

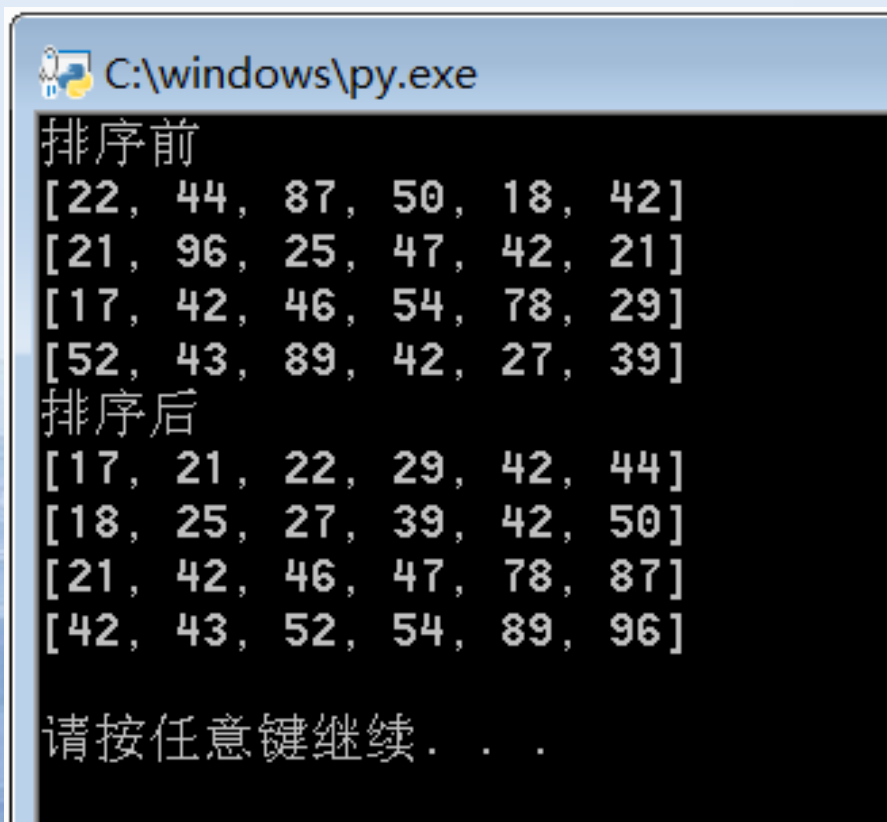
【1】 设计算法，原地将一个序列中所有奇数移到所有偶数的前边，要求：①保持奇数和偶数原来的相对次序；②算法的空间复杂度为 $O(1)$ 。例如：若原序列为：

[3, 2, 6, 1, 4, 9, 8, 5, 11, 7]

则处理后的序列为：

[3, 1, 9, 5, 11, 7, 2, 6, 4, 8]

【2】 试编写对一个 $N \times M$ 矩阵原地进行排序的算法。要求排好序后的矩阵的每一行（从左到右）和每一列（从上到下）都是递增排列（结果可能不唯一）。



```
C:\windows\py.exe
排序前
[22, 44, 87, 50, 18, 42]
[21, 96, 25, 47, 42, 21]
[17, 42, 46, 54, 78, 29]
[52, 43, 89, 42, 27, 39]
排序后
[17, 21, 22, 29, 42, 44]
[18, 25, 27, 39, 42, 50]
[21, 42, 46, 47, 78, 87]
[42, 43, 52, 54, 89, 96]
请按任意键继续...
```

【3】 按照快速排序的思想，编写实现链表排序的算法。

【4】 试编写一个非递归的快速排序算法。

【5】 已知一个由 n 个数字（0到9）组成的非负整数 N ，现要将其中的 m （ $m < n$ ）个数字移除。设计算法，找出移除这 m 个数字后，剩下的数字组成的数中的最小数。要求算法的时间复杂度为 $O(n)$ 。

例如：若非负整数为 1432219， m 为3。任意去掉3个数字后可能得到很多的数，如1432、4322、2219、1219、...，其中1219这个数最小。

【6】 设计算法，将一个形如 m/n 的真分数分解为若干个分子为1的分数之和。例如：

$5/8$ 可分解为 $1/2 + 1/8$

$7/8$ 可分解为 $1/2 + 1/3 + 1/24$

$9/10$ 可分解为 $1/2 + 1/3 + 1/15$

$10/11$ 可分解为 $1/2 + 1/3 + 1/14 + 1/231$

(可利用Python中的fractions模块)

【7】 对于一个长度为 n 的元素序列 S ，在 S 的若干适当的位置插入 S 中已有的元素，可使 S 成为回文序列。例如，序列：

[1, 2, 2, 1]

已经是回文，不需插入（插入0个）元素。序列

[1, 2, 3, 2, 5]

在插入2个元素后，成为回文：

[1, 5, 2, 3, 2, 5, 1]

序列：

[1, 2, 5, 1, 2, 4]

在插入3个元素后，成为回文：

[1, 4, 2, 5, 1, 5, 2, 4, 1]

试编写非递归算法，计算对于任意序列 S ，至少需要在 S 中插入多少个元素可使 S 变成一个回文序列。

递归版本

```
def minAdd( A, a, b ):
    if a >= b :
        return 0
    elif A[a] == A[b]:
        return minAdd(A, a+1, b-1)
    else :
        return min(minAdd(A, a+1, b), \
                    minAdd(A, a, b-1)) + 1
```

【8】 将一个整数 N 拆分为若干个数 n_1 、 n_2 、...、 n_m ，满足 $n_1+n_2+\dots+n_m=N$ 。问：如何拆分，可以使得 $n_1 \times n_2 \times \dots \times n_m$ 为最大。例如，若 N 为8，则将8拆分为2、3、3可以使得 $2 \times 3 \times 3 = 18$ 为最大（对于有些数，比如3，其本身大于任何拆分，所以不需进行拆分）。试编写非递归算法求解，算法的输入为 N （如8），输出为拆分结果（2, 3, 3）及连乘积（18）。

递归版本

```
def maxMulti(n):  
    if n <= 3: # 不拆分大于等于拆分  
        return n, str(n)  
    else : #  $f(n) = \max\{f(i)*f(n-i)\}$   
        maxs = 0  
        for i in range(1, int(n/2)+1):  
            a1, b1 = maxMulti(i)  
            a2, b2 = maxMulti(n-i)  
            if maxs < a1*a2 :  
                maxs = a1*a2  
                s = b1 + ',' + b2  
        return maxs, s
```