# IOI2021 中国国家集训队第一阶段作业试题交流
# 《Game on Tree》解题报告

长沙市长郡中学　徐舟子

## 目录

**1 前言** **2**

**2 简要题意** **2**

**3 定义与约定** **3**

**4 算法介绍** **3**

4.1 基于结论推导的动态规划算法 . . . . . 3

4.1.1 简单的暴力 . . . . . 3

4.1.2 由暴力启发的动态规划算法 . . . . . 4

4.1.3 对动态规划算法的优化 . . . . . 5

4.1.4 任意模数情况下的扩展 . . . . . 5

4.2 更优的动态规划算法 . . . . . 6

4.2.1 重新观察题目性质 . . . . . 6

4.2.2 全新的动态规划状态 . . . . . 6

4.2.3 当只求根节点答案时的优化 . . . . . 7

**5 后记** **7**

**6 参考资料** **7**

# 1 前言

本文探讨了笔者在 IOI2021 国家集训队第一阶段作业试题交流环节中的自选题《Game on Tree》。本题是一道组合计数题，考察了多个计数技巧以及一些结论推导，希望能给读者带来一些启发。

# 2 简要题意

A 和 B 在玩游戏。

给定一棵 $n$ 个点的有根树，标号从 1 到 $n$，根的标号是 1。每个节点的儿子数只能为 0 或 2。将节点分成下列三类：

- $A$：到根距离为偶数的所有非叶节点的集合，其中根到自己的距离为 0
- $B$：到根距离为奇数的所有非叶节点的集合
- $L$：叶节点的集合

而对于所有叶子节点 $u \in L$，都会被分配一个二元组 $(c(u), d(u))$。其中 $c$ 和 $d$ 可以看成是定义域为 $L$ 的函数。

游戏开始前：

A 对于 $A$ 中的每个点 $u$，从 $u$ 的两条指向儿子的边中选择一条，将其标记为重边；

B 对于 $B$ 中的每个点 $u$，从 $u$ 的两条指向儿子的边中选择一条，将其标记为重边。

A 的选择可以看成一个定义域为 $A$ 的映射函数 $f$；

B 的选择可以看成一个定义域为 $B$ 的映射函数 $g$。

那么一个有序对 $(f, g)$ 被称为一组策略。可以看出，A 有 $2^{|A|}$ 种不同的选择，B 有 $2^{|B|}$ 种不同的选择。

所以一共有 $2^{|A|+|B|}$ 组不同的策略。

当策略确定之后，从根开始，不断沿着重边往下走，直到走到某个叶子节点 $u$。这时 A 的分数是 $c(u)$，B 的分数是 $d(u)$。一组策略 $(f, g)$ 如果满足以下条件，那么该组策略是一个纳什均衡点：

- 在 A 的策略不改变的情况下，B 无法通过改变自己的策略获得更高分。即在策略 $(f, g')$ 中 B 的得分总小于等于 $(f, g)$ 中 B 的得分。
- 在 B 的策略不改变的情况下，A 无法通过改变自己的策略获得更高分。即在策略 $(f', g)$ 中 A 的得分总小于等于 $(f, g)$ 中 A 的得分。

给定树形态和 $k$。令 $c$ 和 $d$ 的值域是 $Z \cap [1, k]$，那么一共有 $k^{2|L|}$ 组不同的 $(c, d)$。现在要求对每组不同的 $(c, d)$ 都计算纳什均衡点的个数。由于 $k^{2|L|}$ 过大，所以输出这 $k^{2|L|}$ 个答案的和。这个和可能还是过大，所以输出和对 $p$ 取模后的值。即，求

$$\left( \sum_{c \in L \to [k]} \sum_{d \in L \to [k]} F(c, d) \right) \bmod p$$

其中 $L \to [k]$ 是所有定义域为 $L$，值域为 $[k] = \{1, 2, \ldots, k\}$ 的函数集。$F(c, d)$ 是指给定 $c$ 和 $d$ 情况下的纳什均衡点个数。

对于每一个节点 $s$，删除树的其他部分，只保留点 $s$ 的子树，并以点 $s$ 作为树根开始游戏，将上面问题的答案记为 $Ans_s$。

你需要求出所有 $Ans_s \bmod p$。

$n \leq 5000, k \leq 10^9, p \in [n+1, 10^6]$。

- Subtask 1(20pts):$n = 99, k \leq 100$，$p$ 是质数。
- Subtask 2(30pts):$n = 99$。
- Subtask 3(50pts):$n = 4999$。

# 3 定义与约定

- 我们用 $leaf(x)$ 表示 $x$ 子树中叶子节点的数量，$nleaf(x)$ 表示 $x$ 子树中非叶子结点的数量，$cA(x)$ 表示 $x$ 子树中 $A$ 集合里结点的数量，$cB(x)$ 表示 $x$ 子树中 $B$ 集合里结点的数量。
- 对于一个非叶子结点 $x$，我们用 $ls(x)$ 表示它的左儿子，$rs(x)$ 表示它的右儿子。

# 4 算法介绍

## 4.1 基于结论推导的动态规划算法

先考虑求根结点的答案。

首先枚举最终纳什均衡点的终点 $u$，那么从它到祖先的边都为重边。它的得分为 $c(u), d(u)$。

### 4.1.1 简单的暴力

为了方便描述，我们介绍**支配**的概念。我们称一个非 $u$ 叶子结点 $v$ **被** $c(u)$ **支配**，等价于 A 可以通过修改自己的策略，使最终节点取到点 $v$。**这意味着必须有** $c(v) \leq c(u)$。同理，称一个非 $u$ 叶子结点 $v$ **被** $d(u)$ **支配**，等价于 B 可以通过修改自己的策略，使最终节点取到点 $v$。**这意味着必须有** $d(v) \leq d(u)$。这本质上是一种对叶子节点权值的限制条件。

假设我们先用暴力枚举确定了每个非叶子节点指向的重边，我们可以使用下面的结论求出每个叶子节点被支配的情况，继而直接求出选择权值的方案数。

> 如果有一个点 $x$，且 $y, z$ 为点 $x$ 的两个儿子，$x$ 到 $y$ 的边被设为重边：
>
> 若 $x \in B$：那么 $z$ 子树中所有叶子节点均不会被 $c(u)$ 支配（因为 A 没有办法将边 $x \to z$ 标记为重边）。这意味着，$z$ 子树中所有叶子节点的 $c$ 权值可以任选。
>
> 若 $x \in A$：同理，$z$ 子树中所有叶子节点均不会被 $d(u)$ 支配。

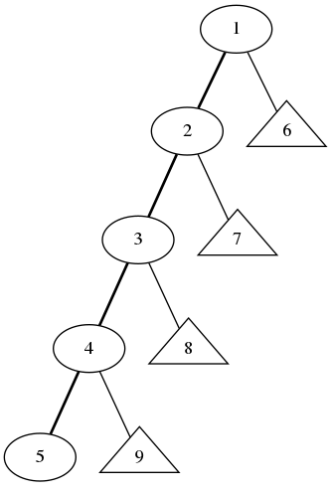
在确定了 $c(u), d(u)$ 和每个点所选的重边以后，我们可以按深度从浅到深使用上述结论，求出每个叶子点被 $c(u)$ 和 $d(u)$ 支配的情况。如果有 $x$ 个叶子节点被 $c(u)$ 支配，$y$ 个叶子节点被 $d(u)$ 支配，则对叶子节点选择权值方案数为 $c(u)^x d(u)^y k^{(|L|-x-1)+(|L|-y-1)}$。

故总的选择权值方案数是 $\sum_{c(u)=1}^{k} \sum_{d(u)=1}^{k} c(u)^x d(u)^y k^{(|L|-x-1)+(|L|-y-1)}$，且对这个式子的计算有很多优化空间。

#### 4.1.2 由暴力启发的动态规划算法

由于效率过低，我们不能直接确定每一条重边。我们尝试利用点 $u$ 先去掉一些支配关系。

根据上面的结论，我们可以将树分为若干子树，每个子树只可能被一种权值支配。如下图，在这张图中 $u=5$，子树 $6,8$ 中的点只可能被 $c(u)$ 支配，记为 $S_c(u)$；子树 $7,9$ 中的点只可能被 $d(u)$ 支配，记为 $S_d(u)$。




仿照上面“按深度从浅到深使用结论”的暴力过程，我们可以得到一个动态规划算法：

设 $f_{c,x,i}$ 表示现在递归到点 $x$ 的子树中，所有叶子节点仍然被 $c(u)$ 支配（且均不被 $d(u)$ 支配），且 $c(u)=i$ 时，选择非叶节点重边和所有 $L$ 中节点的 $c$ 权值的方案数；

设 $f_{d,x,i}$ 表示现在递归到点 $x$ 的子树中，所有叶子节点仍然被 $d(u)$ 支配（且均不被 $c(u)$ 支配），且 $d(u)=i$ 时，选择非叶节点重边和所有 $L$ 中节点的 $d$ 权值的方案数。

由于两个状态本质相同，我们考虑 $f_{c,x,i}$ 的转移，之后的题解中也可能省略只与 $f_d$ 有关的讨论。

- 当 $x$ 是一个叶子节点时，点 $x$ 一定被 $c(u)$ 支配。
- 当 $x\in B$ 时，我们可以通过重边的选择，对于一个 $x$ 儿子的子树，去掉它被 $c(u)$ 的支配关系。对于支配关系被去掉的那个子树，它们选 $c$ 的权值的方案数已经没有任何限制，选择重边也对权值没有影响。对于支配关系仍存在的那个子树，变为一个子问题递归处理。
- 当 $x\in A$ 时，我们可以去掉一个 $x$ 子树被 $d(u)$ 的支配关系，但是子树只被 $c(u)$ 支配，所以这是无用的，只需乘重边选择方案数，对两个子树递归处理。

以下是动态规划的方程式。

$$f_{c,x,i}=\begin{cases} i & x\in L \\ 2\times f_{c,ls(x),i}\times f_{c,rs(x),i} & x\in A \\ f_{c,ls(x),i}\times 2^{nleaf(rs(x))}\times k^{leaf(rs(x))}+f_{c,rs(x),i}\times 2^{nleaf(ls(x))}\times k^{leaf(ls(x))} & x\in B \end{cases}$$

回到原问题。求出两个 DP 之后，枚举最终纳什均衡点的终点 $u$，方案数为

$$\sum_{c(u)=1}^{k}\sum_{d(u)=1}^{k}\left(\left(\prod_{x\in S_{c(u)}} f_{c,x,c(u)}k^{leaf(x)}\right)\left(\prod_{x\in S_{d(u)}} f_{d,x,d(u)}k^{leaf(x)}\right)\right)$$
$$=\sum_{c(u)=1}^{k}\left(\prod_{x\in S_{c(u)}} f_{c,x,c(u)}k^{leaf(x)}\right)\sum_{d(u)=1}^{k}\left(\prod_{x\in S_{d(u)}} f_{d,x,d(u)}k^{leaf(x)}\right)$$

用 DFS 枚举 $u$，使用栈维护 $\prod_{x\in S_{c(u)}} f_{c,x,c(u)}k^{leaf(x)}$，可以做到时间复杂度 $O(nk)$。

对每一个子树求答案时，该做法时间复杂度为 $O(n^2k)$，可以通过 Subtask 1。

#### 4.1.3 对动态规划算法的优化

进一步观察 DP 方程式。可以发现，如果设 $F_{c,x}(y)=f_{c,x,y}$，那么 $F_{c,x}(y)$ 实际上是一个关于 $y$ 的不超过 $leaf(x)$ 次的多项式，所以 $\sum_{c(u)=1}^{k}\prod_{x\in S_{c(u)}} f_{c,x,c(u)}k^{leaf(x)}$ 是一个关于 $k$ 的不超过 $leaf(1)+1$ 次多项式。

我们只需求出前 $leaf(1)+2$ 项的值，即可用插值法求出前 $k$ 项的和。

时间复杂度为 $O(n\cdot\min(n,k))$。

#### 4.1.4 任意模数情况下的扩展

可以发现，上述做法中只有插值涉及到了除法。但事实上插值并不需要除法即可实现。

若 $S-1$ 次多项式 $f(x)$ 在 $1,2,\dots S$ 处的点值分别已知，由拉格朗日插值法，我们可以知道 $f(k)(k>S)$ 的值为：

$$f(k)=\sum_{i=1}^{S}f(i)\prod_{j\neq i,j\in[1,S]}\frac{k-j}{i-j}$$

我们考虑

$$\begin{aligned}&\prod_{j\neq i,j\in[1,S]}\frac{k-j}{i-j}\\=&\frac{(k-1)!}{(k-S-1)!(k-i)}\frac{1}{(i-1)!(S-i)!}(-1)^{S-i}\\=&\frac{(k-1)!}{(k-S-1)!i!(S-i)!}\frac{i}{k-i}(-1)^{S-i}\\=&\binom{k-1}{i}\binom{k-1-i}{S-i}\frac{i}{k-i}(-1)^{S-i}\\=&\binom{k-1}{i-1}\binom{k-1-i}{S-i}(-1)^{S-i}\end{aligned}$$

我们去掉了系数的分母。这也告诉我们，实际上插值时计算的系数都是整数。

使用 exLucas 定理即可求解系数。

对每一个子树求答案时，该做法时间复杂度为 $O(n^2\cdot\min(n,k))$，且支持对任意模数求答案，可以通过 Subtask 2。

## 4.2 更优的动态规划算法

上面的做法并没有逃出一开始对终点 $u$ 的枚举，所以统计答案时必须用一个不优美的方式维护，也不能发现很多优秀的性质。

### 4.2.1 重新观察题目性质

再次分析题目，结合上面的 DP 过程，我们可以发现 $L$ 中节点的 $c$ 权值的选择只和 $B$ 中节点的重边选择有关，与 $A$ 中节点的重边选择无关。同理，$L$ 中节点的 $d$ 权值的选择只和 $A$ 中节点的重边选择有关。而且如果我们分开确定 $A, B$ 中节点的重边，确定后可以唯一确定 $u$ 结点。

据此，我们尝试优化 DP 的状态，顺便省去对结点 $u$ 的枚举。

### 4.2.2 全新的动态规划状态

设 $f_{c,x,i}$ 表示考虑点 $x$ 的子树且 $c(u) = i$ 时选择 $B$ 中节点的重边和 $L$ 中节点的 $c$ 权值的方案数；
设 $f_{d,x,i}$ 表示考虑点 $x$ 的子树且 $d(u) = i$ 时选择 $A$ 中节点的重边和 $L$ 中节点的 $d$ 权值的方案数。
仿照上文的 DP 转移，我们考虑 $f_{c,x,i}$ 的转移，$f_{d,x,i}$ 转移类似。

- 当 $x$ 是一个叶子节点时，如果点 $x$ 不是 $u$，则 $c(x)$ 的方案数为 $i$，否则方案数是 1。在这里，我们直接将方案数乘 $i$，最后再将方案数除 $i$ 即可。
- 当 $x \in B$ 时，处理方式与上文中的 DP 一样。
- 当 $x \in A$ 时，由于 $f_c$ 并不计算选择 $A$ 重边的方案数，所以只需递归到两个子树就可以了。

以下是动态规划的方程式。

$$f_{c,x,i} = \begin{cases} i & x \in L \\ f_{c,ls(x),i} \times f_{c,rs(x),i} & x \in A \\ f_{c,ls(x),i} \times 2^{cB(rs(x))} \times k^{leaf(rs(x))} + f_{c,rs(x),i} \times 2^{cB(ls(x))} \times k^{leaf(ls(x))} & x \in B \end{cases}$$

最后的答案是 $\left(\sum_{i=1}^{k} f_{c,1,i}/i\right)\left(\sum_{i=1}^{k} f_{d,1,i}/i\right)$，可以使用插值优化。
注意，这里统计答案时出现了除法。我们设 $g_{c,x,i} = f_{c,x,i}/i$，可以直接得到 $g_c$ 的转移方程式：

$$g_{c,x,i} = \begin{cases} 1 & x \in L \\ g_{c,ls(x),i} \times g_{c,rs(x),i} \times i & x \in A \\ g_{c,ls(x),i} \times 2^{cB(rs(x))} \times k^{leaf(rs(x))} + g_{c,rs(x),i} \times 2^{cB(ls(x))} \times k^{leaf(ls(x))} & x \in B \end{cases}$$

答案为 $\left(\sum_{i=1}^{k} g_{c,1,i}\right)\left(\sum_{i=1}^{k} g_{d,1,i}\right)$。新的转移方程式中没有除法运算，在模数任意时也可以计算。
由于找到了更多的性质，这个做法可以直接对每一个子树求答案，比上一个 DP 更优秀。
时间复杂度为 $O(n^2)$，可以通过本题。

#### 4.2.3 当只求根节点答案时的优化

如果只需求根节点的答案，我们可以直接维护 DP 时的多项式而不使用插值。

设 $G_{c,a}(x)$ 是一个多项式，满足 $\forall i \in [1,k], G_{c,a}(i) = g_{c,a,i}$，$G_{d,a}(i)$ 类似。我们可以得到 $G_{c,a}(x)$ 的转移：

$$G_{c,a}(x) = \begin{cases} 1 & x \in L \\ G_{c,ls(a)}(x) \times G_{c,rs(a)}(x) \times x & x \in A \\ G_{c,ls(a)}(x) \times 2^{cB(rs(a))} \times k^{leaf(rs(a))} + G_{c,rs(a)}(x) \times 2^{cB(ls(a))} \times k^{leaf(ls(a))} & x \in B \end{cases}$$

使用树链剖分 + 分治 FFT 优化，可以在 $O(n \log^3 n)$ 的时间内求出根节点的多项式 $G_{c,1}, G_{d,1}$ 值。

之后，我们要求

$$\left(\sum_{i=1}^{k} G_{c,1}(i)\right)\left(\sum_{i=1}^{k} G_{d,1}(i)\right)$$

由于 $G_{c,1}(x), G_{d,1}(x)$ 均为 $O(n)$ 次的多项式，我们可以用伯努利数在 $O(n \log n)$ 的时间内求出答案。

时间复杂度为 $O(n \log^3 n)$。

## 5 后记

本题是一道改编题，原题来自江苏省 2019 年集训。本题在原题的基础上对数据范围做了加强，原题的数据范围是 $n \leq 5000, k \leq 20$，$p$ 为质数，只求根节点的答案。

笔者在考试时推导出了本题时间复杂度为 $O(n \cdot \min(n,k))$ 的解法，并于今年 7 月将本题放在了校内测试，当时的数据范围是 $n \leq 5000, k \leq 10^9$，$p$ 为质数，只求根节点的答案。这个方法虽然比较不优美，但笔者认为一些结论的推导比较巧妙，可以作为一道中档题。在尝试继续优化该做法时，笔者将思路局限于用多项式优化 DP 的转移，结果没有找到任何思路。

最近，笔者与北大附中的李白天同学交流，学习到了任意模数时的插值做法。李白天同学还使用数学方法推导出了更优秀的 DP 方程式。笔者受到启发并重新思考此题，又发现了一些隐藏在题目中的结论，并推导出了同样的 DP 方程式。

文中已经提到，本题还有一种加强方式，但笔者认为多项式并不是本题的考察重点，故选择了现在的数据范围。本题考察的知识点有动态规划和拉格朗日插值法，还考察了对题目性质的分析能力与数个计数技巧，需要选手有较强的思维能力和一些数学知识。由于原题目的做法不太优美且难以优化，本题没有给数据范围更小的部分分，希望选手多对问题进行思考分析之后给出优秀的解法。

## 6 参考资料

[1] OI-wiki exLucas 定理

[2] OI-wiki 伯努利数