剑指offer(python版)-21-24

21栈的压入、弹出序列

题目：输入两个整数序列，第一个序列表示栈的压入顺序，请判断第二个序列是否可能为该栈的弹出顺序。假设压入栈的所有数字均不相等。例如序列1,2,3,4,5是某栈的压入顺序，序列4,5,3,2,1是该压栈序列对应的一个弹出序列，但4,3,5,1,2就不可能是该压栈序列的弹出序列。（注意：这两个序列的长度是相等的）

思路：栈的压入顺序是指1,2,3,4,5是依次push到栈的，但并不是说只有push的过程，也可能有pop的操作，比如push 1，2，3，4之后，把4pop出去，然后再push5，也是一样的压入顺序；弹出序列是指每次pop出去的元素都是当时栈顶的元素，比如一开始pop1,2,3,4，然后pop4，再push5，再pop5，然后依次pop3,2,1，那么弹出序列就是4,5,3,2,1；

那么就可以构造一个辅助栈来判断弹出序列是不是和压栈序列对应。首先遍历压栈序列的元素push到辅助栈，判断是不是弹出序列的首元素，如果是，则弹出序列pop首元素（指针后移），如果不是，则继续push，再接着判断；直到遍历完了压栈序列，如果辅助栈或者弹出序列为空，则返回True，否则返回False

代码：

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. **def** IsPopOrder(self, pushV, popV):
4. # write code here
5. stack=[]
6. **for** i **in** pushV:
7. stack.append(i)
8. **while** stack **and** stack[-1]==popV[0]:
9. stack.pop()
10. popV.pop(0)
11. **return** True **if** **not** stack **else** False

22从上往下打印二叉树

题目：从上往下打印出二叉树的每个节点，同层节点从左至右打印。

思路：遍历，采用队列的方法

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. # 返回从上到下每个节点值列表，例：[1,2,3]
9. **def** PrintFromTopToBottom(self, root):
10. # write code here
11. re\_list=[]
12. **if** **not** root:
13. **return** []
14. queue=[root]
15. **while** len(queue):
16. a=queue.pop(0)
17. re\_list.append(a.val)
18. **if** a.left:
19. queue.append(a.left)
20. **if** a.right:
21. queue.append(a.right)
22. **return** re\_list

复习的知识点

pop():用于移除列表中的一个元素（默认最后一个元素），并且返回该元素的值

pop()方法语法：

list.pop([index=-1])

obj -- 可选参数，要移除列表元素的索引值，不能超过列表总长度，默认为 index=-1，删除最后一个列表值

23二叉搜索树的后序遍历序列

题目：输入一个整数数组，判断该数组是不是某二叉搜索树的后序遍历的结果。如果是则输出Yes,否则输出No。假设输入的数组的任意两个数字都互不相同

思路：后序遍历得到的序列，最后一个数是树的根节点的值 ，序列中最后一个数前面的数可以分为两部分：一部分是左子树节点的值，它们都比根节点的值小；第二部分是右子树节点的值，它们都比根节点的值要大。用递归的方法分别判断这两部分是否满足前面说得条件

代码：

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. **def** VerifySquenceOfBST(self, sequence):
4. # write code here
5. **if** len(sequence)==0:
6. **return** False
7. root=sequence[-1]
8. **for** i **in** range(len(sequence)):
9. **if** sequence[i]>root:
10. **break**
11. **for** j **in** range(i,len(sequence)):
12. **if** sequence[j]<root:
13. **return** False
14. left=True
15. **if** i>0:
16. left=self.VerifySquenceOfBST(sequence[0:i])
17. right=True
18. **if** i<len(sequence)-1 **and** left:
19. right=self.VerifySquenceOfBST(sequence[i:-1])
20. **return** right

24二叉树中和为某一值的路径

题目：输入一颗二叉树的跟节点和一个整数，打印出二叉树中结点值的和为输入整数的所有路径。路径定义为从树的根结点开始往下一直到叶结点所经过的结点形成一条路径。(注意: 在返回值的list中，数组长度大的数组靠前)

思路：DFS的思路

首先要理解题意，是从根节点往子节点连。

1、如果只有根节点或者找到叶子节点，我们就把其对应的val值返回

2、如果不是叶子节点，我们分别对根节点的左子树、右子树进行递归，直到找到叶子结点。然后遍历把叶子结点和父节点对应的val组成的序列返回上一层；如果没找到路径，其实也返回了序列，只不过是[]

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. # 返回二维列表，内部每个列表表示找到的路径
9. **def** \_\_init\_\_(self):
10. self.li = []
11. self.liAll = []
12. **def** FindPath(self, root, expectNumber):
13. # write code here
14. **if** root **is** None:
15. **return** self.liAll
16. self.li.append(root.val)
17. expectNumber -= root.val
18. **if** expectNumber==0 **and** **not** root.left **and** **not** root.right:
19. self.liAll.append(self.li[:])
20. self.FindPath(root.left,expectNumber)
21. self.FindPath(root.right,expectNumber)
22. self.li.pop()
23. **return** self.liAll