《Alice、Bob 与 DFS》解题报告

浙江省江山中学 郑路明

2021 年 10 月 3 日

1 题目大意

有一个 n 个点的有向无环图，第 i（$1 \leq i \leq n$）个点有 m_i 条有序的出边 $e_{i,1}, e_{i,2}, \ldots, e_{i,m_i}$。每个点要么是黑点，要么是白点。

有 k 个程序，第 i 个程序形如从 r_i 开始，对 r_i 的直接或间接后继进行深度优先搜索。

对于黑点，程序会无条件地依次访问其出边，并递归搜索；对于白点，程序会在访问每条出边并递归搜索前询问是否跳过该出边。

所有程序在对某个点的搜索开始前会自动暂停。一开始，所有程序均处于开始搜索起始点前的暂停状态。

有两名玩家在进行博弈。双方轮流进行以下操作，不能操作者输：

- 选择一个搜索过程尚未完全结束的程序，将其从暂停状态中恢复；
- 继续执行该程序，在程序将要访问白点的出边时决定是否跳过该边，直到该程序完全结束或再次暂停。

双方均采取最优策略。判断先手是否必胜。

2 数据范围

本题共有 6 个子任务：

1.（5 分）对于所有 $1 \le i \le n$，$c_i = 0$。

2.（15 分）$k = 1$, $r_1 = 1$。

3.（15 分）$n \le 10$, $\sum_{i=1}^{n} m_i \le 10$, $k \le 10$。

4.（20 分）对于所有 $1 \le i \le n$，$c_i = 1$；对于每个点 u，只有至多一条边的终点为 u；对于所有 $1 \le i \le k$，图中没有边的终点为 r_i。

5.（20 分）对于所有 $1 \le i \le n$，$c_i = 1$。

6.（25 分）没有额外的约束条件。

对于所有数据：

- $1 \le n \le 2 \times 10^5$；
- 对于所有 $1 \le i \le n$，$c_i \in \{0, 1\}$；
- 对于所有 $1 \le i \le n$，$0 \le m_i \le 4 \times 10^5$；
- $\sum_{i=1}^{n} m_i \le 4 \times 10^5$；
- 对于所有 $1 \le i \le n$, $1 \le j \le m_i$，$i < e_{i,j} \le n$；
- $1 \le k \le 2 \times 10^5$；
- 对于所有 $1 \le i \le n$，$1 \le r_i \le n$。

3 解题过程

3.1 子任务 1：所有点均为黑点

胜负仅由所有程序暂停次数之和的奇偶性决定。

因为只有黑色的点，所以无论双方如何决策，每个程序的暂停次数是固定的。

理解题意并实现一个非常简单的递推程序即可在 $\Theta(k+n+\sum m)$ 的时间复杂度内通过该子任务。

3.2 子任务 2：只有一个程序

在只有一个程序的情况下只需要考虑每个状态是否是必胜态。

我们将一个过程开始时的轮到的玩家称作先手。对于每个点，如果双方均采取最优策略，从该点出发的搜索过程必定是以下四种类型之一：

1. 一旦进入该过程，先手必胜（先手可以选择该过程结束时轮到的玩家）；
2. 一旦进入该过程，先手必败（后手可以选择该过程结束时轮到的玩家）；
3. 该过程结束时一定轮到先手；
4. 该过程结束时一定轮到后手。

递推计算从每个点出发的搜索过程的类型。为了方便起见，接下来将其称为该点的类型。

所有白点要么是第 1 类，要么是第 3 类：

- 如果该点有指向第 2 类点的出边，该点是第 1 类。先手可以选择跳过该边之前的所有边，并访问该出边，使后手陷入必败局面。
- 如果该点有指向第 3 类点的出边，该点是第 1 类。先手可以选择跳过该边之前的所有边，再跳过该出边。如果这样会导致先手失去胜利，先手可以改为选择该出边，将不可能胜利的局面交给对手。

- 否则，该点是第 3 类：指向第 1 类点的出边一定均被跳过，而第 4 类出边访问与否不改变先后手。

对于黑点而言，模拟程序运行过程，依次遍历其出边，按终点的类型维护当前玩家，直到所有边均被遍历或遍历到第 1 类或第 2 类的点即可得出该黑点的类型。

总时间复杂度 $\Theta(n+m)$。

3.3 子任务 3：点数、总边数、程序数均不超过 10

这是一个平等博弈，因此我们只需要计算从每个点为起始点的程序对应的子博弈的 SG 函数 [1] 值。先手必胜当且仅当这 k 个程序对应的子博弈的 SG 函数值的异或和非 0。

任何一个处于暂停状态的搜索程序的状态都可以用一个栈表示，这个栈存储了暂停时各层 DFS 过程调用下层 DFS 过程时使用的边（包括暂停时即将调用的 DFS 过程）。我们将暂停时即将调用的 DFS 过程的点称为当前点。

我们用 z 表示一个已经结束执行的程序所处的状态，其 SG 函数值为 0。

由于数据范围很小，对于每个程序暴力计算其所有状态的 SG 函数值即可通过该子任务。

时间复杂度 $O(k\,\mathsf{poly}(n,\sum m)2^{\sum m})$。

3.4 子任务 4：所有点均为白点；给定的图是外向树森林；所有程序的起始点均为外向树的根

森林中的树互不影响，只需对每一棵树分别计算。为了方便起见，记该树根为 r，记非根点 u 的父亲为 f_u。

对于每个点，一棵树中只有一条从根到该点的路径。以某个点为当前点的状态是唯一的。在该子任务中，我们直接使用点的编号表示暂停状态。

一次操作一定有跳过若干出边，最后选择一条出边进入或结束搜索这个形式。我们想要知道状态 u 在一次操作之后能到达哪些状态：

- （不从 DFS 当前点的过程中返回）$e_{u,1}, e_{u,2}, \ldots, e_{u,m_u}$
- （返回 1 层）$e_{f_u,j} (u \neq r,\ u = e_{f_u,i},\ j > i)$
- （返回 2 层）$e_{f_{f_u},j} (u \neq r,\ f_u = e_{f_{f_u},i},\ j > i)$
- ……
- （结束搜索）z

定义点 u 的左兄弟 $d_u^L = \{e_{f_u,j} | u = e_{f_u,i} \wedge j < i\}$，右兄弟 $d_u^R = \{e_{f_u,j} | u = e_{f_u,i} \wedge j > i\}$，后续 $h(u) = d_u^R \cup h_{f_u}$（特别地，$d_r^L = d_r^R = \varnothing, h(r) = \{z\}$）。一个点在一次操作之后可以到达的点集为 $t_u = h(u) \cup \{v | e_{u,i} = v\}$。

对于每个点 u 对应的状态计算其 SG 函数值 $\mathrm{sg}(u) = \mathrm{mex}(\{\mathrm{sg}(x) | x \in t_u\})$。

每个点只能转移到 DFS 序编号比它大的点，可以尝试按 DFS 序逆序计算每个点的 SG 函数值。

在这个过程中，我们需要维护 t_u：

- 对于 $h(u)$ 部分，容易发现使 $u \in h(v)$ 的 v 即 d_u^L 中各点子树之并；
- 对于直接出边部分，f_u 可以到达 u。

可以看出，使 $u \in h(v)$ 的 v 在 DFS 序上是连续的一段。因此，我们按逆 DFS 序遍历每个点 u 时用树状数组等数据结构维护多重集 $\{\mathrm{sg}(v) | v \in t_u\}$ 并计算 mex，即可在 $O(k + n \log n)$ 复杂度内通过该子任务。

3.5 子任务 5：所有点均为白点

我们尝试将子任务 4 中的一些概念推广到一般的有向无环图上。

我们用 (u,i) 表示点 u 的第 i 条出边，用 $(r_s; s_1, s_2, \ldots, s_{l(s)})$ 分别表示状态 s 的初始点和状态 s 按从浅至深顺序排列的 $l(s)$ 条出边。

令状态 $s = (r_s; s_1, s_2, \ldots, s_{l(s)-1}, s_{l(s)})$ 的当前点 $p(s)$ 为 $e_{u,i}$ $(s_{l(s)} = (u,i))$。为了方便，记 $tr(s,i) = (r_s; s_1, s_2, \ldots, s_{l(s)}, (p(s), i))$。

定义状态 s 的父状态 $f(s)$ 为 $(r_s; s_1, s_2, \ldots, s_{l(s)-1})$，右兄弟 $d^R(s)$ 为 $\{tr(f(s), j) | s_{l(s)} = (u,i) \wedge i < j\}$，后续 $h(s)$ 为 $d^R(s) \cup h(f(s))$。特别地，对于任意 $1 \le r \le n$，有 $d^R((r;)) = \varnothing$, $h((r;)) = \{z\}$, $p((r;)) = r$。

那么状态 s 的 SG 函数值 $\mathrm{sg}(s)$ 即为 $\mathrm{mex}(\{\mathrm{sg}(t) | t \in \{tr(s,i)\} \cup h(s)\})$。

由于状态数量很多，我们不能直接计算每个状态的 SG 函数值。我们希望找到像子任务 1 和 2 那样能够递推计算的信息。

我们用另外一种方式描述一个当前状态为 s 的程序：

- 从 $p(s)$ 开始，按题目描述中的方式进行搜索；
- 搜索结束时，当前玩家须将程序转移到 $h(s)$ 中的某个状态。

这说明一个状态 s 的 SG 函数值可以用 $g_{p(s)}(\{\mathrm{sg}(x) | x \in h(s)\})$ 表示。设 $u = p(s)$, $S = \{\mathrm{sg}(x) | x \in h(s)\}$，如果我们能找到合适的方法表示并递推计算 $g_u(S)$，就能解决该子任务。

注意到 $h(tr(s, m_u)) = h(s)$, $h(tr(s, m_u - 1)) = h(s) \cup \{tr(s, m_u)\}, \ldots,$

$h(tr(s,1)) = h(s) \cup \{tr(s,2), tr(s,3), \ldots, tr(s,m_u)\}$，我们有：

$$
\begin{aligned}
& g_u(S) \\
=& \mathrm{sg}(s) \\
=& \mathrm{mex}(\{\mathrm{sg}(t) | t \in \{tr(s,i) | 1 \le i \le m_u\} \cup h(s)\}) \\
=& \mathrm{mex}(S \cup \{\mathrm{sg}(tr(s,i)) | 1 \le i \le m_u\}) \\
=& \mathrm{mex}(S \cup \{\mathrm{sg}(tr(s,i)) | 1 \le i \le m_u - 1\} \cup \{\mathrm{sg}(tr(s,m_u))\}) \\
=& \mathrm{mex}(S \cup \{\mathrm{sg}(tr(s,i)) | 1 \le i \le m_u - 1\} \cup \{g_{e_{u,m_u}}(\{\mathrm{sg}(x) | x \in h(tr(s,m_u))\}\} \\
=& \mathrm{mex}(S \cup \{\mathrm{sg}(tr(s,i)) | 1 \le i \le m_u - 1\} \cup \{g_{e_{u,m_u}}(S)\} \\
=& \mathrm{mex}(S \cup \{\mathrm{sg}(tr(s,i)) | 1 \le i \le m_u - 2\} \cup \{\mathrm{sg}(tr(s,m_u-1)), g_{e_{u,m_u}}(S)\} \\
=& \mathrm{mex}(S \cup \{\mathrm{sg}(tr(s,i)) | 1 \le i \le m_u - 2\} \cup \{g_{e_{u,m_u-1}}(S \cup \{g_{e_{u,m_u}}(S)\}), g_{e_{u,m_u}}(S)\} \\
=& \cdots \\
=& \mathrm{mex}(d(u,1))
\end{aligned}
$$

其中

$$
\begin{aligned}
& d(u,i) \\
=& S \cup \{g_{e_{u,j}}(d(u,j)) | j \ge i\} \\
=& \begin{cases} d(u,i+1) \cup \{g_{e_{u,i}}(d(u,i+1))\}, & i \le m_u \\ S, & i > m_u \end{cases}
\end{aligned}
$$

注意到 $g_u(S)$ 的表达式展开后仅由 $\mathrm{mex}(S \cup \cdots)$ 一种运算组成。我们也可以从博弈的角度验证这一点：玩家可以在搜索 u 的过程中的任何一步内选择不断跳过，结束搜索 u 的过程，然后将程序转移到 $h(s)$ 中的某一个状态。

定义 $\mathrm{mex}_i(S)$ 为第 i 小的不属于 S 的自然数。按照定义，$\mathrm{mex} = \mathrm{mex}_1$。

容易证明

$$\mathrm{mex}_i(S \cup \{\mathrm{mex}_{a_1}(S), \mathrm{mex}_{a_2}(S), \ldots, \mathrm{mex}_{a_n}(S)\}) = \mathrm{mex}_{\mathrm{mex}_{i+1}(\{a_1,a_2,\ldots,a_n\})}(S)$$

特别地，

$$\begin{aligned}&\operatorname{mex}_i(\{0\}\cup\{\operatorname{mex}_{a_1}(\{0\}),\operatorname{mex}_{a_2}(\{0\}),\ldots,\operatorname{mex}_{a_n}(\{0\})\})\\=&\operatorname{mex}_{\operatorname{mex}_{i+1}(a_1,a_2,\ldots,a_n)}(\{0\})\end{aligned}$$

因此我们可以归纳证明 $g_u=\operatorname{mex}_{g_u(\{0\})}$。

递推计算所有 $g_u(\{0\})$。令 $d(u,m_u+1)=\{0\}$，倒序枚举 u 的所有出边并用树状数组维护 $d(u,i)$，在 $i\to i-1$ 时查询 $g_{e_{u,i-1}}(d(u,i))=\operatorname{mex}_{g_{e_{u,i-1}}(\{0\})}(d(u,i))$ 并将其插入 $d(u,i)$ 即可得到 $d(u,i-1)$。$\operatorname{mex}(d(u,1))$ 即为 $g_u(\{0\})$。

按照定义，

$$\begin{aligned}&\operatorname{sg}((r;))\\=&\,g_r(\{\operatorname{sg}(x)|x\in h((r;))\})\\=&\,g_r(\{\operatorname{sg}(z)\})\\=&\,g_r(\{0\})\end{aligned}$$

$\operatorname{sg}((r;))$ 即为从 r 开始搜索的程序的 SG 函数值。

总时间复杂度 $O(k+(n+m)\log m)$，可通过该子任务。

3.6 子任务 6：没有特殊限制

我们沿用子任务 5 中的方法。

对于 $f_{p(s)}$ 为黑点的状态 s，

$$h(s)=\begin{cases}\{tr(f(s),i+1)\}, & i<m_{f_{p(s)}}\\ h(f(s)), & i=m_{f_{p(s)}}\end{cases}$$

对于状态 s，设 $u=p(s),S=\{\operatorname{sg}(x)|x\in h(s)\}$，我们需要计算 $g_u(S)$。

u 为黑点　通过与子任务 5 中相似的方法可以得出

$$g_u(S) = \operatorname{mex}(\{g_{e_{u,1}}(\{g_{e_{u,2}}(\{\cdots g_{e_{u,m_u}}(S)\cdots\})\})\})$$

在子任务 2 中，我们已经计算出了每个点的类型。只要知道搜索 u 的过程结束后是否先手必胜，就能知道状态 s 是否先手必胜。

也就是说，我们只要知道 $0 \in S$ 是否为真，就能知道 $g_u(S) = 0$ 是否为真。

在 $m_u > 0$ 时，$g_u(S) \in \{0, 1\}$，u 的四种类型分别意味着：

1. $g_u(S) = 1$
2. $g_u(S) = 0$
3. $g_u(S) = [0 \in S]$
4. $g_u(S) = [0 \notin S]$

其中 $[P]$ 为艾佛森括号，当 P 为真时 $[P] = 1$，当 P 为假时 $[P] = 0$。

这样我们就对 $m_u > 0$ 的黑点 u 计算出了 g_u。没有出边的黑点与白点等价，不妨将其看作白点处理。

u 为白点　对于白点 u，我们不能像子任务 4 中一样直接用 $\operatorname{mex}_{g_u(\{0\})}$ 表示 g_u。

将 $g_u(S)$ 的表达式展开，其由 $\operatorname{mex}(S \cup \cdots)$ 和 $g_v(\cdots)$ $(c_v = 0, m_v > 0)$ 两种运算组成。我们可以归纳证明：对于任意 $i > j$，命题 $i \in S$ 是否为真不影响命题 $g_u(S) = j$ 是否为真。

因此，$g_u(S) \cap \{0, 1\}$ 和 $g_v(\cdots)$ 的值取决于 $0 \in S$ 和 $1 \in S$ 是否为真。

枚举 $0 \in S$ 和 $1 \in S$ 是否为真，对四种情况分别考虑 $g_u(S)$ 的值。如果 $g_u(S) \notin \{0, 1\}$，我们就可以用与子任务 5 中相似的方法证明 $g_u(S) = \operatorname{mex}_{g_u(S \cap \{0,1\})+1}(S \setminus \{0, 1\})$。

因此，我们只需知道 $g_u(T)$ $(T \subseteq \{0, 1\})$ 这四个值即可唯一确定 g_u。

计算 $g_u(T)$。与子任务 5 相似，令 $d(u, m_u + 1) = T$，倒序遍历 u 的出边并用树状数组维护 $d(u, i)$，在 $i \to i - 1$ 时计算 $g_{e_{u,i-1}}(d(u, i))$ 并将其插入 $d(u, i)$ 即可得到 $d(u, i - 1)$。$\operatorname{mex}(d(u, 1))$ 即为 $g_u(T)$。

综上所述，无论 u 是黑点还是白点，g_u 均有容易计算的表示方式。我们可以在 $O((n + m) \log m)$ 时间内递推计算所有 g_u，解决整个问题的时间复杂度为 $O(k + (n + m) \log m)$。

参考文献

[1] P. M. Grundy. Mathematics and games. *Eureka*, 2(6-8), 1939.

集训队队员周航锐参与了本题的验题工作。