1.查找斐波纳契数列中第 N 个数。

所谓的斐波纳契数列是指：

* 前2个数是 0 和 1 。
* 第 *i* 个数是第 *i*-1 个数和第*i*-2 个数的和。

斐波纳契数列的前10个数字是：

0, 1, 1, 2, 3, 5, 8, 13, 21, 34 ...

 注意事项

The *N*th fibonacci number won't exceed the max value of signed 32-bit integer in the test cases.

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给定 1，返回 0

给定 2，返回 1

给定 10，返回 34

思路：通项是f(n)=f(n-1)+f(n-2),所以只要保存二个值就行，

class Solution:

# @param n: an integer

# @return an integer f(n)

def fibonacci(self, n):

# write your code here

'''算法时间复杂度是O(n)

a=[0,1]

if n<2:

return a[n-1]

for i in range(2,n+1):

a.append(a[i-1]+a[i-2])

return a[n-1]

'''

#分治法算法复杂度时o(lg(n))

a=[[1,1],[1,0]]

real=n-1

if real%2==0:

temp=a\*\*(real//2)

return temp\*temp

else:

temp=a\*\*((real-1)//2)

return temp\*temp

2. 给定一个单链表中的一个等待被删除的节点(非表头或表尾)。请在在O(1)时间复杂度删除该链表节点。

就是直接删吧

"""

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

3.笔试题目：

a='0123456789'

k=1

B='111a222b22caaasc'

flag=1

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

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

if B[i] in a:

pass

else:

k+=1

if k>=3:

print('error')

flag=0

break

else:

k=1

str1=''

temp=''

if flag:

for j in B:

if j in a:

str1+=j

else:

if str1=='':

str1=1

temp+=str(j)\*(int(str1))

str1=''

print(temp)

4. 有 n 个硬币排成一条线。两个参赛者轮流从右边依次拿走 1 或 2 个硬币，直到没有硬币为止。拿到最后一枚硬币的人获胜。

请判定 **第一个玩家** 是输还是赢？

思路：多写点通项应该能看出规律，1，2，是赢3是输所以在输的基础上加一加二也是赢，加三就输，在输的基础上在加一加二那么就是3的倍数都输。

class Solution:

# @param n: an integer

# @return: a boolean which equals to True if the first player will win

def firstWillWin(self, n):

# write your code here

if n %3==0:

return False

return True

5. 有 n 个不同价值的硬币排成一条线。两个参赛者轮流从左边依次拿走 1 或 2 个硬币，直到没有硬币为止。计算两个人分别拿到的硬币总价值，价值高的人获胜。

请判定 **第一个玩家** 是输还是赢？

给定数组 A = [1,2,2], 返回 true.

给定数组 A = [1,2,4], 返回 false.

假设3的倍数之和大于和的一半，那么就是必输的。这么想肯定少情况。假如只有3n+1个，那么第二个可以class Solution:

# @param values: a list of integers

# @return: a boolean which equals to True if the first player will win

def firstWillWin(self, values):

# write your code here

length=len(values)

if length<3:

return True

sum=0

sorta=sorted(values,reverse=True)

dict={}

for i in range(length):

sum+=sorta[i]

dict[sorta[i]]=i

j=2

sumb=0

while j<length:

sumb+=values[j]

j+=3

if sumb>sum/2:

return False

return True

6. 假设你有一个数组，它的第i个元素是一支给定的股票在第i天的价格。

设计一个算法来找到最大的利润。你最多可以完成 k 笔交易。

 注意事项

你不可以同时参与多笔交易(你必须在再次购买前出售掉之前的股票)

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给定价格 = [4,4,6,1,1,4,2,5], 且 k = 2, 返回 6.

需要定义三种状态：卖，买，和不卖。买这个状态怎么定义，就是假设我后一天减去前一天这样就定义了买。然后有跳步就是卖了，没有就是不卖，那么剩下就是k了，假设我每个都是每个状态都卖买了0-k次，这样就可以确定。当前状态与下一个状态之间关系，这样每个状态都会有k终情况，这样对以后很有力。这样就把所有信息利用上了。在状态转移过程终会产生多少信息。即每个状态都有k中划分，每种划分都有卖和没卖的概念。因此。

Temp[h]=max（[temp1[h-1]+max(0,nums[j]),temp[h]+nums[j]]）

Temp1=max(temp,temp1)

因为时至多k个那么，需要按时间来一步步来。即前n天卖了k-1次。

class Solution:

"""

@param k: an integer

@param prices: a list of integer

@return: an integer which is maximum profit

"""

def maxProfit(self, k, prices):

# write your code here

'''

length=len(prices)

if length==1 or length==0:

return 0

nums=[]

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

nums.append(prices[i+1]-prices[i])

while h=0

def gettwomax(arr):

length=len(arr)

if length==1:

if arr[-1]>0:

return arr[-1]

else:

return 0

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

def getmax(nums):

max2=-8888

sum=0

for j in range(len(nums)):

sum+=nums[j]

if sum>max2:

max2=sum

if sum<0:

sum=0

if max2<0:

max2=0

return max2

'''

length=len(prices)

if length==1 or length==0:

return 0

nums=[]

temp=[]

te=0

h=0

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

if prices[i+1]-prices[i]>0:

temp.append(prices[i+1]-prices[i])

te+=temp[h]

h+=1

nums.append(prices[i+1]-prices[i])

if len(temp)<=k:

return te

globa=[0 for l in range(k+1)]

local=[0 for p in range(k+1)]

for j in range(len(nums)):

for h in range(k,0,-1):

local[h]=ma x(globa[h-1]+max(0,nums[j]),local[h]+nums[j])

globa[h]=max(globa[h],local[h])

return globa[k]

7. 假设你是一个专业的窃贼，准备沿着一条街打劫房屋。每个房子都存放着特定金额的钱。你面临的唯一约束条件是：相邻的房子装着相互联系的防盗系统，且 **当相邻的两个房子同一天被打劫时，该系统会自动报警**。

给定一个非负整数列表，表示每个房子中存放的钱， 算一算，如果今晚去打劫，你最多可以得到多少钱 **在不触动报警装置的情况下**。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给定 [3, 8, 4], 返回 8.

思路就是动态规划，通项是a[n]=max(a[n-1],a[n-2]+nums[n])

class Solution:

# @param A: a list of non-negative integers.

# return: an integer

def houseRobber(self, A):

# write your code here

'''

A.insert(0,0)

A.insert(0,0)

n=len(A)

if n<=3:

return A[-1]

temp=[0 for i in range(len(A))]

jishuan(n-1,A,temp)

return max(temp[n-1],temp[n-2])

def jishuan(n,A,temp):

i=0

while i<=n:

if i<4:

temp[i]=A[i]

temp[i]=max(temp[i-2],temp[i-3])+A[i]

i+=1

'''

if A==[]:

return 0

temp=[0 for i in range(len(A))]

if len(A)<=2:

return max(A)

temp[0]=A[0]

temp[1]=A[1]

for i in range(2,len(A)):

temp[i]=max(temp[i-1],temp[i-2]+A[i])

return temp[-1]

8. 将一个整数中的数字进行颠倒，当颠倒后的整数溢出时，返回 0 (标记为 32 位整数)。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给定 x = 123，返回 321

给定 x = -123，返回 -321

思路：判断是否溢出，当然符号要确定。

class Solution:

# @param {int} n the integer to be reversed

# @return {int} the reversed integer

def reverseInteger(self, n):

# Write your code here

a=[]

if n<0:

n=abs(n)

j=1

else:

j=2

while n//10!=0:

h=n

a.append(h)

n=n//10

a.append(n)

length=len(a)

sum=0

for i in range(length):

sum=sum+a[i]\*pow(10,(length-i-1))

#print(sum)

if sum>pow(2,32):

return 0

return sum\*pow(-1,j)

9. 格雷编码是一个二进制数字系统，在该系统中，两个连续的数值仅有一个二进制的差异。

给定一个非负整数 n ，表示该代码中所有二进制的总数，请找出其格雷编码顺序。一个格雷编码顺序必须以 0 开始，并覆盖所有的 2n 个整数。

 注意事项

对于给定的 n，其格雷编码顺序并不唯一。

根据以上定义， [0,2,3,1] 也是一个有效的格雷编码顺序。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给定 n = 2， 返回 [0,1,3,2]。其格雷编码顺序为：

00 - 0

01 - 1

11 - 3

10 - 2

思路就是n为二进制有的个数：3个就是由两个扩展来的，n个就是由n-1个扩展来的。所以temp.append(2\*\*n-1(temp))

class Solution:

# @param {int} n a number

# @return {int[]} Gray code

def grayCode(self, n):

# Write your code here

if n==0:

return[0]

b=[0,1]

'''

for i in range(2\*\*n):

temp=i^i>>1

b.append(temp)

return b

'''

for i in range(2,n+1):

length=len(b)

for j in range(length)[::-1]:

b.append((2\*\*(i-1))+b[j])

return b

10．将两个整数相除，要求不使用乘法、除法和 mod 运算符。

如果溢出，返回 2147483647 。

您在真实的面试中是否遇到过这个题？

Yes

**样例**

给定被除数 = 100 ，除数 = 9，返回 11。

思路：应该是减法操作，大于零就减，小于零就可以返回了，假如是负数呢，转换成正数，这样需要一个判断符号的，如果是零就会溢出吧大于也会溢出吧，一个一个减比较浪费时间可以按倍数减。

class Solution:

# @param {int} dividend the dividend

# @param {int} divisor the divisor

# @return {int} the result

def divide(self, dividend, divisor):

# Write your code here

t=2147483647

if divisor==0:

return t

flag1=flag(dividend)

flag2=flag(divisor)

dividend=abs(dividend)

divisor=abs(divisor)

if dividend<divisor:

return 0

else:

result=integer\_div\_2(dividend, divisor)

temp= flag1\*flag2\*result

if temp>t:

return t

else:

return temp

def flag(nums):

flag=1

if nums<0:

flag=-1

return flag

def integer\_div\_2( dividend, divisor):

k = 0;c = divisor;res = 0

while dividend >= c:

if dividend - c < divisor:

return 1<<k

c<<=1

k+=1

return integer\_div\_2(dividend -(c>>1), divisor) + (1<<(k - 1))