车厢重组

在一个旧式的火车站旁边有一座桥，其桥面可以绕河中心的桥墩水平旋转。 一个车站的职工发现桥的长度最多能容纳两节车厢，如果将桥旋转180度，则可以把相邻两节车厢的位置交换，用这种方法可以重新排列车厢的顺序。 于是他就负责用这座桥将进站的车厢按车厢号从小到大排序。 他退休后，火车站决定将这一工作自动化，其中一项重要的工作是编一个程序，输入初始的车厢顺序，计算最少用多少步就能将车厢排序。

输入文件

输入文件有两行数据，第一行是车厢总数N（不大于1000），第二行是N个不同的数表示初始的车厢顺序。

输出文件

一个数据，是最少的旋转次数。

输入样例

```
4
4 3 2 1
```

输出样例

```
6
```

参考代码

```
#include <iostream>
using namespace std;

int n,a[10005],ans=0;
int main(){
    cin >> n;
    for(int i=1;i<=n;i++)
        cin >> a[i];
    for(int i=1;i<=n-1;i++){
        for(int j=1;j<=n-i;j++){
            swap(a[j],a[j+1]);
            ans++;
        }
    }
    cout << ans << endl;

    return 0;
}
```

逆序对与排序的稳定性

逆序对

上题的实质是求**逆序对**。

对于一个数列，如1,2,7,4,3,8,9,5,6共9个数， 其<序号，数值>对<i,a_i>是<1,1>, <2,2>, <3,7>, <4,4>, <5,3>, <6,8>, <7,9>, <8,5>, <9,6>

所谓的逆序对是，对于<i,a_i>和<j,a_j>而言，当i<j时$a_i > a_j$。

对于上例而言:

1. <1,1>没有逆序对
2. <2,2>没有逆序对
3. <3,7>有4个，分别是<4,4>, <5,3>, <8,5>, <9,6>
4. <4,4>有1个，<5,3>
5. <5,3>没有逆序对
6. <6,8>有2个，分别是 <8,5>, <9,6>
7. <7,9>有2个，分别是 <8,5>, <9,6>
8. <8,5>没有逆序对
9. <9,6>没有逆序对

上例共有9个逆序对

冒泡排序的本质是消除逆序对，进而达到数值对位的目的。 用冒泡排序的交换次序可以统计出逆序对的个数，但其计算复杂度是$O(n^2)$相对较慢。

对于归并排序来说，当两个元素相等时并不交换位置，只有真正逆序时交换位置，并且其计算复杂度是$O(n\log^2 n)$， 相对较快，所以求逆序对及其个数可以用归并排序实现。

排序的稳定性

当数值大小相等时，不能使其初始相对位置改变的排序是稳定排序， 如数组排序中的：冒泡排序，插入排序，归并排序，计数排序，桶排序和基数排序。 可以用algorithm包中的stable_sort函数实现。

当数值大小相等时，不能保证其初始相对位置不变的排序是非稳定排序， 如数组排序中的：选择排序，堆排序，快速排序和希尔排序。 可以用algorithm包中的sort函数实现。

algorithm中stable_sort和sort函数可以给出排序结果，但统计逆序对的工作仍需采用传统的排序算法自己完成。

用归并排序统计逆序对

```
#include <iostream>
using namespace std;

int a[100],r[100],ans=0,n;

void merge(int s,int t){
    if(s==t) return ;
    int mid = (s+t)/2;
    merge(s,mid);
    merge(mid+1,t);

    int left=s,right=mid+1,p=s;
    while(left<=mid && right<=t){
        if(a[left]<=a[right]){
            r[p]=a[left];
            p++;
            left++;
        }else{
            r[p]=a[right];
            p++;
            right++;
            ans = ans + mid-left+1; //统计逆序对，跨域了(left,mid]之间的元素个数
        }
    }
    while(left<=mid){
            r[p]=a[left];
            p++;
            left++;
    }
    while(right<=t){
        r[p]=a[right];
        p++;
        right++;
    }
```

```
    for(int i=s;i<=t;i++)
        a[i] = r[i];
}

int main(){
    cin >> n;
    for(int i=0;i<n;i++)
        cin >> a[i];

    merge(0,3);

    cout << ans << endl;
    cout << a[0];
    for(int i=1;i<n;i++)
        cout  << "," << a[i] ;
    cout << endl;

    return 0;
}
```

Exercise

1. 谁考了第k名，http://noi.openjudge.cn/ch0110/01/
2. 奇数单增序列，http://noi.openjudge.cn/ch0110/02/
3. 成绩排序，http://noi.openjudge.cn/ch0110/03/
4. 奖学金，http://noi.openjudge.cn/ch0110/04/
5. 分数线划定，http://noi.openjudge.cn/ch0110/05/
6. 整数奇偶排序，http://noi.openjudge.cn/ch0110/06/
7. 合影效果，http://noi.openjudge.cn/ch0110/07/
8. 病人排队，http://noi.openjudge.cn/ch0110/08/
9. 明明的随机数，http://noi.openjudge.cn/ch0110/09/
10. 单词排序，http://noi.openjudge.cn/ch0110/10/
11. 出现次数超过一半的数，http://noi.openjudge.cn/ch0113/28/
12. 统计字符数，http://noi.openjudge.cn/ch0113/29/

参考文献

1. 董永建，信息学奥数一本通（C++）第五版。
2. http://noi.openjudge.cn

> 辽师张大为@https://daweizh.github.io/csp/