剑指offer(python版) 25-28

25复杂链表的复制

题目：输入一个复杂链表（每个节点中有节点值，以及两个指针，一个指向下一个节点，另一个特殊指针指向任意一个节点），返回结果为复制后复杂链表的head。（注意，输出结果中请不要返回参数中的节点引用，否则判题程序会直接返回空

思路：如果链表为空链表，则返回本身即可，如果非空 需要进行复制操作

第一步 复制原来的链表，顺次连接形成新链表

第二步，利用原节点的random指向，来用复制的相应节点的random

最后一步，将复制好的链表拆分出来，或者说将偶数位的节点重新拆分合成新的链表，得到的就是复制的链表

图解在https://blog.csdn.net/qq\_33431368/article/details/79296360

代码：

1. # -\*- coding:utf-8 -\*-
2. # class RandomListNode:
3. #     def \_\_init\_\_(self, x):
4. #         self.label = x
5. #         self.next = None
6. #         self.random = None
7. **class** Solution:
8. # 返回 RandomListNode
9. def Clone(self, pHead):
10. # write code here
11. **if** not pHead:
12. **return** pHead
13. cloNode = pHead
14. **while** cloNode:
15. node = RandomListNode(cloNode.label)
16. node.next = cloNode.next
17. cloNode.next = node
18. cloNode = node.next
19. cloNode = pHead
20. **while** cloNode:
21. node = cloNode.next
22. **if** cloNode.random:
23. node.random = cloNode.random.next
24. cloNode = node.next
25. cloNode = pHead
26. pHead = pHead.next
27. **while** cloNode.next:
28. node = cloNode.next
29. cloNode.next = node.next
30. cloNode = node
31. **return** pHead

26二叉搜索树与双向链表

题目：输入一棵二叉搜索树，将该二叉搜索树转换成一个排序的双向链表。要求不能创建任何新的结点，只能调整树中结点指针的指向

思路：核心算法依旧是中序遍历

不是从根节点开始，而是从中序遍历得到的第一个节点开始

定义两个辅助节点listHead(链表头节点)、listTail(链表尾节点)。事实上，二叉树只是换了种形式的链表；listHead用于记录链表的头节点，用于最后算法的返回；listTail用于定位当前需要更改指向的节点

图解在：https://blog.csdn.net/u010005281/article/details/79657259

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 \_\_init\_\_(self):
9. self.listHead=None
10. self.listTail=None
11. def Convert(self, pRootOfTree):
12. # write code here
13. **if** pRootOfTree==None:
14. **return**
15. self.Convert(pRootOfTree.left)
16. **if** self.listHead==None:
17. self.listHead=pRootOfTree
18. self.listTail=pRootOfTree
19. **else**:
20. self.listTail.right=pRootOfTree
21. pRootOfTree.left=self.listTail
22. self.listTail=pRootOfTree
23. self.Convert(pRootOfTree.right)
24. **return** self.listHead

27字符串的排列

题目：输入一个字符串,按字典序打印出该字符串中字符的所有排列。例如输入字符串abc,则打印出由字符a,b,c所能排列出来的所有字符串abc,acb,bac,bca,cab和cba

输入描述：输入一个字符串,长度不超过9(可能有字符重复),字符只包括大小写字母。

思路：

1. 挑选出所有可以放在第一位的字符，也就是把第一个字符与剩余部分每个字符交换一次。
2. 求 除第一个字符外的所有字符的全排列。

描述思想很显然可以用递归来实现。对于本题有两点需要额外注意，一是输入可能有重复字符，因此需要在交换的时候判断当前字符是否与待交换字符相等；二是需要按照字典顺序输出，所以需要对最终排列结果进行排序。

代码：

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. def Permutation(self, ss):
4. # write code here
5. out=[]
6. **if** len(ss)==0:
7. **return** out
8. charlist=list(ss)
9. self.permutation(charlist,0,out)
10. out=[''.join(out[i]) **for** i in range(len(out))]
11. out.sort()
12. **return** out
13. def permutation(self,ss,begin,out):
14. **if** begin==len(ss)-1:
15. out.append(ss[:])
16. **else**:
17. **for** i in range(begin,len(ss)):
18. **if** ss[begin]==ss[i] and i!=begin:
19. **continue**
20. **else**:
21. ss[begin],ss[i]=ss[i],ss[begin]
22. self.permutation(ss,begin+1,out)
23. ss[begin],ss[i]=ss[i],ss[begin]

28数组中出现次数超过一半的数字

题目：数组中有一个数字出现的次数超过数组长度的一半，请找出这个数字。例如输入一个长度为9的数组{1,2,3,2,2,2,5,4,2}。由于数字2在数组中出现了5次，超过数组长度的一半，因此输出2。如果不存在则输出0

思路：可以使用hash以空间换时间，把每个数字当作数组的下标，在python中用字典

代码：

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. **def** MoreThanHalfNum\_Solution(self, numbers):
4. # write code here
5. dict={}
6. **for** num **in** numbers:
7. **if** num **not** **in** dict:
8. dict[num]=1
9. **else**:
10. dict[num]+=1
11. **if** dict[num]>len(numbers)/2:
12. **return** num
13. **return** 0