逆序对

什么是逆序对

- 序列中，i < j，且 ai > aj

怎么求逆序对

- O(n^2)暴力枚举
- O(nlogn)分治求
- O(nlogn)树上前缀和统计求

冒泡排序

数组归并

- 在学习归并排序之前，先了解一下数组归并操作
- 简单来说，就是两个有序的数组A、B（可能不等长），通过逐对比较的方式，合并成有序数组C

数组归并

```
while (i <= n && j <= m)      //a、b均未读完
{
    if (a[i] <= b[j]) c[k++] = a[i++];      //归并处理
        else c[k++] = b[j++];
}
while (j <= m) c[k++] = b[j++];     //a数组已读完
while (i <= n) c[k++] = a[i++];     //b数组已读完
```

归并排序

- 归并排序是另一种基于分治思想的排序方法，其基本思想如下：

1. 将数列划分为两半，并一直递归下去直到无法再分
2. 对相邻的两个子序列执行数组归并操作，这样保持归并后长度叠加的数组有序
3. 如此一直往上返回，直到返回成原数组，原数组也是有序的了
4. 要注意归并排序在递归中的处理与返回之间的先后关系，和快速排序刚好相反

- 归并排序在分治思想的体现上，比快速排序更典型
- 其复杂度也是O(nlogn)的，并且是一种稳定的排序方式

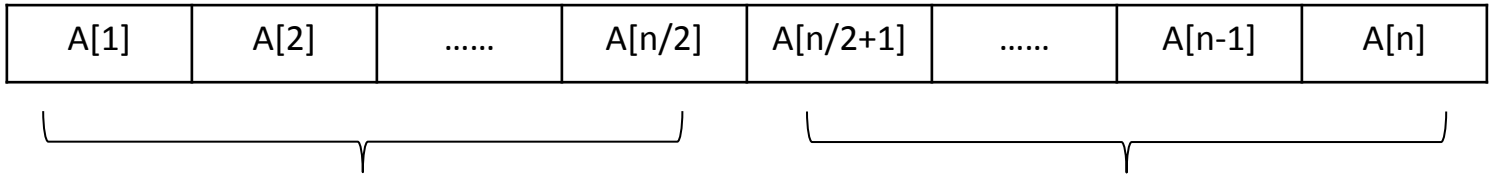
归并排序

逆序对

- 简单来说，如果数列中对于$i<j$，存在$a_i>a_j$，则称(a_i,a_j)构成一对逆序对
- 比如数列：1、2、5、8、10，就没有逆序对
- 而数列：1、2、10、5、8，就存在（10,5）、（10,8）两对逆序对

逆序对

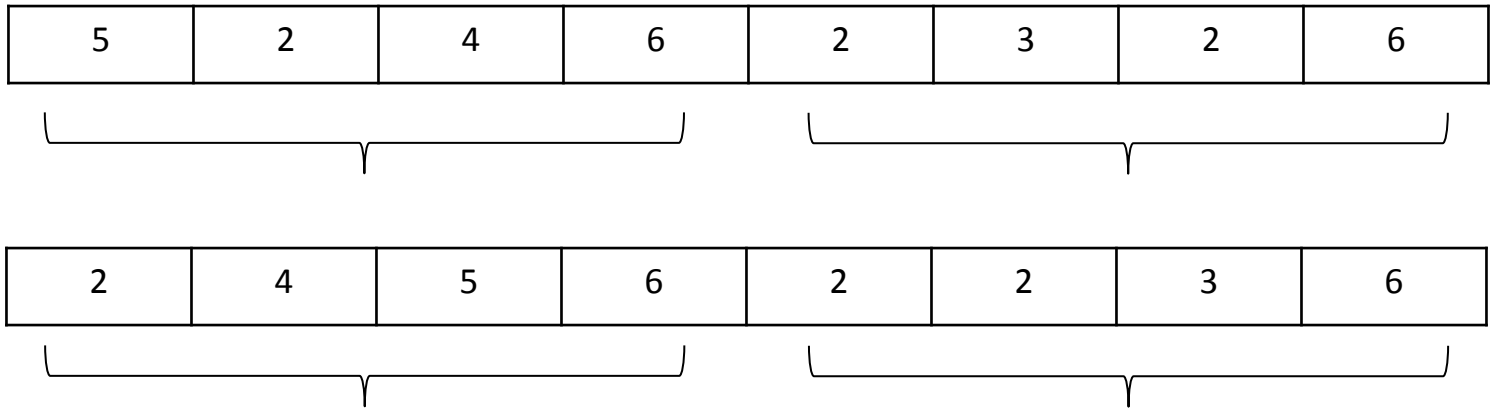
- 逆序对怎么求？
- 暴力的做法是冒泡排序，记录交换次数，复杂度自然是$O(n^2)$的。有什么更好的算法呢？



- 设f[L,R]表示序列[L,R]的逆序对个数，M = (L+R)/2
- 则f[L,R] = f[L,M]+f[L,M,R]+f[M+1,R]
- 其中f[L,M,R]表示跨越两段间（即一个数在左段，一个数在右段）的逆序对个数。

逆序对

- 因为是递归进行的，所以主要的难点在于统计跨区间的逆序对数
- 但是如果对左右两段序列分别排好序的话，要统计f[L,M,R]就比较容易了：



- 在左半序列中的4，5，6与右半序列中的2，2，3构成了“逆序对”，因此跨区间构成的逆序对个数为3+3+3=9。

逆序对

- 对左右两段序列的（递归）排序，会在事实上打乱原序列，这会影响最终答案的正确性吗（因为要问的是原序列的逆序对个数）？
- 现在大家知道为什么是归并排序而不是同样基于分治思想的快速排序被用来做这件事呢？
- 因为按照之前的统计方案，无论是计算f[L,M] / f[M+1,R]，还是f[L,M,R]，我们都需要二分原序列并分别排序，前者自然是递归进行，而后者依赖于二分后的序列有序，因此在递归返回合并答案（就是数组归并操作）的过程中同步统计，一切与归并排序刚好合拍
- 求逆序对有两种主要算法：除了归并排序外还有一种树状数组

逆序对

```
i = k = L; j = mid + 1;
while (i <= mid && j <= R)
{
    if (a[i] <= a[j]) c[k++] = a[i++];
        else c[k++] = a[j++], ans += mid - i + 1;    //统计逆序对
}
```

离散化

- 离散化的思路其实就是排序，将原本十分大的数变小，而其相对大小关系仍然保持不变

- 比如：100、4399、12580、10086、9527
- 其大小关系编号为：1、2、5、4、3

离散化

```
struct node
{
    int v, p;
 } a[N];

int cmp(node x, node y)
{
    return x.v < y.v;
}
```

离散化

```
for (int i = 1; i <= n; ++ i)
{
    scanf("%d", &a[i].v);
    a[i].p = i;
}
sort(a + 1, a + n + 1, cmp);
for (int i = 1; i <= n; ++ i)
    printf("%d ", a[i].p);
```

求逆序对

- 我们已经知道归并排序可以求逆序对，复杂度O(nlogn)

- 树状数组也可以求逆序对，复杂度也是O(nlogn)

求逆序对

- 先来看这样一个例子：
- 已知数列：5、4、3、2、1，其逆序对数为：1+2+3+4 = 10

- 那么怎么求逆序对呢？
- 比如我们要求序列：a、b、c、d中的逆序对，我们可以这样做：

1. 先求出序列a、b、c中的逆序对数sum1
2. 然后再求出序列a、b、c中比d大的数的个数sum2
3. 那么a、b、c、d中的逆序对数即为sum1 + sum2

树状数组求逆序对

- 但是这跟树状数组有什么关系？

- 我们知道树状数组可以在O(logn)的时间内求区间[1，R]中的元素和

树状数组求逆序对

- 所以开一个大小为这些数中最大值的树状数组，并全部置 0
- 然后以数字为下标，每读入一个新的数就让它对应数字下标的数增加1，代表下标在当前已处理的数字中出现的次数

- 比如：读入：9、1、0、5、4
- 那么我们需要建立如下的树状数组c：

i	0	1	2	3	4	5	6	7	8	9
c[i]	1	1	0	0	1	1	0	0	0	1

树状数组求逆序对

- 考虑到空间浪费的问题，所以这里我们需要离散化处理
- 比如：读入：100、4399、12580、10086、9527
- 我们会离散化处理成：

i	1	2	3	4	5
c[i]	1	1	1	1	1

树状数组求逆序对

- 对于每个当前的数，已加入的数字个数（算当前数）即为当前数字在数列中的下标
- 也就是树状数组中已经加入了这么多个数，那么他的前缀和代表小于它且在他前面的数的个数
- 用总数减掉前缀和即为以该数为较小数的逆序对数

树状数组求逆序对

```
for (int i = 1; i <= n; ++ i)
{
    x = a[i].p;
    while (x <= 10000)    //建树状数组
    {
        ++ c[x];
        x += x & (-x);
    }
    x = a[i].p;
    int sum = 0;
    while (x)
    {
        sum += c[x];
        x -= x & (-x);
    }
    ans += i - sum;
    //当前下标减掉当前数字的前缀和即为以该数为较小数的逆序对个数
}
```

火柴排队

- 有a、b两列火柴，每列n（n≤10w）根，每根火柴有一个高度ai、bi。两列之间的距离定义为$\sum(ai-bi)^2$。现在求最少需要交换几根火柴使得该距离最小（只能每列内部交换）

火柴排队

- 先看公式$\sum(ai-bi)^2$，$(ai-bi)2=ai^2+bi^2-2ai\times bi$
- 要使距离和最小，就是每个距离都尽量小，即$ai\times bi$尽量大
- 就是让最大的和最大的乘，次大的和次大的乘
- 那么交换一组还是同时交换两组呢？