胡策的小树 解题报告

杭州第二中学 陈思禹

1 试题来源

2015年集训队互测

2 试题大意

胡策最近从一名神秘男子那里收到了一棵神秘的小树苗。这是一棵n（$n \le 500000$）个节点的有根树，节点标号为$1-n$，其中1号点为根。每个点i都有一个权值a_i，保证权值是一个0到$n-1$的排列，且$a_1 = 0$。

胡大爷十分喜欢猴子，他打算在这棵树上养n只猴子，初始时每个点上将恰好放一只。猴子们很好动，每过一秒在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；若失败，将等概率地随机落到子树i的某个节点上（包括i）。

$p(i)$是一个函数，对于$2 \le i \le n$，$p(i) = \frac{a_i}{n}$；由于根节点没有父亲，$p(1) = 0$。

第i秒胡策会观察记录n只猴子成功跳上父亲结点的猴子所占的比例g_i。胡策认为g_{0-T}的平均值就是这群猴子的幸福指数。为了保证准确，T可以视为无穷大。

为了让猴子们的幸福指数更大，胡策又从那名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，每个点i的权值都将变化成$(a_i + x)\ mod\ n$。因为胡策有钱任性，x可以取任意非负整数。

请你告诉胡策，掺了金坷垃后猴子们的最大幸福指数是多少。四舍五入保留8位小数。

另外所有数据都是随机生成的，即i的父亲是从1到$i-1$中随机选取的，a数组也是一个随机排列。

3 分析

先考虑问题的转化。显然这群猴子的幸福指数就相当于一只猴子在无限长时间中跳跃成功的次数所占的比例。一种想法是最终每个点都会有一个稳定的猴子在其上面的概率x_i，这个用高斯消元解一下即可，但是显然会超时。

另一种想法则是分数规划。设二分的答案为e，则我们需要判定的是$\frac{\sum_{i=1}^{T} f_i}{T} \geq e$。其中若时刻$i$跳跃成功则$f_i = 1$，否则$f_i = 0$。转化一下即得$\sum_{i=1}^{T}(f_i - e) \geq e$，也就相当于把权值$f_i$变为$f_i - e$。而且可以注意到不论掺多少金坷垃必有一个点$i$使得$p_i = 0$，设该点为$root$。且猴子一定在有限次跳跃中跳到该子树中并且再也出不来。因此只需判定在该子树内从$root$回到$root$的期望权值和是否非负即可。

定义out_u表示从$root$为根的子树中的点u（$u \neq root$）出发第一次跳出子树u的期望权值。$size_u$表示u子树大小，$subtree_u$表示子树u 中的点的集合，$path_{u,v}$表示u, v路径上的点的集合。则最简单的递推式如下：

$$out_u = p_u(1 - e) + (1 - p_u)\left(-e + \frac{1}{size_u} \sum_{v \in subtree_u} \sum_{a \in path_{u,v}} out_a\right)$$

其中的二重和式容易发现可以优化为$\sum_{v \in subtree_u} out_v size_v$。这个求和中的$out_u$部分可以移到左边除一下即可，剩下的是一个$u$子树中除了$u$的点的求和，显然可以边求$out_u$边递推一下。这样求所有$out_u$的时间复杂度就优化为$O(size_{root})$了。最终从$root$回到$root$的期望权值即为$-e + \frac{1}{size_{root}} \sum_{u \in subtree_{root}, u \neq root} out_u size_u$。

由于对于不同施金坷垃的量$root$会取遍$1 - n$，再加上每次二分次数设为$O(k)$，总时间复杂度为$O(k\sum_{i=1}^{n} size_i)$。而$\sum_{i=1}^{n} size_i = \sum_{i=1}^{n} dep_i$，其中$dep_i$为$i$号点的深度，$dep_1 = 1$。鉴于这棵树是随机的，这个和其实是$nlogn$级别的。因此这种方法的时间复杂度为$O(knlogn)$，由于$n$较大并不能通过所有测试数据。

考虑利用随机的条件再优化。我们要求的只是最大幸福指数，因此我们可以按随机的顺序枚举施用金坷垃的量，对于一个枚举到的x，先令二分答案e等于当前答案的最大值，检验一下要判定的式子是否成立。如果成立再二分答案。这种最优化剪枝在此题中可以获得很好的效果。对于每一个x必定要先判定一次，总时间复杂度$O(nlogn)$。而第i个枚举的金坷垃量是前i个的最大值的概率是$\frac{1}{i}$，也就是调用二分的期望次数是$\sum_{i=1}^{n} \frac{1}{i} = logn$，单次二分的期望复杂度

为$O(k\frac{\sum_{i=1}^{n} size_i}{n}) = O(klogn)$，总的二分时间复杂度为$O(klog^2n)$。

最后的算法时间复杂度为$O(nlogn + klog^2n)$，可以通过所有测试数据。