剑指offer（python版） 29-32

29最小的K个数

题目：输入n个整数，找出其中最小的K个数。例如输入4,5,1,6,2,7,3,8这8个数字，则最小的4个数字是1,2,3,4,。

思路：利用冒泡法，临近的数字两两进行比较,按照从小到大的顺序进行交换,如果前面的值比后面的大，则交换顺序。这样一趟过去后,最小的数字被交换到了第一位；然后是次小的交换到了第二位，依次直到第k个数，停止交换。返回lists的前k个数（lists[0:k]，前闭后开）

要注意到如果k大于n的情况

代码：

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. **def** GetLeastNumbers\_Solution(self, tinput, k):
4. # write code here
5. length=len(tinput)
6. **if** k>length:
7. **return**[]
8. **for** i **in** range(k):
9. **for** j **in** range(i+1,length):
10. **if** tinput[i]>tinput[j]:
11. tinput[j],tinput[i]=tinput[i],tinput[j]
12. **return** tinput[0:k]

30连续子数组的最大和

题目：HZ偶尔会拿些专业问题来忽悠那些非计算机专业的同学。今天测试组开完会后,他又发话了:在古老的一维模式识别中,常常需要计算连续子向量的最大和,当向量全为正数的时候,问题很好解决。但是,如果向量中包含负数,是否应该包含某个负数,并期望旁边的正数会弥补它呢？例如:{6,-3,-2,7,-15,1,2,2},连续子向量的最大和为8(从第0个开始,到第3个为止)。给一个数组，返回它的最大连续子序列的和，你会不会被他忽悠住？(子向量的长度至少是1)

思路：想到了动态规划，a[i]为结尾的子数组的最大和

代码：

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. **def** FindGreatestSumOfSubArray(self, array):
4. # write code here
5. dp=array[0]
6. maxsum=array[0]
7. **for** i **in** range(1,len(array)):
8. dp=max(array[i],array[i]+dp)
9. maxsum=max(dp,maxsum)
10. **return** maxsum

31整数中1出现的次数（从1到n整数中1出现的次数）

题目：求出1~13的整数中1出现的次数,并算出100~1300的整数中1出现的次数？为此他特别数了一下1~13中包含1的数字有1、10、11、12、13因此共出现6次,但是对于后面问题他就没辙了。ACMer希望你们帮帮他,并把问题更加普遍化,可以很快的求出任意非负整数区间中1出现的次数（从1 到 n 中1出现的次数）

思路：<https://blog.csdn.net/u010005281/article/details/80085255>

1. 如果第i位（自右至左，从1开始标号）上的数字为0，则第i位可能出现1的次数由更高位决定（若没有高位，视高位为0），等于更高位数字X当前位数的权重10i-1。

2. 如果第i位上的数字为1，则第i位上可能出现1的次数不仅受更高位影响，还受低位影响（若没有低位，视低位为0），等于更高位数字X当前位数的权重10i-1+（低位数字+1）。

3. 如果第i位上的数字大于1，则第i位上可能出现1的次数仅由更高位决定（若没有高位，视高位为0），等于（更高位数字+1）X当前位数的权重10i-1。

总结一下以上的算法，可以看到，当计算右数第 i 位包含的 X 的个数时：

1. 取第 i 位左边（高位）的数字，乘以 10i−1，得到**基础值** a。
2. 取第 i 位数字，计算**修正值**：
   1. 如果大于 X，则结果为 a+10i−1。
   2. 如果小于 X，则结果为 a。
   3. 如果等 X，则取第 i 位右边（低位）数字，设为 b，最后结果为 a+b+1。

代码：

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. **def** NumberOf1Between1AndN\_Solution(self, n):
4. # write code here
5. **if** n < 1:
6. **return** 0
7. mult, sumTimes = 1, 0
8. **while** n//mult:
9. div, mod = divmod(n, mult\*10)
10. curNum, curMod = divmod(mod, mult)
11. **if** curNum > 1:
12. sumTimes += div\*mult + mult
13. **elif** curNum == 1:
14. sumTimes += div\*mult + curMod + 1
15. **else**:
16. sumTimes += div\*mult
17. mult = mult\*10
18. **return** sumTimes

32把数组排成最小的数

题目：输入一个正整数数组，把数组里所有数字拼接起来排成一个数，打印能拼接出的所有数字中最小的一个。例如输入数组{3，32，321}，则打印出这三个数字能排成的最小数字为321323

思路：具体思路：  
  （1）先将数字列表转化成字符串链表，这样便于在一个字符串后面直接加上另外一个字符串。也就是 "3"+"321"="3321"。  
  （2）构造一个比较函数，当 *str*1+*str*2>*str*2+*str*1时我们认为字符串 *str*1

>str2。  
  （3）将字符串列表按照比较函数的规定进行冒泡排序（或其它方法排序），将定义为”大”的字符串放到最后。而”小”的字符串放在前面。最后将字符串列表链接起来，便是所求

代码:

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. **def** theMax(self, str1, str2):
4. '''''定义字符串比较函数'''
5. **return** str1 **if** str1+str2 > str2+str1 **else** str2
6. **def** PrintMinNumber(self, numbers):
7. # write code here
8. """使用冒泡进行排序(把最大的放最后)"""
9. string = [str(num) **for** num **in** numbers]
10. res = []
11. flag = True
12. count = len(string) - 1
13. **while** flag **and**  count > 0:
14. flag = False
15. **for** i **in** range(len(string)-1):
16. **if** self.theMax(string[i], string[i+1]) == string[i]:
17. temp = string[i]
18. **del** string[i]
19. string.insert(i+1, temp)
20. flag = True
21. count -= 1
22. string = ''.join(string)
23. **return** string

代码二（比较简单）：

**注意**：在python2.x中可直接对 sort() 函数进行比较函数cmp的传参，可以利用sort()函数直接对字符串列表进行排序。但是，对于python3.x取消了sort()函数中的cmp参数。

1. # -\*- coding:utf-8 -\*-
2. **class** Solution:
3. **def** PrintMinNumber(self, numbers):
4. # write code here
5. **if** **not** numbers:
6. **return** ''
7. numstr = map(str,numbers) #用str方式将int型数字转换成string
8. l = **lambda** n1,n2:int(n1+n2)-int(n2+n1)# 指定比较方式
9. numsort = sorted(numstr,cmp=l) #用特定的比较方式进行比较
10. **return** "".join(i **for** i **in** numsort) #拍完序之后用join进行连接成结果