1. 给一个整数数组，找到两个数使得他们的和等于一个给定的数 *target*。

你需要实现的函数twoSum需要返回这两个数的下标, 并且第一个下标小于第二个下标。注意这里下标的范围是 1 到 *n*，不是以 0 开头。

 注意事项

你可以假设只有一组答案。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给出 numbers = [2, 7, 11, 15], target = 9, 返回 [1, 2].

深度搜索把太麻烦，连续和那么前n和可以。

class Solution:

"""

@param numbers : An array of Integer

@param target : target = numbers[index1] + numbers[index2]

@return : [index1 + 1, index2 + 1] (index1 < index2)

"""

def twoSum(self, numbers, target):

# write your code here

length=len(numbers)

for i in range(length):

for j in range(i+1,length):

sum=numbers[i]+numbers[j]

if sum==target:

return [i+1,j+1]

2. 给定一个整数数组，请找出一个连续子数组，使得该子数组的和最大。输出答案时，请分别返回第一个数字和最后一个数字的下标。（如果两个相同的答案，请返回其中任意一个）

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给定 [-3, 1, 3, -3, 4], 返回[1,4].

这个没办法了前向和

#算法超时

class Solution:

# @param {int[]} A an integer array

# @return {int[]} A list of integers includes the index of the

# first number and the index of the last number

def continuousSubarraySum(self, A):

# Write your code here

length=len(A)

if length==1:

return [0,0]

temp=[]

sum1=0

for key in A:

sum1+=key

temp.append(sum1)

temp.insert(0,0)

max2=0

for i in range(len(temp)):

for j in range(i,len(temp)):

max1=temp[j]-temp[i]

if max1>max2:

max2=max1

start=i

end=j-1

return [start,end]

so需要换个把不遍历找了小于0放弃直接等于0 start也要改。全为负时要注意。

class Solution:

# @param {int[]} A an integer array

# @return {int[]} A list of integers includes the index of the

# first number and the index of the last number

def continuousSubarraySum(self, A):

# Write your code here

length=len(A)

if length==1:

return [0,0]

temp=[]

sum1=0

start=0

max1=-8888

for i in range(len(A)):

sum1+=A[i]

if sum1<0:

sum1=0

start=i+1

if max1<sum1:

max1=sum1

end=i

temp=[max1,start,end]

max1=-8888

if temp[1]>temp[2]:

for i in range(len(A)):

if max1<A[i]:

max1=A[i]

temp=[max1,i,i]

return [temp[1],temp[2]]

3. 求逆波兰表达式的值。

在逆波兰表达法中，其有效的运算符号包括 +, -, \*, / 。每个运算对象可以是整数，也可以是另一个逆波兰计数表达。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

["2", "1", "+", "3", "\*"] -> ((2 + 1) \* 3) -> 9

["4", "13", "5", "/", "+"] -> (4 + (13 / 5)) -> 6

思路：符号之前两个数可以与符号做运算，怎么做出栈做，做好之后在入栈

class Solution:

# @param {string[]} tokens The Reverse Polish Notation

# @return {int} the value

def evalRPN(self, tokens):

# Write your code here

if (len(tokens) == 0):

return 0

tokens\_s = []

result = 0

i = 0

x = 0

y = 0

for i in range(0,len(tokens)):

element = tokens[i]

if ((element== "+") or (element== "-") or (element== "\*") or (element== "/")):

y = tokens\_s.pop()

x = tokens\_s.pop()

tokens\_s.append(int(caculate(element,int(x),int(y))))

else:

tokens\_s.append(int(element))

return (int)(tokens\_s.pop())

def caculate(oper, x, y):

if oper== '+' :

return x+y

if oper== '-' :

return x-y

if oper== '\*' :

return x\*y

if oper== '/' :

return 1.0\*x/y

4.

给定一个字符串所表示的括号序列，包含以下字符： '(', ')', '{', '}', '[' and ']'， 判定是否是有效的括号序列。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

括号必须依照 "()" 顺序表示， "()[]{}" 是有效的括号，但 "([)]"则是无效的括号。

思路有效括号，统计肯定不行，平时思路就是后面与前面对比。相同就出去。

class Solution:

# @param {string} s A string

# @return {boolean} whether the string is a valid parentheses

def isValidParentheses(self, s):

# Write your code here

table = {"(": ")", "[": "]", "{": "}"}

n = len(s)

stack = []

for i in s:

if len(stack) > 0 and stack[-1] in table and table[stack[-1]] == i:

stack.pop()

else:

stack.append(i)

return len(stack) == 0

5. 给定一个字符串， 包含大小写字母、空格' '，请返回其最后一个单词的长度。

如果不存在最后一个单词，请返回 0 。

 注意事项

一个单词的界定是，由字母组成，但不包含任何的空格。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给定 s = "Hello World"，返回 5。

思路：分割空格，然后最后一个单词就行。

class Solution:

# @param {string} s A string

# @return {int} the length of last word

def lengthOfLastWord(self, s):

# Write your code here

a=[]

s=s.strip()

for i in range(len(s)-1,-1,-1):

if s[i] !=' ':

a.append(s[i])

else:

return len(a)

return len(a)

6. 给定一个文档(Unix-style)的完全路径，请进行路径简化。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

"/home/", => "/home"

"/a/./b/../../c/", => "/c"

* 你是否考虑了 路径 = "/../" 的情况？

在这种情况下，你需返回"/"。

* 此外，路径中也可能包含双斜杠'/'，如 "/home//foo/"。

在这种情况下，可忽略多余的斜杠，返回 "/home/foo"。

思路：分情况讨论吧：最后一个字母是/舍弃，"/../"返回/，//返回/

class Solution:

# @param {string} path the original path

# @return {string} the simplified path

def simplifyPath(self, path):

# Write your code here

path\_array = path.split("/")

stack = []

res\_path = ""

for item in path\_array:

if item not in ["",".",".."]:

stack.append(item)

if ".." == item and stack:

del(stack[-1])

if []==stack:

return "/"

for item in stack:

res\_path += "/"+item + ""

return res\_path

7. 给定一个未经排序的数组，请找出其排序表中连续两个要素的最大间距。

如果数组中的要素少于 2 个，请返回 0.

 注意事项

可以假定数组中的所有要素都是非负整数，且最大不超过 32 位整数。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给定数组 [1, 9, 2, 5]，其排序表为 [1, 2, 5, 9]，其最大的间距是在 5 和 9 之间，= 4.

思路直接排序的复杂度是nlg，要在线性时间内完成的话需要

方案二：O(n)

1. 扫描一遍数组，找到数组中的最大值max，最小值min.(时间复杂度为O(n))  
   2.将[min,max]区间分为n-1份区域，则每一个区域的大小为len=(max-min)/(n-1);区间分别为：[min,min+len),[min+len,min+2\*len),[min+2\*len,min+3\*len),...[max-len,max].  
   3.最大间距=相邻的两个区间 or 不相邻的两个区间，但中间区间没有元素 且 是前一个区间的最大值与后一个区间的最小值之间产生。  
   4.求出每个区间的最大值，以及最小值，然后扫描一遍就能求出最大间距。

即全部排序很麻烦，只要能找到最大值最小值便可以得到最远的距离。因此划分为n-1个区间，那每个点所在位置就能被确定。每个区间内部不关心。

class Solution:

# @param nums: a list of integers

# @return: the maximum difference

def maximumGap(self, nums):

# write your code here

'''排序算法cost298ms

length=len(nums)

if length<=1:

return 0

nums.sort()

temp=[]

for i in range(length-1):

temp.append(nums[i+1]-nums[i])

return max(temp)

'''

#use greedy :

length=len(nums)

if length<=1:

return 0

max1=max(nums)

min1=min(nums)

step=1.0\*(max1-min1)/(length-1)#n个数需要n步到达。

if step==0.0:

return 0

max2=min1-2

min2=max1+2

a=[[min2,max2] for i in range(length)]

for i in nums:

j=int((i-min1)//step)

if i>a[j][1]:

a[j][1]=i

if i<a[j][0]:

a[j][0]=i

print(a)

maxlength=0

temp=[]

flag=0

h=0

while h<length:

if a[h][1]==max2 and a[h][0]==min2:

pass

else:

gap=a[h][1]-a[h][0]

if maxlength<gap:

maxlength=gap

temp.append(a[h])

h+=1

for k in range(len(temp)-1):

gap=min(temp[k+1])- max(temp[k])

if maxlength<gap:

maxlength=gap

return maxlength

8. 在一个排序矩阵中找从小到大的第 *k* 个整数。

排序矩阵的定义为：每一行递增，每一列也递增。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给出 *k* = 4 和一个排序矩阵：

[

[1 ,5 ,7],

[3 ,7 ,8],

[4 ,8 ,9],

]

返回 5。

思路：往右往下搜小于的入栈，i大的入丈，i还要加接着比较，否则就是j加，边界条件在n，m上，return语句在个数等于需要，并出栈一个（错误）会出现遗漏现象，必须克服掉。那就是要保存多个值，因此在搜索时进行排序意义不大，甚至于是不可信的。因此比较的点不在是两个点比较。最坏是对角线上的比较。因此需要list保存节点信息，然后排序，寻找。[当存在多个相等的时候，存在巨大问题]。所以不如找了排序，然后根据排序信息来确定需要的节点。在多个排序上。要保留位置信息。所以遍历是要记录以保证所有的节点是未遍历的节点。

class Solution:

# @param matrix: a matrix of integers

# @param k: an integer

# @return: the kth smallest number in the matrix

def kthSmallest(self, matrix, k):

# write your code here

row=len(matrix)

column=len(matrix[0])

record=[[0 for m in range(column)]for n in range(row)]

a=[]

a.append([matrix[0][0],[0,0]])

while k!=0:

k-=1

a=sorted(a,key=lambda x:x[0],reverse=False)

if k==0:

return a[0][0]

t=a[0]

del(a[0])#pop is danchu the first element of list,but del is delete the

if t[1][0]+1<row and record[t[1][0]+1][t[1][1]]==0:

a.append([matrix[t[1][0]+1][t[1][1]],[t[1][0]+1,t[1][1]]])

record[t[1][0]+1][t[1][1]]=1

if t[1][1]+1<column and record[t[1][0]][t[1][1]+1]==0:

a.append([matrix[t[1][0]][t[1][1]+1],[t[1][0],t[1][1]+1]])

record[t[1][0]][t[1][1]+1]=1

91, 11, 21, 1211, 111221, ...

1 读作 "one 1" -> 11.

11 读作 "two 1s" -> 21.

21 读作 "one 2, then one 1" -> 1211.

给定一个整数 n, 返回 第 n 个顺序。

 注意事项

整数的顺序将表示为一个字符串。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给定 n = 5, 返回 "111221".

思路：做过

class Solution:

# @param {int} n the nth

# @return {string} the nth sequence

def countAndSay(self, n):

# Write your code here

a=["1","11"]

for i in range(1,n):

temp=[]

t=1

for j in range(len(a[i])-1):

if a[i][j]==a[i][j+1]:

t+=1

else:

temp.append(t)

temp.append(a[i][j])

t=1

# print(temp)

if j+1==len(a[i])-1:

temp.append(t)

temp.append(int(a[i][j+1]))

a.append(temp)

b=""

for i in range(len(a[n-1])):

b+=str(a[n-1][i])

return b

#### 10. 罗马数字转整数

* [描述](http://www.lintcode.com/zh-cn/problem/roman-to-integer/#description)
* [笔记](http://www.lintcode.com/zh-cn/problem/roman-to-integer/#note)
* [数据](http://www.lintcode.com/zh-cn/problem/roman-to-integer/#testcase)
* [评测](http://www.lintcode.com/zh-cn/problem/roman-to-integer/#judge)

给定一个罗马数字，将其转换成整数。

返回的结果要求在1到3999的范围内。

您在真实的面试中是否遇到过这个题？

Yes

**说明**

什么是 *罗马数字*?

* <https://en.wikipedia.org/wiki/Roman_numerals>
* <https://zh.wikipedia.org/wiki/%E7%BD%97%E9%A9%AC%E6%95%B0%E5%AD%97>
* <http://baike.baidu.com/view/42061.htm>

**样例**

IV -> 4

XII -> 12

XXI -> 21

XCIX -> 99

思路：就是知道罗马的规则，别的没啥办法。观察是观察不出来的。最是小于减大于加最后一个没法比较直接加。

class Solution:

# @param {string} s Roman representation

# @return {int} an integer

def romanToInt(self, s):

# Write your code here

dict={'I':1,'V':5,'X':10,'L':50,'C':100,'D':500,'M':1000}

length=len(s)

if length<2:

return dict[s]

sum=0

for i in range(length-1):

if dict[s[i]]>=dict[s[i+1]]:

sum+=dict[s[i]]

else:

sum-=dict[s[i]]

sum+=dict[s[i+1]]

return sum