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)：

```
void mergesort(int l, int r){
    if(l >= r)  return;
    int mid = (l + r) >> 1;
    mergesort(l, mid);
    mergesort(mid+1, r);
    id = 0;
    int lpt = l, rpt = mid+1;
    while(lpt <= mid && rpt <= r){
        if(a[lpt] <= a[rpt])
            t[++id] = a[lpt++];
        else{
            t[++id] = a[rpt++];
            cnt += 1ll * mid - lpt + 1; // cnt is the number of inversions
        }
    }
    while(lpt <= mid)
        t[++id] = a[lpt++];
    while(rpt <= r)
        t[++id] = a[rpt++];
    for(int i = l; i <= r; i++)
        a[i] = t[i - l + 1];
}
```

Code (BIT)：

```
int c[N];
inline int lowbit(int x){
    return x & -x;
}
void add(int x, int val){
    while(x <= 'z'){
        c[x] += val;
        x += lowbit(x);
    }
}
inline int sum(int x){
    int res = 0;
    while(x){
        res += c[x];
        x -= lowbit(x);
    }
    return res;
}
int cntInverse(int a[]){ // a[] here is already discretized
```

```
    int res = 0;
    memset(c, 0, sizeof c);
    for(int i = n; i >= 1; i--){
        res += sum(a[i]-1);
        add(a[i], 1);
    }
    return res;
}
```