
5. #         self.left = None
6. #         self.right = None
7. **class** Solution:
8. **def** TreeDepth(self, pRoot):
9. # write code here
10. **if** pRoot **is** None:
11. **return** 0
12. **return** max(1+self.TreeDepth(pRoot.left), 1+self.TreeDepth(pRoot.right))

39平衡二叉树

题目：输入一棵二叉树，判断该二叉树是否是平衡二叉树。

思路：平衡二叉树的定义是任何节点的左右子树高度差都不超过1的二叉树

递归判断左右子树高度差是否大于一

代码：

1. # -\*- coding:utf-8 -\*-
2. # class TreeNode:
3. #     def \_\_init\_\_(self, x):
4. #         self.val = x
5. #         self.left = None
6. #         self.right = None
7. **class** Solution:
8. **def** IsBalanced\_Solution(self, pRoot):
9. # write code here
10. **if** **not** pRoot:
11. **return** True
12. left=self.TreeDepth(pRoot.left)
13. right=self.TreeDepth(pRoot.right)
14. **if** abs(left-right)>1:
15. **return** False
16. **return** self.IsBalanced\_Solution(pRoot.left) **and** self.IsBalanced\_Solution(pRoot.right)
17. **def** TreeDepth(self, root):
18. **if** **not** root:
19. **return** 0
20. left=self.TreeDepth(root.left)
21. right=self.TreeDepth(root.right)
22. **return** max(left,right)+1

40数组中只出现一次的数字

题目：一个整型数组里除了两个数字之外，其他的数字都出现了偶数次。请写程序找出这两个只出现一次的数字

思路：ash表方法大家都能想到，但显然不是最佳方法。没有用到描述中其他数字出现偶数次这个条件。

位运算中异或的性质：两个相同数字异或为0，一个数和0异或还是它本身。   
当只有一个数出现一次时，我们把数组中所有的数，依次异或运算，最后剩下的就是落单的数，因为成对儿出现的都抵消了。   
依照这个思路，我们来看两个数（我们假设是AB）出现一次的数组。我们首先还是先异或，剩下的数字肯定是A、B异或的结果，这个结果的二进制中的1，表现的是A和B的不同的位。我们就取第一个1所在的位数，假设是第3位，接着把原数组分成两组，分组标准是第3位是否为1。如此，相同的数肯定在一个组，因为相同数字所有位都相同，而不同的数，肯定不在一组。然后把这两个组按照最开始的思路，依次异或，剩余的两个结果就是这两个只出现一次的数字（回顾思路：首先要想到如何将两个只出现一次的数（假设A和B）分在两个组中，A，B异或后的值中第一个1代表第一个不同，从而把整个数组按照这种不同分成两个组，再在每一个组中异或，从而可以得到A,B的值是多少）

代码：

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. # 返回[a,b] 其中ab是出现一次的两个数字
4. **def** FindNumsAppearOnce(self, array):
5. # write code here
6. **if** **not** array:
7. **return** []
8. # 对array中的数字进行异或运算
9. tmp = 0
10. **for** i **in** array:
11. tmp ^= i
12. # 获取tmp中最低位1的位置
13. idx = 0
14. **while** (tmp & 1) == 0:
15. tmp >>= 1
16. idx += 1
17. a = b = 0
18. **for** i **in** array:
19. **if** self.isBit(i, idx):
20. a ^= i
21. **else**:
22. b ^= i
23. **return** [a, b]
25. **def** isBit(self, num, idx):
26. """
27. 判断num的二进制从低到高idx位是不是1
28. :param num: 数字
29. :param idx: 二进制从低到高位置
30. :return: num的idx位是否为1
31. """
32. num = num >> idx
33. **return** num & 1