《抽奖机》解题报告

1 题目描述

1.1 题目大意

有n个转轮构成的转轮组S，每个转轮的档位在0,1,2之间循环。

给定了m种操作，每种操作a_i, b_i表示从S中随机选出a_i个转一次，b_i个转两次，每轮会选取一种操作的一种转法进行。

求k轮后，最终抽奖机恰好有i个转轮在1档，j个转轮在2档的方案数$mod\ 10^9+9$。

1.2 数据范围

对于所有数据，满足$n \le 120, m \le 10^5, k \le 10^{18}, \forall\ 0 \le a_i, b_i \le n, \forall\ a_i + b_i \le n$。给定的$m$个模式之间**可能有重复。**

特殊性质：有25%的测试点满足$b_i = 0$。

2 解题过程

2.1 算法 0

用三进制存储状态，纯暴力枚举每轮的变化，复杂度$O\left(((3^n)^2 \sim (3^n)^3) \cdot k\right)$

2.2 算法 1

推论 在每轮操作过后，所有恰好有i个转轮在1档，j个转轮在2档的状态方案数相同

设第c轮后，一个恰好有i个转轮在1档，j个转轮在2档的状态的方案数为$f_c(i,j)$，可以用递推的方式（归纳法）证明，同时求得解。

证明与递推

设满足$a_i = x, b_i = y$的操作有$g(i,j)$个，

当$c = 0$时，只有均在0档的为一个状态，$f_0(0,0) = 1$。

当$c = k \ge 1$时，考虑一个恰好有i个转轮在1档，j个转轮在2档的状态，其所对应的父状态。

设在这个状态所有档位为x的转轮中，在上一轮被转了y次的有$a_{x,y}$个，则满足$a_{0,0} + a_{0,1} + a_{0,2} = n - i - j$，$a_{1,0} + a_{1,1} + a_{1,2} = i$，$a_{2,0} + a_{2,1} + a_{2,2} = j$，并且得到：

$$f_c(i,j)=\sum_a \binom{n-i-j}{a_{0,0},a_{0,1}}\binom{i}{a_{1,0},a_{1,1}}\binom{j}{a_{2,0},a_{2,1}} f_{c-1}(a_{0,2}+a_{1,0}+a_{2,1},a_{0,1}+a_{1,2}$$

$$+a_{2,0})g(a_{0,1}+a_{1,1}+a_{2,1},a_{0,2}+a_{1,2}+a_{2,2})$$

容易发现对于任意一个恰好有i个转轮在1档，j个转轮在2档的状态，其前驱状态的方案数均相同，因此推论成立。而枚举a中的6个元素即可确定一组合法的a，暴力实现递推的复杂度为$O(n^8\cdot k)$，实际上常数极小。

2.3 算法 2

优化算法 1 的递推，先枚举i,j之后，令$h_{a,b,x,y}$为前驱状态有a个1，b个2，有x个位置转了1次，有y个位置转了两次的方案数。依次将n个位置中的每一个加入，枚举这个位置转了几次，同时递推h的值。每轮递推复杂度为$O(n^5)$，这样同时还去掉了组合数，总复杂度为$O(n^7k)$。

2.4 算法 3

用矩阵优化上述算法，记录一轮操作中从$(i,j)\to(a,b)$的方案数，一轮操作的转移同上面的递推。矩阵的规模为$O(n^2)$，因此总复杂度为$O(n^7+n^6 log\ k)$。

2.5 算法 4

针对$b_i=0$的优化，此时只需要枚举3个数即可确定一组合法的a，因此递推总复杂度为$O(n^5)$。

2.6 $3-$FWT

下面的优化均基于$3-$FWT，下面简单介绍$3-$FWT：

设ω为$mod\ P=10^9+9$意义下的3次单位根，$\boldsymbol{A}$是一个n维向量，每一维的值$\in\{0,1,2\}$，记这个向量第i维$0\le i<n$的值为A_i。这样的向量对应原问题的一个状态，定义这样的向量之间的运算。

1. 加减法$\boldsymbol{A}\pm\boldsymbol{B}=\left((A_0\pm B_0)mod\ 3,(A_1\pm B_1)mod\ 3,\cdots\right)$

2. 内积$\boldsymbol{A}\cdot\boldsymbol{B}=\sum A_i\cdot B_i$

设$F(x)$为一个多项式，其每一项的指数为类似A的向量，记作$F(x)=f_1x^{\boldsymbol{A1}}+f_2x^{\boldsymbol{A2}}\cdots$

记$[x^{\boldsymbol{A}}]F(x)$为$F(x)$中指数为$\boldsymbol{A}$的一项的系数。同理，记$G(x)=g_1x^{\boldsymbol{B1}}+g_2x^{\boldsymbol{B2}}\cdots$。可以根据向量的加法，定义这样多项式之间的乘法。

2.6.1 正变换

$3-$FWT用于优化这样多项式的乘法，首先需要对于$F(x)$做一步转化：

设$F'(x)$为$F(x)$经过$3-\text{FWT}$变换的结果，则$[x^{\boldsymbol{U}}]F'(x)=\sum_{\boldsymbol{V}}[x^{\boldsymbol{V}}]F(x)\cdot\omega^{\boldsymbol{U}\cdot\boldsymbol{V}}$（这是NTT的高维形式），定义$F'(x)\cdot G'(x)=\sum[x^{\boldsymbol{A}}]F(x)\cdot[x^{\boldsymbol{A}}]G(x)\cdot x^{\boldsymbol{A}}$，则有$F'(x)\cdot G'(x)=\big(F(x)G(x)\big)'$，这容易由$w^{\boldsymbol{U}\cdot(\boldsymbol{A}+\boldsymbol{B})}=w^{\boldsymbol{U}\cdot\boldsymbol{A}+\boldsymbol{U}\cdot\boldsymbol{B}}$得到。

2.6.1 逆变换

$[x^{\boldsymbol{U}}]F(x)=\frac{1}{3^n}\sum_{\boldsymbol{V}}[x^{\boldsymbol{V}}]F'(x)\cdot\omega^{-\boldsymbol{U}\cdot\boldsymbol{V}}$

3次单位根反演：$\sum_{i=0}^{2}(\omega^n)^i=\begin{cases}3 & n=0\\ \frac{(\omega^n)^3-1}{\omega^n-1}=0 & n=1,2\end{cases}$，逆变换是该反演的高维形式。

简要证明

$$\frac{1}{3^n}\sum_{\boldsymbol{V}}[x^{\boldsymbol{V}}]F'(x)\cdot\omega^{-\boldsymbol{U}\cdot\boldsymbol{V}}$$

$$=\frac{1}{3^n}\sum_{\boldsymbol{V}}\omega^{-\boldsymbol{U}\cdot\boldsymbol{V}}\sum_{\boldsymbol{W}}\omega^{\boldsymbol{V}\cdot\boldsymbol{W}}[x^{\boldsymbol{W}}]F(x)$$

$$=\frac{1}{3^n}\sum_{\boldsymbol{W}}[x^{\boldsymbol{W}}]F(x)\sum_{\boldsymbol{V}}\omega^{\boldsymbol{V}\cdot(\boldsymbol{W}-\boldsymbol{U})}$$

容易发现$\sum_{\boldsymbol{V}}\omega^{\boldsymbol{V}\cdot(\boldsymbol{W}-\boldsymbol{U})}$就是单位根反演在每个维度上的式子（即$n=(\boldsymbol{W}-\boldsymbol{U})_0, n=(\boldsymbol{W}-\boldsymbol{U})_1,\cdots$）的乘积，因而$\frac{1}{3^n}\sum_{\boldsymbol{V}}[x^{\boldsymbol{V}}]F'^{(x)}\cdot\omega^{-\boldsymbol{U}\cdot\boldsymbol{V}}=[x^{\boldsymbol{U}}]F(x)$。

2.6.3 3-FWT 与本题的联系

由于本题的模数$P=10^9+9$满足$P\ mod\ 3=1$，因此3次单位根容易通过原根得到。

将每次操作之后的结果用如上的一个多项式$F_c(x)$描述，由推论 1 以及$3-\text{FWT}$正变换的性质，容易发现$[x^{\boldsymbol{A}}]F_c(x), [x^{A}]F_c'(x)$的值都只与$\boldsymbol{A}$中0,1,2的个数有关。因此我们只需要$O(n^2)$个数就可以描述一个多项式，并且在此基础上进行变换。

不妨用$f(i,j)$表示在多项式F中，指数向量包含i个1, j个2的**一项**的系数，初始多项式$F_0(x)$只有$f_0(0,0)=1$有值，模拟$3-\text{FWT}$可知，$F_0(x)$的每一项系数都为 1。一轮操作的转移多项式$G(x)$由题目给定，$g(u,v)$即为$a_i=u, b_i=v$的个数。

我们需要计算$F_k(x)=F_0(x)\cdot G^k(x)$，只要求得$G'(x)$，${F_k}'(x)$，最后再通过逆操作得到答案。

由于$3-\text{FWT}$的正逆变换类似，而完成$3-\text{FWT}$后只需要$O(n^2)$次快速幂，因此这个算法的复杂度主要由$3-\text{FWT}$的速度决定。

2.7 算法 5

考虑变换表达式$[x^{\boldsymbol{U}}]F'(x)=\sum_{\boldsymbol{V}}\omega^{\boldsymbol{U}\cdot\boldsymbol{V}}[x^{\boldsymbol{V}}]F(x)$，要求$[x^{\boldsymbol{U}}]F'(x)$我们可以像前面的暴力算法一样，枚举$\boldsymbol{U},\boldsymbol{V}$对应位置为$U_i=x,V_i=y$的个数，复杂度为$O(n^8)$。

2.8 算法 6

为了便于描述，我们用二元形式幂级数来等价地表示多项式，记$F(x)$为$F(x,y)=\sum f(i,j)\cdot x^i y^j$。

设所求的$[x^{\boldsymbol{U}}]F'(x)=\sum_{\boldsymbol{V}}\omega^{\boldsymbol{U}\cdot\boldsymbol{V}}[x^{\boldsymbol{V}}]F(x)$的$\boldsymbol{U}$中包含$i$个1，$j$个2。用类似算法 2 的优化，我们用递推每一项的贡献来优化，记$h_{a,b}$为$\boldsymbol{V}$包含a个1、b个2的答案，依次考虑$\boldsymbol{U}$的每一位，枚举其对应$\boldsymbol{V}$的这一位为0,1,2，然后将对应的单位根作为权值乘入。

形式化的，就是求$H_{i,j}(x,y)=(1+x+y)^{n-i-j}+(1+\omega x+\omega^2 y)^i+(1+\omega^2 x+\omega^4 y)^j$。单次求系数复杂度为$O(n^3)$，总复杂度为$O(n^5)$。

2.9 算法 7

由$H_{i,j}(x,y)$的定义式，不难得到递推式：

$$H_{i,j}=H_{i,j-1}\cdot\frac{(1+\omega^2 x+\omega^4 y)}{1+x+y}$$

可以在枚举i,j的过程中，用单次$O(n^2)$的时间来维护乘法除法，总复杂度$O(n^4)$。

2.10 常数优化

由于本题需要两次做特殊的$3-\mathrm{FWT}$变换，同时每次又有多次乘除以及取模操作，因此本身常数并不小，本题的最后一档数据点考验轻度的常数优化能力。区分 $n=110$ 和 $n=120$ 也防止了部分实现不佳的代码被卡掉过多的分，而实际验证发现常数优化效果并不足以区分，赛场上还出现了远远快于标算的实现方式。

2.11 无需 FWT 的思路

参考一个类似的二维版本题目 Codechef 2019 OctChanllenge Queries on Matrix

3 总结

这道题思考难度因人而异，实际较为模板，主要考察对于单位根的应用和$3-\mathrm{FWT}$ 的理解、递推维护多项式的方法。标算的实现难度不高。

4 参考文献

[1] 张家琳, 《多项式乘法》, 2002 年集训队论文.
[2] Codechef 2019 OctChanllenge Queries on Matrix, 题解
https://discuss.codechef.com/t/jiit-editorial/39300