


讲选题杂

4182_543_731

2021 年 5 月 8 日

- 因为我很菜，所以这里面基本上都是良心题。
- 12 ~ 15 题，上午 7 题下午 5 题，还有 3 道额外的备选题，20 ~ 40 min 一题。题目可能有梯度或者弧度。
- 数据范围或者限制后面有括号的，表示括号内的范围为部分分的范围。
- 因为某些原因，题目以 DP 题为主。
- 因为某些原因，这里面大多数题目没有公开的提交方式/cy
- 不要喷讲题人/kel
- 

给一个 n 个点 m 条边的 DAG，点 1 的入度为 0。随后向图中再加入一条有向边，加边后图可能不再是 DAG。

求出图中有多少个 $n - 1$ 条有向边的集合，满足只使用集合中的边能从 1 到达其它所有点（即有向生成树），模 $10^9 + 7$

$n \leq 10^5, m \leq 2 \times 10^5$

1s, 128MB

首先考虑不加入边的答案，因为图是一个 DAG，因此给除去 1 外的每个点任选一条入边，一定可以得到合法的方案。

首先考虑不加入边的答案，因为图是一个 DAG，因此给除去 1 外的每个点任选一条入边，一定可以得到合法的方案。

记点 i 的入度为 in_i ，这种情况的答案为 $\prod_{i=2}^n in_i$ 。

首先考虑不加入边的答案，因为图是一个 DAG，因此给除去 1 外的每个点任选一条入边，一定可以得到合法的方案。

记点 i 的入度为 in_i ，这种情况的答案为 $\prod_{i=2}^n in_i$ 。

再考虑选了加入边的情况的答案，设加入边为 (s, t) ，考虑给除去 1, t 之外的每个点选定一条入边。

首先考虑不加入边的答案，因为图是一个 DAG，因此给除去 1 外的每个点任选一条入边，一定可以得到合法的方案。

记点 i 的入度为 in_i ，这种情况的答案为 $\prod_{i=2}^n in_i$ 。

再考虑选了加入边的情况的答案，设加入边为 (s, t) ，考虑给除去 1, t 之外的每个点选定一条入边。

但此时可能出现环，出现环当且仅当给每个点选定入边之后能从 t 到达 s 。考虑算出不合法的方案数，可以发现此时 t 的入边不影响是否不合法，因此可以先加上 t 的入边，最后再除以 in_t 。此时问题变为求有多少个在不加边的图中选有向生成树的方案能从 t 到达 s 。

首先考虑不加入边的答案，因为图是一个 DAG，因此给除去 1 外的每个点任选一条入边，一定可以得到合法的方案。

记点 i 的入度为 in_i ，这种情况的答案为 $\prod_{i=2}^n in_i$ 。

再考虑选了加入边的情况的答案，设加入边为 (s, t) ，考虑给除去 1, t 之外的每个点选定一条入边。

但此时可能出现环，出现环当且仅当给每个点选定入边之后能从 t 到达 s 。考虑算出不合法的方案数，可以发现此时 t 的入边不影响是否不合法，因此可以先加上 t 的入边，最后再除以 in_t 。此时问题变为求有多少个在不加边的图中选有向生成树的方案能从 t 到达 s 。

考虑枚举 t 到 s 的链，则这个值等于：

首先考虑不加入边的答案，因为图是一个 DAG，因此给除去 1 外的每个点任选一条入边，一定可以得到合法的方案。

记点 i 的入度为 in_i ，这种情况的答案为 $\prod_{i=2}^n in_i$ 。

再考虑选了加入边的情况的答案，设加入边为 (s, t) ，考虑给除去 1, t 之外的每个点选定一条入边。

但此时可能出现环，出现环当且仅当给每个点选定入边之后能从 t 到达 s 。考虑算出不合法的方案数，可以发现此时 t 的入边不影响是否不合法，因此可以先加上 t 的入边，最后再除以 in_t 。此时问题变为求有多少个在不加边的图中选有向生成树的方案能从 t 到达 s 。

考虑枚举 t 到 s 的链，则这个值等于：

$$\sum_{a_1, a_2, \dots, a_k, (t, a_1, a_2, \dots, a_k, s) \text{ is a path in DAG}} \frac{\prod_{i=2}^n in_i}{in_s * \prod_{i=1}^k in_{a_i}}$$

记

$$dp_x = \sum_{a_1, a_2, \dots, a_k, (t, a_1, a_2, \dots, a_k, x) \text{ is a path in DAG}} \frac{\prod_{i=2}^n in_i}{in_s * \prod_{i=1}^k in_{a_i}}$$

记

$$dp_x = \sum_{a_1, a_2, \dots, a_k, (t, a_1, a_2, \dots, a_k, x) \text{ is a path in DAG}} \frac{\prod_{i=2}^n in_i}{in_s * \prod_{i=1}^k in_{a_i}}$$

则可以发现：

记

$$dp_x = \sum_{a_1, a_2, \dots, a_k, (t, a_1, a_2, \dots, a_k, x) \text{ is a path in DAG}} \frac{\prod_{i=2}^n in_i}{in_s * \prod_{i=1}^k in_{a_i}}$$

则可以发现：

$$dp_x = \frac{1}{in_x} * \sum_{(i, x)} dp_i$$

记

$$dp_x = \sum_{a_1, a_2, \dots, a_k, (t, a_1, a_2, \dots, a_k, x) \text{ is a path in DAG}} \frac{\prod_{i=2}^n in_i}{in_s * \prod_{i=1}^k in_{a_i}}$$

则可以发现：

$$dp_x = \frac{1}{in_x} * \sum_{(i, x)} dp_i$$

直接 DAG 上 DP 即可，复杂度 $O(n)$

给一棵 n 个点的有根树，定义 $f(x)$ 表示 x 的子树内的点到它的最大距离。

q 次询问，每次给出 x, l, r ，表示将 x 设为根，然后询问 l 到 r 的链上所有点的 $f(x)$ 的异或和。

$n, q \leq 10^6$

2s, 512MB

首先考虑直径长度为偶数的情况，设直径长度为 $2l$ 。考虑以直径中点 u 为根，求出此时每个点的 $f(i)$ 。

首先考虑直径长度为偶数的情况，设直径长度为 $2l$ 。考虑以直径中点 u 为根，求出此时每个点的 $f(i)$ 。
假设当前根变为了 x ，考虑所有 $f(i)$ 的变化。

首先考虑直径长度为偶数的情况，设直径长度为 $2l$ 。考虑以直径中点 u 为根，求出此时每个点的 $f(i)$ 。

假设当前根变为了 x ，考虑所有 $f(i)$ 的变化。

对于不在 x 到 u 路径上的点，换根之后它们的子树不变，因此它们的 $f(i)$ 不变。

首先考虑直径长度为偶数的情况，设直径长度为 $2l$ 。考虑以直径中点 u 为根，求出此时每个点的 $f(i)$ 。

假设当前根变为了 x ，考虑所有 $f(i)$ 的变化。

对于不在 x 到 u 路径上的点，换根之后它们的子树不变，因此它们的 $f(i)$ 不变。

而对于在 x 到 u 路径上的点 v ， v 的子树分为两部分，第一部分为之前现在都是它的子树，第二部分是 u 所在的子树。

首先考虑直径长度为偶数的情况，设直径长度为 $2l$ 。考虑以直径中点 u 为根，求出此时每个点的 $f(i)$ 。

假设当前根变为了 x ，考虑所有 $f(i)$ 的变化。

对于不在 x 到 u 路径上的点，换根之后它们的子树不变，因此它们的 $f(i)$ 不变。

而对于在 x 到 u 路径上的点 v ， v 的子树分为两部分，第一部分为之前现在都是它的子树，第二部分是 u 所在的子树。

根据直径的性质，第一部分内的深度不会超过 l ，而第二部分的深度为 $l + \text{dis}(u, v)$ ，因此一定有 $f(v) = l + \text{dis}(u, v)$ 。

因此问题变为，将 u 到 x 的路径上的权值暂时改为 $l, l+1, \dots, l+dis(u, x)$ ，然后求路径上的所有点权值异或和。

因此问题变为，将 u 到 x 的路径上的权值暂时改为 $l, l+1, \dots, l+dis(u, x)$ ，然后求路径上的所有点权值异或和。求出路径与 u 到 x 路径的交，即可简单维护。

因此问题变为，将 u 到 x 的路径上的权值暂时改为 $l, l+1, \dots, l+dis(u, x)$ ，然后求路径上的所有点权值异或和。求出路径与 u 到 x 路径的交，即可简单维护。根据异或的性质，也可以把询问路径拆成询问一个点 v 到根的路径，这时只要求 $LCA(u, v)$ 即可。

因此问题变为, 将 u 到 x 的路径上的权值暂时改为 $l, l+1, \dots, l+dis(u, x)$, 然后求路径上的所有点权值异或和。求出路径与 u 到 x 路径的交, 即可简单维护。根据异或的性质, 也可以把询问路径拆成询问一个点 v 到根的路径, 这时只需要求 $LCA(u, v)$ 即可。直径中点在边上的情况类似。

因此问题变为, 将 u 到 x 的路径上的权值暂时改为 $l, l+1, \dots, l+dis(u, x)$, 然后求路径上的所有点权值异或和。
求出路径与 u 到 x 路径的交, 即可简单维护。
根据异或的性质, 也可以把询问路径拆成询问一个点 v 到根的路径, 这时只需要求 $LCA(u, v)$ 即可。
直径中点在边上的情况类似。
复杂度 $O(n \log n)$

有 n 个只包含小写字符的字符串 s_1, s_2, \dots, s_n , 你可以把它们按任意顺序拼接起来。

定义一个字符串是好的, 当且仅当它有偶数种本质不同的子序列 (计算空串)。

求出 $n!$ 种拼接方式中满足得到的字符串是好的的方案数。

$$n \leq 20, |s_i| \leq 10^4$$

2s, 512MB

考虑计算子序列的一个 dp :

考虑计算子序列的一个 dp :

记录 dp_c 表示以 c 结尾的子序列数量, su 表示整体字符串数量 (包含空串)。初始时 $su = 1, dp_{a,b,\dots,z} = 0$ 。

考虑计算子序列的一个 dp :

记录 dp_c 表示以 c 结尾的子序列数量, su 表示整体字符串数量 (包含空串)。初始时 $su = 1, dp_{a,b,\dots,z} = 0$ 。

考虑向字符串末尾加入一个 c , 此时可以发现

$dp'_c = su, su' = 2 * su - dp_c$ 。因此, 在模 2 意义下, 这相当于交换 dp_c 与 su 。

考虑计算子序列的一个 dp :

记录 dp_c 表示以 c 结尾的子序列数量, su 表示整体字符串数量 (包含空串)。初始时 $su = 1, dp_{a,b,\dots,z} = 0$ 。

考虑向字符串末尾加入一个 c , 此时可以发现

$dp'_c = su, su' = 2 * su - dp_c$ 。因此, 在模 2 意义下, 这相当于交换 dp_c 与 su 。

因此每个时刻正好有一个值模 2 为 1, 状态只有 27 种。可以对于每个串处理出如果之前的串状态为 $state$, 将这个串加在末尾后的状态。

考虑计算子序列的一个 dp :

记录 dp_c 表示以 c 结尾的子序列数量, su 表示整体字符串数量 (包含空串)。初始时 $su = 1, dp_{a,b,\dots,z} = 0$ 。

考虑向字符串末尾加入一个 c , 此时可以发现

$dp'_c = su, su' = 2 * su - dp_c$ 。因此, 在模 2 意义下, 这相当于交换 dp_c 与 su 。

因此每个时刻正好有一个值模 2 为 1, 状态只有 27 种。可以对于每个串处理出如果之前的串状态为 $state$, 将这个串加在末尾后的状态。

然后设 $dp_{S,state}$ 表示用了集合 S 中的串拼接, 当前 dp 状态为 $state$ 的方案数, 直接做即可。

考虑计算子序列的一个 dp :

记录 dp_c 表示以 c 结尾的子序列数量, su 表示整体字符串数量 (包含空串)。初始时 $su = 1, dp_{a,b,\dots,z} = 0$ 。

考虑向字符串末尾加入一个 c , 此时可以发现

$dp'_c = su, su' = 2 * su - dp_c$ 。因此, 在模 2 意义下, 这相当于交换 dp_c 与 su 。

因此每个时刻正好有一个值模 2 为 1, 状态只有 27 种。可以对于每个串处理出如果之前的串状态为 $state$, 将这个串加在末尾后的状态。

然后设 $dp_{S,state}$ 表示用了集合 S 中的串拼接, 当前 dp 状态为 $state$ 的方案数, 直接做即可。

复杂度 $O(|\sum| (2^n + \sum |s_i|))$

有一个 n 个点的完全图，每条边有一个颜色。

这个完全图满足，对于任意一个简单环，都存在环上两条相邻的边颜色相同。

对于一个点集 S ，定义 $f(S)$ 为最大的 T 的大小，满足 $T \subset S$ 且 T 中任意两点间边的颜色相同。

求 $(\sum_{S \subset \{1,2,\dots,n\}} f(S)) \bmod (10^9 + 7)$ 。

$n \leq 3000$ (300)

1s, 512MB

分析性质可以得到一个结论：

分析性质可以得到一个结论：

对于一个满足条件的图，图中一定存在一个点，满足这个点连出的所有边颜色相同。

分析性质可以得到一个结论：

对于一个满足条件的图，图中一定存在一个点，满足这个点连出的所有边颜色相同。

证明应该有很多种方式，这里讲一个我的垃圾证明方式。

分析性质可以得到一个结论：

对于一个满足条件的图，图中一定存在一个点，满足这个点连出的所有边颜色相同。

证明应该有很多种方式，这里讲一个我的垃圾证明方式。

考虑对 n 归纳, 假设 $n = m - 1$ 时已经成立, 考虑 $n = m$ 的情况。此时, 前 $m - 1$ 个点满足性质, 不妨设点 i 连向 $i + 1, \dots, m - 1$ 的边颜色全部相同, 且点 1 连出的边的颜色为 a 。

考虑对 n 归纳, 假设 $n = m - 1$ 时已经成立, 考虑 $n = m$ 的情况。此时, 前 $m - 1$ 个点满足性质, 不妨设点 i 连向 $i + 1, \dots, m - 1$ 的边颜色全部相同, 且点 1 连出的边的颜色为 a 。

若前 $m - 1$ 个点连出的边颜色都为 a , 此时点 m 不能连出颜色为 a 的边。如果连出两条颜色不同且都不为 a 的边 b, c , 则这三条边不满足条件。因此它连出的边颜色只能相同, 矛盾。

考虑对 n 归纳, 假设 $n = m - 1$ 时已经成立, 考虑 $n = m$ 的情况。此时, 前 $m - 1$ 个点满足性质, 不妨设点 i 连向 $i + 1, \dots, m - 1$ 的边颜色全部相同, 且点 1 连出的边的颜色为 a 。

若前 $m - 1$ 个点连出的边颜色都为 a , 此时点 m 不能连出颜色为 a 的边。如果连出两条颜色不同且都不为 a 的边 b, c , 则这三条边不满足条件。因此它连出的边颜色只能相同, 矛盾。

否则, 点 m 连向 1 的边颜色不能为 a , 设这条边颜色为 b 。因为点 1 连向点 $2, 3, \dots, m - 1$ 的边颜色都为 a , 因此点 m 连向 $2, 3, \dots, m - 1$ 的边颜色只能为 a 或 b , 否则不合题意。

考虑对 n 归纳, 假设 $n = m - 1$ 时已经成立, 考虑 $n = m$ 的情况。此时, 前 $m - 1$ 个点满足性质, 不妨设点 i 连向 $i + 1, \dots, m - 1$ 的边颜色全部相同, 且点 1 连出的边的颜色为 a 。

若前 $m - 1$ 个点连出的边颜色都为 a , 此时点 m 不能连出颜色为 a 的边。如果连出两条颜色不同且都不为 a 的边 b, c , 则这三条边不满足条件。因此它连出的边颜色只能相同, 矛盾。

否则, 点 m 连向 1 的边颜色不能为 a , 设这条边颜色为 b 。因为点 1 连向点 $2, 3, \dots, m - 1$ 的边颜色都为 a , 因此点 m 连向 $2, 3, \dots, m - 1$ 的边颜色只能为 a 或 b , 否则不合题意。

此时 m 一定连出一条颜色为 a 的边, 设这条边为 x , 此时:

考虑对 n 归纳, 假设 $n = m - 1$ 时已经成立, 考虑 $n = m$ 的情况。此时, 前 $m - 1$ 个点满足性质, 不妨设点 i 连向 $i + 1, \dots, m - 1$ 的边颜色全部相同, 且点 1 连出的边的颜色为 a 。

若前 $m - 1$ 个点连出的边颜色都为 a , 此时点 m 不能连出颜色为 a 的边。如果连出两条颜色不同且都不为 a 的边 b, c , 则这三条边不满足条件。因此它连出的边颜色只能相同, 矛盾。

否则, 点 m 连向 1 的边颜色不能为 a , 设这条边颜色为 b 。因为点 1 连向点 $2, 3, \dots, m - 1$ 的边颜色都为 a , 因此点 m 连向 $2, 3, \dots, m - 1$ 的边颜色只能为 a 或 b , 否则不合题意。

此时 m 一定连出一条颜色为 a 的边, 设这条边为 x , 此时:

如果存在 $1 < u < x$, 满足 (u, x) 颜色不为 a , 则选择 $(1, u, x, m)$ 不满足条件。

考虑对 n 归纳, 假设 $n = m - 1$ 时已经成立, 考虑 $n = m$ 的情况。此时, 前 $m - 1$ 个点满足性质, 不妨设点 i 连向 $i + 1, \dots, m - 1$ 的边颜色全部相同, 且点 1 连出的边的颜色为 a 。

若前 $m - 1$ 个点连出的边颜色都为 a , 此时点 m 不能连出颜色为 a 的边。如果连出两条颜色不同且都不为 a 的边 b, c , 则这三条边不满足条件。因此它连出的边颜色只能相同, 矛盾。

否则, 点 m 连向 1 的边颜色不能为 a , 设这条边颜色为 b 。因为点 1 连向点 $2, 3, \dots, m - 1$ 的边颜色都为 a , 因此点 m 连向 $2, 3, \dots, m - 1$ 的边颜色只能为 a 或 b , 否则不合题意。

此时 m 一定连出一条颜色为 a 的边, 设这条边为 x , 此时:

如果存在 $1 < u < x$, 满足 (u, x) 颜色不为 a , 则选择 $(1, u, x, m)$ 不满足条件。

否则 x 连向 $x + 1, \dots, m - 1$ 的边颜色不能为 a , 否则与假设矛盾。此时 $(1, x + 1, x, m)$ 不满足条件。

考虑对 n 归纳, 假设 $n = m - 1$ 时已经成立, 考虑 $n = m$ 的情况。此时, 前 $m - 1$ 个点满足性质, 不妨设点 i 连向 $i + 1, \dots, m - 1$ 的边颜色全部相同, 且点 1 连出的边的颜色为 a 。

若前 $m - 1$ 个点连出的边颜色都为 a , 此时点 m 不能连出颜色为 a 的边。如果连出两条颜色不同且都不为 a 的边 b, c , 则这三条边不满足条件。因此它连出的边颜色只能相同, 矛盾。

否则, 点 m 连向 1 的边颜色不能为 a , 设这条边颜色为 b 。因为点 1 连向点 $2, 3, \dots, m - 1$ 的边颜色都为 a , 因此点 m 连向 $2, 3, \dots, m - 1$ 的边颜色只能为 a 或 b , 否则不合题意。

此时 m 一定连出一条颜色为 a 的边, 设这条边为 x , 此时:

如果存在 $1 < u < x$, 满足 (u, x) 颜色不为 a , 则选择 $(1, u, x, m)$ 不满足条件。

否则 x 连向 $x + 1, \dots, m - 1$ 的边颜色不能为 a , 否则与假设矛盾。此时 $(1, x + 1, x, m)$ 不满足条件。

因此假设不成立, 原结论成立。

因此, 可以求出一个顺序 p_1, p_2, \dots, p_n , 满足 p_i 连向 p_{i+1}, \dots, p_n 的颜色都相同, 设这个颜色为 cl_i 。

因此, 可以求出一个顺序 p_1, p_2, \dots, p_n , 满足 p_i 连向 p_{i+1}, \dots, p_n 的颜色都相同, 设这个颜色为 cl_i 。

可以发现, 对于一个集合 $\{p_{a_1}, p_{a_2}, \dots, p_{a_k}\} (a_1 < a_2 < \dots < a_k)$, 集合中任意两点间边颜色相同当且仅当 $cl_{a_1} = cl_{a_2} = \dots = cl_{a_{k-1}}$ 。

因此，可以求出一个顺序 p_1, p_2, \dots, p_n ，满足 p_i 连向 p_{i+1}, \dots, p_n 的颜色都相同，设这个颜色为 cl_i 。

可以发现，对于一个集合 $\{p_{a_1}, p_{a_2}, \dots, p_{a_k}\} (a_1 < a_2 < \dots < a_k)$ ，集合中任意两点间边颜色相同当且仅当 $cl_{a_1} = cl_{a_2} = \dots = cl_{a_{k-1}}$ 。

此时可以发现， a_k 取最后一个元素一定不劣。因此一个集合的 S 即为除去顺序中最后一个的元素后剩余的中 cl 出现次数的最大值。

因此, 可以求出一个顺序 p_1, p_2, \dots, p_n , 满足 p_i 连向 p_{i+1}, \dots, p_n 的颜色都相同, 设这个颜色为 cl_i 。

可以发现, 对于一个集合 $\{p_{a_1}, p_{a_2}, \dots, p_{a_k}\} (a_1 < a_2 < \dots < a_k)$, 集合中任意两点间边颜色相同当且仅当 $cl_{a_1} = cl_{a_2} = \dots = cl_{a_{k-1}}$ 。

此时可以发现, a_k 取最后一个元素一定不劣。因此一个集合的 S 即为除去顺序中最后一个的元素后剩余的中 cl 出现次数的最大值。

因此可以看成维护一个可重集 S , 做如下操作:

因此, 可以求出一个顺序 p_1, p_2, \dots, p_n , 满足 p_i 连向 p_{i+1}, \dots, p_n 的颜色都相同, 设这个颜色为 cl_i 。

可以发现, 对于一个集合 $\{p_{a_1}, p_{a_2}, \dots, p_{a_k}\} (a_1 < a_2 < \dots < a_k)$, 集合中任意两点间边颜色相同当且仅当 $cl_{a_1} = cl_{a_2} = \dots = cl_{a_{k-1}}$ 。

此时可以发现, a_k 取最后一个元素一定不劣。因此一个集合的 S 即为除去顺序中最后一个的元素后剩余的中 cl 出现次数的最大值。

因此可以看成维护一个可重集 S , 做如下操作:

1. 向集合中加入一个数。
2. 对于集合的每个子集, 求和子集中众数出现次数。

因此，可以求出一个顺序 p_1, p_2, \dots, p_n ，满足 p_i 连向 p_{i+1}, \dots, p_n 的颜色都相同，设这个颜色为 cl_i 。

可以发现，对于一个集合 $\{p_{a_1}, p_{a_2}, \dots, p_{a_k}\} (a_1 < a_2 < \dots < a_k)$ ，集合中任意两点间边颜色相同当且仅当 $cl_{a_1} = cl_{a_2} = \dots = cl_{a_{k-1}}$ 。

此时可以发现， a_k 取最后一个元素一定不劣。因此一个集合的 S 即为除去顺序中最后一个的元素后剩余的中 cl 出现次数的最大值。

因此可以看成维护一个可重集 S ，做如下操作：

1. 向集合中加入一个数。
2. 对于集合的每个子集，求和子集中众数出现次数。

设 i 出现了 ct_i 次，总共有 n 个元素，考虑算出众数出现次数不超过 t 的子集个数，则可以发现答案为：

因此，可以求出一个顺序 p_1, p_2, \dots, p_n ，满足 p_i 连向 p_{i+1}, \dots, p_n 的颜色都相同，设这个颜色为 cl_i 。

可以发现，对于一个集合 $\{p_{a_1}, p_{a_2}, \dots, p_{a_k}\} (a_1 < a_2 < \dots < a_k)$ ，集合中任意两点间边颜色相同当且仅当 $cl_{a_1} = cl_{a_2} = \dots = cl_{a_{k-1}}$ 。

此时可以发现， a_k 取最后一个元素一定不劣。因此一个集合的 S 即为除去顺序中最后一个元素后剩余的 cl 出现次数的最大值。

因此可以看成维护一个可重集 S ，做如下操作：

1. 向集合中加入一个数。
2. 对于集合的每个子集，求和子集中众数出现次数。

设 i 出现了 ct_i 次，总共有 n 个元素，考虑算出众数出现次数不超过 t 的子集个数，则可以发现答案为：

$$\sum_{t=0}^n (2^n - \prod_{i=1}^n (\sum_{j=0}^t c_{ct_i}^j))$$

维护每一个 $\prod_{i=1}^n (\sum_{j=0}^t c_{ct_i}^j)$, 加入一个 i 时只需要更新所有 t 的 $\sum_{j=0}^t c_{ct_i}^j$ 即可。

维护每一个 $\prod_{i=1}^n (\sum_{j=0}^t c_{ct_i}^j)$, 加入一个 i 时只需要更新所有 t 的 $\sum_{j=0}^t c_{ct_i}^j$ 即可。
预处理需要的各种前缀和和逆元即可做到 $O(n^2)$

给出 n 个字符串 s_1, \dots, s_n , 你需要在每个字符串中选出一个可以为空的子串 t_1, \dots, t_n , 将它们顺序拼接得到 $T = t_1 + t_2 + \dots + t_n$ 。

求出可以得到的不同的非空 T 数量, 模 998244353。

$n \leq 10^5, \sum_{i=1}^n |s_i| \leq 3 \times 10^5$, 字符集大小为 $3 \times 10^5(10)$ 。

2s, 512MB

首先考虑一个 $O(n|\Sigma|)$ 的做法:

首先考虑一个 $O(n|\Sigma|)$ 的做法：
考虑判断一个 T 是否能被构造出来，显然贪心匹配，每次在当前串中选尽量长的一段是最优的。

首先考虑一个 $O(n|\Sigma|)$ 的做法:

考虑判断一个 T 是否能被构造出来, 显然贪心匹配, 每次在当前串中选尽量长的一段是最优的。

对于每个串建出 SAM, 在 SAM 上沿着 ch 转移, 如果当前 ch 不存在, 则说明无法在这个串中选更长的, 此时一定需要在之后的串中再选。

首先考虑一个 $O(n|\Sigma|)$ 的做法:

考虑判断一个 T 是否能被构造出来, 显然贪心匹配, 每次在当前串中选尽量长的一段是最优的。

对于每个串建出 SAM, 在 SAM 上沿着 ch 转移, 如果当前 ch 不存在, 则说明无法在这个串中选更长的, 此时一定需要在之后的串中再选。

设当前字符为 c , 则找到下一个能转移 c 的 SAM, 从这个位置开始转移即可。

首先考虑一个 $O(n|\Sigma|)$ 的做法：

考虑判断一个 T 是否能被构造出来，显然贪心匹配，每次在当前串中选尽量长的一段是最优的。

对于每个串建出 SAM，在 SAM 上沿着 ch 转移，如果当前 ch 不存在，则说明无法在这个串中选更长的，此时一定需要在之后的串中再选。

设当前字符为 c ，则找到下一个能转移 c 的 SAM，从这个位置开始转移即可。

这样可以得到一个大的 DAG，可以直接 DP 算答案，这样的复杂度为 $O((\sum_{i=1}^n |s_i|)|\Sigma|)$ 。

考虑一条转移到之后的 SAM 的边的意义。

考虑一条转移到之后的 SAM 的边的意义。
可以发现，它求的实际上是使用之后的串，能构造出多少个以 c 开头的字符串。

考虑一条转移到之后的 SAM 的边的意义。

可以发现，它求的实际上是使用之后的串，能构造出多少个以 c 开头的字符串。

从后向前考虑每个串，记 vl_c 表示使用后面的串能构造出多少个以 c 开头的字符串。考虑在左侧加入一个串时，这个串上的 dp 转移：

考虑一条转移到之后的 SAM 的边的意义。

可以发现，它求的实际上是使用之后的串，能构造出多少个以 c 开头的字符串。

从后向前考虑每个串，记 vl_c 表示使用后面的串能构造出多少个以 c 开头的字符串。考虑在左侧加入一个串时，这个串上的 dp 转移：

对于点 u ，如果 $ch_{u,c}$ 存在，则 $dp_{u+} = dp_{ch_{u,c}}$ ，否则 $dp_{u+} = vl_c$ 。

考虑一条转移到之后的 SAM 的边的意义。

可以发现，它求的实际上是使用之后的串，能构造出多少个以 c 开头的字符串。

从后向前考虑每个串，记 vl_c 表示使用后面的串能构造出多少个以 c 开头的字符串。考虑在左侧加入一个串时，这个串上的 dp 转移：

对于点 u ，如果 $ch_{u,c}$ 存在，则 $dp_u + = dp_{ch_{u,c}}$ ，否则 $dp_u + = vl_c$ 。

只需要记录 vl 的整体和，就可以在 $O(n)$ 的时间内求出这个 SAM 中所有点的答案。

考虑一条转移到之后的 SAM 的边的意义。

可以发现，它求的实际上是使用之后的串，能构造出多少个以 c 开头的字符串。

从后向前考虑每个串，记 vl_c 表示使用后面的串能构造出多少个以 c 开头的字符串。考虑在左侧加入一个串时，这个串上的 dp 转移：

对于点 u ，如果 $ch_{u,c}$ 存在，则 $dp_u + = dp_{ch_{u,c}}$ ，否则 $dp_u + = vl_c$ 。

只需要记录 vl 的整体和，就可以在 $O(n)$ 的时间内求出这个 SAM 中所有点的答案。

然后再更新 vl ，只更新这个串中出现的字符即可。

考虑一条转移到之后的 SAM 的边的意义。

可以发现，它求的实际上是使用之后的串，能构造出多少个以 c 开头的字符串。

从后向前考虑每个串，记 vl_c 表示使用后面的串能构造出多少个以 c 开头的字符串。考虑在左侧加入一个串时，这个串上的 dp 转移：

对于点 u ，如果 $ch_{u,c}$ 存在，则 $dp_u + = dp_{ch_{u,c}}$ ，否则 $dp_u + = vl_c$ 。

只需要记录 vl 的整体和，就可以在 $O(n)$ 的时间内求出这个 SAM 中所有点的答案。

然后再更新 vl ，只更新这个串中出现的字符即可。

复杂度 $O((\sum_{i=1}^n |s_i|) \log m)$

交互题。

你有一棵 n 个点的有根树，根为 1。

对于每个点，你可以钦定它的一条连向儿子的边为重边，也可以不钦定。交互库首先给出这棵树，你需要给出一个钦定重边的方案。

随后交互库会进行 q 次询问，每次询问给定 x 。你可以在方案中进行若干次调整，使得 x 到根的路径上均为重边。一次调整定义为：

选择一个点，如果这个点没有钦定向儿子的重边，你可以钦定一条向儿子的边为重边。

如果这个点有钦定向儿子的重边，则你可以取消这个钦定，使这个点向儿子的边都不是重边。

你可以先进行若干次调整，使得 x 到根的路径上均为重边。在满足条件之后，你还可以进行若干次调整。这次询问中你的代价为两部分调整的次数的和。

你需要满足，在所有询问中，你单次询问的代价最大值不超过 35 (42, 60)。

$$n \leq 5 \times 10^4, q \leq 5 \times 10^5$$

3s, 512MB

考虑一个暴力想法：

考虑一个暴力想法：
对树轻重链剖分，将每个点的重边钦定为剖分中的重儿子。

考虑一个暴力想法：

对树轻重链剖分，将每个点的重边钦定为剖分中的重儿子。

每次询问暴力把经过的轻边进行调整，然后再调整回去。

考虑一个暴力想法：

对树轻重链剖分，将每个点的重边钦定为剖分中的重儿子。

每次询问暴力把经过的轻边进行调整，然后再调整回去。

这样一条轻边需要总共调整 4 次，因此需要 $4 * \log n = 60$ 步。

考虑一个暴力想法：

对树轻重链剖分，将每个点的重边钦定为剖分中的重儿子。

每次询问暴力把经过的轻边进行调整，然后再调整回去。

这样一条轻边需要总共调整 4 次，因此需要 $4 * \log n = 60$ 步。

注意到对于一个点它也可以不钦定儿子，每次询问经过这个点时钦定，这样任何一个儿子在这里都需要 2 步。

考虑一个暴力想法：

对树轻重链剖分，将每个点的重边钦定为剖分中的重儿子。

每次询问暴力把经过的轻边进行调整，然后再调整回去。

这样一条轻边需要总共调整 4 次，因此需要 $4 * \log n = 60$ 步。

注意到对于一个点它也可以不钦定儿子，每次询问经过这个点时钦定，这样任何一个儿子在这里都需要 2 步。

因此可以考虑，如果一个点重儿子太大的情况用之前的方式，否则用这个方式。

考虑一个暴力想法：

对树轻重链剖分，将每个点的重边钦定为剖分中的重儿子。

每次询问暴力把经过的轻边进行调整，然后再调整回去。

这样一条轻边需要总共调整 4 次，因此需要 $4 * \log n = 60$ 步。

注意到对于一个点它也可以不钦定儿子，每次询问经过这个点时钦定，这样任何一个儿子在这里都需要 2 步。

因此可以考虑，如果一个点重儿子太大的情况用之前的方式，否则用这个方式。

设 dp_i 表示使用这种方式， i 个点的子树最多需要多少次操作。

考虑一个暴力想法：

对树轻重链剖分，将每个点的重边钦定为剖分中的重儿子。

每次询问暴力把经过的轻边进行调整，然后再调整回去。

这样一条轻边需要总共调整 4 次，因此需要 $4 * \log n = 60$ 步。

注意到对于一个点它也可以不钦定儿子，每次询问经过这个点时钦定，这样任何一个儿子在这里都需要 2 步。

因此可以考虑，如果一个点重儿子太大的情况用之前的方式，否则用这个方式。

设 dp_i 表示使用这种方式， i 个点的子树最多需要多少次操作。

那么

$$dp_i = \max_{1 \leq j < i, j \leq i-j-1} \min\{\max(dp_j + 4, dp_{i-j-1}), \max(dp_j + 2, dp_{i-j-1} + 2)\}$$

(1)

考虑一个暴力想法：

对树轻重链剖分，将每个点的重边钦定为剖分中的重儿子。

每次询问暴力把经过的轻边进行调整，然后再调整回去。

这样一条轻边需要总共调整 4 次，因此需要 $4 * \log n = 60$ 步。

注意到对于一个点它也可以不钦定儿子，每次询问经过这个点时钦定，这样任何一个儿子在这里都需要 2 步。

因此可以考虑，如果一个点重儿子太大的情况用之前的方式，否则用这个方式。

设 dp_i 表示使用这种方式， i 个点的子树最多需要多少次操作。

那么

$$dp_i = \max_{1 \leq j < i, j \leq i-j-1} \min\{\max(dp_j + 4, dp_{i-j-1}), \max(dp_j + 2, dp_{i-j-1} + 2)\}$$
(1)

可以得到 $dp_{50000} = 40$

6

考虑一个点上其它的方式。

6

考虑一个点上其它的方式。

对于一个点，它可以钦定重儿子，也可以不钦定。

考虑一个点上其它的方式。

对于一个点，它可以钦定重儿子，也可以不钦定。

如果询问的点在重儿子子树内，则最多需要 1 步钦定重边，然后可以不取消钦定，这部分需要最多 1 步。

6

考虑一个点上其它的方式。

对于一个点，它可以钦定重儿子，也可以不钦定。

如果询问的点在重儿子子树内，则最多需要 1 步钦定重边，然后可以不取消钦定，这部分需要最多 1 步。

如果询问的点不在重儿子子树内，则最多需要 2 步钦定到这个儿子，最后用 1 步取消钦定即可。这部分最多需要 3 步。

考虑一个点上其它的方式。

对于一个点，它可以钦定重儿子，也可以不钦定。

如果询问的点在重儿子子树内，则最多需要 1 步钦定重边，然后可以不取消钦定，这部分需要最多 1 步。

如果询问的点不在重儿子子树内，则最多需要 2 步钦定到这个儿子，最后用 1 步取消钦定即可。这部分最多需要 3 步。

因此这种方式重儿子需要 1 步，轻儿子需要 3 步。因此上面的 dp 可以写成：

考虑一个点上其它的方式。

对于一个点，它可以钦定重儿子，也可以不钦定。

如果询问的点在重儿子子树内，则最多需要 1 步钦定重边，然后可以不取消钦定，这部分需要最多 1 步。

如果询问的点不在重儿子子树内，则最多需要 2 步钦定到这个儿子，最后用 1 步取消钦定即可。这部分最多需要 3 步。

因此这种方式重儿子需要 1 步，轻儿子需要 3 步。因此上面的 dp 可以写成：

$$dp_i = \max_{1 \leq j < i, j \leq i-j-1} \min \{ \max(dp_j + 4, dp_{i-j-1}), \max(dp_j + 3, dp_{i-j-1} + 1), \max(dp_j + 2, dp_{i-j-1} + 2) \} \quad (2)$$

考虑一个点上其它的方式。

对于一个点，它可以钦定重儿子，也可以不钦定。

如果询问的点在重儿子子树内，则最多需要 1 步钦定重边，然后可以不取消钦定，这部分需要最多 1 步。

如果询问的点不在重儿子子树内，则最多需要 2 步钦定到这个儿子，最后用 1 步取消钦定即可。这部分最多需要 3 步。

因此这种方式重儿子需要 1 步，轻儿子需要 3 步。因此上面的 dp 可以写成：

$$dp_i = \max_{1 \leq j < i, j \leq i-j-1} \min \{ \max(dp_j + 4, dp_{i-j-1}), \max(dp_j + 3, dp_{i-j-1} + 1), \max(dp_j + 2, dp_{i-j-1} + 2) \} \quad (2)$$

此时可以发现 $dp_{50000} = 35$ ，因此可以在 35 步内解决问题。

考虑一个点上其它的方式。

对于一个点，它可以钦定重儿子，也可以不钦定。

如果询问的点在重儿子子树内，则最多需要 1 步钦定重边，然后可以不取消钦定，这部分需要最多 1 步。

如果询问的点不在重儿子子树内，则最多需要 2 步钦定到这个儿子，最后用 1 步取消钦定即可。这部分最多需要 3 步。

因此这种方式重儿子需要 1 步，轻儿子需要 3 步。因此上面的 dp 可以写成：

$$dp_i = \max_{1 \leq j < i, j \leq i-j-1} \min \{ \max(dp_j + 4, dp_{i-j-1}), \max(dp_j + 3, dp_{i-j-1} + 1), \max(dp_j + 2, dp_{i-j-1} + 2) \} \quad (2)$$

此时可以发现 $dp_{50000} = 35$ ，因此可以在 35 步内解决问题。

可以先打表出 dp 的分界点，然后在一个点上通过儿子的子树大小决定使用哪种方式，也可以在树上做一遍 dp 。

考虑一个点上其它的方式。

对于一个点，它可以钦定重儿子，也可以不钦定。

如果询问的点在重儿子子树内，则最多需要 1 步钦定重边，然后可以不取消钦定，这部分需要最多 1 步。

如果询问的点不在重儿子子树内，则最多需要 2 步钦定到这个儿子，最后用 1 步取消钦定即可。这部分最多需要 3 步。

因此这种方式重儿子需要 1 步，轻儿子需要 3 步。因此上面的 dp 可以写成：

$$dp_i = \max_{1 \leq j < i, j \leq i-j-1} \min \{ \max(dp_j + 4, dp_{i-j-1}), \max(dp_j + 3, dp_{i-j-1} + 1), \max(dp_j + 2, dp_{i-j-1} + 2) \} \quad (2)$$

此时可以发现 $dp_{50000} = 35$ ，因此可以在 35 步内解决问题。

可以先打表出 dp 的分界点，然后在一个点上通过儿子的子树大小决定使用哪种方式，也可以在树上做一遍 dp 。

复杂度 $O((n+q) \log n)$

Source: 「2020-2021 集训队作业」大鱼治水

给定 N, m, k , 求有多少个正整数序列 h 满足:

1. h 的长度 n 大于 0 不超过 N 。
2. $\forall i \in \{1, 2, \dots, n\}, 1 \leq h_i \leq m$ 。
3. 正好存在 k 个 $i \in \{1, 2, \dots, n-1\}$ 满足 $h_i < h_{i+1}$ 。

答案模 998244353。

$$2 \leq N, m, k \leq 2^{19}, (N - k + 1) * m \leq 2^{20}$$

1s, 512MB

考虑最直接的 dp : $dp_{i,j,k}$ 表示填了前 i 个数, 前面有 j 个位置不满足 $h_i < h_{i+1}$, 最后一个位置的值为 k 。初始状态可以看成 $dp_{0,-1,m} = 1$ 。

考虑最直接的 dp : $dp_{i,j,k}$ 表示填了前 i 个数, 前面有 j 个位置不满足 $h_i < h_{i+1}$, 最后一个位置的值为 k 。初始状态可以看成 $dp_{0,-1,m} = 1$ 。考虑它的转移, 可以发现 $dp_{i,j,k}$ 可以转移到 $dp_{i+1,j,k+1}, dp_{i+1,j,k+2}, \dots, dp_{i+1,j,m}, dp_{i+1,j+1,1}, dp_{i+1,j+1,2}, \dots, dp_{i+1,j+1,k}$ 。

考虑最直接的 dp : $dp_{i,j,k}$ 表示填了前 i 个数, 前面有 j 个位置不满足 $h_i < h_{i+1}$, 最后一个位置的值为 k 。初始状态可以看成 $dp_{0,-1,m} = 1$ 。

考虑它的转移, 可以发现 $dp_{i,j,k}$ 可以转移到

$dp_{i+1,j,k+1}, dp_{i+1,j,k+2}, \dots, dp_{i+1,j,m}, dp_{i+1,j+1,1}, dp_{i+1,j+1,2}, \dots, dp_{i+1,j+1,k}$ 。

如果将后二维压缩在一起, 设 $s = j * m + k$, 则初值为 $dp_{0,0} = 1$, $dp_{i,s}$ 可以转移到 $dp_{i+1,s+1}, dp_{i+1,s+2}, \dots, dp_{i+1,s+m}$ 。

考虑最直接的 dp : $dp_{i,j,k}$ 表示填了前 i 个数, 前面有 j 个位置不满足 $h_i < h_{i+1}$, 最后一个位置的值为 k 。初始状态可以看成 $dp_{0,-1,m} = 1$ 。

考虑它的转移, 可以发现 $dp_{i,j,k}$ 可以转移到

$dp_{i+1,j,k+1}, dp_{i+1,j,k+2}, \dots, dp_{i+1,j,m}, dp_{i+1,j+1,1}, dp_{i+1,j+1,2}, \dots, dp_{i+1,j+1,k}$ 。

如果将后二维压缩在一起, 设 $s = j * m + k$, 则初值为 $dp_{0,0} = 1$, $dp_{i,s}$ 可以转移到 $dp_{i+1,s+1}, dp_{i+1,s+2}, \dots, dp_{i+1,s+m}$ 。

因此将 dp 写成生成函数, 则 $dp_i = (x + x^2 + \dots + x^m)^i$ 。

求的可以看成有 $n - 1 - k$ 个位置不满足 $h_i < h_{i+1}$ 的序列个数。但序列长度 n 难以处理。

求的可以看成有 $n - 1 - k$ 个位置不满足 $h_i < h_{i+1}$ 的序列个数。但序列长度 n 难以处理。

考虑对于一个长度小于 N 的序列，在后面补上 $N - n$ 个位置，这些位置都看成不满足 $h_i < h_{i+1}$ 。这样即为求 $N - 1 - k$ 个位置不满足的序列个数，可以发现此时 dp 的生成函数为：

求的可以看成有 $n - 1 - k$ 个位置不满足 $h_i < h_{i+1}$ 的序列个数。但序列长度 n 难以处理。

考虑对于一个长度小于 N 的序列，在后面补上 $N - n$ 个位置，这些位置都看成不满足 $h_i < h_{i+1}$ 。这样即为求 $N - 1 - k$ 个位置不满足的序列个数，可以发现此时 dp 的生成函数为：

$$\sum_{i=1}^N (x + x^2 + \dots + x^m)^i * (x^m)^{N-i} = \frac{(x + x^2 + \dots + x^m)^{N+1} - (x^m)^{N+1}}{x + x^2 + \dots + x^{m-1}} - x^{Nm}$$

求的可以看成有 $n - 1 - k$ 个位置不满足 $h_i < h_{i+1}$ 的序列个数。但序列长度 n 难以处理。

考虑对于一个长度小于 N 的序列，在后面补上 $N - n$ 个位置，这些位置都看成不满足 $h_i < h_{i+1}$ 。这样即为求 $N - 1 - k$ 个位置不满足的序列个数，可以发现此时 dp 的生成函数为：

$$\sum_{i=1}^N (x + x^2 + \dots + x^m)^i * (x^m)^{N-i} = \frac{(x + x^2 + \dots + x^m)^{N+1} - (x^m)^{N+1}}{x + x^2 + \dots + x^{m-1}} - x^{Nm}$$

而答案即为 $x^{(N-1-k)*m+1}, x^{(N-1-k)*m+2}, \dots, x^{(N-k)*m}$ 项的系数和。

考虑求 $(1 + x + \dots + x^{m-1})^{N+1}$, 设

$F(x) = (1 + x + \dots + x^{m-1})^{N+1}$, $G(x) = 1 + x + \dots + x^{m-1}$, 则

$F(x) = G(x)^{N+1}$ 。

考虑求 $(1 + x + \dots + x^{m-1})^{N+1}$, 设

$F(x) = (1 + x + \dots + x^{m-1})^{N+1}$, $G(x) = 1 + x + \dots + x^{m-1}$, 则

$F(x) = G(x)^{N+1}$ 。

求导后有 $F'(x) = (N+1)G'(x)G(x)^N$, 所以 $F'(x)G(x) = (N+1)G'(x)F(x)$

考虑求 $(1 + x + \dots + x^{m-1})^{N+1}$, 设

$F(x) = (1 + x + \dots + x^{m-1})^{N+1}$, $G(x) = 1 + x + \dots + x^{m-1}$, 则

$F(x) = G(x)^{N+1}$ 。

求导后有 $F'(x) = (N+1)G'(x)G(x)^N$, 所以 $F'(x)G(x) = (N+1)G'(x)F(x)$

设 $F(x) = \sum f_i x^i$, 则比对 x^n 的系数可以得到:

考虑求 $(1 + x + \dots + x^{m-1})^{N+1}$, 设

$F(x) = (1 + x + \dots + x^{m-1})^{N+1}$, $G(x) = 1 + x + \dots + x^{m-1}$, 则

$F(x) = G(x)^{N+1}$ 。

求导后有 $F'(x) = (N+1)G'(x)G(x)^N$, 所以 $F'(x)G(x) = (N+1)G'(x)F(x)$

设 $F(x) = \sum f_i x^i$, 则比对 x^n 的系数可以得到:

$$\sum_{i=0}^{m-1} (n-i+1) f_{n-i+1} = (N+1) \sum_{i=1}^{m-1} i * f_{n-i+1}$$

考虑求 $(1 + x + \dots + x^{m-1})^{N+1}$, 设

$F(x) = (1 + x + \dots + x^{m-1})^{N+1}$, $G(x) = 1 + x + \dots + x^{m-1}$, 则

$F(x) = G(x)^{N+1}$ 。

求导后有 $F'(x) = (N+1)G'(x)G(x)^N$, 所以 $F'(x)G(x) = (N+1)G'(x)F(x)$

设 $F(x) = \sum f_i x^i$, 则比对 x^n 的系数可以得到:

$$\sum_{i=0}^{m-1} (n-i+1) f_{n-i+1} = (N+1) \sum_{i=1}^{m-1} i * f_{n-i+1}$$

于是有:

考虑求 $(1 + x + \dots + x^{m-1})^{N+1}$, 设

$F(x) = (1 + x + \dots + x^{m-1})^{N+1}$, $G(x) = 1 + x + \dots + x^{m-1}$, 则

$F(x) = G(x)^{N+1}$ 。

求导后有 $F'(x) = (N+1)G'(x)G(x)^N$, 所以 $F'(x)G(x) = (N+1)G'(x)F(x)$

设 $F(x) = \sum f_i x^i$, 则比对 x^n 的系数可以得到:

$$\sum_{i=0}^{m-1} (n-i+1) f_{n-i+1} = (N+1) \sum_{i=1}^{m-1} i * f_{n-i+1}$$

于是有:

$$(n+1)f_{n+1} = \sum_{i=1}^{m-1} ((N+2) * i - n - 1) f_{n-i+1}$$

考虑求 $(1 + x + \dots + x^{m-1})^{N+1}$, 设

$F(x) = (1 + x + \dots + x^{m-1})^{N+1}$, $G(x) = 1 + x + \dots + x^{m-1}$, 则

$F(x) = G(x)^{N+1}$ 。

求导后有 $F'(x) = (N+1)G'(x)G(x)^N$, 所以 $F'(x)G(x) = (N+1)G'(x)F(x)$

设 $F(x) = \sum f_i x^i$, 则比对 x^n 的系数可以得到:

$$\sum_{i=0}^{m-1} (n-i+1) f_{n-i+1} = (N+1) \sum_{i=1}^{m-1} i * f_{n-i+1}$$

于是有:

$$(n+1) f_{n+1} = \sum_{i=1}^{m-1} ((N+2) * i - n - 1) f_{n-i+1}$$

维护前缀和即可做到线性。

原式可以看成：

原式可以看成：

$$x^N * \frac{(1 + x + \dots + x^{m-1})^{N+1} - (x^{m-1})^{N+1}}{1 + x + \dots + x^{m-2}} - x^{Nm}$$

原式可以看成：

$$x^N * \frac{(1 + x + \dots + x^{m-1})^{N+1} - (x^{m-1})^{N+1}}{1 + x + \dots + x^{m-2}} - x^{Nm}$$

只需要考虑求逆的过程。分析一下可以得到 $f_i = -\sum_{j=1}^{m-2} f_{i-j}$ ，也可以线性。

原式可以看成：

$$x^N * \frac{(1 + x + \dots + x^{m-1})^{N+1} - (x^{m-1})^{N+1}}{1 + x + \dots + x^{m-2}} - x^{Nm}$$

只需要考虑求逆的过程。分析一下可以得到 $f_i = -\sum_{j=1}^{m-2} f_{i-j}$ ，也可以线性。

因此总复杂度 $O((N-k)m)$ 。

上午和下午的分界线

给出 n, k , 有 n 个物品, 每个物品有重量 c_i 和价值 v_i 。
对于每个 $i \in \{1, 2, \dots, k\}$, 求出选择重量和不超过 i 的物品能得到的最大价值和。

$n \leq 10^6, k \leq 5 \times 10^4, c_i \leq 300$

2s, 256MB

将物品按照重量分类, 设 $dp_{i,j}$ 表示只考虑重量小于等于 i 的物品, 选择的物品重量和不超过 j 时的最大收益。

将物品按照重量分类, 设 $dp_{i,j}$ 表示只考虑重量小于等于 i 的物品, 选择的物品重量和不超过 j 时的最大收益。

考虑加入一种重量时的转移, 设 $f_{i,j}$ 表示选 j 个重量为 i 的物品的最大收益, 则 $dp_{i,j} = \max_k \{dp_{i-1,j-i*k} + f_{i,k}\}$ 。

将物品按照重量分类, 设 $dp_{i,j}$ 表示只考虑重量小于等于 i 的物品, 选择的物品重量和不超过 j 时的最大收益。

考虑加入一种重量时的转移, 设 $f_{i,j}$ 表示选 j 个重量为 i 的物品的最大收益, 则 $dp_{i,j} = \max_k \{dp_{i-1,j-i*k} + f_{i,k}\}$ 。

转移可以按照 $j \bmod k$ 分类, 对于一类可以看成

$$dp_{i,k*j+t} = \max_k \{dp_{i-1,(j-i)*k+t} + f_{i,k}\}$$

将物品按照重量分类, 设 $dp_{i,j}$ 表示只考虑重量小于等于 i 的物品, 选择的物品重量和不超过 j 时的最大收益。

考虑加入一种重量时的转移, 设 $f_{i,j}$ 表示选 j 个重量为 i 的物品的最大收益, 则 $dp_{i,j} = \max_k \{dp_{i-1,j-i*k} + f_{i,k}\}$ 。

转移可以按照 $j \bmod k$ 分类, 对于一类可以看成

$$dp_{i,k*j+t} = \max_k \{dp_{i-1,(j-i)*k+t} + f_{i,k}\}$$

又因为 $f_{i,j}$ 是质量为 i 的物品中最大的 j 个价值的和, 显然对于 $s \leq t$ 有 $f_{i,s+1} + f_{i,t} \geq f_{i,s} + f_{i,t+1}$ 。因此转移满足决策单调性。

将物品按照重量分类，设 $dp_{i,j}$ 表示只考虑重量小于等于 i 的物品，选择的物品重量和不超过 j 时的最大收益。

考虑加入一种重量时的转移，设 $f_{i,j}$ 表示选 j 个重量为 i 的物品的最大收益，则 $dp_{i,j} = \max_k \{dp_{i-1,j-i*k} + f_{i,k}\}$ 。

转移可以按照 $j \bmod k$ 分类，对于一类可以看成

$$dp_{i,k*j+t} = \max_k \{dp_{i-1,(j-i)*k+t} + f_{i,k}\}$$

又因为 $f_{i,j}$ 是质量为 i 的物品中最大的 j 个价值的和，显然对于 $s \leq t$ 有 $f_{i,s+1} + f_{i,t} \geq f_{i,s} + f_{i,t+1}$ 。因此转移满足决策单调性。

复杂度 $O(k * \max c_i * \log k)$

给一个长度为 $n - 1$ 的正整数数组 b ，称一个长度为 n 的正实数数组 a 为合法的，当且仅当它满足 $\forall i \in \{1, 2, \dots, n - 1\}, a_i * a_{i+1} \geq b_i$ 。

求出所有合法的 a 中 $\sum a_i$ 的最小值。

$n \leq 3000, b_i \leq 40000$

1s, 512MB

考虑最优的解 a_i 满足的性质。

考虑最优的解 a_i 满足的性质。

如果 $a_{i-1} * a_i > b_{i-1}$, $a_i * a_{i+1} > b_i$, 则减小 a_i 一定可以得到 $\sum a_i$ 最小的答案。

考虑最优的解 a_i 满足的性质。

如果 $a_{i-1} * a_i > b_{i-1}$, $a_i * a_{i+1} > b_i$, 则减小 a_i 一定可以得到 $\sum a_i$ 最小的答案。

因此在最优解中, 一定可以将序列划分成若干个长度大于 1 的段 $[l_1, r_1], \dots, [l_k, r_k]$, 满足:

考虑最优的解 a_i 满足的性质。

如果 $a_{i-1} * a_i > b_{i-1}$, $a_i * a_{i+1} > b_i$, 则减小 a_i 一定可以得到 $\sum a_i$ 最小的答案。

因此在最优解中, 一定可以将序列划分成若干个长度大于 1 的段 $[l_1, r_1], \dots, [l_k, r_k]$, 满足:

1. $\forall i, j \in [l_i, r_i), a_j * a_{j+1} = b_j$

考虑最优的解 a_i 满足的性质。

如果 $a_{i-1} * a_i > b_{i-1}$, $a_i * a_{i+1} > b_i$, 则减小 a_i 一定可以得到 $\sum a_i$ 最小的答案。

因此在最优解中, 一定可以将序列划分成若干个长度大于 1 的段

$[l_1, r_1], \dots, [l_k, r_k]$, 满足:

1. $\forall i, j \in [l_i, r_i), a_j * a_{j+1} = b_j$
2. $\forall i, a_{r_i} * a_{l_{i+1}} > b_{r_i}$

考虑最优的解 a_i 满足的性质。

如果 $a_{i-1} * a_i > b_{i-1}$, $a_i * a_{i+1} > b_i$, 则减小 a_i 一定可以得到 $\sum a_i$ 最小的答案。

因此在最优解中, 一定可以将序列划分成若干个长度大于 1 的段 $[l_1, r_1], \dots, [l_k, r_k]$, 满足:

1. $\forall i, j \in [l_i, r_i), a_j * a_{j+1} = b_j$
2. $\forall i, a_{r_i} * a_{l_{i+1}} > b_{r_i}$

即段内相邻两数乘积等于限制, 不同段间相邻两数乘积大于限制。

考虑最优的解 a_i 满足的性质。

如果 $a_{i-1} * a_i > b_{i-1}$, $a_i * a_{i+1} > b_i$, 则减小 a_i 一定可以得到 $\sum a_i$ 最小的答案。

因此在最优解中, 一定可以将序列划分成若干个长度大于 1 的段 $[l_1, r_1], \dots, [l_k, r_k]$, 满足:

1. $\forall i, j \in [l_i, r_i), a_j * a_{j+1} = b_j$
2. $\forall i, a_{r_i} * a_{l_{i+1}} > b_{r_i}$

即段内相邻两数乘积等于限制, 不同段间相邻两数乘积大于限制。

对于一段 $[l_i, r_i]$, 显然可以将这一段内的所有数都表示为 $v_1 * a_{l_i}$ 或 $\frac{v_2}{a_{l_i}}$ 的形式。因此这一段数的总和可以被表示为 $x * a_{l_i} + \frac{y}{a_{l_i}}$ 的形式。

考虑最优的解 a_i 满足的性质。

如果 $a_{i-1} * a_i > b_{i-1}$, $a_i * a_{i+1} > b_i$, 则减小 a_i 一定可以得到 $\sum a_i$ 最小的答案。

因此在最优解中, 一定可以将序列划分成若干个长度大于 1 的段 $[l_1, r_1], \dots, [l_k, r_k]$, 满足:

1. $\forall i, j \in [l_i, r_i], a_j * a_{j+1} = b_j$
2. $\forall i, a_{r_i} * a_{l_{i+1}} > b_{r_i}$

即段内相邻两数乘积等于限制, 不同段间相邻两数乘积大于限制。

对于一段 $[l_i, r_i]$, 显然可以将这一段内的所有数都表示为 $v_1 * a_{l_i}$ 或 $\frac{v_2}{a_{l_i}}$ 的形式。因此这一段数的总和可以被表示为 $x * a_{l_i} + \frac{y}{a_{l_i}}$ 的形式。

显然这是一个关于 a_{l_i} 的单峰函数, 且极值在 $a_{l_i} = \sqrt{\frac{y}{x}mn}$ 取到。

考虑最优的解 a_i 满足的性质。

如果 $a_{i-1} * a_i > b_{i-1}$, $a_i * a_{i+1} > b_i$, 则减小 a_i 一定可以得到 $\sum a_i$ 最小的答案。

因此在最优解中, 一定可以将序列划分成若干个长度大于 1 的段 $[l_1, r_1], \dots, [l_k, r_k]$, 满足:

1. $\forall i, j \in [l_i, r_i], a_j * a_{j+1} = b_j$
2. $\forall i, a_{r_i} * a_{l_{i+1}} > b_{r_i}$

即段内相邻两数乘积等于限制, 不同段间相邻两数乘积大于限制。

对于一段 $[l_i, r_i]$, 显然可以将这一段内的所有数都表示为 $v_1 * a_{l_i}$ 或 $\frac{v_2}{a_{l_i}}$ 的形式。因此这一段数的总和可以被表示为 $x * a_{l_i} + \frac{y}{a_{l_i}}$ 的形式。

显然这是一个关于 a_{l_i} 的单峰函数, 且极值在 $a_{l_i} = \sqrt{\frac{y}{x}mn}$ 取到。

如果在最优解中, 有一段内部没有取到这一段的最小值, 则将这一段的 a_{l_i} 向这一段和更小的方向调整, 可以得到更优的方案。因此最优解一定满足每一段内部都取到了最小值。

对于每一个可能出现的段 $[l, r]$, 求出这一段取到最小值时, a_l 的值 $lv_{l,j}$, a_r 的值 $rv_{i,j}$ 以及这一段的总和 $su_{i,j}$ 。

对于每一个可能出现的段 $[l, r]$, 求出这一段取到最小值时, a_l 的值 $lv_{l,j}$, a_r 的值 $rv_{i,j}$ 以及这一段的总和 $su_{i,j}$ 。
枚举 i 再从小到大枚举 j 就可以做到 $O(n^2)$ 预处理。

对于每一个可能出现的段 $[l, r]$, 求出这一段取到最小值时, a_l 的值 $lv_{l,j}$, a_r 的值 $rv_{i,j}$ 以及这一段的总和 $su_{i,j}$ 。

枚举 i 再从小到大枚举 j 就可以做到 $O(n^2)$ 预处理。

判断一种划分段的方式是否合法只需要判断相邻两段之间的限制是否满足, 可以考虑 dp: 设 $dp_{l,r}$ 表示将 $[1, r]$ 划分成若干段, 相邻两段间满足 b_i 的限制, 且最后一段为 $[l, r]$ 的方案中, $\sum a_i$ 的最小值。

对于每一个可能出现的段 $[l, r]$, 求出这一段取到最小值时, a_l 的值 $lv_{l,j}$, a_r 的值 $rv_{i,j}$ 以及这一段的总和 $su_{i,j}$ 。

枚举 i 再从小到大枚举 j 就可以做到 $O(n^2)$ 预处理。

判断一种划分段的方式是否合法只需要判断相邻两段之间的限制是否满足, 可以考虑 dp: 设 $dp_{l,r}$ 表示将 $[1, r]$ 划分成若干段, 相邻两段间满足 b_i 的限制, 且最后一段为 $[l, r]$ 的方案中, $\sum a_i$ 的最小值。

枚举下一段, 可以得到转移:

对于每一个可能出现的段 $[l, r]$, 求出这一段取到最小值时, a_l 的值 $lv_{l,j}$, a_r 的值 $rv_{i,j}$ 以及这一段的总和 $su_{i,j}$ 。

枚举 i 再从小到大枚举 j 就可以做到 $O(n^2)$ 预处理。

判断一种划分段的方式是否合法只需要判断相邻两段之间的限制是否满足, 可以考虑 dp: 设 $dp_{l,r}$ 表示将 $[1, r]$ 划分成若干段, 相邻两段间满足 b_i 的限制, 且最后一段为 $[l, r]$ 的方案中, $\sum a_i$ 的最小值。

枚举下一段, 可以得到转移:

$$dp_{l,r} = su_{l,r} + \min_{a < l-1, rv_{a,l-1} * lv_{l,r} > b_{l-1}} dp_{a,l-1}$$

直接做的复杂度为 $O(n^3)$, 对于每个 l , 将所有 $rv_{a,l-1}$ 和 $lv_{l,r}$ 分别排序之后即可做到 $O(n \log n)$ 转移。复杂度 $O(n^2 \log n)$ 。

二维平面上有 n 个点, 点 i 的坐标为 (x_i, y_i) 。

有 p 个向量 $(u_1, v_1), (u_2, v_2), \dots, (u_n, v_n)$, 你可以选一个 n 阶排列 p , 之后点 i 的坐标会变为 $(x_i + u_{p_i}, y_i + v_{p_i})$ 。

你选择的排列 p 需要满足对于每一个点对, 它们在坐标变化后的距离不小于坐标变化之前的距离。在此基础上, 你需要最大化变化后每一对点间距离的平方的和。

输出方案。

$n \leq 500$

1s, 512MB

考虑需要最大化的东西：

考虑需要最大化的东西：

$$\begin{aligned} & \sum_{i,j} (x_i + u_{p_i} - x_j - u_{p_j})^2 + (y_i + v_{p_i} - y_j - v_{p_j})^2 \\ = & \sum_{i,j} (x_i - x_j)^2 + (u_{p_i} - u_{p_j})^2 - (x_i u_{p_j} + x_j u_{p_i}) + x_i u_{p_i} + x_j u_{p_j} \\ & + (y_i + v_{p_i} - y_j - v_{p_j})^2 \end{aligned}$$

考虑需要最大化的东西：

$$\begin{aligned} & \sum_{i,j} (x_i + u_{p_i} - x_j - u_{p_j})^2 + (y_i + v_{p_i} - y_j - v_{p_j})^2 \\ = & \sum_{i,j} (x_i - x_j)^2 + (u_{p_i} - u_{p_j})^2 - (x_i u_{p_j} + x_j u_{p_i}) + x_i u_{p_i} + x_j u_{p_j} \\ & + (y_i + v_{p_i} - y_j - v_{p_j})^2 \end{aligned}$$

可以发现 $\sum_{i,j} (x_i - x_j)^2 + (u_{p_i} - u_{p_j})^2 - (x_i u_{p_j} + x_j u_{p_i})$ 是定值，因此相当于在满足条件的要求下最大化：

考虑需要最大化的东西：

$$\sum_{i,j} (x_i + u_{p_i} - x_j - u_{p_j})^2 + (y_i + v_{p_i} - y_j - v_{p_j})^2$$

$$= \sum_{i,j} (x_i - x_j)^2 + (u_{p_i} - u_{p_j})^2 - (x_i u_{p_j} + x_j u_{p_i}) + x_i u_{p_i} + x_j u_{p_j}$$

可以发现 $\sum_{i,j} (x_i - x_j)^2 + (u_{p_i} - u_{p_j})^2 - (x_i u_{p_j} + x_j u_{p_i})$ 是定值，因此相当于在满足条件的要求下最大化：

$$\sum_i x_i u_{p_i} + y_i v_{p_i}$$

考虑选择使 $\sum_i x_i u_{p_i} + y_i v_{p_i}$ 最大的排列 p , 如果在这个排列中存在两个点 i, j 间的距离变小, 则:

考虑选择使 $\sum_i x_i u_{p_i} + y_i v_{p_i}$ 最大的排列 p , 如果在这个排列中存在两个点 i, j 间的距离变小, 则:

$$(x_i - x_j)^2 + (y_i - y_j)^2 > (x_i + u_{p_i} - x_j - u_{p_j})^2 + (y_i + v_{p_i} - y_j - v_{p_j})^2$$

$$\begin{aligned} (x_i u_{p_i} + y_i v_{p_i}) + (x_j u_{p_j} + y_j v_{p_j}) - (x_i u_{p_j} + y_i v_{p_j}) - (x_j u_{p_i} + y_j v_{p_i}) \\ < -(u_{p_i} - u_{p_j})^2 - (v_{p_i} - v_{p_j})^2 \leq 0 \end{aligned}$$

考虑选择使 $\sum_i x_i u_{p_i} + y_i v_{p_i}$ 最大的排列 p , 如果在这个排列中存在两个点 i, j 间的距离变小, 则:

$$(x_i - x_j)^2 + (y_i - y_j)^2 > (x_i + u_{p_i} - x_j - u_{p_j})^2 + (y_i + v_{p_i} - y_j - v_{p_j})^2$$

$$\begin{aligned} (x_i u_{p_i} + y_i v_{p_i}) + (x_j u_{p_j} + y_j v_{p_j}) - (x_i u_{p_j} + y_i v_{p_j}) - (x_j u_{p_i} + y_j v_{p_i}) \\ < -(u_{p_i} - u_{p_j})^2 - (v_{p_i} - v_{p_j})^2 \leq 0 \end{aligned}$$

因此交换 p_i, p_j 后 $\sum_i x_i u_{p_i} + y_i v_{p_i}$ 更大, 这说明满足 $\sum_i x_i u_{p_i} + y_i v_{p_i}$ 最大的排列 p 一定合法。

考虑选择使 $\sum_i x_i u_{p_i} + y_i v_{p_i}$ 最大的排列 p , 如果在这个排列中存在两个点 i, j 间的距离变小, 则:

$$(x_i - x_j)^2 + (y_i - y_j)^2 > (x_i + u_{p_i} - x_j - u_{p_j})^2 + (y_i + v_{p_i} - y_j - v_{p_j})^2$$

$$\begin{aligned} (x_i u_{p_i} + y_i v_{p_i}) + (x_j u_{p_j} + y_j v_{p_j}) - (x_i u_{p_j} + y_i v_{p_j}) - (x_j u_{p_i} + y_j v_{p_i}) \\ < -(u_{p_i} - u_{p_j})^2 - (v_{p_i} - v_{p_j})^2 \leq 0 \end{aligned}$$

因此交换 p_i, p_j 后 $\sum_i x_i u_{p_i} + y_i v_{p_i}$ 更大, 这说明满足 $\sum_i x_i u_{p_i} + y_i v_{p_i}$ 最大的排列 p 一定合法。

复杂度 $O(n^3)$

交互题。

有一个 n 个点的竞赛图，你每次可以向交互库询问一条边的方向。找到任意一个出度大于等于 $n - 2$ 的点或者报告不存在这样的点。询问次数不超过 $4n$ ($5n$)。

$n \leq 1000$

3s, 512MB

考虑先对每个点找到一条连向它的边。考虑如下过程：

考虑先对每个点找到一条连向它的边。考虑如下过程：
选择一个标记点 x ，初始 $x = 1$ 。按照编号依次遍历每个点，询问当前点和 x 间的边的方向。

考虑先对每个点找到一条连向它的边。考虑如下过程：
选择一个标记点 x ，初始 $x = 1$ 。按照编号依次遍历每个点，询问当前点和 x 间的边的方向。
如果为 x 连向当前点则不改变 x ，否则将 x 变为当前点。

考虑先对每个点找到一条连向它的边。考虑如下过程：
选择一个标记点 x ，初始 $x = 1$ 。按照编号依次遍历每个点，询问当前点和 x 间的边的方向。
如果为 x 连向当前点则不改变 x ，否则将 x 变为当前点。
随后再按照编号遍历点，直到找到一个点有连向 x 的边或者发现 x 连向所有点。

考虑先对每个点找到一条连向它的边。考虑如下过程：
选择一个标记点 x ，初始 $x = 1$ 。按照编号依次遍历每个点，询问当前点和 x 间的边的方向。

如果为 x 连向当前点则不改变 x ，否则将 x 变为当前点。
随后再按照编号遍历点，直到找到一个点有连向 x 的边或者发现 x 连向所有点。

显然，在这个过程后，如果发现 x 连向所有点，则可以直接回答这个点，否则，这个做法可以对于每个点至少找到一个连向它的边。

设对于点 i , 连向它的点编号为 f_i 。

设对于点 i , 连向它的点编号为 f_i 。

对于点 i, j , 只需要 $f_i \neq j, f_j \neq i$, 则询问 (i, j) 的方向, 至少能排除 i, j 中的一个点。

设对于点 i , 连向它的点编号为 f_i 。

对于点 i, j , 只需要 $f_i \neq j, f_j \neq i$, 则询问 (i, j) 的方向, 至少能排除 i, j 中的一个点。

因此每次选两个这样的点, 直到不能选为止。最后会留下若干个点。

设对于点 i , 连向它的点编号为 f_i 。

对于点 i, j , 只需要 $f_i \neq j, f_j \neq i$, 则询问 (i, j) 的方向, 至少能排除 i, j 中的一个点。

因此每次选两个这样的点, 直到不能选为止。最后会留下若干个点。

可以发现, 如果有 4 个点, 则这里面一定能选出一对满足上面条件的, 因此最多会留下 3 个点。

设对于点 i , 连向它的点编号为 f_i 。

对于点 i, j , 只需要 $f_i \neq j, f_j \neq i$, 则询问 (i, j) 的方向, 至少能排除 i, j 中的一个点。

因此每次选两个这样的点, 直到不能选为止。最后会留下若干个点。

可以发现, 如果有 4 个点, 则这里面一定能选出一对满足上面条件的, 因此最多会留下 3 个点。

考虑对于剩余的点暴力枚举判断。如果上一部分询问了 $n + k$ 次, 则一定对于 k 个点找到了两条连向它的边。因此可以排除 k 个点。

设对于点 i , 连向它的点编号为 f_i 。

对于点 i, j , 只需要 $f_i \neq j, f_j \neq i$, 则询问 (i, j) 的方向, 至少能排除 i, j 中的一个点。

因此每次选两个这样的点, 直到不能选为止。最后会留下若干个点。

可以发现, 如果有 4 个点, 则这里面一定能选出一对满足上面条件的, 因此最多会留下 3 个点。

考虑对于剩余的点暴力枚举判断。如果上一部分询问了 $n + k$ 次, 则一定对于 k 个点找到了两条连向它的边。因此可以排除 k 个点。

对于第二部分, 需要的操作次数为删掉的点数加 $3n$, 因此总操作次数不超过 $5n$ 。

如果剩下 2 个点可以直接做，只需要再考虑剩 3 个点的情况。

如果剩下 2 个点可以直接做，只需要再考虑剩 3 个点的情况。
可以发现，此时这三个点的 f_i 会构成一个三元环。

如果剩下 2 个点可以直接做，只需要再考虑剩 3 个点的情况。
可以发现，此时这三个点的 f_i 会构成一个三元环。
可以发现，对于一个三元环外的点，一定存在一个方案使得可以询问这个点和三元环上的一个点。

如果剩下 2 个点可以直接做，只需要再考虑剩 3 个点的情况。
可以发现，此时这三个点的 f_i 会构成一个三元环。
可以发现，对于一个三元环外的点，一定存在一个方案使得可以询问这个点和三元环上的一个点。
考虑尽量每次询问包含一个环上的点，如果最后三元环还存在，则这部分所有的询问都以环上某个点为一个端点。因此后面的询问可以使用这部分询问的结果。

如果剩下 2 个点可以直接做，只需要再考虑剩 3 个点的情况。

可以发现，此时这三个点的 f_i 会构成一个三元环。

可以发现，对于一个三元环外的点，一定存在一个方案使得可以询问这个点和三元环上的一个点。

考虑尽量每次询问包含一个环上的点，如果最后三元环还存在，则这部分所有的询问都以环上某个点为一个端点。因此后面的询问可以使用这部分询问的结果。

同时，设上一部分询问了 $n + k$ 次，则最后 k 次询问的端点也以环上某个点为一个端点，因此可以发现最多有 $4n$ 个不同的询问。

有 n 个人，初始每个人有 1 个宝石。

每天晚上，等概率在当前所有的宝石中选出一个，这个宝石的拥有者获得 1 个宝石。

求出 m 个晚上后，拥有宝石数量前 k 多的人拥有的宝石数量和的期望，模 998244353。

$$n, m, k \leq 1.5 \times 10^7$$

1s, 512MB

假设最后第 i 个人有 a_i 个宝石, 则一定有 $a_i \geq 1, \sum a_i = n + m$ 。考虑一种这样的情况出现的方案数。

假设最后第 i 个人有 a_i 个宝石, 则一定有 $a_i \geq 1, \sum a_i = n + m$ 。考虑一种这样的情况出现的方案数。

考虑枚举每一天是谁 $+1$, 则这样的方案数为 $m! * \prod \frac{1}{(a_i-1)!}$

假设最后第 i 个人有 a_i 个宝石, 则一定有 $a_i \geq 1, \sum a_i = n + m$ 。考虑一种这样的情况出现的方案数。

考虑枚举每一天是谁 $+1$, 则这样的方案数为 $m! * \prod \frac{1}{(a_i-1)!}$

然后考虑枚举每次 $+1$ 是哪个宝石的贡献, 如果一个人最后有 a 个, 则方案数为 $1 * 2 * \dots * a - 1 = (a - 1)!$, 因此这部分的方案数为 $\prod (a_i - 1)!$ 。

假设最后第 i 个人有 a_i 个宝石, 则一定有 $a_i \geq 1, \sum a_i = n + m$ 。考虑一种这样的情况出现的方案数。

考虑枚举每一天是谁 $+1$, 则这样的方案数为 $m! * \prod \frac{1}{(a_i-1)!}$

然后考虑枚举每次 $+1$ 是哪个宝石的贡献, 如果一个人最后有 a 个, 则方案数为 $1 * 2 * \dots * a - 1 = (a - 1)!$, 因此这部分的方案数为 $\prod (a_i - 1)!$ 。因此可以发现, 对于所有满足条件的 a , 它们出现的概率相同。因此可以计算每种方案的前 k 大和。

考虑枚举 a , 计算有 $0, 1, \dots, n$ 个人的宝石数量大于 a 的方案数 $f_{a,0}, f_{a,1}, \dots, f_{a,n}$ 。

考虑枚举 a , 计算有 $0, 1, \dots, n$ 个人的宝石数量大于 a 的方案数

$f_{a,0}, f_{a,1}, \dots, f_{a,n}$ 。

考虑容斥计算, 先求出 $g_{a,i} = \sum C_j^i f_{a_j}$, 这相当于选 i 个点, 让他们的宝石数量大于等于 a 的方案数, 这相当于 $C_n^i * C_{m-(a-1)i+n}^n$ 。

考虑枚举 a , 计算有 $0, 1, \dots, n$ 个人的宝石数量大于 a 的方案数

$f_{a,0}, f_{a,1}, \dots, f_{a,n}$ 。

考虑容斥计算, 先求出 $g_{a,i} = \sum C_j^i f_{a,j}$, 这相当于选 i 个点, 让他们的宝石数量大于等于 a 的方案数, 这相当于 $C_n^i * C_{m-(a-1)i+n}^n$ 。

根据二项式反演可以得到 $f_{a,i} = \sum C_j^i (-1)^{j-i} g_{a,j}$, 这部分的贡献即为 $\sum_i f_{a,i} * \min(i, k)$ 。

考虑枚举 a , 计算有 $0, 1, \dots, n$ 个人的宝石数量大于 a 的方案数

$f_{a,0}, f_{a,1}, \dots, f_{a,n}$ 。

考虑容斥计算, 先求出 $g_{a,i} = \sum C_j^i f_{a,j}$, 这相当于选 i 个点, 让他们的宝石数量大于等于 a 的方案数, 这相当于 $C_n^i * C_{m-(a-1)i+n}^n$ 。

根据二项式反演可以得到 $f_{a,i} = \sum C_j^i (-1)^{j-i} g_{a,j}$, 这部分的贡献即为 $\sum_i f_{a,i} * \min(i, k)$ 。

直接做复杂度为 $O(n^2)$ 。考虑计算一个 $g_{a,i}$ 的贡献, 可以发现它等于 $\sum_{i \leq j} C_j^i (-1)^{j-i} * \min(i, k)$

设 $v_{n,k} = \sum_{i \leq n} C_n^i (-1)^{n-i} * \min(i, k)$, 考虑用组合数递推计算:

设 $v_{n,k} = \sum_{i \leq n} C_n^i (-1)^{n-i} * \min(i, k)$, 考虑用组合数递推计算:

$$v_{n,k} = \sum_{i \leq n} (C_{n-1}^i + C_{n-1}^{i-1}) (-1)^{n-i} * \min(i, k)$$

$$v_{n,k} = \sum_{i \leq n} C_{n-1}^i (-1)^{n-i} * \min(i, k) + \sum_{i \leq n} C_{n-1}^{i-1} (-1)^{n-i} * \min(i, k)$$

$$v_{n,k} = - \sum_{i \leq n} C_{n-1}^i (-1)^{(n-1)-i} * \min(i, k) +$$

$$\sum_{i \leq n} C_{n-1}^{i-1} (-1)^{(n-1)-(i-1)} * (\min(i-1, k-1) + 1)$$

$$v_{n,k} = -v_{n-1,k} + v_{n-1,k-1} + \sum_{i \leq n} C_{n-1}^{i-1} (-1)^{(n-1)-(i-1)}$$

$$v_{n,k} = -v_{n-1,k} + v_{n-1,k-1} + [n=1]$$

$$v_{n,k} = -v_{n-1,k} + v_{n-1,k-1} + [n = 1]$$

$$v_{n,k} = -v_{n-1,k} + v_{n-1,k-1} + [n=1]$$

初值有 $v_{1,k} = 1 (k > 0)$, $dp_{n,0} = 0$, 则考虑组合意义则有
 $v_{n,k} = \sum_{i=0}^{k-1} (-1)^{n-1-i} C_{n-1}^i$.

$$v_{n,k} = -v_{n-1,k} + v_{n-1,k-1} + [n = 1]$$

初值有 $v_{1,k} = 1 (k > 0)$, $dp_{n,0} = 0$, 则考虑组合意义则有

$$v_{n,k} = \sum_{i=0}^{k-1} (-1)^{n-1-i} C_{n-1}^i.$$

再做一次递推可以得到：

$$v_{n,k} = -v_{n-1,k} + v_{n-1,k-1} + [n=1]$$

初值有 $v_{1,k} = 1 (k > 0)$, $dp_{n,0} = 0$, 则考虑组合意义则有

$$v_{n,k} = \sum_{i=0}^{k-1} (-1)^{n-1-i} C_{n-1}^i.$$

再做一次递推可以得到：

$$v_{n,k} = \sum_{i=0}^{k-1} (-1)^{n-1-i} (C_{n-2}^i + C_{n-2}^{i-1}) (n > 2)$$

$$v_{n,k} = - \sum_{i=0}^{k-1} (-1)^{n-2-i} C_{n-2}^i + \sum_{i=0}^{k-2} (-1)^{n-2-i} C_{n-2}^i$$

$$v_{n,k} = (-1)^{n-k} C_{n-2}^{k-1}$$

$$v_{n,k} = -v_{n-1,k} + v_{n-1,k-1} + [n=1]$$

初值有 $v_{1,k} = 1 (k > 0)$, $dp_{n,0} = 0$, 则考虑组合意义则有

$$v_{n,k} = \sum_{i=0}^{k-1} (-1)^{n-1-i} C_{n-1}^i.$$

再做一次递推可以得到：

$$v_{n,k} = \sum_{i=0}^{k-1} (-1)^{n-1-i} (C_{n-2}^i + C_{n-2}^{i-1}) (n > 2)$$

$$v_{n,k} = - \sum_{i=0}^{k-1} (-1)^{n-2-i} C_{n-2}^i + \sum_{i=0}^{k-2} (-1)^{n-2-i} C_{n-2}^i$$

$$v_{n,k} = (-1)^{n-k} C_{n-2}^{k-1}$$

因此, $v_{1,k} = 1$, $v_{n,k} = (-1)^{n-k} C_{n-2}^{k-1} (n > 1)$

现在考虑答案的式子。

现在考虑答案的式子。

$$ans = n + \sum_{a>0} \sum_{j=1}^n C_n^j * C_{m-a*j+n}^n * v_{j,k}$$

$$ans = n + \sum_{a>0} n * C_{m-a+n}^n + \sum_{a>0} \sum_{j=2}^n C_n^j * C_{m-a*j+n}^n * (-1)^{j-k} C_{j-2}^{k-1}$$

现在考虑答案的式子。

$$ans = n + \sum_{a>0} \sum_{j=1}^n C_n^j * C_{m-a*j+n}^n * v_{j,k}$$

$$ans = n + \sum_{a>0} n * C_{m-a+n}^n + \sum_{a>0} \sum_{j=2}^n C_n^j * C_{m-a*j+n}^n * (-1)^{j-k} C_{j-2}^{k-1}$$

只用考虑最后一部分，先枚举 $a * j$ ，则变为：

现在考虑答案的式子。

$$ans = n + \sum_{a>0} \sum_{j=1}^n C_n^j * C_{m-a*j+n}^n * v_{j,k}$$

$$ans = n + \sum_{a>0} n * C_{m-a+n}^n + \sum_{a>0} \sum_{j=2}^n C_n^j * C_{m-a*j+n}^n * (-1)^{j-k} C_{j-2}^{k-1}$$

只用考虑最后一部分，先枚举 $a*j$ ，则变为：

$$\sum_t C_{m-t+n}^n \sum_{j|t} C_n^j * (-1)^{j-k} * C_{j-2}^{k-1}$$

最后的问题可以看成, 给出 v_1, \dots, v_m , 求出 $t_i = \sum_{j|i} v_j$ 。

最后的问题可以看成, 给出 v_1, \dots, v_m , 求出 $t_i = \sum_{j|i} v_j$ 。

设 $1, 2, \dots, m$ 内所有质数为 p_1, p_2, \dots, p_k , 依次考虑每一个 p_i , 对于一个 p_i , 从小到大枚举 x , 将 $v_{x \cdot p_i}$ 加上 v_x 。

最后的问题可以看成, 给出 v_1, \dots, v_m , 求出 $t_i = \sum_{j|i} v_j$ 。

设 $1, 2, \dots, m$ 内所有质数为 p_1, p_2, \dots, p_k , 依次考虑每一个 p_i , 对于一个 p_i , 从小到大枚举 x , 将 $v_{x \cdot p_i}$ 加上 v_x 。

可以发现这个做法类似于高维前缀和, 容易证明它的正确性。可以发现复杂度为 $O(n \log \log n)$

备选题 1

给一棵 n 个点的树，每个点有点权 v_i ，每条边有权值 a_i, b_i 。
现在依次考虑每条边，对于第 i 条边，它有 $\frac{a_i}{a_i+b_i}$ 的概率被删掉，有 $\frac{b_i}{a_i+b_i}$ 的概率留下。
这样操作后，定义分数为每个连通块中的最大点权的最小值。求出分数的期望模 $10^9 + 7$ 。

$$n \leq 10^5$$

2s, 512MB

备选题 2

给一棵 n 个点的有根树, 1 为根, 求有多少个 n 阶排列 p 满足:

$$\forall i \in \{2, 3, \dots, n-1\}, \text{dep}(\text{lca}(p_{i-1}, p_i)) \leq \text{dep}(\text{lca}(p_i, p_{i+1}))$$

答案模 $10^9 + 7$ 。

$$n \leq 80$$

2s, 512MB

备选题 3

给两个长度为 n 的正整数序列 a, b 以及 m , 满足 $b_i \leq a_i \leq m$ 。
你可以进行不超过 l 次操作, 每次操作你可以给定正整数 l, r, k , 满足 $l \leq r, 1 \leq k \leq m$, 表示将 a_l, a_{l+1}, \dots, a_r 都对 k 取 \min 。
如果一次操作后序列 a 等于序列 b , 则你应该立即停止。
求在不超过 l 次操作内, 让序列 a 等于序列 b 的方案数, 模 998244353。
 $n, m, l \leq 100$
3s, 512MB