胡策的小树解题报告

长郡中学 陈胤伯

1 试题大意

在 OI 界，有一位无人不知无人不晓，OI 水平前无古人后无来者的胡策，江湖人称一眼秒题胡大爷。

胡策最近从一名自称是小 O 的神秘男子那里收到了一棵神奇的小树苗。

这是一棵 n 个节点的有根树，节点标号为 $1 \sim n$，其中 1 号点为根。

这棵有根树上每个点都有一个权值，点 i 的权值为 a_i。$a_{1\sim n}$ 构成了一个 $0 \sim n-1$ 的排列，且 $a_1 = 0$。

胡策大爷十分喜欢猴子，他打算在这棵树上养 n 只猴子。初始时，每个节点上将放着恰好一只猴子。猴子们十分好动，每过一秒，每只在 i 节点的猴子会设法往 i 的父亲节点上跳，有 $p(i)$ 的概率成功跳到父亲节点；否则跳跃失败，将等概率地随机落到子树 i 里某个节点上（包括点 i）。

对于 $2 \le i \le n$，有 $p(i) = a_i/n$；因为根节点没有父亲，所以 $p(1) = 0$。

在第 i 秒，胡策会观察并记录这 n 只猴子中成功跳上父亲结点的猴子所占的比例 g_i。胡策认为 $g_{0\sim T}$ 的平均值就是这群猴子们生活的幸福指数，为保证准确，其中 T 很大很大，为 $(n+1)^{(99999^{(99999^{99999})})}$。

为了让猴子们的幸福指数的期望更大，胡策又从那名自称是小 O 的神秘男子那里买来了一袋叫“金坷垃”的肥料。如果给这棵有根树掺 x 克的金坷垃，那么这棵树每个点 i 的权值将变化成 $(a_i + x) \mod n$。因为胡策是土豪有钱任性，x 可以取任意非负整数。

请你告诉胡策，他该掺多少克的金坷垃，才能使猴子们幸福指数的期望最大呢？你只需要输出这个最大的幸福指数期望，四舍五入保留 8 位小数。

2 输入格式

第一行一个正整数 n。

第二行 n 个用空格隔开的非负整数，第 i 个为节点 i 的父亲节点编号 fa_i。（$fa_1 = 0$，对于 $i > 1$ 有 $1 \le fa_i < i$）

第三行 n 个用空格隔开的非负整数，为一个 $0 \sim n-1$ 的排列，第 i 个表示 a_i。

3 输出格式

一行一个实数，表示掺适量的金坷垃时的最大幸福指数期望，四舍五入保留 8 位小数。你的输出和标准输出完全一致时将获得该测试点的分数。

4 数据范围

对于 10% 的数据：$n = 2$。

对于 20% 的数据：$n \le 5$。

对于 30% 的数据：$n \le 100$。

对于 50% 的数据：$n \le 2000$。

对于 70% 的数据：$n \le 100000$。

对于 100% 的数据：$2 \le n \le 500000$。

数据保证有一定梯度。

另外，因为出题人很懒，所以数据都是随机生成的。即：节点 i 的父亲是从 $1 \sim i-1$ 中随机选取的，$a_{1\sim n}$ 是一个 $0 \sim n-1$ 的随机排列。

5 算法介绍

首先，不考虑掺金坷垃的影响，我们对题目稍微作一下分析。

本来我们是每一天统计 n 只猴子中成功的猴子比例，再求这个比例的平均值，即幸福指数。但注意到，每只猴子对幸福指数的贡献其实是独立的。进一步地，对于一只猴子，它初始时在什么位置也是无关紧要的。因为 T 足够大，可以认为是无穷大，而一只猴子总是会在有限步数内跳到根，即便我们忽视这只猴子之前的贡献，把它看作是从根出发的，对答案的影响也是可以忽略的。

定义 P 为：一只猴子，随便选一个点出发（比如说根），然后跳 T 轮，跳跃成功的次数占总次数的比例的期望。那么原来一只猴子对幸福指数的期望贡献为 $\frac{1}{n}P$，因此 n 只猴子对幸福指数的期望贡献为 P，即 P 就是幸福指数的期望。

5.1 算法 0

对于 $n = 2$ 的数据，我们注意到掺了金坷垃后一定 $a_1 = 0$、$a_2 = 1$，否则若 $a_2 = 0$ 将导致 $p(1) = 0, p(2) = 0$，猴子们的跳跃将永远无法成功。

根据我们的跳跃规则，不难发现，$1, 2$ 每个点都有 0.5 的概率跳到另一个点、0.5 的概率原地不动。那么我们可以认为，猴子在每个点待的时间占总时间的 50%。又因为只有在节点 2 有 0.5 的概率成功，那么幸福指数的期望就是 $0.5 \times 0.5 = 0.25$。

时间复杂度：$O(1)$

期望得分：10

5.2 算法 1

我们枚举掺多少克金坷垃，注意只需要从 0 枚举到 $n - 1$，然后考虑怎么计算幸福指数。

类似算法 0 的思想，只要求出在每个点 i 期望待的时间比例 x_i，最后 $\sum_{i=1}^{n} x_i p(i)$ 即为答案。

那么 x_i 怎么求呢？

我们可以高斯消元。首先，我们把猴子的随机跳跃看作在某个有向图中随机沿着边走。对于一个点 i，假如有 t 条进入 i 的入边，其中第 j 条边走的概率

为 h_j，那么有等式：

$$x_i = \sum_{j=1}^{t} x_{from(j)} h_j$$

最后，根据定义有 $\sum_{i=1}^{n} x_i = 1$。

联立上述所有方程就可以解出来 x 了。

时间复杂度：$O(n \times n^3)$

期望得分：$20 \sim 40$

5.3 算法 1 迭代版

用算法 1 的思路，不妨设猴子从根出发，令 $f_{i,v}$ 表示第 i 轮猴子在点 v 的概率。对于同一个 v，随着 i 的增加 $f_{i,v}$ 会逐渐趋向一个定值。

在程序里设一个枚举的 i 的上限，然后暴力递推 f 即可。因为本题精度要求不高，适当调整参数可以获得不错的分数。

期望得分：$20 \sim 60$

5.4 算法 2

这里介绍另一种计算幸福指数的方法。

考虑二分答案，那就需要对于给定的 e 判断是否 $ans \geq e$（ans 表示幸福指数）。记 f_i 表示第 i 天猴子是否跳跃成功，那么：

$$ans = \frac{\sum_{i=1}^{T} f_i}{T}$$

我们要判定的式子也就是：

$$\frac{\sum_{i=1}^{T} f_i}{T} \geq e$$

$$\sum_{i=1}^{T} f_i \geq e \times T$$

$$\sum_{i=1}^{T} (f_i - e) \geq 0$$

于是二分答案之后，模型变成了：猴子跳跃成功一次获得 $1 - e$ 的权值，失败一次获得 $-e$ 的权值，问跳无穷多轮之后手上的权值期望是否非负。

这个怎么求呢？我们首先找到 $a_i = 0$ 的那个 i（一定有且仅有一个），显然猴子在有限步内会跳进子树 i，并且一辈子也出不去了。这样我们只需考虑在子树 i 里跳即可。

接下来的讨论都是针对子树 i 的。

不妨让这只猴子从根 $root$ 出发（之前讨论过了，这是不影响答案的），由于无穷多的跳跃等价于一次又一次从根回到根的过程，我们只要想办法统计出“从根再一次回到根”这个过程中的权值收益是否非负就好了。

定义 out_u 表示，从点 u 出发，经过一系列跳跃后，第一次跳出了子树 u，这个过程期望拿了多少权值。

对于 $u \neq root$，讨论一下第一步跳跃成功与否，令 p 表示 $p(u)$，得到递推式：

$$out_u = p \times (1 - e) + (1 - p) \times (-e + \sum_{v \in subtree(u)} \frac{1}{size_u} (\sum_{d \in path(v,u)} out_d))$$

值得注意的是等式右边有 out_u 这一项，移项到左边除掉即可。

求出了 out_u，从根回到根期望拿的权值和也就是：

$$-e + \sum_{v \in subtree(root)} \frac{1}{size_{root}} (\sum_{d \in path(v,root) \text{ and } d \neq root} out_d)$$

时间复杂度：$O(n \times n^3 \log v)$

期望得分：20

5.5 算法 3

稍微优化一下算法 2。

观察之前的递推式：

$$out_u = p \times (1 - e) + (1 - p) \times (-e + \sum_{v \in subtree(u)} (\frac{1}{size_u} \times \sum_{d \in path(v,u)} out_d))$$

我们对 out_d 进行了过多的重复统计。事实上对于某一个 out_d，在统计的时候被枚举到了恰好 $size_d$ 次。

所以得到新的递推式：

$$out_u = p \times (1 - e) + (1 - p) \times (-e + \sum_{d \in subtree(u)} \frac{1}{size_u} \times out_d size_d)$$

时间复杂度：$O(n \times n^2 \log v)$

期望得分：30

5.6 算法 4

接着算法 3。

那个 $\sum_{d\in subtree(u)} \frac{1}{size_u} \times out_d size_d$ 事实上是一个子树和，顺便记录并递推一下每次就不用枚举 d 了。

时间复杂度：$O(n \times n \log v)$

期望得分：50

5.7 算法 5

注意枚举掺多少金坷垃的这个过程中，每个点的 a 都有且仅有一次 $= 0$。

每次我们解决的问题规模是 $size_d$ 的，如果稍微注意一点的话，很容易就能做到 $O(size_d)$ 解决问题，这样总复杂度是 $O(\sum_{i=1}^{n} size_i)$ 的。

因为树的形态随机，事实上 $\sum_{i=1}^{n} size_i = O(n \log n)$。

证明：考虑 $\sum_{i=1}^{n} size_i$ 中，每个点 i 被统计了深度次，所以等于 $\sum_{i=1}^{n} deep_i$。

令 f_i 表示 i 个点的随机树中点的深度和的期望，显然有 $f_i = f_{i-1} + (f_{i-1}/(i-1)+1)$，则 $f_n = \frac{n}{1} + \frac{n}{2} + \frac{n}{3} + ... + \frac{n}{n} = O(n \log n)$。

时间复杂度：$O(n \log n \log v)$

期望得分：$60 \sim 70$

5.8 算法 6

在算法 5 的基础上我们再稍作优化。

首先，我们以随机的顺序枚举掺金坷垃的量 x。

注意到之前我们二分之后有个过程是 $check(e)$，能够在 $O(size)$ 的时间里判定答案是否 $\geq e$，那么在枚举到某个 x 的时候，我们先 $check()$ 一下之前已经求出的最大幸福指数，看看这一轮是否有可能出现更优的值。如果是，我们再进去二分求解。

考虑加上这个小优化之后的复杂度。

对于外层的 $check()$，调用一次是 $O(size)$ 的，一共调用 n 次，总复杂度为 $O(\sum_{i=1}^{n} size_i) = O(n \log n)$。

对于内层的二分求解，调用一次是 $O(size \log v)$的。由于我们是以随机顺序枚举 x，那么枚举到第 i 次的时候，这一次是前 i 次中幸福指数最高的概率

是 $\frac{1}{i}$，所以一共期望进入 $O(\log n)$ 次内层二分求解。由于 a 是随机的，每一次进入内层后问题的 $size$ 的期望大小是 $\frac{\sum_{i=1}^{n} size_i}{n} = O(\log n)$ 的，因此总复杂度为 $O(\log^2 n \times \log v)$。

时间复杂度：$O(n \log n + \log^2 n \log v)$

期望得分：100

6 得分估计

在集训队互测中，预计有 3 人能得到 100 分，有 5 人能得到 60 ~ 70 分，有 4 人能得到 10 ~ 50 分。