

2025/7/22 课程

数据结构专题



陈景泰

2025-07-21

1. 第一部分：线段树的各种用法



洛谷 P5278 算术天才⑨与等差数列

问题 1

有一个长度为 n 的序列，其中第 i 个数为 a_i ，每次给出询问 l, r, k ，问区间 $[l, r]$ 内的数从小到大排序后能否形成公差为 k 的等差数列。需要支持 m 次单点修改。强制在线。

数据范围： $1 \leq n, m \leq 3 \times 10^5, 0 \leq a_i, k \leq 10^9$ 。



一个序列 b_1, b_2, \dots, b_k 在排序后形成公差为 $k > 0$ 的等差数列，当且仅当：

- 令 $D = \max_{i=1}^k b_i - \min_{i=1}^k b_i$ ，那么 $D = (n-1)k$ ；
- 记相邻两个元素的差 $c_i = |b_{i+1} - b_i|$ ，那么 $\gcd_{i=1}^{k-1} c_i = k$ ；
- b_1, b_2, \dots, b_k 互不相同。

而对于 $k = 0$ 的情况，只需要满足第一个条件，也即 $D = 0$ 即可。



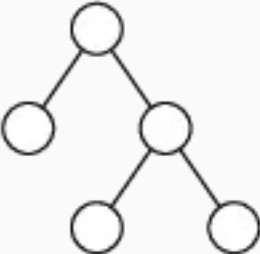
因此在线段树上，需要启用一个线段树，在 a_i 上维护区间最大值、区间最小值以及 a_i 的上一次出现位置；还需要启用另一个线段树，在 $d_i = |a_{i+1} - a_i|$ 上维护区间 gcd。一次单点修改只需要进行 $\mathcal{O}(1)$ 次线段树单点修改即可维护，时间复杂度 $\mathcal{O}(n \log n)$ 。



CF1149C Tree Generator™

问题 2

一棵包含 k 个点的无标号有根树可以使用一个长度为 $2(k - 1)$ 的括号序列表示：

Description	$(())$	$()()$	$()(())()$
Tree			

给定一个长度为 $2(n - 1)$ 的括号序列，以及 q 次修改。每次修改会交换括号序列上的两个字符，保证交换后的括号序列可以作为一个无标号有根树对应的括号序列。需要在**交换前**，以及**每次交换后**输出对应有根树的直径。

数据范围： $3 \leq n \leq 10^5, 1 \leq q \leq 10^5$ 。



此处的 (和) 本质上等价于欧拉序中通过一条边搜索以及回溯的过程，对应深度分别自增或者自减。对于括号序列，将 (和) 看作 -1 和 $+1$ ，并计算前缀和数组 p_i ，那么 p_i 对应就是欧拉序中第 $i + 1$ 个点的深度（第一个点是根）。

那么对于欧拉序上第 l 个点 u 和第 r 个点 v ，令他们的最近公共祖先为 w ，考虑到两点之间的 LCA 为欧拉序区间内最浅的点，就有

$$\text{dep}_u = p_{l-1}, \text{dep}_v = p_{r-1}, \text{dep}_w = \min_{i=l-1}^{r-1} p_i$$

$$\text{dis}(u, v) = \text{dep}_u + \text{dep}_v - 2\text{dep}_w = p_{l-1} + p_{r-1} - 2 \min_{i=l-1}^{r-1} p_i$$

问题变成：在支持区间 ± 2 的同时，对所有的 $1 \leq l \leq r \leq 2n - 1$ 求 $p_{l-1} + p_{r-1} - 2 \min_{i=l-1}^{r-1} p_i$ 的最大值。



不妨考虑在线段树上维护答案，并且通过加上适当的信息实现答案的合并。考虑在线段树的 $[l, r]$ 区间处，我们已经计算了两个子区间 $[l, m], [m + 1, r]$ 的答案，那么这个区间对应的答案还包含了跨过中点的情况。

为这一种情况中 w 在左侧还是右侧进行分类讨论，就可以得到需要维护的信息：

- mn_x 维护区间的最小值；
- mx_x 维护区间的最大值；
- lmx_x 维护 $\max_{i < j} (p_i - 2p_j)$ ，对应区间内包含 u, w 的情况；
- rmx_x 维护 $\max_{i > j} (p_i - 2p_j)$ ，对应区间内包含 v, w 的情况；
- ans_x 维护当前区间的答案。

在合并时， mn 和 mx 的处理是简单的，而对于 lmx_x ，除了需要从 lmx_l 和 lmx_r 转移之外，还需要进行跨中点的转移，也就是

$$lmx_x \leftarrow \max\{lmx_l, lmx_r, mx_l - 2mn_r\}$$

对于 rmx 同理。这样我们就可以得到跨区间的答案为 $\max\{lmx_l + mx_r, mx_l + rmx_r\}$ 。这一公式包含一些不合法的情况，也即 u 或者 v 的深度比 w 低的情况，但是这部分显然是不优的，即使计算了也对答案没有影响。时间复杂度 $\mathcal{O}(q \log n)$ 。



ARC069D Flags

问题 3

有 n 个旗帜，第 i 个旗帜可以被放置在数轴上的 x_i 或者 y_i 处。令旗帜之间距离的最小值为 d ，求出 d 的最大可能值。

数据范围： $2 \leq n \leq 10^4, 1 \leq x_i, y_i \leq 10^9$ 。



这一题实际上是一道线段树优化建图的题目。这一方法在点向区间连边的最短路上被经常使用，这里提供一个不太经典的例子。

首先考虑二分 d ，这样就只需要判断如下问题：是否存在一种方案，使得旗帜两两之间至少间隔了 $d + 1$ 个单位。又注意到每个旗帜可以放置在两个位置之一，并且选择之间存在约束，故使用 2-SAT。

为每个旗帜的两种放置方式，建立点 X_i, Y_i ，这样在图中就有 $2n$ 个点。随后枚举旗帜 i, j ，进行如下操作：

- 如果 $x_j \in [x_i - d, x_i + d]$ ，那么连边 $X_i \rightarrow Y_j$ ，代表只要第 i 个旗帜放置在 x_i 处，第 j 个旗帜就无法放在 x_j 处，而只能放置在 y_j 处。
- 对于其他三种情况同理。

在这一策略下，在最坏情况下会建立 $\mathcal{O}(n^2)$ 条边。加上前面二分的复杂度，这显然是不能接受的。然而我们可以注意到，对于上面提到的 $X_i \rightarrow Y_j$ 的连边，实际上是以 y_j 作为值，向一个区间内所有的点连边。



因此考虑在 Y_j 之上建立线段树的结构，那么只需要在这棵线段树上模拟区间修改的流程，就可以找到 $\mathcal{O}(\log n)$ 个点，其能恰好到达所需的所有 Y_j ，最后从 X_i 连向这些点即可。

在上述优化下，点数为 $\mathcal{O}(n)$ ，边数为 $\mathcal{O}(n \log n)$ ，我们就可以通过 Tarjan 算法在 $\mathcal{O}(n \log n \log V)$ 的复杂度下解决这个问题（其中 V 为数轴的值域）。



CF932F Escape Through Leaf

问题 4

有一颗 n 个节点的树（节点从 1 到 n 依次编号），根节点为 1。每个节点有两个权值，第 i 个节点的权值为 a_i, b_i 。

你可以从一个节点跳到它的子树内任意一个节点上。从节点 x 跳到节点 y 一次的花费为 $a_x \times b_y$ 。跳跃多次走过一条路径的总费用为每次跳跃的费用之和。请分别计算出每个节点到达树的每个叶子节点的费用中的最小值。

注意：就算根节点的度数为 1，根节点也不算做叶子节点。另外，不能从一个节点跳到它自己。

数据范围： $2 \leq n \leq 10^5, -10^5 \leq a_i \leq 10^5, -10^5 \leq b_i \leq 10^5$ 。



设 f_u 代表从 u 跳到叶节点的答案，枚举 u 子树内的所有点 v ，则

$$f_u = \min_v (a_u \times b_v + f_v)$$

令函数 $g_v(x) = b_v x + f_v$ ，则 f_u 就是 u 子树内所有点对应的一次函数在 $x = a_u$ 时的最小值，使用李超线段树维护即可。

进一步的，由于函数 $g_v(x)$ 的内容不随着 u 的变化而变换，故可以将重儿子已经加入的函数进行复用，也即 DSU on Tree。最终的时间复杂度为 $\mathcal{O}(n \log n \log V)$ ，其中 $V = 2 \times 10^5$ 代表值域。

2. 第二部分：笛卡尔树、字典树



ARC104F Visibility Sequence

问题 5

有 n 个房子，其中 i 号房子的高度为 $[1, X_i]$ 之间的整数。通过这 n 个房子可以生成一个序列 P ，其中 P_i 代表在 i 号建筑之前第一个比它高的建筑编号（若不存在则为 -1 ）。求出在 $\prod X_i$ 种不同的高度下，可能产生的序列 P 的不同种类数。

数据范围： $1 \leq n \leq 100, 1 \leq X_i \leq 10^5$ 。



首先可以明确：一个序列 P 唯一对应一个笛卡尔树形态，从笛卡尔树推到 P 是自然的，同时也存在构造算法从 P 得到唯一的笛卡尔树，此处不再展开。

因为设立 $f_{l,r,m}$ 表示在 $[l, r]$ 范围内，房子的高度不高于 m 对应的笛卡尔树种类数。我们枚举根对应的位置 $l \leq k \leq r$ ，此时为了涵盖尽可能多的形态，我们必然会使这一位置的房子高度尽可能大，也就是 $h_k = \min(m, X_k)$ 。在钦定同高度情况下下标更小的位置更小的前提下，左子树 $[l, k-1]$ 对应的最大值就是 $h_k - 1$ ，而右子树 $[k+1, r]$ 对应的最大值就是 h_k ，因此有转移：

$$f_{l,r,x} \leftarrow \sum_{k=l}^r f_{l,k,h_k-1} \times f_{k+1,r,h_k}$$

本题 X_i 较大，需要进行离散化操作。最终的时间复杂度为 $\mathcal{O}(n^4)$ 。



洛谷 P6453 [COCI 2008/2009 #4] PERIODNI

问题 6

给定一个包含 n 列的表格，表格的每列下对齐，而高度通过数组给定。你需要在这个表格上放置 k 个车（一步可以横向或者纵向走任意格），在保证车不能在一歩内攻击到彼此的前提下，求出放置方案数。

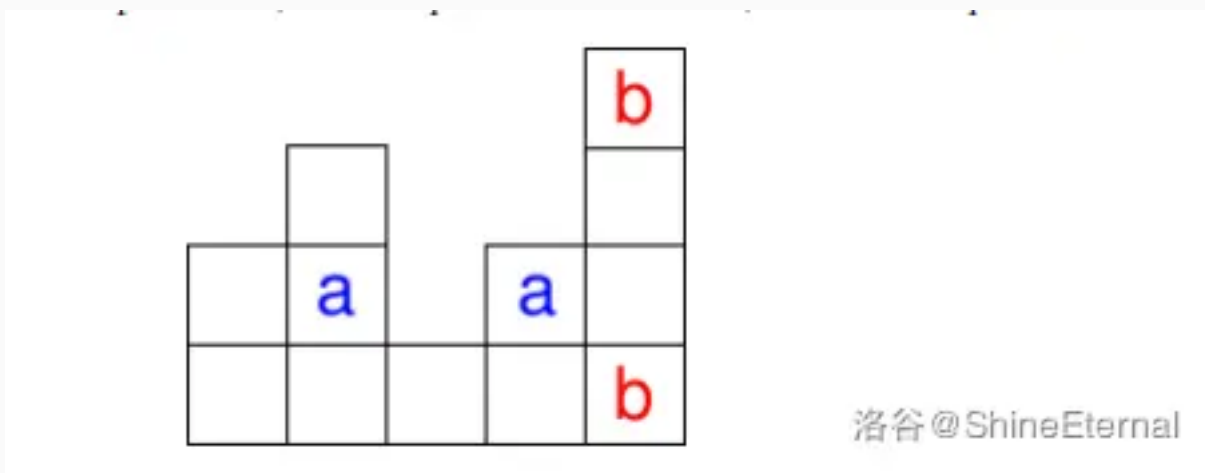


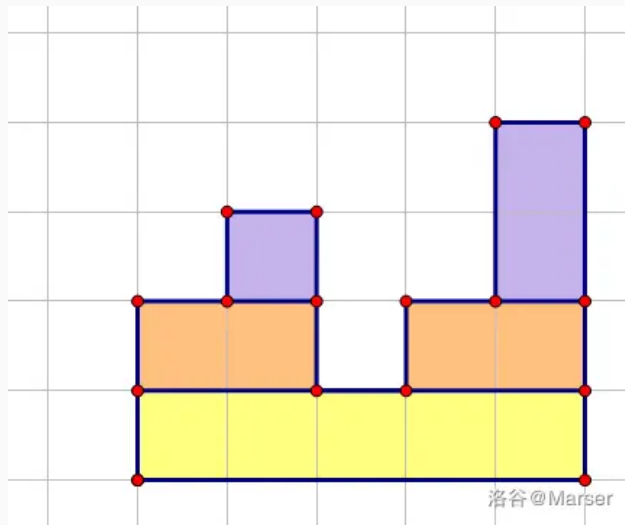
图 2 高度为 $\{2, 3, 1, 2, 4\}$ 且 $k = 2$ 的情况，此时 a 合法，但 b 不合法

数据范围： $1 \leq n, k \leq 500$ ，层高不超过 10^6 。



需要注意到：枚举列的顺序是比较重要的。如果先决定第二列放在哪再决定第一列放在哪，那么第二列是否放在最高的格子，就会对第一列的可放置位置个数产生不同影响；反之，只要第一列先放了一个车，那么第二列就必然会减少一个可以放置的位置，故从小到大安排车的位置是较为容易的。

对高度数组建立笛卡尔树，其中小的数在根上。可以根据笛卡尔树对场地进行划分：



这样就可以对笛卡尔树的每个点划分一个区域，而区域之间是互相嵌套的。设 $f_{u,i}$ 表示在 u 号点的子树对应的区域中，选择 i 个车的方案数，而 $g_{u,i}$ 表示在子树对应的区域除去 u 自身的矩形中放置车的方案数。



首先考虑计算 $g_{u,i}$ 。由于此时对应的区域不包含 u 自身的矩形，那么剩余的就恰好是两个子树对应的区域的并，而且两个区域之间没有互相连接，故直接转移：

$$g_{u,i} \leftarrow \sum_{j=0}^i f_{l,j} + f_{r,i-j}$$

随后考虑利用 $g_{u,i}$ 计算 $f_{u,i}$ 。设 u 自身的矩形高度为 H ，宽度为 W ，并枚举在除了这个矩形外的区域总共放了 j 个车。此时矩形内只有 $W - j$ 列可用，而所有行可用，为了放置 $i - j$ 个车，对应的方案数就是下面标注的部分，可以得到如下转移：

$$f_{u,i} \leftarrow \sum_{j=0}^i g_{u,j} \times \underbrace{\binom{H}{i-j} \times \binom{W-j}{i-j} \times (i-j)!}_{\text{方案数}}$$

最终的时间复杂度为 $\mathcal{O}(n^3 + V)$ ，此处 $V = 10^6$ 代表最大高度。



洛谷 P10919 运输规划

问题 7

n 座城市按顺序排成一排，每座城市存在限高 h_i 。共有 m 辆卡车，第 i 辆卡车将会从 S_i 出发，高度为 h_{S_i} 。它可以移动到相邻的城市，但前提是经过的城市限高不能低于卡车的高度。另外有 m 个城市有机场，分别为 T_1, T_2, \dots, T_m 。

每辆卡车需要前往不同的机场，令 P_i 代表到达 T_i 机场的卡车编号，显然此时 P 是一个 $1 \sim m$ 的排列。需要求出字典序最小的 P_i 。

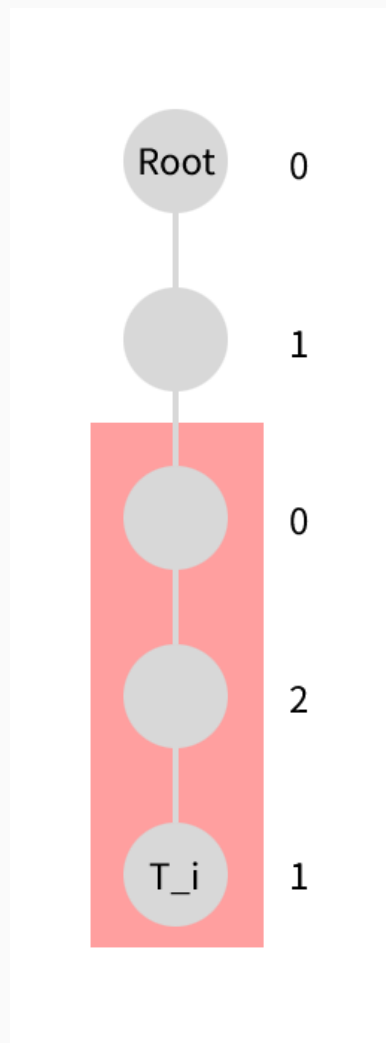
数据范围： $1 \leq m \leq n \leq 2 \times 10^5, 0 \leq h_i \leq 10^9$ ，另外有 h_i 互不相同。



对 h_i 序列建立笛卡尔树，那么第 i 辆卡车能到达的城市在笛卡尔树上等于以 S_i 为子树的所有点。由于这里需要将每辆卡车匹配到所有机场，因此可以使用 Hall 定理分析匹配成功所需的条件。最终可以发现，卡车和机场之间存在匹配，当且仅当对于笛卡尔树上任意的子树，都有卡车的数量不大于机场的数量。

因此对笛卡尔树上的每个点维护 v_i ，代表子树内机场的数量减去卡车的数量，那么在将一辆卡车和一个机场匹配后，只需要将机场到卡车的链上所有点的 v_i 减一即可维护 v_i （这里的点包括机场的点，但不包括卡车的点）。为了保证字典序最小，需要按顺序枚举每个机场，并确定满足如下条件的卡车：

- 该卡车在笛卡尔树上根到机场的链之间；
- 该卡车在和机场匹配后，依然可以保证 $v_i \geq 0$ ，也就是子树内卡车数量不大于机场数量的条件；
- 在所有符合上述两个条件的卡车中，需要选择编号最小的一个。



左侧是一个示例，其中每个点右侧标注的是 v_i 。考虑到我们修改时对应的是机场到根链上的一个前缀（越靠近机场则越前），那么可选的卡车位置对应的是机场到链上第一个 $v_i = 0$ 的点之间的链，对应红色背景的部分。

使用树链剖分加线段树维护区间的 v_i 以及卡车下标的最小值，那么只需要：

- 跳重链的同时在线段树上二分，找到链上第一个 0 所在的位置；
- 得到查询的区间后查询区间内卡车下标的最小值；
- 此时就得到了当前机场匹配的卡车，根据前面提到的方案在线段树上修改 v_i 和卡车下标。

最终的时间复杂度为 $\mathcal{O}((n + m) \log^2 n)$ 。



洛谷 P5841 [CTSC2011] 字符串重排

问题 8

定义 $\text{lcp}(A, B)$ 为字符串 A 和 B 的最长公共前缀长度。给定 n 个两两不同的字符串 S_1, S_2, \dots, S_n ，对于一个 $1 \sim n$ 的排列 P_1, P_2, \dots, P_n ，其价值为

$$\sum_{i=2}^n \left(\text{lcp}(S_{P_{i-1}}, S_{P_i}) \right)^2$$

称一个排列是好的，当且仅当其取到了价值的最大值。另外，还有 q 个附加任务，第 i 个任务给出两个不同的整数 X_i, Y_i ，如果在排列中 X_i 之后紧跟着 Y_i ，则获得 2^i 的收益。求出好的排列对应的收益最大值，并构造一个这样的排列。

数据范围： $1 \leq n \leq 4 \times 10^4, 1 \leq q \leq 10^5$ ，所有字符串的长度和不超过 2×10^5 ，字符集 Σ 为小写字母。



为了方便起见，可以在每个字符串最后加入一个特殊字符，避免字符串的前缀关系。

将字典树建出，通过调整法可以证明，一个排列是好的，当且仅当存在字典树上的 DFS 顺序，按照排列给出的顺序经过所有字符串。

由于附加任务的奖励是指数增长的，显然需要先尝试满足最后一个条件，再满足倒数第二个条件，以此类推。因此，需要在字典树上维护一个数据结构，每次尝试满足一个条件或者指出条件不能成立。

不妨考虑在 DFS 顺序中限制两个元素的顺序会发生什么。不妨考虑此时要求紧跟在 u 后面的是 v ，设 $w = \text{lca}(u, v)$ ，且 x 为 w 的所有儿子中子树包含 u 的儿子， y 则是包含 v 的儿子，则（这里加了特殊字符，因此 u, v 不存在祖先后代关系）：

- 对 $[u, x)$ 链上的所有点需要满足其作为父节点搜索时的最后一个节点；
- 对 $[v, y)$ 链上的所有点需要满足其作为父节点搜索时的第一个节点；
- 对于 w 的搜索顺序，需要满足 y 紧跟在 x 后面。



考虑维护每个非叶子节点的儿子搜索顺序，就相当于要处理如下操作，并判断顺序是否存在：

- 选定一个点作为第一个/最后一个点；
- 选定两个点并钦定相邻关系。

可以使用链表 $\mathcal{O}(1)$ 维护，此处不再展开。

最后考虑到，每次操作时都需要将 u, v 的路径完全搜索，在最劣情况下复杂度会达到 $\mathcal{O}(q \max |S_i|)$ ，无法通过。实际上，由于本题只关注儿子的搜索顺序，因此对于只有一个儿子的点，可以将该点和儿子合并为一个点。此时可以证明树的深度不会超过 $\mathcal{O}(\sqrt{\sum |S_i|})$ ，因此最终的时间复杂度为 $\mathcal{O}(q\sqrt{\sum |S_i|})$ 。



洛谷 P11611 [PA 2016] 归约 / CNF-SAT

问题 9

对于一个长度为 n 的 01 串 T ，存在如下 m 条限制：

- 给定一个区间 $[l, r]$ 以及一个长度为 $r - l + 1$ 的 01 串 s ，需要存在至少一个位置 $l \leq i \leq r$ ，使得 $T_i \neq s_{i-l+1}$ 。

求出满足所有 m 条限制的字符串 T 的个数。

数据范围： $1 \leq n \leq 10^6, 1 \leq L \leq 10^6$ ，其中 $L = \sum(r - l + 1)$ 。



考虑设立状态 $f_{i,S}$ 表示已经确定了 T 的前 i 个字符，当前没有被满足的条件集合为 S ，对应的方案数。这里给出关于 S 更严格的定义：

- S 集合仅记录了满足 $l \leq i \leq r$ 的限制，对于在 i 之前的限制，则默认它们已经被满足了。

如果一个限制 j 在此刻没有被满足，那么可以说明此时 T 的某个后缀与 j 对应的字符串的某个前缀完全相同。

0001
01001
110101
0111010????

以上是一种可能的情况，其中前三行表示了当前没能满足的三个限制。观察红色部分可以发现，它们在原字符串的 $[1, i]$ 部分截断得到的子串存在后缀关系。因此，我们可以使用“截断后得到的子串最长的限制”作为集合 S 的标识，状态也就从 $f_{i,S}$ 变成 $f_{i,j}$ 了。



考虑在 i 时刻，将截断的子串逆序建立字典树，那么后缀性质就变成了前缀性质，所有不满足的限制在字典树上就形成了一条 j 到根的链。接下来考虑从 i 转移到 $i + 1$ ，逐步考虑如下情况：

- 一些限制在 i 处结束，而如果这些限制没有被满足，则不进行转移。在字典树上，这表现为将一些点的子树对应的贡献清空；
- 一些限制从 $i + 1$ 开始，我们预先将它们放在字典树的根上；
- 一些限制随着转移会多出一个字符，如果多出的字符为 θ ，则称为 θ -扩展限制，否则称为 1 -扩展限制。对于 $f_{i,j}$ ，如果 T 的下一个字符为 θ ，则需要找到字典树中 j 到根的链上第一个 θ -扩展的位置，并转移到此处，对 1 同理。在字典树上，只需要进行一次 DFS，储存最近的 θ -扩展和 1 -扩展位置即可；
- 由于状态转移，字典序的形态发生改变。由于我们在字典树上逆序储存字符串，只需要：
 - 通过暴力搜索得到所有 θ -扩展位置的虚树，记为 T_0 ，同理得到 T_1 ；
 - 定义一个新的根 R ，将 R 的“ θ -儿子”设为 T_0 ，“ 1 -儿子”设为 T_1 ， R 就是 $i + 1$ 时刻对应的字典树。

如果使用 `map` 储存 $f_{i,j}$ ，则复杂度为 $\mathcal{O}((n + L) \log n)$ ；如果直接在字典树的点上储存 $f_{i,j}$ ，则复杂度为 $\mathcal{O}(n + L)$ 。

3. 第三部分：数据结构杂题



洛谷 P7011 [CERC2013] Escape

问题 10

有一个树状地图，包含 n 个点，其中第 i 个点包含一个点权 a_i ，点权为正代表可以在此回血 a_i 个单位，为负则代表需要在此掉血 $-a_i$ 个单位。每个点的事件仅在第一次到达的时候触发一次。

一个人携带初始血量 0，需要从 1 号点前往 t 号点，这个人每次可以前往与其所在位置相邻的点。求出这个人是否能在保证中途血量不小于 0 的前提下完成目标。

数据范围： $2 \leq n \leq 2 \times 10^5, \sum n \leq 5 \times 10^5, 2 \leq t \leq n, |a_i| \leq 10^6$ 。



为了判断是否能到达 t 号点，可以新建一个回大量血 M 的点 $n+1$ 并与 t 相连，只要最终能够得到对应的血量则说明能够完成目标。

以 1 为根进行搜索，尝试对每个点 u 维护以 u 为子树对应的策略。令 S_u 为如下策略的集合：

- (x, y) ，表示付出 x 的血量进入 u 所在的子树，可以获得 y 的血量。这一策略应该满足如下性质：
 - 我们只关注正向收益，因此这里有 $x < y$ ；
 - 收益应当是循序渐进的，更大的付出对应更大的收益，也即不存在两个策略 $(x_1, y_1), (x_2, y_2)$ ，使得 $x_1 < x_2$ ，而 $y_1 > y_2$ 。
 - 在循序渐进的前提下，策略应当是最简的，也即不存在两个策略 $(x_1, y_1), (x_2, y_2)$ ，使得 $x_1 \leq x_2 \leq y_1$ （此时可以直接将第一个策略回的血用于第二个策略中，得到更大的收益）。

此时可以证明，一个点对应的策略集合 $S_u = \{(x_1, y_1), (x_2, y_2), \dots\}$ 在数轴上可以表示为不交的区间。对于当前节点 u ，只需要将其所有儿子 v 对应的 S_v 进行启发式合并，得到临时集合 T ，然后加上 a_u 本身的贡献即可得到 S_u 。



在启发式合并的过程中，根据最简的原则，如果发现插入 (x_1, y_1) 后，集合上的下一个策略 (x_2, y_2) 满足 $x_1 \leq x_2 \leq y_1$ ，那么就可以将二者进行合并，得到 $(x_1, y_1 + y_2 - x_2)$ ，这一流程持续到不能合并为止。可以证明，在这一策略下得到的策略收益也是循序渐进的。

随后就是处理 u 号点自身的贡献：

- $a_u > 0$ ，说明进入即可获得血量，向集合 T 加入 $(0, a_u)$ 即可，需要进行合并。
- $a_u < 0$ ，为了能吃到正收益，我们需要保留不低于 $-a_u$ 的血 k ，贪心选取策略直到达到正收益。考虑模拟这一思想：
 - 初始时 $k = -a_u$ ；
 - 不断从 T 中选取 x 最小，也就是最容易达成的策略 (x, y) ，此时：
 - 如果 $x + k < y$ ，说明加上这一策略就可以达成正收益，故向 T 加入 $(x + k, y)$ 并结束流程；
 - 否则，贪心的加上这一策略， $k \leftarrow k + x - y$ 。

合并完成后，检查根处的第一个策略是否为：“无需付出血量就可以得到至少 M 的血量”，即可得到答案。时间复杂度 $\mathcal{O}(n \log^2 n)$ 。



洛谷 P5113 Sabbath of the witch

问题 11

给定一个长度为 n 的序列 a_i ，需要支持 m 次操作，**强制在线**。操作的种类如下：

- 区间赋值；
- 求区间和；
- 撤销曾经的一次区间复制操作。

数据范围： $1 \leq n, m \leq 10^5$ 。



考虑进行分块，对于一个块内部记录如下信息：

- S ，代表整块赋值的操作对应的集合，用于撤销后得到最近一次整块赋值，另外记 T 为其中最大的元素；
- V_i ，代表散块赋值时第 i 个位置对应的操作集合，另外计 T_i 为其中最大的元素。

称 $T_i > T$ 的位置 i 为自由的位置（因为散块修改比整块晚，值等于最近一次散块修改的值），而 $T_i < T$ 的位置 i 为受控的位置。要想查询整块的和，只需要额外维护自由位置的修改值总和 Σ 和受控位置的个数 C 即可。

接下来考虑处理所有的操作：

- 赋值（整块）：向 S 插入当前的修改时刻。此时所有的位置都是受控的， Σ 变为 0， C 变为块长度；
- 赋值（散块）：暴力插入 V_i 并直接重构；
- 查询：散块查询是平凡的，整块查询已经提到过了；



- 撤销（整块）：将对应的时刻从 S 移除，这会使得一些受控位置变成自由位置，枚举对应的位置并修改 Σ 和 C 即可；
- 撤销（散块）：暴力从 V_i 删除并直接重构。

注意到我们需要在撤销整块的时候高效枚举从受控转为自由的位置。为此考虑在重构块的同时将所有的位置按照 T_i 排序，那么自由的位置对应为排序后的后缀。维护这一后缀对应的指针，每次撤销的时候不断向左移动即可做到高效枚举。整块赋值的时候，所有位置都是受控的，将这一指针移动到最右边即可。

令块长度为 B ，接下来分析复杂度。散块赋值和撤销的复杂度为单次操作 $\mathcal{O}(B \log B)$ 或者 $\mathcal{O}(B)$ （若使用基数排序）。整块赋值的复杂度为单次操作 $\mathcal{O}(\frac{n}{B})$ ，查询的总复杂度为单次操作 $\mathcal{O}(\frac{n}{B} + B)$ 。而对于整块撤销，考虑到重构的次数最多为 $\mathcal{O}(m + \frac{n}{B})$ 次（包括初始化阶段），每次移动次数不超过 B ，则对应的复杂度为总体 $\mathcal{O}(mB + n)$ 。

认为 n, m 同阶，在平衡后可以得到本题的时间复杂度为 $\mathcal{O}(n\sqrt{n \log n})$ 或 $\mathcal{O}(n\sqrt{n})$ 。



CF102586 L. Yosupo's Algorithm

问题 12

在二维平面上给出 n 个红点和 n 个蓝点，所有红点位于第二象限，所有蓝点位于第一象限，且每个点存在一个正权值。需要回答 q 次询问，每次询问给出两个数字 $L < 0 < R$ ，你需要选出满足如下条件的红点和蓝点，并求出二者的权值和的最大值（找不到则输出 -1）：

- 蓝点的纵坐标比红点大；
- “红点在 $x = L$ 左侧”与“蓝点在 $x = R$ 左侧”中恰有一个成立。

数据范围： $1 \leq n \leq 10^5, 1 \leq q \leq 5 \times 10^5$ ，题目中出现的所有横坐标（包括 L, R ）互不相同，所有纵坐标也互不相同。



一种最基本的想法就是：枚举所有满足蓝点比红点高的点对，假设其中红点和蓝点的横坐标分别为 a, b ，那么将 (a, b) 放在另一个二维问题上，询问就变成了对如下两个矩形包含点的最大权值的问题：

- $a < L, b > R$;
- $a > L, b < R$ 。

这两个矩形都是延伸到无穷远处的，故离线后扫描线就能够解决。

本题的优化瓶颈在于减少点的数量。实际上，只有 $\mathcal{O}(n \log n)$ 个点对是有用的。将所有 $2n$ 个点按照纵坐标排序，并进行分治，在考虑 $[l, r]$ 区间的点对的时候，我们将其分为如下部分：

- $[l, m]$ 内部的点对，以及 $[m + 1, r]$ 内部的点对，这是递归的子问题；
- 红点在 $[l, m]$ 内部且蓝点在 $[m + 1, r]$ 内部的点对，这就是目前需要讨论的内容。



找到 $[l, m]$ 内部的最大权红点 $\max r$ ，以及 $[m + 1, r]$ 内部的最大权蓝点 $\max b$ ，可以证明：这部分的点对中只有包含了 B 或者 R 才是有用的。

证明：使用反证法，假设对于一次查询 L, R ，仅考虑当前讨论的点对时，点对 (b_i, r_j) 取得了最大权，其中 $b_i \neq \max b$ ， $r_j \neq \max r$ 。

- 如果 b_i 和 $\max b$ 落在 $x = R$ 同侧，那么将 b_i 换为 $\max b$ 就得到了权值更大的点对，不符合条件，对于 r_i 和 $\max r$ 同理。
- 如果 b_i 和 $\max b$ 落在 $x = R$ 异侧， r_j 和 $\max r$ 也落在 $x = L$ 异侧，那么 $(\max b, \max r)$ 就是符合条件的点对，显然也比 (b_i, r_j) 拥有更大的权值，不符合条件。

因此，区间 $[l, r]$ 内只会产生最多 $r - l + 1$ 个有用的点对，最终就会得到 $\mathcal{O}(n \log n)$ 个点对。时间复杂度为 $\mathcal{O}(n \log^2 n + q \log n)$ 。



洛谷 P5044 [IOI 2018] meetings 会议

问题 13

有一个长度为 n 的数组 H ，需要在线回答 q 次询问。每次询问提供一个区间 $[l, r]$ ，对于一个位置 $l \leq k \leq r$ ，通过如下方式定义其代价：

- 对 $\forall i \in [l, r]$ ，计算 H 数组上第 i 个位置与第 k 个位置之间的最大值，并累加。

求出所有位置对应的代价最小值。

数据范围： $1 \leq n \leq 7.5 \times 10^5, 1 \leq q \leq 7.5 \times 10^5, 0 \leq H_i \leq 10^9$ ，时间限制为 4.5 秒。



对于此类问题，显然需要建立笛卡尔树，那么区间最大值就变成了笛卡尔树上的 LCA。考虑到直接处理询问存在笛卡尔树高度等问题，考虑使用类似于猫树的思想，找到笛卡尔树上的一个点 u ，使得：

- 当前查询的区间 $[l, r]$ 被完全包含在 u 的子树内；
- u 自身被包含在 $[l, r]$ 中。

那么区间 $[l, r]$ 就被拆分为 $[l, u-1], \{u\}, [u+1, r]$ 三部分，而前后两部分恰好对应笛卡尔树中序遍历上的一段前缀或者后缀，这样就可以通过计算前后缀答案合并出最终的结果了。

设 u 号点对应的子树包含了序列的 $[L_u, R_u]$ 区间，并设 $f_{u,i}$ 表示 $[L_u, i]$ 区间对应的答案， $g_{u,i}$ 则是 $[i, R_u]$ 区间对应的答案。在 $L_u \leq i \leq R_u$ 的前提下，整个问题就只需要在子树内考虑了，而且最后只需要通过 $g_{l_{s_u}, l}$ 和 $f_{r_{s_u}, r}$ 即可合并答案。

接下来讨论如何计算 $f_{u,i}$ ， $g_{u,i}$ 同理。



对于笛卡尔树中序遍历的一个前缀，可以通过 i 和 u 的大小关系进行分类：

- 如果 $i < u$ ，说明 $[L_u, i]$ 区间被完全包含在 u 的左子树中，有 $f_{u,i} \leftarrow f_{\text{ls}_u,i}$ ；
- 如果 $i = u$ ，选择 u 肯定是不优的，自然可以得到 $f_{u,i} \leftarrow f_{\text{ls}_u,u-1} + H_u$ 。
- 如果 $i > u$ ，说明区间 $[L_u, i]$ 完全包含了左子树，同时也包含了右子树的一个前缀。此时就需要讨论选择的位置在左子树还是右子树，可以得到：

$$f_{u,i} \leftarrow \min \left\{ f_{\text{ls}_u,u-1} + (i - u + 1)H_u, f_{\text{rs}_u,i} + (u - L_u + 1)H_u \right\}$$

需要注意到， i 每增加 1， \min 的左参数变化量恰好为 H_u ，而右参数变化量则不超过 H_u （相当于在右子树的问题中加入了一个点，这个点的新贡献只会比 H_u 小），那么存在一个时刻 i' ，使得在 $i \leq i'$ 时，左参数较小，而 $i > i'$ 时右参数较小。

使用动态开点线段树维护每个位置对应的 $f_{u,i}$ ，在直接继承 $f_{\text{ls}_u,i}$ 之后，线段树上二分找到上述问题对应的 i' ，随后在 $[u, i']$ 处区间加等差数列，在 $[i' + 1, R_u]$ 处复用 $f_{\text{rs}_u,i}$ 的后半部分，然后加上常数 $(u - L_u + 1)H_u$ 。时间复杂度为 $\mathcal{O}((n + q) \log n)$ 。



洛谷 P6109 [Ynoi2009] rprmql

问题 14

给定一个 $n \sim n$ 的矩阵，初始全为 0，接下来将会对该矩阵进行如下操作：

- 先进行 m 次修改操作，给定一个子矩阵和一个整数 x ，并进行子矩阵加 x 的操作；
- 再进行 q 次查询操作，给定一个子矩阵，求出这个子矩阵中的数字最大值。

本题先修改后查询，并且修改次数和查询次数的量级不同。

数据范围： $1 \leq n, m \leq 5 \times 10^4, 1 \leq q \leq 5 \times 10^5, 1 \leq x < 2^{31}$ 。



考虑到对操作处理的复杂度要求较高，这里就不能将询问放在线段树的 $\mathcal{O}(\log n)$ 个区间处理了。依然使用类似猫树的思想，根据第一个维度建立线段树，在尝试插入一个询问矩形的时候，在线段树遇到的第一个分界点上就变成了一个后缀问题和一个前缀问题。

至于修改部分，考虑到修改较少，故直接采用和线段树分治类似的方式将其拆分成 $\mathcal{O}(\log n)$ 个线段树位置上的操作即可：

- 如果矩形的第一维度未能完全覆盖当前的第一维区间，则需要将矩形分成两半，作为历史最大值线段树操作的一部分，并且递归下去；
- 如果矩形的第一维度完全覆盖当前的第一维区间，则相当于在此处打下了一个永久标记，只需要在递归子问题前加上这个矩形，在最后删掉即可。

这样就在线段树上把一个矩阵上的问题转换为了询问紧贴左边界或者右边界的矩形的修改和查询问题，再通过扫描线即可转换为区间历史最大值问题。需要精细化处理矩阵边界的操作顺序。

在计算中途，需要将维护的“历史最大值”置为当前的最大值。为了实现这一点，可以全局加上一个足够大的数字，使得当前的数字加上后必然会超过历史的最大值即可（加上 $\max x \times m$ 就是一个选择，而且刚好不会爆 long long）。最终的时间复杂度为 $\mathcal{O}(m \log^2 n + q \log n)$ 。



CF1110H Modest Substrings

问题 15

给定两个大数 l, r ，对于一个十进制下的 n 位大数 X （可以有前导 0），定义其分数为这个大数的十进制表示中满足如下条件的子串数：

- 这个子串不包含前导 0；
- 这个子串在十进制表示下恰好在 l, r 之间。

给出大数能取得的最大分数，并构造出取得最大分数的最小 X 。

数据范围： $1 \leq n \leq 2000, 1 \leq l \leq r \leq 10^{800}$ 。



一种最暴力的想法就是将 $[l, r]$ 之间所有数字的十进制表示插入到一个 AC 自动机上，然后在这个自动机上跑动态规划。简单来说，令 $f_{i,j}$ 为填入了前 i 个字符，并且当前在 AC 自动机的 j 状态上对应的最大分数。 j 状态带来的贡献 g_j 定义为从该状态开始跳 fail 树能跳到的所有点的权值和。当然，这种做法的时间复杂度显然无法接受。

事实上， $[l, r]$ 之间的整数可以使用大约 $\Sigma \times (|l| + |r|)$ 量级的正则表达式完全覆盖，其中每个正则表达式的形式如下： $\text{xxxx}[0-9]\{\text{xxx}\}$ 。例如 $[1789, 2100]$ 中的整数，就可以使用 1789 , $179[0-9]\{1\}$, $18[0-9]\{2\}$, $19[0-9]\{2\}$, $20[0-9]\{2\}$, 2100 这六个正则表示覆盖。每个正则表达式在插入到字典树之后呈现的就是一条链下连着满十叉树的情况。

考虑在字典树上将这些满十叉树缩到其根节点上，而只保留深度信息。设 $g_{u,i}$ 代表在 u 状态下，任意追加 i 个字符，能得到的固定收益是多少，那么对于 $\text{xxx}[0-9]\{\text{cnt}\}$ 的情况，先将其前缀插入到字典树上，得到状态 u ，然后让 $g_{u, \text{cnt}}$ 加上 1 即可。



剩余的部分就和初始的暴力算法很类似了。求出字典树的 fail 指针，然后对每个 u ，通过 fail 数将所有祖先的 g 数组累加到 g_u 上即可。紧接着考虑动态规划，令 $f_{i,j}$ 为填入了前 i 个字符，并且当前在 AC 自动机的 j 状态上对应的最大分数，那么：

$$f_{i,j} \leftarrow \max_{i \in \text{trans}(k)} f_{i-1,k} + \sum_{l=0}^{n-i} g_{i,l}$$

其中 $\text{trans}(k)$ 表示 k 通过单个字符能够到达的状态集合。处理 g_u 的前缀和即可快速转移，总的复杂度为 $\mathcal{O}(\Sigma^2 \times (|l| + |r|) \times n)$ ，其中利用了上述正则表达式的一些性质。

为了构造出方案，只需要对每个动规状态 (i, j) 从其更新来源的所有 $\arg \max$ 向其连边，然后从头开始搜索，贪心选择字符小的边，在到达最终状态之后写下路径上所有字符即可。复杂度和转移的复杂度一样。



CF1637H Minimize Inversions Number

问题 16

给定一个 $1 \sim n$ 的排列 p 。对所有 $0 \leq k \leq n$ 求解如下问题的答案：

- 你可以将序列的 k 个位置取出，并按照顺序放置在 p 的最前面。求出在执行该操作恰好一次后得到的新排列中逆序对个数的最小值。

数据范围： $1 \leq \sum n \leq 5 \times 10^5$ 。



CF1637H Minimize Inversions Number

问题 17

给定一个 $1 \sim n$ 的排列 p 。对所有 $0 \leq k \leq n$ 求解如下问题的答案：

- 你可以将序列的 k 个位置取出，并按照顺序放置在 p 的最前面。求出在执行该操作恰好一次后得到的新排列中逆序对个数的最小值。

数据范围： $1 \leq \sum n \leq 5 \times 10^5$ 。

这里提供一个提示：通过打表发现， $k + 1$ 对应的答案选出的下标恰好是 k 对应的答案选出的下标新增一个下标的结果。



对于每个位置 i ，将其移动到最左边带来的逆序对减少量为 i 前面比 p_i 大的数字个数减去 i 前面比 p_i 小的数字个数。下面使用 d_i 表示这一值。

假设选出的长度为 k 的子序列为 i_1, i_2, \dots, i_k （这是下标组成的序列，有 $i_i < i_{i+1}$ ），另有 $q_j = p_{i_j}$ ，那么只需要将 d_{i_j} 涉及到的范围除掉被选出的位置即可，可以得到减少量等于

$$\sum_{i=1}^k d_{i_j} + \binom{k}{2} - 2 \operatorname{inv}(q_1, \dots, q_k)$$

其中 inv 代表序列的逆序对数量。因此本题的任务是最大化

$$\sum_{i=1}^k d_{i_j} - 2 \operatorname{inv}(q_1, \dots, q_k)$$

接下来证明一个结论，对于一个逆序对 (i, j) ，若选择了 i 作为子序列的一部分，而没有选择 j ，则放弃 i ，转而选 j 一定不会更优。

3.7 题目 7



以 (i, p_i) 作为平面上的点，通过 (i, p_i) 和 (j, p_j) 可以将平面划分为九个部分。下面使用 \overline{S} 表示 S 部分被选中的位置个数。令 (i, j) 是满足“ i 被选， j 未被选”的最近逆序对（也即 $j - i$ 最小），那么 $B1 = \overline{B1}$, $B2 = 0$, $\overline{B3} = 0$ 。

在放弃 i 而选择 j 之后， $\sum d_i$ 的增加量为：

$$\begin{aligned} d_j - d_i &= (A3 + B3 + A2 + B2 - A1 - B1 + 1) - (A3 - A2 - A1) \\ &= B3 + 2A2 - B1 + 1 \end{aligned}$$

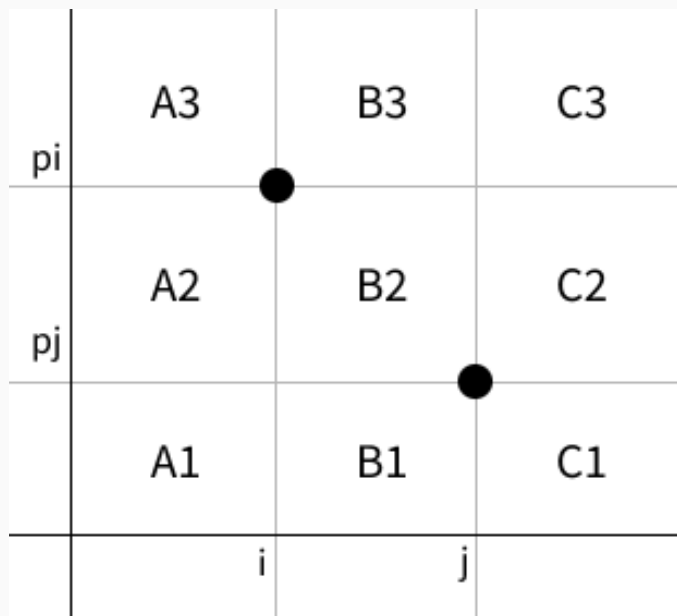
而 $\text{inv}(q_1, \dots, q_k)$ 的增加量为：

$$\begin{aligned} &(\overline{A3 + B3 + A2 + B2 + C1}) - (\overline{A3 + B1 + C1 + B2 + C2}) \\ &= \overline{A2 - C2 - B1} \end{aligned}$$

则 $\sum_{i=1}^k d_{i_j} - 2 \text{inv}(q_1, \dots, q_k)$ 的增加量为

$$\begin{aligned} &B3 + 2A2 - B1 + 1 - 2 \times \overline{A2 - C2 - B1} \\ &= B3 + 2(A2 - \overline{A2}) + 2\overline{C2} + B1 + 1 \end{aligned}$$

由于 $A2 \geq \overline{A2}$ ，那么增加量为正，选择 j 一定更优，使用调整法即证。





记 u_i 为 i 后面且比 p_i 小的位置个数。根据上述定理，可以直接改写 $\text{inv}(q_1, \dots, q_k)$:

- 对于每个位置 i_j ，本来需要计算 i_j 之后且比 q_j 小且被选中的位置个数（也即 u_{i_j} 中被选中的部分），但是根据定理，只要一个位置满足在 i_j 之后且比 q_j 小，那么这个位置就必然被选中，故位置的个数恰好等于 u_{i_j} 。

综上所述，需要最大化的值就变成了

$$\sum_{j=1}^k (d_{i_j} - 2u_{i_j})$$

选择 $d_i - 2u_i$ 的前 k 大值，从原序列的逆序对中减去即为答案。时间复杂度为 $\mathcal{O}(n \log n)$ 。



CF1707F Bugaboo

问题 18

对于一个序列 a_1, a_2, \dots, a_n ，定义一次变换为变为序列 b_1, b_2, \dots, b_n ，其中 $b_i = a_i \oplus a_{\{(i \bmod n)+1\}}$ ， \oplus 表示异或。

给定 n, t, w ，称一个序列 c_1, c_2, \dots, c_n ($0 \leq c_i < 2^w$) 是好的，当且仅当存在一个序列 a_1, a_2, \dots, a_n 经过 t 次变换后恰好为 c 。

给定序列 c ，但其中有一些位置缺失了，只有 m 个位置知道，又有 q 次修改，每次修改给定 f, g, p ，表示把 c_f 变为 g ，若 $g = -1$ 表示变为缺失。你需要在每次修改后求出有多少把缺失位置填上任意数的方案是好的 c 是一个好的序列，答案模 p 输出。

数据范围： $2 \leq n \leq 10^7, 0 \leq m \leq \min(n, 10^5), 1 \leq t \leq 10^9, 1 \leq w \leq 30, 1 \leq q \leq 10^5$ 。



为了方便后续的讨论，数组下标统一改成 0。

首先考虑操作之后的情况（指数代表每个数字被异或的次数）：

$$\bullet a_0^1 \rightarrow a_0^1 a_1^1 \rightarrow a_0^1 a_1^2 a_2^1 \rightarrow a_0^1 a_1^3 a_2^3 a_3^1 \rightarrow \dots$$

可以发现指数实际上就是组合数，那么对应的，在操作 t 此后， a_j 对 a_i 的贡献就是 $\binom{t}{(j-i) \bmod n} + \binom{t}{(j-i) \bmod n+n} + \dots$ 。由于只有贡献次数为奇数时才会计算答案，根据卢卡斯定理，组合数 $\binom{t}{k}$ 是奇数当且仅当二进制下 k 是 t 的子集，我们就刻画出了贡献的表达式：

$$a'_i = \bigoplus_{d=0}^{n-1} \bigoplus_{k \geq 0, dn+k \leq t} [dn+k \text{ 在二进制上是 } t \text{ 的子集}] a_{(i+k) \bmod n}$$



为了方便后续讨论，我们首先考虑 $n = 2^K$ 的情况。此时可以发现，在 $t = n$ 时，所有位置对应的贡献都是 0，因此 t 次操作后数组就全为 0 了。此处继续考虑 $t < n$ 的情况。

接下来对 t 的奇偶性进行讨论：

- 如果 t 是偶数，则上式中所有奇数 d 都不可能带来任何贡献（因为 $dn + k$ 是奇数，不可能是偶数 t 的二进制子集），因此这里偶数下标和奇数下标的贡献是恰好分离的。将偶数下标和奇数下标的数组分离，分别计算 $\frac{t}{2}$ 对应的答案之后相乘即可；
- 如果 t 是奇数，那么先进行一次操作，这样就变成了偶数的情况。需要注意，为了使得偶数情况下得到的数组能够进行逆操作，需要满足奇数位置和偶数位置的异或和相等。

这样我们就能够设立状态了： $f_{d,k,t,\Delta}$ 表示在递归的第 d 层， $\{c_\Delta, c_{2^d+\Delta}, \dots, c_{n-2^d+\Delta}\}$ 数组通过填写 -1 能产生多少个数组 c' ，使得存在一个异或和为 k 的数列在操作 t 次后得到 c' 。



那么有

- $d = \log n = K$ 时, 答案是平凡的, 否则:
- t 是偶数时,

$$f_{d,k,t,\Delta} \leftarrow \sum_{k'=0}^{2^w-1} f_{d+1,k',\frac{t}{2},\Delta} \times f_{d+1,k\oplus k',\frac{t}{2},\Delta+2^d}$$

- t 是奇数时, 在操作一次后奇偶两部分的异或和恰好等于原序列的异或和 k , 因此

$$f_{d,k,t,\Delta} \leftarrow f_{d+1,k,\lfloor \frac{t}{2} \rfloor, \Delta} \times f_{d+1,k,\lfloor \frac{t}{2} \rfloor, \Delta+2^d}$$

如果能做到只储存 d, t, Δ , 那么就只包含 $\mathcal{O}(n)$ 个状态, 此时 $0 \leq d \leq \log n = K, 0 \leq \Delta < 2^d$, 而每个 d 唯一确定一个 t 。自然想到特殊考虑 k 对应的维度, 那么考虑固定 d, t, Δ , 观察 $f_{d,k,t,\Delta}$ 的形态:



- 在 $d = K$ 时, $f_{d,t,k,\Delta}$ 要么是全 1, 要么只有一个位置是 1;
- 在 t 为偶数时, 相当于对两个数组进行了异或卷积。可以发现, 卷积结果应当只有如下情况:
 - 只有一个位置的值等于 2 的幂次, 其他位置全 0;
 - 所有位置的值相同, 均为 2 的幂次。
- 在 t 为奇数时, 则是对位相加, 因此除了上面提到的情况外, 还有另一种情况:
 - 所有位置的值都是 0。

不难发现上述三种情况无论是对位相加还是异或卷积都是封闭的, 我们就可以通过维护情况和幂次 $\mathcal{O}(1)$ 合并数组。对给定的数组进行下标二进制位反转后建立线段树, 上面给出的 $\{c_\Delta, c_{2^d+\Delta}, \dots, c_{n-2^d+\Delta}\}$ 的形式就变成了线段树上的一个点, 因此可以在 $\mathcal{O}(n + (m + q) \log n)$ 的时间复杂度内完成这个问题。

实际上, 在最后直接将 f 求和并不总是良定义的, 因为定义式中, 如果 $t \geq 1$, 那么每个 c' 可能对多个 k 有贡献。为了解决这个问题, 有一个实现技巧: 如果 $t \geq n$, 只需要判断能否做到每个位置全 0, 即可直接回答; 否则, 在递归到叶子的时候, $t = 0$, 那么每个 c' 只会对其异或和产生贡献, 整体就是良定义的。具体的论证细节不再展开。



最后讨论 n 不是 2 的幂次的情况。假设 K 为 n 的 lowbit 的对数，也就是 n 等于 2^K 乘以一个奇数 R 。

和上面的讨论类似，我们进行 K 次分治后，得到的子问题形如：

- 求异或和为 k 且 t 次操作后能得到 $\{c_\Delta, c_{2^K+\Delta}, \dots, c_{n-2^K+\Delta}\}$ 的数组数量。

此时这一数组的长度恰好为 R ，是一个奇数。以下对奇数情况下对应的操作性质进行讨论：

奇数长度下满足的性质

定理 19

对 a_0, \dots, a_{R-1} 进行一次操作后得到 b_0, \dots, b_{R-1} ，此时可以发现 b 数组的异或和为 0。和偶数长度的情况不同，给定 c ，只要其异或和为 0，我们就可以执行逆操作得到一种可能的 a 。

例如， $R = 5$ 时，就可以通过 $c_1 \oplus c_3$ 从总的异或和中消去 $a_1 \oplus a_2, a_3 \oplus a_4$ ，最终得到 a_0 。其他位置同理。

此时得到的 a 依然满足异或和为 0（因为每个 c_i 贡献了恰好两次），我们就可以继续对 a 进行逆操作，以此类推。考虑对逆操作 t 次得到的数组 a ，将它们全部异或上 k ，就是唯一的一个异或和为 k ，且在 t 次操作后等于 c 的数列了。



因此，对于该位置的 $f_{d,k,t,\Delta}$ ， $t = 0$ 的情况是平凡的，而如果 $t \geq 1$ ，考虑到只要对 c 填写 -1 得到的数组 c' 的异或和是 0，就总能通过前面的性质构造出一个异或和为 k ，且 t 次操作后等于 c' 的数列。

复用前面提到的实现技巧，如果 $t \geq 2^K$ ，只需要对每个 Δ 确定使得 $\{c_\Delta, c_{2^K+\Delta}, \dots, c_{n-2^K+\Delta}\}$ 异或和为 0 的方案数，全部乘起来就是答案；否则， $t < 2^K$ ，递归到上面的情况时就有 $t = 0$ ，处理起来就和 n 是 2 的幂次部分完全一样了。

因此对整道题目来说，只需要维护一个高度为 $K + 1$ 的线段树，利用前面的讨论在叶节点稍微处理一下权值情况就好了。时间复杂度为 $\mathcal{O}(n + (m + q) \log n)$ 。