剑指offer(1)—python版本

1二维数组中的查找

题目：在一个二维数组中（每个一维数组的长度相同），每一行都按照从左到右递增的顺序排序，每一列都按照从上到下递增的顺序排序。请完成一个函数，输入这样的一个二维数组和一个整数，判断数组中是否含有该整数。

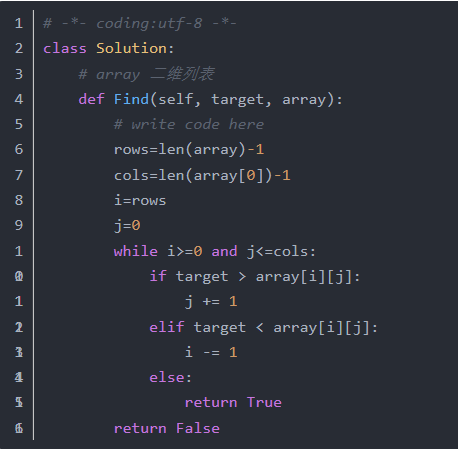
思路：首先选取右上角的数字，如果该数字大于target，则该列全大于target，删除该列；如果该数字小于小于target，则该列全小于target，删除该行。从右上角元素开始，当没到左下角元素前，不断判断右上角元素和target的关系，可以不断缩小查找范围

Ps:

Python中没有数组的概念，在此用list类型来表示数组，二维数组可以用二维列表来表示。对于一个二维列表array，array[0]表示第一行，len(array)表示行数，len(array[0])则表示列数

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. # array 二维列表
4. **def** insert2darray(self,seq,row,col):
5. array=[[0 **for** i **in** range(row)] **for** i **in** range(col)]
6. **for** i **in** range(row):
7. **for** j **in** range(col):
8. array[i][j]=seq[i\*row+j]
9. **return** array
10. **def** Find(self, target, array):
11. # write code here
12. # 主要思路：首先选取右上角的数字，如果该数字大于target，则该列全大于target，删除该列；         # 如果该数字小于小于target，则该列全小于target，删除该行。
13. flag=false
14. row=len(array)
15. **if** row:
16. col=len(array[0])
17. **else**:
18. col=0
19. **if** row>0 **and** col>0:
20. i=0
21. j=col-1
22. **while** i<row **and** j>=0:
23. **if** array[i][j]==target:
24. flag=true
25. **break**
26. **elif** array[i][j]>target:
27. j-=1
28. **elif** array[i][j]<target:
29. i+=1
30. **return** flag
32. **if** \_\_name\_\_=='\_\_main\_\_':
33. answer=Solution()
34. seq=[1,2,8,9,2,4,9,12,4,7,10,13,6,8,11,15]
35. matrix=answer.insert2darray(seq,4,4)
36. **print**(matrix)
37. **print**(answer.Find(7,matrix))

其实在题目中不需要写上array的代码，所以直接利用思路的方法写出来即可



https://www.cnblogs.com/yajun-yin/p/8401949.html

https://blog.csdn.net/Blueberry\_l/article/details/79375514

2替换空格

题目：请实现一个函数，将一个字符串中的每个空格替换成“%20”。例如，当字符串为We Are Happy.则经过替换之后的字符串为We%20Are%20Happy

思路：对空格split得到list，用‘%20’连接（join）这个list（只要一行代码即可）

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. # s 源字符串
4. **def** replaceSpace(self, s):
5. **return** '%20'.join(s.split(' '))

8. **if** \_\_name\_\_ == '\_\_main\_\_':
9. a = Solution()
10. **print**(a.replaceSpace('r y uu'))

思路二（比较麻烦）：由于替换空格后，字符串长度需要增大。先扫描空格个数，计算字符串应有的长度，从后向前一个个字符复制（需要两个指针）。这样避免了替换空格后，需要移动的操作。

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. # s 源字符串
4. **def** replaceSpace(self, s):
5. num\_space = 0
6. **for** i **in** s:
7. **if** i == ' ':
8. num\_space += 1
10. new\_length = len(s) + 2 \* num\_space
11. index\_origin = len(s) - 1
12. index\_new = new\_length - 1
13. new\_string = [None **for** i **in** range(new\_length)]
15. **while** index\_origin >= 0 & (index\_new > index\_origin):
16. **if** s[index\_origin] == ' ':
17. new\_string[index\_new] = '0'
18. index\_new -= 1
19. new\_string[index\_new] = '2'
20. index\_new -= 1
21. new\_string[index\_new] = '%'
22. index\_new -= 1
23. **else**:
24. new\_string[index\_new] = s[index\_origin]
25. index\_new -= 1
26. index\_origin -= 1
27. **return** ''.join(new\_string)

30. **if** \_\_name\_\_ == '\_\_main\_\_':
31. a = Solution()
32. **print**(a.replaceSpace('r y uu'))

思路三：用python字符串的replace方法

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. # s 源字符串
4. **def** replaceSpace(self, s):
5. # write code here
6. **return** s.replace(' ', '%20')

本题需要回顾的知识点

1 split()：拆分字符串。通过指定分隔符对字符串进行切片，并返回分割后的字符串列表（list）

语法：str.split(str="",num=string.count(str))[n]

参数说明：  
str:表示为分隔符，默认为空格，但是不能为空('')。若字符串中没有分隔符，则把整个字符串作为列表的一个元素  
num:表示分割次数。如果存在参数num，则仅分隔成 num+1 个子字符串，并且每一个子字符串可以赋给新的变量  
[n]:表示选取第n个分片

注意：当使用空格作为分隔符时，对于中间为空的项会自动忽略

举例：

string = "www.gziscas.com.cn"

1.以'.'为分隔符

print(string.split('.'))

['www', 'gziscas', 'com', 'cn']

2.分割两次

print(string.split('.'，2))

['www', 'gziscas', 'com.cn']

3.分割两次，并取序列为1的项

print(string.split('.',2)[1])

gziscas

2 join()： 连接字符串数组。将字符串、元组、列表中的元素以指定的字符(分隔符)连接生成一个新的字符串

语法：  'sep'.join(seq)

参数说明  
sep：分隔符。可以为空  
seq：要连接的元素序列、字符串、元组、字典  
上面的语法即：以sep作为分隔符，将seq所有的元素合并成一个新的字符串

返回值：返回一个以分隔符sep连接各个元素后生成的字符串

举例：

#对序列进行操作（分别使用' '与':'作为分隔符）

>>> seq1 = ['hello','good','boy','doiido']

>>> print ' '.join(seq1)

hello good boy doiido

>>> print ':'.join(seq1)

hello:good:boy:doiido

#对字符串进行操作

>>> seq2 = "hello good boy doiido"

>>> print ':'.join(seq2)

h:e:l:l:o: :g:o:o:d: :b:o:y: :d:o:i:i:d:o

#对元组进行操作

>>> seq3 = ('hello','good','boy','doiido')

>>> print ':'.join(seq3)

hello:good:boy:doiido

#对字典进行操作

>>> seq4 = {'hello':1,'good':2,'boy':3,'doiido':4}

>>> print ':'.join(seq4)

boy:good:doiido:hello

3从尾到头打印链表

题目：输入一个链表，按链表值从尾到头的顺序返回一个ArrayList

思路1：把链表上的值转换成一个list,然后利用reverse函数/或者一个反转[::-1]

1. # -\*- coding:utf-8 -\*-
2. # class ListNode:
3. #     def \_\_init\_\_(self, x):
4. #         self.val = x
5. #         self.next = None
7. **class** Solution:
8. # 返回从尾部到头部的列表值序列，例如[1,2,3]
9. **def** printListFromTailToHead(self, listNode):
10. # write code here
11. out=[]
12. **while** ListNode:
13. out.append(ListNode.val)
14. ListNode=ListNode.next
15. out.reverse()  /out[::-1]
16. **return** out

思路二：把每个节点的值取下来插到前端即可

1. # -\*- coding:utf-8 -\*-
2. # class ListNode:
3. #     def \_\_init\_\_(self, x):
4. #         self.val = x
5. #         self.next = None
7. **class** Solution:
8. # 返回从尾部到头部的列表值序列，例如[1,2,3]
9. **def** printListFromTailToHead(self, listNode):
10. # write code here
11. list\_val = []
12. **while** listNode:
13. list\_val.insert(0,listNode.val)#list.insert（a，b）a为插入的索引值 b为插入的值，此句的意思为每次都插入到列表的最前面
14. listNode = listNode.next
15. **return** list\_val

本题需要回顾的知识点：

list的反转可以使用[::-1]（切片）

4重建二叉树

题目：输入某二叉树的前序遍历和中序遍历的结果，请重建出该二叉树。假设输入的前序遍历和中序遍历的结果中都不含重复的数字。例如输入前序遍历序列{1,2,4,7,3,5,6,8}和中序遍历序列{4,7,2,1,5,3,8,6}，则重建二叉树并返回。

思路：递归的思路

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. # 返回构造的TreeNode根节点
9. **def** reConstructBinaryTree(self, pre, tin):
10. # write code here
11. **if** len(pre)==0:
12. **return** none
13. **if** len(pre)==1:
14. **return** TreeNode(pre[0])
15. **else**:
16. flag=TreeNode(pre[0])
17. flag.left = self.reConstructBinaryTree(pre[1:tin.index(pre[0])+1],tin[:tin.index(pre[0])])
18. flag.right = self.reConstructBinaryTree(pre[tin.index(pre[0])+1:],tin[tin.index(pre[0])+1:])
19. **return** flag