给定一个整数，将其转换成罗马数字。

返回的结果要求在1-3999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**

4 -> IV

12 -> XII

21 -> XXI

99 -> XCIX

更多案例，请戳 <http://literacy.kent.edu/Minigrants/Cinci/romanchart.htm>

思路：有些是特殊的要做成dict，其余的按规则走。罗马数字都是加法做做出来的，因此要把所有基本单元都做出来，然后从大到小排序，这样能整除几个就有几个，然后余数接着做，直到==0，要是都遍历还是麻烦一点。不过也不会差太多就复杂度上说。

class Solution:

# @param {int} n The integer

# @return {string} Roman representation

def intToRoman(self, n):

# Write your code here

dict={1:'I',2:'II',3:'III',4:'IV',5:'V',6:'VI',7:'VII',8:'VIII',9:'IX',10:'X',\

50:'L',100:'C',500:'D',1000:'M',99:'XCIX',999:'CMXCIX'}

t=sorted(dict.items(),key=lambda x:x[0],reverse=True)

dict1={}

a=''

for i in t:

dict1[i[1]]=n//i[0]

n=n%i[0]

for key in t:

if dict1[key[1]]>0:

a+=key[1]\*dict1[key[1]]

return a

2. 给定一个字符串，验证其是否为数字。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

"0" => true

" 0.1 " => true

"abc" => false

"1 a" => false

"2e10" => true

思路验证字符串是否是数字。那么数字有好多种整形，小数，科学技术法，假如我可以把数值都换成float那么就是数字，否则就不是

class Solution:

# @param {string} s the string that represents a number

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

def isNumber(self, s):

# Write your code here

t=s.strip()

try:

float(t)

except (ValueError):

return False

else:

return True

3. 给定一个字符串，判断其是否为一个回文串。只包含字母和数字，忽略大小写。

 注意事项

你是否考虑过，字符串有可能是空字符串？这是面试过程中，面试官常常会问的问题。

在这个题目中，我们将空字符串判定为有效回文。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

"A man, a plan, a canal: Panama" 是一个回文。

"race a car" 不是一个回文。

思路把字符串变成list只取字母和数字，并全部转换为小写，因此就是i++，j—比较出口为不同或者i>j这时return true

class Solution:

# @param {string} s A string

# @return {boolean} Whether the string is a valid palindrome

def isPalindrome(self, s):

# Write your code here

'''ç¬¬ä¸ç§æ¹æ³

import string

s.strip()

s=s.lower()

for i in s:

if i in string.punctuation:

s=s.replace(i, '')

s=s.replace(' ', '')

if len(s)==0:

return True

for i in range(len(s)):

if s[i]!=s[len(s)-i-1]:

return False

return True

'''

h='abcdefghijklmnopqrstuvwxyz0123456789'

s=s.lower()

for i in s:

if i not in h:

s=s.replace(i, '')

if len(s)==0:

return True

for i in range(len(s)//2+1):

if s[i]!=s[len(s)-i-1]:

return False

return True

4.

给定一个由 n 个整数组成的数组和一个正整数 s ，请找出该数组中满足其和 ≥ s 的最小长度子数组。如果无解，则返回 -1。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给定数组 [2,3,1,2,4,3] 和 s = 7, 子数组 [4,3] 是该条件下的最小长度子数组。

思路找吧，子数组意味着连续，所以直接遍历很多，做个前向和这样就是n2个。能不能在降低，好像不太可行，因为我这里没有多余计算

class Solution:

# @param nums: a list of integers

# @param s: an integer

# @return: an integer representing the minimum size of subarray

def minimumSize(self, nums, s):

# write your code here

'''

#算法复杂度是O(N\*2)

length=len(nums)

if length==0:

return -1

if length==1:

if nums[0]==s:

return 1

else:

return -1

sum=0

min=8888

for i in range(length):

sum+=nums[i]

nums[i]=sum

nums.insert(0,0)

for i in range(length+1):

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

if nums[j]-nums[i]>= s:

leng=j-i

if min>leng:

min=leng

break

if min==8888:

return -1

return min

#算法复杂度在O（n）

'''

length=len(nums)

if length==0:

return -1

if length==1:

if nums[0]==s:

return 1

else:

return -1

sum=0

temp=8888

start=0

i=0

while i <length:

sum+=nums[i]

if sum>=s:

temp=min(temp,i-start+1)

while sum>=s and start<=i:

temp=min(temp,i-start+1)

sum-=nums[start]

start+=1

i+=1

if temp==8888:

return -1

return temp

5. 给定一个整数数组（下标从 0 到 n-1， n 表示整个数组的规模），请找出该数组中的最长上升连续子序列。（最长上升连续子序列可以定义为从右到左或从左到右的序列。）

 注意事项

time

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给定 [5, 4, 2, 1, 3], 其最长上升连续子序列（LICS）为 [5, 4, 2, 1], 返回 4.

给定 [5, 1, 2, 3, 4], 其最长上升连续子序列（LICS）为 [1, 2, 3, 4], 返回 4.

思路要是左右都遍历一遍是不是麻烦点，要是只遍历一次怎么想就是由大变小或者由小变大都要重新计数。就是flag变化。Flag不变就加一变就变成1

class Solution:

# @param {int[]} A an array of Integer

# @return {int} an integer

def longestIncreasingContinuousSubsequence(self, A):

# Write your code here

'''

#耗时1022ms

length=len(A)

if length==0:

return 0

if length==1:

return 1

a=[]

for i in range(length-1):

if A[i]>A[i+1]:

a.append(1)

else:

a.append(0)

b=[]

max1=2

if len(a)==1:

return 2

for i in range(len(a)-1):

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

max1+=1

b.append(max1)

else:

max1=2

return int(max(b))

'''

#第二种方法耗时816ms

length=len(A)

if length==0:

return 0

if length==1:

return 1

a=[]

t=1

if A[0]>=A[1]:

pre=1

else:

pre=0

max1=0

for i in range(length-1):

if A[i]>=A[i+1] and pre==1:

pre=1

t+=1

elif A[i]<A[i+1] and pre==0:

pre=0

t+=1

elif A[i]<A[i+1] and pre==1:

t=2

pre=0

else:

t=2

pre=1

if t>max1:

max1=t

return max1

6. 分割一个整数数组，使得奇数在前偶数在后。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给定 [1, 2, 3, 4]，返回 [1, 3, 2, 4]。

思路：分类把基数和偶数分类，这样就得到了想要的，不过要多出来两个内容，能不能在原数组直接做，那就删除了然后加上去吧

class Solution:

# @param nums: a list of integers

# @return: nothing

def partitionArray(self, nums):

# write your code here

length=len(nums)

i=j=0

while j<length:

if nums[i]&1!=0:

j+=1

i+=1

else:

t=nums[i]

del(nums[i])

nums.append(t)

j+=1

return nums

#### 7. 最长回文子串

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

给出一个字符串（假设长度最长为1000），求出它的最长回文子串，你可以假定只有一个满足条件的最长回文串。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给出字符串 "abcdzdcab"，它的最长回文子串为 "cdzdc"

暴力解决，把字符串划分成不同的窗口然后判断窗口是否为回文先从大到小，得到就返回。在外部做一个可以判断是否是回文字符串的。

时间复杂度较高，看看有什么可以改进的，总耗时5053ms，好像没有。

class Solution:

# @param {string} s input string

# @return {string} the longest palindromic substring

def longestPalindrome(self, s):

# Write your code here

'''暴力枚举算法时间太高为立方形式

length=len(s)

b=[]

for j in range(length):

for h in range(j,length):

temp=s[j:h+1]

lentemp=len(temp)

a=[]

lentemp1=lentemp

for k in range(lentemp):

if temp[k]!=temp[lentemp-1-k]:

lentemp1=-1

#print(lentemp1,temp)

b.append([lentemp1,temp])

c=sorted(b,key=lambda x:x[0])

return c[-1][1]

'''

#动态规划去处理'''

length=len(s)

maxlength=0

leng=3

if length==1:

return s

elif length==2:

if s[0]==s[-1]:

return s

return s[0]

b=[[False for col in range(length)]for row in range(length)]

for i in range(length):

b[i][i]=True

if i<length-1 and s[i]==s[i+1]:

b[i][i+1]=True

start=i

maxlength=2

while leng<=length:

j=0

while j <=length-leng :

end=j+leng-1

if b[j+1][end-1] and s[j]==s[end]:

b[j][end]=True

maxlength=leng

start=j

j+=1

leng+=1

return s[start:start+maxlength]

'''

length=len(s)

for i in range(length, 0, -1):

for j in range(length):

if i+j<=length:

temp=s[j:i+j]

if panduan(temp):

return temp

def panduan(temp):

if temp==temp[::-1]:

return True

return False

'''

8. 给定一个若干整数的排列，给出按正数大小进行字典序从小到大排序后的下一个排列。

如果没有下一个排列，则输出字典序最小的序列。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

左边是原始排列，右边是对应的下一个排列。

1,2,3 → 1,3,2

3,2,1 → 1,2,3

1,1,5 → 1,5,1

下一个排列就是大的和他前一个小的交换一下位置即可。要是都时最后一个，最推出第一个。然后后面的要排序，排成最小的。

class Solution:

# @param nums: a list of integer

# @return: return nothing (void), do not return anything, modify nums in-place instead

def nextPermutation(self, nums):

# write your code here

length=len(nums)

if length==1:

return nums

if length==0:

return None

t=length

h=1

a=[]

if nums[length-2]<nums[length-1]:

nums[length-2],nums[length-1]=nums[length-1],nums[length-2]

else:

while nums[length-2]>=nums[length-1] and h<t:

a.append(nums[length-1])

length-=1

h+=1

a.append(nums[length-1])

for i in range(len(a)):

if a[i]>nums[length-2]:

nums[length-2],nums[t-1-i]=nums[t-1-i],nums[length-2]

nums[length-1:t]=sorted(nums[length-1:t])

break

if h==t:

nums.sort()

9. 给出若干闭合区间，合并所有重叠的部分。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给出的区间列表 => 合并后的区间列表：

[ [

[1, 3], [1, 6],

[2, 6], => [8, 10],

[8, 10], [15, 18]

[15, 18] ]

]

思路两个两个比较若有重叠最小最大即可。找不到就加一找到合并后还要作为下一次的更改对象重叠判断方式为前一个最大值比后一个最小值大，假设是有序的，然后最小最大在删掉一个。

"""

Definition of Interval.

class Interval(object):

def \_\_init\_\_(self, start, end):

self.start = start

self.end = end

"""

class Solution:

# @param intervals, a list of Interval

# @return a list of Interval

def merge(self, intervals):

# write your code here

intervals=sorted(intervals,key=lambda x:x.start)

# return intervals

a=getqujian(intervals=intervals,i=0)

return intervals

#for i in range(len(a)):

#print(a[i].start)

def getqujian(intervals,i):

if i>=len(intervals)-1:

#print(intervals)

return 1

if intervals[i].end >= intervals[i+1].start and intervals[i].end<=intervals[i+1].end:

intervals[i].end=intervals[i+1].end

del(intervals[i+1])

elif intervals[i].end>= intervals[i+1].start and intervals[i].end>intervals[i+1].end:

del(intervals[i+1])

else:

i+=1

getqujian(intervals,i)

10. 计算在一个 32 位的整数的二进制表示中有多少个 1.

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给定 32 (100000)，返回 1

给定 5 (101)，返回 2

给定 1023 (111111111)，返回 9

思路移位操作与吧

class Solution:

# @param num: an integer

# @return: an integer, the number of ones in num

def countOnes(self, num):

# write your code here

sum=0

t=0

while t<32 and num!=0:

h=num&1

sum+=h

num=num>>1

t+=1

return sum