

# 逆序对

**Property:** 逆序对对数等于序列通过交换相邻两数变成全排列的交换次数。

**Idea:**

- method 1: 在归并排序的 merge 中, 每次取出右边元素时统计左边未取出的元素数量, 累和。
- method 2: 离散化之后, 开值域树状数组, 按逆序插入树状数组时, 统计已插入的元素中比当前元素小的数量。

**Complexity:**

- method 1:  $O(n \lg n)$
- method 2:  $O(n \lg n)$

**Code (Mergesort):**

```
1 void mergesort(int l, int r){
2     if(l >= r) return;
3     int mid = (l + r) >> 1;
4     mergesort(l, mid);
5     mergesort(mid+1, r);
6     id = 0;
7     int lpt = l, rpt = mid+1;
8     while(lpt <= mid && rpt <= r){
9         if(a[lpt] <= a[rpt])
10            t[++id] = a[lpt++];
11        else{
12            t[++id] = a[rpt++];
13            cnt += 1ll * mid - lpt + 1; // cnt is the number of
inversions
14        }
15    }
16    while(lpt <= mid)
17        t[++id] = a[lpt++];
18    while(rpt <= r)
19        t[++id] = a[rpt++];
20    for(int i = l; i <= r; i++)
21        a[i] = t[i - l + 1];
22 }
```

## Code (BIT):

```
1  int c[N];
2  inline int lowbit(int x){
3      return x & -x;
4  }
5  void add(int x, int val){
6      while(x <= 'z'){
7          c[x] += val;
8          x += lowbit(x);
9      }
10 }
11 inline int sum(int x){
12     int res = 0;
13     while(x){
14         res += c[x];
15         x -= lowbit(x);
16     }
17     return res;
18 }
19 int cntInverse(int a[]){ // a[] here is already discretized
20     int res = 0;
21     memset(c, 0, sizeof c);
22     for(int i = n; i >= 1; i--){
23         res += sum(a[i]-1);
24         add(a[i], 1);
25     }
26     return res;
27 }
```