给一组整数，按照升序排序。使用归并排序，快速排序，堆排序或者任何其他 O(*n* log *n*) 的排序算法。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给出 [3, 2, 1, 4, 5], 排序后的结果为 [1, 2, 3, 4, 5]。

[**标签**](http://www.lintcode.com/zh-cn/problem/sort-integers-ii/#tags)

[**相关题目**](http://www.lintcode.com/zh-cn/problem/sort-integers-ii/#related)

思路：快速排序主要是拍一遍，分割，然后在排，相当于二分法应用。归并排序相当于快速排序的逆过程。也是二分法应用。堆排我理解为树的形式，即根节点是最大的或者是最小的，这样比较的话可以得到。

class Solution:

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

# @return nothing

def sortIntegers2(self, A):

# Write your code here

'''quick sort cost 1937ms

length=len(A)

if length==0:

return None

quicksort(A,left=0,right=length-1)

def quicksort(A,left,right):

low=left

high=right

if low < high:

key=A[low]

while low<high:

while low < high and A[high]>key:

high-=1

A[low]=A[high]

while A[low]<=key and low < high:

low+=1

A[high]=A[low]

A[low]=key

quicksort(A,left,low-1)

quicksort(A,low+1,right)

'''

#归并排序

if len(A)<=1:

return A

mid=len(A)//2

B=A

left=self.sortIntegers2(B[:mid])

right=self.sortIntegers2(B[mid:])

return merge(left,right,A)

def merge(a,b,A):

j=0

i=0

k=0

while j<len(b) and i<len(a):

if b[j]>a[i]:

A[k]=a[i]

i+=1

else:

A[k]=b[j]

j+=1

k+=1

if j<len(b):

for j in range(j,len(b)):

A[k]=b[j]

k+=1

if i<len(a):

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

A[k]=a[i]

k+=1

return A[:k]

#推排序

思路首先递归创建最大堆把最大值放在堆首。创建时要从底部创建，那么地步根应该是长度的一半。然后依次与首字母交换在递归创建即可。

def adjust\_heap(lists, i, size):

    lchild = 2 \* i + 1

    rchild = 2 \* i + 2

    max = i

    if i < size / 2:

        if lchild < size and lists[lchild] > lists[max]:

            max = lchild

        if rchild < size and lists[rchild] > lists[max]:

            max = rchild

        if max != i:

            lists[max], lists[i] = lists[i], lists[max]

            adjust\_heap(lists, max, size)

def build\_heap(lists, size):

    for i in range(0, (size/2))[::-1]:

        adjust\_heap(lists, i, size)

def heap\_sort(lists):

    size = len(lists)

    build\_heap(lists, size)

    for i in range(0, size)[::-1]:

        lists[0], lists[i] = lists[i], lists[0]

        adjust\_heap(lists, 0, i)

设计一种方法，将一个字符串中的所有空格替换成 %20 。你可以假设该字符串有足够的空间来加入新的字符，且你得到的是“真实的”字符长度。

你的程序还需要返回被替换后的字符串的长度。

 注意事项

如果使用 Java 或 Python, 程序中请用字符数组表示字符串。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

对于字符串"Mr John Smith", 长度为 13

替换空格之后，参数中的字符串需要变为"Mr%20John%20Smith"，并且把新长度 17 作为结果返回。

思路：遍历字符串，找到空格，然后替换，并且计数最后在长度上加上二倍的空格值

class Solution:

# @param {char[]} string: An array of Char

# @param {int} length: The true length of the string

# @return {int} The true length of new string

def replaceBlank(self, string, length):

# Write your code here

if length==0:

return 0

for i in range(len(string)):

if string[i]==' ':

string[i]='%20'

i+=1

return i+1

3. 实现 int sqrt(int x) 函数，计算并返回 *x* 的平方根。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

sqrt(3) = 1

sqrt(4) = 2

sqrt(5) = 2

sqrt(10) = 3

利用牛顿下降法来逼近真值。Sqrt看成x2-y的解。那么x1=x0-(x02-y)/2x0另x0=y即可。精度可控制

class Solution:

"""

@param x: An integer

@return: The sqrt of x

"""

def sqrt(self, x):

# write your code here

if x==0:

return 0

val = x

last=0

eps=0.001

while abs(val-last) > eps:

last = val

val =(val + x/val) / 2

return int(val)

4给定两个字符串，请设计一个方法来判定其中一个字符串是否为另一个字符串的置换。

置换的意思是，通过改变顺序可以使得两个字符串相等。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

"abc" 为 "cba" 的置换。

"aabc" 不是 "abcc" 的置换。

思路统计每个字符串的字母出现的次数，然后比较即可。

class Solution:

# @param {string} A a string

# @param {string} B a string

# @return {boolean} a boolean

def stringPermutation(self, A, B):

# Write your code here

Alength=len(A)

Blength=len(B)

Adict={}

Bdict={}

if Alength!=Blength:

return False

for i in range(Alength):

Adict[A[i]]=Adict.get(A[i],0)+1

for j in range(Blength):

Bdict[B[j]]=Bdict.get(B[j],0)+1

for i in Adict:

if Adict[i]!=Bdict[i]:

return False

return True

5. 给定一个整数数组，找到和为零的子数组。你的代码应该返回满足要求的子数组的起始位置和结束位置

 注意事项

There is at least one subarray that it's sum equals to zero.

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给出 [-3, 1, 2, -3, 4]，返回[0, 2] 或者 [1, 3].

思路：做数组一个个来，遍历的复杂度很高n！，要是做前n项和那么可以把复杂度降到n2。

class Solution:

"""

@param nums: A list of integers

@return: A list of integers includes the index of the first number

and the index of the last number

"""

def subarraySum(self, nums):

# write your code here

a=[0]

temp=0

for i in range(len(nums)):

temp+=nums[i]

a.append(temp)

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

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

if a[i]-a[j]==0:

return [j,i-1]

6. 有两个不同大小的二进制树： T1 有上百万的节点； T2 有好几百的节点。请设计一种算法，判定 T2 是否为 T1的子树。

 注意事项

若 T1 中存在从节点 n 开始的子树与 T2 相同，我们称 T2 是 T1 的子树。也就是说，如果在 T1 节点 n 处将树砍断，砍断的部分将与 T2 完全相同。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

下面的例子中 T2 是 T1 的子树：

1 3

/ \ /

T1 = 2 3 T2 = 4

/

4

下面的例子中 T2 不是 T1 的子树：

1 3

/ \ \

T1 = 2 3 T2 = 4

/

4

思路：找到判断的入口，即相等的节点，然后递归判断就行，相等节点可能不止一个。递归查找。判断的递归时要单做即满足要求。即空都是空，不空是节点值相同。这些都要递归，否则可以返回了。

def \_\_init\_\_(self, val):

self.val = val

self.left, self.right = None, None

"""

class Solution:

# @param T1, T2: The roots of binary tree.

# @return: True if T2 is a subtree of T1, or false.

def isSubtree(self, T1, T2):

# write your code here

if T2==None:

return True

elif T1==None:

return False

t=[]

h=[]

findT2(T1,T2,t)

for i in range(len(t)):

b=[]

chenck(t[i],T2,b)

if len(b)==0:

return True

return False

def chenck(T,T2,b):

if T!=None and T2!=None:

if T.val==T2.val:

chenck(T.left,T2.left,b)

chenck(T.right,T2.right,b)

else:

b.append(0)

return

elif T==None and T2==None :

pass

else:

b.append(0)

return

def findT2(T1,T2,t):

if T1!=None and T2!=None:

if T1.val!=T2.val:

pass

else:

t.append(T1)

findT2(T1.left,T2,t)

findT2(T1.right,T2,t)

7. 给一个链表，两两交换其中的节点，然后返回交换后的链表。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给出 1->2->3->4, 你应该返回的链表是 2->1->4->3。

思路：交换两个节点值然后next移步两个。

# Definition for singly-linked list.

# class ListNode:

# def \_\_init\_\_(self, x):

# self.val = x

# self.next = None

class Solution:

# @param head, a ListNode

# @return a ListNode

def swapPairs(self, head):

# Write your code here

s=head

while s !=None and s.next!=None:

s.val,s.next.val=s.next.val,s.val

s=s.next.next

return head

8．设计一个算法，计算出n阶乘中尾部零的个数

您在真实的面试中是否遇到过这个题？

Yes

**样例**

**11! = 39916800**，因此应该返回 2

等于0的乘法，即和5相关，出现一个5的倍数就有1个，但25怎么办，出现一个25也有一个。

class Solution:

"""

@param: : An integer

@return: An integer, denote the number of trailing zeros in n!

"""

def trailingZeros(self, n):

# write your code here, try to do it without arithmetic operators.

i=1

count=0

while n>=5\*\*i:

count+=n//(5\*\*i)

i+=1

return count

9. 给定一个数字三角形，找到从顶部到底部的最小路径和。每一步可以移动到下面一行的相邻数字上。

 注意事项

如果你只用额外空间复杂度O(n)的条件下完成可以获得加分，其中n是数字三角形的总行数。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

比如，给出下列数字三角形：

[

[2],

[3,4],

[6,5,7],

[4,1,8,3]

]

从顶到底部的最小路径和为11 ( 2 + 3 + 5 + 1 = 11)。

思路：深度优先搜索可以做。不过只要求综合直接相加更方便。这应该是动态规划了。Temp[i][j]=min(temp[i-1][j-1],temp[i-1][j])+temp[i][j],边界条件是超出范围。最后在做最后一排的min。不过好像可以倒着做。这样最后就剩下一个。

def minimumTotal(self, triangle):

# write your code here

length=len(triangle)

triangle.append([0]\*(length+1))

#print(triangle)

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

for j in range(len(triangle[i])):

triangle[i][j]=min(triangle[i][j]+triangle[i+1][j],triangle[i][j]+triangle[i+1][j+1])

return triangle[0][0]

10. 写出一个函数 anagram(s, t) 判断两个字符串是否可以通过改变字母的顺序变成一样的字符串。

您在真实的面试中是否遇到过这个题？

Yes

**说明**

What is **Anagram**?  
- Two strings are anagram if they can be the same after change the order of characters.

**样例**

给出 s = "abcd"，t="dcab"，返回 true.  
给出 s = "ab", t = "ab", 返回 true.  
给出 s = "ab", t = "ac", 返回 false.

思路：通过改变字符相等意味着个数相同，统计量相同。

class Solution:

"""

@param s: The first string

@param b: The second string

@return true or false

"""

def anagram(self, s, t):

# write your code here

A=s

B=t

Alength=len(A)

Blength=len(B)

Adict={}

Bdict={}

if Alength!=Blength:

return False

for i in range(Alength):

Adict[A[i]]=Adict.get(A[i],0)+1

for j in range(Blength):

Bdict[B[j]]=Bdict.get(B[j],0)+1

for i in Adict:

if i in Bdict:

if Adict[i]!=Bdict[i]:

return False

else:

return False

return True