轮盘赌游戏 解题报告

华中师范大学第一附属中学 徐骁扬

题目大意

基于长为 n 的序列 p_i 有轮盘赌游戏如下：均匀随机地从 n 颗子弹中选择一颗子弹开始进行轮盘赌，每一轮都会激发一颗子弹，假设某一轮激发第 i 颗子弹，则子弹有 $1-p_i$ 的概率成功打出，游戏结束；子弹有 p_i 的概率卡壳，轮盘会转到第 $(i+d) \bmod n$ 颗子弹上，游戏进入下一轮。

由于 n 过大，本题将通过一个长为 m 的序列 p'_i 和一个二元组集合 S 生成一个长为 n 的序列 p_i：

首先对于所有 $i=0,1\dots n-1$，令 $p_i=p'_{i \bmod m}$，然后再对于所有 $(x,y)\in S$，将 p_x 修改成 y。

给出 $1+q+t$ 个二元组集合 $S_i,(i=0,1,\dots,q+t)$，生成方式如下：

对于 $i=0$，有 $S_i=\varnothing$；对于 $0<i\le q$，有 $S_i=\{(x_i,y_i)\}$；对于 $i>q$，有 $S_i=S_{a_{i-q}}\cap S_{b_{i-q}}$。

m 和 p'_i 全局固定，需要对所有 S_i，求出使用其生成的序列进行轮盘赌游戏的游戏轮数期望乘 n 的结果。

数据范围

$1\le d\le n\le 10^{16}$，$m\le 5000$。$1\le q,t\le 10^5$，$1\le x_i\le n$ 且 $\forall i\neq j,x_i\neq x_j$，$0\le p'_i,y_i<998244353$，$1\le a_i,b_i<i+q$ 且保证所有的 a_i,b_i 均不相同，数据保证 $\gcd(d,n)=1$，且对于任何询问，所有子弹被卡壳的概率之积对 998244353 取模不等于 1。

子任务

- Subtask 1（10 pts）：$1\le q,t,n\le 10^3$。
- Subtask 2（15 pts）：$1\le n\le 10^6$。
- Subtask 3（30 pts）：$d=1$。
- Subtask 4（20 pts）：$q=t=0$。
- Subtask 5（25 pts）：无特殊限制。

解题思路

做法 1（Subtask 1）

考虑如何计算答案。

记 f_i 表示即将激发第 i 颗子弹，期望还需多少轮结束游戏。

由于第 i 颗子弹有 p_i 的概率卡壳，所以有 $f_i=1+p_if_{(i+d)\bmod n}$。

对于所有 $i=0,1,2\dots n-1$，我们都可以得到上述方程。

考虑使用主元法解出该方程，例如对于 $n-d$，有 $f_{n-d}=1+p_{n-d}f_0$，同理 $f_{(n-2d)\bmod n}=1+p_{(n-2d)\bmod n}(1+p_{n-d}f_0)$，以此类推。由于 $\gcd(d,n)=1$，所以 $\{0,d,2d,\dots(n-1)d\}$ 刚好构成模 n 意义下的剩余系。则所有 f_i 都可以表示为 $f_i=a_i+b_if_0$ 的形式，同理同样存在 $f_0=a_0+b_0f_0$，其中有 $b_0=\prod\limits_{i=0}^{n-1}p_i$，由于数据保证 $\prod\limits_{i=0}^{n-1}p_i\neq 1$。所以可以解

出 $f_0 = \dfrac{a_0}{1-b_0}$，将 f_0 带入 $f_i = a_i + b_i f_0$ 得到所有 f_i。发现答案等于 $\sum\limits_{i=0}^{n-1} f_i$，对所有的 f_i 求和即可。

这样就在 $O(n)$ 的时间复杂度内求出答案。

对于修改，我们可以直接存储下来并维护合并，每一次都使用上面的 $O(n)$ 求解即可。时间复杂度 $O((q+t)n)$，期望获得 10 分。

做法 2（Subtask 2）

做法 1 中提到 $\{0, d, 2d, \dots (n-1)d\}$ 刚好构成模 n 意义下的剩余系，那么我们将子弹的顺序重排，让新序列中的第 i 颗子弹对应原序列的第 $id \bmod n$ 颗子弹。那么在新序列中的第 i 颗子弹卡壳后，就会转到第 $(i+1) \bmod n$ 颗子弹。相当于将 d 变成了 1。

那么做法一中的方程会变成 $f_i = 1 + p_i f_{(i+1) \bmod n}$。

考虑如何让求值支持快速修改。

如果要使用数据结构，就需要支持区间信息的快速合并。

考虑对于区间 $[l, r]$ 维护四个数 a, b, c, d，其表示 $f_l = a + b f_{r+1}$，$\sum\limits_{i=l}^{r} f_i = c + d f_{r+1}$（如果 $r+1 = n$ 则将 f_{r+1} 改为 f_0）。

发现这个信息是半群信息且可以快速合并。

设 $[l, mid]$ 处维护 $a_{ls}, b_{ls}, c_{ls}, d_{ls}$，$[mid+1, r]$ 处维护 $a_{rs}, b_{rs}, c_{rs}, d_{rs}$。

那么区间 $[l, r]$ 的信息可以快速计算出来：$a = a_{ls} + b_{ls} a_{rs}$，$b = b_{ls} b_{rs}$，$c = c_{ls} + c_{rs} + d_{ls} a_{rs}$，$d = d_{ls} b_{rs} + d_{rs}$。

这样就能够使用线段树等数据结构进行快速修改了。可以先将没有修改的线段树建出来，然后基于这棵树进行修改。

使用线段树合并维护修改，时间复杂度为 $O((n+q)\log n + t)$，期望获得 25 分。

做法 3 Pre

发现已经有 $d = 1$ 了，所以我们可以直接跳过调换序列顺序的步骤。

由于 n 很大，无法支持建出整棵树，那么我们需要考虑如何快速求出一个区间 $[l, r]$ 对应的信息 (a, b, c, d)。

对序列 p'_i 上的信息建立线段树，那么对于 $\left\lfloor \dfrac{l}{m} \right\rfloor = \left\lfloor \dfrac{r}{m} \right\rfloor$ 的部分直接区间查询。对于其余的部分我们可以将其拆成 p'_i 序列的一个前缀，一个后缀，以及全局信息的若干次幂的查询。两个部分的复杂度分别是 $O(\log m)$ 和 $O(\log \dfrac{n}{m})$ 的，在下文中我们统一认为是 $O(\log n)$ 的。

考虑使用动态开点线段树，对于所有非叶子节点，同时建立它的两个儿子节点，对于没有修改的一侧，使用上述方法直接求出其对应的半群信息。

后续就可以直接套用做法 2 了。

使用线段树合并维护修改，时间复杂度为 $O(q \log^2 n + t)$。

做法 3（Subtask 3）

考虑虽然没有办法直接建立线段树，但是我们可以尝试建立和线段树类似的结构。发现合并信息的时候都是二元合并，而二元合并结构树刚好是一棵二叉树。这启发我们将之前在线段树上维护的操作在这个二叉树上维护。

发现原序列 p_i 可以拆分成 $\left\lfloor \frac{n}{m} \right\rfloor$ 个序列 p'_i，以及序列 p'_i 的一个长度为 $(n \bmod m)$ 的前缀。

我们对于序列 p'_i 建立一棵线段树，然后使用类似倍增（快速幂）的方式将其复制成 $\left\lfloor \frac{n}{m} \right\rfloor$ 份，那么这个过程中，只会出现 $O(\log \frac{n}{m})$ 个本质不同的节点，最后拼上 p'_i 的前缀构成的线段树。虽然我们不能建出完整的二叉树，但是我们可以通过每一个节点由哪两个节点合并而来的，也就是我们可以从这些节点中来得到整个二叉树的结构，也就是我们得到了一个被压缩了的二叉树，其中最深节点的深度是 $O(\log \frac{n}{m} + \log m) = O(\log n)$ 的。仍然可以通过动态开点在这颗二叉树上进行修改和查询。

直接在这颗二叉树的基础上进行修改和查询，任何节点的信息都是可以 $O(1)$ 直接得到的，时间复杂度优化到了 $O(q \log n + t)$，期望获得 30 分，拼上做法 2 有 55 分。

做法 4（Subtask 4）

只需要对原序列求出答案即可。

考虑重排之后，新序列上的 p_i，就是 $p'_{(id \bmod n) \bmod m}$。$(id \bmod n) \bmod m$ 有两次取模，不好直接处理，考虑拆掉内部的那一个。

有 $id \bmod n = id - n \left\lfloor \frac{id}{n} \right\rfloor$，那么就有 p_i 的值为 p'_j，其中 $j = (id - n \left\lfloor \frac{id}{n} \right\rfloor) \bmod m$。

发现出现结构 $\left\lfloor \frac{id}{n} \right\rfloor$，神似 $\left\lfloor \frac{ai+b}{c} \right\rfloor$。所以尝试使用万能欧几里得来维护。

这里简述一下万能欧几里得的过程：

考虑直线 $\frac{ax+b}{c}$ $(0 \le b < c)$ 从 $x = 0$ 到 $x = L$ 的过程，每当遇到一次 $y = h$ 就执行一次 U 操作，遇到一次 $x = k$ 就执行一次 R 操作，如果同时与 $y = h$ 和 $x = k$ 相遇，优先执行 U 操作。其中 $k, h \in \mathbb{N}+$。

我们通过 a 和 c 的大小关系递归操作：

如果 $a \ge c$，那么每一个 R 操作之前至少有 $\left\lfloor \frac{a}{c} \right\rfloor$ 次 U 操作，不妨将 R 替换成 $U^{\left\lfloor \frac{a}{c} \right\rfloor} R$，那么就变成了对直线 $\frac{(a \bmod c)x + b}{c}$ 考虑问题了。

如果 $a < c$，那么我们考虑第 i 个 R 操作之前有 $\left\lfloor \frac{ai+b}{c} \right\rfloor$ 个 U 操作，那么第 j 个 U 操作之前就会有 $\left\lfloor \frac{cj-b-1}{a} \right\rfloor$ 个 R，为了使得结构与原问题相同，也就是满足 $(0 \le b < c)$ 的限制，那么我们首先可以变成 $\left\lfloor \frac{c(j-1)+c-b-1}{a} \right\rfloor$，最后变成 $\left\lfloor \frac{c-b-1}{a} \right\rfloor + \left\lfloor \frac{c(j-1)+(c-b-1) \bmod a}{a} \right\rfloor$ 的结构，相当于要递归到 $\left\lfloor \frac{c(j-1)+(c-b-1) \bmod a}{a} \right\rfloor$ 的结构，然后特殊处理一下前缀的 R 和第一个 U，以及后缀的 R。

由于这个递归过程和欧几里得算法是一致的，所以可以证明其复杂度是 $O(\log \max(a, c))$ 的。

现在考虑在这一个问题中的 U 操作和 R 操作分别是什么：

我们实时维护一个下标指针 $flag$，初始时有 $flag = 0$，且 $flag$ 的值是在模 m 的意义下进行的。

U 操作是让 $flag \leftarrow flag - n$。

R 操作是让 $flag \leftarrow flag + d$，同时在序列末尾加上 p_{flag}。

我们尝试维护前面做法中提到的半群信息 (a, b, c, d)。

但是由于 $flag$ 是不确定的，但是由于 $flag$ 只可能在 $\{0, 1 \dots m-1\}$ 中取到，同时 m 并不是很大，所以我们可以对于每一个 $flag = i$ 都维护出其对应的信息，那么我们就可以支持 $O(m)$ 合并了。

由于万能欧几里得的时间复杂度是 $O(\log n)$ 的，也就意味着本质不同的 R 和 U 加起来最多只有 $O(\log n)$ 个，所以暴力存储下来的空间复杂度就是 $O(m \log n)$ 的。

这样我们可以在 $O(m \log n)$ 的时间复杂度内解决问题，期望得分 20 分，拼上做法 2 和做法 3 有 75 分。

做法 5（Subtask 5）

仿照 做法3 中的操作，将万能欧几里得的合并结构看作一棵被压缩了的、最深深度为 $O(\log n)$ 的二叉树，所以仍然可以使用动态开点的方式进行修改和查询。

使用线段树合并可以做到 $O(q \log n + t)$，期望得分 100 分。

参考资料

北大集训 2021（CTT 2021）Day4 T3 以及相关题解。