题目一：

一个数字串可以被拆开成多个数字串，例如12345拆成12 3 45或者123 45。给一个正整数类型的数字串n，求n拆开后的数能被3整除的最大数量m是多少。(0也算3的倍数)
举例：n=12345，拆成：
1): 12, 3, 45, m=3
2): 123, 45, m=2

输入描述:

输入一个正整数类型的数字串n(字符串长度<100)

输出描述:

输出一个数字表示n拆开后能被3整除最多的数量

示例1

输入 复制

```
321
```

输出 复制

```
2
```

思路：首先判断整体能否被 3 整除，如果不能，返回 0。如果能被 3 整除，这从左往右一次读数字，当能整除时，数目加一，temp 归零，否则向右移动，temp 加上该位的数字

```
class Solution:
    def find3(self,nums):
        if nums<0:
            return 0
        NumToStr = list(str(nums))
        maxN = 0
        temp = 0
        for i in range(len(NumToStr)):
            temp = temp * 10 + int(NumToStr[i])
            if temp % 3 == 0:
                maxN += 1
                temp = 0
        if temp % 3 == 0:
            return maxN
        else:
            return 0


num = int(input())
a = Solution()
print(a.find3(num))
```

第二题

思路：暴力解法会超时，暴力解法就是 findkth_2 依次加一，找到第 k 个数。题目中说道了位运算，所以可以考虑使用位运算来结果

加入 x 为 5（二进制为 101），如果要符合 x+y=x|y，则说明只能在为 0 的位置进行运算。

当 k 为 1(0b01)时，x|y=0b111，（红色说明是原来的 1）

当 k 为 2(0b10)时，x|y=0b1101

当 k 为 3(0b11)是，x|y=0b1111

说明需要将 k 的二进制插入到 x 的二进制的为 0 的位置中。

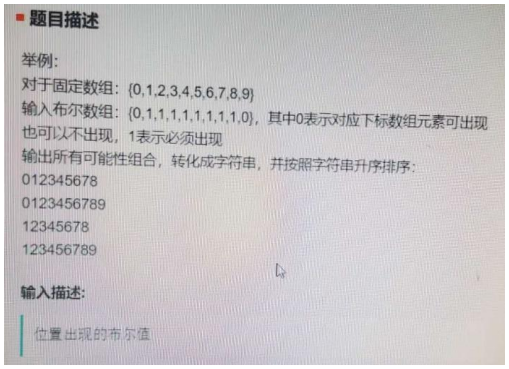
```
class Solution:
    def findkth(self,num,k):
        numtostr = list(str(bin(num)))[2:]
        ktostr = list(str(bin(k)))[2:]
        for i in range(len(numtostr)-1,-1,-1):
            if numtostr[i] == '0':
                numtostr[i] = ktostr[-1]
                ktostr.pop(-1)
                if not ktostr:
                    break
        if ktostr:
            numtostr = ktostr + numtostr
        s = int(''.join(numtostr), 2)
        return s - num


    def findkth_2(self,num,k):
        y = 1#返回的 y 值
        n = 0#表示次数
        while n <= k:
            temp = num | y
```

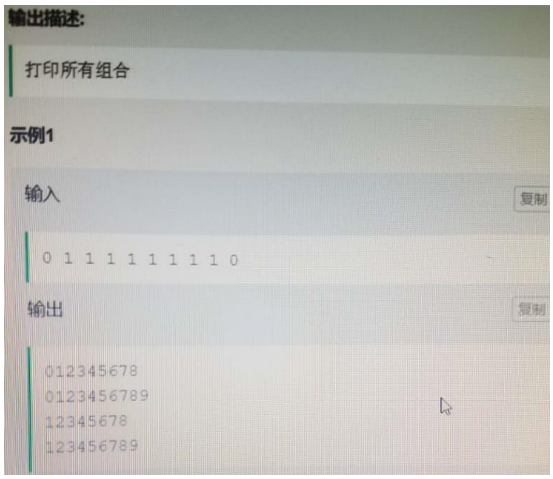
```
            if num + y == temp:
                n += 1
            if n == k:
                return y
            y += 1
        return y


a = Solution()
# t = int(input())
# for i in range(t):
#     xandk = list(map(int,input().strip().split()))
#     print(a.findkth(xandk[0],xandk[1]))
num = 7
k = 125156416
print(a.findkth(num,k))
```

题目 3: :输入字符串 s1, 将 s1 中的子串 s2 替换成 s3。例如 s1='this is a chick', s2='is',s3='are', 最终输出的结果是'thare are a chick'。
思路：利用 Python 的 find 函数，找到出现 s2 的位置，然后进行替换即可。

```
s1 = 'this is a chick'
s2 = 'is'
s3 = 'are'

res = ''
x = s1.find(s2)
while x != -1:
    res += s1[:x] + s3
    s1 = s1[x + len(s2):]
    x = s1.find(s2)
res += s1
print(res)
```

题目 4:





思路: 方法一是我使用了全排列的思路, 但貌似复杂了。方法二是网上大佬给的, 十分简洁。方法二的思路是, 当此时 boolset 为 1 时, 则让 temp_s = temp_s+str(i), 当 boolset 为 0 时, 就有两种情况, 让 temp_s = temp_s+str(i)或者 temp_s 不变。方法一是改变 boolset, 最后将其变回来。

```
class Solution:
    def __init__(self):
        self.res = []

    def findall(self,nums,boolset):
        self.findall_core(nums,boolset,0)
        return self.res

    def findall_core(self,num,boolset,index):
        i = index
        while i < len(boolset):
            if boolset[i] != 0:
                i += 1# 找到下一个为 0 的位置
            else:
                break
        if i == len(boolset):
            temp = []
            for i in range(len(boolset)):
                if boolset[i]==0:
                    # temp.append(0)
                    continue
                else:
                    temp.append(num[i])
            self.res.append(temp[:])
            return

        self.findall_core(num, boolset, i + 1)  # 处理 0
```

```
        boolset[i] = 1# 变成 1
        self.findall_core(num, boolset, i + 1)  # 处理 1
        boolset[i] = 0

class Solution2:
    def __init__(self):
        self.res = []

    def findall(self,boolset):
        self.findall_core(boolset,0,'')
        return self.res

    def findall_core(self,boolset,index,temp_s):
        if index == len(boolset):
            self.res.append(temp_s)
            return
        if boolset[index] == 1:
            self.findall_core(boolset, index + 1, temp_s + str(index))
        else:
            self.findall_core(boolset, index + 1, temp_s)
            self.findall_core(boolset, index + 1, temp_s + str(index))

nums = [0,1,2,3,4,5,6,7,8,9]
boolset = [0,1,1,1,1,1,1,1,1,0]
# a = Solution()
# b=a.findall(nums,boolset)
# b.sort()
# print(b)

c = Solution2()
d = c.findall(boolset)
print(d)
```

题目 5: 最大上升子序列和。即给出一个数组，如[5,1,3,4,9,7,6,8]，找到所有它的上升子序列，如[5,6,8],[1,3,4,7,8]，输出这些子序列的最大和。
思路：利用回溯法，找到所有上升子序列，然后利用 sum 相加，输出最大值。当网上的大佬提供了另外一种思路，动态规划，这种方法很简洁 https://www.nowcoder.com/discuss/100129。

```
class Solution:
    def __init__(self):
        self.res = []
        self.maxN = 0
        self.length = 0
```

```
6.
7.      def findmaxN(self,lis):
8.          self.length = len(lis)
9.          temp = []
10.         for i in range(len(lis)):
11.             temp.append(lis[i])
12.             self.findmaxN_core(lis,i+1,temp,lis[i])
13.             temp.pop()
14.         for i in self.res:
15.             temp = sum(i)
16.             if temp>self.maxN:
17.                 self.maxN = temp
18.         print(self.maxN)
19.
20.     def findmaxN_core(self,lis,index,temp,cur_max):
21.         if index == self.length:
22.             self.res.append(temp[:])
23.             return
24.         for i in range(index,self.length):
25.             if lis[i]>cur_max:
26.                 cur_max = lis[i]
27.                 temp.append(lis[i])
28.                 self.findmaxN_core(lis,i+1,temp,cur_max)
29.                 temp.pop(-1)
30.                 cur_max = temp[-1]
31.             if i == self.length - 1:
32.                 self.res.append(temp[:])
33.                 return
34.
35. # lis = list(map(int,input().strip().split()))
36. a = Solution()
37. lis = [5,1,3,4,9,7,6,8]
38. a.findmaxN(lis)
```

题目 6：n 面筛子，m 面有奖，有奖继续掷筛子，没奖结束。输入给定每面分数的数组 s, len(s) = n， 求期望。

思路：http://www.cnblogs.com/hgc-bky/p/9553202.html