组合计数趣题选讲

华东师范大学第二附属中学 郭羽冲

2024.5.17

- 组合计数是 OI 中的一大板块。其中不乏许多趣味的、具有挑战性的题目。

- 组合计数是 OI 中的一大板块。其中不乏许多趣味的、具有挑战性的题目。
- 在本次交流中，我想与大家分享一些组合计数问题中偏向思维，通过分析性质来解决的题目。

LOJ177 生成子群阶数

题目描述

给定 m 个 n 阶置换，求它们生成的子群的阶数。
相当于给定 m 个排列，求它们相互复合共能得到多少种不同的排列。
$n, m \leq 50$。

LOJ177 生成子群阶数

题目描述

给定 m 个 n 阶置换，求它们生成的子群的阶数。
相当于给定 m 个排列，求它们相互复合共能得到多少种不同的排列。
$n, m \leq 50$。

- 例如 $(2, 1, 3, 4, 5), (5, 1, 2, 3, 4)$，它们可以得到所有 5 阶排列。

LOJ177 生成子群阶数

分析与观察

- 我们抛开群论，从组合的角度分析这个问题。

LOJ177 生成子群阶数

分析与观察

- 我们抛开群论，从组合的角度分析这个问题。
- 令给定的 m 个排列为 $a_1 \dots a_m$。若最终复合得出的排列为 b，考虑 $b_1 = x$ 的方案数。

LOJ177 生成子群阶数

分析与观察

- 我们抛开群论，从组合的角度分析这个问题。
- 令给定的 m 个排列为 $a_1 \dots a_m$。若最终复合得出的排列为 b，考虑 $b_1 = x$ 的方案数。
- 从 j 向 $a_{i,j}$ 连有向边，边权为 a_i。初始 $b_1 = 1$，每复合一个排列相当于是在有向图中选一条出边移动，最终到达 x。

LOJ177 生成子群阶数

分析与观察

- 我们抛开群论，从组合的角度分析这个问题。
- 令给定的 m 个排列为 $a_1 \dots a_m$。若最终复合得出的排列为 b，考虑 $b_1 = x$ 的方案数。
- 从 j 向 $a_{i,j}$ 连有向边，边权为 a_i。初始 $b_1 = 1$，每复合一个排列相当于是在有向图中选一条出边移动，最终到达 x。
- a_i 自复合若干次之后一定会得到 a_i^{-1}，我们不妨将所有 a_i^{-1} 也考虑进来。这正好会补上有向图中每条边的反向边，且边权互逆。

LOJ177 生成子群阶数

分析与观察

- 将它大致看作无向图。从 1 移动到 x 的过程相当于是选择了无向图上的一条路径，最终得到的排列即为路径上所有边权依次复合。

LOJ177 生成子群阶数

分析与观察

- 将它大致看作无向图。从 1 移动到 x 的过程相当于是选择了无向图上的一条路径，最终得到的排列即为路径上所有边权依次复合。
- 任选一个生成树，则任意一条 $1 \to x$ 的路径等价于一个经过 1 的圈与树上 $1 \to x$ 的唯一路径的拼接。

LOJ177 生成子群阶数

分析与观察

- 将它大致看作无向图。从 1 移动到 x 的过程相当于是选择了无向图上的一条路径，最终得到的排列即为路径上所有边权依次复合。
- 任选一个生成树，则任意一条 $1 \to x$ 的路径等价于一个经过 1 的圈与树上 $1 \to x$ 的唯一路径的拼接。
- 而排列的复合具有可逆性，因此 $b_1 = x$ 的方案数只与图中 1 是否能到达 x 有关。

LOJ177 生成子群阶数

分析与观察

- 将它大致看作无向图。从 1 移动到 x 的过程相当于是选择了无向图上的一条路径，最终得到的排列即为路径上所有边权依次复合。
- 任选一个生成树，则任意一条 $1 \to x$ 的路径等价于一个经过 1 的圈与树上 $1 \to x$ 的唯一路径的拼接。
- 而排列的复合具有可逆性，因此 $b_1 = x$ 的方案数只与图中 1 是否能到达 x 有关。
- 只需计算 $b_1 = 1$ 的方案数。

LOJ177 生成子群阶数

分析与观察

- 对于图中每条边 e，考虑它所对应的一个圈 C_e：$1 \to u_e, u_e \to v_e, v_e \to 1$。则任意一个经过 1 的圈等价于若干个 C_e 的拼接。

LOJ177 生成子群阶数

分析与观察

- 对于图中每条边 e，考虑它所对应的一个圈 C_e：$1 \to u_e, u_e \to v_e, v_e \to 1$。则任意一个经过 1 的圈等价于若干个 C_e 的拼接。
- 将每个圈对应的权值拿出来，构成一组新的排列 $a'_1 \dots a'_{m'}$。则它们必定满足 $a'_{i,1} = 1$，且这些排列恰好能生成所有 $b_1 = 1$ 的方案。

LOJ177 生成子群阶数

分析与观察

- 对于图中每条边 e，考虑它所对应的一个圈 C_e：$1 \to u_e, u_e \to v_e, v_e \to 1$。则任意一个经过 1 的圈等价于若干个 C_e 的拼接。
- 将每个圈对应的权值拿出来，构成一组新的排列 $a'_1 \dots a'_{m'}$。则它们必定满足 $a'_{i,1} = 1$，且这些排列恰好能够生成所有 $b_1 = 1$ 的方案。
- 归约为 $n-1$ 阶子问题。

LOJ177 生成子群阶数

时间复杂度问题

- 若直接实现上述算法，每轮均有 $m' = O(nm)$，指数级上升。时间复杂度不可接受。

LOJ177 生成子群阶数

时间复杂度问题

- 若直接实现上述算法，每轮均有 $m' = O(nm)$，指数级上升。时间复杂度不可接受。
- 需要用到一个数学上的结论进行优化：最多选出 $k = O(n)$ 个排列，使得其中任意一个都不能被剩余 $k-1$ 个生成。

LOJ177 生成子群阶数

时间复杂度问题

- 若直接实现上述算法，每轮均有 $m' = O(nm)$，指数级上升。时间复杂度不可接受。
- 需要用到一个数学上的结论进行优化：最多选出 $k = O(n)$ 个排列，使得其中任意一个都不能被剩余 $k-1$ 个生成。
- 事实上，若能在每层子问题中筛选出 $O(n)$ 个有效排列，则总时间复杂度可降为 $O(\text{poly}(n))$。

LOJ177 生成子群阶数

算法一（Schreier-Sims）

- 依次尝试加入每个排列 a_i，维护当前每层子问题对应的一组有效排列。“尝试加入”代表先判断是否有效再加入。

LOJ177 生成子群阶数

算法一（Schreier-Sims）

- 依次尝试加入每个排列 a_i，维护当前每层子问题对应的一组有效排列。“尝试加入”代表先判断是否有效再加入。
- 先考虑第一层：若 a_i 的加入使得 1 所在的连通块增大，则 a_i 有效排列，终止过程。否则任选第一层的从 $a_{i,1}$ 到 1 的一条路径，将它复合到 a_i 上。此时 $a_{i,1} = 1$，递归到第二层中继续判断。

LOJ177 生成子群阶数

算法一（Schreier-Sims）

- 依次尝试加入每个排列 a_i，维护当前每层子问题对应的一组有效排列。“尝试加入”代表先判断是否有效再加入。
- 先考虑第一层：若 a_i 的加入使得 1 所在的连通块增大，则 a_i 有效排列，终止过程。否则任选第一层的从 $a_{i,1}$ 到 1 的一条路径，将它复合到 a_i 上。此时 $a_{i,1}=1$，递归到第二层中继续判断。
- 若直到第 n 层依然未终止，意味着找到了用之前的排列表示出 a_i 的一组方案，a_i 必然无效，舍弃。

LOJ177 生成子群阶数

算法一（Schreier-Sims）

- 若 a_i 有效，则需要改变结构：将 a_i 加入第一层中，并将所有新产生的 C_e 尝试加入第二层中。对于每个成功加入到某一层的排列，继续向下一层进行类似的“尝试加入”操作。

LOJ177 生成子群阶数

算法一（Schreier-Sims）

- 若 a_i 有效，则需要改变结构：将 a_i 加入第一层中，并将所有新产生的 C_e 尝试加入第二层中。对于每个成功加入到某一层的排列，继续向下一层进行类似的“尝试加入”操作。
- 每层的有效排列个数为 $O(n)$，总计 $O(n^2)$，因此共进行 $O(n^3)$ 次“尝试加入”。判断一个排列是否有效的时间复杂度为 $O(n^2)$。总时间复杂度为 $O(n^5)$。

LOJ177 生成子群阶数

算法二（随机化）

- 考虑将 m' 个排列减少为 $O(n)$ 个有效排列。并没有非常直观的处理方式。
- 我们充分发扬人类智慧，随机取 $O(n)$ 个将原有排列相互复合的方案，并将它们当作一组有效排列即可。

LOJ177 生成子群阶数

算法二（随机化）

- 考虑将 m' 个排列减少为 $O(n)$ 个有效排列。并没有非常直观的处理方式。
- 我们充分发扬人类智慧，随机取 $O(n)$ 个将原有排列相互复合的方案，并将它们当作一组有效排列即可。
- 一种随机方式为：随机一个长度为 $O(m')$ 的序列，每个元素均为某个原有排列，并将它们依次复合。

LOJ177 生成子群阶数

算法二（随机化）

- 考虑将 m' 个排列减少为 $O(n)$ 个有效排列。并没有非常直观的处理方式。
- 我们充分发扬人类智慧，随机取 $O(n)$ 个将原有排列相互复合的方案，并将它们当作一组有效排列即可。
- 一种随机方式为：随机一个长度为 $O(m')$ 的序列，每个元素均为某个原有排列，并将它们依次复合。
- 时间复杂度依赖于随机方式在 $O(n^5)$ 左右不等。

LOJ177 生成子群阶数

算法二（随机化）

- 考虑将 m' 个排列减少为 $O(n)$ 个有效排列。并没有非常直观的处理方式。
- 我们充分发扬人类智慧，随机取 $O(n)$ 个将原有排列相互复合的方案，并将它们当作一组有效排列即可。
- 一种随机方式为：随机一个长度为 $O(m')$ 的序列，每个元素均为某个原有排列，并将它们依次复合。
- 时间复杂度依赖于随机方式在 $O(n^5)$ 左右不等。
- 在 LOJ177 的数据中，甚至只随机取 n 个方案即可通过。

CF1909I Short Permutation Problem

题目描述

给定 n，对于每一对 m, k 求出有多少个排列满足 $\sum_{i=1}^{n-1}[a_i + a_{i+1} \geq m] = k$。答案对 998244353 取模。

$n \leq 4 \times 10^3$。

CF1909I Short Permutation Problem

算法

- 对于每个 m 在 $O(n\log n)$ 的时间复杂度内求解。

CF1909I Short Permutation Problem

算法

- 对于每个 m 在 $O(n\log n)$ 的时间复杂度内求解。
- 考虑 $k=n-1$ 的情况。将 $\geq \frac{m}{2}$ 和 $< \frac{m}{2}$ 的数分开考虑。

CF1909I Short Permutation Problem

算法

- 对于每个 m 在 $O(n\log n)$ 的时间复杂度内求解。
- 考虑 $k=n-1$ 的情况。将 $\geq \frac{m}{2}$ 和 $< \frac{m}{2}$ 的数分开考虑。
- 将所有数按照 $\min(i, m-i)$ 从小到大排序。则两个数和 $\geq m$ 当且仅当靠前的一个 $\geq \frac{m}{2}$。

CF1909I Short Permutation Problem

算法

- 对于每个 m 在 $O(n\log n)$ 的时间复杂度内求解。
- 考虑 $k=n-1$ 的情况。将 $\geq \frac{m}{2}$ 和 $<\frac{m}{2}$ 的数分开考虑。
- 将所有数按照 $\min(i,m-i)$ 从小到大排序。则两个数和 $\geq m$ 当且仅当靠前的一个 $\geq \frac{m}{2}$。
- 初始有 $t=1$ 个可选空位，每加入一个 $\geq \frac{m}{2}$ 的数时 t 增大 1，反之 t 减小 1。方案数即为过程中所有 t 的乘积。

CF1909I Short Permutation Problem

算法

- 对于每个 k 计算钦定 k 对满足条件的方案数，二项式反演即可得到恰好 k 对的方案数。

CF1909I Short Permutation Problem

算法

- 对于每个 k 计算钦定 k 对满足条件的方案数，二项式反演即可得到恰好 k 对的方案数。
- 钦定 k 对满足条件，相当于划分为 $n-k$ 段，每段中任意相邻两个数均满足条件。

CF1909I Short Permutation Problem

算法

- 对于每个 k 计算钦定 k 对满足条件的方案数，二项式反演即可得到恰好 k 对的方案数。
- 钦定 k 对满足条件，相当于划分为 $n-k$ 段，每段中任意相邻两个数均满足条件。
- 将初始值 $t=1$ 改为 $t=n-k$ 即可。可能会出现一些链为空的情况，再次容斥即可。

CF1909I Short Permutation Problem

算法

- 对于每个 k 计算钦定 k 对满足条件的方案数，二项式反演即可得到恰好 k 对的方案数。
- 钦定 k 对满足条件，相当于划分为 $n-k$ 段，每段中任意相邻两个数均满足条件。
- 将初始值 $t=1$ 改为 $t=n-k$ 即可。可能会出现一些链为空的情况，再次容斥即可。
- 现在只需对于每个 k 求出初始值为 $t=k$ 时的乘积。观察 t 变化的方式即可做到 $O(1)$ 或 $O(\log n)$ 计算单项。

CF1909I Short Permutation Problem

算法

- 对于每个 k 计算钦定 k 对满足条件的方案数，二项式反演即可得到恰好 k 对的方案数。
- 钦定 k 对满足条件，相当于划分为 $n-k$ 段，每段中任意相邻两个数均满足条件。
- 将初始值 $t=1$ 改为 $t=n-k$ 即可。可能会出现一些链为空的情况，再次容斥即可。
- 现在只需对于每个 k 求出初始值为 $t=k$ 时的乘积。观察 t 变化的方式即可做到 $O(1)$ 或 $O(\log n)$ 计算单项。
- 二项式反演需要一次卷积。因此对于一个 m 计算的时间复杂度为 $O(n\log n)$。总时间复杂度 $O(n^2\log n)$。

CF1896H Cyclic Hamming

题目描述

给定 n，对于两个长度相等的序列 a, b，定义 $f(a,b) = \sum[a_i \neq b_i]$。
定义一个序列二元组 (a, b) 为好的当且仅当满足以下条件：

- a, b 均为长度为 2^{n+1} 的 01 序列，且均恰好包含 2^n 个 1。
- 对于 b 的每个循环移位 b'，均有 $f(a, b') = 2^n$。

现在给定两个包含 $0, 1, ?$ 的序列，求有多少种将所有 $?$ 替换为 $0, 1$ 的方案使得它们组成一个好的序列二元组。答案对 998244353 取模。
$n \leq 7$。
加强：$n \leq 12$。

CF1896H Cyclic Hamming

算法

- 令 $A(x)=\sum_{i=0}^{2^{n+1}-1} a_i x^i, B(x)=\sum_{i=0}^{2^{n+1}-1} b_{2^{n+1}-i-1} x^i, I(x)=\sum_{i=0}^{2^{n+1}-1} x^i$。将它们看作 $x^{2^{n+1}}-1$ 的多项式。则 (a,b) 为好的当且仅当 $A(I-B)+(I-A)B=2^n I$。稍作变形可得 $AB=2^{n-1}I$。

CF1896H Cyclic Hamming

算法

- 令 $A(x)=\sum_{i=0}^{2^{n+1}-1}a_ix^i,B(x)=\sum_{i=0}^{2^{n+1}-1}b_{2^{n+1}-i-1}x^i,I(x)=\sum_{i=0}^{2^{n+1}-1}x^i$。将它们看作 $x^{2^{n+1}}-1$ 的多项式。则 (a,b) 为好的当且仅当 $A(I-B)+(I-A)B=2^nI$。稍作变形可得 $AB=2^{n-1}I$。
- 两边 FFT。相当于要求 $\forall 1\le i<2^{n+1}$，有 $A(\omega^i)B(\omega^i)=0$。

CF1896H Cyclic Hamming

算法

- 令 $A(x)=\sum_{i=0}^{2^{n+1}-1} a_i x^i, B(x)=\sum_{i=0}^{2^{n+1}-1} b_{2^{n+1}-i-1} x^i, I(x)=\sum_{i=0}^{2^{n+1}-1} x^i$。将它们看作 $x^{2^{n+1}}-1$ 的多项式。则 (a,b) 为好的当且仅当 $A(I-B)+(I-A)B=2^n I$。稍作变形可得 $AB=2^{n-1}I$。
- 两边 FFT。相当于要求 $\forall 1\le i<2^{n+1}$，有 $A(\omega^i)B(\omega^i)=0$。
- 令 ω 为 2^{n+1} 次单位根。由分圆多项式可知，$1,\omega,\omega^2,\dots,\omega^{2^{n+1}-1}$ 的线性组合为 0 当且仅当 ω^i 与 ω^{2^n+i} 的系数一致。

CF1896H Cyclic Hamming

算法

- 令 $A(x)=\sum_{i=0}^{2^{n+1}-1} a_i x^i, B(x)=\sum_{i=0}^{2^{n+1}-1} b_{2^{n+1}-i-1} x^i, I(x)=\sum_{i=0}^{2^{n+1}-1} x^i$。将它们看作 $x^{2^{n+1}}-1$ 的多项式。则 (a,b) 为好的当且仅当 $A(I-B)+(I-A)B=2^nI$。稍作变形可得 $AB=2^{n-1}I$。
- 两边 FFT。相当于要求 $\forall 1\le i<2^{n+1}$，有 $A(\omega^i)B(\omega^i)=0$。
- 令 ω 为 2^{n+1} 次单位根。由分圆多项式可知，$1,\omega,\omega^2,\dots,\omega^{2^{n+1}-1}$ 的线性组合为 0 当且仅当 ω^i 与 ω^{2^n+i} 的系数一致。
- 因此 $A(\omega^i)$ 是否为 0 只与 $\gcd(i,2^{n+1})$ 有关。只需考虑每个 $A(\omega^{2^i})$ 是否为 0!

CF1896H Cyclic Hamming

算法

- 钦定一个子集 S，要求 $\forall i \in S$，有 $A(\omega^{2^i}) = 0$。ω^{2^i} 相当于是将所有 $\bmod 2^{n-i+1}$ 相同的数叠加起来。

CF1896H Cyclic Hamming

算法

- 钦定一个子集 S，要求 $\forall i \in S$，有 $A(\omega^{2^i}) = 0$。ω^{2^i} 相当于是将所有 $\bmod 2^{n-i+1}$ 相同的数叠加起来。
- 因此令 $dp_{i,j,k}$ 表示 $\bmod 2^{n-i+1} = j$ 的数和为 k，且符合 S 的限制的方案数。S 的限制恰好可以在转移过程中刻画。

CF1896H Cyclic Hamming

算法

- 钦定一个子集 S，要求 $\forall i \in S$，有 $A(\omega^{2^i}) = 0$。ω^{2^i} 相当于是将所有 $\bmod 2^{n-i+1}$ 相同的数叠加起来。
- 因此令 $dp_{i,j,k}$ 表示 $\bmod 2^{n-i+1} = j$ 的数和为 k，且符合 S 的限制的方案数。S 的限制恰好可以在转移过程中刻画。
- 对于一个 S 计算的时间复杂度为 $O(4^n)$，总时间复杂度为 $O(8^n)$。可以通过 $n \le 7$。

CF1896H Cyclic Hamming

算法

- 钦定一个子集 S，要求 $\forall i \in S$，有 $A(\omega^{2^i}) = 0$。ω^{2^i} 相当于是将所有 $\bmod 2^{n-i+1}$ 相同的数叠加起来。
- 因此令 $dp_{i,j,k}$ 表示 $\bmod 2^{n-i+1} = j$ 的数和为 k，且符合 S 的限制的方案数。S 的限制恰好可以在转移过程中刻画。
- 对于一个 S 计算的时间复杂度为 $O(4^n)$，总时间复杂度为 $O(8^n)$。可以通过 $n \leq 7$。
- 加强版只需精细地剪去无用状态，以及用 FFT 优化转移中的卷积即可通过。

CF1774G Segment Covering

题目描述

给定 n 个区间 $[l_i, r_i]$。令 $f(l,r)$ 表示从 n 个区间中选择偶数个区间使得它们并集恰为 $[l,r]$ 的方案数，令 $g(l,r)$ 表示选择奇数个区间的方案数。共 m 次询问，每次给定 l,r，求 $f(l,r)-g(l,r)$。答案对 998244353 取模。

$n,m \le 2\times 10^5$。

CF1774G Segment Covering

算法

- 容斥，钦定若干个关键点未被覆盖。若一个区间包含至少一个关键点，则它不可选，否则它可选可不选。对于一个可选可不选的区间，它对答案的贡献为 $1+(-1)=0$。因此可以看作要求每个区间都包含至少一个关键点。

CF1774G Segment Covering

算法

- 容斥，钦定若干个关键点未被覆盖。若一个区间包含至少一个关键点，则它不可选，否则它可选可不选。对于一个可选可不选的区间，它对答案的贡献为 $1+(-1)=0$。因此可以看作要求每个区间都包含至少一个关键点。
- 令 p_i 表示最小的位置使得 (p_i,i) 不包含任何区间，dp_i 表示最后一个关键点为 i 的权值和。则有 $dp_i=-\sum\limits_{j=p_i}^{i-1}dp_j$。初值为 $dp_{l-1}=1$，答案为 dp_{r+1}。

CF1774G Segment Covering

算法

- 容斥，钦定若干个关键点未被覆盖。若一个区间包含至少一个关键点，则它不可选，否则它可选可不选。对于一个可选可不选的区间，它对答案的贡献为 $1+(-1)=0$。因此可以看作要求每个区间都包含至少一个关键点。
- 令 p_i 表示最小的位置使得 (p_i,i) 不包含任何区间，dp_i 表示最后一个关键点为 i 的权值和。则有 $dp_i=-\sum\limits_{j=p_i}^{i-1}dp_j$。初值为 $dp_{l-1}=1$，答案为 dp_{r+1}。
- 令 $s_i=\sum\limits_{j=l-1}^{i}dp_j$。则有 $s_i-s_{i-1}=-(s_{i-1}-s_{p_i-1})$，即 $s_i=s_{p_i-1}$。初值为 $s_{l-2}=0,s_{l-1}=1$，答案为 $s_{r+1}-s_r$。

CF1774G Segment Covering

算法

- 容斥，钦定若干个关键点未被覆盖。若一个区间包含至少一个关键点，则它不可选，否则它可选可不选。对于一个可选可不选的区间，它对答案的贡献为 $1+(-1)=0$。因此可以看作要求每个区间都包含至少一个关键点。
- 令 p_i 表示最小的位置使得 (p_i, i) 不包含任何区间，dp_i 表示最后一个关键点为 i 的权值和。则有 $dp_i = -\sum_{j=p_i}^{i-1} dp_j$。初值为 $dp_{l-1}=1$，答案为 dp_{r+1}。
- 令 $s_i = \sum_{j=l-1}^{i} dp_j$。则有 $s_i - s_{i-1} = -(s_{i-1} - s_{p_i-1})$，即 $s_i = s_{p_i-1}$。初值为 $s_{l-2}=0, s_{l-1}=1$，答案为 $s_{r+1}-s_r$。
- 倍增优化即可做到 $O(n\log n)$。

CF1889E Doremy's Swapping Trees

题目描述

给定两个 n 个点的树 T_1, T_2。你可以执行任意多次如下操作：

- 从 T_1, T_2 中分别选择一个边集 E_1, E_2，满足：
  - $\forall u \neq v$，u, v 只经过 E_1 中的边连通当且仅当 u, v 只经过 E_2 中的边连通。
- 将 T_1 中的边集 E_1 与 T_2 中的边集 E_2 交换。

求共能得到多少种不同的 T_1。答案对 $10^9 + 7$ 取模。

$n \leq 2 \times 10^5$。

CF1889E Doremy's Swapping Trees

算法

- 对于 $e_1 \in T_1, e_2 \in T_2$ 满足 T_2 中 e_1 两个端点之间的路径经过 e_2。则有限制：若 $e_1 \in E_1$，则 $e_2 \in E_2$。反之亦然。

CF1889E Doremy's Swapping Trees

算法

- 对于 $e_1 \in T_1, e_2 \in T_2$ 满足 T_2 中 e_1 两个端点之间的路径经过 e_2。则有限制：若 $e_1 \in E_1$，则 $e_2 \in E_2$。反之亦然。
- 进一步地，E_1, E_2 合法当且仅当满足上述条件。

CF1889E Doremy's Swapping Trees

算法

- 对于 $e_1 \in T_1, e_2 \in T_2$ 满足 T_2 中 e_1 两个端点之间的路径经过 e_2。则有限制：若 $e_1 \in E_1$，则 $e_2 \in E_2$。反之亦然。
- 进一步地，E_1, E_2 合法当且仅当满足上述条件。
- 建立有向图 G，G 的每个点对应 T_1 或 T_2 中的一条边。上述限制在 G 中表示为 $e_1 \to e_2$ 的边。同时每个点有一个 $0/1$ 标记表示它当前是否在 T_1 中。

CF1889E Doremy's Swapping Trees

算法

- 操作相当于选择一个传递闭包，将其中所有点的标记取反，并重新建图。实际上 G 的结构并不会改变。

CF1889E Doremy's Swapping Trees

算法

- 操作相当于选择一个传递闭包，将其中所有点的标记取反，并重新建图。实际上 G 的结构并不会改变。
- G 中每个 SCC 的标记一定同时变化。正常情况下，每个 SCC 应当贡献 2 种方案，但可能有一些 SCC 在 T_1, T_2 中边集完全一致，此时它只会贡献 1 种方案。求出 SCC 后暴力计算即可。

CF1889E Doremy's Swapping Trees

算法

- 操作相当于选择一个传递闭包，将其中所有点的标记取反，并重新建图。实际上 G 的结构并不会改变。
- G 中每个 SCC 的标记一定同时变化。正常情况下，每个 SCC 应当贡献 2 种方案，但可能有一些 SCC 在 T_1, T_2 中边集完全一致，此时它只会贡献 1 种方案。求出 SCC 后暴力计算即可。
- 使用你喜欢的数据结构优化建图即可。一种方法是倍增优化建图，时间复杂度为 $O(n\log n)$。

QOJ8049 Equal Sums

题目描述

有两个整数序列 a, b，长度分别为 n, m。a_i 在 $[la_i, ra_i]$ 中，b_i 在 $[lb_i, rb_i]$ 中。对于每一对 i, j 求出 $\sum_{k=1}^{i} a_k = \sum_{k=1}^{i} b_k$ 的方案数。答案对 998244353 取模。
$n, m, la_i, ra_i, lb_i, rb_i \le 500$。

QOJ8049 Equal Sums

算法

- 令 $dp_{i,j,x}$ 表示考虑 $a_{1\dots i}, b_{1\dots j}$，$\sum\limits_{k=1}^{i} a_k - \sum\limits_{k=1}^{i} b_k = x$ 的方案数。

QOJ8049 Equal Sums

算法

- 令 $dp_{i,j,x}$ 表示考虑 $a_{1\dots i}, b_{1\dots j}$，$\sum_{k=1}^{i} a_k - \sum_{k=1}^{i} b_k = x$ 的方案数。
- 若 $x < 0$ 则加入 a_{i+1}，否则加入 b_{j+1}。

QOJ8049 Equal Sums

算法

- 令 $dp_{i,j,x}$ 表示考虑 $a_{1\dots i}, b_{1\dots j}$，$\sum_{k=1}^{i} a_k - \sum_{k=1}^{i} b_k = x$ 的方案数。
- 若 $x < 0$ 则加入 a_{i+1}，否则加入 b_{j+1}。
- 始终有 $|x| \le V$。转移可以前缀和优化，总复杂度为 $O(n^2 V)$。

QOJ5016 Range Minimum Element

题目描述

有一个长度为 n，值域为 $[1,c]$ 的正整数序列 a。给定 m 个区间 $[l_i,r_i]$，设长度为 m 的序列 b 满足 $\forall i\in[1,m],b_i=\min\limits_{l_i\le j\le r_i} a_j$。求 a 在范围内任取的情况下共能得到多少种不同的 b。答案对 998244353 取模。

$n\le 100$。

QOJ5016 Range Minimum Element

算法

- 考虑如何判断一组 b 是否合法。贪心地，令 $a_i = \max\limits_{l_j \le i \le r_j} b_j$。则合法当且仅当上述 a 生成 b。只需计数上述方式能够生成多少种不同的 a。

QOJ5016 Range Minimum Element

算法

- 考虑如何判断一组 b 是否合法。贪心地，令 $a_i = \max_{l_j \le i \le r_j} b_j$。则合法当且仅当上述 a 生成 b。只需计数上述方式能够生成多少种不同的 a。
- 从 1 开始填。考虑一个极大的区间 $[l, r]$ 满足 $a_l \dots a_r > 1$。则要求 $[l, r]$ 能表示为若干个 $[l_i, r_i]$ 的并。> 1 的部分为每个划分出的 $[l, r]$ 中的子问题，区间 dp 即可。

QOJ5016 Range Minimum Element

算法

- 考虑如何判断一组 b 是否合法。贪心地，令 $a_i = \max_{l_j \le i \le r_j} b_j$。则合法当且仅当上述 a 生成 b。只需计数上述方式能够生成多少种不同的 a。
- 从 1 开始填。考虑一个极大的区间 $[l, r]$ 满足 $a_l \dots a_r > 1$。则要求 $[l, r]$ 能表示为若干个 $[l_i, r_i]$ 的并。> 1 的部分为每个划分出的 $[l, r]$ 中的子问题，区间 dp 即可。
- 对 c 这一维插值即可做到 $O(n^4)$。

QOJ7759 Permutation Counting 2

题目描述

给定 n，对于每组 $x,y\in[0,n)$ 求出有多少个 $1\sim n$ 的排列 p 满足以下条件：

- $\sum_{i=1}^{n-1}[p_i<p_{i+1}]=x$。
- $\sum_{i=1}^{n-1}[p_i^{-1}<p_{i+1}^{-1}]=y$。

其中 p^{-1} 表示 p 的逆排列，满足 $p_{p_i}^{-1}=i$。

答案对给定的质数 MOD 取模。

$n\leq 500$。

QOJ7759 Permutation Counting 2

算法

- 分别钦定 p, p^{-1} 中的 i, j 个上升，计算其方案数 $f_{i,j}$。对两维分别进行 $O(n^3)$ 的容斥即可得到答案。

QOJ7759 Permutation Counting 2

算法

- 分别钦定 p, p^{-1} 中的 i, j 个上升，计算其方案数 $f_{i,j}$。对两维分别进行 $O(n^3)$ 的容斥即可得到答案。
- i 个上升相当于 $n-i$ 个非空上升段。依次加入 p^{-1} 中的每个上升段，类似于从小到大地将每个数填入 p。任意时刻 p 的每个上升段中已经被填入的应当是一段前缀。

QOJ7759 Permutation Counting 2

算法

- 分别钦定 p, p^{-1} 中的 i, j 个上升，计算其方案数 $f_{i,j}$。对两维分别进行 $O(n^3)$ 的容斥即可得到答案。
- i 个上升相当于 $n-i$ 个非空上升段。依次加入 p^{-1} 中的每个上升段，类似于从小到大地将每个数填入 p。任意时刻 p 的每个上升段中已经被填入的应当是一段前缀。
- 令 $a_{k,l}$ 表示加入 p^{-1} 中的第 k 上升段时在 p 中的第 l 个上升段中填入了 $a_{k,l}$ 个数。如果确定了所有 $a_{k,l}(k \in [1, n-i], l \in [1, n-j])$ 则可以唯一确定 p。

QOJ7759 Permutation Counting 2

算法

- 对 $a_{k,l}$ 的限制为：总和为 n，并且不能存在一行或一列全为 0，否则会导致 p 或 p^{-1} 中某个上升段为空。

QOJ7759 Permutation Counting 2

算法

- 对 $a_{k,l}$ 的限制为：总和为 n，并且不能存在一行或一列全为 0，否则会导致 p 或 p^{-1} 中某个上升段为空。
- 对这个限制再进行一次容斥，即钦定部分行列全为 0。依然对两维分别进行 $O(n^3)$ 的容斥即可得到答案。

QOJ8047 DFS Order 4

题目描述

求有多少个 n 阶排列 p 使得至少存在一个满足以下条件的有根树:

- $\forall 2 \le i \le n, fa_i < i$。
- 若 dfs 的过程中按照编号从小到大的顺序访问一个点的所有儿子，则得到的 dfs 序为 p。

答案对给定的质数 P 取模。

$n \le 800$。

QOJ8047 DFS Order 4

算法

- 考虑如何判断 p 是否合法。

QOJ8047 DFS Order 4

算法

- 考虑如何判断 p 是否合法。
- 令 p_{i-1} 的父链从上往下依次设为 $anc_{1\dots m}$。若 $p_{i-1} < p_i$ 则可以直接将 p_i 接到 p_{i-1} 下方。否则选择最大的 k 使得 $anc_k < p_i$ 并将 p_i 接到 anc_{k-1} 下方。

QOJ8047 DFS Order 4

算法

- 考虑如何判断 p 是否合法。
- 令 p_{i-1} 的父链从上往下依次设为 $anc_{1\dots m}$。若 $p_{i-1} < p_i$ 则可以直接将 p_i 接到 p_{i-1} 下方。否则选择最大的 k 使得 $anc_k < p_i$ 并将 p_i 接到 anc_{k-1} 下方。
- 考虑对这样贪心连出的树计数。等价于以下限制：
  - 令某个点的儿子为 $u_1 \dots u_m$，u_i 的最后一个儿子为 v_i。则 $u_{i-1} < u_i < v_{i-1}$。

QOJ8047 DFS Order 4

算法

- 将问题看作在一个有向图上进行拓扑序计数。而 $u_i < v_{i-1}$ 的限制会导致这个图不是一棵外向树。

QOJ8047 DFS Order 4

算法

- 将问题看作在一个有向图上进行拓扑序计数。而 $u_i < v_{i-1}$ 的限制会导致这个图不是一棵外向树。
- 将它容斥为 $[u_i < v_{i-1}] = 1 - [u_i > v_{i-1}]$，即要么删除要么改成一条反向边。

QOJ8047 DFS Order 4

算法

- 将问题看作在一个有向图上进行拓扑序计数。而 $u_i < v_{i-1}$ 的限制会导致这个图不是一棵外向树。
- 将它容斥为 $[u_i < v_{i-1}] = 1 - [u_i > v_{i-1}]$，即要么删除要么改成一条反向边。
- 若改成了反向边，显然可以删除 $u_i > u_{i-1}$ 这条限制。

QOJ8047 DFS Order 4

算法

- 将问题看作在一个有向图上进行拓扑序计数。而 $u_i < v_{i-1}$ 的限制会导致这个图不是一棵外向树。
- 将它容斥为 $[u_i < v_{i-1}] = 1 - [u_i > v_{i-1}]$，即要么删除要么改成一条反向边。
- 若改成了反向边，显然可以删除 $u_i > u_{i-1}$ 这条限制。
- 这样操作之后当前这层的限制会形成一个链状结构，并且整体形成树形结构，考虑其形态可以设计出状态。

QOJ8047 DFS Order 4

算法

- 将问题看作在一个有向图上进行拓扑序计数。而 $u_i < v_{i-1}$ 的限制会导致这个图不是一棵外向树。
- 将它容斥为 $[u_i < v_{i-1}] = 1 - [u_i > v_{i-1}]$，即要么删除要么改成一条反向边。
- 若改成了反向边，显然可以删除 $u_i > u_{i-1}$ 这条限制。
- 这样操作之后当前这层的限制会形成一个链状结构，并且整体形成树形结构，考虑其形态可以设计出状态。
- 令 $dp_{i,j}$ 表示大小为 i 的树，在根节点的最后一个儿子 u_m 后面额外增加一个子树大小为 j 的儿子 u_{m+1}，并且要求 $u_m < u_{m+1}$ 时前面 i 个点对拓扑序的贡献。答案即为 $dp_{n,0}$。时间复杂度 $O(n^3)$。

Luogu P8923 Many Minimizations

题目描述

对于一个长度为 n 的序列。定义 $f(a)$ 表示 b 为单调不降序列时 $\sum |a_i - b_i|$ 的最小值。
给定 n, m，求所有长度为 n，值域为 $[1, m]$ 的正整数序列 a 对应的 $f(a)$ 之和。
$n \le 5 \times 10^3$。
加强：$n \le 5 \times 10^5, P = 998244353$。

Luogu P8923 Many Minimizations

算法

- 用维护凸函数的方法解决最优化问题。问题转化为：每次操作先加入两个 a_i 再删除最大值。求所有方案中最终剩余的所有数的和。

Luogu P8923 Many Minimizations

算法

- 用维护凸函数的方法解决最优化问题。问题转化为：每次操作先加入两个 a_i 再删除最大值。求所有方案中最终剩余的所有数的和。
- 拆开每个数的贡献。固定 x，将 $\leq x$ 的数设为 0，$> x$ 的数设为 1。只需求出每个 x 的答案和。

Luogu P8923 Many Minimizations

算法

- 用维护凸函数的方法解决最优化问题。问题转化为：每次操作先加入两个 a_i 再删除最大值。求所有方案中最终剩余的所有数的和。
- 拆开每个数的贡献。固定 x，将 $\le x$ 的数设为 0，$> x$ 的数设为 1。只需求出每个 x 的答案和。
- 观察堆的变化方式，相当于每次向右上或右下走一步，右上的方案数为 $m-x$，右下的方案数为 x。特殊地，若纵坐标为 0 且选择右下则水平向右走一步。从 $(0,0)$ 开始，最终走到 $(n,*)$。

Luogu P8923 Many Minimizations

算法

- 水平向右的情况难以处理，考虑通过变换将它去掉。

Luogu P8923 Many Minimizations

算法

- 水平向右的情况难以处理，考虑通过变换将它去掉。
- 若过程中有 k 步水平向右，则将起点改为 $(0,k)$，并不再允许水平向右的特殊情况。另外要求过程中至少一次触碰但从不跨越 $y=0$。

Luogu P8923 Many Minimizations

算法

- 水平向右的情况难以处理，考虑通过变换将它去掉。
- 若过程中有 k 步水平向右，则将起点改为 $(0,k)$，并不再允许水平向右的特殊情况。另外要求过程中至少一次触碰但从不跨越 $y=0$。
- 可以利用反射容斥得到式子：

$$\sum_{i\geq 0} x^i(m-x)^{n-i} \sum_{j\geq \max\{i,n-i\}} (j-i)\left(\binom{n}{j}-\binom{n}{j+1}\right)$$

- 可以用卷积求出这个多项式，然后将自然数幂和代入计算。时间复杂度 $O(n\log n)$。

UOJ813 反重：求熵

题目描述

有 n 个整数变量 $x_1 \dots x_n$ 满足 $x_i \in [0, c]$。给定 m 个限制，每个限制形如 $x_u - x_v \le w$。求这些变量有多少种合法的取值方案。答案对 998244353 取模。$n \le 8$。

UOJ813 反重：求熵

算法

- 增加一个恒为 0 的变量 x_0，用于处理每个数在 $[0,c]$ 的限制。

UOJ813 反重：求熵

算法

- 增加一个恒为 0 的变量 x_0，用于处理每个数在 $[0, c]$ 的限制。
- u, v 相同的限制，可以只保留最紧的一个。保留之后形如 $x_i - x_j \le w_{i,j}$。

UOJ813 反重：求熵

算法

- 增加一个恒为 0 的变量 x_0，用于处理每个数在 $[0,c]$ 的限制。
- u,v 相同的限制，可以只保留最紧的一个。保留之后形如 $x_i - x_j \le w_{i,j}$。
- 固定 $x_1 \dots x_{n-1}$，则对 x_n 的限制形如 $x_i - w_{i,n} \le x_n \le x_i + w_{n,i}$。

UOJ813 反重：求熵

算法

- 增加一个恒为 0 的变量 x_0，用于处理每个数在 $[0, c]$ 的限制。
- u, v 相同的限制，可以只保留最紧的一个。保留之后形如 $x_i - x_j \le w_{i,j}$。
- 固定 $x_1 \dots x_{n-1}$，则对 x_n 的限制形如 $x_i - w_{i,n} \le x_n \le x_i + w_{n,i}$。
- 取两端最紧的限制得到 $l \le x_n \le r$，其中 $l = \max(x_i - w_{i,n}), r = \min(x_i + w_{n,i})$。

UOJ813 反重：求熵

算法

- 增加一个恒为 0 的变量 x_0，用于处理每个数在 $[0,c]$ 的限制。
- u,v 相同的限制，可以只保留最紧的一个。保留之后形如 $x_i - x_j \le w_{i,j}$。
- 固定 $x_1 \dots x_{n-1}$，则对 x_n 的限制形如 $x_i - w_{i,n} \le x_n \le x_i + w_{n,i}$。
- 取两端最紧的限制得到 $l \le x_n \le r$，其中 $l = \max(x_i - w_{i,n}), r = \min(x_i + w_{n,i})$。
- 定义一组 $x_1 \dots x_{n-1}$ 的方案为好的当且仅当 x_n 至少有一种合法取值。

UOJ813 反重：求熵

算法

- 增加一个恒为 0 的变量 x_0，用于处理每个数在 $[0, c]$ 的限制。
- u, v 相同的限制，可以只保留最紧的一个。保留之后形如 $x_i - x_j \le w_{i,j}$。
- 固定 $x_1 \dots x_{n-1}$，则对 x_n 的限制形如 $x_i - w_{i,n} \le x_n \le x_i + w_{n,i}$。
- 取两端最紧的限制得到 $l \le x_n \le r$，其中 $l = \max(x_i - w_{i,n}), r = \min(x_i + w_{n,i})$。
- 定义一组 $x_1 \dots x_{n-1}$ 的方案为好的当且仅当 x_n 至少有一种合法取值。
- 相当于求所有好方案的 $r - l + 1$ 之和。由线性性将 l, r 的贡献分开算，不妨考虑 l 的贡献。

UOJ813 反重：求熵

算法

- 枚举 p 使得 $x_i - w_{i,n}$ 在 p 处取到最大。

UOJ813 反重：求熵

算法

- 枚举 p 使得 $x_i - w_{i,n}$ 在 p 处取到最大。
- 即求所有 $x_i - w_{i,n}$ 在 p 处取到最大值的好方案对应的 $x_p - w_{p,n}$ 之和。

UOJ813 反重：求熵

算法

- 枚举 p 使得 $x_i - w_{i,n}$ 在 p 处取到最大。
- 即求所有 $x_i - w_{i,n}$ 在 p 处取到最大值的好方案对应的 $x_p - w_{p,n}$ 之和。
- “$x_i - w_{i,n}$ 在 p 处取到最大值”，“好方案”这两个限制均可用形如 $x_u - x_v \leq w$ 的限制表出。因此“几乎”递归到了 $n-1$ 个变量的子问题。

UOJ813 反重：求熵

算法

- 枚举 p 使得 $x_i - w_{i,n}$ 在 p 处取到最大。
- 即求所有 $x_i - w_{i,n}$ 在 p 处取到最大值的好方案对应的 $x_p - w_{p,n}$ 之和。
- “$x_i - w_{i,n}$ 在 p 处取到最大值”，“好方案”这两个限制均可用形如 $x_u - x_v \le w$ 的限制表出。因此“几乎”递归到了 $n-1$ 个变量的子问题。
- 唯一的区别在于，原来每种方案贡献为 1，而现在每种方案贡献为 $x_p - w_{p,n}$。

UOJ813 反重：求熵

算法

- 因此我们需要扩展问题的形式，在每个点上增加权值函数 $f_i(x)$，每种方案贡献为 $\prod f_i(x_i)$。

UOJ813 反重：求熵

算法

- 因此我们需要扩展问题的形式，在每个点上增加权值函数 $f_i(x)$，每种方案贡献为 $\prod f_i(x_i)$。
- 归纳可得 $f_i(x)$ 始终为不超过 n 次的多项式，每次需要支持类似于点值前缀和的操作。

UOJ813 反重：求熵

算法

- 因此我们需要扩展问题的形式，在每个点上增加权值函数 $f_i(x)$，每种方案贡献为 $\prod f_i(x_i)$。
- 归纳可得 $f_i(x)$ 始终为不超过 n 次的多项式，每次需要支持类似于点值前缀和的操作。
- 总时间复杂度为 $O(n! \times 2^n \times \text{poly}(n))$。

AGC060F Spanning Trees of Interval Graph

题目描述

给定 n，对于每一对 $1 \le i \le j \le$ 给定 $a_{i,j}$。无向图 G 中有 $a_{i,j}$ 个编号为 (i,j) 的点。编号为 $(l_1,r_1),(l_2,r_2)$ 的两个点之间有边当且仅当 $[l_1,r_1]\bigcap[l_2,r_2] \neq \varnothing$。

求 G 的生成树个数。答案对 998244353 取模。

$n \le 400$。

AGC060F Spanning Trees of Interval Graph

算法

- 由 Matrix-Tree，答案即为 $\det(D-A)$。其中 D 为对角矩阵。

AGC060F Spanning Trees of Interval Graph

算法

- 由 Matrix-Tree，答案即为 $\det(D-A)$。其中 D 为对角矩阵。
- 令 $U_{(i,j),k} = k \in [i,j], V_{(i,j),k} = [k,k+1] \subseteq [i,j]$，则 $A = \begin{pmatrix} U \\ V \end{pmatrix} \begin{pmatrix} U^T & -V^T \end{pmatrix}$。

AGC060F Spanning Trees of Interval Graph

算法

- 由线性代数，$\det(I-AB)=\det(I-BA)$。因此有：

$$\det(D-A)=\det\left(D-\begin{pmatrix}U\\V\end{pmatrix}\begin{pmatrix}U^T & -V^T\end{pmatrix}\right)=$$

$$\det(D)\det\left(I-\begin{pmatrix}U\\V\end{pmatrix}\begin{pmatrix}U^T & -V^T\end{pmatrix}D^{-1}\right)=$$

$$\det(D)\det\left(I-\begin{pmatrix}U^T & -V^T\end{pmatrix}D^{-1}\begin{pmatrix}U\\V\end{pmatrix}\right)$$

AGC060F Spanning Trees of Interval Graph

算法

- 由线性代数，$\det(I-AB)=\det(I-BA)$。因此有：

$$\det(D-A)=\det\left(D-\begin{pmatrix}U\\V\end{pmatrix}\begin{pmatrix}U^T & -V^T\end{pmatrix}\right)=$$

$$\det(D)\det\left(I-\begin{pmatrix}U\\V\end{pmatrix}\begin{pmatrix}U^T & -V^T\end{pmatrix}D^{-1}\right)=$$

$$\det(D)\det\left(I-\begin{pmatrix}U^T & -V^T\end{pmatrix}D^{-1}\begin{pmatrix}U\\V\end{pmatrix}\right)$$

- 后者为 $2n-1$ 阶矩阵。$O(n^3)$ 计算其行列式即可。

AGC061F Perfect Strings

题目描述

定义一个 01 串 a 为好的，当且仅当 a 中 0 的个数为 n 的倍数，1 的个数为 m 的倍数。
给定 n, m，求有多少个 01 串 a 满足它为好的，且 a 的所有子串均不为好的。
$n, m \leq 40$。

AGC061F Perfect Strings

算法

- 令 x_i 表示 $a_{1\dots i}$ 中 0 的个数 $\bmod n$，y_i 表示 $a_{1\dots i}$ 中 1 的个数 $\bmod m$。

AGC061F Perfect Strings

算法

- 令 x_i 表示 $a_{1\dots i}$ 中 0 的个数 $\bmod n$，y_i 表示 $a_{1\dots i}$ 中 1 的个数 $\bmod m$。
- 则 a 合法当且仅当 $x_{|a|}=y_{|a|}=0$，且 $\forall 1\le i<j\le |a|$，有 $(x_i,y_i)\neq(x_j,y_j)$。

AGC061F Perfect Strings

算法

- 令 x_i 表示 $a_{1\dots i}$ 中 0 的个数 $\bmod n$，y_i 表示 $a_{1\dots i}$ 中 1 的个数 $\bmod m$。
- 则 a 合法当且仅当 $x_{|a|}=y_{|a|}=0$，且 $\forall 1\le i<j\le |a|$，有 $(x_i,y_i)\ne(x_j,y_j)$。
- 从 $(0,0)$ 开始，每次向右或者向上走一步，横纵坐标分别对 n,m 取模，问最终回到 $(0,0)$，且除了起点之外无重复点的方案数。

AGC061F Perfect Strings

算法

- 固定每次到达左下边界的点以及上一步的方向，则每次从右上边界走出网格的点以及方向均已确定。称这些点为关键点。

AGC061F Perfect Strings

算法

- 固定每次到达左下边界的点以及上一步的方向，则每次从右上边界走出网格的点以及方向均已确定。称这些点为关键点。
- 令左下边界的点关键点依次（从左上到右下）为 $a_1 \dots a_k$，右上边界的关键点依次为 $b_1 \dots b_k$。

AGC061F Perfect Strings

算法

- 固定每次到达左下边界的点以及上一步的方向，则每次从右上边界走出网格的点以及方向均已确定。称这些点为关键点。
- 令左下边界的点关键点依次（从左上到右下）为 $a_1 \dots a_k$，右上边界的关键点依次为 $b_1 \dots b_k$。
- 则方案一定形如：对于每个 i 选择一条 $a_i \to b_i$ 的路径，且互不相交。方案数可使用 LGV Lemma 计算。

AGC061F Perfect Strings

算法

- 固定每次到达左下边界的点以及上一步的方向，则每次从右上边界走出网格的点以及方向均已确定。称这些点为关键点。
- 令左下边界的点关键点依次（从左上到右下）为 $a_1 \dots a_k$，右上边界的关键点依次为 $b_1 \dots b_k$。
- 则方案一定形如：对于每个 i 选择一条 $a_i \to b_i$ 的路径，且互不相交。方案数可使用 LGV Lemma 计算。
- 但这样可能会形成多个环。恰好形成一个环的条件为 $\gcd(k_1, k_2) = 1$。其中 k_1, k_2 分别为上一步为向右、向下的左下边界关键点数量。

AGC061F

算法

- 问题转化为：给定 $n+m$ 阶矩阵 A，选择一个 $1\sim n+m$ 的子集 S，要求 $\gcd(k_1,k_2)=1$。其中 k_1 为 S 中 $\le n$ 的元素个数，k_2 为 S 中 $>n$ 的元素个数。S 对答案的贡献为 $\det(A_{S,S})$。(实际贡献可能相差一个来自 LGV Lemma 的正负号)

AGC061F

算法

- 问题转化为：给定 $n+m$ 阶矩阵 A，选择一个 $1\sim n+m$ 的子集 S，要求 $\gcd(k_1,k_2)=1$。其中 k_1 为 S 中 $\le n$ 的元素个数，k_2 为 S 中 $>n$ 的元素个数。S 对答案的贡献为 $\det(A_{S,S})$。(实际贡献可能相差一个来自 LGV Lemma 的正负号)
- 在对角线的前 n 个元素加上 x，后 m 个元素加上 y，将行列式表示为 x,y 的多项式。则 $x^{n-k_1}y^{m-k_2}$ 项的系数即为 k_1,k_2 对应的答案。

AGC061F

算法

- 问题转化为：给定 $n+m$ 阶矩阵 A，选择一个 $1\sim n+m$ 的子集 S，要求 $\gcd(k_1,k_2)=1$。其中 k_1 为 S 中 $\le n$ 的元素个数，k_2 为 S 中 $>n$ 的元素个数。S 对答案的贡献为 $\det(A_{S,S})$。（实际贡献可能相差一个来自 LGV Lemma 的正负号）
- 在对角线的前 n 个元素加上 x，后 m 个元素加上 y，将行列式表示为 x,y 的多项式。则 $x^{n-k_1}y^{m-k_2}$ 项的系数即为 k_1,k_2 对应的答案。
- 插值即可做到 $O(n^5)$。也可以只对 y 插值，并使用 $O(n^3)$ 的方法计算 $\det(A+Bx)$ 来做到 $O(n^4)$。

- 谢谢大家！