Day1 讲题 + 胡测选讲

Scape

June 9, 2018

得分情况

100 分:12 人
20 分:ckw 等若干
0 分: 若干

Solution

大概是最简单的一题了。
大概就要算一个欧拉回路状物。
其实就是每条边可以选/不选，每个点的要求被选的相邻边是奇数/偶数。最小化权值
显然任意一个生成树都可以调整出来，然后选最小生成树肯定是最优的。就没了。

得分情况

100 分:4 人
80 分:1 人
<=20 分: 若干

Solution

$(i,j) -> (i-j,j)$

这样就把它扭成了一个矩形。每一次可以往左边或者上边走。

注意到这其实是一个杨氏矩阵。然后杨氏矩阵的方案数是 $\frac{n!}{\prod h_i}$。

预处理阶乘之后都是可以直接算的。

得分情况

100 分:tyc
<=40 分: 若干

Solution

$O(nq)$?
每次 dp。

Solution

先想一下两个人的策略？

Solution

令 S 是一个集合，满足 $\forall x \in S, \forall i > x, p_x < p_i$。
对于 $i \notin S, i$ 属于 $i \bmod 2$。
对于 $i \in S$,i 属于 $last \bmod 2$,$last$ 是 S 集合里最大的 $< i$ 的数。
相信大家都是数据结构大师维护这东西还是非常轻松的。

Solution

大概感性思考一下这玩意为啥是对的。
注意到如果到了第 $n-1$ 次 $a[n-1]$ 之前没被翻转，那 $a[n]$ 就会被翻转，$n \bmod 2$ 那一方就会丢掉 $a[n]$。
于是你可以令 $f[i]$ 代表如果到了第 i 轮,$a[i]$ 之前没被翻转过，那么另外一方就会丢掉 $f[i]$。
然后你注意到 $f[i]$ 就是 $\min a[k]|k = i \cdots n$
因为如果你在一个最小值上你无论如何都会优先翻后面，宁愿丢掉这个最小值。
其余的位置都是优先保障的，即如果第 i 轮 $a[i]$ 之前被翻过了那么策略一定是把 $a[i]$ 翻回来 $(i \notin S)$。

Description

有排成一列的 n 个球，编号依次为 0 到 $n-1$，初始都为白色。重复以下操作 2 次：随机选择其中的 m 个球将它们染成绿色（可以重复染绿球）。如果令 A 为编号最小的绿球的编号，求 $F(A)$ 的期望对 998244353 取模的值。其中，$F(x)$ 为一个次数不超过 m 的多项式，$F(A)$ 表示 $x = A$ 时多项式的值。

$n \leq 998244353$，$m \leq 1000000$，$F(x)$ 以 $x = 0 \ldots m$ 的点值形式给出。

暴力 1

首先插值将多项式转化成系数表示的形式，并求出该多项式在 $0 \cdots n-1$ 的点值。
记 $f_{i,j,k}$ 为从后往前染色染到编号为 i 的球，且第一次染色中染了 j 个球，第二次 k 个球的方案数。有四种转移，分别为，当前球不染色，第一次染色中染黑，第二次中染黑，两次都染. 则 $f_{i,j,k} = f_{i+1,j,k} + [j > 0] \times f_{i+1,j-1,k} + [k > 0] \times f_{i+1,j,k-1} + [j > 0] \times [k > 0] \times f_{i+1,j-1,k-1}$;
$ans = \sum_{i=0}^{n-1} F(i) \times (f_{i,m,m} - f_{i+1,m,m})$。
复杂度为 $O(nm^2)$。

暴力 2

实际上 $f_{i,m,m}$ 可以直接计算而不用递推得到。
考虑大于等于 i 的数的个数为 $n-i$，因此每种颜色的染色方案实际都为 $\binom{n-i}{m}$。
即 $f_{i,m,m}=\binom{n-i}{m}^2$。
使用同样的方式计算贡献，总复杂度为 $O(nm)$。

稍微优化一下暴力

如果编号最小位置 A 的贡献是关于 A 的多项式，将其求前缀和，得到的也会是一个多项式。
那答案就是将 $n-1$ 代入得到的答案。
考虑暴力，$\binom{n-A}{m}^2$，是关于 A 的 $2m$ 次多项式。
而它要乘上的贡献，$F(x)$ 是一个 m 次多项式。
因此最终每个位置的贡献就是一个 $3m$ 次多项式，其前缀和是一个 $3m+1$ 次多项式。
我们取 $0\cdots 3m+1$ 算得其点值，使用拉格朗日插值就能线性得到该多项式在 $n-1$ 处的点值，也就是答案。
复杂度 $O(m\log m)$，常数较大。

标算？

还是讲一下已经在天堂的出题人标算吧。

$$F(x) = x$$

寻找期望的等价形式。即编号小于 A 的球的数量的期望。
再根据期望的线性性，我们要算的就是每个球编号小于 A 的概率的和。
我们枚举真实被染色的球的数量来统计答案。
如果当前枚举的是 k 个，由于两次各染 m 个，那么 $k \in [m, 2m]$。
不妨直接先选出 $k+1$ 个球，方案数为 $\binom{n}{k+1}$。

$$F(x) = x$$

由于我们多选择的球编号比 A 小，因此真实染色的就是最后的 k 个，是一一对应的。
但是我们枚举的是染色的位置，真实的染色情况还需要考虑。
第一次随意染，方案数为 $\binom{k}{m}$。
由于选择的位置都要有颜色，因此第一次没染的一定要染黑，而多余的只能染到已经被染黑的球上，方案数为 $\binom{m}{2m-k}$。
那么总的方案数 $H_k = \binom{k}{m} \times \binom{m}{2m-k}$。
H 数组在预处理阶乘和阶乘逆元后可以线性求出。
$ans = \sum_{k=m}^{2m} \binom{n}{k+1} \times H_k$。
总复杂度 $O(m)$。

$$F(x) = x^c$$

等价形式为，可重复的选择 c 个球，编号全都小于 A 的期望。
根据期望的线性性，要算的是每种可重复选择 c 个球的方案，c 个球编号都小于 A 的概率的和。
注意到被选择的球可重复，我们不妨求去重后选择的球数量为 $q(q = 0 \cdots c)$ 时的方案数 G_q。

$$F(x) = x^c$$

考虑二项式反演。随意染的总次数为 $q^c = \sum_{i=0}^{q} \binom{q}{i} G_i$，那么 $G_q = \sum_{i=0}^{q} \binom{q}{i} (-1)^{q-i} i^c$。当然，实际上 G_q 就是 $S(c, q)q!$，其中 S 表示第二类斯特林数。
G 数组可以在 $O(m^2)$ 时间求出。
令 T_i 表示染色过的球与被选择的球总数为 i 的方案数，由于选择的球的编号一定小于染色的球编号，因此方案一一对应。容易发现 $T_i = \sum_{j=0}^{i} G_j H_{i-j}$。$T$ 数组可以在 $O(m^2)$ 时间求出。
最终 $ans = \sum_{i=0}^{3m} \binom{n}{i} T_i$。
总复杂度 $O(m^2)$。

$$F(x) = \sum_{i=0}^{m} a_i x^i$$

注意到当 $F(x) = x^c$ 时，$G_q = \sum_{i=0}^{q} \binom{q}{i} (-1)^{q-i} i^c$
那么求和之后，现在的
$G_q = \sum_{i=0}^{q} \binom{q}{i} (-1)^{q-i} \sum_{j=0}^{m} a_j i^j = \sum_{i=0}^{q} \binom{q}{i} (-1)^{q-i} F(i)$
其余部分和之前相同。
总复杂度 $O(m^2)$。

对 G 数组的计算推导后可以得到，$G(q)/q! = \sum_{i=0}^{q} \frac{(-1)^{q-i}}{(q-i)!} \times \frac{F(i)}{i!}$。
可以用 NTT 求出 G 数组。同样的道理用 NTT 求出 T 数组。
总复杂度为 $O(m \log m)$，常数较小。

Description

通过交互的方式，要求用二叉树维护若干个序列。
初始时有 n 个序列，每个序列仅有一个元素。接下来会发生 m 个事件，事件有如下三种类型：

- (join) 把两个序列拼接成一个序列。
- (split) 把某个序列拆成两个序列。
- (visit) 询问某个序列的信息和。

交互库会对每棵树提供两个指针，分别指向树上的某个结点，需要通过对指针进行操作来维护这些事件。允许修改某个指针指向结点的儿子、更新某个指针指向结点的子树信息和以及把某个指针移动到相邻结点。交互库把这三种操作封装在了 move 函数中，并限制调用该函数的总次数不能超过 $2 * 10^6$ 次，在每次事件中调用该函数的次数不能超过 $maxcnt$。另外还限制查询的二叉树的最大深度不能超过 $maxdep$。

数据范围

- **子任务 1（15 分）**：
  $1 \leq n, m \leq 1000$，$maxdep = 2 * 10^5$，$maxcnt = 2 * 10^6$。
- **子任务 2（25 分）**：
  $1 \leq n, m \leq 30000$，$maxdep = 2 * 10^5$，$maxcnt = 2 * 10^6$。
- **子任务 3（15 分）**：
  $1 \leq n, m \leq 10^5$，$k \leq 10000$，$maxdep = 2 * 10^5$，
  $maxcnt = 2 * 10^6$。
- **子任务 4（15 分）**：
  $1 \leq n, m \leq 2 * 10^5$，$k \leq 20000$，**没有 split 事件**，
  $maxdep = 60$，$maxcnt = 250$。
- **子任务 5（30 分）**：
  $1 \leq n, m \leq 2 * 10^5$，$k \leq 20000$，$maxdep = 60$，
  $maxcnt = 250$。

其中 k 表示 split 和 visit 事件的总个数。

暴力

维护序列是一个非常经典的问题。我们可以使用任意支持合并与分裂的平衡树来实现这个二叉树。复杂度通常都是一个 log 的，可以通过子任务 1、2。实现得比较好的话子任务 3 也可以通过。

Solution

子任务 4、5 中限制了树的深度以及每次事件中的调用次数，这意味着我们不能再使用 splay 这种均摊复杂度的平衡树。我们可以改为使用 treap 来维护。然而，如果直接使用普通的合并方式的话仅能通过子任务 1、2。
我们来仔细地观察一下合并 treap 的过程。合并两棵 treap T1、T2 的时候，其实就是自顶向下地把 T1 的右链以及 T2 的左链按照优先级归并了起来。这样做的复杂度是 $O(\log \max(|T1|, |T2|))$。然而在极端情况下，例如，T1 是一棵很大的 treap，T2 则仅有一个结点，那么 T2 的那个结点很大概率会被合并到接近底层的位置，然而我们却需要遍历 T1 的几乎整条右链，这就导致了很多冗余的代价。

Solution

我们不妨把每棵树的两个指针维护在第一个结点和最后一个结点上，然后自底向上地进行归并，当归并完其中一棵树的根结点时就结束这个过程。假设我们先归并到的是 T1 的根结点，那么这个过程就相当于在新树中遍历了 T1 的最后一个结点或者 T2 的第一个结点到 T1 的根结点的路径。而 treap 中排名相差 d 的两个结点之间的路径长度是期望 $O(\log d)$ 的，因此这样合并的复杂度就是 $O(\log \min(|T1|, |T2|))$。
然而为了保持这个复杂度，我们合并后就不能再往上更新结点的子树信息和了。解决方法也很简单，显然每棵树未更新的结点一定都在左链或右链顶部的一段上，那么我们可以直接延迟到下次 split 或者 visit 事件时再更新。这样就可以通过全部测试点了。

Solution

我们来分析一下整个做法的复杂度。定义一个长度为 l 的序列的势能函数为 $\log l$，那么合并两个长度分别为 $l1, l2$ 的序列时势能的增量为

$$\begin{aligned}&\log(l1+l2)-\log l1-\log l2\\=&\log(l1+l2)-\log\max(l1,l2)-\log\min(l1,l2)\\\leq&1-\log\min(l1,l2)\end{aligned}$$

初始时总势能为 0，且最终总势能非负，因此只有 join 事件时复杂度是 $O(n)$ 的。而一次 split 事件只会额外增加 $O(\log n)$ 的势能。那么，这个算法的时间复杂度就是 $O(n+m+k\log n)$。

Description

这是一道交互题。
小修和小栋喜欢玩一个叫做猜♂数字的游戏。
小修会先在心中想好一个包含 n 个互不相同的正整数的序列 $a_1, a_2, \cdots, a_n$。
小栋每次会向小修询问 a_x, a_y, a_z 的中位数，小修会准确地回答。
然后小栋则需要利用这些信息来尽可能地还原出序列 a。
然而小修实在是太厉♂害了，选取的 n 都会特别♂大，导致小栋根本不知如何处理。
请你帮助小栋，让他能和小修愉快地玩耍。

具体实现

你需要实现一个函数 guess，以帮助小栋完成游戏。
你可以调用函数 ask 来向小修进行询问。我们会根据你调用这个函数的总次数评分。
$ask(x, y, z)$，其中 x, y, z 为三个不同的下标，这个函数会返回 a_x, a_y, a_z 的中位数。
同时，你还可以调用函数 answer 来回答你还原出的信息。你不能对于同一个 x 进行两次回答，但可以对于某些 x 不进行回答，表示你不能准确地还原出这个下标上的值。

暴力

首先看到题目里说有有些东西是不能确定的，于是先思考哪些数是不能确定的。
不难发现最小的两个数和最大的两个数显然是没法确定的。
当我们知道这个信息后，我们发现我们只要知道哪两个是最小的那两个是最大的我们就可以确定剩下的每一个数的值了。

暴力

所以我们考虑怎么样找到最小的和最大的 2 个数。因为是对称的，我们只需要考虑其中一个。
对于一个四元组 (x, y, z, k)，我们对于其中任意三元组都询问一遍，这样可以得到两个较小的值和两个较大的值，然后我们把两个回答较小的值的三元组取并集，所得到的就是较小的两个值。
这样我们用增量法构造就能得到两个最小的值，然后同理得到两个最大的值，之后再对于每一个数都询问一下就能知道每个数是多少了。
这样要 $5n$ 次询问。但实际上如果 random_shuffle 之后再卡卡常实际次数要小很多，亲测大概有 75 分。

能过的做法

考虑改进，我们次数太多的原因是我们在询问四元组时，信息没有完全利用好。
我们现在改成维护四元组 (x, y, z, k)，且 $\max(x, y) < \min(z, k)$。
表示当前遇到的最小的两个数和最大的两个数。并且我们还能顺便知道 (x, y) 中较大数的值和 (z, k) 中较小数的值。
每次加进来一个 i，先询问 (x, i, k)。
如果中位数在 (y, z) 之间，那说明这个中位数就是 i 的值，直接回答即可。

能过的做法

否则看它是比较小还是比较大。
如果是比较大的话就询问 (x, i, k)，然后分类讨论下就可以知道 x y 哪个更大来决定踢哪个。在踢的时候还能顺便回答一次询问。
如果是大于也是对称的。
考虑复杂度，如果我们事先 random_shuffle 过了，那么一个序列的 LIS 期望是 $O(\log)$ 的。所以要询问两次的期望次数是 $O(\log)$ 次，所以总询问次数是 $n + O(\log)$。

Description

小 A 有一张 n 个点的图，点的标号为 0 到 $n-1$。点 i 到点 j 有 $A_{i,j}$ 条有向边。可能有自环。
现在小 A 要在图上进行若干次旅行。每次旅行都是选任意一个起点，走至少一步，走到任意一个终点。定义一次旅行的愉悦值为起点与终点编号的按位与的值。
好奇的小 B 想要知道：对于所有 $x \in [1, m]$ 和 $y \in [0, n)$，小 A 进行了若干次旅行，总共走了 x 步，且所有旅行的愉悦值的按位与为 y 的方案数。
两种方案不同当且仅当旅行次数不同或某一次旅行不完全相同。
为了防止输出过多，你只需要输出这 $n \times m$ 个数对 998244353 取模后的结果的按位异或值。
为方便起见，保证 n 是 2 的幂次。
$n \leq 64, m \leq 20000$。

Solution

首先 and 可以容斥掉。
然后我们对于一次的旅行，其实就是求 $A^1 + A^2 + \cdots + A^k$ 状物。
考虑怎么算这个，直接把特征多项式爆插出来然后直接算就可以了。
然后如果我们如果求出了 f_k 表示一次旅行走 k 步，那么 $g_k = \sum g_i f_{k-i}$，多项式求逆直接算一波就可以了。

Description

小 A 有一棵 N 个点的带边权的树，树的每个节点有重量 w_i 和价值 v_i。
现在小 A 要从中选出若干个节点形成一个集合 S，满足这些节点重量之和 $\leq M$ 并且构成一个连通块。小 A 是一个完美主义者，因此他只会选择节点价值之和最大的那些 S。我们称这样的集合 S 为完美的集合。
现在小 A 要从所有完美的集合中选出 K 个，并对这 K 个完美的集合分别进行测试。在这 K 次测试开始前，小 A 首先需要选择一个点 x 来放置他的测试装置，这个测试装置的最大功率为 Max。

Description

接下来的每次测试，小 A 会对测试对象 S 中的所有点进行一次能量传输，对一个点 y 进行能量传输需要的功率为 $dist(x,y)\times v_y$，其中 $dist(x,y)$ 表示点 x,y 在树上的最短路长度。因此，如果 S 中存在一个点 y，满足 $dist(x,y)\times v_y > Max$，测试就会失败。同时，为了保证能量传输的稳定性，测试装置所在的点 x 需要在集合 S 中，否则测试也会失败。
现在小 A 想知道，有多少种从所有完美的集合中选出 K 个的方法，使得他能找到一个放置测试装置的点，来完成他的测试呢？
你只需要输出方案数对 $1192092895507812 5$ 取模的结果
$n \le 60, m \le 10^4$。

Solution

注意到合法的关键点是联通的，计算一个点 x 的答案时可以容斥，用 x 合法的减去 x 和 fa_x 都合法的。
枚举之后将合法点取出来，变成了一棵有根树，可以利用 DFS 序 DP 求出最大权值以及方案数。
剩余部分是组合数对质数的幂次取模的问题。即求 $\prod_{1\leq i\leq n, i \mod 5\neq 0} i$。维护一个多项式 $f_n(x)=\prod_{1\leq i\leq n, i \mod 5\neq 0}(x+i)$，然后找到一个最大的 $10k$ 满足 $10k\leq n$，求出 f_{10k} 后其余的直接乘。由于 $f_{10k}(x)=f_{5k}(x)f_{5k}(x+5k)$，前一项递归求，后一项展开到 23 项后贡献为 0，暴力维护即可。

Description

给定 n 个字符串 s_i 和 m 个 $1 \sim n$ 之间的整数 a_i，令母串为 $s_{a_1} + s_{a_2} + \cdots + s_{a_m}$，回答 q 次询问，每次给出一个字符串 t_i，询问这个串在母串中的出现次数。
保证 s_i 和 t_i 都只由字母 a,b 组成。
$1 \leq n, m, q \leq 5 \times 10^4$
$\sum_{1 \leq i \leq n} |s_i|, \sum_{1 \leq i \leq q} |t_i| \leq 10^5$。

Solution

询问串在母串中的一次匹配只有两种情况：要么是询问串在单个字典串内出现，要么询问串横跨了若干个个字典串，我们对两部分分别处理。

Part 1

我们每次给出一个询问串 t，要计算 $\sum_{i=1}^{n} c(t, s_{a_i})$。这是一个比较常见的问题。我们对字典建 trie，再对这个 trie 建出它的 SAM。
类似解法一，我们对于 SAM 上的每个节点，预处理出它所代表的字符串在 S 中的出现次数。
每次询问时找到询问串 t 在 parent 树上所对应的节点，就能直接得到答案了。
这一部分的复杂度是 $O(\sum |s_i| + \sum |t_i|)$。

Part 2

我们设置一个 B，对于询问串长度小于 B 的我们都用 Part2 的算法结局。
形式化地说，计算跨越多个字典串的出现次数也就是计算 $\sum_{i=1}^{n-1} c(t, S[\max(lp_i, rp_i - |t| + 2) : rp_i + |t| - 1])$ 。
我们可以把所有 $S[\max(lp_i, rp_i - L + 2) : rp_i + L - 1]$ 拿出来建 Trie ，并对该 Trie 建出 SAM。对于前缀 $S[\max(lp_i, rp_i - L + 2) : rp_i + k - 1]$ 我们将其标号为 k 。

Part 2

统计 $\sum_{i=1}^{n-1} c(t, S[\max(lp_i, rp_i - |t| + 2) : rp_i + |t| - 1])$ 也就是相当于在统计 parent 树上 t 所代表的节点的子树中有多少节点的标号在 $[2, |t| - 1]$ 中，对 parent 树求 dfs 序后可以转化为二维数点问题。
我们注意到这个二维数点的第二维是 $O(L)$ 级别的。并且总点数是 $O(mL)$ 级别的，询问是 $O(q)$ 级别的。我们可以轻松地通过离线做到加入 $O(1)$ 询问 $O(L)$，因此这一部分的总复杂度是 $O(mL)$ 的。

综上所述，我们可以在 $O(\sum |s_i| + \sum |t_i| + mL)$ 的复杂度内解决这个部分。

Part 3

我们考虑对于询问串 $> B$ 的询问。
我们先介绍两个引理：

引理

如果 p, q 都是字符串 S 的周期，且 $p + q \leq |S|$，那么 $\gcd(p, q)$ 也是字符串 S 的周期。

Proof.

令 $d = q - p(p < q)$，则由 $i - p > 0$ 或 $i + q \leq |S|$ 均可推出 $S_i = S_{i+d}$。

□

Part 3

引理

令 l 是周期长度，对于 $i > 2l, next_i = i - l$。

Proof.

这等价于对于 $i > 2l$ 的前缀，它的最短周期是 l。我们使用反证法：如果存在比 l 更短的周期 T，那么 $T + l < i$, 所以由引理 3.1 可得 $\gcd(T, l)$ 也是这个串的周期，注意到 $\gcd(T, l) \mid l$，这与 l 是整个串的最短周期矛盾，故得证。

□

Part 3

考虑优化 KMP 算法。我们考虑把询问串放在母串上跑 KMP。
我们注意到我们只需要计算串之间的匹配就可以了。
我们令 i 表示当前 KMP 算法指向字典串的指针，j 表示当前 KMP 算法指向询问串的指针,L 表示询问串的长度。
如果存在某个 k，使得 $i \in [lp_k, rp_k]$ 且 $i-j+1 \in [lp_k, rp_k]$，那么说明当前正在匹配的是串内的，是不需要被计算的。所以我们直接令 j 等于最长的询问串前缀的长度，并且这个询问串的前缀是 s_{a_k} 的一个后缀，再把 i 设为那个后缀开始的地方再继续 KMP 即可。

Part 3

由于我们可以快速求出两个后缀的 LCP ，因此我们可以很容易地计算下一次失配的位置。因此我们只需要优化跳 next 的过程即可。
在失配之后如果 $next_j < \frac{j}{2}$ 我们就直接执行 $j := next_j$，否则：
如果执行 $j := next_j$ 后还会失配，根据引理，我们可以直接跳到 $l + j \bmod l$ 的位置（l 是周期长度）(Case (1))。

Part 3

否则我们统计接下来 $S[i+1:]$ 还能继续匹配多少个 $t[:j]$ 的周期并找到失配时的位置，此时有两种可能的情况，第一种情况是 $S[i+1] = t_j$(Case (2))，此时我们继续向下匹配，第二种情况同之前 Case (1)。
有一个特殊情况是我们匹配到了串末。这时候我们可以计算 $S[i+1:]$ 还能匹配多少个 $t[:j]$ 的周期，能匹配的周期数就是对答案的贡献。之后我们和失配情况一样处理即可。

Part 3

如何证明之前所述的算法的时间复杂度是 $O(m \log n)$ 的？
对于 Case (1)，考虑 $j-(i-lp_k)$ 的值。显然当 $j-(i-lp_k) \leq 0$ 时，我们就会重置 i 和 j 的值。
首先，求 LCP 时，$j-(i-lp_k)$ 的值并不会变。
而在跳 next 时，对于 $next_j \leq \frac{j}{2}$ 时，执行 $j := next_j$ 会让 j 变小 $\frac{1}{2}$。对于 $next_j > \frac{j}{2}$ 时，当我们执行 $j := l + j \bmod l$ 时，$l + j \bmod l \leqslant \lceil \frac{2l}{3} \rceil$，$j$ 会变小 $\frac{1}{3}$。所以每次 Case（1）操作就会让 $j-(i-lp_k) \leq 0$ 变小至少 $\frac{1}{3}$。
所以我们发现 Case (1) 的总复杂度是 $O(m \log n)$ 的。

Part 3

对于 Case (2)，我们先引入一个关于层的定义：
我们令 $b_i = \left[next_i \geqslant \frac{i}{2} \right]$，将 b 序列中从左到右第 i 段连续的 1 称作第 i 层。
我们再证明一个有关层的性质：

Part 3

定理

令第 i 层的前缀的周期为 per_i，那么有 $\frac{3}{2}|per_i| \leq |per_{i+1}|$

Proof.

令 pos_i 表示第 i 层最右边的位置，那么显然 per_i 和 per_{i+1} 都是 $S[:pos_i]$ 的周期。
如果 $|per_i| + |per_{i+1}| \leq pos_i$，那么根据引理 3.2, 可以知道 $\gcd(|per_i|, |per_{i+1}|)$ 也是 $S[:pos_i]$ 的周期。注意到 $\gcd(|per_i|, |per_{i+1}|)$ 整除 $|per_{i+1}|$，这与 per_{i+1} 是第 $i+1$ 层的最短周期矛盾，故我们得到 $|per_i| + |per_{i+1}| > pos_i$
又有 $pos_i \geq 2|per_i|$，联立两式可得 $\frac{3}{2}|per_i| \leq |per_{i+1}|$。

□

Part 3

那么对于 Case (2)，而根据定理 3.3 显然一共只有 $\log$ 层。注意到只有两种操作，而 Case 1 的操作每次只会倒退一层,Case 2 的操作每次只会增加一层。所以 Case (2) 的操作最多只有 $m \log n$ 次。
因此我们的总失配次数是 $O(m \log n)$ 的，而对于求 LCP 部分，我们建出字典串的 SAM on Trie 后直接查询对应他们在 SAM 上节点的 LCA 就做到查询他们的 LCP。注意到 LCA 问题可以转化成欧拉序上的 RMQ 问题，是可以做到 $O(1)$ 回答的。而对于重置 i, j 部分，我们也可以快速地在 $O(\log n)$ 的时间内求两个串的最长公共前后缀，所以我们总的时间复杂度也是 $O(m \log n)$ 的。

Solution

综上所述，我们取 $B = \sqrt{nlogn}$ 时复杂度最低，为 $O(n\sqrt{nlogn})$。

Description

Description

- 给你一个 $n \times m$ 的棋盘

Description

- 给你一个 $n \times m$ 的棋盘
- 上面有障碍, 你可以在上面不是障碍的某些地方放上棋子

Description

- 给你一个 $n \times m$ 的棋盘
- 上面有障碍，你可以在上面不是障碍的某些地方放上棋子
- 每次移动是全部棋盘上全部棋子一起移动，如果移动方向是障碍则不移动，一个棋子如果到了边缘会掉落

Description

- 给你一个 $n \times m$ 的棋盘
- 上面有障碍，你可以在上面不是障碍的某些地方放上棋子
- 每次移动是全部棋盘上全部棋子一起移动，如果移动方向是障碍则不移动，一个棋子如果到了边缘会掉落
- 一个摆放方式是好的当且仅当存在一种移动方式使得一些棋子掉落而另一些不掉落，求方案数

Description

- 给你一个 $n \times m$ 的棋盘
- 上面有障碍，你可以在上面不是障碍的某些地方放上棋子
- 每次移动是全部棋盘上全部棋子一起移动，如果移动方向是障碍则不移动，一个棋子如果到了边缘会掉落
- 一个摆放方式是好的当且仅当存在一种移动方式使得一些棋子掉落而另一些不掉落，求方案数
- $n, m \leq 50$

Solution

Solution

- 显然一个有两个棋子的好的方案往上面任意添加棋子还是好的

Solution

- 显然一个有两个棋子的好的方案往上面任意添加棋子还是好的
- 先 BFS 算出每个地方和其它多少棋子放一起是好的

Solution

- 显然一个有两个棋子的好的方案往上面任意添加棋子还是好的
- 先 BFS 算出每个地方和其它多少棋子放一起是好的
- 注意到如果 $(a, b), (a, c)$ 不是好的, 那么 (b, c) 也不是好的

Solution

- 显然一个有两个棋子的好的方案往上面任意添加棋子还是好的
- 先 BFS 算出每个地方和其它多少棋子放一起是好的
- 注意到如果 $(a,b),(a,c)$ 不是好的，那么 (b,c) 也不是好的
- $\sum_{t=2}^{n} \frac{\sum_{i=1}^{n} \binom{n-1}{t-1} - \binom{n-valid[i]-1}{t-1}}{t}$

Solution

- 显然一个有两个棋子的好的方案往上面任意添加棋子还是好的
- 先 BFS 算出每个地方和其它多少棋子放一起是好的
- 注意到如果 $(a,b),(a,c)$ 不是好的，那么 (b,c) 也不是好的
- $\sum_{t=2}^{n} \frac{\sum_{i=1}^{n} \binom{n-1}{t-1} - \binom{n-valid[i]-1}{t-1}}{t}$
- 时间复杂度 $O(n^2m^2)$

Description

Description

- 给你一个字符串集合 X

Description

- 给你一个字符串集合 X
- 等几率从字符串集合中挑出 K 个字符串 (可以重复) 并拼成一个大字符串

Description

- 给你一个字符串集合 X
- 等几率从字符串集合中挑出 K 个字符串 (可以重复) 并拼成一个大字符串
- 问 S 在大字符串中的期望出现次数

Description

- 给你一个字符串集合 X
- 等几率从字符串集合中挑出 K 个字符串 (可以重复) 并拼成一个大字符串
- 问 S 在大字符串中的期望出现次数
- $|S| \leq 500, |X| \leq 50, |X_i| \leq 50, K \leq 10^{12}$

Solution

Solution

- 首先考虑暴力 dp,$f(i,j)$ 表示挑了 i 个串，当前 kmp 指针在 j 的期望次数

Solution

- 首先考虑暴力 dp,$f(i,j)$ 表示挑了 i 个串，当前 kmp 指针在 j 的期望次数
- 那么显然枚举下一个是哪个可以做到时间复杂度 $O(K|X||S|)$

Solution

- 首先考虑暴力 dp,$f(i,j)$ 表示挑了 i 个串，当前 kmp 指针在 j 的期望次数
- 那么显然枚举下一个是哪个可以做到时间复杂度 $O(K|X||S|)$
- 考虑 f_i 向 f_{i+1} 转移的矩阵是固定的可以矩乘

Solution

- 首先考虑暴力 dp,$f(i,j)$ 表示挑了 i 个串，当前 kmp 指针在 j 的期望次数
- 那么显然枚举下一个是哪个可以做到时间复杂度 $O(K|X||S|)$
- 考虑 f_i 向 f_{i+1} 转移的矩阵是固定的可以矩乘
- 然而我们注意到这样一个事实

Solution

Solution

- 假设 Ans_i 表示 $K = i$ 时的答案

Solution

- 假设 Ans_i 表示 $K = i$ 时的答案
- 那么由对称性，考虑每个答案的增量，可得当 $i \geq |S|$ 时

Solution

- 假设 Ans_i 表示 $K = i$ 时的答案
- 那么由对称性，考虑每个答案的增量，可得当 $i \geq |S|$ 时
- $Ans_{i+2} - Ans_{i+1} = Ans_{i+1} - Ans_i$

Solution

- 假设 Ans_i 表示 $K = i$ 时的答案
- 那么由对称性，考虑每个答案的增量，可得当 $i \geq |S|$ 时
- $Ans_{i+2} - Ans_{i+1} = Ans_{i+1} - Ans_i$
- $i \geq |S|$ 的原因是可以保证每一个挑出的串长度都 $\geq |S|$

Solution

- 假设 Ans_i 表示 $K = i$ 时的答案
- 那么由对称性, 考虑每个答案的增量, 可得当 $i \geq |S|$ 时
- $Ans_{i+2} - Ans_{i+1} = Ans_{i+1} - Ans_i$
- $i \geq |S|$ 的原因是可以保证每一个挑出的串长度都 $\geq |S|$
- 直接暴力 DP 就好, 时间复杂度 $O(|S|^3)$

Description

Description

- 给你一个 $n \times m$ 的网格图, 每个点上有数

Description

- 给你一个 $n \times m$ 的网格图, 每个点上有数
- 两个矩形的 LCM 为所有在两个矩形中至少出现过一次的数的 LCM

Description

- 给你一个 $n \times m$ 的网格图, 每个点上有数
- 两个矩形的 LCM 为所有在两个矩形中至少出现过一次的数的 LCM
- 两个矩形的亲密度为最小的数 X, 使得 X 不整除 LCM

Description

- 给你一个 $n \times m$ 的网格图, 每个点上有数
- 两个矩形的 LCM 为所有在两个矩形中至少出现过一次的数的 LCM
- 两个矩形的亲密度为最小的数 X, 使得 X 不整除 LCM
- 求所有矩形对的亲密度和

Description

- 给你一个 $n \times m$ 的网格图, 每个点上有数
- 两个矩形的 LCM 为所有在两个矩形中至少出现过一次的数的 LCM
- 两个矩形的亲密度为最小的数 X, 使得 X 不整除 LCM
- 求所有矩形对的亲密度和
- $n, m, v \leq 50$

Description

- 给你一个 $n \times m$ 的网格图, 每个点上有数
- 两个矩形的 LCM 为所有在两个矩形中至少出现过一次的数的 LCM
- 两个矩形的亲密度为最小的数 X, 使得 X 不整除 LCM
- 求所有矩形对的亲密度和
- $n, m, v \leq 50$
- 难度:Hard

Solution

Solution

- 显然能成为答案的只有 p^t

Solution

- 显然能成为答案的只有 p^t
- 枚举答案, 计算答案 $\geq cur$ 的个数

Solution

- 显然能成为答案的只有 p^t
- 枚举答案, 计算答案 $\geq cur$ 的个数
- 考虑怎么样的能成为答案, 那么显然每个 p_i 的次数都 $\geq log_{p_i} cur$

Solution

- 显然能成为答案的只有 p^t
- 枚举答案, 计算答案 $\geq cur$ 的个数
- 考虑怎么样的能成为答案, 那么显然每个 p_i 的次数都 $\geq log_{p_i} cur$
- 因为是 LCM, 所以每个矩阵压成若干个二进制位, 每个二进制位表示当前指数是否满足要求

Solution

- 显然能成为答案的只有 p^t
- 枚举答案, 计算答案 $\geq cur$ 的个数
- 考虑怎么样的能成为答案, 那么显然每个 p_i 的次数都 $\geq log_{p_i} cur$
- 因为是 LCM, 所以每个矩阵压成若干个二进制位, 每个二进制位表示当前指数是否满足要求
- FWT(or) 合并答案

Solution

- 显然能成为答案的只有 p^t
- 枚举答案, 计算答案 $\geq cur$ 的个数
- 考虑怎么样的能成为答案, 那么显然每个 p_i 的次数都 $\geq log_{p_i} cur$
- 因为是 LCM, 所以每个矩阵压成若干个二进制位, 每个二进制位表示当前指数是否满足要求
- FWT(or) 合并答案
- 时间复杂度 $O(25(n^2m^2 + 15\ 2^15))$

题面描述

题面描述

- 你有 n 个球和 m 个筐

题面描述

- 你有 n 个球和 m 个筐
- 每个球可以放到特定的两个筐内

题面描述

- 你有 n 个球和 m 个筐
- 每个球可以放到特定的两个筐内
- 选择一种合适的放球方案使得有奇数个球的筐最少

题面描述

- 你有 n 个球和 m 个筐
- 每个球可以放到特定的两个筐内
- 选择一种合适的放球方案使得有奇数个球的筐最少
- $n, m \leq 10^5$

题解

题解

- 首先不考虑"每个球可以放到特定的两个筐内"

题解

- 首先不考虑"每个球可以放到特定的两个筐内"
- 如果两个球可以放到同一个筐内，那么他们就可以抵消

题解

- 首先不考虑"每个球可以放到特定的两个筐内"
- 如果两个球可以放到同一个筐内，那么他们就可以抵消
- 一般图最大匹配

题解

- 首先不考虑"每个球可以放到特定的两个筐内"
- 如果两个球可以放到同一个筐内，那么他们就可以抵消
- 一般图最大匹配
- (现场有小哥带花树跑过去了不得不服

题解

题解

- 考虑”每个球可以放到特定的两个筐内”

题解

- 考虑" 每个球可以放到特定的两个筐内"
- 把球看成边, 把筐看成点

题解

- 考虑”每个球可以放到特定的两个筐内”
- 把球看成边，把筐看成点
- 拥有同一端点的边可以抵消

题解

- 考虑" 每个球可以放到特定的两个筐内"
- 把球看成边, 把筐看成点
- 拥有同一端点的边可以抵消
- 最后最少剩几条边?

题解

题解

- dfs 整个图

题解

- dfs 整个图
- 考虑树边。树边最开始暂时分配给儿子。

题解

- dfs 整个图
- 考虑树边。树边最开始暂时分配给儿子。
- 如果儿子最后还剩一条边，那就把树边还给父亲

题解

- dfs 整个图
- 考虑树边。树边最开始暂时分配给儿子。
- 如果儿子最后还剩一条边，那就把树边还给父亲
- 遇到返祖边就给祖先

题解

- dfs 整个图
- 考虑树边。树边最开始暂时分配给儿子。
- 如果儿子最后还剩一条边，那就把树边还给父亲
- 遇到返祖边就给祖先
- 证明用交换法显然

题解

- dfs 整个图
- 考虑树边。树边最开始暂时分配给儿子。
- 如果儿子最后还剩一条边，那就把树边还给父亲
- 遇到返祖边就给祖先
- 证明用交换法显然
- 时间复杂度 $O(n+m)$

Description

Description

- 有 N 个人在玩猜拳游戏。

Description

- 有 N 个人在玩猜拳游戏。
- 每个人都有一个长度为 K 的出拳序列，序列里的每个元素是 R,P 或 S，表示他们出的是石头，剪刀还是布。

Description

- 有 N 个人在玩猜拳游戏。
- 每个人都有一个长度为 K 的出拳序列，序列里的每个元素是 R,P 或 S，表示他们出的是石头，剪刀还是布。
- 每个人的出拳都是循环的，即他们会不断地重复他们的出拳序列。

Description

- 有 N 个人在玩猜拳游戏。
- 每个人都有一个长度为 K 的出拳序列，序列里的每个元素是 R,P 或 S，表示他们出的是石头，剪刀还是布。
- 每个人的出拳都是循环的，即他们会不断地重复他们的出拳序列。
- 两个人之间是好的当且仅当他们两个人从他们出拳序列任意地方开始出拳，循环地进行了 K 场比赛之后，三元组 (赢的场数, 输的场数, 平的场数) 都一样

Description

- 有 N 个人在玩猜拳游戏。
- 每个人都有一个长度为 K 的出拳序列，序列里的每个元素是 R,P 或 S，表示他们出的是石头，剪刀还是布。
- 每个人的出拳都是循环的，即他们会不断地重复他们的出拳序列。
- 两个人之间是好的当且仅当他们两个人从他们出拳序列任意地方开始出拳，循环地进行了 K 场比赛之后，三元组 (赢的场数, 输的场数, 平的场数) 都一样
- 问有多少个不同的这 N 个人的子集，使得他们两两之间都是好的。答案对 $10^9 + 7$ 取模。

Description

- 有 N 个人在玩猜拳游戏。
- 每个人都有一个长度为 K 的出拳序列，序列里的每个元素是 R,P 或 S，表示他们出的是石头，剪刀还是布。
- 每个人的出拳都是循环的，即他们会不断地重复他们的出拳序列。
- 两个人之间是好的当且仅当他们两个人从他们出拳序列任意地方开始出拳，循环地进行了 K 场比赛之后，三元组 (赢的场数, 输的场数, 平的场数) 都一样
- 问有多少个不同的这 N 个人的子集，使得他们两两之间都是好的。答案对 $10^9 + 7$ 取模。
- $N \leq 10^5, K \leq 15$

Solution

Solution

- 我们考虑怎样的两个出拳序列是好的。

Solution

- 我们考虑怎样的两个出拳序列是好的。
- 我们令 w 为 1 的三次单位根, 令 w^2 表示石头,w^1 表示剪刀,w^0 表示布, 并且把每个出拳序列写成多项式的形式。

Solution

- 我们考虑怎样的两个出拳序列是好的。
- 我们令 w 为 1 的三次单位根，令 w^2 表示石头，w^1 表示剪刀，w^0 表示布，并且把每个出拳序列写成多项式的形式。
- 那两个出拳序列是好的，当且仅当把其中一个出拳序列的多项式系数全部取倒数之后，他们的循环卷积出的多项式每一位都相同。

Solution

- 我们考虑怎样的两个出拳序列是好的。
- 我们令 w 为 1 的三次单位根，令 w^2 表示石头，w^1 表示剪刀，w^0 表示布，并且把每个出拳序列写成多项式的形式。
- 那两个出拳序列是好的，当且仅当把其中一个出拳序列的多项式系数全部取倒数之后，他们的循环卷积出的多项式每一位都相同。
- 我们考虑一下一个每一位系数都一样的多项式的性质，显然我们对这个多项式 FFT 之后，除了 1 那一项的点值，其他每一位的点值都是 0。

Solution

- 我们考虑怎样的两个出拳序列是好的。
- 我们令 w 为 1 的三次单位根，令 w^2 表示石头，w^1 表示剪刀，w^0 表示布，并且把每个出拳序列写成多项式的形式。
- 那两个出拳序列是好的，当且仅当把其中一个出拳序列的多项式系数全部取倒数之后，他们的循环卷积出的多项式每一位都相同。
- 我们考虑一下一个每一位系数都一样的多项式的性质，显然我们对这个多项式 FFT 之后，除了 1 那一项的点值，其他每一位的点值都是 0。
- 所以我们对每一个出拳序列暴力插值之后得出了 $K+1$ 个点值，我们把 1 的点值扔掉后对于剩下的 K 个点值压缩成一个 $0/1$ 状态表示它这一位是否是 0，我们就可以依此暴力子集 dp，可以得到 3^K 的做法。。

Description

Description

- 给你一张 M 个点的图。图中任意一条从 S 到 T 的长度为 K 的倍数的路径会对答案产生 $\binom{n}{Len}$ 的贡献

Description

- 给你一张 M 个点的图。图中任意一条从 S 到 T 的长度为 K 的倍数的路径会对答案产生 $\binom{n}{Len}$ 的贡献
- 路径可以经过重复的边。答案对 P 取模

Description

- 给你一张 M 个点的图。图中任意一条从 S 到 T 的长度为 K 的倍数的路径会对答案产生 $\binom{n}{Len}$ 的贡献
- 路径可以经过重复的边。答案对 P 取模
- $M \leq 10, K \leq 1000, P \equiv 1 (\mathrm{mod}\, K), N \leq 10^18$

Solution

Solution

- 先考虑怎么样让一条长度为 Len 的路径贡献 $\binom{n}{Len}$。

Solution

- 先考虑怎么样让一条长度为 Len 的路径贡献 $\binom{n}{Len}$。
- 让总时间为 n, 每个时间可以选择走一步或者不走，这样长度为 Len 的路径就会产生 $\binom{n}{Len}$ 的贡献

Solution

- 先考虑怎么样让一条长度为 Len 的路径贡献 $\binom{n}{Len}$。
- 让总时间为 n, 每个时间可以选择走一步或者不走，这样长度为 Len 的路径就会产生 $\binom{n}{Len}$ 的贡献
- 那么接下来考虑要求长度 $\bmod K = 0$。

Solution

- 先考虑怎么样让一条长度为 Len 的路径贡献 $\binom{n}{Len}$。
- 让总时间为 n, 每个时间可以选择走一步或者不走，这样长度为 Len 的路径就会产生 $\binom{n}{Len}$ 的贡献
- 那么接下来考虑要求长度 $\bmod K = 0$。
- 考虑在矩阵上维护一个最大系数为 K 的多项式表示 $\bmod K = i$ 的所有路径的和。

Solution

Solution

- 如果转移的时候直接暴力乘循环卷积时间复杂度 $O(M^3K^2)$ 接受不了。

Solution

- 如果转移的时候直接暴力乘循环卷积时间复杂度 $O(M^3K^2)$ 接受不了。
- 注意到 $P \equiv 1 (\bmod K)$, 所以肯定存在 $r^k \equiv 0 (\bmod P)$

Solution

- 如果转移的时候直接暴力乘循环卷积时间复杂度 $O(M^3K^2)$ 接受不了。
- 注意到 $P \equiv 1 (\bmod K)$, 所以肯定存在 $r^k \equiv 0 (\bmod P)$
- 把 $r^0 \cdots r^{k-1}$ 插值的话就能实现一个循环卷积。

Solution

- 如果转移的时候直接暴力乘循环卷积时间复杂度 $O(M^3K^2)$ 接受不了。
- 注意到 $P \equiv 1 (\bmod K)$, 所以肯定存在 $r^k \equiv 0 (\bmod P)$
- 把 $r^0 \cdots r^{k-1}$ 插值的话就能实现一个循环卷积。
- 所以就插值之后矩阵乘法再暴力插回来就可以算答案了。

Solution

- 如果转移的时候直接暴力乘循环卷积时间复杂度 $O(M^3K^2)$ 接受不了。
- 注意到 $P \equiv 1 (\bmod K)$, 所以肯定存在 $r^k \equiv 0 (\bmod P)$
- 把 $r^0 \cdots r^{k-1}$ 插值的话就能实现一个循环卷积。
- 所以就插值之后矩阵乘法再暴力插回来就可以算答案了。
- 时间复杂度 $O(M^3K + K^2)$