2026 华南理工大学程序设计竞赛题解

2026 华南理工大学程序设计竞赛出题组

South China University of Technology

2026 年 3 月 29 日

赛题分析

预期难度：

Easy:	BDIL
Easy-Medium:	HKE
Medium:	GJC
Medium-Hard:	AF

实际过题人数情况：

A	B	C	D	E	F	G	H	I	J	K	L
0	61	0	50	6	0	3	1	38	3	29	42

Basic Matrix Recurrence Practice

问题描述

给定二阶向量递推 $v_{n+1} \equiv Pv_n + Q(v_{n-1} \odot v_{n-1}) \pmod{61}$，其中所有运算均在模 61 意义下进行。
已知初始状态 v_0, v_1，需要在线回答 q 次询问。每次给出下标 $n_i \le 10^{18}$，要求输出 v_{n_i} 的压缩哈希值。

核心约束

- 询问是强制在线的：n_i 依赖前一次答案 last_ans 与伪随机数生成器的结果；
- 内存限制仅有 **8 MB**。

Basic Matrix Recurrence Practice

有限状态下的周期性

递推式中，下一项 v_{n+1} 只由 $S_n = (v_n, v_{n-1})$ 唯一决定，因此整个序列可以视为有限状态空间上的确定性转移。

设每个向量有 4 个分量，则单个向量共有 61^4 种取值，因此状态总数为 $|S| = (61^4)^2 = 61^8 \approx 1.91 \times 10^{14}$。

状态空间有限而转移唯一，根据鸽巢原理，序列必然在有限步后进入循环。也就是说，存在前导长度 μ 与循环节长度 λ，使得当 $n \geq \mu$ 时恒有 $S_n = S_{n+\lambda}$。

Basic Matrix Recurrence Practice

随机映射模型下的规模估计

若循环长度达到状态总数的量级，则预处理显然不可行。但本题还有一个关键条件：矩阵 P, Q 的元素均在 $[0, 60]$ 中独立均匀随机生成。
这使得状态转移函数在宏观上接近随机映射。根据随机映射函数图的经典结论，其轨迹通常呈现“前导链 + 环”的 ρ 形结构，并满足 $\mathbb{E}[\mu + \lambda] \sim \sqrt{\frac{\pi |S|}{2}}$。
代入本题规模，可估计 $\mathbb{E}[\mu + \lambda] \approx \sqrt{1.91 \times 10^{14}} \approx 1.38 \times 10^{7}$。
这一数量级虽然不小，但仍处于可线性迭代的范围内。因此本题的主要困难并非时间，而是空间限制。

Basic Matrix Recurrence Practice

显式存储轨迹不可行

即使可以在线性时间内遍历完整个前导链与循环节，若将这段长度约为 10^7 的状态轨迹全部存储，所需内存也会显著超过题目的 8 MB 限制。
因而判环阶段不能依赖哈希表或整段数组，而必须采用额外空间为 $O(1)$ 的方法。

Brent 判环

这里采用 Brent 算法：

- 先利用指数扩张方式寻找第一次重复状态，从而得到循环节长度 λ；
- 再令两个指针相距 λ 同步前进，即可求出入环位置 μ。

整个过程只需维护常数个状态变量，不需要长期保存完整轨迹。

Basic Matrix Recurrence Practice

在线询问的处理

当 μ, λ 已知后，对任意 $n_i \geq \mu$，都可以先映射到等价位置 $\tilde{n}_i = \mu + (n_i - \mu) \bmod \lambda$。
然而，由于无法保存整条序列，而询问又是在线生成的，因此不能在每次询问时重新从头递推至 $\tilde{n}_i$。

分块保存检查点

设有效轨迹长度为 $L = \mu + \lambda$。假设在内存限制下最多可保存 M 个状态，则取步长 $B = \lceil \frac{L}{M} \rceil$。
再次遍历轨迹时，仅在下标为 $0, B, 2B, \ldots$ 的位置保存检查点状态。回答某个 $\tilde{n}_i$ 时，先定位到其左侧最近的检查点，再向前递推至多 $B - 1$ 步即可得到答案。

Basic Matrix Recurrence Practice

复杂度分析

- **预处理**：Brent 判环及后续重走轨迹都线性依赖于 $L = \mu + \lambda$，因此期望复杂度为 $O(\sqrt{|S|})$。
- **单次询问**：定位检查点为 $O(1)$，之后暴力递推最多 B 步，因此为 $O(B)$。
- **总询问复杂度**：$O(qB) = O\left(q \cdot \frac{L}{M}\right)$。

仅保存约 M 个检查点状态及常数个辅助变量，因此空间复杂度为 $O(M)$，其中 M 是由题目内存上限决定的常数。

The Light Boat Has Passed Ten Thousand Mountains

问题描述

给定长度为 n 的数组 a，求最小的正整数 $x \in [1, 10^9]$，使得数组中恰好有 k 个元素小于等于 x。若存在多个可行解，输出其中最小者；若不存在可行解，输出 -1。

The Light Boat Has Passed Ten Thousand Mountains

排序后的刻画

将数组升序排序，记为 $a_0 \le a_1 \le \cdots \le a_{n-1}$。排序后，若希望恰好有 k 个元素小于等于 x，则被计入的元素必然对应前 k 个数。

可行解条件

当 $0 < k < n$ 时，x 必须满足 $a_{k-1} \le x < a_k$。前一条件保证前 k 个元素全部被计入，后一条件保证第 $k+1$ 个元素不会被额外计入。
由于题目要求输出最小可行解，只要该区间非空，答案即为其左端点，即 $x = a_{k-1}$。

The Light Boat Has Passed Ten Thousand Mountains

边界情况

- 当 $k = 0$ 时，要求不存在元素小于等于 x。由于 x 必须为正整数，最小候选值为 $x = 1$。若此时 $a_0 = 1$，则已经有元素被计入，无解；否则答案为 1。
- 当 $k = n$ 时，要求全部元素均小于等于 x，因此最小可行值为 $x = a_{n-1}$。
- 当 $0 < k < n$ 时，需满足 $a_{k-1} \le x < a_k$。若 $a_{k-1} = a_k$，则可行区间为空，无解；否则答案为 $x = a_{k-1}$。

The Light Boat Has Passed Ten Thousand Mountains

复杂度分析

- 时间复杂度：排序为 $O(n \log n)$，其余判断为 $O(1)$，总复杂度为 $O(\sum n \log n)$。
- 空间复杂度：$O(n)$。

Leyline Resonance

问题描述

给定一棵以 1 为根的树，每个节点 i 有初始权值 a_i。需要为每个节点赋一个实数 x_i，使总代价 $\mathrm{Cost} = \sum_{i=1}^{n} |x_i - a_i| + \sum_{i=2}^{n} w_i \cdot \max(0, x_{p_i} - x_i)$ 最小，其中 p_i 为 i 的父节点，w_i 为边 (p_i, i) 的惩罚权重。

Leyline Resonance

树形动态规划

设 $F_i(x)$ 表示在节点 i 的整棵子树中，强制令 $x_i = x$ 时的最小代价。
若 c 是 i 的儿子，则当父节点取值为 x 时，儿子子树的最优贡献可写为 $G_c(x) = \min_{y\in\mathbb{R}} \{F_c(y) + w_c \max(0, x - y)\}$。
因而有转移 $F_i(x) = |x - a_i| + \sum_{c\in \mathrm{child}(i)} G_c(x)$。

函数的结构性质

绝对值函数是凸的，而上述“加函数再取最小”的操作同样保持凸性。因此对任意节点，$F_i(x)$ 与 $G_i(x)$ 都是分段线性凸函数，仅在有限个横坐标处发生斜率变化。

Leyline Resonance

$G_c(x)$ 的导数截断性质

考察 $G_c(x) = \min_y\{F_c(y) + w_c \max(0, x - y)\}$。
其几何意义如下：

- 当 x 不超过 F_c 的最优位置时，可以取一个 $y \geq x$ 使惩罚项为 0，此时 $G_c(x)$ 保持常值；
- 当 x 继续向右移动时，新增惩罚项的斜率至多为 w_c，因此 $G_c(x)$ 的导数不会超过 w_c。

因而从函数形态上看，这一步等价于对 F_c 的右侧斜率实施上界截断。

最左侧斜率恒为 -1

由 $F_i(x) = |x - a_i| + \sum G_c(x)$ 可知，当 $x \to -\infty$ 时，每个 $G_c(x)$ 的斜率都趋于 0，而 $|x - a_i|$ 的斜率恒为 -1。
因而对任意节点 i，函数 $F_i(x)$ 在最左侧区间的斜率始终为 -1。这一结论极为关键：只要维护斜率发生增加的位置，就足以确定函数的最优点。

Leyline Resonance

以多重集合维护斜率变化点

既然 $F_i(x)$ 是分段线性凸函数，则无需维护其完整解析式。更自然的做法是仅维护其不可导点，也即“斜率增加”的位置。
设多重集合 S_i 中的每个元素 v 表示函数在 $x = v$ 处发生一次斜率 $+1$ 的增加。

集合上的对应操作

- **加入 $|x - a_i|$**：该函数在 a_i 处总共带来 $+2$ 的斜率增量，因此向集合中插入两个 a_i。
- **确定当前最优点**：由于最左侧斜率固定为 -1，斜率第一次上升到 0 的位置即为全局最小值点，它恰好对应当前集合中的最小元素。
- **由 F_i 转化为 G_i**：这一步等价于将最左侧那段斜率 -1 抹平成 0，因此只需删除集合中的最小元素一次。

Leyline Resonance

最小值增量的直接计算

设在合并完儿子信息后，当前集合最小值为 $\min(S_i)$。再加入两个 a_i 后，新函数最小值相较原函数增加的代价恰为 $\max(0, a_i - \min(S_i))$。
因而答案无需在最后统一回溯计算，而可以在 DFS 转移过程中直接累加。

右侧斜率的截断实现

对非根节点 i，由前述分析可知，函数 $G_i(x)$ 的最右侧斜率不能超过 w_i。
在多重集合模型中，集合大小正好对应函数最右侧的总斜率。因此只需不断删除集合中的最大元素，直到 $|S_i| \leq w_i$，即完成导数上界的截断。

Leyline Resonance

整棵树上的计算流程

采用后序遍历，自底向上完成转移：

1. 先递归处理所有儿子；
2. 将所有儿子的多重集合合并到当前节点；
3. 插入两个 a_i；
4. 取出并删除当前集合最小值，完成从 F_i 到 G_i 的转换，并同时累加答案；
5. 若当前节点不是根，再不断删除集合最大值，直到集合大小不超过 w_i。

启发式合并的必要性

若直接将每个儿子集合逐个元素插入当前集合，则在极端树形结构下会退化为平方复杂度。
因此必须使用启发式合并：始终将较小的集合并入较大的集合。这样每个元素被搬运的次数至多为对数级别。

Leyline Resonance

复杂度分析

- 在启发式合并下，每个元素至多被搬运 $O(\log n)$ 次；
- 每次插入、删除、取最值在 multiset 中均为 $O(\log n)$。

因而总时间复杂度为 $O(n \log^2 n)$。
所有集合中的元素总数始终保持在线性级别，因此空间复杂度为 $O(n)$。

SCUT Classroom Relocation

问题描述

有 m 间教室与 n 个学生群体。

- 教室 j 的剩余容量为 C_j；
- 群体 i 的人数为 Q_i，原所在教室为 P_i。

现需重新安置所有学生，并要求：来自群体 i 的任意学生若被分配到教室 j，都必须满足 $j \neq P_i$。

问题本质

群体可按人数任意拆分，因此本题等价于判定：是否存在一种合法分配，使全部学生均被安置，且没有任何教室超出容量。

SCUT Classroom Relocation

按来源教室聚合

约束仅与“学生原本来自哪间教室”有关，而与其所属的具体群体无关。因此首先按来源教室聚合，记 $W_k = \sum_{i:P_i=k} Q_i$，表示原本位于教室 k、现需迁出的学生总数。

二分图匹配转化

构造二分图：

- 左部表示所有待安置学生，按来源教室划分为若干类 U_k，其中 $|U_k| = W_k$；
- 右部表示所有空座位，按教室划分为若干类 V_j，其中 $|V_j| = C_j$；
- 若学生来自教室 k，则其可以连向除 V_k 外的所有座位。

问题等价于：该二分图中是否存在一个匹配，覆盖左部全部学生。

SCUT Classroom Relocation

霍尔定理

对于二分图，存在覆盖左部全部点的匹配，当且仅当对任意左部子集 A，其邻域 $N(A)$ 满足 $|A| \leq |N(A)|$。

对子集的化简

左部总人数可能极大，显然无法直接枚举所有子集。
但同一来源教室中的学生具有完全相同的邻域，因此霍尔条件中真正需要讨论的并非“选中了哪些学生”，而是“这些学生来自哪些教室”。

SCUT Classroom Relocation

设某个子集 A 中的学生来自教室集合 $K \subseteq \{1, 2, \ldots, m\}$。分类讨论 $|K|$ 的大小。

情形一：$|K| = 1$

若 A 中学生全部来自同一间教室 k，则他们不能去教室 k，但可以去其余任意教室。因此 $N(A) = V \setminus V_k$，且 $|N(A)| = \sum_{j=1}^{m} C_j - C_k$。
为使霍尔条件最紧，应取教室 k 中全部待迁出的学生，即 $|A| = W_k$，从而得到 $W_k \le \sum_{j=1}^{m} C_j - C_k$。

SCUT Classroom Relocation

情形二：$|K| \geq 2$

若 A 中学生至少来自两间不同教室，例如 k_1, k_2，则：

- 来自 k_1 的学生可去除 k_1 外的所有教室；
- 来自 k_2 的学生可去除 k_2 外的所有教室。

由于 $k_1 \neq k_2$，两部分邻域并起来后已经覆盖全部座位，因此 $N(A) = V$，且 $|N(A)| = \sum_{j=1}^{m} C_j$。
为使条件最紧，应取全部学生，即 $\sum_{k=1}^{m} W_k \leq \sum_{j=1}^{m} C_j$。

充要条件的整理

上述两类情况已经覆盖霍尔条件中最强的限制，因此原问题有解，当且仅当同时满足 $\sum_{k=1}^{m} W_k \leq \sum_{j=1}^{m} C_j$，且对任意 k 都有 $W_k + C_k \leq \sum_{j=1}^{m} C_j$。

SCUT Classroom Relocation

复杂度分析

实现时只需统计所有 W_k 并检查上述条件。

- 时间复杂度：$\mathcal{O}(n+m)$；
- 空间复杂度：$\mathcal{O}(m)$。

Phantoms of the XOR Tree

问题描述

给定 n 个互不相同的非负整数 $A_1, A_2, \ldots, A_n$。
在这 n 个点上建立完全图，边权定义为 $w(i,j) = A_i \oplus A_j$。
记 G_{MST} 为这样一张图：若一条边至少出现在原图的某一棵最小生成树中，则将其加入 G_{MST}。题目要求求出该图的边数。

Phantoms of the XOR Tree

01-Trie 上的分治结构

将所有数插入 01-Trie。考虑某个同时拥有左右儿子的 Trie 节点，设其对应当前二进制位 k，左、右子树覆盖的数集分别为 L 与 R。
根据异或的性质：

- L 内部任意两数、R 内部任意两数的最高不同位都在 k 以下，因此边权严格小于 2^k；
- L 与 R 之间任意一条边在第 k 位必然不同，因此边权至少为 2^k。

Kruskal 视角下的解释

按 Kruskal 算法的处理顺序，所有更小的内部边一定先于跨越 L, R 的边被考察。因此，当开始处理连接 L 与 R 的边时，两边内部已经分别连通。
由此可知，在该分叉点处，只需讨论左右两部分之间的最小跨边。

Phantoms of the XOR Tree

哪些边会出现在某棵 MST 中

设某个分叉点左右两侧之间的最小跨边权为 $W_{\min}$。
则：

- 任意权值大于 $W_{\min}$ 的跨边，不可能出现在任何最小生成树中；
- 任意权值等于 $W_{\min}$ 的跨边，都可以作为连接左右两连通块的候选边，因此都可能出现在某一棵最小生成树中。

问题的等价转化

一条边属于 G_{MST}，当且仅当它在 01-Trie 的某个分叉节点处，是连接左右子树的最小异或跨边之一。
因而原问题转化为：对 Trie 中每个同时具有左右儿子的节点，求出其左右子树之间的最小异或值，并统计达到该最小值的数对个数，最后对这些计数求和。

Phantoms of the XOR Tree

Trie 节点与排序区间的对应关系

先将数组 A 升序排序，再建立 01-Trie。
排序后，具有相同高位前缀的数必然连续。因此 Trie 中每个节点都对应原数组中的一个连续区间 $[L_u, R_u]$，记其覆盖元素集合为 S_u。

每个分叉点的局部任务

对于某个分叉节点 u，设左右儿子对应的集合分别为 S_{lc}, S_{rc}，需要计算：
$W_{\min} = \min_{x \in S_{lc},\, y \in S_{rc}} (x \oplus y)$，以及
$\mathrm{Count}_u = \#\{(x, y) \mid x \in S_{lc},\, y \in S_{rc},\, x \oplus y = W_{\min}\}$。
最终答案即为所有分叉点的 Count_u 之和。

Phantoms of the XOR Tree

启发式枚举较小一侧

若直接枚举 S_{lc} 与 S_{rc} 的全部数对，则最坏复杂度会退化为 $O(n^2)$。
有效做法是始终由较小的一侧去查询较大的一侧：

- 若 $|S_{lc}| \le |S_{rc}|$，则枚举 $x \in S_{lc}$；
- 对每个 x，在以 rc 为根的子 Trie 中贪心查询与其异或值最小的 y；
- 查询时优先走与当前位相同的分支，从而在 $O(\log \max A)$ 的时间内得到最优匹配值。

最小值与计数的同步维护

在遍历较小集合的过程中，维护当前最优跨边权 $MinVal$ 及其出现次数 Cnt：

- 若某次查询结果小于 $MinVal$，则更新 $MinVal$ 并重置 Cnt；
- 若某次查询结果等于 $MinVal$，则继续累加计数。

Phantoms of the XOR Tree

复杂度分析

- 排序与建 Trie 的复杂度为 $O(n \log n + n \log(\max A))$。
- 在分叉点匹配阶段，始终由较小集合去查询较大集合。一个元素只有在自己所在一侧是较小集合时才会被单独枚举，而每经历一次这种情况，其所在集合规模至少翻倍，因此每个元素最多参与 $O(\log n)$ 次查询。
- 每次 Trie 查询复杂度为 $O(\log(\max A))$。

因而总时间复杂度为 $O(n \log n \log(\max A))$。
Trie 节点总数与总二进制位数同阶，因此空间复杂度为 $O(n \log(\max A))$。

PigeonG's Encoded Sequences

问题描述

给定 n 个字符串 $S_1, S_2, \ldots, S_n$，满足总长度 $\sum_{i=1}^{n} |S_i| \leq 2 \times 10^5$。现有 q 次询问，每次给出一对下标 (u, v)，要求计算拼接串 $T = S_u \circ S_v$ 的本质不同子串个数。

数据性质

题目保证所有字符串均由字符集 $\Sigma = \{\mathrm{A}, \mathrm{B}, \ldots, \mathrm{Z}\}$ 上独立、均匀、随机生成。

PigeonG's Encoded Sequences

随机模型下的核心观察

若不利用随机性，则每次询问均需重新处理拼接串，代价至少与 $|S_u| + |S_v|$ 同阶，难以满足要求。

本题的关键在于：对两段独立随机生成的字符串而言，较长公共子串出现的概率极低。

设两个长度均为 L 的随机串片段 A, B 相互独立，则 $\mathbb{P}(A = B) = |\Sigma|^{-L} = 26^{-L}$。另一方面，全体字符串总长度不超过 2×10^5，因此任取两个长度为 L 的子串进行比较，其候选对子总数至多为 $(2 \times 10^5)^2 = 4 \times 10^{10}$。

PigeonG's Encoded Sequences

期望层面的估计

记随机变量 X 为“全集中相等的长度为 L 的子串对数”，则由期望的线性性质，有 $\mathbb{E}[X] \le 4 \times 10^{10} \cdot 26^{-L}$。
代入具体数值：

- 当 $L = 8$ 时，$\mathbb{E}[X] \approx \frac{4\times10^{10}}{2.08\times10^{11}} \approx 0.19$；
- 当 $L = 9$ 时，$\mathbb{E}[X] \approx \frac{4\times10^{10}}{5.42\times10^{12}} \approx 0.007$。

因而在题目的随机数据下，可以取一个很小的常数阈值 M（例如 $M = 8$），并将“不同字符串之间存在长度大于 M 的公共子串”视为极小概率事件。

PigeonG's Encoded Sequences

答案的容斥拆分

设 $\mathcal{D}(S)$ 表示字符串 S 的本质不同子串集合。对拼接串 $S_u \circ S_v$ 而言，其子串可分为三类：

- 完全位于 S_u 内部的子串；
- 完全位于 S_v 内部的子串；
- 跨越拼接位置的子串。

因而有 $|\mathcal{D}(S_u \circ S_v)| = |\mathcal{D}(S_u)| + |\mathcal{D}(S_v)| - |\mathcal{D}(S_u) \cap \mathcal{D}(S_v)| + |\mathcal{C}(u, v)|$，其中 $\mathcal{C}(u, v)$ 表示跨越边界、且未在原有两串中出现过的新子串集合。

PigeonG's Encoded Sequences

阈值 M 带来的简化

由于不同字符串之间几乎不会共享长度大于 M 的子串，上式中的后两项均可限制在短串范围内处理。

- 对于交集项 $|\mathcal{D}(S_u) \cap \mathcal{D}(S_v)|$，只需统计长度不超过 M 的公共子串数。
- 对于跨界部分，设其由长度为 x 的后缀与长度为 y 的前缀拼接而成：
  - 若 $x + y > M$，则其长度已超过阈值，不会在原串中重复出现，且这些长跨界串彼此也互不相同，因此该部分可以直接计数；
  - 若 $x + y \leq M$，则候选总数不超过 $\frac{M(M-1)}{2}$，可直接枚举并判断其是否已经出现。

PigeonG's Encoded Sequences

预处理内容

- **单串本质不同子串数**：对每个 S_i 独立建立后缀自动机，可在线性时间内求出 $|\mathcal{D}(S_i)|$。
- **短子串哈希**：枚举每个字符串中长度不超过 M 的子串，并利用字符串哈希进行编码。
- **离散化建集合**：将全部短子串哈希统一离散化，记每个字符串对应的去重后整数集合为 $\mathcal{H}_i$。

于是交集项转化为 $|\mathcal{H}_u \cap \mathcal{H}_v|$。

PigeonG's Encoded Sequences

离线求交的均摊分析

为处理大量询问，需要避免对每一对集合直接双指针或暴力求交。
一种有效做法是按某一端点离线分组。处理与某个 v 相关的询问时，先利用布尔数组或时间戳标记 $\mathcal{H}_v$ 中的全部元素，再遍历较小的另一侧集合 $\mathcal{H}_u$，即可在线性于 $\min(|\mathcal{H}_u|, |\mathcal{H}_v|)$ 的时间内得到交集大小。
在整体均摊分析下，这一阶段的总复杂度可控制为 $O\left(\sum |S_i| \sqrt{\sum |S_i|}\right)$。

自拼接的特殊性

当 $u = v$ 时，拼接的是 $S_u \circ S_u$。此时“两串独立随机”的前提不再成立，前述关于长公共子串几乎不存在的结论不能直接套用。
因此对 $u = v$ 的询问，需要显式构造拼接串，并重新建立后缀自动机计算答案。

PigeonG's Encoded Sequences

复杂度分析

- 预处理阶段：建立后缀自动机为 $O(\sum |S_i|)$；提取长度不超过 M 的短子串并完成哈希离散化，代价为 $O(\sum |S_i| \cdot M \log(|S_i| \cdot M))$。
- 跨界短串判定：每次询问需枚举 $O(M^2)$ 个长度不超过 M 的候选，单次复杂度为 $O(M^2 \log M)$，总计 $O(q \cdot M^2 \log M)$。
- 离线求交阶段：总复杂度均摊为 $O\left(\sum |S_i| \sqrt{\sum |S_i|}\right)$。

因而总时间复杂度为 $O\left(\sum |S_i| \cdot M + q \cdot M^2 \log M + \sum |S_i| \sqrt{\sum |S_i|}\right)$。

空间复杂度

后缀自动机节点总数不超过 $2\sum |S_i|$；短子串哈希映射占用至多 $M\sum |S_i|$ 级别空间，因此整体空间复杂度为 $O(\sum |S_i| \cdot |\Sigma| + q)$。

Delete or not

差分转化

记 $d_i = a_{i+1} - a_i$。由于原序列严格递增，故 $d_i > 0$。
对于询问 $[L, R]$，只需考虑差分区间 $d_L, d_{L+1}, \ldots, d_{R-1}$。原操作中删除中间元素 B_i 的条件 $2B_i \le B_{i-1} + B_{i+1}$ 等价于 $B_i - B_{i-1} \le B_{i+1} - B_i$，即左侧差分不超过右侧差分。

等价问题

因而一次删除操作，等价于合并两个相邻差分。原问题转化为：将差分区间划分为尽量少的若干块，使得每一块都可以完全合并为一个差分。

Delete or not

一段差分可以完全合并的充要条件

对于连续区间 $d_l, d_{l+1}, \ldots, d_r$，其能够完全合并，当且仅当对任意 $i \in [l, r-1]$ 都有 $d_i \le \sum_{t=i+1}^{r} d_t$。

证明

必要性：d_i 被合并时，其右侧已经先合并成一段，故右侧总长度至少为 $\sum_{t=i+1}^{r} d_t$。

充分性：若上述条件对所有 i 成立，则从右向左依次合并即可，每一步均合法。

Delete or not

最优分块的唯一性

固定右端点 k，考虑最后一块。若最后一块为 $d_{x+1}, d_{x+2}, \ldots, d_k$，则由上一页的结论，这一块向左再扩展一个位置的充要条件为 $d_x \le \sum_{t=x+1}^{k} d_t$。
因而最后一块的左边界由最靠右的失败位置唯一决定，即最大的 $x < k$，满足 $d_x > \sum_{t=x+1}^{k} d_t$。若不存在这样的 x，则最后一块可以一直延伸到最左端。

递推结构

最后一块一旦确定，左侧剩余部分仍是同类问题。因此最优分块可以从右向左唯一递推。

Delete or not

断点条件的化简

由 $\sum_{t=x+1}^{k} d_t = a_{k+1} - a_{x+1}$ 与 $d_x = a_{x+1} - a_x$，条件 $d_x > \sum_{t=x+1}^{k} d_t$ 等价于 $2a_{x+1} - a_x > a_{k+1}$。

记 $V_x = 2a_{x+1} - a_x$，则当右端点为 k 时，最后一块左侧的断点就是满足 $V_x > a_{k+1}$ 的最大下标 $x < k$。

意义

断点位置只依赖于 V_x 与 a_{k+1}，与更左侧如何分块无关，因此可以整体预处理。

Delete or not

单调栈

对每个 k，记 $h(k)$ 为满足 $V_x > a_{k+1}$ 的最大下标 $x < k$，若不存在则记为 0。从左到右扫描时，维护一组候选下标，使其下标递增、对应的 V 严格递减。若 $i < j$ 且 $V_i \le V_j$，则位置 i 对任何未来的右端点都不可能优于位置 j，故可以删去。

求值方式

这样一来，$h(k)$ 就是候选集合中最后一个满足 $V_x > a_{k+1}$ 的位置，可以二分得到。

Delete or not

询问的数学形式

对询问 $[L, R]$，若 $L = R$，答案为 1。否则从 $R-1$ 出发，不断作用映射 h，直到结果小于 L 为止。
若在此过程中一共得到 c 个不小于 L 的位置，则差分区间被划分为 $c+1$ 块，因此原序列的最小长度为 $c+2$。

复杂度优化

为了快速计算多次复合，可以预处理映射 h 的二进制幂次迭代。记 $h^{(m)}$ 为 h 的 m 次复合，则可预处理全部 $h^{(2^j)}$，再用倍增统计区间内部经过的断点个数。

Delete or not

复杂度分析

计算全部 $h(k)$ 的复杂度为 $O(n \log n)$，预处理二进制幂次迭代的复杂度为 $O(n \log n)$，单次询问的复杂度为 $O(\log n)$。
因而总时间复杂度为 $O(\sum n \log n + \sum q \log n)$，空间复杂度为 $O(\sum n \log n)$。

Substring Game

题意

- q 个询问，每个询问给定一个长度为 n 的由 s、c、u、t 4 个字符组成的字符串 s
- 两个玩家轮流操作，每次操作选择从 s 的头或尾取出一个字符加入自己的字符串
- 当 s 取空游戏结束，如果先手的字符串有 ‘scut‘子串则先手获胜，否则后手获胜
- $1 \leq n \leq 10^4$ ， $\sum n \leq 5 \cdot 10^4$

Substring Game

- 每次只能从两端取字符，具有明显的“区间缩小”特征，不妨考虑使用区间 DP
- 又我们只在意一个区间里，Hiraethsoul 能否取出 *scut* 的后缀，不妨定义状态为 $f[i][j][k]$ 表示在区间 $[i, j]$ Hiraethsoul 在已经取得 *scut* 的长度为 k 的前缀结尾的前提下，能否取出带 *scut* 子串的字符串
- 状态设计后，我们考虑转移，根据当前轮到谁操作，有不同的转移方式

Substring Game

- **Hiraethsoul（先手）**
  - 只要加上选择取走的字符后，可以匹配到 scut 即可，取头取尾都可以
  - $\Rightarrow$ **OR 转移**
  - 假设 t 为目标字符，cl, cr 为左右端字符，转移方程为

$$f[l][r][k] = (f[l+1][r][k+1] \wedge cl == t) \ \vee (f[l][r-1][k+1] \wedge cr == t)$$

- **hjh（后手）**
  - 必须两种取法都到得了目标状态，否则就可以阻止先手到这个状态，但是要注意如果某种取法后剩下的区间先手可以直接得到 $scut$，后手一定会选另一种取法
  - $\Rightarrow$ **AND 转移**
  - 后手操作不影响转移进度，在不考虑先手直接取出 $scut$ 的情况下，转移方程为

$$f[l][r][k] = f[l+1][r][k] \wedge f[l][r-1][k]$$

Substring Game

- 最终答案为 $f[1][n][0]$，复杂度为 $O(n^2)$，但是要枚举第 3 维的 k，常数较大
- 考虑用一个长度为 4 的二进制掩码替代掉第 3 维的 k，掩码的第 i 位表示是否匹配到了 *scut* 的后缀长度为 i 的状态
- 掩码有 16 种值，先手转移需要考虑左右两个字符，最多有 16*16*4*4 种情况，后手转移则有 16*16 种情况
- 故可以考虑先预处理出所有转移情况，总体复杂度为 $O(n^2)$，常数较小

Parallel Pipeline Scheduling

题意

- 给定长度为 n 的序列 $T_1, T_2, ..., T_n$，给定 k
- 问最小的 x 满足，可以将序列划分成 $\leq k$ 段，使得每段满足 $\sum_{i=l}^{r} T_i - \max_{l \leq j \leq r} T_j \leq x$ 且 $\max_{1 \leq j \leq n} T_j \leq x$
- $1 \leq k \leq n \leq 10^6$, $1 \leq T_i \leq 10^6$

- 不难发现，随着划分的段数增加，最小的 x 是单调递减的，因此可以考虑二分答案。
- 二分答案的判定过程中，因为段是连续的，我们可以使用贪心策略：当前段尽量加入元素，若加入后代价 $> mid$，则必须新开一段。
- 假设当前在考虑第 l 个元素，$[l, n]$ 段的最优划分数不少于 $[l+1, n]$ 段的最优划分数，因此能加就加是不劣的
- 时间复杂度为 $O(n \log V)$，其中 V 是 $\sum T_i$

GCD and LCM SubSequences

题意

给定长度为 n 的序列 $a_1, a_2, ..., a_n$，给定 L 和 G
问有多少个子序列满足子序列的 GCD 为 G，LCM 为 L
$1 \le n \le 5 \times 10^5$, $1 \le a_i, G, L \le 10^{12}$

- 简化问题：
  首先我们先取 $T = L/G$，如果某个数字 $A[i]$ 不是 G 的倍数或不是 L 的因数，那这个数字就是没用的，否则令 $A[i]/ = G$；这样题目变为统计 $\gcd = 1$, $\mathrm{lcm} = T$ 的子序列个数

GCD and LCM SubSequences

- 首先我们对 T 进行质因数分解，记为 $p_1^{e_1} p_2^{e_2} \dots p_x^{e_x}$ （复杂度 $O(\sqrt{T})$）
- 不难发现，一个合法的子序列要满足，对每个 p_i，一定存在数 a 和 b 满足 $p_i \nmid a$ 和 $p_i^{e_i} \mid b$
- 又可以可以证明 $x \leq 11$ ，固我们考虑将每个数用一个长 $2x$ 位的掩码表示：
  - 前 x 位表示第 i 个质数的幂是否为 0
  - 后 x 位表示第 $i-x$ 个质数的幂是否为 e_{i-x}
- 满足条件的子序列的数字掩码按位或恰为 $2^{2x}-1$
- 我们不妨设 $G[S]$ 表示掩码是 S 子集的数的个数
- $g[S]$ 表示掩码恰是 S 的数的个数
- $F[S]$ 表示按位或是 S 子集的子序列个数
- $f[S]$ 表示按位或恰是 S 的子序列个数，我们想要的答案为 $f[2^{2x}-1]$

GCD and LCM SubSequences

- 利用前面的 p 数组，我们可以快速得到每个 a_i 的掩码表示，并将其加入到对应的 $g[S]$ 里，复杂度 $O(n \cdot x)$
- 现在已经有了 $g[S]$，$G[S]$ 为 $g[S]$ 的高维前缀和，可以通过 SOS DP 得到对应的 $G[S]$
- $F[S]$ 里的每个子序列都是 $G[S]$ 里的数取或不取生成，故 $F[S] = 2^{G[S]} - 1$
- $f[S]$ 可以由 $F[S]$ 通过逆 SOS DP 得到：

$$f[S] = F[S] - \sum_{T \subset S} f[T]$$

- 最终答案为：$f[2^{2x} - 1]$，SOSDP 复杂度为 $O(x \cdot 2^{2x})$
- 总体复杂度为 $O(\sqrt{T} + n \cdot x + x \cdot 2^{2x})$

Choose

给定两组数组 $a_1, a_2, \ldots, a_n$ 和 $b_1, b_2, \ldots, b_n$，我们首先做初始选择：

$$\forall 1 \le i \le n, \quad \text{选择} a_i$$

定义初始异或和：

$$sum = \bigoplus_{i=1}^{n} a_i$$

其中 $\oplus$ 表示按位异或运算。

Choose

若将第 i 个位置的 a_i 替换为 b_i，等价于对当前异或和 sum 异或 $(a_i \oplus b_i)$。据此构造辅助数组 c：

$$c_i = a_i \oplus b_i \quad (1 \le i \le n)$$

Choose

问题转化：从数组 c 中选择若干元素与初始 sum 异或，使得最终结果最大化。
核心解法：线性基

- 线性基可高效处理异或最大化问题；
- 时间复杂度：$O(n \log V)$（V 为数值的取值范围）。

Matrix Construction

- 假设 $h \leq d$
- 每个数字在矩阵中最多出现 h 次
- 求和约束：$sum = \sum_{i=1}^{n} \min(h, a_i)$，需满足 $h \times d \leq sum$

Matrix Construction

矩阵坐标 (x, y) 表示第 x 行、第 y 列：

- **同一斜列**：若 $x_1 - x_2 = y_1 - y_2$，则 (x_1, y_1) 与 (x_2, y_2) 属于同一斜列
- **相邻斜列**：若 $|(x_1 - x_2) - (y_1 - y_2)| = 1$，则两坐标属于相邻斜列

Matrix Construction

- 同一数字仅分布在**一个斜列**：最多出现 h 次
- 同一数字仅分布在**相邻两个斜列**：最多出现 $h-1$ 次

Matrix Construction

构造策略：

1. 将数组 a 从大到小排序
2. 按相邻斜列的顺序遍历矩阵并放置数字

策略逻辑：

1. 数量 $\geq h$ 的数仅处于同一斜列，数量 $< h$ 的数最多处于相邻两个斜列

时间复杂度： $O(\max(h \times d), n \log n)$