第 11 届中国大学生程序设计竞赛郑州站

题目分析

2025 年 11 月 23 日

总体情况

预期难度顺序

MB $<$ GJ $<$ ID $<$ HEK $<$ AC $<$ LF

奖牌线（封榜前）

Au: 5 题 553
Ag: 3 题 359
Cu: 2 题 264

M. 替换

Description

给定一个 01 串，选择一个位置改成 $\oplus$，使得最后表达式结果最大。

通过情况

- AC before frozen: 282
- First Solve: 四川大学：你们在干什么喵？在罚时喵！补药罚我时喵！ 11min
- Shortest Judge Solution: 856 Bytes
- Does gpt5-pro pass: Yes

M. 替换

Solution

不妨令从串的第一个 1 到结尾共有 x 位，则容易发现最终答案的位数不会超过 x。

另一方面，当 $x \geq 4$ 的时候，选择 $n-1$ 处改为 $\oplus$ 能保证答案至少有 $x-2$ 位。

因而只需要枚举替换的位置在开头 $O(1)$ 位和结尾 $O(1)$ 位的情况，并取最大值即可。

如果通过更细致的分析，可以说明只有 $2, 3, n-1$ 可以成为可能的位置。

B. 切巧克力

Description

给一个三维长方体，其中有 n 个关键单位立方体，需要在三个维度分别切 p, q, r 刀，使得切出来的每个小长方体包含的关键单位立方体的个数相同。
问有多少种本质不同的切法。

通过情况

- AC before frozen: 189
- First Solve: CF 皇帝 13min
- Shortest Judge Solution: 2126 Bytes
- Does gpt5-pro pass: Yes

B. 切巧克力

Solution

首先 n 是 $(p+1)(q+1)(r+1)$ 的倍数的时候才可能答案不为 0。
观察发现三个维度的方案数独立，比方说对于 x 这一维，把所有关键点按 x 排序，那么应该在第 $\frac{n}{p+1}$ 和第 $\frac{n}{p+1}+1$ 个关键点之间切一刀，第 $\frac{2n}{p+1}$ 和第 $\frac{2n}{p+1}+1$ 个关键点之间切一刀，……
把每一刀切的方案数乘起来便是答案。
注意，最后还要检查一下是否真的每个块里关键点数量恰好相同。（实现方法之一就是开一个三维的桶来统计每个块里关键点的数量）

G. Plus Xor

Description

给定 a, b, c，判定能否通过将 a 加上 b 或者异或上 b 的操作变成 c。

通过情况

- AC before frozen: 176
- First Solve: 中山大学 - 中山大学-中大 25 一队 11min
- Shortest Judge Solution: 1167 Bytes
- Does gpt5-pro pass: Yes

G. Plus Xor

Solution

注意到如果 a 和 c 模 b 意义下同余，且 $a \leq c$，则可以通过不断加 b 达成条件。

令 b 在二进制下有 L 位。则只需要记录 $a \bmod 2^L$ 和 $a \bmod b$ 的值，可以唯一确定在进行一系列操作之后 a 的变化量。具体来说，令当前的 $(a \bmod 2^L, a \bmod b)$ 为 (x, y)：

- a 加上 b 会使 (x, y) 变为 $((x + b) \bmod 2^L, y)$
- a 异或上 b 会使 (x, y) 变为 $((x \oplus b) \bmod 2^L, (y + x \oplus b - x) \bmod b)$。

令 $f(x, y)$ 表示抵达状态 (x, y) 时，a 的最小值，在这张转移图上跑单源最短路算法即可。注意到这张图有负权边，但是可以证明 Dij 算法在图上仍然正确工作。另外由于图的形状特殊，spfa 算法也可以通过。

如果你想有理有据的使用 dij 通过这个题，可以注意到不会有连续两次异或操作，而将操作重新修改为 $a + b$ 和 $a \oplus b + b$。此时边权均为正数。

J. 子矩形计数

Description

给一个长度为 n 的序列 a 和一个长度为 m 的序列 b，构造 $c_{i,j} = a_i \oplus b_j$ 。

问有多少个位置 (i, j) 满足

$a_{i,j+1} = a_{i,j} + 1, a_{i+1,j} = a_{i,j} + 2, a_{i+1,j+1} = a_{i,j} + 3$ 。

通过情况

- AC before frozen: 73
- First Solve: 清华大学 - 零基础新生 1 队 39min
- Shortest Judge Solution: 1085 Bytes
- Does gpt5-pro pass: Yes

J. 子矩形计数

Solution

设 $x = c_{i,j}$

把一个非负整数 x 变成 $x+1$ ，相当于要异或一个全 1 的二进制数，其长度与 x 中最低位的 0 位置相同。例如，

$11010111_2 + 1_2 = 11010111_2 \oplus 1111_2$ 。

类似地，从 x 变成 $x+2$ 相当于异或一个从第二位开始地一段连续的 1 。

例如，$11010111_2 + 10_2 = 11010111_2 \oplus 1110_2$ 。

J. 子矩形计数

Solution

考虑按照 x 的前两个二进制位分类：

- 11 或 01：此时 $x \oplus (x+1) \neq (x+2) \oplus (x+3)$，但是它们都应该等于 $b_j \oplus b_{j+1}$，因此不存在这种情况。
- 00：不会发生进位，因此要求 $b_j \oplus b_{j+1} = 1$，$a_i \oplus a_{i+1} = 10_2$。
- 10：此时需要 $b_i \oplus b_{i+1} = 1$ 且 $a_i \oplus b_j + 2 = a_{i+1} \oplus b_j$。

J. 子矩形计数

Solution

总结一下，$c_{i,j}$ 满足条件的要求是：

- $b_j \oplus b_{j+1} = 1$
- 存在一个形如 01110_2 的串，中间的 1 个数可以任意。满足 $a_i \oplus b_j$ 的低位部分是这个串，而 $a_i \oplus a_{i+1}$ 的低位部分是 11110_2 。

用字典树或 map 等的数据结构维护一下即可。

时间复杂度 $O(m \log v)$ 或 $O(m \log^2 v)$

I. 笨蛋题 II

Description

求 k 个长度为 n 的随机排列本质不同的前缀 max 数字集合个数。

通过情况

- AC before frozen: 46
- First Solve: 清华大学 - Escape 49min
- Shortest Judge Solution: 1282 Bytes
- Does gpt5-pro pass: Yes

I. 笨蛋题 II

Solution

不妨假设前缀 max 对应的数字集合为 S，则考虑从大到小插入每个数 x 的过程：

- 如果 $x \in S$，则只有一种插入的方式（插在序列开头）
- 否则，有 $(n - x)$ 种插入的方式（除了开头都行）

故集合 S 作为前缀 max 集合的方案数为 $f(S) = \prod_{x \notin S}(n - x)$。

I. 笨蛋题 II

Solution

令 g_i 表示 i 个排列的前缀 max 集合相同的概率，则 $g_i = \frac{\sum_S f(S)^i}{(n!)^i} = \frac{\prod_{1\le x\le n}(1+(n-x)^i)}{(n!)^i}$。容易在 $O(nk)$ 时间内计算 g。

考虑枚举一个下标集合 P，计算 P 恰好是一个极大的，前缀 max 集合相同的排列集合的概率，将所有 P 的概率相加即为答案。其概率为 $\sum_{P\subseteq Q} g_{|Q|}\times(-1)^{|Q|-|P|} = \sum_{j\ge|P|}\binom{n-|P|}{j-|P|} g_j \times (-1)^{j-|P|}$。显然这个式子只和 $|P|$ 有关，枚举 $|P|$ 计算即可。

D. 树的直径

Description

给定一棵树，填入点权为一个排列使得字典序最大的直径字典序尽量小。

通过情况

- AC before frozen: 27
- First Solve: 清华大学 - 队仨 68min
- Shortest Judge Solution: 3359 Bytes
- Does gpt5-pro pass: No

D. 树的直径

Solution

首先，所有的直径中点一定相同（如果直径长度为偶数），或者中边一定相同（如果直径长度为奇数）。不失一般性，假设直径的中点相同。以下讨论将树的根设为直径中点，并删去所有不可能在直径上的点。

考虑所有可能成为直径端点的叶子，假设共有 k 个，显然应该在它们中填入 $1 \dots k$，并且字典序最大的直径一定以这个点为起始点。假设确定了起始点，则从起始点到中点一定是 $k, k+1, k+2 \dots$，因为此时直径前半部分经过的点是唯一的。

抵达中点之后，不妨认为我们已经决定了直径的另一端是什么。令中点到直径另一端的序列为 $v_m, v_{m+1} \dots v_k$。则 v_{m+1} 一定是 v_m 的孩子中点权最大的，v_{m+2} 一定是 v_{m+1} 的孩子中点权最大的。令 d_i 表示 i 的孩子个数，p_{v_m} 表示 v_m 的点权。则点权序列为 $(p_{v_m}, p_{v_m} + d_{v_m}, p_{v_m} + d_{v_m} + d_{v_{m+1}}) \dots$。需要特判最后一步抵达叶子的点权（因为叶子点权 $\leq k$）。

D. 树的直径

Solution

故我们发现，后半部分最优解是找到 d_i 字典序最小的，从根到叶子的一条路径。这一步可以通过 bfs 实现（只保留当前层中字典序最小的节点们）。

确定后半部分序列后，前半部分一定是 $k, k+1 \ldots$。
对于直径长度为奇数的情况也是类似的，可以通过添加一个不占用点权的虚点处理。

H. 随机打乱

Description

给定长度为 n 的数组 a 和分数数组 c，第 i 个位置的分数为 c_i，Alice 和 Bob 轮流操作：

- 当前玩家复制 a 到 b，并随机打乱 b（所有排列等概率）
- 设 L 为 a 与 b 的最长公共前缀长度
- 玩家可选择 $k \in [0, L]$，删除 a 长度为 k 的前缀，并获得这些位置的对应分数。

当 a 为空时游戏结束。双方均采取最优策略以最大化自己的得分期望。求 Alice 能获得的分数期望。（要求时间复杂度线性）

通过情况

- AC before frozen: 3
- First Solve: 北京大学 - 飞带不长队 144min
- Shortest Judge Solution: 1224 Bytes
- Does gpt5-pro pass: No

H. 随机打乱

Solution

设 $p_i = \frac{cnt_{i,a_i}}{n-i+1}$，其中 cnt_{i,a_i} 表示 $a[i \dots n]$ 中等于 a_i 的个数。假设目前 a 还剩下 $a[i \dots n]$，那么 b 和剩下的 a 的 LCP 长度大于等于 j 的概率为 $\prod_{k=1}^{j} p_{i+k-1}$。

设 dp_i 表示剩余 $a[i \dots n]$ 的时候 Alice 还能获得的最大得分。

那么只需要考虑 $dp_{j \dots n}$ 构成的单调栈。注意 dp_i 能从自己转移到自己，所以假如我们假定了 dp_i 的值，那么便可以算出 dp_i 的" 真实值"，当两者相同时，意味着这个假定的值是对的。

H. 随机打乱

Solution

倘若知道了 dp_i 的值在单调栈中哪两个相邻元素之间，我们便可以把 dp_i 解出来。

容易发现如果我们假定了 dp_i 的值为 x，那么 x 越小，计算出来的 dp_i 的真实值越大。（意味着其实有个暴力的做法是二分 dp_i）

于是可以先假定 dp_i 的值为栈顶，然后 $O(1)$ 地计算出此时 dp_i 的真实值，若此时真实值小于假定的值，那么便可以弹出栈顶。当无法弹出栈顶的时候，便可以 $O(1)$ 计算出 dp_i。

具体实现的时候需要用单调栈同时维护一下 dp_j 乘以概率的后缀和，才能 $O(1)$ 计算 dp_i。

K. 括号与交换

Description

给一个长度为 n 的序列 a，一个序列 c 初始为 a。
要构造一个长度为 n 的合法括号序列 b，然后对于一对位置 (i, j) $(i < j)$，如果它们在 b 中匹配，则交换 c_i 和 c_j。求一个 b 使得 c 的字典序尽可能小。

通过情况

- AC before frozen: 3
- First Solve: 北京大学 - 飞带不长队 125min
- Shortest Judge Solution: 1278 Bytes
- Does gpt5-pro pass: No

K. 括号与交换

Solution

观察到一个结论：设字典序最小的 c 是序列 ans 。对于两个相邻的位置 i 和 $i+1$ ，如果 $c_i = ans_{i+1}$ 且 $c_{i+1} = ans_i$ ，那么一定存在一组最优解，使得 b_i 和 b_{i+1} 是一对匹配的括号。

证明就是如果 $i, i+1$ 没匹配，可以通过将它们原本匹配的位置匹配起来，然后再匹配 i 和 $i+1$ 。这样一定不劣。

K. 括号与交换

Solution

设 $f(l, r)$ 表示区间 $[l, r]$ 中，与 l 的奇偶性不一样的位置的 a_i 的最小值，$S(l, r)$ 表示这些最小值的位置集合。
从左往右依次考虑每个位置，对于一个位置 i:
如果它前面的所有位置都相互匹配完了，那么它会匹配 $S(i, n)$ 中的一个位置，但是我们还不能确定。
否则，考虑 i 之前最近的还没被确认的位置 j（它的奇偶性一定和 i 不同）。设 r_j 是 j 能够匹配的最远位置，那么 i 要么和 j 匹配，要么和 $S(i, r_j - 1)$ 中的位置匹配。
此时 $[j + 1, i - 1]$ 这一段都是确定相互匹配的，因此 j 和 i 可以看作是相邻的。故若 $a_i = F(j, r_j)$ 且 $a_j \leq F(i, r_j - 1)$，将 i 和 j 匹配起来一定不劣。
否则，i 会匹配 $S(i, r_j - 1)$ 中的位置。
用两个 ST 表分别维护奇数和偶数位置的区间最小值，用一个栈维护前面所有未确定的位置即可。
时间复杂度 $O(n \log n)$

E. 嘟嘟嘟

Description

有 n 堵墙，所有墙都是同心圆，每面墙上有若干个门，有 q 次询问，每次给出一个保证仅贴某个墙面的位置，问从这个位置出发到圆心的最短路。

通过情况

- AC before frozen: 1
- First Solve: CF 皇帝 235min
- Shortest Judge Solution: 4138 Bytes
- Does gpt5-pro pass: Yes

E. 嘟嘟嘟

Solution

考虑动态规划，设 $dp_{i,j}$ 表示从圆心走到第 i 面墙上第 j 个门的最短路。
考虑如何从 $dp_{i,*}$ 转移到 $dp_{i+1,*}$。
分为两种情况：是否贴着墙走一段圆弧。
对于第 $i+1$ 个圆上的某个门 u，第 i 个圆上在 u 和圆的两条切线以内的点是不需要贴着墙走的就能走到 u 的，容易发现这是环上的一段区间。

E. 嘟嘟嘟

Solution

需要贴着墙走的转移是容易的（其实只需要从离切点最近的门转移即可）。
对于不需要贴着墙走的转移，考虑断环成链，那么第 $i+1$ 个圆上的门能从第 i 个圆上的门转移到的是左右端点都递增的区间。
再观察到这个转移满足四边不等式（即转移的路径并不会交叉），所以决策单调性分治即可。

A. 挑战矩阵乘法 II

Description

构造一张不超过 85 个点的图，满足其 apsp 之和等于 x。

通过情况

- AC before frozen: 0
- First Solve: N/A
- Shortest Judge Solution: 4353 Bytes
- Does gpt5-pro pass: No

A. 挑战矩阵乘法 II

Solution

注意到 85 的限制非常紧。核心 idea 是构造一个比较容易计算答案的 pattern，并算出是否可以覆盖所有 $1 \dots 10^5$ 的 case（除了 2 和 5 无解）

构造有许多，std 的构造如下：

注意到 x 较小时，有比较简单的确定性构造：构造一个大小为 n 的完全图，再额外加入一个点连向之前的点。额外加入的点有直接连边的点贡献为 1，否则为 2。需要特判 $x = \frac{i(i-1)}{2} + 2i$ 的情况。此时需要 $O(\sqrt{x})$ 个点来处理 x 较小的情况。

A. 挑战矩阵乘法 II

Solution

x 较大时，采用一个 n 个点的链，并在链上选若干点不交的区间连边（即选取 $1 \le i_1 < j_1 < i_2 < j_2 \cdots < i_k < j_k \le n$，并连边 $(i_1, j_1), (i_2, j_2) \ldots (i_k, j_k)$）。容易发现每条边的贡献是独立的，且只和 n 和 (i, j) 有关。枚举 n 后，令 f_i 表示前 i 个点可能形成哪些贡献，用 bitset 优化 dp 即可。

计算可知上述构造能覆盖所有 $[500, 10^5]$ 之间的 case。多组询问只需离线所有询问后枚举 n 进行 dp。

C. 基础数数练习题

Description

给定一点权为排列的树，令 f_i 表示 i 出发只向子树移动，且不允许经过点权大于 p_i 的点，能到达的点的个数。求对于所有 $(i,j),p_i=j$ 时所有情况的 q_{f_i} 的和，其中 q 为一价值序列。

通过情况

- AC before frozen: 1
- First Solve: CF 皇帝 86min
- Shortest Judge Solution: 5211 Bytes
- Does gpt5-pro pass: No

C. 基础数数练习题

Solution

令 $dp_{i,j,k}$ 表示只考虑 i 的子树，且 i 的子树内填入 $1 \dots siz_i$，并且只允许经过点权 $\leq j$ 的点时，从 i 出发能到达 k 个点，这种情况下的填入点权的方案数。子树的 dp 值是容易合并的，$dp_{v_1,j,k} \cdot dp_{v_2,j',k'} \cdot \binom{j+j'}{j} \cdot \binom{siz_{v_1}-j+siz_{v_2}-j'}{siz_{v_1}-j} \to dp_{v_{new},j+j',k+k'}$。

合并完 u 的子树之后，暂时不加入点 u，可以以如下方式计算答案。枚举点 u 的权值 j，则 $dp_{u,j,k} \cdot \binom{j'-1}{j-1} \cdot \binom{n-j'}{siz_u-j} \to ans_{u,j',k}$。这个做法的时间复杂度为 $O(n^4)$。

C. 基础数数练习题

Solution

一个容易想到的优化方法是发现 k 这一维的 dp 值是个纯粹的卷积。故将 k 这一维视为一个多项式，维护其在 $\{1, 2, \ldots n\}$ 处的点值，而在 j 这一维使用普通的树上背包，此时求 dp 数组的复杂度被优化至 $O(n^3)$。

此时求答案变为复杂度瓶颈。如果我们尝试对每个多项式 $dp_{u,j}$ 进行插值还原多项式，再求出答案，则复杂度仍为 $O(n^4)$。不妨令 $dp_{u,j} = \sum a_k \cdot x^k$，其中多项式系数 $a_k = dp_{u,j,k}$，即前文中的 dp 值。我们发现我们并非关心每个 a_k 是多少，而关心 $\sum a_k \cdot q_k$，即 a_k 的某个线性组合的值。

C. 基础数数练习题

Solution

更进一步的观察是，从点值 $(v_0 \dots v_n)$ 还原出多项式系数 $(a_0 \dots a_n)$ 的过程也为一个固定的线性变换，因而多项式系数 a 的线性组合也为点值 v 的线性组合。形式化的，假设你维护了 $[x_0 \dots x_n]$ 处的点值，令范德蒙矩阵

$$M = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix}$$

，且令 $Q = [q_0, q_1 \dots q_n], A = [a_0, a_1 \dots a_n]^T, V = [v_0, v_1 \dots v_n]^T$，则有 $M \times A = V$，所求答案为 $Q \times A = Q \times M^{-1} \times V$。预处理 $Q \times M^{-1}$ 即可得到关于点值的线性组合。

C. 基础数数练习题

Solution

求出 $dp_{u,j}$ 对应的多项式点值之后，乘上预处理出的线性组合，即可得到 $dp_{u,j}$ 对应的答案。再枚举 j' 贡献进真正的答案 $ans_{u,j'}$ 处即可。时间复杂度 $O(n^3)$。

F. 挑战 NPC II

Description

给定一张图上的若干组边集，对每个 x 求出至少需要加入几组边集，使得连通块个数不超过 x。

通过情况

- AC before frozen: 1
- First Solve: CF 皇帝 158min
- Shortest Judge Solution: 2246 Bytes
- Does gpt5-pro pass: No

F. 挑战 NPC II

Solution

先做一些预处理：有经典的 trick，是对于每条边 (u, v)，定义长度为 n 的向量 $[0, 0, 0 \ldots 1, 0, 0 \cdots -1, 0, 0 \ldots]$，其中 u 处为 1，v 处为 -1，其余位置为 0。则一组边不成环当且仅当其线性无关。故选择若干个边集，使其连通块个数小于等于 x，等价于选择边集后，可以从中选出 $n - x$ 条边不成环，即从中选出 $n - x$ 个向量线性无关。

考虑如何维护线性相关的信息

F. 挑战 NPC II

Solution

外代数

不严谨的说，对于域 $\mathbb{F}$ 上的 n 维线性空间 V。定义在 V 上的外代数，大致是 2^n 个元素的线性组合，其中每个基本元素可以用集合 S 表示为 $\bigwedge_{x\in S} v_x$。例如 $v_1 \wedge v_3$，$v_2 \wedge v_4 \wedge v_5$ 等。

我们定义外代数仍有以下运算规则：

- $a + b = b + a$
- 对于 $b \in \mathbb{F}$，$(ub + v) \wedge p = b(u \wedge p) + v \wedge p$
- $v \wedge v = 0$

以上规则的直接推论是 $u \wedge v = -v \wedge u$

注意到一个外代数信息可以被 2^n 个域上元素表达。

F. 挑战 NPC II

Solution

考虑线性空间的一组基 $e_1, e_2 \dots e_n$，对于线性空间的向量 $w = \sum a_i \cdot e_i$，其对应的外代数元素为 $w' = \sum a_i \cdot v_i$。则 $\bigwedge_{i=1}^{k} w_i = 0$ 等价于这组向量线性相关。更进一步的，对于 w 中的每个向量分配一个变元 x_i，则对于 w 上若干个向量子集，关于 $x_1, x_2 \dots x_n$ 的多项式 $\sum_S \bigwedge_{i \in S} x_i w_i = 0$，当且仅当每个子集均线性相关。

根据熟知的 Schwartz–Zippel 引理，随机将 x_i 带入域中的元素只有 $\frac{n}{|\mathbb{F}|}$ 的判定失败概率。

F. 挑战 NPC II

Solution

接下来是 dp 的部分：将域定义为 $\mathbb{F}_p$，其中 p 为大质数。令 $f_{i,j,k}$ 表示考虑前 i 组边，选取了 j 个边集，并从中选取了 k 条边，对应的所有外代数信息。对于每条边，随机一个数字作为变元 x_i 的取值。转移时枚举当前边集是否选择，以及选择当前边集时按顺序考虑每条边是否选择。注意到此时的外代数乘法是 $O(1)$ 的，因为新增的每个向量只有两处非零的值。最后只需找到最小的 j 满足 $f_{k,j,n-x}$ 非零。这个做法是 $O(2^n \cdot n^2 \sum c)$ 的，难以通过。

进一步发现 k 是没用的，因为选取恰好 k 条边的外代数信息中只有大小为 k 的子集有值。只需要维护 $f_{i,j}$ 对应的外代数信息即可。复杂度 $O(2^n \cdot n \sum c)$。

L. 数三角形

Description

- 给定一个高度为 n 的正三角形网格，由边长为 1 的小正三角形（正向或反向）密铺而成。
- 一个正三角形用 (x, y, d) 描述：
  - 正向（尖朝上）：(x, y) 为底边对面的顶点，d 为边长（底边长度）。
  - 反向（尖朝下）：(x, y) 为底边左端点，d 为边长。
- 两种操作：
  1. **更新**：选一个三角形 (x, y, d)，将其覆盖的所有单位小三角形的数字加 w。
  2. **查询**：选一个三角形 (x, y, d)，求其覆盖的所有单位小三角形的数字之和，对 2^{32} 取模输出。

通过情况

- AC before frozen: 0
- First Solve: N/A
- Shortest Judge Solution: 15324 Bytes
- Does gpt5-pro pass: No

L. 数三角形

Solution

考虑把一个三角形拆成两个紧贴底边的正三角形和两个紧贴左下角的平行四边形。
所以有 4 部分贡献需要统计：正三角形对正三角形，正三角形对平行四边形，平行四边形对正三角形，平行四边形对平行四边形。
这些贡献都可以写成三位偏序的形式，于是可以 CDQ 分治然后用树状数组来维护。

L. 数三角形

Solution

比如正三角形对正三角形的贡献，不妨设两个正三角形覆盖的底边区间分别为 $[l_1, r_1]$ 和 $[l_2, r_2]$，那么贡献就是区间交长度的平方乘以权值，可以写成二位偏序的形式（加上时间维度就是三维）。
剩下的部分同理。需要非常细致的分类讨论。

谢谢！