模拟评讲

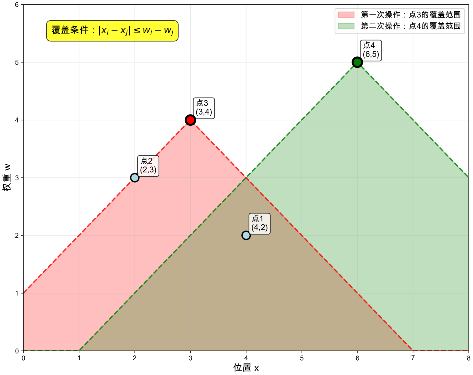
出题人：宋佳兴
评讲人：杨轶涵

2025 年 7 月 23 日

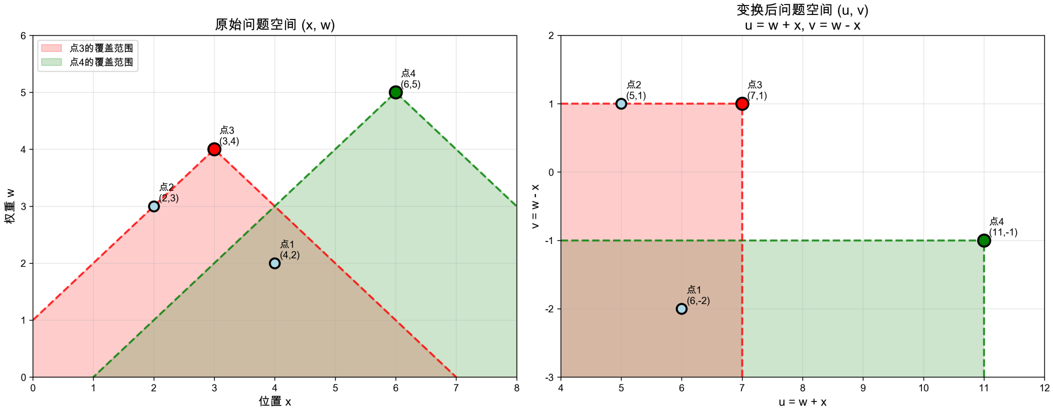
排序题 (sort)

核心问题

操作点 i 能涂黑哪些点?



排序题 (sort)



原始问题空间 (x, w)
点3的覆盖范围
点4的覆盖范围
点4 (6,5)
点3 (3,4)
点2 (2,3)
点1 (4,2)
权重 w
位置 x
变换后问题空间 (u, v)
u = w + x, v = w - x
点2 (5,1)
点3 (7,1)
点4 (11,-1)
点1 (6,-2)
v = w - x
u = w + x

排序题 (sort)

关键问题

什么时候点 i 能涂黑点 j?

条件：$|x_i - x_j| \le w_i - w_j$

排序题 (sort)

关键问题

什么时候点 i 能涂黑点 j?

条件：$|x_i - x_j| \le w_i - w_j$

绝对值展开

$$|x_i - x_j| \le w_i - w_j \Rightarrow \begin{cases} x_i - x_j \le w_i - w_j \\ x_j - x_i \le w_i - w_j \end{cases}$$

排序题 (sort)

关键问题

什么时候点 i 能涂黑点 j?

条件：$|x_i - x_j| \leq w_i - w_j$

绝对值展开

$$|x_i - x_j| \leq w_i - w_j \Rightarrow \begin{cases} x_i - x_j \leq w_i - w_j \\ x_j - x_i \leq w_i - w_j \end{cases}$$

移项整理

$$\begin{cases} x_i - x_j \leq w_i - w_j \\ x_j - x_i \leq w_i - w_j \end{cases} \Rightarrow \begin{cases} w_i - x_i \geq w_j - x_j \\ w_i + x_i \geq w_j + x_j \end{cases}$$

排序题 - 坐标变换

新坐标定义

- $u_i = w_i + x_i$
- $v_i = w_i - x_i$

排序题 - 坐标变换

新坐标定义

- $u_i = w_i + x_i$
- $v_i = w_i - x_i$

条件转化

点 i 能涂黑点 $j \Leftrightarrow u_i \geq u_j$ 且 $v_i \geq v_j$

排序题 - 坐标变换

新坐标定义

- $u_i = w_i + x_i$
- $v_i = w_i - x_i$

条件转化

点 i 能涂黑点 $j \Leftrightarrow u_i \geq u_j$ 且 $v_i \geq v_j$

几何意义

在二维平面 (u, v) 上，点 i 能涂黑点 j 当且仅当点 j 在点 i 的左下角!

排序题 - 问题转化

新问题

二维平面上有 n 个点，选择一个点会把它左下角的所有点都删除。
问：最少选择多少个点，才能删除所有点?

排序题 - 问题转化

新问题

二维平面上有 n 个点，选择一个点会把它左下角的所有点都删除。
问：最少选择多少个点，才能删除所有点?

关键观察

- 只有" 右上角" 的点才需要主动选择
- 因为它们不能被任何其他点连带删除
- 删除这些点后，可以连带删除其他所有点

排序题 - 问题转化

新问题

二维平面上有 n 个点，选择一个点会把它左下角的所有点都删除。
问：最少选择多少个点，才能删除所有点?

关键观察

- 只有" 右上角" 的点才需要主动选择
- 因为它们不能被任何其他点连带删除
- 删除这些点后，可以连带删除其他所有点

" 右上角" 的定义

一个点是" 右上角" 的，当且仅当不存在其他点同时在它的右边和上边。

排序题 - 算法实现

算法步骤

1. 按 u_i 从大到小排序（相同时按 v_i 从大到小排序）
2. 维护 v 的前缀最大值
3. 当遇到 $v_i >$ 当前前缀最大值时，该点是右上角的

计数 (count)

简化

如果所有条件中 X_i 相同，则可以把 $A_i = X_i$ 看作染黑一个位置，否则认为染白一个位置，限制变成了所有区间中必须有至少一个黑点。

DP 转移

设 $dp[i][j]$ 表示 i 前面的点已考虑完，j 是最后一个被染黑的点，$R \leq i$ 的限制被全部满足的方案数。设 l_i 表示以 i 作为右端点的区间的左端点的最大值。

- 如果 i 被染黑，则 $dp[i][i] = \sum_j dp[i-1][j]$ 。
- 如果 i 没有被染黑，则 $dp[i][j] = dp[i][j] + Xdp[i-1][j]$ 。
- 对于 $j < l_i$ ，设 $dp[i][j] = 0$ 。

复杂度 $O(n^2)$ 。

计数 (count)

一般情况

如果 X_i 互不相同，对于每个位置 A_i 可以找到最小的限制 X_i 。将所有最大值相同的限制取出，每个区间内部只有最小限制恰好是 X_i 的位置可能取到 X_i ，且其他位置对这个限制无影响。对最大值限制不同的区间独立计算方案数后乘起来即可。

离散化

本题区间总长度太大，直接 DP 复杂度过高，但是区间数很少，因此可以将原序列根据区间端点分段。每段内部只有至少有一个黑点和没有黑点两种情况，分别的方案数是 $X^n - (X-1)^n$ 和 $(X-1)^n$ 。

优化

前缀清零、前缀和、全局乘可以通过前缀和和打标记的方法优化，求出最小限制的过程可以通过差分或者线段树优化，最优复杂度是 $O(n \log n)$ 的。

预处理器 (preprocessor)

奇偶性转换

设第 i 行的长度为 $2X_i + P_i$，其中 $X_i \geq 0$。

- 当 $P_i = 0$ 时，长度 $= 2X_i$（偶数）
- 当 $P_i = 1$ 时，长度 $= 2X_i + 1$（奇数）

预处理器 (preprocessor)

奇偶性转换

设第 i 行的长度为 $2X_i + P_i$，其中 $X_i \geq 0$。

- 当 $P_i = 0$ 时，长度 $= 2X_i$（偶数）
- 当 $P_i = 1$ 时，长度 $= 2X_i + 1$（奇数）

约束转换

- 原约束：$2X_i + P_i \in [A_i, B_i]$
- 新约束：$X_i \in [A'_i, B'_i]$，其中：

$$A'_i = \lceil \frac{A_i - P_i}{2} \rceil, \quad B'_i = \lfloor \frac{B_i - P_i}{2} \rfloor$$

预处理器 (preprocessor)

总和约束转换

原约束：$\sum(2X_i + P_i) \in [S, T]$
转换为：$\sum X_i \in [S', T']$，其中：

$$S' = \lceil \frac{S - \sum P_i}{2} \rceil, \quad T' = \lfloor \frac{T - \sum P_i}{2} \rfloor$$

预处理器 (preprocessor)

总和约束转换

原约束：$\sum(2X_i + P_i) \in [S, T]$
转换为：$\sum X_i \in [S', T']$，其中：

$$S' = \lceil \frac{S - \sum P_i}{2} \rceil, \quad T' = \lfloor \frac{T - \sum P_i}{2} \rfloor$$

问题简化

求满足以下条件的 X_i 的方案数：

- $X_i \in [A'_i, B'_i]$ for all i
- $\sum X_i \in [S', T']$

预处理器 (preprocessor)

总和约束转换

原约束：$\sum(2X_i + P_i) \in [S, T]$
转换为：$\sum X_i \in [S', T']$，其中：

$$S' = \lceil \frac{S - \sum P_i}{2} \rceil, \quad T' = \lfloor \frac{T - \sum P_i}{2} \rfloor$$

问题简化

求满足以下条件的 X_i 的方案数：

- $X_i \in [A'_i, B'_i]$ for all i
- $\sum X_i \in [S', T']$

进一步拆解

答案 $= f(T') - f(S' - 1)$，其中 $f(k)$ 表示 $\sum X_i \leq k$ 的方案数。

预处理器 (preprocessor)

直接处理 $X_i \in [A'_i, B'_i]$ 很复杂。

预处理器 (preprocessor)

直接处理 $X_i \in [A'_i, B'_i]$ 很复杂。

解决方案：容斥原理去掉上界约束

- 枚举子集 $S \subseteq \{1, 2, \ldots, N\}$
- 对于 $i \in S$：强制 $X_i \geq B'_i + 1$（违反上界）
- 对于 $i \notin S$：只要求 $X_i \geq A'_i$
- 容斥系数：$(-1)^{|S|}$

预处理器 (preprocessor)

直接处理 $X_i \in [A'_i, B'_i]$ 很复杂。

解决方案：容斥原理去掉上界约束

- 枚举子集 $S \subseteq \{1, 2, \ldots, N\}$
- 对于 $i \in S$：强制 $X_i \geq B'_i + 1$（违反上界）
- 对于 $i \notin S$：只要求 $X_i \geq A'_i$
- 容斥系数：$(-1)^{|S|}$

变量替换

设 $Y_i = X_i - L_i$，其中：

$$L_i = \begin{cases} B'_i + 1 & \text{if } i \in S \\ A'_i & \text{if } i \notin S \end{cases}$$

新约束：$Y_i \geq 0$ 且 $\sum Y_i \leq k - \sum L_i$

预处理器 (preprocessor)

问题转化

问题转化为： $Y_i \geq 0$ 且 $\sum Y_i \leq K$（其中 $K = k - \sum L_i$）

预处理器 (preprocessor)

问题转化

问题转化为： $Y_i \geq 0$ 且 $\sum Y_i \leq K$（其中 $K = k - \sum L_i$）

经典插板法

$\sum_{i=1}^{N} Y_i \leq K(Y_i \geq 0)$ 等价于 $\sum_{i=1}^{N+1} Y_i = K$（引入一个虚拟变量 Y_{N+1}），进一步等价于 $\sum_{i=1}^{N+1} Z_i = K + N(Z_i \geq 1)$，其中 $Z_i = Y_i + 1$。
$K + N$ 个位置中选择 N 个位置放" 板子":

$$\text{方案数} = \binom{K+N}{N}$$

预处理器 (preprocessor)

问题

M 不一定是质数，不能借助逆元计算组合数！

预处理器 (preprocessor)

问题

M 不一定是质数，不能借助逆元计算组合数！

解决方案：直接计算组合数

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n \times (n-1) \times \cdots \times (n-k+1)}{k!}$$

预处理器 (preprocessor)

问题

M 不一定是质数，不能借助逆元计算组合数！

解决方案：直接计算组合数

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n \times (n-1) \times \cdots \times (n-k+1)}{k!}$$

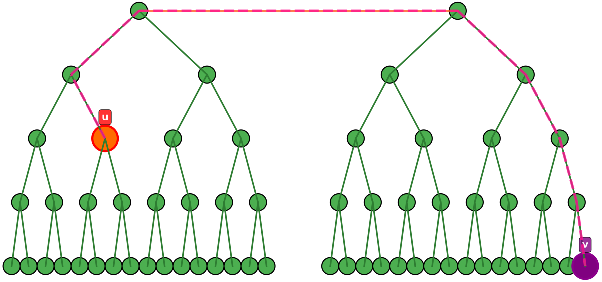
具体做法

- 分子：$n \times (n-1) \times \cdots \times (n-k+1)$
- 分母：$k! = 1 \times 2 \times \cdots \times k$
- 逐步约分：每次用分母的一个因子去约分子
- 最终分母变成 1，分子就是答案

树上最远距离问题的分解技巧

核心技巧

树上最远距离相关问题，有一个常用技巧：把直径最中间的一条边拎出来（若有两条任选一条），将这条边的两个端点作为根。



路径分类与最远点性质

重要性质

- 对于任意一个点 u，离它最远的点一定在另一棵树中最深的位置。
- 对于任意一条不跨过中心边的路径，离它最远的点也在另一棵树中最深的位置。

路径分类与最远点性质

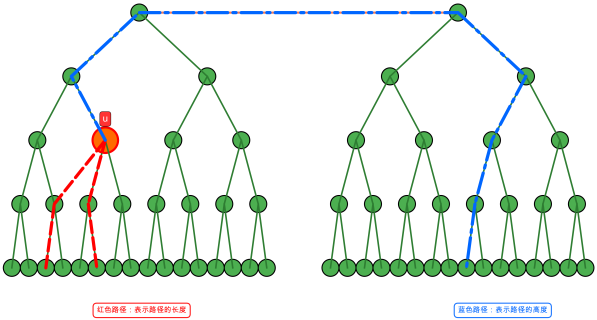
重要性质

- 对于任意一个点 u，离它最远的点一定在另一棵树中最深的位置。
- 对于任意一条不跨过中心边的路径，离它最远的点也在另一棵树中最深的位置。

路径分类

- 在本题中，我们将路径分为两类，第一类是不跨过中心边的路径。
- 第二类是跨过中心边的路径。

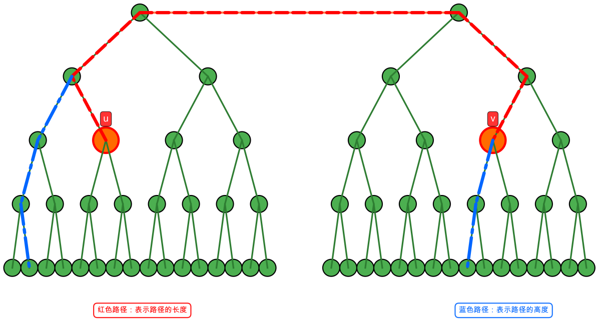
路径分类：不跨中心边



算法思路

- 红线表示路径的长度，蓝线表示路径的高度。
- 先通过简单树 DP 预处理，然后枚举路径的 LCA 点 u，可以统计出这类路径的信息。

路径分类：跨中心边



算法思路

- 两条蓝线表示可能的路径高度。
- 这类路径的特点是可以用 u 和 v 的信息合并出整条路径的信息。
- 对每个点 u 预处理出 u 到根路径的长度 L_u 和高度 H_u。

跨中心边路径的计算公式

计算公式

一条跨中心边路径的长度高度乘积为:

$$(L_u + L_v + 1)\max(H_u, H_v)$$

算法

- 枚举 $\max(H_u, H_v)$ 可以统计出这类路径的信息。
- 整体复杂度 $O(n)$。

最后一道 (last)

添加额外点

我们认为原题中所有点是黑点，并且在黑点间的边上添加一个白点。
删除点时，与这个点相邻的白点全部被合并成了一个白点，那么两个黑点之间有边的条件就是存在一个白点与这两个黑点相邻。

合并额外点

如果设 n 为根，那么根一定被最后删除，因此每次删除点时可以将这个点的所有儿子的白点合并到父亲上。这个合并可以通过并查集实现。

最后一道 (last)

答案计算

考虑三个满足条件的点，必须满足 a, b 之间有白点 d ，b, c 之间有白点 e 。设 $B[u]$ 为白点 u 儿子中的黑点个数。

- $d = e$ 时，贡献为

$$\sum_{d \in \text{white}} (B[d] + 1)B[d](B[d] - 1)$$

- $d \neq e$ 时，且 b 在 d, e 的上面。此时贡献可以枚举 b 的儿子，贡献为

$$2 \sum_{b \in \text{black}} \sum_{d,e \in child(b)} B[d]B[e]$$

最后一道 (last)

答案计算

考虑三个满足条件的点，必须满足 a, b 之间有白点 d ，b, c 之间有白点 e 。设 $B[u]$ 为白点 u 儿子中的黑点个数。

设 $S[d]$ 为白点 d 的下方距离为 2 的白点的 B 的和。

- $d \neq e$ 时，且 d, e 中 d 是 b 的父亲，此时贡献为：

$$2 \sum_{d \in \text{white}} B[d]S[d]$$

这样就可以 $O(n^2)$ 计算答案了。

最后一道 (last)

优化

我们需要维护白点的黑儿子个数 $B[u]$ ，黑点的儿子的 B 之和 $A[u]$ 以及白点的儿子的 A 之和 $S[u]$ 。每个点对答案的贡献可以通过这三个值计算得到。
删除一个点的时候，只有三个点的 A, B, S 会发生变化，也就是这个点和它的两级祖先。每次更新后重新计算修改点的贡献即可。

谢谢大家！

欢迎交流与提问