HSFZPC 2026（Ucup Version）题解

AK TALE 命题组

2026/06

A. Hi Young Guangzhou

给定一棵 n 个点的树和 m 个关键点。给每个点赋值，使得：

- m 个关键点的值分别为 $1 \sim m$；
- 每个点的值是周围所有点的值的众数（之一）。

$\sum n \leq 10^6$。

Author: Otomachi_Una

A. Hi Young Guangzhou

- 考虑让每个值在树上形成连通块。

A. Hi Young Guangzhou

- 考虑让每个值在树上形成连通块。
- 问题转化为每个点周围至少有一个与它值相同的点。

A. Hi Young Guangzhou

- 考虑让每个值在树上形成连通块。
- 问题转化为每个点周围至少有一个与它值相同的点。
- 贪心将关键点与周围点匹配。

A. Hi Young Guangzhou

- 考虑让每个值在树上形成连通块。
- 问题转化为每个点周围至少有一个与它值相同的点。
- 贪心将关键点与周围点匹配。
- 如果匹配成功，每次将未确定值的点的值赋为周围确定值的点即可。

A. Hi Young Guangzhou

- 考虑让每个值在树上形成连通块。
- 问题转化为每个点周围至少有一个与它值相同的点。
- 贪心将关键点与周围点匹配。
- 如果匹配成功，每次将未确定值的点的值赋为周围确定值的点即可。
- 注意 $n = 1$ 的边界情况。

A. Hi Young Guangzhou

- 考虑让每个值在树上形成连通块。
- 问题转化为每个点周围至少有一个与它值相同的点。
- 贪心将关键点与周围点匹配。
- 如果匹配成功，每次将未确定值的点的值赋为周围确定值的点即可。
- 注意 $n = 1$ 的边界情况。
- 时间复杂度 $O(n)$。

B. Welcome to Math Kingdom!

给定 p, q，构造 $\frac{a^3+b^3}{c^3+d^3} = \frac{p}{q}$。

$1 \le p, q \le 10^4$，需要保证 $1 \le a, b, c, d \le 10^7$。

Author: shinzanmono

B. Welcome to Math Kingdom!

- 当 $\frac{1}{2} < \frac{p}{q} < 2$ 时，取 $a = c = p + q$，$b = 2p - q$，$d = 2q - p$ 即可满足 $\frac{a^3+b^3}{c^3+d^3} = \frac{p}{q}$。

B. Welcome to Math Kingdom!

- 当 $\frac{1}{2} < \frac{p}{q} < 2$ 时，取 $a = c = p + q$，$b = 2p - q$，$d = 2q - p$ 即可满足 $\frac{a^3+b^3}{c^3+d^3} = \frac{p}{q}$。
- 不妨设 $p < q$，将 p 每次 $\times 8$ 直到大于 q。

B. Welcome to Math Kingdom!

- 当 $\frac{1}{2} < \frac{p}{q} < 2$ 时，取 $a = c = p + q$，$b = 2p - q$，$d = 2q - p$ 即可满足 $\frac{a^3+b^3}{c^3+d^3} = \frac{p}{q}$。
- 不妨设 $p < q$，将 p 每次 $\times 8$ 直到大于 q。
- 此时如果不符合 $\frac{1}{2} < \frac{p}{q} < 2$ 的条件则将 p 乘 8，将 q 乘 27，不难发现此时符合条件。

B. Welcome to Math Kingdom!

- 当 $\frac{1}{2} < \frac{p}{q} < 2$ 时，取 $a = c = p + q$，$b = 2p - q$，$d = 2q - p$ 即可满足 $\frac{a^3+b^3}{c^3+d^3} = \frac{p}{q}$。
- 不妨设 $p < q$，将 p 每次 $\times 8$ 直到大于 q。
- 此时如果不符合 $\frac{1}{2} < \frac{p}{q} < 2$ 的条件则将 p 乘 8，将 q 乘 27，不难发现此时符合条件。
- 以上操作过程中同时维护 a, b, c, d 的系数。

B. Welcome to Math Kingdom!

- 当 $\frac{1}{2} < \frac{p}{q} < 2$ 时，取 $a = c = p + q$，$b = 2p - q$，$d = 2q - p$ 即可满足 $\frac{a^3+b^3}{c^3+d^3} = \frac{p}{q}$。
- 不妨设 $p < q$，将 p 每次 $\times 8$ 直到大于 q。
- 此时如果不符合 $\frac{1}{2} < \frac{p}{q} < 2$ 的条件则将 p 乘 8，将 q 乘 27，不难发现此时符合条件。
- 以上操作过程中同时维护 a, b, c, d 的系数。
- 时间复杂度 $O(T \log V)$。

B. Welcome to Math Kingdom!

- 当 $\frac{1}{2} < \frac{p}{q} < 2$ 时，取 $a = c = p + q$，$b = 2p - q$，$d = 2q - p$ 即可满足 $\frac{a^3+b^3}{c^3+d^3} = \frac{p}{q}$。
- 不妨设 $p < q$，将 p 每次 $\times 8$ 直到大于 q。
- 此时如果不符合 $\frac{1}{2} < \frac{p}{q} < 2$ 的条件则将 p 乘 8，将 q 乘 27，不难发现此时符合条件。
- 以上操作过程中同时维护 a, b, c, d 的系数。
- 时间复杂度 $O(T \log V)$。
- Bonus：构造 $1 \le a, b, c, d \le 10^6$ 并证明。

C. Shiroi Album II

定义一个序列为三角序列当且仅当其中任意三个元素均构成三角形。
给定一张 n 个点 m 条边的带权无向图，求两两点对之间是否存在路径使得其中所有边权构成的序列为三角序列。
$\sum n^2, \sum m^2 \leq 3000^2$。
Author: Otomachi_Una

C. Shiroi Album II

- 枚举权值最大的边，要求最小的边与第二小的边和大于权值最大的边。

C. Shiroi Album II

- 枚举权值最大的边，要求最小的边与第二小的边和大于权值最大的边。
- 双指针，问题转化为 $O(n^2)$ 次连边与更新两个连通块所有点之间的答案的操作。

C. Shiroi Album II

- 枚举权值最大的边，要求最小的边与第二小的边和大于权值最大的边。
- 双指针，问题转化为 $O(n^2)$ 次连边与更新两个连通块所有点之间的答案的操作。
- 连边维护连通块用并查集维护，建出合并树。

C. Shiroi Album II

- 枚举权值最大的边，要求最小的边与第二小的边和大于权值最大的边。
- 双指针，问题转化为 $O(n^2)$ 次连边与更新两个连通块所有点之间的答案的操作。
- 连边维护连通块用并查集维护，建出合并树。
- 更新连通块所有点之间答案被转化为更新两个子树所有点之间答案。

C. Shiroi Album II

- 枚举权值最大的边，要求最小的边与第二小的边和大于权值最大的边。
- 双指针，问题转化为 $O(n^2)$ 次连边与更新两个连通块所有点之间的答案的操作。
- 连边维护连通块用并查集维护，建出合并树。
- 更新连通块所有点之间答案被转化为更新两个子树所有点之间答案。
- 使用 bitset 维护，在合并树上跑一遍前缀 or 即可。

C. Shiroi Album II

- 枚举权值最大的边，要求最小的边与第二小的边和大于权值最大的边。
- 双指针，问题转化为 $O(n^2)$ 次连边与更新两个连通块所有点之间的答案的操作。
- 连边维护连通块用并查集维护，建出合并树。
- 更新连通块所有点之间答案被转化为更新两个子树所有点之间答案。
- 使用 bitset 维护，在合并树上跑一遍前缀 or 即可。
- 时间复杂度 $O(\frac{n^3}{w} + n^2\alpha(n))$。

D. Judgment of Mahou Shoujo

有 n 个人排成一个环，其中一些为魔女。每一天晚上，都有一个魔女使相邻的两个人中的一个出局。每次给出当天晚上出局的人的编号，并根据当前信息求出魔女的最小数量。

$\sum n \leq 10^6$。

Author: Otomachi_Una

D. Judgment of Mahou Shoujo

- 考虑 $dp_{i,0/1,0/1}$，表示 i 到下一个未出局的人 i' 之间，i 是否为魔女，i' 是否为魔女的情况下这一段中至少有多少个魔女。

D. Judgment of Mahou Shoujo

- 考虑 $dp_{i,0/1,0/1}$，表示 i 到下一个未出局的人 i' 之间，i 是否为魔女，i' 是否为魔女的情况下这一段中至少有多少个魔女。
- 一个人出局时，将上一个到他和他到下一个的 dp 值合并起来，并将新的 $dp_{i_0,0,0}$ 赋值为 $\inf$（因为这样的话这个人是不可能出局的）。

D. Judgment of Mahou Shoujo

- 考虑 $dp_{i,0/1,0/1}$，表示 i 到下一个未出局的人 i' 之间，i 是否为魔女，i' 是否为魔女的情况下这一段中至少有多少个魔女。
- 一个人出局时，将上一个到他和他到下一个的 dp 值合并起来，并将新的 $dp_{i_0,0,0}$ 赋值为 $\inf$（因为这样的话这个人是不可能出局的）。
- 将 dp 的转移看作矩阵，使用线段树维护，即可求出全局魔女的最小数量。

D. Judgment of Mahou Shoujo

- 考虑 $dp_{i,0/1,0/1}$，表示 i 到下一个未出局的人 i' 之间，i 是否为魔女，i' 是否为魔女的情况下这一段中至少有多少个魔女。
- 一个人出局时，将上一个到他和他到下一个的 dp 值合并起来，并将新的 $dp_{i_0,0,0}$ 赋值为 $\inf$（因为这样的话这个人是不可能出局的）。
- 将 dp 的转移看作矩阵，使用线段树维护，即可求出全局魔女的最小数量。
- 时间复杂度 $O(n \log n)$。

E. Hoshizora

给定 n 个物体，第 i 个物体有质量 $\frac{1}{m_i}$，在 0 时刻在 (x_i, y_i)，在 ϵ 时刻在 (x'_i, y'_i)，并以极快的速度匀速直线运动。对于每个物体，求所有时刻受到的引力的最大值。不考虑引力对物体轨迹的影响。你只需要输出其倒数乘 G 后的值即可。

$n \leq 10^5$，$m_i \leq 100$，$|x_i - x'_i|, |y_i - y'_i| \leq 5$，$|x_i|, |y_i| \leq 10^8$。

Author: Otomachi_Una

E. Hoshizora

记 $V_c = 5$，$V_m = 100$，$V = 10^8$。

- 本质不同速度的物体只有 $O(V_c^2)$ 种。考虑称速度相同的物体为一类，分开计算同类和不同类之间的答案，最后取 $\min$ 即可。

E. Hoshizora

记 $V_c = 5$，$V_m = 100$，$V = 10^8$。

- 本质不同速度的物体只有 $O(V_c^2)$ 种。考虑称速度相同的物体为一类，分开计算同类和不同类之间的答案，最后取 $\min$ 即可。
- 对于不同类的，考虑先枚举两类，计算他们之间的答案。

E. Hoshizora

记 $V_c = 5$，$V_m = 100$，$V = 10^8$。

- 本质不同速度的物体只有 $O(V_c^2)$ 种。考虑称速度相同的物体为一类，分开计算同类和不同类之间的答案，最后取 $\min$ 即可。
- 对于不同类的，考虑先枚举两类，计算他们之间的答案。
- 由于我们只在乎某两种速度的物体，考虑切换参照系，设其中我们希望计算答案的那一类物体为不动的。

E. Hoshizora

记 $V_c = 5$，$V_m = 100$，$V = 10^8$。

- 本质不同速度的物体只有 $O(V_c^2)$ 种。考虑称速度相同的物体为一类，分开计算同类和不同类之间的答案，最后取 $\min$ 即可。
- 对于不同类的，考虑先枚举两类，计算他们之间的答案。
- 由于我们只在乎某两种速度的物体，考虑切换参照系，设其中我们希望计算答案的那一类物体为不动的。
- 此时，另一类物体的速度在该参照系下的速度应当减去原始参照系中我们希望计算答案的那一类物体的速度。

E. Hoshizora

- 不难发现另一类物体的运动轨迹是一条直线。

E. Hoshizora

- 不难发现另一类物体的运动轨迹是一条直线。
- 找出垂直于这些直线的直线，并将坐标系旋转。

E. Hoshizora

- 不难发现另一类物体的运动轨迹是一条直线。
- 找出垂直于这些直线的直线，并将坐标系旋转。
- 考虑如何计算答案。由于 m_i 的影响，直接找距离最近的直线是错的。

E. Hoshizora

- 不难发现另一类物体的运动轨迹是一条直线。
- 找出垂直于这些直线的直线，并将坐标系旋转。
- 考虑如何计算答案。由于 m_i 的影响，直接找距离最近的直线是错的。
- 考虑决策单调性，在只考虑每个我们要计算答案的物体左边的另一类物体的情况下，决策点具有单调性。

E. Hoshizora

- 不难发现另一类物体的运动轨迹是一条直线。
- 找出垂直于这些直线的直线，并将坐标系旋转。
- 考虑如何计算答案。由于 m_i 的影响，直接找距离最近的直线是错的。
- 考虑决策单调性，在只考虑每个我们要计算答案的物体左边的另一类物体的情况下，决策点具有单调性。
- 使用各种方法（分治后决策单调性，二分栈，Simplified LARSCH，LARSCH 等）做到 $O(nV_c^2 \log^k n)$，$k \in \{0, 1, 2\}$。

E. Hoshizora

- 对于同类的，他们的相对位置永远不变。因此，我们只需要计算初始状态的最小值即可。

E. Hoshizora

- 对于同类的，他们的相对位置永远不变。因此，我们只需要计算初始状态的最小值即可。
- 考虑物体 m_i 相等的情况。

E. Hoshizora

- 对于同类的，他们的相对位置永远不变。因此，我们只需要计算初始状态的最小值即可。
- 考虑物体 m_i 相等的情况。
- 从大到小枚举块长 $B = 2^k$，将横纵均按 B 分块。

E. Hoshizora

- 对于同类的，他们的相对位置永远不变。因此，我们只需要计算初始状态的最小值即可。
- 考虑物体 m_i 相等的情况。
- 从大到小枚举块长 $B = 2^k$，将横纵均按 B 分块。
- 每次在某一个点第一次满足其所在块只有它一个点的时候，扫描其周围 $O(1)$ 的所有块并计算距离。

E. Hoshizora

- 对于同类的，他们的相对位置永远不变。因此，我们只需要计算初始状态的最小值即可。
- 考虑物体 m_i 相等的情况。
- 从大到小枚举块长 $B = 2^k$，将横纵均按 B 分块。
- 每次在某一个点第一次满足其所在块只有它一个点的时候，扫描其周围 $O(1)$ 的所有块并计算距离。
- 正确性上，距离它最近的点不超过 $2\sqrt{2}B$。

E. Hoshizora

- 对于同类的，他们的相对位置永远不变。因此，我们只需要计算初始状态的最小值即可。
- 考虑物体 m_i 相等的情况。
- 从大到小枚举块长 $B = 2^k$，将横纵均按 B 分块。
- 每次在某一个点第一次满足其所在块只有它一个点的时候，扫描其周围 $O(1)$ 的所有块并计算距离。
- 正确性上，距离它最近的点不超过 $2\sqrt{2}B$。
- 时间复杂度上，每个点每一层只会跟周围 $O(1)$ 个点计算距离，因此时间复杂度为 $O(n \log V)$。

E. Hoshizora

- m_i 不同时，只需要改成周围 $O(V_m)$（每个方向上 $O(\sqrt{V_m})$）个块内的所有点即可。

E. Hoshizora

- m_i 不同时，只需要改成周围 $O(V_m)$（每个方向上 $O(\sqrt{V_m})$）个块内的所有点即可。
- 这是因为所有不在周围 $O(V_m)$ 个块内的点的答案均大于 $O(B^2V_m)$，而因为上一层该点不独立，因此答案最小的点一定不超过 $O(B^2V_m)$。具体系数留给选手自行分析。

E. Hoshizora

- m_i 不同时，只需要改成周围 $O(V_m)$（每个方向上 $O(\sqrt{V_m})$）个块内的所有点即可。
- 这是因为所有不在周围 $O(V_m)$ 个块内的点的答案均大于 $O(B^2V_m)$，而因为上一层该点不独立，因此答案最小的点一定不超过 $O(B^2V_m)$。具体系数留给选手自行分析。
- 时间复杂度的分析与 m_i 相等的情况一样，每个点每一层只会跟周围 $O(V_m)$ 个点计算距离，这一部分时间复杂度是 $O(nV_m \log V)$。

E. Hoshizora

- m_i 不同时，只需要改成周围 $O(V_m)$（每个方向上 $O(\sqrt{V_m})$）个块内的所有点即可。
- 这是因为所有不在周围 $O(V_m)$ 个块内的点的答案均大于 $O(B^2V_m)$，而因为上一层该点不独立，因此答案最小的点一定不超过 $O(B^2V_m)$。具体系数留给选手自行分析。
- 时间复杂度的分析与 m_i 相等的情况一样，每个点每一层只会跟周围 $O(V_m)$ 个点计算距离，这一部分时间复杂度是 $O(nV_m \log V)$。
- 也可以使用期望时间复杂度 $O(nV_m \log V)$ 的 KDT 计算。

E. Hoshizora

- m_i 不同时，只需要改成周围 $O(V_m)$（每个方向上 $O(\sqrt{V_m})$）个块内的所有点即可。
- 这是因为所有不在周围 $O(V_m)$ 个块内的点的答案均大于 $O(B^2V_m)$，而因为上一层该点不独立，因此答案最小的点一定不超过 $O(B^2V_m)$。具体系数留给选手自行分析。
- 时间复杂度的分析与 m_i 相等的情况一样，每个点每一层只会跟周围 $O(V_m)$ 个点计算距离，这一部分时间复杂度是 $O(nV_m \log V)$。
- 也可以使用期望时间复杂度 $O(nV_m \log V)$ 的 KDT 计算。
- 总时间复杂度 $O(nV_m \log V + nV_c^2 \log n)$。

F. Sakura no Mahou

给定长度为 n 的序列 a。每次操作可以选择 $1 \le l_1 < r_1 < l_2 < r_2 < \cdots < l_k < r_k \le n$，其中 k 为任意正整数，并对所有 $a_{l_i} \sim a_{r_i}$ 取前缀异或和。求最少操作次数使得 a 变为 b 并构造。

$\sum n^2 \le 10^6$。

Author: migita_suzu

F. Sakura no Mahou

- 考虑倒着从 b 向 a 构造。则操作变成选择是否将 b_i 变为 $b_i \oplus b_{i-1}$。

F. Sakura no Mahou

- 考虑倒着从 b 向 a 构造。则操作变成选择是否将 b_i 变为 $b_i \oplus b_{i-1}$。
- 维护 b_i 可能的值。若 b_i 既可以是 1 又可以是 0 则设为 $?$。

F. Sakura no Mahou

- 考虑倒着从 b 向 a 构造。则操作变成选择是否将 b_i 变为 $b_i \oplus b_{i-1}$。
- 维护 b_i 可能的值。若 b_i 既可以是 1 又可以是 0 则设为 $?$。
- 倒推，每次如果 b_{i-1} 为 1 或 $?$ 则将 b_i 赋值为 $?$。

F. Sakura no Mahou

- 考虑倒着从 b 向 a 构造。则操作变成选择是否将 b_i 变为 $b_i \oplus b_{i-1}$。
- 维护 b_i 可能的值。若 b_i 既可以是 1 又可以是 0 则设为 $?$。
- 倒推，每次如果 b_{i-1} 为 1 或 $?$ 则将 b_i 赋值为 $?$。
- 当某一次操作后 a 与 b 成功匹配时，即为最少操作次数。

F. Sakura no Mahou

- 考虑倒着从 b 向 a 构造。则操作变成选择是否将 b_i 变为 $b_i \oplus b_{i-1}$。
- 维护 b_i 可能的值。若 b_i 既可以是 1 又可以是 0 则设为 $?$。
- 倒推，每次如果 b_{i-1} 为 1 或 $?$ 则将 b_i 赋值为 $?$。
- 当某一次操作后 a 与 b 成功匹配时，即为最少操作次数。
- 构造时，每次找到可以由目前的 a 操作出且符合匹配要求的 a' 即可。

F. Sakura no Mahou

- 考虑倒着从 b 向 a 构造。则操作变成选择是否将 b_i 变为 $b_i \oplus b_{i-1}$。
- 维护 b_i 可能的值。若 b_i 既可以是 1 又可以是 0 则设为 $?$。
- 倒推，每次如果 b_{i-1} 为 1 或 $?$ 则将 b_i 赋值为 $?$。
- 当某一次操作后 a 与 b 成功匹配时，即为最少操作次数。
- 构造时，每次找到可以由目前的 a 操作出且符合匹配要求的 a' 即可。
- 时间复杂度 $O(n^2)$。

G. Stop Plang!

有 $n = 300$ 个人参加比赛，其中有 48 个人获得了金牌。
当你问一个人他是否获得金牌时，如果他没有获得就会如实回答，否则会以 $\frac{1}{2}$ 的概率说真的，$\frac{1}{2}$ 的概率说假话。每一次问他的回答独立随机。
你需要**平均**进行不超过 568 次询问以确定哪 48 个人获得了金牌。每次询问一个子集 S，交互库会问 S 中的每一个人是否获得金牌，并返回回答是的人的数量。
多测 $T = 20$（也就是总共不超过 20×568 次询问），交互库不自适应，至多 10 组测试点。
Author: Otomachi_Una

G. Stop Plang!

- 首先把人打乱，将所有人是否获得金牌视为随机变量。

G. Stop Plang!

- 首先把人打乱，将所有人是否获得金牌视为随机变量。
- 考虑通过计算一次询问后该随机变量信息熵的最小值，选取可以带来最大信息熵降低的询问。

G. Stop Plang!

- 首先把人打乱，将所有人是否获得金牌视为随机变量。
- 考虑通过计算一次询问后该随机变量信息熵的最小值，选取可以带来最大信息熵降低的询问。
- 存储状态集合 V，维护每种状态的概率，等到某种状态概率小于 ϵ 时将其移出考虑集合。

G. Stop Plang!

- 首先把人打乱，将所有人是否获得金牌视为随机变量。
- 考虑通过计算一次询问后该随机变量信息熵的最小值，选取可以带来最大信息熵降低的询问。
- 存储状态集合 V，维护每种状态的概率，等到某种状态概率小于 ϵ 时将其移出考虑集合。
- 设阈值 B，当 $2|V| < B$ 时将下一个人是否获得金牌也加入状态。

G. Stop Plang!

- 首先把人打乱，将所有人是否获得金牌视为随机变量。
- 考虑通过计算一次询问后该随机变量信息熵的最小值，选取可以带来最大信息熵降低的询问。
- 存储状态集合 V，维护每种状态的概率，等到某种状态概率小于 ϵ 时将其移出考虑集合。
- 设阈值 B，当 $2|V| < B$ 时将下一个人是否获得金牌也加入状态。
- 对于询问，考虑爬山，每次将目前最优的询问集合 S 进行一些位上的异或并再次信息熵。这里不用模拟退火的主要原因是我们可以计算的信息熵次数较少。

G. Stop Plang!

- 首先把人打乱，将所有人是否获得金牌视为随机变量。
- 考虑通过计算一次询问后该随机变量信息熵的最小值，选取可以带来最大信息熵降低的询问。
- 存储状态集合 V，维护每种状态的概率，等到某种状态概率小于 ϵ 时将其移出考虑集合。
- 设阈值 B，当 $2|V| < B$ 时将下一个人是否获得金牌也加入状态。
- 对于询问，考虑爬山，每次将目前最优的询问集合 S 进行一些位上的异或并再次信息熵。这里不用模拟退火的主要原因是我们可以计算的信息熵次数较少。
- 当目前所有可能状态某个人均为同种状态（是否获得金牌），即可确认这个人的牌子。

G. Stop Plang!

- 考虑如何计算信息熵，对于条件概率使用贝叶斯公式。

G. Stop Plang!

- 考虑如何计算信息熵，对于条件概率使用贝叶斯公式。
- 计算出根据目前的概率估计，回答 $0 \sim |S|$ 的概率，以及每一个目前可能状态回答的 $0 \sim |S|$ 的概率。

G. Stop Plang!

- 考虑如何计算信息熵，对于条件概率使用贝叶斯公式。
- 计算出根据目前的概率估计，回答 $0 \sim |S|$ 的概率，以及每一个目前可能状态回答的 $0 \sim |S|$ 的概率。
- 先对于每一个回答，计算出其实际上是每一个状态的概率，根据这个概率计算信息熵，最后计算所有回答的信息熵的期望即可，时间复杂度单次 $O(B|S|)$。

G. Stop Plang!

- 考虑如何计算信息熵，对于条件概率使用贝叶斯公式。
- 计算出根据目前的概率估计，回答 $0 \sim |S|$ 的概率，以及每一个目前可能状态回答的 $0 \sim |S|$ 的概率。
- 先对于每一个回答，计算出其实际上是每一个状态的概率，根据这个概率计算信息熵，最后计算所有回答的信息熵的期望即可，时间复杂度单次 $O(B|S|)$。
- 考虑优化，设不同状态在询问的 S 中获得金牌人数的数量为 x，将 x 相同的状态合并计算。

G. Stop Plang!

- 考虑如何计算信息熵，对于条件概率使用贝叶斯公式。
- 计算出根据目前的概率估计，回答 $0 \sim |S|$ 的概率，以及每一个目前可能状态回答的 $0 \sim |S|$ 的概率。
- 先对于每一个回答，计算出其实际上是每一个状态的概率，根据这个概率计算信息熵，最后计算所有回答的信息熵的期望即可，时间复杂度单次 $O(B|S|)$。
- 考虑优化，设不同状态在询问的 S 中获得金牌人数的数量为 x，将 x 相同的状态合并计算。
- 将信息熵中的 $\log(\frac{P_{i,j}}{Q_i})$ 项（其中 Q_i 是回答 i 的概率，$P_{i,j}$ 为状态 j 回答 i 的概率）拆成 $\log(p_j) + \log(\frac{\binom{x}{i}}{2^x P_i})$（其中 p_j 为根据以前信息状态 j 的概率），并分开维护和，时间复杂度单次 $O(B + |S|^2)$。

G. Stop Plang!

实现细节：

- 取 $B = 800$，限制目前考虑的人数不超过阈值 $C = 30$ 即可保证 B 和 $|S|^2$ 两部分时间复杂度接近。

G. Stop Plang!

实现细节：

- 取 $B = 800$，限制目前考虑的人数不超过阈值 $C = 30$ 即可保证 B 和 $|S|^2$ 两部分时间复杂度接近。
- 初始可以将每个人金牌的概率设为 0.16 以得到更加准确的估计。

G. Stop Plang!

实现细节：

- 取 $B = 800$，限制目前考虑的人数不超过阈值 $C = 30$ 即可保证 B 和 $|S|^2$ 两部分时间复杂度接近。
- 初始可以将每个人金牌的概率设为 0.16 以得到更加准确的估计。
- 认为一个状态不可能的 ϵ 可以设在 10^{-6} 级别，且计算其与最可能的状态之间的比值并判断可以增加正确性。

G. Stop Plang!

实现细节：

- 取 $B = 800$，限制目前考虑的人数不超过阈值 $C = 30$ 即可保证 B 和 $|S|^2$ 两部分时间复杂度接近。
- 初始可以将每个人金牌的概率设为 0.16 以得到更加准确的估计。
- 认为一个状态不可能的 ϵ 可以设在 10^{-6} 级别，且计算其与最可能的状态之间的比值并判断可以增加正确性。
- 当扔掉状态后，可以将其他状态再划算为总概率 1。

G. Stop Plang!

实现细节：

- 取 $B = 800$，限制目前考虑的人数不超过阈值 $C = 30$ 即可保证 B 和 $|S|^2$ 两部分时间复杂度接近。
- 初始可以将每个人金牌的概率设为 0.16 以得到更加准确的估计。
- 认为一个状态不可能的 ϵ 可以设在 10^{-6} 级别，且计算其与最可能的状态之间的比值并判断可以增加正确性。
- 当扔掉状态后，可以将其他状态再划算为总概率 1。
- 精细实现大概可以做到期望 500 次询问且方差足以通过。实现得不够精细可能会比较卡操作次数/时间，被卡了建议洗把脸。

G. Stop Plang!

- 首先把人打乱，将所有人是否获得金牌视为随机变量。

G. Stop Plang!

- 首先把人打乱，将所有人是否获得金牌视为随机变量。
- 考虑设计一个函数，表示对于一个人的集合，在其中找出至少 k 个金牌。

G. Stop Plang!

- 首先把人打乱，将所有人是否获得金牌视为随机变量。
- 考虑设计一个函数，表示对于一个人的集合，在其中找出至少 k 个金牌。
- 对于每一个人，维护他在此之前在多少次回答为 0 人获得金牌的询问中出现了，设为 g_i。不难发现这个次数越少，其获得金牌的概率越高。

G. Stop Plang!

- 首先把人打乱，将所有人是否获得金牌视为随机变量。
- 考虑设计一个函数，表示对于一个人的集合，在其中找出至少 k 个金牌。
- 对于每一个人，维护他在此之前在多少次回答为 0 人获得金牌的询问中出现了，设为 g_i。不难发现这个次数越少，其获得金牌的概率越高。
- 每次挑选 g_i 最小的若干人进行询问，设回答为 x 个人获得金牌。$x \neq 0$ 时可以递归子问题（这些人中找至少 k 个金牌），否则将所有人 g_i 加一。

G. Stop Plang!

实现细节：

- 当 $|V| = k$ 时可以确定所有人都金牌了，当找到超过 k 个金牌后直接返回。

G. Stop Plang!

实现细节：

- 当 $|V| = k$ 时可以确定所有人都金牌了，当找到超过 k 个金牌后直接返回。
- 限制每次询问的人数在 $\frac{|V|}{3}$ 以内。

G. Stop Plang!

实现细节：

- 当 $|V| = k$ 时可以确定所有人都金牌了，当找到超过 k 个金牌后直接返回。
- 限制每次询问的人数在 $\frac{|V|}{3}$ 以内。
- 询问时让其中每个人是金牌的概率之和尽量接近阈值 $C = 12$，可以用 2^{-g_i} 对一个人的金牌概率进行估计。

G. Stop Plang!

实现细节：

- 当 $|V| = k$ 时可以确定所有人都金牌了，当找到超过 k 个金牌后直接返回。
- 限制每次询问的人数在 $\frac{|V|}{3}$ 以内。
- 询问时让其中每个人是金牌的概率之和尽量接近阈值 $C = 12$，可以用 2^{-g_i} 对一个人的金牌概率进行估计。
- 递归内层的 g_i 在外层复用（即，累加到外层的统计中）。

H. Shiratama

给定排列 p，在 $2n$ 次 1 操作和 $2n^2$ 次 2 操作内将排列排序：

- 操作 1：对于所有 $i = 1, 3, \ldots, 2n-1$，交换 p_i, p_{i+1}；
- 操作 2：选择一个 $i \in \{2, 4, \ldots, 2n\}$，交换 p_i, p_{i+1}。我们认为 $p_1 = p_{2n+1}$。

$n \leq 100$，$\sum n^2 \leq 10^6$。

Author: Otomachi_Una

H. Shiratama

- 考虑将 p 序列重排，形成如下顺序：

p_1	p_4	p_5	p_8	p_9	p_{12}	p_{13}	$\cdots$
p_2	p_3	p_6	p_7	p_{10}	p_{11}	p_{14}	$\cdots$

H. Shiratama

- 考虑将 p 序列重排，形成如下顺序：

p_1	p_4	p_5	p_8	p_9	p_{12}	p_{13}	$\cdots$
p_2	p_3	p_6	p_7	p_{10}	p_{11}	p_{14}	$\cdots$

- 观察可能会有的交换。不难发现要么是上下两行之间交换，要么是相邻两列上下行有一行可以左右交换。

H. Shiratama

- 考虑将 p 序列重排，形成如下顺序：

p_1	p_4	p_5	p_8	p_9	p_{12}	p_{13}	$\cdots$
p_2	p_3	p_6	p_7	p_{10}	p_{11}	p_{14}	$\cdots$

- 观察可能会有的交换。不难发现要么是上下两行之间交换，要么是相邻两列上下行有一行可以左右交换。
- 考虑上面一行按 $n+1 \sim 2n$ 标号，下面一行按 $1 \sim n$ 标号，得到排列 q，转化为将 q 排序。

H. Shiratama

- n 为偶数时，上一行和下一行是固定的，不会发生行之间的交换。n 为奇数时，q_n 可以与 q_{n+1} 交换。

H. Shiratama

- n 为偶数时，上一行和下一行是固定的，不会发生行之间的交换。n 为奇数时，q_n 可以与 q_{n+1} 交换。
- 贪心，每次将所有行内（或 $(n, n+1)$）需要交换（即 $q_i > q_{i+1}$）的相邻元组交换。

H. Shiratama

- n 为偶数时，上一行和下一行是固定的，不会发生行之间的交换。n 为奇数时，q_n 可以与 q_{n+1} 交换。
- 贪心，每次将所有行内（或 $(n, n+1)$）需要交换（即 $q_i > q_{i+1}$）的相邻元组交换。
- 将两次 1 操作对应的 2 操作视为一组，则所有 $(i, i+1)$ 在一组内必然可以被交换一次。

H. Shiratama

- n 为偶数时，上一行和下一行是固定的，不会发生行之间的交换。n 为奇数时，q_n 可以与 q_{n+1} 交换。
- 贪心，每次将所有行内（或 $(n, n+1)$）需要交换（即 $q_i > q_{i+1}$）的相邻元组交换。
- 将两次 1 操作对应的 2 操作视为一组，则所有 $(i, i+1)$ 在一组内必然可以被交换一次。
- 容易发现，在该策略下，每一个 i 都可以在 n 组内归为。

H. Shiratama

- n 为偶数时，上一行和下一行是固定的，不会发生行之间的交换。n 为奇数时，q_n 可以与 q_{n+1} 交换。
- 贪心，每次将所有行内（或 $(n, n+1)$）需要交换（即 $q_i > q_{i+1}$）的相邻元组交换。
- 将两次 1 操作对应的 2 操作视为一组，则所有 $(i, i+1)$ 在一组内必然可以被交换一次。
- 容易发现，在该策略下，每一个 i 都可以在 n 组内归为。
- 每一组内有 2 次 1 操作和至多 $2n$ 次 2 操作，因此总共最多需要 $2n$ 次 1 操作与 $2n^2$ 次 2 操作。

H. Shiratama

- n 为偶数时，上一行和下一行是固定的，不会发生行之间的交换。n 为奇数时，q_n 可以与 q_{n+1} 交换。
- 贪心，每次将所有行内（或 $(n, n+1)$）需要交换（即 $q_i > q_{i+1}$）的相邻元组交换。
- 将两次 1 操作对应的 2 操作视为一组，则所有 $(i, i+1)$ 在一组内必然可以被交换一次。
- 容易发现，在该策略下，每一个 i 都可以在 n 组内归为。
- 每一组内有 2 次 1 操作和至多 $2n$ 次 2 操作，因此总共最多需要 $2n$ 次 1 操作与 $2n^2$ 次 2 操作。
- 时间复杂度 $O(n^2)$。

I. Embedding Trees

定义 R_k 为直径为 $2k$，每个点度数不超过 3 的点数最多的树。
定义 $f(T)$ 为最大的 k 使得存在 T 的导出子树与 R_k 同构。求一个 n 个点的 T 的所有导出子树 T'，$f(T')$ 的和。
$n \le 10^5$，$\sum n \le 10^6$。
Author: sszcdjr

I. Embedding Trees

- 考察 R_k 的形态，容易证明其为两个深度为 k 的满二叉树，根节点用一条边链接。

I. Embedding Trees

- 考察 R_k 的形态，容易证明其为两个深度为 k 的满二叉树，根节点用一条边链接。
- R_k 的点数是 $O(2^k)$ 的，因此 $f(T)$ 的上限是 $O(\log n)$。

I. Embedding Trees

- 考察 R_k 的形态，容易证明其为两个深度为 k 的满二叉树，根节点用一条边链接。
- R_k 的点数是 $O(2^k)$ 的，因此 $f(T)$ 的上限是 $O(\log n)$。
- 考虑如何计算 $f(T)$。二分 k，试图在一个子树内尽可能多地放置 R_k 的一部分。

I. Embedding Trees

- 考察 R_k 的形态，容易证明其为两个深度为 k 的满二叉树，根节点用一条边链接。
- R_k 的点数是 $O(2^k)$ 的，因此 $f(T)$ 的上限是 $O(\log n)$。
- 考虑如何计算 $f(T)$。二分 k，试图在一个子树内尽可能多地放置 R_k 的一部分。
- 子树内的一部分必然是 R_k 切去一条边后的某一个部分，总共有 $2k-1$ 种可能。

I. Embedding Trees

- 考察 R_k 的形态，容易证明其为两个深度为 k 的满二叉树，根节点用一条边链接。
- R_k 的点数是 $O(2^k)$ 的，因此 $f(T)$ 的上限是 $O(\log n)$。
- 考虑如何计算 $f(T)$。二分 k，试图在一个子树内尽可能多地放置 R_k 的一部分。
- 子树内的一部分必然是 R_k 切去一条边后的某一个部分，总共有 $2k-1$ 种可能。
- 可以证明存在将 $2k-1$ 种状态排序的方式，使得对于任意一种子树外的情况，其可以组成 R_k 对应的子树内状态必然是一个后缀。

I. Embedding Trees

- 考察 R_k 的形态，容易证明其为两个深度为 k 的满二叉树，根节点用一条边链接。
- R_k 的点数是 $O(2^k)$ 的，因此 $f(T)$ 的上限是 $O(\log n)$。
- 考虑如何计算 $f(T)$。二分 k，试图在一个子树内尽可能多地放置 R_k 的一部分。
- 子树内的一部分必然是 R_k 切去一条边后的某一个部分，总共有 $2k-1$ 种可能。
- 可以证明存在将 $2k-1$ 种状态排序的方式，使得对于任意一种子树外的情况，其可以组成 R_k 对应的子树内状态必然是一个后缀。
- 具体而言，前 k 种状态为深度为 $1 \sim k$ 的满二叉树，后 $k-1$ 种状态为仅缺一个深度为 $k-1 \sim 1$ 的满二叉树。

I. Embedding Trees

- 一个观察是，对于任意一个仅缺一个深度为 i 的满二叉树，其必然包含一个深度为 i 的二叉树且以子树根节点为根。同时，一个深度为 i 的满二叉树必然包含一个同根的深度为 $i-1$ 的满二叉树。

I. Embedding Trees

- 一个观察是，对于任意一个仅缺一个深度为 i 的满二叉树，其必然包含一个深度为 i 的二叉树且以子树根节点为根。同时，一个深度为 i 的满二叉树必然包含一个同根的深度为 $i-1$ 的满二叉树。
- 考虑如何转移。

I. Embedding Trees

- 一个观察是，对于任意一个仅缺一个深度为 i 的满二叉树，其必然包含一个深度为 i 的二叉树且以子树根节点为根。同时，一个深度为 i 的满二叉树必然包含一个同根的深度为 $i-1$ 的满二叉树。
- 考虑如何转移。
- 对于所有儿子都属于前 $k-1$ 种状态的情况，该节点的状态为儿子的 $\max$。

I. Embedding Trees

- 一个观察是，对于任意一个仅缺一个深度为 i 的满二叉树，其必然包含一个深度为 i 的二叉树且以子树根节点为根。同时，一个深度为 i 的满二叉树必然包含一个同根的深度为 $i-1$ 的满二叉树。
- 考虑如何转移。
- 对于所有儿子都属于前 $k-1$ 种状态的情况，该节点的状态为儿子的 $\max$。
- 对于存在至少一个儿子属于后 k 种状态的情况，不妨设其中最靠后的状态为缺一个深度为 i 的满二叉树。此时，我们希望剩余部分有两个深度为 $i-1$ 的满二叉树。

I. Embedding Trees

- 一个观察是，对于任意一个仅缺一个深度为 i 的满二叉树，其必然包含一个深度为 i 的二叉树且以子树根节点为根。同时，一个深度为 i 的满二叉树必然包含一个同根的深度为 $i-1$ 的满二叉树。
- 考虑如何转移。
- 对于所有儿子都属于前 $k-1$ 种状态的情况，该节点的状态为儿子的 $\max$。
- 对于存在至少一个儿子属于后 k 种状态的情况，不妨设其中最靠后的状态为缺一个深度为 i 的满二叉树。此时，我们希望剩余部分有两个深度为 $i-1$ 的满二叉树。
- 当然，对于 $i=1$ 的情况，直接将这个点拼上去就得到了一个符合要求的答案（R_k 的一种嵌入）。

I. Embedding Trees

- 对于任意其他儿子，若其状态也是缺一个深度为 j 的满二叉树，则必然有一个以该儿子为根的深度为 $i-1$ 的满二叉树，因为 $j > i$。

I. Embedding Trees

- 对于任意其他儿子，若其状态也是缺一个深度为 j 的满二叉树，则必然有一个以该儿子为根的深度为 $i-1$ 的满二叉树，因为 $j > i$。
- 否则，存在以该儿子为根的深度为 $i-1$ 的满二叉树当且仅当其状态为深度为 j 的满二叉树，其中 $j \geq i-1$。

I. Embedding Trees

- 对于任意其他儿子，若其状态也是缺一个深度为 j 的满二叉树，则必然有一个以该儿子为根的深度为 $i-1$ 的满二叉树，因为 $j > i$。
- 否则，存在以该儿子为根的深度为 $i-1$ 的满二叉树当且仅当其状态为深度为 j 的满二叉树，其中 $j \geq i-1$。
- 若这些儿子中有至少两个符合要求（存在以该儿子为根的深度为 $i-1$ 的满二叉树），则找到了一个符合要求的答案。

I. Embedding Trees

- 对于任意其他儿子，若其状态也是缺一个深度为 j 的满二叉树，则必然有一个以该儿子为根的深度为 $i-1$ 的满二叉树，因为 $j > i$。
- 否则，存在以该儿子为根的深度为 $i-1$ 的满二叉树当且仅当其状态为深度为 j 的满二叉树，其中 $j \geq i-1$。
- 若这些儿子中有至少两个符合要求（存在以该儿子为根的深度为 $i-1$ 的满二叉树），则找到了一个符合要求的答案。
- 若这些儿子中只有一个符合要求，则可以转移到仅缺一个深度为 $i-1$ 的满二叉树的状态。

I. Embedding Trees

- 对于任意其他儿子，若其状态也是缺一个深度为 j 的满二叉树，则必然有一个以该儿子为根的深度为 $i-1$ 的满二叉树，因为 $j > i$。
- 否则，存在以该儿子为根的深度为 $i-1$ 的满二叉树当且仅当其状态为深度为 j 的满二叉树，其中 $j \geq i-1$。
- 若这些儿子中有至少两个符合要求（存在以该儿子为根的深度为 $i-1$ 的满二叉树），则找到了一个符合要求的答案。
- 若这些儿子中只有一个符合要求，则可以转移到仅缺一个深度为 $i-1$ 的满二叉树的状态。
- 若这些儿子中没有符合要求的，则可以转移到最大的 $j+1$ 使得存在一个儿子状态为深度为 j 的满二叉树。（显然 $j < i-1$，因此另一个深度为 j 的满二叉树可以由状态为缺一个深度为 i 的满二叉树的儿子提供）

I. Embedding Trees

- 考虑如何计算 $\sum f(T')$，首先先枚举 $f(T') < k$。

I. Embedding Trees

- 考虑如何计算 $\sum f(T')$，首先先枚举 $f(T') < k$。
- 设计 $dp_{i,j}$ 表示 i 子树，对应的状态编号为 j。所有儿子状态编号 $< k$ 是容易转移的。

I. Embedding Trees

- 考虑如何计算 $\sum f(T')$，首先先枚举 $f(T') < k$。
- 设计 $dp_{i,j}$ 表示 i 子树，对应的状态编号为 j。所有儿子状态编号 $< k$ 是容易转移的。
- 先选定有 $0/1$ 儿子符合上述“要求”。对于有 1 个儿子符合要求的情况，选定 i 进行转移。对于有 0 个儿子符合要求的情况，选定 j 进行转移。

I. Embedding Trees

- 考虑如何计算 $\sum f(T')$，首先先枚举 $f(T') < k$。
- 设计 $dp_{i,j}$ 表示 i 子树，对应的状态编号为 j。所有儿子状态编号 $< k$ 是容易转移的。
- 先选定有 $0/1$ 儿子符合上述“要求”。对于有 1 个儿子符合要求的情况，选定 i 进行转移。对于有 0 个儿子符合要求的情况，选定 j 进行转移。
- 使用前缀和优化，时间复杂度 $O(n \log^2 n)$。

I. Embedding Trees

- 考虑如何计算 $\sum f(T')$，首先先枚举 $f(T') < k$。
- 设计 $dp_{i,j}$ 表示 i 子树，对应的状态编号为 j。所有儿子状态编号 $< k$ 是容易转移的。
- 先选定有 $0/1$ 儿子符合上述“要求”。对于有 1 个儿子符合要求的情况，选定 i 进行转移。对于有 0 个儿子符合要求的情况，选定 j 进行转移。
- 使用前缀和优化，时间复杂度 $O(n \log^2 n)$。
- Bonus: 精细实现并分析状态数，并证明时间复杂度为 $O(n \log n)$。

J. Hiking Trip

给定一个残缺的序列。求将其中残缺的位置随机替换为 $[1, m]$ 的整数后，单峰子区间的期望数量。

$\sum n \leq 10^7$，$m \leq 10^9$。

Author: sszcdjr

J. Hiking Trip

- 将期望转为方案数，分类讨论，枚举峰值的位置。

J. Hiking Trip

- 将期望转为方案数，分类讨论，枚举峰值的位置。
- 峰在给定的数上时，枚举左右端点所在的区间（一段 -1），通过预处理出两个给定的数之间的填数方案以及在每一段内结束的填数方案即可 $O(n)$ 计算。

J. Hiking Trip

- 将期望转为方案数，分类讨论，枚举峰值的位置。
- 峰在给定的数上时，枚举左右端点所在的区间（一段 -1），通过预处理出两个给定的数之间的填数方案以及在每一段内结束的填数方案即可 $O(n)$ 计算。
- 峰在未给定的数上，枚举这个段。

J. Hiking Trip

- 将期望转为方案数，分类讨论，枚举峰值的位置。
- 峰在给定的数上时，枚举左右端点所在的区间（一段 -1），通过预处理出两个给定的数之间的填数方案以及在每一段内结束的填数方案即可 $O(n)$ 计算。
- 峰在未给定的数上，枚举这个段。
- 首先考察整个子区间都在这一段内的情况，计算一个长度为 len 的序列填为单峰序列的方案数，然后做两次前缀和即可。

J. Hiking Trip

- 将期望转为方案数，分类讨论，枚举峰值的位置。
- 峰在给定的数上时，枚举左右端点所在的区间（一段 -1），通过预处理出两个给定的数之间的填数方案以及在每一段内结束的填数方案即可 $O(n)$ 计算。
- 峰在未给定的数上，枚举这个段。
- 首先考察整个子区间都在这一段内的情况，计算一个长度为 len 的序列填为单峰序列的方案数，然后做两次前缀和即可。
- 枚举最大值 i，可以得到式子 $\sum_{i=1}^{m} \binom{2(i-1)}{len-1}$。

J. Hiking Trip

- 将期望转为方案数，分类讨论，枚举峰值的位置。
- 峰在给定的数上时，枚举左右端点所在的区间（一段 -1），通过预处理出两个给定的数之间的填数方案以及在每一段内结束的填数方案即可 $O(n)$ 计算。
- 峰在未给定的数上，枚举这个段。
- 首先考察整个子区间都在这一段内的情况，计算一个长度为 len 的序列填为单峰序列的方案数，然后做两次前缀和即可。
- 枚举最大值 i，可以得到式子 $\sum_{i=1}^{m} \binom{2(i-1)}{len-1}$。
- 对于这个子区间有一边在段内和两边都不在段内的情况，前者枚举一边的位置，均可得到形如 $\sum_{i=1}^{x} \binom{2i+y}{z}$ 的式子。

J. Hiking Trip

- 设 $f_z = \sum_{i=1}^{x} \binom{2i+y}{z}$。

J. Hiking Trip

- 设 $f_z = \sum_{i=1}^{x} \binom{2i+y}{z}$。
- $\sum_{i=1}^{x} \binom{2i+y+1}{z} = \sum_{i=1}^{x} \binom{2i+y}{z-1} + \sum_{i=1}^{x} \binom{2i+y}{z} = f_{z-1} + f_z$。

J. Hiking Trip

- 设 $f_z = \sum_{i=1}^{x} \binom{2i+y}{z}$。
- $\sum_{i=1}^{x} \binom{2i+y+1}{z} = \sum_{i=1}^{x} \binom{2i+y}{z-1} + \sum_{i=1}^{x} \binom{2i+y}{z} = f_{z-1} + f_z$。
- $\sum_{i=1}^{x} \binom{2i+y+1}{z} + f_z = \sum_{i=2}^{2x+1} \binom{i+y}{z}$

J. Hiking Trip

- 设 $f_z = \sum_{i=1}^{x} \binom{2i+y}{z}$。
- $\sum_{i=1}^{x} \binom{2i+y+1}{z} = \sum_{i=1}^{x} \binom{2i+y}{z-1} + \sum_{i=1}^{x} \binom{2i+y}{z} = f_{z-1} + f_z$。
- $\sum_{i=1}^{x} \binom{2i+y+1}{z} + f_z = \sum_{i=2}^{2x+1} \binom{i+y}{z}$
- $\sum_{i=2}^{2x+1} \binom{i+y}{z} = \binom{2x+y+2}{z+1} - \binom{y+2}{z+1}$。

J. Hiking Trip

- 设 $f_z = \sum_{i=1}^{x} \binom{2i+y}{z}$。
- $\sum_{i=1}^{x} \binom{2i+y+1}{z} = \sum_{i=1}^{x} \binom{2i+y}{z-1} + \sum_{i=1}^{x} \binom{2i+y}{z} = f_{z-1} + f_z$。
- $\sum_{i=1}^{x} \binom{2i+y+1}{z} + f_z = \sum_{i=2}^{2x+1} \binom{i+y}{z}$
- $\sum_{i=2}^{2x+1} \binom{i+y}{z} = \binom{2x+y+2}{z+1} - \binom{y+2}{z+1}$。
- 当然，也可以用组合意义理解。用总方案数减去最大值出现两次的方案数再除以二即为 z 的答案，而最大值出现两次的方案数与选出 $z-1$ 个数的方案数构成双射，亦可以得到上述公式。

J. Hiking Trip

- 设 $f_z = \sum_{i=1}^{x} \binom{2i+y}{z}$。
- $\sum_{i=1}^{x} \binom{2i+y+1}{z} = \sum_{i=1}^{x} \binom{2i+y}{z-1} + \sum_{i=1}^{x} \binom{2i+y}{z} = f_{z-1} + f_z$。
- $\sum_{i=1}^{x} \binom{2i+y+1}{z} + f_z = \sum_{i=2}^{2x+1} \binom{i+y}{z}$
- $\sum_{i=2}^{2x+1} \binom{i+y}{z} = \binom{2x+y+2}{z+1} - \binom{y+2}{z+1}$。
- 当然，也可以用组合意义理解。用总方案数减去最大值出现两次的方案数再除以二即为 z 的答案，而最大值出现两次的方案数与选出 $z-1$ 个数的方案数构成双射，亦可以得到上述公式。
- 对于每一个区间需要求出 $f_0 \sim f_{len}$，因此复杂度为 $O(len)$。

J. Hiking Trip

- 设 $f_z = \sum_{i=1}^{x} \binom{2i+y}{z}$。
- $\sum_{i=1}^{x} \binom{2i+y+1}{z} = \sum_{i=1}^{x} \binom{2i+y}{z-1} + \sum_{i=1}^{x} \binom{2i+y}{z} = f_{z-1} + f_z$。
- $\sum_{i=1}^{x} \binom{2i+y+1}{z} + f_z = \sum_{i=2}^{2x+1} \binom{i+y}{z}$
- $\sum_{i=2}^{2x+1} \binom{i+y}{z} = \binom{2x+y+2}{z+1} - \binom{y+2}{z+1}$。
- 当然，也可以用组合意义理解。用总方案数减去最大值出现两次的方案数再除以二即为 z 的答案，而最大值出现两次的方案数与选出 $z-1$ 个数的方案数构成双射，亦可以得到上述公式。
- 对于每一个区间需要求出 $f_0 \sim f_{len}$，因此复杂度为 $O(len)$。
- 时间复杂度 $O(n + \log mod)$，需要预处理一下逆元。

K. 0721 Master

有一个集合 S。每次操作可以插入或删除一个数。构造一种满足以下条件的操作序列，使得在每次操作后集合内第 k 小的数所构成的集合为 T：

- $1 \sim n$ 均被插入集合至少一次，且 i 在 $i+1$ 之前被插入即可。

$\sum n \leq 5000$。

Author: 251Sec

K. 0721 Master

- 按顺序考虑每个数，如果这个数直接加入与这个数是否在集合 T 内相符则直接加入。

K. 0721 Master

- 按顺序考虑每个数，如果这个数直接加入与这个数是否在集合 T 内相符则直接加入。
- 若这个数没有出现在集合 T 中但加入它后会成为第 k 小，则把目前最大的删掉。

K. 0721 Master

- 按顺序考虑每个数，如果这个数直接加入与这个数是否在集合 T 内相符则直接加入。
- 若这个数没有出现在集合 T 中但加入它后会成为第 k 小，则把目前最大的删掉。
- 若这个数出现在集合 T 中但加入它后不会成为第 k 小，如果目前集合大小大于等于 k 则一直弹最大值即可。否则必然不合法。

K. 0721 Master

- 按顺序考虑每个数，如果这个数直接加入与这个数是否在集合 T 内相符则直接加入。
- 若这个数没有出现在集合 T 中但加入它后会成为第 k 小，则把目前最大的删掉。
- 若这个数出现在集合 T 中但加入它后不会成为第 k 小，如果目前集合大小大于等于 k 则一直弹最大值即可。否则必然不合法。
- 注意 $k = 1$ 的边界情况。

K. 0721 Master

- 按顺序考虑每个数，如果这个数直接加入与这个数是否在集合 T 内相符则直接加入。
- 若这个数没有出现在集合 T 中但加入它后会成为第 k 小，则把目前最大的删掉。
- 若这个数出现在集合 T 中但加入它后不会成为第 k 小，如果目前集合大小大于等于 k 则一直弹最大值即可。否则必然不合法。
- 注意 $k = 1$ 的边界情况。
- 时间复杂度 $O(n)$。

L. Senren Banka

有一个包含 n 个关卡的游戏，第 j 次玩第 i 关有 $p_{i,\min(j,m)}$ 概率通过，如果没有通过则需要从 1 关卡开始重新一轮玩。求第一次通过第 n 关时的期望轮数。

$n \leq 20$，$m \leq 5 \times 10^4$。

Author: Otomachi_Una

L. Senren Banka

- 注意到在通过关卡 i 至少 m 次后，对于任意关卡 i 之前的关卡均至少通过 m 次。

L. Senren Banka

- 注意到在通过关卡 i 至少 m 次后，对于任意关卡 i 之前的关卡均至少通过 m 次。
- 考虑 $dp_{i,j}$ 表示玩了关卡 i 恰好 j 次的期望轮数，f_i 表示 $dp_{i,k+1} = dp_{i,k} + f_i$，其中 $k \geq m$。

L. Senren Banka

- 注意到在通过关卡 i 至少 m 次后，对于任意关卡 i 之前的关卡均至少通过 m 次。
- 考虑 $dp_{i,j}$ 表示玩了关卡 i 恰好 j 次的期望轮数，f_i 表示 $dp_{i,k+1} = dp_{i,k} + f_i$，其中 $k \geq m$。
- 考虑转移，如果关卡 $i-1$ 玩了不超过 m 次则计算关卡 $i-1$ 玩了第 j 次时恰好关卡 i 玩了 j' 次的概率，从而计算期望。

L. Senren Banka

- 注意到在通过关卡 i 至少 m 次后，对于任意关卡 i 之前的关卡均至少通过 m 次。
- 考虑 $dp_{i,j}$ 表示玩了关卡 i 恰好 j 次的期望轮数，f_i 表示 $dp_{i,k+1} = dp_{i,k} + f_i$，其中 $k \geq m$。
- 考虑转移，如果关卡 $i-1$ 玩了不超过 m 次则计算关卡 $i-1$ 玩了第 j 次时恰好关卡 i 玩了 j' 次的概率，从而计算期望。
- 如果关卡 $i-1$ 玩了超过 m 次，则计算关卡 $i-1$ 玩了 m 次时关卡 i 玩了 j 次的概率，接下来玩一次关卡 i 所需要的期望轮数恰好为 $\frac{f_{i-1}}{p_{i-1,m}}$，使用前缀和计算。

L. Senren Banka

- 注意到在通过关卡 i 至少 m 次后，对于任意关卡 i 之前的关卡均至少通过 m 次。
- 考虑 $dp_{i,j}$ 表示玩了关卡 i 恰好 j 次的期望轮数，f_i 表示 $dp_{i,k+1} = dp_{i,k} + f_i$，其中 $k \geq m$。
- 考虑转移，如果关卡 $i-1$ 玩了不超过 m 次则计算关卡 $i-1$ 玩了第 j 次时恰好关卡 i 玩了 j' 次的概率，从而计算期望。
- 如果关卡 $i-1$ 玩了超过 m 次，则计算关卡 $i-1$ 玩了 m 次时关卡 i 玩了 j 次的概率，接下来玩一次关卡 i 所需要的期望轮数恰好为 $\frac{f_{i-1}}{p_{i-1,m}}$，使用前缀和计算。
- 这个概率可以使用简单的背包 dp 计算，时间复杂度为 $O(nm^2)$。

L. Senren Banka

- 使用分治 NTT 可以直接计算关卡 $i-1$ 玩了 m 次时关卡 i 玩了 j 次的概率。

L. Senren Banka

- 使用分治 NTT 可以直接计算关卡 $i-1$ 玩了 m 次时关卡 i 玩了 j 次的概率。
- 对于前一类转移，考虑在分治 NTT 到对应的 j 时选定第 j 轮通过，直接记录期望。

L. Senren Banka

- 使用分治 NTT 可以直接计算关卡 $i-1$ 玩了 m 次时关卡 i 玩了 j 次的概率。
- 对于前一类转移，考虑在分治 NTT 到对应的 j 时选定第 j 轮通过，直接记录期望。
- 每次合并两个儿子时，不妨设通关概率对应的多项式为 f，选定某个 j 进行转移的期望对应的多项式为 g，则有 $(f,g) \leftarrow (f_l \times f_r, g_l + f_l \times g_r)$。

L. Senren Banka

- 使用分治 NTT 可以直接计算关卡 $i-1$ 玩了 m 次时关卡 i 玩了 j 次的概率。
- 对于前一类转移，考虑在分治 NTT 到对应的 j 时选定第 j 轮通过，直接记录期望。
- 每次合并两个儿子时，不妨设通关概率对应的多项式为 f，选定某个 j 进行转移的期望对应的多项式为 g，则有 $(f,g) \leftarrow (f_l \times f_r, g_l + f_l \times g_r)$。
- 时间复杂度 $O(nm \log^2 m)$。

M. Put Happiness in your Pocket

进行至多 n 次以下询问，猜出一棵 n 个节点的有标号树的树形态或报告无论进行多少次以下询问，都不可能确定树形态：

- 询问排列 p，交互库会回答将编号为 i 的点点权赋值为 2^{p_i} 后树的直径。

$n \leq 100$，$\sum n^2 \leq 10^6$。

Author: Otomachi_Una

M. Put Happiness in your Pocket

- 考虑什么情况会不可能确定树形态，不难发现当且仅当有相邻两个度数不超过 2 的点（特判 $n = 2$ 的情况），因为我们永远无法确认这两个点的顺序。

M. Put Happiness in your Pocket

- 考虑什么情况会不可能确定树形态，不难发现当且仅当有相邻两个度数不超过 2 的点（特判 $n = 2$ 的情况），因为我们永远无法确认这两个点的顺序。
- 考虑交互库怎么写。维护目前路径，贪心从大往小枚举点权为 2^i 的点，并判断是否可以加入目前路径。

M. Put Happiness in your Pocket

- 考虑什么情况会不可能确定树形态，不难发现当且仅当有相邻两个度数不超过 2 的点（特判 $n = 2$ 的情况），因为我们永远无法确认这两个点的顺序。
- 考虑交互库怎么写。维护目前路径，贪心从大往小枚举点权为 2^i 的点，并判断是否可以加入目前路径。
- 因此，可以通过给定某两个点 n 和 $n - 1$ 以得到一条包含这两个点的路径。

M. Put Happiness in your Pocket

- 对于两条不同的路径，如果他们有一个共同的端点，我们可以得到一个“大致”的形态，也就是两条路径交的部分在链的一侧，然后分叉出两条路径独有的部分。

M. Put Happiness in your Pocket

- 对于两条不同的路径，如果他们有一个共同的端点，我们可以得到一个“大致”的形态，也就是两条路径交的部分在链的一侧，然后分叉出两条路径独有的部分。
- 以上述操作为基础，考虑如何得到整棵树。

M. Put Happiness in your Pocket

- 对于两条不同的路径，如果他们有一个共同的端点，我们可以得到一个“大致”的形态，也就是两条路径交的部分在链的一侧，然后分叉出两条路径独有的部分。
- 以上述操作为基础，考虑如何得到整棵树。
- 选定一个点为根，先进行一次询问得到一条过根的链。

M. Put Happiness in your Pocket

- 对于两条不同的路径，如果他们有一个共同的端点，我们可以得到一个“大致”的形态，也就是两条路径交的部分在链的一侧，然后分叉出两条路径独有的部分。
- 以上述操作为基础，考虑如何得到整棵树。
- 选定一个点为根，先进行一次询问得到一条过根的链。
- 接下来，每一次将一个没有出现的点设为 $n-1$ 并询问，然后维护这个“大致”的树形态。

M. Put Happiness in your Pocket

- 对于两条不同的路径，如果他们有一个共同的端点，我们可以得到一个“大致”的形态，也就是两条路径交的部分在链的一侧，然后分叉出两条路径独有的部分。
- 以上述操作为基础，考虑如何得到整棵树。
- 选定一个点为根，先进行一次询问得到一条过根的链。
- 接下来，每一次将一个没有出现的点设为 $n-1$ 并询问，然后维护这个“大致”的树形态。
- 换句话说，我们将一条链缩成了一个点，并边询问边维护这些缩点后的点构成的树。

M. Put Happiness in your Pocket

- 所有点都出现过一次后就可以得到若干个形成一条链的点缩成一个点后整棵树的形态。接下来，我们需要确定一条链内的点的顺序。

M. Put Happiness in your Pocket

- 所有点都出现过一次后就可以得到若干个形成一条链的点缩成一个点后整棵树的形态。接下来，我们需要确定一条链内的点的顺序。
- 如果一条链长度大于等于 3，则必然有相邻两个度数不超过 2 的点，也就是说无法确定树形态。

M. Put Happiness in your Pocket

- 所有点都出现过一次后就可以得到若干个形成一条链的点缩成一个点后整棵树的形态。接下来，我们需要确定一条链内的点的顺序。
- 如果一条链长度大于等于 3，则必然有相邻两个度数不超过 2 的点，也就是说无法确定树形态。
- 否则，考虑求出该链深度较深的点的编号。

M. Put Happiness in your Pocket

- 所有点都出现过一次后就可以得到若干个形成一条链的点缩成一个点后整棵树的形态。接下来，我们需要确定一条链内的点的顺序。
- 如果一条链长度大于等于 3，则必然有相邻两个度数不超过 2 的点，也就是说无法确定树形态。
- 否则，考虑求出该链深度较深的点的编号。
- 显然，该链缩成的点至少有两个儿子，在这两个儿子中各选一个点设为 $n, n-1$，则询问出来的链必然包含该链深度较深的点且不包含该链深度较浅的点。

M. Put Happiness in your Pocket

- 然而，根节点所在的链可能长度等于 3（根节点度数为 2 且两边的点度数大于 2）。由于无法分清链缩成的点的每个儿子是在哪一侧，因此无法确定这三个点的顺序。

M. Put Happiness in your Pocket

- 然而，根节点所在的链可能长度等于 3（根节点度数为 2 且两边的点度数大于 2）。由于无法分清链缩成的点的每个儿子是在哪一侧，因此无法确定这三个点的顺序。
- 希望通过找到一个更好的根节点（或者说，度数不为 2 的）来规避这种情况。

M. Put Happiness in your Pocket

- 然而，根节点所在的链可能长度等于 3（根节点度数为 2 且两边的点度数大于 2）。由于无法分清链缩成的点的每个儿子是在哪一侧，因此无法确定这三个点的顺序。
- 希望通过找到一个更好的根节点（或者说，度数不为 2 的）来规避这种情况。
- 考虑先进行两次询问，得到两条链。将两条链的不出现在另一条链的部分各选一个点并查询出第三条链。此时，同时在三条链上的点有且仅有一个且度数大于 2。以这个点为根进行询问即可。

M. Put Happiness in your Pocket

- 然而，根节点所在的链可能长度等于 3（根节点度数为 2 且两边的点度数大于 2）。由于无法分清链缩成的点的每个儿子是在哪一侧，因此无法确定这三个点的顺序。
- 希望通过找到一个更好的根节点（或者说，度数不为 2 的）来规避这种情况。
- 考虑先进行两次询问，得到两条链。将两条链的不出现在另一条链的部分各选一个点并查询出第三条链。此时，同时在三条链上的点有且仅有一个且度数大于 2。以这个点为根进行询问即可。
- 时间复杂度根据实现不同在 $O(n^2)$ 到 $O(n^3)$ 之间。由于每一次询问可以将原来一个部分拆分为至少两个部分，因此操作次数至多为 $n-1$。