线段树选讲

Part I

wYYSZL

2026.8.15

Outline

主要讲线段树，以及相关的一些题目。

题单在 vJudge 上面，密码是 tnfsTNFS（Taiyuan No.Five School）。

可以看到题目比较多，主要是考虑到大家的应该做过不少，而且线段树的所谓“类型题”比较多，即使有这么多的题目仍无法把主要的方法都涵盖。题目的权值大体反映难度，可以看自己的情况来做。

线段树实际上有各种各样的理论或者抽象，但是它们理解难度太大，也需要大量的背景知识，故略去。

虽然这么说，但是有些东西还是可以讲的，比如：什么问题可以用线段树解决？线段树构造是否一定要与线段或者区间有关？这些问题在思考问题时尤为重要。

最后，欢迎大家挑战难度 ≥ 30 的题目。

本节课内容

1. 线段树基础与小技巧
2. 线段树优化 DP
3. 线段树上二分
4. 扫描线
5. 线段树小技巧
6. 线段树维护矩阵
7. 线段树优化建图
8. 杂题

目录

线段树基础知识

考虑到大家已经在暑假学过一遍线段树，基本知识就不再赘述。

只问一个问题：

线段树基础知识

考虑到大家已经在暑假学过一遍线段树，基本知识就不再赘述。

只问一个问题：

Question

线段树区间操作的复杂度为什么是 $\mathcal{O}(\log n)$？

线段树基础知识

Proof:

模拟实现过程并分类讨论：在每个节点 $[p_l, p_r]$ 上，可能会出现以下几种情况：

1. $l \le p_l \le p_r \le r$，即完全覆盖了当前节点，直接返回。
2. $p_l \le l \le p_r \le r$，即只有 l 处于节点之内。
   - $l > mid$，只会递归右子树。
   - $l \le mid$，虽然递归两棵子树，但是右子节点会在递归后直接返回
3. $l \le p_l \le r \le p_r$，同 2。
4. $p_l \le l \le r \le p_r$，即 l 与 r 都位于节点之内。
   - l, r 都位于 mid 的一侧，只会递归一棵子树。
   - l, r 分别位于 mid 的两侧，递归左右两棵子树。

线段树基础知识

Proof:

也就是说，只有情况 $4(2)$ 会真正产生对左右两棵子树的递归。这种情况至多发生一次，之后在子节点上就会变成情况 2 或 3。

复杂度 $\mathcal{O}(2\log n) = \mathcal{O}(\log n)$。$\square$

从宏观上理解，相当于 l, r 两个端点分别在线段树上划分出一条递归访问路径，情况 $4(2)$ 在两条路径于从下往上的第一次交会处产生。

例题 A,B

A:P3372：区间加，区间求和。（模板题）

B:P3373：区间加，区间乘，区间求和。

例题 A,B

A:P3372：区间加，区间求和。(模板题)

B:P3373：区间加，区间乘，区间求和。

Sol B：

分别维护区间加，区间乘的懒标记，固定懒标记的顺序（下传时先应用乘再应用加），注意区间乘时区间加的懒标记也要变化。

例题 C

C - Mancala 2

有 N 个盒子编号为 0 到 $N-1$，初始时盒子 i 中有 A_i 个球。按顺序执行 M 次操作，第 i 次操作（$1 \leq i \leq M$）如下：

1. 将变量 C 置为 0。
2. 从盒子 B_i 中取出所有球拿在手中。
3. 当手中至少有一个球时，重复执行：
   - C 增加 1。
   - 从手中取出一个球放入盒子 $(B_i + C) \bmod N$。

求所有操作结束后每个盒子中的球数。

$N, M \leq 2 \times 10^5$

例题 C 题解

简单题。

注意到每次操作时，$+1$ 的是一个连续的区间。设有 num 个球，也就是先给所有的盒子加 $\lfloor \frac{num}{N} \rfloor$ 个球，之后剩下的 $d = num - N \cdot \lfloor \frac{num}{N} \rfloor$ 个球分别加到下标在 $[b_i + 1, b_i + d]$ 区间的盒子。如果 $b_i + d > N$ 就把超出来的加给开头即可。

时间复杂度 $\mathcal{O}(M \log N)$。

例题 D

D - Petya and Array

有 n 个整数 $a_1, a_2, \ldots, a_n$。求有多少对下标 (l, r) 满足 $1 \le l \le r \le n$ 且

$$\sum_{k=l}^{r} a_k < t.$$

$n \le 2 \times 10^5$, a 可为负

“权值线段树”（例题 D 题解）

记前缀和 $S_i = \sum_{k=1}^{i} a_k$，则条件为 $S_r - S_{l-1} < t \Leftrightarrow S_{l-1} > S_r - t\,(0 \le l-1 < r)$，即统计满足 $i < j$ 且 $S_i > S_j - t$ 的对数。

计算 S_i，用“权值线段树”（下标为值）维护已出现的前缀和个数。

按右端点 j 递增扫描：

- 查询值域中大于 $S_j - t$ 的元素个数，累加至答案；
- 将 S_j 插入树状数组（对应值位置加 1）。

需要离散化。单点修改，区间查询，没必要线段树，树状数组即可。

Hint

基本上可以看作二维偏序问题，这类问题（如：求逆序对）的标准解法就是“权值线段树”，此外分治做法也很常见。

目录

例题 E

E - 楼兰图腾

给定长度为 n 的排列 $p_1, \ldots, p_n$。

求满足 $i < j < k$ 且 $p_i > p_j < p_k$ 的三元组个数，以及满足 $p_i < p_j > p_k$ 的三元组个数。

数据规模：$n \leq 2 \times 10^5$，答案在 64 位带符号整数范围内。

例题 E 题解

以第一种为例，第二种同理。

我们在 j 处统计答案。如果我们能求出来 j 右边 $< p_j$ 的数目 a_j 和 j 左边 $> p_j$ 的数目 b_j，则由乘法原理得到

$$ans = \sum j = 1^n a_j b_j$$

考虑如何求 a_j，从右到左枚举 j，建立权值线段树/树状数组，每到一个 j，查询线段树上 $[1, p_j)$ 的 sum，即 a_j，然后在线段树的 p_j 位置 $++$。

时间复杂度 $\mathcal{O}(n \log n)$。

例题 F

F - 小白逛公园

有 n 个公园排成一排，第 i 个公园有一个整数得分 a_i。需要依次处理 m 个操作，每个操作为以下两种之一：

- **查询操作**：给定 a, b（$1 \le a, b \le n$），令 $l = \min(a, b)$，$r = \max(a, b)$，求区间 $[l, r]$ 内连续子段的最大和（即最大子段和）。
- **修改操作**：给定 p, s（$1 \le p \le n$，$|s| \le 1000$），将第 p 个公园的得分改为 s。

$n \le 5 \times 10^5$

单开一页

线段树的关键在于如何合并两个子区间以求得大区间的答案。
能否使用线段树之一就在于能否高效合并两个子区间。

线段树设计技巧（例题 F 题解）：

显然要维护最大子段和。我们要求从 $[l, mid]$ 和 $[mid+1, r]$ 的信息推出 $[l, r]$。
若仅维护最大子段和 ans，合并时取子区间的最大子段和的较大值，则得到左子区间或右子区间的最大子段和，这忽略了左端点在 $[l, mid]$ 且右端点在 $[mid+1, r]$ 的区间。跨过中点的最大区间是 $[l, mid]$ 的最大后缀与 $[mid+1, r]$ 的最大前缀拼在一起，于是维护区间最大前后缀和 pre, suf，这样就能从子区间的信息得到当前区间的最大子段和。

$$ans \leftarrow \max(ans_{ls}, ans_{rs}, suf_{ls} + pre_{rs})$$

线段树设计技巧（例题 F 题解）：

那么最大前后缀和怎么从子区间合并？考虑到前缀是左子区间的某个前缀，或者整个左子区间接上右子区间的前缀，所以还要维护整个区间的和 sum。后缀和同理。

$$pre \leftarrow \max(pre_{ls}, sum_{ls} + pre_{rs})$$
$$suf \leftarrow \max(suf_{rs}, sum_{rs} + suf_{ls})$$

此时信息已经封闭（也就是，我们不需要其他的信息了）：区间和等于子区间的区间和相加。

$$sum \leftarrow sum_{ls} + sum_{rs}$$

一些概述性的内容 - 信息设计

讲一些概述性的内容。

用线段树解决问题，首先得考虑维护哪些信息。若不带修，任何满足结合律且封闭的信息都是可维护的。结合律一般都有，封闭性帮助我们设计信息。

一些概述性的内容 - 信息设计

讲一些概述性的内容。

用线段树解决问题，首先得考虑维护哪些信息。若不带修，任何满足结合律且封闭的信息都是可维护的。结合律一般都有，封闭性帮助我们设计信息。

关键

说人话就是，我们在设计状态时，要考虑如何从 $[l, m]$ 和 $[m+1, r]$ 的信息推出 $[l, r]$，如果我们做不到，就需要多维护一些东西，或者这类问题不能用线段树来做。

一些概述性的内容 - 信息设计

一个通常的过程是：从要求的答案开始，考虑答案如何从子区间的答案合并。为此，可能需要维护一些辅助信息。再考虑辅助信息如何由子区间的辅助信息合并。重复该过程直到信息封闭。这种略显机械性的方法比一下子想出所有要维护的信息更简单。

一些概述性的内容 - 信息设计

一个通常的过程是：从要求的答案开始，考虑答案如何从子区间的答案合并。为此，可能需要维护一些辅助信息。再考虑辅助信息如何由子区间的辅助信息合并。重复该过程直到信息封闭。这种略显机械性的方法比一下子想出所有要维护的信息更简单。

对于懒标记，同样需要满足结合律和封闭性。但不用满足交换律，因为及时 push_down 保证每次打懒标记都是将当前懒标记对应的操作序列接在原懒标记对应的操作序列之后，即我们按时间顺序处理所有懒标记。

区间修改时，不仅标记和信息各自封闭，还要求标记和信息之间相互配合，使得原信息根据懒标记能够快速计算新的信息。因此，区间修改相较单点修改可能需要维护更多信息。

一些概述性的内容 - 实现技巧

当所维护信息太多时，用结构体会更有条理。此时一个结构体就是一个具有前文所说结合律和封闭性的元素，我们只需考虑如何合并两个结构体。
体现到代码上，就是我们用重载运算符的方式来代替 pushup。

```
1  struct P{...}tree[N];
2  P operator + (P x,P y){
3      P c={...};
4      return c;
5  }
6  tree[t]=tree[2*t]+tree[2*t+1];
```

图: pushup 的一种写法

当然，对于区间修改，可以用另一类结构体表示标记，这对于我们一会要讲到的区间历史问题有帮助。

例题 G

G - 脑洞治疗仪

有一个长度为 n 的 01 序列，初始时所有位置均为 1（表示没有脑洞）。需要依次处理 m 个操作，每个操作为以下三种之一：

- **挖脑洞**：给定 l, r，将区间 $[l, r]$ 内的所有位置变为 0。
- **脑洞治疗**：给定 l_0, r_0, l_1, r_1，用区间 $[l_0, r_0]$ 中的 1（脑组织）去填补区间 $[l_1, r_1]$ 中的 0（脑洞）。填补规则为：从左到右依次处理 $[l_1, r_1]$ 中的每个位置，若该位置为 0 且有可用的 1，则将其变为 1，同时从 $[l_0, r_0]$ 中取出一个 1（即该位置变为 0）。若 1 的数量不足，则尽可能填补靠前的脑洞；若 1 的数量过多，则直接扔掉。注意，$[l_0, r_0]$ 中的 1 被取出后变为 0，不会重复使用。
- **询问**：给定 l, r，求区间 $[l, r]$ 中最长连续 0 的段（即最大脑洞）的长度。

$n, m \leq 2 \times 10^5$

例题 G 题解

和上一个题基本一样，可以维护 lx, rx, mx 分别表示左端脑洞大小，右端脑洞大小和最大脑洞大小。

其中涉及需要赋 $1, 0$ 的地方，额外打标记。

其中还涉及“优先填补靠前的脑洞”，需要二分出能填到哪个位置，将这个区间全赋值为 1。

二分一个 $\log$，二分中在线段树上查询一个 $\log$，总复杂度 $\mathcal{O}(n \log^2 n)$。

可以通过线段树上二分做到一个 $\log$，不过这是明天的内容了。

例题 G 题解

和上一个题基本一样，可以维护 lx, rx, mx 分别表示左端脑洞大小，右端脑洞大小和最大脑洞大小。

其中涉及需要赋 $1, 0$ 的地方，额外打标记。

其中还涉及“优先填补靠前的脑洞”，需要二分出能填到哪个位置，将这个区间全赋值为 1。

二分一个 $\log$，二分中在线段树上查询一个 $\log$，总复杂度 $\mathcal{O}(n \log^2 n)$。

可以通过线段树上二分做到一个 $\log$，不过这是明天的内容了。

?

喜欢随机均摊算法的可以拿颜色段均摊来做。

例题 H

H - Generic Cow Protests G

给定长度为 n 的整数序列 $a_1, \ldots, a_n$。

求将该序列划分为若干个非空连续段（段数任意），使得每段内元素之和 ≥ 0 的方案数，对 $10^9 + 9$ 取模。

数据规模：$n \leq 10^5$，$|a_i| \leq 10^4$。

线段树处理 DP 限制 - 例题 H 题解

显然使用 DP，设 f_i 表示到第 i 个整数的方案数。

考虑最后一段，设前一个段的结尾为 j，则我们要求 $\sum_{k=j+1}^{i} \geq 0$，设 $fsum$ 为前缀和，转移显然有：

$$f_i = \sum_{fsum_i - fsum_j \geq 0} f_j$$

固定一个 i，则限制变为 $fsum_j \leq fsum_i$。考虑建出权值线段树，第 k 个位置存储所有 $fsum_j = k$ 的 f_j 的和。

于是对于一个 i，问题转化为求 $[1, fsum_i]$ 的区间和。总的复杂度 $\mathcal{O}(n \log n)$。

例题 I

I - Intervals

有一个长度为 N 的 01 串（未知）。给定 M 个区间，第 i 个区间为 $[l_i, r_i]$，权值为 a_i。若该区间内至少包含一个 1，则可获得权值 a_i。求所有长度为 N 的串中，能获得的最大权值总和。

$N \leq 2 \times 10^5$, a 可为负。

线段树“记载”状态（例题 I 题解）：

首先按右端点排序，设计状态 $f_{i,j}$ 表示考虑前 i 个点和 $r \le i$ 的区间，最后一个是 1 的位置是 j（没有 1 则 $j=0$）的最大价值。

不难得到转移方程：

$$f_{i,j}=\begin{cases}\max_{k=0}^{i-1}\{f_{i-1,k},0\}+\sum_{r_k=i} w_k & i=j\\ f_{i-1,j}+\sum_{r_k=i,l_k\le j} w_k & i\neq j\end{cases}$$

压掉 i 这维，可以用这个过程计算：

- 求 $f_i=\max_{k=0}^{i-1}\{f_k,0\}$；
- 枚举 $r_k=i,\forall l_k\le j\le i, f_j\leftarrow f_j+c_k$。

直接用线段树维护区间加、区间 max 即可。时间复杂度 $\mathcal{O}(n\log n)$。

线段树优化 DP：

总结一下，线段树优化 DP 分为 3 类。

线段树优化 DP：

总结一下，线段树优化 DP 分为 3 类。

第一类是属性在一个区间内的转移

这是讨论的是形如

$$f_i = Y(i) + \max_{j<i, P(j)\in[L(i),R(i)]} \{X(j)\}$$

或者

$$f_i = Y(i) + \sum_{j<i, P(j)\in[L(i),R(i)]} X(j)$$

的方程，其中 $P(i), L(i), R(i), X(i), Y(i)$ 都是关于 i 的一个函数。

一个显然的优化是：以 $P(j)$ 为下标建立线段树（可能需要离散化），每次在线段树上区间查询进行转移。如果是求和或者取前缀 max，也可以用树状数组代替。

常见的模型有最长上升子序列以及它的各种扩展，还有例题 H。

线段树优化 DP：

第二类是把状态按阶段拍到线段树上

有时状态为 $f_{i,j}$，第 i 个阶段的状态 $f_{i,j}$ 可以 $O(1)$ 从第 $i-1$ 个阶段的状态转移过来，那可以考虑用一棵线段树维护当前阶段状态的值。

一般，如果可以划分成几个区间进行贡献计算，则可以尝试线段树优化。

思考时可以把每个阶段分为几个步骤，把每个步骤在线段树上需要进行的操作考虑清楚，特别注意边界情况，方便思考和编码。

例题：例题 I、P8476 「GLR-R3」惊蛰。

线段树优化 DP

第三类是线段树合并优化树上 DP（整体 DP）（不过这个需要更多的知识）

树上 DP 的状态是 $f_{u,i}$ 的形式，且节点 u 的有效状态数量是 $O(size_u)$ 的，其中 $size_u$ 表示子树的大小或者其它权值，满足

$$size_u = w_u + \sum_{v \in son(u)} size_v, \quad size_{\text{root}} = O(n),$$

可以考虑用动态开点线段树记录状态，线段树合并进行转移，这时可能需要记录前后缀的信息，有时为了处理合并时一边空一边非空的情况，需要用懒标记。

例题：[NOI2020] 命运。

目录

线段树上二分

详见“线段树上二分.pdf”

目录

扫描线

详见“扫描线.pdf”

目录

“查理线段树”

二倍空间线段树的一种实现方式：

设节点 $tree_x$，表示区间 $[l, r]$ 的信息，其左右子树编号为 $2 * mid$ 与 $2 * mid + 1$。

“查理线段树”

二倍空间线段树的一种实现方式：

设节点 $tree_x$，表示区间 $[l, r]$ 的信息，其左右子树编号为 $2 * mid$ 与 $2 * mid + 1$。

正确性证明 *

假设存在节点冲突，考虑到不可能跟 1 重叠，其他节点都是由 mid 计算来的，也就是某两个节点的 mid 相同。

显然有 $[l_1, r_1] \cap [l_2, r_2] \neq \emptyset$。

又：$[l_1, r_1], [l_2, r_2]$ 是线段树上的两个不同区间，为包含关系，此处假设 $[l_2, r_2] \subseteq [l_1, r_1]$，进行分类讨论。

“查理线段树”

正确性证明 *

由于是线段树上的节点，所以不存在 $l_2 \le mid_1 < r_2$ 的情况，所以可以分为两类：

1. $r_2 \le \left\lfloor \frac{l_1+r_1}{2} \right\rfloor$，此时

$$\left\lfloor \frac{l_2+r_2}{2} \right\rfloor \le \left\lfloor \frac{2r_2}{2} \right\rfloor = r_2 \le \left\lfloor \frac{l_1+r_1}{2} \right\rfloor.$$

   当且仅当 $l_2 = r_2 = \left\lfloor \frac{l_1+r_1}{2} \right\rfloor$ 时，等号成立，满足条件。但是，$l_2 = r_2$ 时该线段不存在子节点，不影响，可以忽略。

2. $l_2 \ge \left\lfloor \frac{l_1+r_1}{2} \right\rfloor + 1$，此时 $l_2 > \left\lfloor \frac{l_1+r_1}{2} \right\rfloor$，有 $\left\lfloor \frac{l_2+r_2}{2} \right\rfloor \ge l_2 > \left\lfloor \frac{l_1+r_1}{2} \right\rfloor$，不满足条件.

综上，l_1, r_1, l_2, r_2 满足条件当且仅当较小区间为叶子节点时。□

动态开点

当线段树数量太多（线段树合并）或下标值域太大（权值线段树）（而且不方便离散化时）时，由于空间限制，无法使用左儿子两倍，右儿子两倍加一的编号方法。

解决方法是动态开点：对于每个结点，维护其左右儿子的编号，向下递归时，若当前结点为空则新建结点。查询时走到空结点就返回。

一般通过传递引用或返回结点编号的方式快速更新子结点编号。

目录

历史最值

O - CPU 监控

有一个长度为 T 的整数序列 $a_1, a_2, \ldots, a_T$，初始值给定。需要依次处理 E 个操作，每个操作为以下四种之一：

- Q l r：询问区间 $[l, r]$ 的当前最大值。
- A l r：询问区间 $[l, r]$ 的历史最大值（即从开始到当前时刻曾达到过的最大值）。
- P l r z：将区间 $[l, r]$ 的每个数增加 z。
- C l r z：将区间 $[l, r]$ 的每个数赋值为 z。

对于每个询问，输出一行一个整数表示答案。

z 可以是负数。

历史最值（O 题题解）

注意

接下来讲通常的做法，这个做法非常麻烦，不要求理解。可以从 P 题题面开始看。

第一步：状态设计

显然维护当前最大值 mx 和历史最大值 h，此时信息已经封闭：

$$mx \leftarrow \max(mx_{\text{ls}}, mx_{\text{rs}}), \qquad h \leftarrow \max(h_{\text{ls}}, h_{\text{rs}}).$$

第二步：标记设计与下传

首先少不了区间加区间赋值的经典标记。如果被赋值过，那么维护值 as 表示赋值标记，否则维护的值 ad 为加标记。赋值 as 之后加上 ad 等价于赋值 $as + ad$，所以要么只有 as 标记，要么只有 ad 标记。

维护 as 和 ad 用于更新 mx。如果有 as 标记，那么 $mx \leftarrow as$，否则 $mx \leftarrow mx + ad$。

历史最值（O 题题解）

续：标记下传

我们要求一段操作之后的历史最大值。以第一次赋值为分界线，之前的操作都是加法，之后的操作都是赋值。这两种操作需要分开考虑，因为加的值和赋的值显然不能混为一谈。于是维护**没有下传的**加法操作前缀和的历史最大值的标记 ha_{ad}，以及赋值操作的历史最大值 ha_{as}。那么新的历史最大值

$$h \leftarrow \max(h,\ mx + ha_{ad},\ ha_{as}).$$

同一般的标记一样，下传标记后清空标记。

历史最值（O 题题解）

第三步：标记合并

不用动脑子的方法是分成两个具有先后关系的标记是否有 as 标记的四种情况讨论，根据实际意义合并。

- 如果两个都没有 as 标记：

  $ad \leftarrow ad_{old} + ad_{new}, \qquad ha_{ad} \leftarrow \max(ha_{ad_{old}},\ ad_{old} + ha_{ad_{new}}).$

- 如果只有 old 有 as 标记：

  $as \leftarrow as_{old} + ad_{new}, \qquad ha_{ad} \leftarrow ha_{ad_{old}}, \qquad ha_{as} \leftarrow \max(ha_{as_{old}},\ as_{old} + ha_{ad_{new}}).$

- 如果只有 new 有 as 标记：

  $as \leftarrow as_{new}, \qquad ha_{ad} \leftarrow \max(ha_{ad_{old}},\ ad_{old} + ha_{ad_{new}}), \qquad ha_{as} \leftarrow ha_{as_{new}}.$

- 如果两个都有 as 标记：$as \leftarrow as_{new}, \qquad ha_{ad} \leftarrow ha_{ad_{old}},$

  $ha_{as} \leftarrow \max(ha_{as_{old}}, as_{old} + ha_{ad_{new}}, ha_{as_{new}}).$

历史和

区间历史和问题与历史最值问题的求解思路类似，需要在标记中多维护求历史和轮数。

对于区间加、区间历史和，维护以下信息：

- 区间长度 len
- 区间和 s
- 区间历史和 h

标记维护每个位置加的值 ds。

区间和 s 的维护是常规的（通过 ds 和 len），关键在于 h 的更新。

历史和

设每次求历史和时加的值分别为 $ds_1, \ldots, ds_t$，则历史和新增

$$\sum_{i=1}^{t}(s + len \cdot ds_i) = s \cdot t + len \cdot \sum_{i=1}^{t} ds_i.$$

因此标记需要维护：

- 每个位置加的值的历史和 $hs = \sum_{i=1}^{t} ds_i$
- 求历史和轮数 t（一般让 t 单独加 1 表示进行一轮历史和查询）

标记作用在 h 上：

$$h \leftarrow h + s \cdot t + len \cdot hs.$$

历史和

根据实际意义合并两个标记（假设当前结点已有标记 (ds, hs, t)，需要合并来自父结点的标记 (ds', hs', t')）：

- ds 直接相加：$ds \leftarrow ds + ds'$
- t 直接相加：$t \leftarrow t + t'$
- hs 需要综合考虑：$hs \leftarrow hs + ds \cdot t' + hs'$

解释：原有的 hs 是之前所有轮的历史和，现在加上新的一批轮次中，原有的 ds 会在新轮次中贡献 $ds \cdot t'$（因为每轮每个位置都加 ds），再加上新轮次本身的历史和 hs'。

历史和

P - 比赛

给定两个长度为 n 的排列 a 和 b。有 Q 个询问，每个询问给出区间 $[l, r]$。对于每个询问，求所有满足 $l \le p \le q \le r$ 的子区间 $[p, q]$ 的权值之和，其中权值为 $\max\{a_i \mid p \le i \le q\} \times \max\{b_i \mid p \le i \le q\}$。答案对 2^{64} 取模。

$1 \le n, Q \le 2.5 \times 10^5$

$1 \le a_i, b_i \le n$，且 a_i 和 b_i 均为 1 到 n 的排列。

历史和（P 题题解）：

设 $T_{i,j}$ 表示 $q=i$ 且 $p=j$ 时的答案。对 T_i，设 $c_j=\max_{k=j}^{i} a_k$，$d_j=\max_{k=j}^{i} b_k$，则 $T_{i,j}=c_jd_j$。

扫描线，考虑 $T_{i-1}\to T_i$。加入 a_i 时，考虑它对 c_j 的影响，类似前面 C 题的单调栈做法，我们可以知道整个过程的总段数为 $\mathcal{O}(n)$。加入 b_i 同理。因此可以用线段树维护 T_i。

询问 $[l,r]$ 的答案相当于对所有 $T_i(l\le i\le r)$ 的 $[l,i]$ 区间和求和。因为 $i<j$ 时 $T_{i,j}=0$，所以又可以写成 $T_{i,j}(i\le r)$ 之和，即 T_r 上 $[l,r]$ 的区间历史和。

问题转化为 c_i,d_i 区间加，查询区间 $\sum c_id_i$ 历史和。

这样直接做，需要 5 个懒标记和 5 个维护信息，比较麻烦但是可以做。

好麻烦

这么多标记，这么多情况，好容易错，好麻烦。
有没有一种方便一点的呢？哪怕常数大一点。
有的，有的。

线段树维护矩阵

一切具有线性性的都应该想想矩阵。矩阵是一种认识世界的方法。

线段树维护矩阵

一切具有线性性的都应该想想矩阵。矩阵是一种认识世界的方法。
以历史和为例，我们看看矩阵的力量。

线段树维护矩阵

一切具有线性性的都应该想想矩阵。矩阵是一种认识世界的方法。
以历史和为例，我们看看矩阵的力量。

线性性？

设区间长度 len，区间和 s，区间历史和 h。整体加 v 是 $s \leftarrow s + v \cdot len$，求历史和是 $h \leftarrow h + s$，都是 len, s, h 的线性组合。将一些数的每个数同时变成所有数的线性组合，我们立刻想到向量和矩阵乘法。

历史和（矩阵）

将信息写成向量 $[h \quad s \quad len]^T$（历史和，和，区间长），则整体加 v 等价于左乘矩阵

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} h \\ s \\ len \end{bmatrix} = \begin{bmatrix} h \\ s + v \cdot len \\ len \end{bmatrix}$$

求 t 次历史和等价于左乘矩阵

$$\begin{bmatrix} 1 & t & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} h \\ s \\ len \end{bmatrix} = \begin{bmatrix} h + t \cdot s \\ s \\ len \end{bmatrix}$$

矩阵乘法满足结合律，可以用线段树维护区间向量乘矩阵，区间向量和。

历史最值（矩阵版）

线段树历史版本最大/最小值。此时还是只有区间加这个修改操作，但是发现我们需要有 $\max / \min$ 的运算，需要引入一些新的东西，即广义矩阵，对于这个矩阵，矩阵乘法应该支持 $\max / \min$，以 $\max$ 为例。

我们在这个广义矩阵中，加法运算定义为取两个数的最大值，记作 $a \oplus b = \max(a, b)$，乘法运算定义为普通加法，记作 $a \otimes b = a + b$。
因此，这种广义矩阵的乘法规则为：

$$(A \otimes B)_{ij} = \bigoplus_k (A_{ik} \otimes B_{kj}) = \max_k (A_{ik} + B_{kj}).$$

这种广义矩阵乘法有一些很厉害的名称，$(\max,+)$ 矩阵乘法，或者叫热带半环上的矩阵乘法。

历史最值（矩阵版）

线段树上的向量为：$\vec{a} = \begin{bmatrix} \mathrm{mx} \\ \mathrm{his} \end{bmatrix}$

其中 mx 表示当前的最值，his 表示历史的最值。

对于区间加操作，我们不难构造出矩阵：

$$\begin{bmatrix} \mathrm{mx} + x \\ \mathrm{his} \end{bmatrix} = \begin{bmatrix} x & -\infty \\ -\infty & 0 \end{bmatrix} \otimes \begin{bmatrix} \mathrm{mx} \\ \mathrm{his} \end{bmatrix}$$

（之后都是广义乘法，$\otimes$ 将省略）

历史最值（矩阵版）

对于 mx 刷到 his 的操作，我们也不难构造出矩阵：

$$\begin{bmatrix} mx \\ \max(\mathrm{mx}, \mathrm{his}) \end{bmatrix} = \begin{bmatrix} 0 & -\infty \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathrm{mx} \\ \mathrm{his} \end{bmatrix}$$

然而发现只有在利用加法改变这个值的时候，才需要将 mx 刷到 his，所以可以将两个矩阵合在一起，这样就不用单独的刷 his 了：

$$\begin{bmatrix} mx + x \\ \max(\mathrm{mx} + x, \mathrm{his}) \end{bmatrix} = \begin{bmatrix} x & -\infty \\ x & 0 \end{bmatrix} \begin{bmatrix} \mathrm{mx} \\ \mathrm{his} \end{bmatrix}$$

如何赋值（矩阵版）

我们发现前面都是区间加什么的，没有说到如何赋值。

其实是容易的，以历史和为例，假设有赋值为 x 的操作（不含求历史和的过程），有：

$$\begin{bmatrix} h \\ x \cdot len \\ len \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & x \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} h \\ s \\ len \end{bmatrix}$$

以及，有时候，我们会在向量里面放一个 1 来支持更多的操作。

再战 P 题

首先我们还是考虑扫描线，以及利用单调栈将区间取 max 变为区间加，所有子区间，于是还要有一个历史版本和。

问题转化为：有两个序列 A, B，需要分别支持区间加，以及查询 $\sum_i A_i \times B_i$。

考虑只改变 A 的影响：$ab \leftarrow (a+x)b = ab + xb$。
于是需要维护向量：

$$\vec{v} = \begin{bmatrix} his \\ mul \\ a \\ b \\ len \end{bmatrix}$$

其中 his 是历史和，$mul = \sum(A_i \times B_i)$，$a = \sum A_i$，$b = \sum B_i$，$len$ 为区间长。

再战 P 题

对 A 区间加 x 的矩阵：

$$\begin{bmatrix} his \\ mul + x \cdot b \\ a + x \cdot len \\ b \\ len \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & x & 0 \\ 0 & 0 & 1 & 0 & x \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} his \\ mul \\ a \\ b \\ len \end{bmatrix}$$

对 B 区间加 x 的矩阵：同理

但是由于这个矩阵是 5×5 的，有 $5^3 = 125$ 的常数，需要卡常。
不过这类问题有通常的卡常方法：

减去无用转移

直接矩阵乘法的常数为 $\mathcal{O}(k^3)$，虽然小，但是有些时候也不容小觑，于是我们此刻利用矩阵操作本身的一些性质来减少矩阵乘法的复杂度。

以如下乘法为例：

$$\begin{bmatrix} his + sum \\ sum \\ len \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} his \\ sum \\ len \end{bmatrix}$$

操作的矩阵有两种：

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & x \\ 0 & 0 & 1 \end{bmatrix}$$

减去无用转移

我们注意到有些地方的值是固定的，我们把两个矩阵乘起来，

$$\begin{bmatrix}1 & a & 0\\0 & 1 & b\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & c & 0\\0 & 1 & d\\0 & 0 & 1\end{bmatrix}=\begin{bmatrix}1 & c+a & ad\\0 & 1 & b+d\\0 & 0 & 1\end{bmatrix}$$

发现多了一个值不确定的地方，然后将这个“最不确定的矩阵”乘起来：

$$\begin{bmatrix}1 & a & b\\0 & 1 & c\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & d & e\\0 & 1 & f\\0 & 0 & 1\end{bmatrix}=\begin{bmatrix}1 & a+d & e+af+b\\0 & 1 & c+f\\0 & 0 & 1\end{bmatrix}$$

发现没有新的不确定位置。

减去无用转移

因此，只需要维护右上角的三个位置，常数大大减小。

例如，对于 P 题，只有右上角 10 个数字有位置（可以认为是对角线为 1 的上三角矩阵）。对于这个矩阵，不难发现只需要进行 10 次乘法。

减去无用转移

因此，只需要维护右上角的三个位置，常数大大减小。
例如，对于 P 题，只有右上角 10 个数字有位置（可以认为是对角线为 1 的上三角矩阵）。对于这个矩阵，不难发现只需要进行 10 次乘法。

一般来说，我们需要构造出来的向量，对于每一个操作都应该是一个上/下三角矩阵的形式，这样更加方便我们观察与理解，以及优化常数。
而如果要成为一个上三角，就意味着对于 a_i 只会由 $j \geq i$ 的 a_j 转移而来。
于是一般来说，会将不变的长度放在最下面，将历史版本信息放在最上面，一般的信息则放在中间。（虽然我一般反着写，形成下三角矩阵）

减去无用转移

如果极限压制，对于 P 题，有用的位置只会有 9 个（对角线全是 1，还有一个位置为 0），于是只需要维护这 9 个位置的值即可。

此外，我们发现，这种过程是在将这个矩阵转移转化为 tags 的转移。

当然，实际上，这两个东西就是同出异名，矩阵角度更好思考。

减去无用转移

如果极限压制，对于 P 题，有用的位置只会有 9 个（对角线全是 1，还有一个位置为 0），于是只需要维护这 9 个位置的值即可。

此外，我们发现，这种过程是在将这个矩阵转移转化为 tags 的转移。

当然，实际上，这两个东西就是同出异名，矩阵角度更好思考。

要注意，线段树维护矩阵远远不止于维护历史信息，它可以处理几乎所有需要很多 tag 的线段树问题。甚至，你可以拿它来写经典的线段树模板 2。

不知道谁说的

“矩阵与向量是认识世界的一种方法。”

目录

例题 O：

O - Legacy

有 n 个星球编号 1 到 n，Rick 初始在星球 s。有 q 种传送门计划，每种计划可以购买一次并立即使用（可重复购买），每种计划的花费为 w，有以下三种类型：

- 从星球 v 到星球 u 的单向传送门。
- 从星球 v 到任意（可以理解为“所有”）星球 $u \in [l, r]$ 的单向传送门。
- 从任意（可以理解为“所有”）星球 $v \in [l, r]$ 到星球 u 的单向传送门。

求从 s 出发到达每个星球的最小总花费。

$1 \leq n, q \leq 10^5$

线段树优化建图（例题 Q 题解）

不难发现实际上是求单源最短路。

关键在于我们要给一个点向一个连续段连边（或反过来）。

考虑线段树是在做什么：将一个区间破拆为 $\log$ 个线段。于是我们可以借用这种破拆来连边，将向连续段连边转化为向代表线段的特殊节点连边。

具体的，

- 建立两棵线段树，分别称为出树和入树。
- 出树中，边由子节点指向父节点，权值为 0，表示从小区间可以无代价地汇聚到大区间。
- 入树中，边由父节点指向子节点，权值为 0，同理
- 两棵树的叶子节点（对应实际星球 1 到 n）之间用双向边连接，权值为 0。

线段树优化建图（例题 Q 题解）

考虑连边：

1. 类型 1 $(v \to u)$：直接连边 $\text{out}_v \to \text{in}_u$，权值 w。
2. 类型 2 $(v \to [l, r])$：在入树中定位覆盖 $[l, r]$ 的 $O(\log n)$ 个节点，设这些节点集合为 P，对每个 $p \in P$，连边 $\text{out}_v \to p$，权值 w。
3. 类型 3 $([l, r] \to u)$：在出树中定位覆盖 $[l, r]$ 的 $O(\log n)$ 个节点，设这些节点集合为 P，对每个 $p \in P$，连边 $p \to \text{in}_u$，权值 w。

最后建出来的图边数位 $O(n \log n)$，求最短路一个 $\log$，总计两个 $\log$。

目录

例题 R

R - 上帝造题的七分钟 2 / 花神游历各国

有一个长度为 n 的正整数数列。需要依次处理 m 个操作，每个操作为以下两种之一：

- 0 l r：将区间 $[l, r]$ 内的每个数开平方（下取整）。
- 1 l r：询问区间 $[l, r]$ 的和。

$1 \leq n, m \leq 10^5$

数列中的数初始不超过 10^{12}

例题 R 题解

注意到，每个数 $x \le 10^{12}$ 在开平方下取整时，$\lfloor \sqrt{x} \rfloor$ 会迅速减小，且当 $x \le 1$ 时不再变化。每个数最多被修改 $\log \log M \approx 6$ 次。
递归处理，若当前区间最大值 ≤ 1 则直接返回；否则递归到叶子节点进行修改。

考虑到每次进入深的节点，要么是寻找操作对应区间，要么会对至少一个数字修改，故时间复杂度 $O(n \log \log M \log n)$。

例题 S

S - Palindrome Query

有一个长度为 N 的字符串 S，由小写英文字母组成。依次处理 Q 个操作，每个操作为以下两种之一：

- 1 x c：将第 x 个字符改为小写字母 c。
- 2 L R：询问子串 $S[L..R]$ 是否为回文串。

$1 \leq N \leq 10^6$

例题 S 题解

考虑哈希，正反建立哈希，用线段树维护哈希数组，判定正反串是否相同。
修改一个字母相当于修改一个后缀，然后询问相当于单点查询。
或者维护“字母 $\times base$ ”，相当于单点修改区间查询。
都一样，树状数组就行。

例题 T

T - 方差

有一个长度为 N 的实数数列，初始值给定。依次处理 M 个操作，每个操作为以下三种之一：

- 1 x y k：将区间 $[x, y]$ 内的每个数加上实数 k。
- 2 x y：询问区间 $[x, y]$ 的平均数。
- 3 x y：询问区间 $[x, y]$ 的方差。

$1 \leq N, M \leq 10^5$

例题 T 题解

方差和平均数不好维护，拆解所求内容，维护其他：

对于区间 $[x, y]$，设长度为 $L = y - x + 1$，则：

- 平均数 $\bar{a} = \dfrac{S}{L}$。
- 方差 $D = \dfrac{1}{L}\sum_{i=x}^{y}(a_i - \bar{a})^2 = \dfrac{1}{L}\left(\sum a_i^2 - \dfrac{(\sum a_i)^2}{L}\right) = \dfrac{Q}{L} - \left(\dfrac{S}{L}\right)^2$。

L 为定值，只需维护 S（区间和），Q（区间平方和）。

平方和和方差很像，类似拆解：

$$Q' = \sum (a_i + k)^2 = \sum a_i^2 + 2k\sum a_i + Lk^2 = Q + 2kS + Lk^2.$$

发现线段树节点存储 S 和 Q，即可。每次操作 $O(\log N)$，总复杂度 $O((N+M)\log N)$

例题 U

U - 楼房重建

有 N 栋楼房，第 i 栋位于 $x = i$，初始高度均为 0。按顺序进行 M 次操作，每次将第 X_i 栋的高度改为 Y_i（Y_i 为整数）。每次操作后，问从原点 $(0, 0)$ 能看到的楼房数量。一栋楼房可见当且仅当存在一点 (i, h)（$0 < h \leq H_i$）使得该点与原点连线的斜率大于之前所有楼房的斜率（即不被遮挡）。

数据范围

$1 \leq N, M \leq 10^5$

“楼房重建线段树”（例题 U 题解）

将每栋楼房视为从原点出发的射线斜率 $k_i = \frac{H_i}{i}$。一栋楼能被看到当且仅当其斜率大于之前所有楼的最大斜率。因此问题转化为：动态修改单点高度，维护全局“斜率递增”的序列长度（即从左边开始，每次取更大的斜率，统计个数）。

使用线段树，每个节点维护：

- mx：区间内最大斜率。
- cnt：只考虑该区间时，从该区间左端点能看到的楼房数量（即该区间内的“可见序列”长度）。

“楼房重建线段树”（例题 U 题解）

依然考虑如何合并：合并两个子区间 L 和 R 时：

- 当前节点的最大斜率为 $\max(L.mx, R.mx)$。
- 当前节点的可见数量 $= L.cnt + R$ 中在 $L.mx$ 之后可见的数量。

其中第二部分需要递归计算：给定一个起始最大斜率 k，求一个区间内从左边开始斜率大于 k 的可见序列长度。可以写一个函数 $\mathrm{calc(node, k)}$ 实现：

- 若区间最大斜率 $\leq k$，则无贡献，返回 0。
- 若区间为叶子，返回 1。
- 否则，先看左子区间：若左子区间的最大斜率 $\leq k$，则左子无贡献，直接递归右子；若左子最大斜率 $> k$，则递归左子得到左子贡献，再加上右子在左子最大斜率之后的贡献（无需递归，只需用父亲节点的 cnt 减去左子节点的 cnt 即可）。

这样每次合并复杂度 $O(\log N)$，单点修改时沿路径更新，总复杂度 $O(M \log^2 N)$。

* 例题 V

*V - k-d-sequence

我们称一组整数序列为好的 k-d 序列，是指我们最多可以向序列中插入 k 个数，使得排序后该序列成为公差为 d 的等差数列。

你得到了一个由 n 个整数构成的序列 a，你的任务是找到该序列的最长连续子段，使其是一个好的 k-d 序列。

$n, k \leq 2 \cdot 10^5$

* 例题 V 题解

首先，如果不限制 k，一个区间要满足条件当且仅当两个条件成立：$[l,r]$ 不含相同的数，$[l,r]$ 模 d 均相同。
加上限制 k 实际上就是对极差有了要求，即

$$\max(l,r)-\min(l,r)\le(r-l+k)d$$

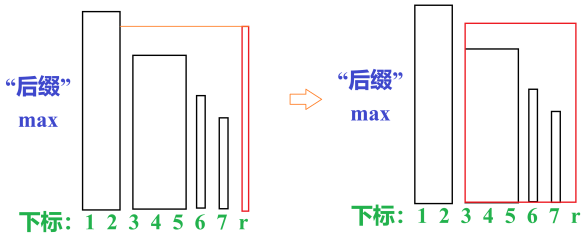
现在我们枚举右端点 r，稍微控制左端点最左可能位置可以实现前两个条件，不妨设这个最左的可能位置为 L。考虑第三个条件，变形：

$$\max(l,r)-\min(l,r)+ld\le(r+k)d$$

右边是一个定值。我们来维护左边这个值，这样，在线段树上二分查找 $[L,r]$ 中最小 l，使得满足这个不等式即可。关键是如何维护 $\max$ 和 $\min$。

* 例题 V 题解

考虑使用单调栈，以最大值为例，一些部分的最大值是相同的，形成一个平台，这个平台一定是一个区间。那么，当右端点 r 加入时，最大值小于这个值的会形成一个新的平台，对这些部分依次进行区间加，显然这个区间加的次数是均摊 $\mathcal{O}(n)$ 的。总的时间复杂度 $\mathcal{O}(n\log n)$。



例题 W

W - 冰红茶

有 n 个机器人排成一排，初始全部存活。每个机器人有一个喝冰红茶的口味记录（原味或热带风味）。依次进行 m 个操作，操作有三种类型：

- 1 l r k：将区间 $[l, r]$ 内的每个机器人增加 k 瓶原味冰红茶，区间外的每个机器人增加 k 瓶热带风味冰红茶。
- 2 l r k：对于区间 $[l, r]$ 内的每个机器人，检查其记录中最后连续相同口味的次数是否至少为 k，若是，则将该机器人击毁（变为不存活）。击毁后该机器人不再参与后续操作，但编号保留。
- 3：询问当前存活的机器人数量。

$1 \leq n, m \leq 2 \times 10^5$，$1 \leq k \leq 10^6$

例题 W 题解

先考虑操作 1。注意到执行一次操作 1 后，所有机器人最后一次喝的类型就已经确定了。所以我们执行操作 1 的时候只需要考虑和上一次操作 1 的关系。设这次操作 1 的区间为 $[l, r]$，上一次为 $[l', r']$，分类讨论：

- 两个区间无交集，则两个区间中的机器人连续喝的同种饮料数量都会**变成** k，而其他的则**增加** k。
- 两个区间有交集，则两个区间中不相交的区间的机器人连续喝的同种饮料数量都会变成 k，而其他的则增加 k。

这两种综合，可以更简单：将 $l, r, l', r', 0, n$ 全部视作端点，则第一个段是增加 k，之后每遇到一个端点将增加/变成 k 切换成变成/增加 k。

对于操作 2，注意到一个机器人最多被击毁一次。维护最后相同口味 $\max$，当一个节点的最大值 $\geq k$ 时继续递归。这样每次递归至少删掉一个，总复杂度 $\mathcal{O}(n \log n)$。

例题 W 题解

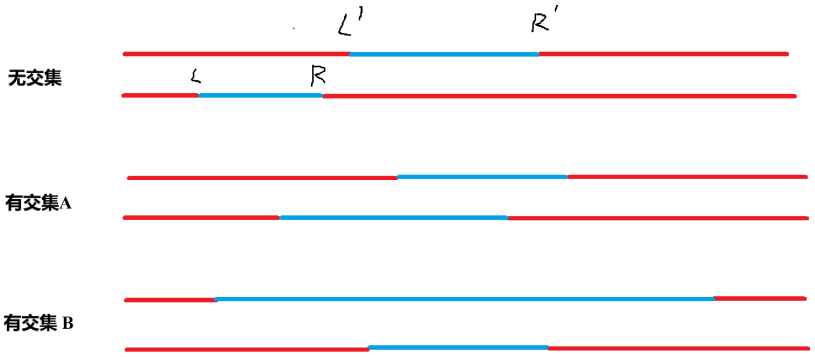


图: 操作 1 分类图示

* 例题 X

*X - 幻梦

有一个长度为 n 的序列，开始时第 i 位为 a_i 。你需要完成 q 次操作：

- 对于所有的 $l \leq i \leq r$，令 $a_i \leftarrow a_i + x$。
- 对于所有的 $l \leq i \leq r$，令 $a_i \leftarrow \text{popcount}(a_i)$。
- 查询 a_p 的值。

其中 $\text{popcount}(x)$ 为 x 的二进制表示中 1 的个数。

$1 \leq n \leq 3 \times 10^5$，$1 \leq q \leq 10^6$

* 例题 X 题解

如果没有操作二，那么直接用线段树维护即可。

考虑加入操作二。显然 $0 \le \text{popcount}(a_i) \le \log V$，那么说明一个区间在整体进行一次操作二之后只会有 $\log V$ 种取值。因此，对于每个位置，先处理到出现第一个操作二，之后可以直接维护区间里 $1 \sim \log V$ 内所有数在第一次操作二之后会变成的数。

具体的，维护标记 $t_{p,i}$ 代表区间 p 内数 i 在（第一个没有下传的 2 操作）之后的操作会变成 $t_{p,i}$，add_p 表示区间 p 的加法标记（仅限区间 p 没有 t 标记时）。

- 如果存在 t 标记，操作一是 $t_{p,i} \leftarrow t_{p,i} + x$，操作二是 $t_{p,i} \leftarrow \text{popcount}(t_{p,i})$；
- 如果不存在，操作一是 $add_p \leftarrow add_p + x$，操作二是先下传加法标记，然后 $t_{p,i} = i$。

* 例题 X 题解

下传 t 标记时：

- 如果子区间 s 还没有 t 标记，那么先下传 s 的加法标记，然后 $t_{s,i} \leftarrow t_{p,i}$
- 否则 $t_{s,i} \leftarrow t_{p,\text{popcount}(t_{s,i})}$。

询问时从叶子向上跳，设当前值为 x，当前跳到区间 p，如果 p 存在 t 标记，那么 $x \leftarrow t_{p,\text{popcount}(x)}$，否则 $x \leftarrow x + add_p$。
时间复杂度 $\mathcal{O}(n \log n \log V)$。

* 例题 Y

*Y - Souvenirs

给定整数序列 $a_1, a_2, \ldots, a_n$。

m 次询问，给定区间 $[l, r]$，求区间内任意两个不同位置上的元素之差的绝对值的最小值。

$2 \leq n \leq 10^5$, $1 \leq m \leq 3 \times 10^5$

* 例题 Y 题解

设答案为 $i < j$。不妨设 $a_i \geq a_j$。对于 $a_i < a_j$ 的情况，可取反后再做一遍。

按 r 扫描线对每个 l 维护询问 $[l, r]$ 的答案 f_l，则扫到 r 时每个 $i < r$ 且 $a_i \geq a_r$ 会对 $l \leq i$ 产生 $a_i - a_r$ 的贡献。l 向左跳动时，a_l 只有减小才可能更新答案。

因此，直接查询所有 $\geq a_r$ 且小于当前 a_i 的 i 左边的第一个数 $a_{i'}$，则 i 跳动至 i'。对 i' 的查询可以按 a_j 从大到小建可持久化线段树，维护区间最小值，然后在 a_r 对应线段树上二分即可。

* 例题 Y 题解

但这样复杂度依然无法承受。进一步考察性质，我们发现，如果 (i', r) 可能贡献至答案，则因为包含 $[i', r]$ 的区间必然包含 $[i', i]$ 和 $[i, r]$，所以：

$$a_{i'} - a_r < \frac{a_i - a_r}{2}$$

否则 $a_i - a_{i'}$ 一定比 $a_{i'} - a_r$ 优。

这样我们就可以将跳动 i 的次数降至 $\mathcal{O}(\log V)$。

进一步离线 i' 的查询可以将可持久化线段树转化为普通线段树。

时间复杂度 $\mathcal{O}(n \log n \log V)$。