在数据结构中，哈希函数是用来将一个字符串（或任何其他类型）转化为小于哈希表大小且大于等于零的整数。一个好的哈希函数可以尽可能少地产生冲突。一种广泛使用的哈希函数算法是使用数值33，假设任何字符串都是基于33的一个大整数，比如：

hashcode("abcd") = (ascii(a) \* 333 + ascii(b) \* 332 + ascii(c) \*33 + ascii(d)) % HASH\_SIZE

                              = (97\* 333 + 98 \* 332 + 99 \* 33 +100) % HASH\_SIZE

                              = 3595978 % HASH\_SIZE

其中HASH\_SIZE表示哈希表的大小(可以假设一个哈希表就是一个索引0 ~ HASH\_SIZE-1的数组)。

给出一个字符串作为key和一个哈希表的大小，返回这个字符串的哈希值。

您在真实的面试中是否遇到过这个题？

Yes

**说明**

For this problem, you are not necessary to design your own hash algorithm or consider any collision issue, you just need to implement the algorithm as described.

**样例**

对于key="abcd" 并且 size=100， 返回 78

思路：就是做出字符串的二进制数，然后利用公式求解就行，唯一缺陷就是算法复杂度很高，数值越小计算难度也会越小。因此有必要把求雨拿到里面来。

class Solution:

"""

@param key: A String you should hash

@param HASH\_SIZE: An integer

@return an integer

"""

def hashCode(self, key, HASH\_SIZE):

# write your code here

sum=0

length=len(key)

for i in range(length):

sum=(sum\*33)%HASH\_SIZE+ord(key[i])

return sum%HASH\_SIZE

2.检查两棵二叉树是否等价。等价的意思是说，首先两棵二叉树必须拥有相同的结构，并且每个对应位置上的节点上的数都相等。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

1 1

/ \ / \

2 2 and 2 2

/ /

4 4

就是两棵等价的二叉树。

1 1

/ \ / \

2 3 and 2 3

/ \

4 4

就不是等价的。

思路：如果要确定两个树是否等价就要遍历两个树然后判断是否相等，那两个树就要同时做递归，考虑到空节点不存在值，一个有空节点一个没有，两个值相等，和不等，因此入口就在相等处，出口就在不等处。是不是麻烦些，如果直接能通过前序或者中序遍历获得包括空节点的。那么比较即可，下面就是找到那些点是空节点，好像也有点麻烦。如果每种遍历的顺序是唯一的，那么两种遍历同时便可以确定树的唯一结构好像更麻烦。还是使用最简单一

def isIdentical(self, a, b):

# Write your code here

if a==None and b==None:

return True

if (a==None and b!=None) or (a!=None and b==None):

return False

if a.val==b.val:

return Solution.isIdentical(self, a.left, b.left) and \

Solution.isIdentical(self, a.right, b.right)

return False

3.

给出一个**无重叠的**按照区间起始端点排序的区间列表。

在列表中插入一个新的区间，你要确保列表中的区间仍然有序且**不重叠**（如果有必要的话，可以合并区间）。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

插入区间**[2, 5]** 到 **[[1,2], [5,9]]**，我们得到 **[[1,9]]**。

插入区间**[3, 4]** 到 **[[1,2], [5,9]]**，我们得到**[[1,2], [3,4], [5,9]]**。

思路：两个区间无非就是包含和不包含，不包含就是小的太大，或者大的太小，包含就直接取两个区间小值的最小值大值的最大值。包含要做flag来区分是否要改变

我觉得这样就ok

"""

Definition of Interval.

class Interval(object):

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

self.start = start

self.end = end

"""

class Solution:

"""

Insert a new interval into a sorted non-overlapping interval list.

@param intevals: Sorted non-overlapping interval list

@param newInterval: The new interval.

@return: A new sorted non-overlapping interval list with the new interval.

"""

def insert(self, intervals, newInterval):

result= []

# write your code here

#709ms

length=len(intervals)

if length==0:

return [newInterval]

for i in range(length):

flag=0

if intervals[i].end<newInterval.start:

flag=1

result.append( intervals[i])

elif intervals[i].start>newInterval.end:

flag=2

result.append(newInterval)

newInterval=intervals[i]#在这里可以优化，比如跳出，保留序号，以后直接append就行

else:

flag=3

newInterval.start=min(newInterval.start,intervals[i].start)

newInterval.end=max(newInterval.end,intervals[i].end)

if flag==1 or flag==3:

result.append(newInterval)

elif flag==2:

result.append( intervals[-1])

return result

4. 给定一棵二叉查找树和一个新的树节点，将节点插入到树中。

你需要保证该树仍然是一棵二叉查找树。

 注意事项

You can assume there is no duplicate values in this tree + node.

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给出如下一棵二叉查找树，在插入节点6之后这棵二叉查找树可以是这样的：

2 2

/ \ / \

1 4 --> 1 4

/ / \

3 3 6

思路：小于找左支大于找右枝，知道空时才插入。这样空时才返回node，否则就是一直走。

"""

Definition of TreeNode:

class TreeNode:

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

self.val = val

self.left, self.right = None, None

"""

class Solution:

"""

@param root: The root of the binary search tree.

@param node: insert this node into the binary search tree.

@return: The root of the new binary search tree.

"""

def insertNode(self, root, node):

# write your code here

if root==None:

return node

if node.val<root.val:

root.left=Solution.insertNode(self, root.left, node)

elif node.val>root.val:

root.right=Solution.insertNode(self, root.right, node)

return root

5. 用插入排序对链表排序

您在真实的面试中是否遇到过这个题？

Yes

**样例**

Given 1->3->2->0->null, return 0->1->2->3->null

无语，用什么排序不一样，直接取了拍下，按序号做链表即可。2.插入排序，就是从第三个开始找到要插入的地方，找的过程重复次数太多，做个函数仍外面，里面就是取出这个node，怎么取单项链表，只有把后面值与该节点交换下，然后next调到next.next上，然后把next拿出来，大致如此（没实现代码）

"""

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 linked list.

@return: The head of linked list.

"""

'''第一种方法用时617ms

a=getnumbers(head,a=[])

length=len(a)

a=sorted(a,key=lambda x:x.val)

a.append(None)

for i in range(length):

a[i].next=a[i+1]

return a[0]

def getnumbers(t,a):

while t!=None:

a.append(t)

t=t.next

return a

'''

'''第二种方法'''

def insertionSortList(self, head):

# write your code here

j=getnumbers(head,i=0)

t=head

for j in range(1,j):

t=head

while t.val>t.next.val:

t.next.val,t.val=t.val,t.next.val

t=t.next

if t.next==None:

break

return head

def getnumbers(t,i):

while t!=None:

i+=1

t=t.next

return i

6. 返回两个数组的交

 注意事项

* Each element in the result must be unique.
* The result can be in any order.

您在真实的面试中是否遇到过这个题？

Yes

**样例**

nums1 = [1, 2, 2, 1], nums2 = [2, 2], 返回 [2].

思路：就是做集合交就行，

class Solution:

# @param {int[]} nums1 an integer array

# @param {int[]} nums2 an integer array

# @return {int[]} an integer array

def intersection(self, nums1, nums2):

# Write your code here

'''first use definition cost 870ms

nums1=set(nums1)

nums2=set(nums2)

a=[]

for i in nums1:

if i in nums2:

a.append(i)

return a

'''

'''second cost 729ms

nums1=set(nums1)

nums2=set(nums2)

a=nums1&nums2

return list(a)

'''

7. 计算两个数组的交

 注意事项

每个元素出现次数得和在数组里一样  
答案可以以任意顺序给出

您在真实的面试中是否遇到过这个题？

Yes

**样例**

nums1 = [1, 2, 2, 1], nums2 = [2, 2], 返回 [2, 2].

思路：就是做两个字典加入有公共部分，查看交集的最小个数。

class Solution:

# @param {int[]} nums1 an integer array

# @param {int[]} nums2 an integer array

# @return {int[]} an integer array

def intersection(self, nums1, nums2):

# Write your code here

'''迭代算法很占内存，此题对空间复杂度要求比较高

nums1.sort()

nums2.sort()

return serach(nums1,nums2,a=[])

def serach(nums1,nums2,a):

if len(nums1)!=0 and len(nums2)!=0:

if nums1[0]>nums2[0]:

serach(nums1,nums2[1:],a)

elif nums1[0]<nums2[0]:

serach(nums1[1:],nums2,a)

else:

a.append(nums1[0])

serach(nums1[1:],nums2[1:],a)

return a

'''

第二中

'''

nums1 = sorted(nums1)

nums2 = sorted(nums2)

ans = []

i = 0

j = 0

while i < len(nums1) and j < len(nums2):

if nums1[i] < nums2[j]:

i += 1

elif nums1[i] > nums2[j]:

j +=1

else:

ans+=nums1[i],

i += 1

j += 1

return ans

'''

第三种

dict1={}

dict2={}

for i in nums1:

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

for j in nums2:

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

nums1=set(nums1)

nums2=set(nums2)

a=nums1&nums2

temp=[]

for h in a:

t=min(dict1[h],dict2[h])

for k in range(t):

temp.append(h)

return temp

8. 翻转一棵二叉树

您在真实的面试中是否遇到过这个题？

Yes

**样例**

1 1

/ \ / \

2 3 => 3 2

/ \

4 4

思路：直接交换就行

"""

Definition of TreeNode:

class TreeNode:

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

self.val = val

self.left, self.right = None, None

"""

class Solution:

# @param root: a TreeNode, the root of the binary tree

# @return: nothing

def invertBinaryTree(self, root):

# write your code here

if root.left !=None or root.right !=None:

root.left,root.right=root.right,root.left

if root.left !=None:

Solution.invertBinaryTree(self,root.left)

if root.right !=None:

Solution.invertBinaryTree(self,root.right)

9. 实现一个leftpad库，如果不知道什么是leftpad可以看样例

您在真实的面试中是否遇到过这个题？

Yes

**样例**

leftpad("foo", 5)

>> " foo"

leftpad("foobar", 6)

>> "foobar"

leftpad("1", 2, "0")

>> "01"

思路就是左填充。

class StringUtils:

# @param {string} originalStr the string we want to append to

# @param {int} size the target length of the string

# @param {string} padChar the character to pad to the left side of the string

# @return {string} a string

@classmethod

def leftPad(self, originalStr, size, padChar=' '):

# Write your code here

str1=originalStr

length=len(str1)

remaind=size-length

str=''

for i in range(remaind):

str+=padChar

str=str+str1

return str

10. 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：以空格分隔的话就得到单词和空格两种情况，空格就返回0，单词就返回长度，不过倒着走复杂度应该会低点。

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)