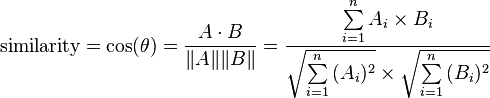
1. 余弦相似度

思路：按公式求解，考虑边界条件，不能计算的状态，即分母为零，向量不同维。

class Solution:

"""

@param A: An integer array.

@param B: An integer array.

@return: Cosine similarity.

"""

def cosineSimilarity(self, A, B):

# write your code here

lengtha=len(A)

lengthb=len(B)

a=0.0

b=0.0

sum=0.0

if lengtha!=lengthb:

return 2

for i in range(lengtha):

sum+=(A[i]\*B[i])

a+=A[i]\*A[i]

b+=B[i]\*B[i]

if a==0 or b==0:

return 2

a=a\*\*0.5

b=b\*\*0.5

return sum/(a\*b)

1. 计算在一个 32 位的整数的二进制表示中有多少个 1.

两种思路：一种是把整数转换成二进制的数，然后转换成字符串并统计一出现的个数。

伪代码：

得到输入的整数-》转换成二进制-》转换成字符串-》统计1出现的次数。

思路二：

假如能与1按位与那么只要计算和就可以知道整个数中有多少个1，但是整数转换为二进制位数不可知，因此只能通过右移直到该整数为

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 sum0。

3. 报数指的是，按照其中的整数的顺序进行报数，然后得到下一个数。如下所示：

1, 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。

首先要有循环来计数，保证是第n个顺序，第二个就是制定规则，发现单个的读one有相邻的则计数直到不相同，不相同在出栈入栈，当然这样是最后一个无法比较的，因此要把最后一位数入栈。然后输出即可

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

#### 4在O(1)时间复杂度删除链表节点

思路给出链表的节点，因为是单向链表所以只会有next实现，因此把要去掉的节点的next的值取出来，代替要去掉节点的值，这个再把next往后移位，当然要删除节点不会是空。

"""

Definition of ListNode

class ListNode(object):

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

self.val = val

self.next = next

"""

class Solution:

# @param node: the node in the list should be deleted

# @return: nothing

def deleteNode(self, node):

# write your code here

node.val=node.next.val

node.next=node.next.next

5. 给定一个排序的整数数组（升序）和一个要查找的整数target，用O(logn)的时间查找到target第一次出现的下标（从0开始），如果target不存在于数组中，返回-1

在数组 [1, 2, 3, 3, 4, 5, 10] 中二分查找3，返回2。

二分查找，递归实现，出口为两个一个是找到了，返回index，但这是index不能保证是第一个出现的，因此需要在递减查找，最后跳出时+1即可，另一种是找不到，返回-1，找不到就是start比end要大，利用循环也行。

class Solution:

# @param nums: The integer array

# @param target: Target number to find

# @return the first position of target in nums, position start from 0

def binarySearch(self, nums, target):

# write your code here

length=len(nums)

start=0

end=length-1

t=-2

while(start<=end):

mid=(start+end)//2

if nums[mid]>target:

end=mid-1

elif nums[mid]<target:

start=mid+1

else:

t=mid

break

if t==-2:

return -1

while nums[t]==target:

if t>0:

t-=1

else:

return 0

return t+1

6. 给你一个整数*n*. 从 *1* 到 *n* 按照下面的规则打印每个数：

* 如果这个数被3整除，打印fizz.
* 如果这个数被5整除，打印buzz.
* 如果这个数能同时被3和5整除，打印fizz buzz.

比如 *n* = 15, 返回一个字符串数组：

[

"1", "2", "fizz",

"4", "buzz", "fizz",

"7", "8", "fizz",

"buzz", "11", "fizz",

"13", "14", "fizz buzz"

]

思路没啥说的，就是假如能同时被3和5整出输出也能被3整出，因此这个同时优先度最高，要放在最前面。当然能同时被3和5整出就是被15整除。

class Solution:

"""

@param n: An integer as description

@return: A list of strings.

For example, if n = 7, your code should return

["1", "2", "fizz", "4", "buzz", "fizz", "7"]

"""

def fizzBuzz(self, n):

results = []

for i in range(1, n+1):

if i % 15 == 0:

results.append("fizz buzz")

elif i % 5 == 0:

results.append("buzz")

elif i % 3 == 0:

results.append("fizz")

else:

results.append(str(i))

return results

7. 将一棵二叉树按照前序遍历拆解成为一个假链表。所谓的假链表是说，用二叉树的 right 指针，来表示链表中的 next 指针。

**样例**

1

\

1 2

/ \ \

2 5 => 3

/ \ \ \

3 4 6 4

\

5

\

6

[**挑战**](http://www.lintcode.com/zh-cn/problem/flatten-binary-tree-to-linked-list/#challenge)

不使用额外的空间耗费。

要是不使用额外空间就是要在原树上改，算了，太难了，直接做个list吧前序遍历结果都存起来，然后一直right就行了

"""

Definition of TreeNode:

class TreeNode:

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

this.val = val

this.left, this.right = None, None

"""

class Solution:

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

# @return: nothing

def flatten(self, root):

# write your code here

'''第一种方法用时338ms

if root==None:

return

qianxu(root)

def qianxu(root):

if root==None:

return None

leftlink=qianxu(root.left)

rightlink=qianxu(root.right)

if leftlink!= None:

leftlink.right = root.right

root.right = root.left

root.left = None

if rightlink!= None:

return rightlink

elif leftlink != None:

return leftlink

else :

return root

'''

'''第二中方法用时306ms,关键在于python变量分为可变和不可变-字典和list，对内部即.形式访问也不可变'''

if root==None:

return

b=qianxu(root,a=[])

for i in range(1,len(b)):

root.right=TreeNode(b[i])

root.left=None

root=root.right

def qianxu(root,a=[]):

if root!=None:

a.append(root.val)

qianxu(root.left,a)

qianxu(root.right,a)

return a

8. 给定一个列表，该列表中的每个要素要么是个列表，要么是整数。将其变成一个只包含整数的简单列表。

 注意事项

如果给定的列表中的要素本身也是一个列表，那么它也可以包含列表。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给定 [1,2,[1,2]]，返回 [1,2,1,2]。

给定 [4,[3,[2,[1]]]]，返回 [4,3,2,1]。

[**挑战**](http://www.lintcode.com/zh-cn/problem/flatten-list/#challenge)

请用非递归方法尝试解答这道题。

主要思路就是判断数据类型，然后入list，嵌套型必然会用到递归，当然比较麻烦，是不是可以转换成字符串然后直接判断了，风格依据为逗号-要考虑情况太多

class Solution(object):

# @param nestedList a list, each element in the list

# can be a list or integer, for example [1,2,[1,2]]

# @return {int[]} a list of integer

def flatten(self, nestedList):

# Write your code here

'''

if type(nestedList)==int:

return [nestedList]

a=[]

getint(nestedList,a=a)

return a

def getint(nestedList,a):

if type(nestedList)== int:

return nestedList

for i in range(len(nestedList)):

b=getint(nestedList[i],a)

if type(b)==int:

a.append(b)

'''

第二种时间复杂度比较高，还可以改进，不过走偏了不推荐。

a=str(nestedList)

str1=''

for i in a:

try :

(i=='-' or i==','or int(i))

str1+=str(i)

except:

pass

l=str1.split(',')

temp=[]

for j in l:

if j =='':

pass

else:

temp.append(int(j))

return temp

9. 如果要将整数A转换为B，需要改变多少个bit位？

 注意事项

Both *n* and *m* are 32-bit integers.

您在真实的面试中是否遇到过这个题？

Yes

**样例**

如把31转换为14，需要改变2个bit位。

(**31**)10=(**11111**)2

(**14**)10=(**01110**)2

思路，按位异或吧，在统计有多少个1

class Solution:

"""

@param a, b: Two integer

return: An integer

"""

def bitSwapRequired(self, a, b):

# write your code here

t=a^b

temp=0

s=0

while s < 32:

k=t&1

temp+=k

s+=1

t=t>>1

return temp

10.写一个算法来判断一个数是不是"快乐数"。

一个数是不是快乐是这么定义的：对于一个正整数，每一次将该数替换为他每个位置上的数字的平方和，然后重复这个过程直到这个数变为1，或是无限循环但始终变不到1。如果可以变为1，那么这个数就是快乐数。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

19 就是一个快乐数。

1^2 + 9^2 = 82

8^2 + 2^2 = 68

6^2 + 8^2 = 100

1^2 + 0^2 + 0^2 = 1

思路换成支符串，按位运算，然后循环，直到一位数，是一就是快乐书，否则就不是。

class Solution:

# @param {int} n an integer

# @return {boolean} true if this is a happy number or false

def isHappy(self, n):

# Write your code here

j=1

while n !=1:

j+=1

n=str(n)

sum=0

for i in range(len(n)):

sum=sum+int(n[i])\*\*2

n=int(sum)

if len(str(n))==1 and n!=1:

return False

return True