题目选讲

翁文涛 [1]

中山纪念中学

January 22, 2017

[1] qq:815939360

Contents I

XORGRID

AGENTS

GEOCHEAT

THREECOL

Tastes Like Winning

Fibonacci gcd again

Catcation rental

How many substrings

Tree coordinates

Definite Random Walks

XORGRID [2]

给定一个 $n \times n$ 的矩阵，每个位置有一个权值 $A[i][j]$。一开始你在左上角 $(1,1)$，要走到右下角 (N,N)，每一步只能往右或往下走，一条路径的权值定义为经过的点的权值的异或和。问权值最大可以是多少？

$n \leq 18, 0 \leq A[i][j] \leq 10^9$

[2] https://www.codechef.com/problems/XORGRID

- 直接爆枚路径的做法当然是不科学的。注意到这里 n 比较小，有没有什么折半之类的做法？

- 直接爆枚路径的做法当然是不科学的。注意到这里 n 比较小，有没有什么折半之类的做法？
- 不妨将对角线抽出来，做一个二维 meet in the middle。爆搜起点到对角线，以及终点到对角线的所有可能路径。对于对角线上每个点，开个 TRIE 记录一下两边的情况，然后就相当于给两个集合，求最大的 xor，直接在 TRIE 上走一遍就好了。

- 直接爆枚路径的做法当然是不科学的。注意到这里 n 比较小，有没有什么折半之类的做法?
- 不妨将对角线抽出来，做一个二维 meet in the middle。爆搜起点到对角线，以及终点到对角线的所有可能路径。对于对角线上每个点，开个 TRIE 记录一下两边的情况，然后就相当于给两个集合，求最大的 xor，直接在 TRIE 上走一遍就好了。
- 考虑起点到对角线上的方案数，恰好就是 2^n，因此这题的复杂度就是 $O(2^n \log A[i][j])$。

AGENTS [3]

给定两个多项式 $Q(x), T(x)$，满足 $Q(x)=\sum_{i=0}^{n} q_i x^i, T(x)=\sum_{i=0}^{n} t_i x^i$，需要你求出多项式 $P(x)$，满足：

$$P(x)=Q(x)+\int_0^1 T(x\cdot s)P(s)\,ds$$

$n, m \leq 50, q_n, t_m \neq 0$，你可以假定解总是存在的。

[3] https://www.codechef.com/problems/AGENTS

$$P(x)=\sum_{i=0}^{n} q_i x^i + \int_0^1 P(s)[\sum_{i=0}^{m} t_i (xs)^i] ds$$
$$P(x)=\sum_{i=0}^{n} q_i x^i + \sum_{i=0}^{m} t_i x^i \int_0^1 P(s) s^i ds$$

注意到 $\int_0^1 P(s)s^i ds$ 只与 i 有关，不妨设为 c_i，那么有：

$$
\begin{aligned}
P(x) &= \sum_{i=0}^{n} q_i x^i + \sum_{i=0}^{m} t_i x^i c_i \\
c_i &= \int_0^1 P(s) s^i ds \\
&= \int_0^1 s^i \left(\sum_{j=0}^{n} q_j s^j + \sum_{j=0}^{m} t_j s^j c_j \right) \\
&= \sum_{j=0}^{n} q_j \int_0^1 s^{i+j} + \sum_{j=0}^{m} t_j c_j \int_0^1 s^{i+j} \\
&= \sum_{j=0}^{n} \frac{q_j}{i+j+1} + \sum_{j=0}^{m} \frac{t_j}{i+j+1} c_j
\end{aligned}
$$

那么我们就可以得到 c 之间的关系，$m+1$ 条方程 $m+1$ 个变量，高斯消元解出 c 的值这题就做完了。

GEOCHEAT [4]

给定平面上的 n 个点 (x_i, y_i)，对于每个 $1 \leq i \leq n$，求出只保留前 i 个点，平面上点对的最远距离。

$n \leq 750000|x_i|, |y_i| \leq 10^9$

数据的生成方式是：以某种方式把所有点的坐标生成后，把点随机排列。

[4] https://www.codechef.com/problems/GEOCHEAT

题解

- 假如只求前 n 个的最远距离，那么直接用经典的旋转卡壳就好了。

题解

- 假如只求前 n 个的最远距离，那么直接用经典的旋转卡壳就好了。
- 点是随机排列的？有什么用吗？和最小圆覆盖那个随机排列有什么联系？能不能把暴力改成一个能套用最小圆覆盖时间复杂度分析的暴力？

题解

- 假如只求前 n 个的最远距离，那么直接用经典的旋转卡壳就好了。
- 点是随机排列的？有什么用吗？和最小圆覆盖那个随机排列有什么联系？能不能把暴力改成一个能套用最小圆覆盖时间复杂度分析的暴力？
- 考虑 solve(n) 表示要求出前 n 个点的最远距离。直接跑一遍旋转卡壳，得到最远点分别为 x, y，那么对于 $max(x, y) \sim n$ 的答案，都应该等于 n 时的答案。那么我们就忽略掉 $max(x, y)$ 以后的东西，直接递归求解 solve($\max(x, y) - 1$)。

时间复杂度分析 I

我们先求出从 n 个数中随机选两个不同的数，他们最大值的期望是多少。

$$E[max(x,y)] = \frac{\sum_{x=1}^{n}\sum_{y=x+1}^{n} y}{\binom{n}{2}} = \frac{\sum_{i=1^n} i^2 - i}{\binom{n}{2}} = \frac{2n-1}{3}$$

时间复杂度分析 II

也就是说，令 $T(n)$ 表示把前 n 个的答案算出来的期望复杂度，那么有

$$\begin{aligned} T(n) &= T(\frac{2}{3}n) + O(n\log n) \\ &= (1 + \frac{2}{3} + \frac{4}{9} + \cdots)n\log n \\ &= O(n\log n) \end{aligned}$$

THREECOL [5]

给定一个 $n \times n$ 的网格图，图上每个格子一开始会有种颜色，总共只有三种颜色 A, B, C。每次，你可以选定两个相邻的颜色不同的格子，把他们的颜色都变成剩下的那种颜色，假如你选定了两个颜色相同的格子，那么不产生影响。现在希望你给出一种操作序列，使得对于所有的初始颜色，这种操作序列都能将所有格子的颜色都变成相同的某种颜色。

$n \leq 20$ 且 n 不是 3 的倍数。要求序列长度不超过 100000。checker 会尽量判出不正确的序列。

[5] https://www.codechef.com/problems/THREECOL

转换模型

不妨将这个网格图用一条链串起来，假如我们能把链的问题解决，那么这个网格图自然可解。考虑一条链，我们从左往右搞，每 4 个一组，下一组的第一个位置是这一组的最后一个位置，对于每一组，我们把他们变成相同的颜色（但不能保证变成确定的某种颜色）。比如说我们先操作 $[1, 4]$，再操作 $[4, 7]$ 这样。由于 $n \bmod 3 > 0$，所以 $n^2 \bmod 3 = 1$，因此必然每个点都能被叠到。把 4 个东西变成一样的很简单嘛，自己手玩一下就好了。但现在的问题是，我们无法保证所有的操作最后都变成同种颜色。

这样操作过去后，我们会发现链会长成类似于这种样子：AAACCCBBBAAAA 这样，最后 4 个是相同的，那么对于像 BBBA 这种情况，我们可以 BBBA->BBCC->BAAC->->CCAC->CBBC->AAAA。那么以此类推，我们做完第一遍后，再倒过来做，就能全部变成一样的了。

Tastes Like Winning [6]

给定 n, m，对于一个序列 $a_1, \cdots, a_n$，他是合法的当且仅当 $a_i \in [1, 2^m - 1]$ 且 a_i 互不相同且他们的异或和不为 0。问有多少合法的序列。

$n, m \leq 10^7$

[6] https://www.hackerrank.com/contests/w26/challenges/taste-of-win

- 首先我们可以先求出 $w[n]$ 表示确定 n 个数且两两不同的方案数。$w[n] = w[n-1] \times (2^m - n)$。令 $f[n]$ 表示 n 个不同的数异或和为 0 的方案数。那么答案就是 $w[n] - f[n]$。剩下的问题就是怎么求 $f[n]$。

- 首先我们可以先求出 $w[n]$ 表示确定 n 个数且两两不同的方案数。$w[n] = w[n-1] \times (2^m - n)$。令 $f[n]$ 表示 n 个不同的数异或和为 0 的方案数。那么答案就是 $w[n] - f[n]$。剩下的问题就是怎么求 $f[n]$。
- 因为我们要强制使得 n 个数异或和为 0，那么假如前 $n-1$ 个数确定了，这第 n 个数就确定了。现在的问题就是第 n 个数不能取为 0 且不能和之前的数重复。那么首先 $f[n]$ 初值为 $w[n-1]$。那么当前 $n-1$ 个数异或起来为 0 时，第 n 个数取值就是 0，不合法，因此 $f[n] = w[n-1] - f[n-1]$。但还有一个不能重复的问题。

假如说第 n 个数和之前某个数重复了，相当于剩下的 $n-2$ 个数的异或和为 0，那么这个数有 $n-1$ 种位置，并且第 n 个数和这个数有 $2^m-(n-1)$ 种取值（只要不和剩下的数重复就好了），因此

$f[n]=w[n-1]-f[n-1]-(n-2)[2^m-(n-1)]f[n-2]$，递推一遍就好了。

Fibonacci gcd again [7]

给定 $a_0, a_1, a_2, b_0, b_1, b_2, n$，定义 f_n 表示斐波拉契数列的第 n 项，试求出 $\gcd(a_0F_n + a_1F_{n+1} + a_2F_{n+2}, b_0F_n + b_1F_{n+1} + b_2F_{n+2}) \bmod 10^9 + 7$。

$0 \le a_0, a_1, a_2, b_0, b_1, b_2, n \le 10^9$

[7] https://www.hackerrank.com/contests/infinitum11/challenges/fibonacci-gcd-again

- 问题显然可以变成求 $\gcd(a_0F_n + a_1F_{n+1}, b_0F_n + b_1F_{n+1})$。注意，这里的 a_0, a_1, b_0, b_1 不一定是题目中的 a_0, a_1, b_0, b_1，只是某个常数的代号。接下来的部分同理。

- 问题显然可以变成求 $\gcd(a_0F_n + a_1F_{n+1}, b_0F_n + b_1F_{n+1})$。注意，这里的 a_0, a_1, b_0, b_1 不一定是题目中的 a_0, a_1, b_0, b_1，只是某个常数的代号。接下来的部分同理。
- 我们可以先对 a_1, b_1 进行辗转相除，在辗转相处的同时，维护出新的 a_0 和 b_0，那么我们必能变成求 $\gcd(a_0F_n + a_1F_{n+1}, b_0F_n)$。

- 问题显然可以变成求 $\gcd(a_0F_n+a_1F_{n+1}, b_0F_n+b_1F_{n+1})$。注意，这里的 a_0, a_1, b_0, b_1 不一定是题目中的 a_0, a_1, b_0, b_1，只是某个常数的代号。接下来的部分同理。
- 我们可以先对 a_1, b_1 进行辗转相除，在辗转相处的同时，维护出新的 a_0 和 b_0，那么我们必能变成求 $\gcd(a_0F_n+a_1F_{n+1}, b_0F_n)$。
- 令 $g=\gcd(a_0F_n+a_1F_{n+1}, F_n)$，因为 $\gcd(F_n, F_{n+1})=1$，所以 $g=\gcd(F_n, a_1)=\gcd(F_n \bmod a_1, a_1)$。那么我们所求的答案 ans 就等于 $g\times\gcd(\frac{a_0F_n+a_1F_{n+1}}{g}, b_0)$，把 g 乘进去，就有 $ans=\gcd(a_0F_n+a_1F_{n+1}, b_0g)$，那么只要分别求出 $F_n \bmod b_0g$ 和 $F_{n+1} \bmod b_0g$ 再代进去 gcd 中即可。

Catcation rental [8]

一个数轴有 n 个点，有一个常数 k，还有 m 次操作。每次操作，会给定 l_i, r_i，假如 $r_i - l_i + 1 < k$ 或者说 $[l_i, r_i]$ 这一段有点已经被覆盖了，那么这个操作会被跳过，否则会把这一段中所有点都标记为已覆盖。现在问你，当 $k = 1, 2, \cdots, n$ 时，最终数轴上分别会有多少点被覆盖。不同 k 之间独立。

$n, m, k \leq 10^5$

[8] https://www.hackerrank.com/contests/w20/challenges/catcation-rental

- 考虑直接枚举 k，求出 k 的答案。当长度至少为 k 时，总共可能不被跳过的操作个数只会有 $\frac{n}{i}$ 个。

- 考虑直接枚举 k，求出 k 的答案。当长度至少为 k 时，总共可能不被跳过的操作个数只会有 $\frac{n}{i}$ 个。
- 不妨设一个过程 $\text{solve}(l, r)$ 表示我们只考虑操作区间完全在 $[l, r]$ 中的操作。那么每次相当于找到一个完全在这个区间中，长度大于等于 k，并且出现时间最早的一个操作 l_i, r_i，把 $r_i - l_i + 1$ 加入答案，接着递归 $\text{solve}(l, l_i - 1), \text{solve}(r_i + 1, r)$。要找这样的一个操作，我们可以按 k 从大到小做，维护一棵树套树，每次就相当于查 $r_i \le r$ 且 $l_i \ge l$ 且已经被加入的最早的区间。在树套树上维护一个加入时间的最小值即可。

- 考虑直接枚举 k，求出 k 的答案。当长度至少为 k 时，总共可能不被跳过的操作个数只会有 $\frac{n}{i}$ 个。
- 不妨设一个过程 $\mathrm{solve}(l, r)$ 表示我们只考虑操作区间完全在 $[l, r]$ 中的操作。那么每次相当于找到一个完全在这个区间中，长度大于等于 k，并且出现时间最早的一个操作 l_i, r_i，把 $r_i - l_i + 1$ 加入答案，接着递归 $\mathrm{solve}(l, l_i - 1), \mathrm{solve}(r_i + 1, r)$。要找这样的一个操作，我们可以按 k 从大到小做，维护一棵树套树，每次就相当于查 $r_i \leq r$ 且 $l_i \geq l$ 且已经被加入的最早的区间。在树套树上维护一个加入时间的最小值即可。
- 每次查询都要 $O(\log^2 n)$ 的时间，而 $\sum_{i=1}^{n} \frac{n}{i} = O(n \log n)$，因此总复杂度就是 $O(n \log^2 n)$。

How many substrings [9]

给定一个长度为 n 的只包含小写字母的字符串 S。有 m 个询问，每次给定 l, r，问 $S[l..r]$ 中有多少不同的子串。允许离线。

$n, m \le 100000$

[9]https://www.hackerrank.com/contests/w27/challenges/how-many-substrings

- 假如只有一个询问怎么做？那当然就是对查询串跑 SAM，然后对于 fail 树上的点，维护出其对应串集合的长度区间 $(l_i, r_i]$，其中 l_i 就是其父亲的 r_j。那么这个点对答案的贡献就是 $r_i - l_i$。

- 假如只有一个询问怎么做？那当然就是对查询串跑 SAM，然后对于 fail 树上的点，维护出其对应串集合的长度区间 $(l_i, r_i]$，其中 l_i 就是其父亲的 r_j。那么这个点对答案的贡献就是 $r_i - l_i$。
- 现在有多个询问，不妨离线地来做。把右端点从左往右扫，尝试用数据结构维护出对于每个 l，$S[l..r]$ 的答案是多少。

- 假如只有一个询问怎么做？那当然就是对查询串跑 SAM，然后对于 fail 树上的点，维护出其对应串集合的长度区间 $(l_i, r_i]$，其中 l_i 就是其父亲的 r_j。那么这个点对答案的贡献就是 $r_i - l_i$。
- 现在有多个询问，不妨离线地来做。把右端点从左往右扫，尝试用数据结构维护出对于每个 l，$S[l..r]$ 的答案是多少。
- 考虑先把整个串的 SAM 以及 fail 树建出。每次从左往右扫，相当于不停地加入一个前缀 r，也就是会更新这个前缀在 fail 树上的所有祖先的 right 集合。对于会被更新到的一个点 i，他的对应长度为 $(len_{fa_i}, len_i]$，假设这个点目前 right 集合中最大的位置为 $last$，那么对于每个长度 d 满足 $len_{fa_i} < d \le len_i$，$[last - d + 2, r - d + 1]$ 这些位置的答案都会加一（为什么？）。

- 因为查询是单点查询，那么我们可以用一棵线段树来维护 ans 的差分，即线段树上存储的是 $ans_i - ans_{i-1}$，那么之前的修改就比较好维护了（把图形画出来就好理解了。

- 因为查询是单点查询，那么我们可以用一棵线段树来维护 ans 的差分，即线段树上存储的是 $ans_i - ans_{i-1}$，那么之前的修改就比较好维护了（把图形画出来就好理解了。
- 直接暴力往根跳复杂度还是不对的。

- 因为查询是单点查询，那么我们可以用一棵线段树来维护 ans 的差分，即线段树上存储的是 $ans_i - ans_{i-1}$，那么之前的修改就比较好维护了（把图形画出来就好理解了。
- 直接暴力往根跳复杂度还是不对的。
- 可以发现，假如说某个点 i 及其父亲 fa_i 对应的 $last$ 都相同，那么他们的贡献可以一起算。也就是说对于连续一段 $last$ 相同的，他们的贡献都可以一起算。那么我们可以维护一棵 LCT，LCT 上每条重链对应的 $last$ 都相同。每次插入一个 r，就相当于把这个点 Access 一下。

- 因为查询是单点查询，那么我们可以用一棵线段树来维护 ans 的差分，即线段树上存储的是 $ans_i - ans_{i-1}$，那么之前的修改就比较好维护了（把图形画出来就好理解了。
- 直接暴力往根跳复杂度还是不对的。
- 可以发现，假如说某个点 i 及其父亲 fa_i 对应的 $last$ 都相同，那么他们的贡献可以一起算。也就是说对于连续一段 $last$ 相同的，他们的贡献都可以一起算。那么我们可以维护一棵 LCT，LCT 上每条重链对应的 $last$ 都相同。每次插入一个 r，就相当于把这个点 Access 一下。
- 总时间复杂度是 $O(n \log^2 n)$。

Tree coordinates [10]

给定一棵 n 个点的树。有 M 个二元组 (l_i, r_i)，二元组 $(l_i, r_i), (l_j, r_j)$ 的距离定义为 $dis(l_i, l_j) + dis(r_i, r_j)$，其中 $dis(u, v)$ 表示 u, v 在树上的距离。试求出二元组之间的最大距离。

$n \leq 500000$

[10] https://www.hackerrank.com/contests/world-codesprint-8/challenges/tree-coordinates

- 假如说不是二元组而只是一个点，那么启发式合并就能解决了。那么对于二元组，我们能不能对其中一个启发式合并，然后另外一个用一些奇怪的手段去考虑其贡献？

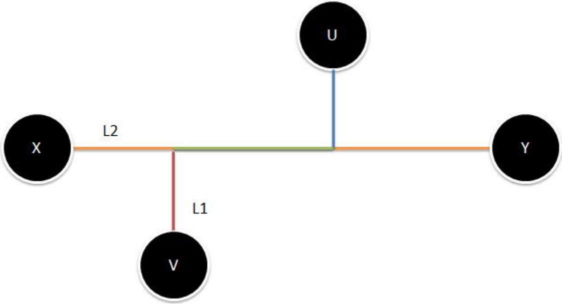
- 假如说不是二元组而只是一个点，那么启发式合并就能解决了。那么对于二元组，我们能不能对其中一个启发式合并，然后另外一个用一些奇怪的手段去考虑其贡献？
- 不妨考虑对 l_i 启发式合并，比如说二元组 (l_i, r_i) 和 (l_j, r_j) 是在点 u 处被合并的，那么 l_i, l_j 的贡献就是 $dep(l_i) + dep(l_j) - 2dep(u)$，也就是说，对于在点 u 处被合并的二元组，我们只需要知道 $\max(dep(l_i) + dep(l_j) + dis(r_i, r_j))$。

- 假如说不是二元组而只是一个点，那么启发式合并就能解决了。那么对于二元组，我们能不能对其中一个启发式合并，然后另外一个用一些奇怪的手段去考虑其贡献？
- 不妨考虑对 l_i 启发式合并，比如说二元组 (l_i, r_i) 和 (l_j, r_j) 是在点 u 处被合并的，那么 l_i, l_j 的贡献就是 $dep(l_i) + dep(l_j) - 2dep(u)$，也就是说，对于在点 u 处被合并的二元组，我们只需要知道 $\max(dep(l_i) + dep(l_j) + dis(r_i, r_j))$。
- 我们知道假如要求 $\max(dis(r_i, r_j))$，有一种做法就是维护出全局最远的两个点，那么每个点对应的最远点必然在这两个点中。

- 同理，对于 $\max(dep(l_i) + dep(l_j) + dis(r_i, r_j))$，因为 $dep(l_i)$ 和 r_j 无关，$dep(l_j)$ 和 r_i 无关，也就是说 $dep(l_i)$ 只是 r_i 的一个属性，我们不妨也动态的维护出这个距离定义下的最远的两个点 r_i, r_j，那么可以证明，每个点对应的最远点必然也在这两个点之间。

- 同理，对于 $\max(dep(l_i) + dep(l_j) + dis(r_i, r_j))$，因为 $dep(l_i)$ 和 r_j 无关，$dep(l_j)$ 和 r_i 无关，也就是说 $dep(l_i)$ 只是 r_i 的一个属性，我们不妨也动态的维护出这个距离定义下的最远的两个点 r_i, r_j，那么可以证明，每个点对应的最远点必然也在这两个点之间。
- 那么在启发式合并的过程中，我们只要能维护出这两个端点，这题就做完了。维护端点很简单，合并的时候暴力更新就好了。注意更新答案时必须要保证这两个端点不在同一棵树内。
  只要用 $O(n \log n)$ 预处理，$O(1)$ 求 LCA 的 RMQ 做法，这题就可以 $O(n \log n)$ 解决了。

正确性证明 I



不妨令图中 X, Y 为在这个距离定义下的最远点，U, V 为另外两个点，并且满足 U, V 的距离大于 U, X 与 U, Y 的距离，这里的距离是之前的定义 $dep(l_i) + dep(l_j) + dis(r_i, r_j)$。那么首先假如 U, V 不经过 X, Y 这条链，那么 X, Y 必然不是最远点。否

正确性证明 II

则，不妨假设 V 更考虑 X 这一侧，另外的情况同理。由于 U, V 距离大于 U, X 距离，那么就有 $L2 > L1$，那么最远点应该是 V, Y 而不是 X, Y，与假设矛盾，因此不可能出现 U, V 距离同时大于 U, X 与 U, Y 距离的情况，Q.E.D。

Definite Random Walks [11]

给定 n 个点，每个点 i 都有一条出边 go_i。还有一个 M 面的骰子，对于第 i 面，这一面朝上的概率为 p_i，保证 $p_1 + \cdots + p_M = 1$。一开始你以等概率站在某一个点上。现在你要扔 K 次骰子，每次假如是第 i 面朝上，那么你会从当前节点出发，走 $i-1$ 条出边。问最后停在点 $i = 1..n$ 上的概率。答案对 998244353 取模。

$n \leq 60000, 4 \leq m \leq 100000, 1 \leq K \leq 1000$

[11] https://www.hackerrank.com/contests/w28/challenges/definite-random-walks

- 这是一个环套树林，我们直接每个联通块单独考虑就好了。接下来都假设这个图是联通的。

- 这是一个环套树林，我们直接每个联通块单独考虑就好了。接下来都假设这个图是联通的。
- 不妨令这个联通块的环长为 C，树的部分最大深度为 D，那么假如说最终走了 x 步，若 $x < D$，那么不做修改，否则，我们令 $x' = (x - D) \bmod C + D$，这样走 x 步和走 x' 步其实是等价的。那么我们不妨令 $P[i]$ 表示最终等效为走了 i 步的概率，那么我们直接用初始的 p 用快速幂加 NTT 求出 p^K，在做卷积时顺便维护一下这个等价关系就好了。

- 这是一个环套树林，我们直接每个联通块单独考虑就好了。接下来都假设这个图是联通的。
- 不妨令这个联通块的环长为 C，树的部分最大深度为 D，那么假如说最终走了 x 步，若 $x < D$，那么不做修改，否则，我们令 $x' = (x - D) \bmod C + D$，这样走 x 步和走 x' 步其实是等价的。那么我们不妨令 $P[i]$ 表示最终等效为走了 i 步的概率，那么我们直接用初始的 p 用快速幂加 NTT 求出 p^K，在做卷积时顺便维护一下这个等价关系就好了。
- 更具体地来讲，假设我们当前求出 $C = A \times B$，其中 A, B 是两个长度为 $C + D$ 的数组，$\times$ 表示卷积，那么对于 $i < D$ 的 $C[i]$，我们不做处理，对于 $i \geq D$，我们就可以把 i 等价为 $(i - D) \bmod C + D$

- 求出转移的概率后，我们就考虑怎么求出最终停在某个点上的概率。有环的情况不好处理，我们看看能不能把环干掉。随便令环上的一点为根，设为 $root$，以 $root$ 为第 0 个点，把环上接下来的点都确定了，令为 $cir[i]$。我们假如把必须经过 $go[root]$ 的情况统计完，那么这条边就能拆掉了。

- 求出转移的概率后，我们就考虑怎么求出最终停在某个点上的概率。有环的情况不好处理，我们看看能不能把环干掉。随便令环上的一点为根，设为 $root$，以 $root$ 为第 0 个点，把环上接下来的点都确定了，令为 $cir[i]$。我们假如把必须经过 $go[root]$ 的情况统计完，那么这条边就能拆掉了。
- 假如说一个点 j 到环上的 $cir[i]$，那么经过的步数就是 $i + dep(j)$，这里 $dep(j)$ 的定义就是，把 $go[root]$ 拆掉后，以 $root$ 为根的树中，j 的深度。也就是说，$ans[cir[i]] = \sum_{j=1}^{n} P[i + dep(j)]$，我们只要把 ans 翻转就可以用一次卷积来算出答案了。

接下来考虑树的情况，也就是说，对于一个点 i，$ans[i] = \sum_{j \in \mathrm{subtree}(i)} P[dep(j) - dep(i)]$。直接做似乎很难做，接下来我们讲两种做法来解决树上的情况。

分块

我们把这棵树的 dfs 序求出来，令点 i 对应子树的区间是 $[l_i, r_i]$，$refer[i]$ 表示 dfs 序为 i 的点，那么有 $ans[i] = \sum_{l_i \le j \le r_i} P[dep(refer[j]) - dep(i)]$，我们把 dfs 序分块，对于每一块，我们令 Cnt_i 表示这一块中深度为 i 的有这么多，求出 C 和 P 的卷积，设块数为 L，那么这里的复杂度就是 $O(Ln\log n)$，当我们要查 i 点的答案时，对于左右多出来的我们暴力枚举，中间的一整块我们就直接用 $(C \times P)[dep(i)]$ 的值来更新就好了，这里的复杂度是 $O(n(L + \frac{n}{L}))$。

令 $L = \sqrt{\frac{n}{\log n}}$，总复杂度就是 $O(n\sqrt{n\log n})$。

点分治

不妨用点分治来考虑子树对自身的影响。我们知道一个点和他的祖先的路径必然会经过某个重心。在点分治的过程中，令当前重心为 R，找到当前联通块中深度最小的点，假设为 u，把 u 到 R 这一段抽出来，令 L_i 表示第 i 个点。记 Cnt_i 统计不经过 u 到 R 中任意点，但在联通块中的，到 R 距离为 i 的点的个数。那么 $Ans[L_i] = Ans[L_i] + \sum_{j=0}^{maxdis} P[i+j]Cnt_j$，做一遍卷积，把答案加到祖先的答案中。

总的复杂度就是 $O(n\log^2 n)$。