



CSP-S 公益营第一套

性质转化与解法推导
2026年10月18日

信星社 | 2026

编者:姜宗沛

四题的切入点

题目	先解决的问题	方法
排队	删谁会改变右侧记录?	固定右端点, 维护最后一个不合格位置
字符串	前缀关系怎样变成位置关系?	二叉树区间, 前后缀贪心
凸序列	怎样只数不同的截平结果?	最小延拓, 计数 DP, 后缀和
gcd 完备	缺哪个值会破坏完整性?	约数窗口, 扫描线, 历史贡献

T1 排队

给定一个 $1 \sim n$ 的排列, 每个人记录左侧比自己矮的人数。

删除一个非空区间, 要求所有留下来的人的记录不变。求合法删除区间数。

$1 \leq n \leq 2 \times 10^5$; 允许删除整个排列。

样例: $a = (3, 1, 2)$, 答案为 4。

哪些人的记录会改变?

删除 $[l, r]$, 左侧的人不受影响。

对右侧位置 $j > r$, 记录减少量为

$$\#\{i \in [l, r] : a_i < a_j\}.$$

因此, 删除合法当且仅当

$$\min_{l \leq i \leq r} a_i > \max_{r < j \leq n} a_j.$$

右侧为空时, 把右侧最大值记为 0。

朴素计数:枚举删除区间

记右侧最大值为 $M_r, M_n = 0$ 。由合法条件, 答案为

$$\text{ans} = \sum_{r=1}^n \sum_{l=1}^r \left[\min_{l \leq i \leq r} a_i > M_r \right].$$

固定 r , 让 l 从 r 向左移动, 维护区间最小值。

后缀最大值 M_r 预处理后, 每个区间可 $O(1)$ 判定, 总时间 $O(n^2)$ 。

接下来研究: 固定 r 后, 哪些 l 会通过判定?

定一求一: 固定右端点

设 $M_r = \max(a_{r+1}, \dots, a_n)$, $M_n = 0$ 。

在 $1 \sim r$ 中, 设最后一个满足 $a_i < M_r$ 的位置为 p_r ; 不存在时 $p_r = 0$ 。

合法左端点恰好是

$$l \in [p_r + 1, r].$$

所以

$$\text{ans} = \sum_{r=1}^n (r - p_r).$$

排列元素互异, 左侧不会出现 $a_i = M_r$ 。

栈内按什么顺序存下标？

栈中保存已扫描、且尚未弹出的下标。从栈底到栈顶：

$$s_1 < s_2 < \cdots < s_k \leq r.$$

下标递增，身高 a_{s_i} 不要求单调。每次从最靠右的位置开始检查。

栈内可能还藏着合格位置；我们只需找到最靠右的不合格位置 p_r 。

栈初始为空，右端点 r 从 1 扫到 n 。

右端点每右移一次,做什么?

当前扫描到 r , 右侧最大值为 $M_r = \max(a_{r+1}, \dots, a_n)$; $M_n = 0$ 。

1. 将新进入扫描范围的下标 r 入栈。
2. 若栈非空且 $a_{\text{top}} > M_r$, 弹栈并重复检查。
3. 停止时, 栈顶就是 p_r ; 栈空则 $p_r = 0$ 。
4. 答案累加 $r - p_r$, 再让 r 右移一位。

排列元素互异, 栈顶留下时满足 $a_{\text{top}} < M_r$ 。

为什么弹掉的位置不用再看？

r 右移时, 右侧范围缩小, M_r 非增。

位置 i 被弹出时, 已有 $a_i > M_r$, 以后也始终合格, 不会再成为分界线。

真正的 p_r 不会被提前弹掉; 清除它后面的合格位置后, 它就在栈顶。

新下标入栈可能使 p_r 向右跳, 不能认为 p_r 单调左移。

每个下标入栈一次、出栈至多一次: 时间 $O(n)$, 空间 $O(n)$ 。

手推: 每轮入栈、弹栈与贡献

对 $a = (3, 1, 2)$, 栈从底到顶书写:

r	M_r	本轮操作	p_r / 贡献
1	2	入 1, 弹 1, 栈空	0 / 1
2	2	入 2, 保留栈 (2)	2 / 0
3	0	入 3, 依次弹 3, 2, 栈空	0 / 3

答案为 $1 + 0 + 3 = 4$; p_r 先右跳, 再左移。

$r = n$ 时任意 $l \leq n$ 都合法。答案最多为 $n(n+1)/2$, 用 64 位整数保存。

T2 字符串

有 n 条记录,按原顺序各分配一个长度为 l_i 的二进制串。

删除恰好一条记录,要求剩余串字典序严格递增,且两两不存在前缀关系。
求所有可删除位置。

$$2 \leq n \leq 2 \times 10^5, 1 \leq l_i \leq 60。$$

样例: $l = (2, 1, 2, 2)$, 可删除位置为 1, 2。

前缀编码与二叉树

要求任意一个串都不是另一个串的前缀,这就是前缀编码的限制。

从根出发,读到 0 向左,读到 1 向右。每个串对应一个节点,串长就是节点深度。

- ▶ 前缀关系:一个节点是另一个节点的祖先。
- ▶ 无前缀关系:选中的节点互不为祖先。
- ▶ 字典序递增:这些节点按从左到右的顺序排列。

本题给定各节点的深度,删除一个后,判断能否按原顺序选出这些节点。

节点对应一段叶子区间

以深度 2 为例, 底层叶子依次为 00, 01, 10, 11, 编号为 0, 1, 2, 3。

串	下面的叶子	对应区间
0	00, 01	$[0, 2)$
01	01	$[1, 2)$
1	10, 11	$[2, 4)$

一个节点下面的叶子连续, 祖先关系对应区间包含。

互不为祖先的节点, 对应区间互不相交。

因此, 要按原顺序从左到右放置互不相交的区间。

区间长度与起点对齐

将二叉树补到深度 60, 底层叶子编号为 $0 \sim U - 1, U = 2^{60}$ 。

长为 l 、二进制值为 b 的串, 对应区间

$$[b2^{60-l}, (b+1)2^{60-l}).$$

第 i 个区间长 $w_i = 2^{60-l_i}$, 起点必须是 w_i 的整数倍。

按给定顺序放置这些对齐区间, 且互不相交, 就得到所需编码。

总长度够,不代表能按顺序放下

用深度 2 的二叉树演示,叶区间为 $[0, 4)$ 。

记录长度为 $(2, 1, 2)$, 对应块长 $(1, 2, 1)$, 总长恰好为 4。

第一块最早放在 $[0, 1)$; 第二块必须对齐, 只能放在 $[2, 4)$ 。

第三块已无位置。

不能忽略原顺序, 也不能忽略起点对齐。

朴素贪心: 枚举删除, 再靠左放

枚举删除位置 k , 从左到右放置其余区间。每块都放在最靠左的合法位置。前面结束于 u , 当前块长 w_i , 起点为 $w_i \lceil u/w_i \rceil$, 所以

$$u \leftarrow w_i \left(\left\lceil \frac{u}{w_i} \right\rceil + 1 \right), \quad u \text{ 初始为 } 0.$$

前面结束得越早, 后面可选位置越多, 因此该贪心不会损害可行性。若 $u > U$ 则失败; 全部放下即成功。一次 $O(n)$, 枚举删除共 $O(n^2)$ 。

前缀预处理: 复用重复的贪心

删除位置不同时, 相同的前缀不必反复放置, 只需预处理一次。

设 F_i 为前 i 块最靠左放置时的结束位置, $F_0 = 0$ 。

第 i 块最早起点为 $w_i \lceil F_{i-1}/w_i \rceil$, 因此

$$F_i = w_i \left(\left\lceil \frac{F_{i-1}}{w_i} \right\rceil + 1 \right).$$

从左到右扫描一遍, $O(n)$ 求出所有 F_i 。

后缀预处理: 从右侧靠右放

对称地, 从右到左, 把每块放在最靠右的合法位置。

设 G_i 为第 $i \sim n$ 块最靠右放置时的开始位置, $G_{n+1} = U$ 。

第 i 块的结束位置至多为 G_{i+1} , 且必须对齐到 w_i 的整数倍, 因此

$$G_i = w_i \left(\left\lfloor \frac{G_{i+1}}{w_i} \right\rfloor - 1 \right).$$

从右到左扫描一遍, $O(n)$ 求出所有 G_i 。

删除一个位置, 只检查两端边界

删除第 k 块, 左侧最早结束在 F_{k-1} , 右侧最晚开始在 G_{k+1} 。

可行当且仅当

$$F_{k-1} \leq G_{k+1}.$$

若成立, 两边互不相交, 中间允许留空, 直接拼接即可。

若不成立, 两端已分别尽量靠左、靠右, 仍然重叠, 其他放法也不可行。

两次预处理共 $O(n)$, 每个删除位置 $O(1)$ 判定, 总时间 $O(n)$ 。

样例与失败状态

仍用深度 2 演示 $l = (2, 1, 2, 2)$:

删除位置 k	F_{k-1}	G_{k+1}	可行?
1	0	0	是
2	1	2	是
3	4	3	否
4	超出 4	4	否

正式实现用 $U = 2^{60}$;

前缀失败记为 $U + 1$, 后缀失败记为 -1 , 后续保持失败状态。

使用 64 位整数及整数取整运算; 时间、空间均为 $O(n)$ 。

T3 凸序列

原序列 A 非负、总和为 M , 且

$$A_{i-1} + A_{i+1} \geq 2A_i.$$

将每项截为 $B_i = \min(A_i, H)$ 。求不同 B 的数量, 对 998244353 取模。

$1 \leq N \leq 40, 0 \leq M \leq 1000, 1 \leq H \leq 50$ 。

样例: $(N, M, H) = (3, 6, 2)$, 答案为 15。

先数结果, 不数原像

$(2, 0, 4)$ 、 $(3, 0, 3)$ 、 $(4, 0, 2)$ 都会得到 $(2, 0, 2)$, 只能计一次。

也不能要求 B 仍然凸:

$$(6, 3, 0) \longrightarrow (4, 3, 0) \quad (H = 4).$$

原序列满足凸性, 截平后却有 $4 + 0 < 2 \times 3$ 。

先描述 B 的形状, 再判断是否存在总和恰好为 M 的凸原序列。

凸性决定结果的形状

原序列的凸性等价于差分非降：

$$A_{i-1} + A_{i+1} \geq 2A_i \iff A_{i+1} - A_i \geq A_i - A_{i-1}.$$

若两端都小于 H ，中间项也小于 H ，所以低值位置连续。

除全为 H 外，结果唯一写成

$$H^l \mid b_1, \dots, b_c \mid H^r, \quad l + c + r = N.$$

H^l 表示连续 l 个 H 。中间各项小于 H ，仍然凸；两侧可以为空。

先枚举低值段的位置

低值段从原数组位置 p 开始, 在位置 t 结束, 则

$$l = p - 1, \quad c = t - p + 1, \quad r = N - t.$$

例如 $N = 5, p = 3, t = 4$, 结果为

$$B = (H, H, b_1, b_2, H).$$

左侧有几项由起点确定, 不需要从低值段的数值推断。

先枚举位置与内容, 再判断这个结果是否存在合法原像。

左侧的最小等差延拓

先考虑低值段至少两项。左侧若非空,从 b_1 向左的第一步增量满足

$$q_L \geq H - b_1, \quad q_L \geq b_1 - b_2.$$

向外的增量不能减小,所以每步都取

$$q_L = \max(H - b_1, b_1 - b_2)$$

时,总和最小。左侧最小代价为

$$C_L = lb_1 + \frac{l(l+1)}{2}q_L.$$

例如 $H = 4$ 、低值段为 $(3, 0)$, 左邻至少取 6, 不能只取 4。

右侧延拓与统一代价

右侧同理：

$$q_R = \max(H - b_c, b_c - b_{c-1}),$$

$$C_R = rb_c + \frac{r(r+1)}{2}q_R.$$

记边界低值为 x 、内侧相邻低值为 y ，向外补 k 项的最小代价为

$$E(k, x, y) = kx + \frac{k(k+1)}{2} \max(H - x, x - y).$$

于是 $C_L = E(l, b_1, b_2), C_R = E(r, b_c, b_{c-1}); k = 0$ 时代价为 0。

最小和为什么足够？

固定一个结果，原序列最小总和为

$$S = \sum_{i=1}^c b_i + C_L + C_R.$$

若 $l + r > 0$ ，可行当且仅当 $S \leq M$ 。

取最小延拓，把差额 $M - S$ 加到某个截平侧的最外端点，只会放宽端点附近的凸性条件，且不改变结果。

若 $l = r = 0$ ，没有位置可补差额，必须满足 $\sum b_i = M$ 。

单独处理两种边界

没有低值: 结果全为 H , 在 $M \geq NH$ 时贡献 1。

只有一个低值 $b < H$, 位于 p , 令 $l = p - 1, r = N - p$ 。

两侧最小增量均为 $H - b$, 最小总和为

$$S = Nb + (H - b) \left(\frac{l(l+1)}{2} + \frac{r(r+1)}{2} \right).$$

$N > 1$ 时判断 $S \leq M$; $N = 1$ 时必须 $b = M$ 。

朴素 DP: 保留低值段起点

顺序: 当前结束位置 t 从左到右。

状态 $F_{p,t}(a, b, s)$: 低值段长度至少为 2 的结果前缀数。

- ▶ p : 低值段的起点。
- ▶ a, b : 末两项, $0 \leq a, b < H$ 。
- ▶ s : 左侧最小代价加当前低值段之和, $0 \leq s \leq M$ 。

这里 s 是已用背包容量; 先保留起点, 完整写出转移与结算。

朴素初始化:枚举起点和首两项

枚举起点 $1 \leq p < N$ 、首两项 $a, b < H$ 。

左侧长度为 $p - 1$, 初始已用代价为

$$s = a + b + E(p - 1, a, b).$$

若 $s \leq M$, 执行

$$F_{p,p+1}(a, b, s) += 1.$$

初始结束位置是 $p + 1$; 左侧代价已经一次计入 s 。

朴素转移: 追加一个低值

当前状态为 $F_{p,t}(a, b, s)$, 追加 $0 \leq x < H$ 。

末尾三项需要满足凸性, 容量不能超过 M :

$$a + x \geq 2b, \quad s + x \leq M.$$

满足时执行

$$F_{p,t+1}(b, x, s + x) += F_{p,t}(a, b, s).$$

起点 p 不变, 末两项变为 (b, x) , 已用容量增加 x 。

朴素结算: 在当前位置结束

状态 $F_{p,t}(a, b, s)$ 在 t 结束, 右侧还有 $r = N - t$ 项。

$$C_R = E(r, b, a) = rb + \frac{r(r+1)}{2} \max(H - b, b - a).$$

将状态计数加入答案的条件:

- ▶ $p > 1$ 或 $r > 0$: 有截平位置, 要求 $s + C_R \leq M$ 。
- ▶ $p = 1$ 且 $r = 0$: 无截平位置, 要求 $s = M$ 。

起点、低值序列、终点共同唯一确定结果, 不计原像的种数。

状态优化: 起点还影响什么?

观察朴素初始化、转移和结算:

- ▶ 左侧最小延拓代价, 已经计入容量 s 。
- ▶ 追加下一项, 只依赖末两项 a, b 和容量 s 。
- ▶ 右侧长度由当前原数组位置 t 决定。
- ▶ 结束时, 起点只影响左侧是否有截平位置, 即 $p > 1$ 。

因此保留容量 s 和标记 $z = [p > 1]$, 可以消去完整起点这一维。

按后效性合并起点状态

将未来选择、结算条件完全相同的状态合并：

$$f_t(a, b, s, z) = \sum_{\substack{1 \leq p < t \\ z = [p > 1]}} F_{p,t}(a, b, s).$$

新状态保留：当前结束位置 t 、末两项 a, b 、已用容量 s 、左侧截平标记 z 。

合并状态时累加方案数，不是把不同的结果只算成一种。

$z = 0$ 对应 $p = 1$ ； $z = 1$ 合并所有 $p > 1$ 的起点。

代回初始化与转移

每层 $t \geq 2$ 新开长度为 2 的低值段, 此时 $p = t - 1$:

$$s = a + b + E(t - 2, a, b), \quad z = [t > 2].$$

若 $s \leq M$, 执行 $f_t(a, b, s, z) += 1$ 。

追加 $0 \leq x < H$, 且 $a + x \geq 2b, s + x \leq M$ 时,

$$f_{t+1}(b, x, s + x, z) += f_t(a, b, s, z).$$

这里 t 始终是原数组位置; $t - 2$ 只用于新开状态的左侧长度。

代回结束条件

在 t 结束, 令 $r = N - t$, 补上右侧代价 $E(r, b, a)$ 。

满足下列条件时, 执行 $\text{ans} += f_t(a, b, s, z)$:

- ▶ $z = 1$ 或 $r > 0: s + E(r, b, a) \leq M$ 。
- ▶ $z = 0$ 且 $r = 0: s = M$ 。

起点的精确信息已不参与后续计算, 但左侧是否可补差额仍由 z 保留。

所有方案数运算对 998244353 取模。

转移优化: 先写来源求和

固定目标末两项 b, c 、旧容量 s 和标记 z :

$$f_{t+1}(b, c, s + c, z) += \sum_{a=0}^{H-1} [a + c \geq 2b] f_t(a, b, s, z).$$

由 $a + c \geq 2b$, 整理出来源范围:

$$f_{t+1}(b, c, s + c, z) += \sum_{a=\max(0, 2b-c)}^{H-1} f_t(a, b, s, z).$$

来源是 a 这一维的一个后缀。这里仍要求 $s + c \leq M$ 。

后缀和定义后, 代回转移

沿 a 这一维预处理, 约定 $\text{Suf}_t(H, b, s, z) = 0$:

$$\text{Suf}_t(u, b, s, z) = \sum_{a=u}^{H-1} f_t(a, b, s, z).$$

因此, 原转移完整改写为

$$f_{t+1}(b, c, s + c, z) += \text{Suf}_t(\max(0, 2b - c), b, s, z).$$

下界不小于 H 时贡献为 0; 否则一次查询替代枚举 a 。

适用条件: $0 \leq b, c < H, s + c \leq M$ 。

两次优化的复杂度

阶段	时间	优化点
保留起点	$O(N^2 H^3 (M + 1))$	先按起点分别生成低值段
消去起点	$O(N H^3 (M + 1))$	容量加存在性标记合并状态
后缀和	$O(N H^2 (M + 1))$	批量求合法来源的计数和

优化后滚动空间 $O(H^2 (M + 1))$, 用独立数组保存后缀和。

每层依次初始化、结算、求后缀和、转移; 原地求后缀和时, 必须先结算当前层。

代价乘法使用 64 位整数, 计数及时取模。

两条检查路线

$H > M$: 所有原项都小于 H , 没有截平; 问题退化为原凸序列的精确总和计数。

$N = 1$: 无论 M, H 如何取, 结果只有一个, 即 $\min(M, H)$ 。

小规模检查: 枚举总和为 M 的原序列, 筛凸性, 将截平结果放进集合去重。

样例 $(3, 6, 2)$: 凸原序列 18 个, 不同截平结果 15 个。

T4 gcd 完备

序列元素两两不同。

若区间内任意两数的 gcd 都在该区间出现,则称该区间完整。

每次询问 $[l, r]$, 求其中完整子段的数量。

$n, q \leq 2 \times 10^5, a_i \leq 4 \times 10^5$ 。

单点总是完整段;答案用 64 位整数保存。

完整性不是普通的区间单调性

$[2, 3]$ 不完整, 加入 1 后, $[2, 3, 1]$ 又完整。

不能因为当前不完整, 就断言继续扩展也不完整。

小规模可以逐段检查“两两 gcd 是否出现”, 再统计询问内的完整段。

整除链性质: 任意两数有一个整除另一个, 则所有子段都完整, 询问长度为 m 时答案为 $m(m+1)/2$ 。

量化合法性: 缺失的 gcd

区间对两两 gcd 封闭, 等价于任意非空子集的整体 gcd 都在区间出现。

固定 d , 取区间内所有 d 倍数组成集合 S_d 。

区间不完整, 当且仅当存在 d 同时满足

$$S_d \neq \emptyset, \quad \gcd(S_d) = d, \quad d \text{ 未出现}.$$

若某个子集的 gcd 已经是 d , 加入其余 d 的倍数, gcd 仍为 d , 所以不用枚举子集。

集合 gcd 与两两 gcd

若区间对两两 gcd 封闭, 反复合并就得到: 任意非空子集的整体 gcd 都会出现。

反过来, 若一对数的 gcd 为缺失值 d ,
所有 d 倍数的整体 gcd 既是 d 的倍数、又整除 d , 只能为 d 。

因此按缺失的 d 判定是等价转化。

不能说“整体 gcd 一定等于某一对数的 gcd”:
(6, 10, 15) 整体 gcd 为 1, 两两 gcd 却是 2, 3, 5。

朴素询问:枚举子段的两端

扫描当前右端点 R , 记每个左端点的合法性为

$$g_R(L) = [[L, R] \text{ 完整}].$$

直接计算询问, 要枚举其中所有子段:

$$\text{ans}(l, r) = \sum_{E=l}^r \sum_{L=l}^E g_E(L).$$

同一个左端点会在多个右端点下贡献, 考虑把这些历史贡献累计起来。

固定左端点, 累计历史贡献

定义截至 R 、以 L 为左端点的完整段数量:

$$h_R(L) = \sum_{E=L}^R g_E(L) = h_{R-1}(L) + g_R(L).$$

交换原求和顺序并代入, 得到

$$\text{ans}(l, r) = \sum_{L=l}^r \sum_{E=L}^r g_E(L) = \boxed{\sum_{L=l}^r h_r(L)}.$$

当前合法性 g 沿右端点累计成 h , 再沿左端点求区间和。

询问按右端点离线, 不需要保存全部历史版本。

固定右端点: 每个 d 只有一个阈值

扫描右端点 R 。

设 T_d 是满足“ $[L, R]$ 内所有 d 的倍数的 gcd 为 d ”的最大 L ; 无解取 0。

若 L 满足条件, 向左加入更多倍数, gcd 仍是 d 。

因此满足条件的左端点恰为

$$L \in [1, T_d].$$

随 R 增大, 已有的这些左端点不会失效, 所以 T_d 单调不减。

坏区间覆盖左端点

记 p_d 为 d 在已扫描部分的出现位置;尚未出现取 0。

gcd 条件要求 $L \leq T_d$, 缺少 d 要求 $L > p_d$, 取交得到

$$I_d(R) = [p_d + 1, T_d].$$

设 $\text{cover}_R(L)$ 为所有坏区间对 L 的覆盖次数, 则

$$g_R(L) = [\text{cover}_R(L) = 0].$$

空区间不操作;覆盖为 0, 表示没有任何 d 违反完整性条件。

样例: 值出现时要撤销限制

序列为 $(6, 2, 3, 1)$, 关注 $d = 1$ 。

$R = 3$: 最短 gcd 为 1 的后缀是 $(2, 3)$, 故 $T_1 = 2$ 。

值 1 还没出现, 坏左端点是 $[1, 2]$ 。

$R = 4$: 值 1 出现, 旧坏区间撤销。此时 $T_1 = 4$, 新坏区间 $[5, 4]$ 为空。

四个右端点新增的完整段数依次为 1, 2, 1, 4, 询问 $[1, 4]$ 的答案为 8。

只有新增值的约数会变化

新增 $x = a_R$, 只更新 $d \mid x$ 。

- ▶ 若 $x = d$: 撤销旧坏区间 $[1, T_d^{\text{old}}]$, 再令 $p_d = R$, 此刻不新增覆盖。
- ▶ 若 $x \neq d$: p_d 不变, T_d 只会右移, 新增一次覆盖的区间为

$$[\max(p_d + 1, T_d^{\text{old}} + 1), T_d^{\text{new}}].$$

空区间不操作。值 d 出现后, 未来若 $T_d > p_d$, 仍可能重新产生坏区间。

倍数子序列上的双指针

对固定 d , 只看它的倍数, 保留原下标, 并把数值除以 d 。

T_d 就是 gcd 为 1 的最短非空后缀的起点原下标。

- ▶ 右指针: 新增倍数进入队尾。
- ▶ 若整体 gcd 不为 1, 当前 $T_d = 0$ 。
- ▶ 否则, 只要删队首后剩余非空窗口的 gcd 仍为 1, 左指针就继续右移。

删掉的位置不必恢复: 保留窗口的 gcd 已为 1, 以后加入新数仍为 1。

子序列总长度的调和级数上界

设 $A = \max a_i$ 。元素互异, 所以所有倍数子序列的总长度满足

$$\sum_i \tau(a_i) = \sum_{d=1}^A \#\{i : d \mid a_i\} \leq \sum_{d=1}^A \left\lfloor \frac{A}{d} \right\rfloor = O(A \log(A+1)).$$

这里限制的是总长度和约数更新次数, 不是子序列条数, 也不是整题总时间。

预处理最小质因子; 扫描到 $a_R = x$ 时生成约数, 把 $(R, x/d)$ 加入对应队列即可。

不必预先建出所有完整倍数子序列。

双栈维护滑动窗口 gcd

入队栈与出队栈, 每层保存从栈底到该层的 gcd。出队栈为空时倒栈, 出队栈顶就是队首。

整个窗口的 gcd 为

$$g_{\text{window}} = \text{gcd}(g_{\text{in}}, g_{\text{out}}).$$

排除出队栈顶后, 其余部分的 gcd 记为 g_{out}^- , 则

$$g_{\text{remove front}} = \text{gcd}(g_{\text{in}}, g_{\text{out}}^-).$$

空栈 gcd 取 0。每条记录至多入队、换栈、出队各一次, 均摊常数 gcd 运算。

历史数组的朴素维护

已有当前覆盖数后, 每轮枚举所有已激活的左端点, 执行

$$h_R(L) = h_{R-1}(L) + [\text{cover}_R(L) = 0].$$

询问 $[l, r]$ 在扫描到 r 时, 求 $h_r(l) + \cdots + h_r(r)$ 。

即使坏区间更新已经很快, 逐个位置累计历史仍需 $O(n^2)$ 时间。

接下来用线段树, 批量识别零覆盖位置并累加历史贡献。

历史贡献也可以按持续时间理解

一个左端点从扫描时刻 s 开始零覆盖, 在时刻 t 离开零覆盖, 这一段贡献为

$$t - s.$$

若查询时刻 R 仍为零, 尚未结算的贡献为 $R - s + 1$ 。

同一 R 内所有修改共用一个时刻, 中间的临时零覆盖状态不额外贡献。

这与每轮给零覆盖位置加 1 等价; 下面用逐轮累加的标签实现。

线段树维护的信息

- ▶ mn : 结点区间的覆盖次数最小值。
- ▶ cnt : 达到最小值的位置数。
- ▶ sum : 区间内全部位置的历史计数之和。
- ▶ add : 覆盖次数的整段加法标签。
- ▶ $hist$: 待补给最小值位置的历史贡献标签。

sum 保存全部历史, 不只保存当前最小值位置的历史。

用最小值位置累计一次贡献

每轮覆盖更新结束后,已激活位置的覆盖数非负。

单点 $[R, R]$ 完整,因此根的 $mn = 0$ 。

根的最小值位置,恰好就是所有合法左端点。执行

$$sum_{\text{root}} += cnt_{\text{root}}, \quad hist_{\text{root}} += 1.$$

就完成了全部零覆盖位置的历史计数加一,不必逐个访问叶子。

未激活位置设为足够大的覆盖数,不参与最小值计数。

历史标签怎样下传

部分修改前先下传,避免旧贡献跟随新的最小值位置。

1. 先下传覆盖加法 add ,统一父子最小值的基准。
2. 只给 $mn_v = mn_u$ 的孩子 v 下传父结点 u 的历史标签。
3. 清空父结点标签,再递归修改、合并信息。

步骤2完整写为

$$sum_v += hist_u cnt_v, \quad hist_v += hist_u.$$

整段统一覆盖加减不改变最小值位置集合,可以暂存标签。

过去的贡献不能跟随新最小值

某个内部结点有两个位置,覆盖数与历史计数分别为

$$\text{cover} = (0, 1), \quad h = (0, 0).$$

累计一次贡献后, $h = (1, 0)$ 。

随后第一个位置覆盖加 2, 应得到

$$\text{cover} = (2, 1), \quad h = (1, 0).$$

当前最小值换到了第二个位置,但过去那次贡献仍属于第一个位置。

扫描线的处理顺序

对每个右端点 R :

1. 激活左端点 $L = R$, 令覆盖数和历史贡献为 0。
2. 枚举 a_R 的约数, 更新窗口与坏区间覆盖。
3. 给所有覆盖数为 0 的位置增加一次历史贡献。
4. 回答右端点为 R 的询问, 查询 $[l, R]$ 的历史和。

未激活位置的覆盖数设为足够大的数, 不参与历史累计。

复杂度: 区分长度和值域

设 $A = \max a_i$ 。元素互异, 所以约数更新总次数不超过

$$\sum_{d=1}^A \left\lfloor \frac{A}{d} \right\rfloor = O(A \log(A+1)).$$

最小质因子筛预处理 $O(A)$; 按质因数分解生成每个输入值的约数。

计入 gcd 运算与线段树后, 总时间为

$$O(A \log^2(A+1) + (n+q) \log(n+1)).$$

总空间为 $O(A \log(A+1) + n + q)$ 。

不能在没有任何额外条件时把 A 直接换成 n 。

四个可以带走的做法

排队: 合法性化为极值关系, 固定右端点后找最后一个障碍。

字符串: 前缀关系转成区间包含; 前后缀各取极值, 判断拼接。

凸序列: 先确定结果的唯一形状, 再用最小原像判断可行, 最后计数。

gcd 完备: 将坏情况拆成覆盖区间; 区分当前合法数量与完整历史贡献。

保持联系

信星社官方 QQ 群: 1091707974

Polaris 官方 QQ 群: 587390359

信星社 OJ: oj.xinxingshe.cn

Polaris: polaris.xinxingshe.cn

