前缀和/差分

对于一个序列 $a[1...n]$，定义前缀和数组 $s[i]=\sum_{j\le i}a[j]$ 和差分数组 $d[i]=a[i]-a[i-1]$：

```
s[i] = a[1] + a[2] + ... + a[i] = s[i-1] + a[i]
d[i] = a[i] - a[i-1]
```

可以看到，前缀和和差分互为逆运算，将差分数组 d 做前缀和、或者将前缀和数组 s 做差分，都能得到原数组 a 。同理，对于二维数组可以定义二维前缀和/二维差分及其递推（基于容斥原理）：

```
s[i][j] = s[i][j-1] + s[i-1][j] - s[i-1][j-1] + a[i][j]
a[i][j] = s[i][j] + s[i-1][j-1] - s[i][j-1] - s[i-1][j]
d[i][j] = a[i][j] + a[i-1][j-1] - a[i][j-1] - a[i-1][j]
```

任意子矩阵 $(r1,c1)-(r2,c2)$ 的和 = `s[r2][c2] - s[r1-1][c2] - s[r2][c1-1] + s[r1-1][c1-1]`

一个常见的转化思路是**把原数组上的区间修改，视为其差分数组上的2个单点修改**:

- 一维 (l,r,x) 修改：`d[l]+=x, d[r+1]-=x`
- 二维 $(r1,c1,r2,c2,x)$ 修改：`d[r1][c1]+=x, d[r1][c1+1]-=x, d[r2+1][c1]-=x, d[r2+1][c2+1]+=x`

树上前缀和和树上差分留到后面再讲。

前缀和时常用于优化枚举、优化DP转移等。

双指针模型

又称**尺取法**，用于解决在序列上选取最优连续子区间/统计合法的子区间的问题。

- 设 $f[l]=$ 以 l 为左端点的区间的**最优右端点（或者满足限制的最小有端点）**

考虑 l 从 $1\to n$ 变化时，若满足以下条件:

1. $f[l]$ 单调变化
2. 删除左端点 l、加入右端点 r 同时可以快速检查最优性/合法性（莫队信息）

那么可以使用双指针在 $O(n)$ 时间内求出所有 $f[1\sim n]$ 的所有值，一般使用 `for` 里面嵌套 `while` 来实现。

```
bool ok(int l, int r);//判断区间[l,r]合法性
void add(int i);//加入第i个元素
void del(int i);//删除第i个元素
for(int l=1,r=0;l<=n;++l){
    while(r<n && ok(l,r)==0) add(++r);//推右指针
    if(ok(l,r)) f[l] = r;//更新答案
    del(l);
}
```

序列统计量

子段和

预处理前缀和，可以 $O(1)$ 查询子段和

利用分治结构（维护区间最大前后缀+最大子段和）可以快速查询区间最大子段和，支持修改

子段最值

利用单调栈可以快速处理出以某个 r 结尾的所有子段最值

可以使用链表快速处理出前驱/后继信息，从而获知每个 $a[i]$ 在哪些区间能成为最值

中位数

中位数定义为排序之后的第 $\lfloor \frac{n}{2} \rfloor$ 个元素。

对于序列中的某个数 x，若 $<x, =x, >x$ 的数分别 $n1, n2, n3$ 个，那么 x 能成为中位数的充要条件为 $n1+n2 \geq n3 \wedge n1 \leq n2+n3$。

中位数：$min\{x | \sum_i [a[i] \leq x] \geq \lceil n/2 \rceil\}$，在从小->大加数过程中恰好达到 $\lfloor \frac{n}{2} \rfloor$ 个数的时刻

区间中位数（区间第 k 大）主席树上二分，1log。

中位数和所谓**【山区建小学】**模型息息相关，对于 **递增序列** $d[1...n]$，确定整数 x 使得下面式子取到最小值：

- $ans = \sum_{i=1}^{n} |d[i] - x|$

最优的 x 应取 $d[1, 2, \ldots, n]$ 的**中位数**，可以从函数最值、调整法等多种方向理解。需要对形如 $\sum_{i=1}^{n} |d[i] - x|$ 或者 $\sum_{i=1}^{n} |d[i] - d[x]|$ 的式子非常敏感。

可以使用**对顶堆**的方法动态维护中位数，如需支持删除则需要使用值域线段树。

众数

区间众数查询离线用莫队，在线用分块（见P4168 [Violet] 蒲公英）。

如果是绝对众数（出现次数严格大于一半），可以使用摩尔投票法支持合并合并（维护众数和抵消之后的出现次数），并且利用分治结构支持区间查询（见P8496 [NOI2022] 众数）。

mex

区间mex查询用记录每个值最后一次出现下标 $last[x]$ 的值域线段树，线段树（维护区间min）上二分查询出现下标 $last[x] < l$ 的最小 x 值。在线可以可持久化（见P4137 Rmq Problem / mex）

排序

3种简单排序：选择，冒泡，插入

01序列的冒泡排序就是把最靠前的 1 移动到末尾

对于排列上的某个位置 $x=p[i]$，令 $rev[x]=i$ 之前 $>x$ 的数字数量，即 $rev[p[i]]=\sum_{j<i}[p[j]>p[i]]$。若 $rev[x]>0$，那么在一趟冒泡排序中，肯定有且只有1个数从其前面跨越到其后面。也就是说：

- 一趟冒泡过后 $rev[x]=max(rev[x]-1,0)$
- 冒泡排序所需趟数为 $max\{rev[x]\}$

基于分治的排序：归并，快速排序

排列等价于若干置换环（$i\to p[i]$ 连边），排序的过程就是将其变为 n 个自环的过程

- 每次任意交换2个元素 `swap(p[i],p[j])`，将一个置换环分裂为两个（多一个环）；那么交换次数等于 $n-$ 初始环数