数据结构 2

黎伟诺

7.18.2024

给定一个长度为 n 的数列，q 次询问，每次询问一个区间的和？

令 $s_i = \sum_{j=1}^{i} a_j$。

递推：$s_i = s_{i-1} + a_i$。

回答询问：$\sum_{i=l}^{r} a_i = s_r - s_{l-1}$。

预处理时间复杂度 $\Theta(n)$，单次回答询问 $\Theta(1)$。

什么时候可以用前缀和？
运算满足可减性：加法、模质数意义下的乘法，异或等

给一个 $n \times m$ 的矩阵 A，每次询问 $\sum_{i=a}^{b} \sum_{j=c}^{d} A_{i,j}$

令 $s_{i,j} = \sum_{k=1}^{i} \sum_{m=1}^{j} a_{k,m}$。

递推：$s_{i,j} = s_{i,j-1} + s_{i-1,j} - s_{i-1,j-1} + a_{i,j}$。

回答询问：$\sum_{i=a}^{b} \sum_{j=c}^{d} a_{i,j} = s_{b,d} - s_{a-1,d} - s_{b,c-1} + s_{a-1,c-1}$。

预处理 $\Theta(nm)$，回答询问 $\Theta(1)$。

给一个数组 a，初始都为 0，有 q 次操作，每次让 $[l, r]$ 区间加 k，问最后数组里的数是多少？

令 $d_i = a_i - a_{i-1}$。那么 a_i 为 d_i 的前缀和数组。
对于一个修改 (l, r, k)，等价于 $d_l = d_l + k$，$d_{r+1} = d_{r+1} - k$。
每次修改 $\Theta(1)$，最后做一遍前缀和得到答案。

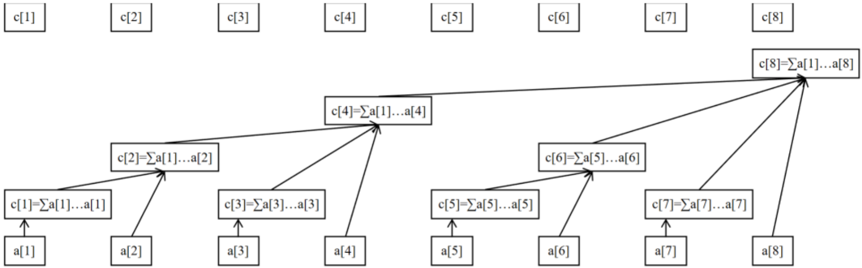
二维矩形加：类似于一维前缀和与一维差分的关系。

树状数组

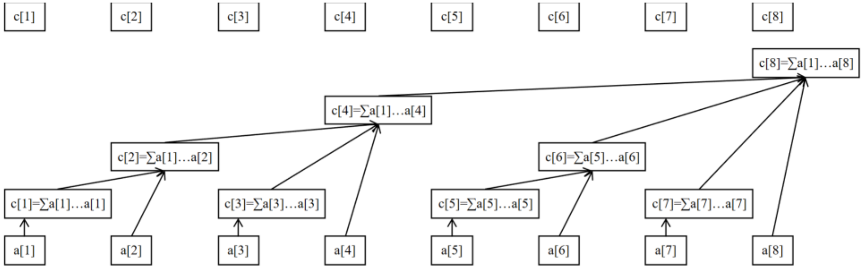
给一个长度为 n 的序列，每次操作：
将 a_x 修改为 $a_x + y$
询问 $\sum_{i=l}^{r} a_i$

树状数组

树状数组是一种树形数据结构，用于维护序列的前缀和。
每个前缀可以被拆分为 $\Theta(\log n)$ 个树上的节点，每个位置被包含在 $\Theta(\log n)$ 个节点中。



树状数组



令 $lowbit(x) = \max\{2^k | x \bmod 2^k = 0\}$，那么节点 i 储存 $[i - lowbit(i) + 1, i]$ 的信息。
特别的 $lowbit(x) = (x)\&(-x)$。

树状数组

我们发现，如果我们想查询 $\sum_{i=1}^{x} a_i$，那么我们只需要查询 c_x，$c_{x-lowbit(x)}$，$c_{x-lowbit(x)-lowbit(x-lowbit(x))}\cdots$。

```
int query(int x) {
    int ret = 0;
    while(x) {
        ret += c[x];
        x -= x & -x;
    }
    return ret;
}
```

树状数组

我们发现节点 x 的父亲为 $x + lowbit(x)$，那么我们只要将 x 到根一路更新即可。

```
void add(int x, int v) {
    while(x <= n) {
        c[x] += v;
        x += x & -x;
    }
}
```

树状数组

树状数组修改和询问时间复杂度均为 $\Theta(\log n)$。
优势：常数小，代码短。

- 要求支持区间加 k 和单点询问。

- 要求支持区间加 k 和单点询问。
- 用树状数组维护差分数组 d，那么就变成了单调修改和询问前缀和。

要求支持区间加 k，区间求和。

树状数组只能修改单点信息，维护前缀和，那么是否可以通过差分求出前缀和呢？

$d_i = a_i - a_{i-1}, s_i = \sum_{j=1}^{i} a_j$。

$s_i = (d_1) + (d_1 + d_2) + (d_1 + d_2 + d_3) + \ldots$

$s_i = \sum_{j=1}^{i} d_j(i - j + 1) = (i + 1) \cdot \sum_{j=1}^{i} d_j - \sum_{j=1}^{i} (j \cdot d_j)$

我们只要维护 d_i 和 $t_i = d_i \cdot i$ 即可！

树状数组

区间修改，区间查询

```
void add(int x, int v) {
    for(int i = x; i <= n; i += i & -i) {
        c[i] += v;
        d[i] += x * v;
    }
}

int qry(int x) {
    int ret = 0;
    for(int i = x; i; i -= i & -i) {
        ret += c[i] * (x + 1);
        ret -= d[i];
    }
    return ret;
}
```

```
void Add(int l, int r, int k) {
    add(l, k); add(r + 1, -k);
}

int Qsum(int l, int r) {
    return qry(r) - qry(l - 1);
}
```

单点加一，子矩阵求和。
$c_{i,j}$ 维护 $(i - lowbit(i) + 1, j - lowbit(j) + 1)$ 为左下角，(i, j) 为右上角的信息。
修改，询问时间复杂度为 $\Theta(\log^2 n)$。
需要注意空间复杂度为 $\Theta(nm)$。

```
int c[N][N];

void add(int x, int y, int v) {
    for(int i = x; i <= n; i += i & -i)
        for(int j = y; j <= m; j += j & -j)
            c[i][j] += v;
}

int qry(int x, int y) {
    int ret = 0;
    for(int i = x; i; i -= i & -i)
        for(int j = y; j; j -= j & -j)
            ret += c[i][j];
    return ret;
}
```

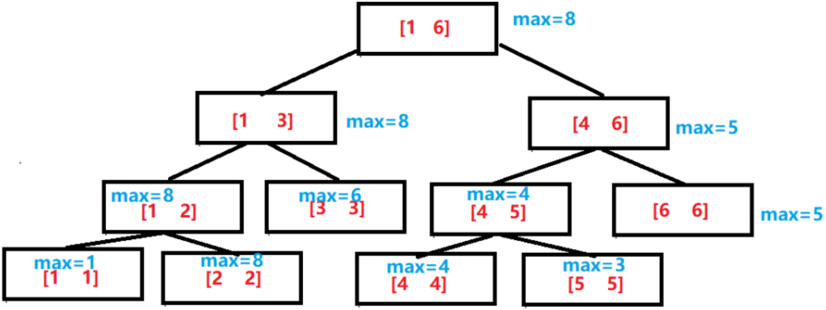
类似的，矩形加，单点询问；矩形加，矩形询问都可以用树状数组来解决

只要和一维同样思路解决即可

区间最值

单点修改，求区间最大值。
树状数组难以解决，因为不具有可减性。
但是最大值仍然有结合律，我们希望找一个数据结构，满足每个位置被尽量少的节点包含。
并且区间可以被表示成数据结构上尽量少的“块”。

线段树是一棵二叉树，维护了动态的分治结构。每个节点维护一个区间内 $[l,r]$ 的信息，左儿子区间维护 $[l,\lfloor\frac{l+r}{2}\rfloor]$ 的信息，右儿子维护 $[\lfloor\frac{l+r}{2}\rfloor+1,r]$ 的信息。



根据线段树的性质，每个节点维护的值可以由儿子推出来。

```
struct T{
    int l, r;
    int ls, rs;
    int mx;
}t[N * 2 + 1];

void update(int x) {
    t[x].mx = max(t[t[x].ls].mx, t[t[x].rs].mx);
}
```

```
int build(int l, int r) {
    int x = ++cnt;
    t[x].l = l, t[x].r = r;
    if(l == r) {
        t[x].mx = a[l];
        return x;
    }
    int mid = (l + r) >> 1;
    t[x].ls = build(l, mid);
    t[x].rs = build(mid + 1, r);
    update(x);
    return x;
}
```

时间复杂度 $\Theta(\log n)$。

```
void change(int x, int p, int v) {
    if(t[x].l == t[x].r) {
        t[x].mx = v;
        return;
    }
    int mid = (t[x].l + t[x].r) >> 1;
    if(p <= mid)
        change(t[x].ls, p, v);
    else
        change(t[x].rs, p, v);
    update(x);
}
```

时间复杂度 $\Theta(\log n)$。怎么分析？

```
int query(int x, int l, int r) {
    if(l <= t[x].l && t[x].r <= r)
        return t[x].mx;
    int mid = (t[x].l + t[x].r) >> 1;
    int ret = 0;
    if(l <= mid)
        ret = max(ret, query(t[x].ls, l, r));
    if(r > mid)
        ret = max(ret, query(t[x].rs, l, r));
    return ret;
}
```

要求支持区间加 k，区间求和？
如果暴力进行单点修改，每次修改时间复杂度高达 $\Theta(n \log n)$。
考虑借鉴查询对区间的操作？

对线段树上每个节点维护一个 lazy tag，代表这个区间内都被进行了一次加操作。只有用到这个节点的子节点时，才会将标记下传。

```
struct T{
    int l, r, ls, rs;
    int sum, tag;
}t[N * 2 + 1];

void Add(int x, int v) {
    t[x].sum += v * (t[x].r - t[x].l + 1);
    t[x].tag += v;
}

void pushdown(int x) {
    Add(t[x].ls, t[x].tag);
    Add(t[x].rs, t[x].tag);
    t[x].tag = 0;
}
```

注意修改查询操作在递归之前都需要 pushdown。

```
void change(int x, int l, int r, int v) {
    if(l <= t[x].l && t[x].r <= r) {
        Add(x, v);
        return;
    }
    int mid = (t[x].l + t[x].r) >> 1;
    pushdown(x);
    if(l <= mid)
        change(t[x].ls, l, r, v);
    if(r > mid)
        change(t[x].rs, l, r, v);
    update(x);
}
```

要求支持区间加 k，区间乘 m，区间求和。

维护一个标记，每个标记是一个 pair，(a, b) 代表原来的数为 x，现在为 $ax + b$。

如何合并标记？

$(a \cdot x + b) \cdot c + d = (ac) \cdot x + b \cdot c + d$

维护一个 01 序列，要求支持区间翻转，区间求和。
区间翻转即 0 变成 1，1 变成 0。

维护区间内是否翻转。

```
void Rev(int x) {
    t[x].rev ^= 1;
    t[x].sum = t[x].r - t[x].l + 1 - t[x].sum;
}
```

支持区间 and k，区间 or k，区间 xor k，区间求和。

每一位独立，分别用一棵线段树维护。维护区间内有多少个 1

and/or 操作相当于区间赋值，xor 操作相当于区间翻转

线段树

如何正确的在线段树/树状数组上二分？

Things I don't know

By Um_nik, history, 3 years ago,

If you know at least 3 of these things and you are not red — you are doing it wrong. Stop learning useless algorithms, go and solve some problems, learn how to use binary search.

树状数组：按位从高往低枚举决定这一位的 1 放不放
线段树：类似区间查询的手法，先看目前的整个区间能不能放，假如能就返回；
不能就递归左半边，如果左半边放的是整个区间才递归右半边

中位数

P1168

给定一个长度为 N 的非负整数序列 A，对于前奇数项求中位数。

树状数组做法：
先将数组离散化使得 a_i 变成 $[1, n]$ 的范围，然后建关于权值的树状数组，每次 $add(a_i, 1)$
树状数组每次二分前缀和小于 $\frac{i+1}{2}$ 的最大位置

中位数
P1168

对顶堆做法:
一个大根堆 $H1$ 存较小部分的值，一个小根堆 $H2$ 存较大部分的值
加入的值假如小于等于 $H1$ 最大值就加到 $H1$ 否则加入到 $H2$
每次做完之后把 $H2$ 的元素调整到 $H1$ 或者 $H1$ 的元素调整到 $H2$ 使得 sz 差为 1
$H1$ 的堆顶就是答案

二维数点

给定平面上 n 个点 (x_i, y_i)，对于每个 i 询问有多少个 j 满足 $x_j \le x_i, y_j \le y_i$。

二维数点

思路：一维排序扫描，另一维数据结构维护。
我们按照 x 排序并且从小到大扫描，当我们扫描到 $x = a$ 时，维护序列 c_t 代表有多少个 (x_i, y_i) 满足 $x_i \le a, y_i = t$。
问题转化为单点加一，前缀求和问题。

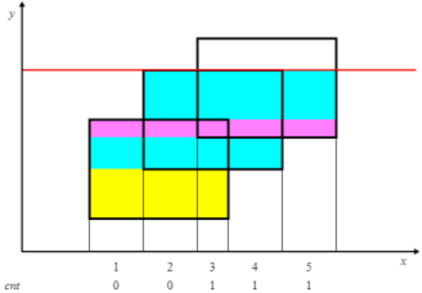
扫描线

P3755

给你平面上 n 个点，每个点有点权，询问 m 个矩形内部的点权和。

按照 y 扫描将矩形差分为上边界和下边界相减，再差分之后就变成若干个查询左下角点权和，然后就和上一题类似。

给出平面上 n 个矩形，求这些矩形面积的并。



扫描线，扫描 x 一维，维护有多长的 y 被覆盖了，等价于区间加，有多少位置 > 0。
不好维护，考虑反过来，维护有多少位置 $= 0$，发现直接维护全局最小值和最小值出现的次数即可。

给出平面上 n 条水平或竖直的线段，保证平行的线段不会相交，求他们的交点数量。

扫描 y 轴，那么我们对于一个竖直的线段，我们在 y_{min} 将其加入，将 c_x+1，再 y_{max} 将其删除，将 c_x-1，那么对于一条水平的线段，我们只需要在 y 询问 $[x_{min}, x_{max}]$ 的和即可。

给出序列 a，每次询问区间 $[l, r]$ 有多少种不同的数。

将询问按 r 排序，枚举到 r，并且维护所有左端点的答案
每个数只在第一次出现计算贡献，我们考虑 a_r 的贡献，那么 $[last_{a_r}+1, r]$ 区间内 a_r 是第一次出现的，再往前的之前已经在 $last_{a_r}$ 计算过贡献。
问题转化为区间加一，单点求值。

给你平面上 n 个点，对每个点计算出其他点到它的曼哈顿距离 $\|x_1 - x_2\| + \|y_1 - y_2\|$ 的距离最小值、最大值或者和。

每个点考虑 180° 到 270° 的点，这些点到它的距离是可以把绝对值拆掉的

贡献是 $(-x_1+y_1)+(-x_2+y_2)$，所以需要把 x_1+y_1 添加到权值树状数组的 y_1 位置

按 x 从小往大加，按 y 值查树状数组里的区间和

魔卡少女

GDKOI2016D1T1

君君是中山大学的四年级学生。有一天在家不小心开启了放置在爸爸书房中的一本古书。于是，君君把放在书中最上面的一张牌拿出来观摩了一下，突然掀起一阵大风把书中的其她所有牌吹散到各地。这时一只看上去四不像的可爱生物"封印之兽"可鲁贝洛斯从书中钻了出来，它告诉君君书中的牌叫"库洛牌"，现在散落各地已实体化，要君君将它们全部再次封印起来，以免危害世界，于是君君开始过上了收服"库洛牌"的旅程。 经过不懈努力，君君集齐了N 张库洛牌，最后的审判就要来临，为了战胜审判者月，君君开始研究起这N 张库洛牌的魔法效果。君君已经将N 张库洛牌从左到右依次排列好，这N 张库洛牌的魔法值从左到右依次为a1; a2; a3; ...aN。她将告诉你这N 张库洛牌的魔法值。在最后的审判时，审判者月将会选择一个区间进行PK，君君预测了可能进行PK 的若干区间，她想请你帮助她计算这些区间的魔法效果，以便她更好地布置战术。一个区间内，所有连续子序列都会产生魔法效果。一个连续子序列p1; p2; p3; ...; pk 的魔法效果定义为p1^p2^p3^...^pk(^表示异或）。一个区间的魔法效果定义为所有连续子序列的魔法效果的和。例如有5 张库洛牌，魔法值为1; 1; 2; 4; 5，询问区间[2; 4] 的魔法效果。区间[2; 4] 包含的连续子序列为"1"; "2"; "4"; "1; 2"; "2; 4"; "1; 2; 4", 它们的魔法值分别为1; 2; 4; 3; 6; 7，所以区间[2,4] 的魔法效果为1 + 2 + 4 + 3 + 6 + 7 = 23。

库洛牌的魔法效果狂拽炫酷吊炸天，这个值可能很大，所以你只需要输出这个值模100,000,007。另外，任性的君君可以在询问的过程中对库洛牌的魔法值进行修改。 现在，君君给出了M 个操作，操作格式如下： 1. M p x 表示将第p 张库洛牌的魔法值修改为x。 2. Q l r 表示询问区间[l; r] 的魔法效果。 Pascal 语言中，异或操作符为xor，C++ 语言中，异或操作符为^。

$$N, M \leq 10^5, a_i, x \leq 1000$$

魔卡少女

GDKOI2016D1T1

因为 ai<=1000，我们可以拆位处理。拆成 10 个二进制位，每位开 1 棵线段树。

对于每个节点，维护：

d：这段区间的异或和

L[0]，L[1]：子区间一定从左端点开始，异或和为 0，1 的子区间分别有多少个

R[0]，R[1]：子区间一定从右端点开始，异或和为 0，1 的子区间分别有多少个

s[0]，s[1]: 异或和为 0，1 的子区间分别有多少个

蒟蒻 HansBug 在一本数学书里面发现了一个神奇的数列，包含 N 个实数。他想算算这个数列的平均数和方差。

输入格式

第一行包含两个正整数 N, M，分别表示数列中实数的个数和操作的个数。

第二行包含 N 个实数，其中第 i 个实数表示数列的第 i 项。

接下来 M 行，每行为一条操作，格式为以下三种之一：

操作 1：`1 x y k`，表示将第 x 到第 y 项每项加上 k，k 为一实数。
操作 2：`2 x y`，表示求出第 x 到第 y 项这一子数列的平均数。
操作 3：`3 x y`，表示求出第 x 到第 y 项这一子数列的方差。

方差

P1471

平均数需要维护区间和
方差需要维护区间平方和

区间

P1712

在数轴上有 n 个闭区间从 1 至 n 编号，第 i 个闭区间为 $[l_i, r_i]$ 。

现在要从中选出 m 个区间，使得这 m 个区间共同包含至少一个位置。换句话说，就是使得存在一个 x ，使得对于每一个被选中的区间 $[l_i, r_i]$，都有 $l_i \leq x \leq r_i$ 。

对于一个合法的选取方案，它的花费为被选中的最长区间长度减去被选中的最短区间长度。

区间 $[l_i, r_i]$ 的长度定义为 $(r_i - l_i)$ ，即等于它的右端点的值减去左端点的值。

求所有合法方案中最小的花费。如果不存在合法的方案，输出 -1。

区间

P1712

首先分析一下题目，l,r 这么大很显然就是要离散化了。
既然是跟区间长度有关，那我们不妨就先按区间长度排个序好了，反正这样子并不会影响答案。
然后我们思考一种最朴素的做法，那就是按排序后的顺序逐一加入区间，然后看看是否有一个点的被覆盖次数 $>=$m。
如果有的话那就统计一下答案，然后将前面加入的按顺序删掉，直到 $<$m。（双指针）
那么问题就是我们如何快速地得知是否有一个点的被覆盖次数 $>=$m。
那就很显然维护一棵线段树就好了。

区间斐波那契

一个数列 a_i，支持区间 $+k$，区间求 $\sum F_{a_i}$，F 为斐波那契数列
答案对 10^9+9 取模

矩阵乘法方法？
模数有点特殊，注意到 5 是关于 10^9+9 的二次剩余（什么注意力惊人）
$F_n = \frac{\alpha^n - \beta^n}{\sqrt{5}}$
对 α 的幂和 β 的幂分别维护，相当于区间乘法，区间加法

给出一个长度为 $n \le 10^5$ 的数组。你需要将其分为几段连续的子区间，每一段区间的权值为：
如果区间和大于 0，则为区间长度
如果区间和等于 0，则为 0
如果区间和小于 0，则为区间长度的相反数

$O(n^2)$ 的暴力很好想

```
for (int i = 1; i <= n; i++)
    pre[i] = pre[i - 1] + a[i];
for (int i = 1; i <= n; i++) {
    f[i] = -INF;
    for (int j = 0; j < i; j++) {
        if (pre[i] - pre[j] > 0)
            f[i] = max(f[i],f[j] + (i - j));
        if (pre[i] == pre[j])
            f[i] = max(f[i], f[j]);
        if (pre[i] - pre[j] < 0)
            f[i] = max(f[i], f[j] + (j - i));
    }
}
```

我们发现， $f[j]+i-j$ 可以变成 $(f[j]-j)+i$

所以观察上面的代码，我们考虑维护 $f[j]-j$，$f[j]$，$f[j]+j$ 三个值。

优化下枚举 j 的代码，发现本质上就是求。

$pre[j]$ 小于 $pre[i]$ 的最大值

$pre[j]$ 等于$pre[i]$ 的最大值

$pre[j]$ 大于 $pre[i]$ 的最大值

也就是用线段树的下标来表示 $pre[j]$ ，那么对于上述操作，可以理解为是

$que(1, pre[i]-1)$

$que(pre[i], pre[i])$

$que(pre[i]+1, n)$

但是我们的 pre 会出现负数，而且很大，我们可以使用动态开点线段树。不过也可以对 pre 进行离散化。

谢谢