1. 给出一个整数数组 *nums* 和一个整数 *k*。划分数组（即移动数组 *nums* 中的元素），使得：

* 所有小于k的元素移到左边
* 所有大于等于k的元素移到右边

返回数组划分的位置，即数组中第一个位置 *i*，满足 *nums*[*i*] 大于等于 *k*。

 注意事项

你应该真正的划分数组 *nums*，而不仅仅只是计算比 *k* 小的整数数，如果数组 *nums* 中的所有元素都比 *k* 小，则返回 nums.length。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给出数组 nums = [3,2,2,1] 和 k = 2，返回 1.

思路：就是快速排序的一种变形，然后能找到第一个大于等于改数值的位置。不如先插入到开始，然后运行算法，最后在删除这个点。

class Solution:

"""

@param nums: The integer array you should partition

@param k: As description

@return: The index after partition

"""

def partitionArray(self, nums, k):

# write your code here

# you should partition the nums by k

# and return the partition index as description

length=len(nums)

if length==0:

return 0

nums.append(k)

nums[-1],nums[0]=nums[0],nums[-1]

high=length;low=0;x=nums[0]

while low<high:

while low < high and nums[high] >= x:

high-=1

nums[low] = nums[high]

while low < high and nums[low] <x:

low+=1

nums[high] = nums[low]

nums[low] = x

return low

2. 给定一个字符串**source**和一个目标字符串**target**，在字符串**source**中找到包括所有目标字符串字母的子串。要求是最小的一个。

 注意事项

如果在**source**中没有这样的子串，返回""，如果有多个这样的子串，返回起始位置最小的子串。

您在真实的面试中是否遇到过这个题？

Yes

**说明**

在答案的子串中的字母在目标字符串中是否需要具有相同的顺序？

——不需要。

**样例**

给出source = **"ADOBECODEBANC"**，target = **"ABC"** 满足要求的解  **"BANC"**

思路：加入做个二维矩阵用i、j来表示，i到j中是否包含目标，然后在包含的里面找出距离最小的一个。原则上是可行的。怎么计算包含目标，加入i，j里包含了至少一遍target里面的字符，比较复杂，而且常用，不妨把他放到函数里面。统计吧算法复杂是o（n2）（试验了3次还是有不行的地方，要争取直接通过）

class Solution:

"""

@param source: A string

@param target: A string

@return: A string denote the minimum window

Return "" if there is no such a string

"""

def minWindow(self, source, target):

# write your code here

#思路没问题需要空间太大

dict={}

if len(source)==0 and len(target)==0:

return ""

if len(source)==0:

return ""

if len(target)==0:

return ""

minx=8888

temp1=0

temp2=0

flag=0

for i in range(len(source)):

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

if get(source,i,j,target):

if minx>j-i:

minx=j-i

temp1=i

temp2=j

flag=1

if flag:

return source[temp1:temp2+1]

else:

return ""

def get(source,i,j,target):

p=i

dict1={}

dict2={}

while i <=j:

dict1[source[i]]=dict1.get(source[i],0)+1

i+=1

for k in target:

dict2[k]=dict2.get(k,0)+1

for m in dict2.keys():

if m in dict1.keys():

if dict2[m]<=dict1[m]:

pass

else:

return False

else:

return False

return True

3. n皇后问题是将n个皇后放置在n\*n的棋盘上，皇后彼此之间不能相互攻击。

给定一个整数n，返回所有不同的n皇后问题的解决方案。

每个解决方案包含一个明确的n皇后放置布局，其中“Q”和“.”分别表示一个女王和一个空位置。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

对于4皇后问题存在两种解决的方案：

[

    [".Q..", // Solution 1

     "...Q",

     "Q...",

     "..Q."],

    ["..Q.", // Solution 2

     "Q...",

     "...Q",

     ".Q.."]

]

思路：深度搜索吧，同一行、同一列、同一斜线上的皇后都会自动攻击，这样需要设定一个行判断，列判断，斜线判断的程序。然后深度搜索，保存结果。

超时，主要问题时循环递归时出现问题因为循环递归只能在尾部才能做出判断，因为若是直接返回，会返回到要想出一个可以在任意时刻进行判断的。

class Solution:

"""

Get all distinct N-Queen solutions

@param n: The number of queens

@return: All distinct solutions

"""

def solveNQueens(self, n):

# write your code here

temp=[0 for i in range(0,n)]

temp1=[0 for i in range(0,n)]

temp2=[0 for i in range(0,n)]

temp3=[0 for i in range(0,n)]

temp5=[]

a=[['.' for i in range(n)] for j in range(n)]

i=0

j=0

get(i,temp,temp1,temp2,temp3,a,n,temp5,j)

return temp5

def get(i,temp,temp1,temp2,temp3,a,n,temp5,j):

if i==n:

if len(set(temp))==n:

if len(set(temp1))==n:

if len(set(temp2))==n:

if len(set(temp3))==n:

b=[]

for m in range(n):

str1=''

for h in range(n):

str1+=a[m][h]

b.append(str1)

temp5.append(b)

return

if i==n:

return

for k in range(n):

j=k

a[i][j]='Q'

temp[i]=i

temp1[i]=j

temp2[i]=i+j

temp3[i]=i-j+n

get(i+1,temp,temp1,temp2,temp3,a,n,temp5,j)

a[i][j]='.'

4. 根据n皇后问题，现在返回n皇后不同的解决方案的数量而不是具体的放置布局。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

比如n=4，存在2种解决方案

思路和上一题一样就是时间复杂度大了点。

class Solution:

"""

Get all distinct N-Queen solutions

@param n: The number of queens

@return: All distinct solutions

"""

def solveNQueens(self, n):

# write your code here

temp=[0 for i in range(0,n)]

temp1=[0 for i in range(0,n)]

temp2=[0 for i in range(0,n)]

temp3=[0 for i in range(0,n)]

temp5=[]

a=[['.' for i in range(n)] for j in range(n)]

i=0

j=0

get(i,temp,temp1,temp2,temp3,a,n,temp5,j)

return temp5

def get(i,temp,temp1,temp2,temp3,a,n,temp5,j):

if i==n:

if len(set(temp))==n:

if len(set(temp1))==n:

if len(set(temp2))==n:

if len(set(temp3))==n:

b=[]

for m in range(n):

str1=''

for h in range(n):

str1+=a[m][h]

b.append(str1)

temp5.append(b)

return

if i==n:

return

for k in range(n):

j=k

a[i][j]='Q'

temp[i]=i

temp1[i]=j

temp2[i]=i+j

temp3[i]=i-j+n

get(i+1,temp,temp1,temp2,temp3,a,n,temp5,j)

a[i][j]='.'

5.

翻转一个链表

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给出一个链表**1->2->3->null**，这个翻转后的链表为**3->2->1->null**

思路：做个list然后倒叙给

"""

Definition of ListNode

class ListNode(object):

def \_\_init\_\_(self, val, next=None):

self.val = val

self.next = next

"""

class Solution:

"""

@param head: The first node of the linked list.

@return: You should return the head of the reversed linked list.

Reverse it in-place.

"""

def reverse(self, head):

# write your code here

a=[]

while head!=None:

a.append(head.val)

head=head.next

a=a[::-1]

if len(a)==0:

return None

headcopy=ListNode(a[0])

t=createnode(headcopy,a[1:])

return t

def createnode(head,a):

if len(a)>0:

head.next=ListNode(a[0])

createnode(head.next,a[1:])

return head

6. 翻转链表中第m个节点到第n个节点的部分

 注意事项

m，n满足1 ≤ m ≤ n ≤ 链表长度

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给出链表**1->2->3->4->5->null**， m = 2 和n = 4，返回**1->4->3->2->5->null**

**思路：和上体一样，就是我需要遍历次数多点。不过最好还是更改值，要保存需要的节点。"""**

**Definition of ListNode**

**class ListNode(object):**

**def \_\_init\_\_(self, val, next=None):**

**self.val = val**

**self.next = next**

**"""**

**class Solution:**

**"""**

**@param head: The head of linked list**

**@param m: start position**

**@param n: end position**

**"""**

**def reverseBetween(self, head, m, n):**

**# write your code here**

**self.b=[]**

**self.m=m**

**self.n=n**

**self.getnumbers(head)**

**while self.m<=self.n:**

**self.b[self.m-1],self.b[self.n-1]=self.b[self.n-1],self.b[self.m-1]**

**self.m+=1**

**self.n-=1**

**self.m=m**

**self.n=n**

**print(self.b)**

**temp=head**

**k=0**

**while k<self.m-1:**

**k+=1**

**head=head.next**

**while self.m<=self.n:**

**head.val=self.b[self.m-1]**

**print(head.val)**

**head=head.next**

**self.m+=1**

**return temp**

**def getnumbers(self,head):**

**if head==None:**

**return**

**else:**

**self.b.append(head.val)**

**return self.getnumbers(head.next)**

**7.** 写出一个高效的算法来搜索m×n矩阵中的值，返回这个值出现的次数。

这个矩阵具有以下特性：

* 每行中的整数从左到右是排序的。
* 每一列的整数从上到下是排序的。
* 在每一行或每一列中没有重复的整数。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

考虑下列矩阵：

**[**

**[1, 3, 5, 7],**

**[2, 4, 7, 8],**

**[3, 5, 9, 10]**

**]**

给出target = **3**，返回**2**

[**挑战**](http://www.lintcode.com/zh-cn/problem/search-a-2d-matrix-ii/#challenge)

**思路：高笑算法往往比较复杂，不容易想，不过无非还是二分吧。先找到列，这个列不是一个而是一半，然后是行，也是一个个来的。没办法。**

**class Solution:**

**"""**

**@param matrix: An list of lists of integers**

**@param target: An integer you want to search in matrix**

**@return: An integer indicates the total occurrence of target in the given matrix**

**"""**

**def searchMatrix(self, matrix, target):**

**# write your code here**

**# cost**

**m=len(matrix)**

**count=0**

**for i in range(m):**

**if matrix[i][0]<=target:**

**for j in range(len(matrix[i])):**

**if matrix[i][j]==target:**

**count+=1**

**break**

**if matrix[i][j]>target:**

**break**

**return count**

**'''**

**#cost**

**m=len(matrix)**

**n=len(matrix[0])**

**count=0**

**for i in range(n-1,-1,-1):**

**if matrix[0][i]>target:**

**pass**

**elif matrix[0][i]==target:**

**count+=1**

**else:**

**resn=i**

**break**

**for i in range(m-1,-1,-1):**

**if matrix[i][0]>target:**

**pass**

**elif matrix[i][0]==target:**

**count+=1**

**else:**

**resm=i**

**break**

**for k in range(resm+1):**

**for h in range(resn+1):**

**if matrix[k][h]==target:**

**count+=1**

**return count**

**'''**

**8.**

给定一个**旋转**排序数组，在原地恢复其排序。

您在真实的面试中是否遇到过这个题？

Yes

**说明**

什么是旋转数组？

* 比如，原始数组为[1,2,3,4], 则其旋转数组可以是[1,2,3,4], [2,3,4,1], [3,4,1,2], [4,1,2,3]

**样例**

[4, 5, 1, 2, 3] -> [1, 2, 3, 4, 5]

思路找到改变值，然后删除了在append就行，不过最好直接拍下多省事。

class Solution:

"""

@param nums: The rotated sorted array

@return: nothing

"""

def recoverRotatedSortedArray(self, nums):

# write your code here

nums.sort()

return nums

9. 给定一个整数数组，找到一个具有最大和的子数组，返回其最大和。

 注意事项

子数组最少包含一个数

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给出数组[−2,2,−3,4,−1,2,1,−5,3]，符合要求的子数组为[4,−1,2,1]，其最大和为6

思路：一直加吧，小于0就舍弃，重新开始，不然就一直加，贪心求最大值。

def maxSubArray(self, nums):

# write your code here

length=len(nums)

max=-8888

sum=0

if length==1:

return nums[0]

for i in range(length):

sum+=nums[i]

if sum>max:

max=sum

if sum<0:

sum=0

return max

10.

给定一个整数数组，找出两个 *不重叠* 子数组使得它们的和最大。  
每个子数组的数字在数组中的位置应该是连续的。  
返回最大的和。

 注意事项

子数组最少包含一个数

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给出数组 [1, 3, -1, 2, -1, 2]  
这两个子数组分别为 [1, 3] 和 [2, -1, 2] 或者 [1, 3, -1, 2] 和 [2]，它们的最大和都是 7

思路：两个子数组的最大和，及分割下求每个区间内的最大值，只要分割下，然后套用上一题就行。

class Solution:

"""

@param nums: A list of integers

@return: An integer denotes the sum of max two non-overlapping subarrays

"""

def maxTwoSubArrays(self, nums):

# write your code here

length1=len(nums)

if length1==1 or length1==0:

return 0

arr=nums

sum1=[0 for row in range(len(arr)-1)]

sum2=[0 for colum in range(len(arr)-1)]

sum = arr[0]

sum1[0] = sum

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

if (sum <= 0):

sum = arr[i]

else:

sum += arr[i]

if (sum1[i-1] < sum):

sum1[i] = sum

else:

sum1[i] = sum1[i-1]

sum = arr[-1]

sum2[-1] = sum

for h in range(length-2,0,-1):

if (sum <= 0):

sum = arr[h]

else:

sum += arr[h]

if (sum2[h] < sum):

sum2[h-1] = sum

else:

sum2[h-1] = sum2[h]

max1 = sum1[0] + sum2[0]

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

if max1 < sum1[i] + sum2[i]:

max1 = sum1[i] + sum2[i]

return max1