NOI 2020 部分题目命题报告

李嘉图* 杨天祺†

1 命运 Destiny

1.1 题目描述

给定一棵有标号有根树 T 和一个返祖边的集合 E。更严谨地，$E \subseteq T \times T$，并且对于任何 $(u,v) \in E$，都有 $u \neq v$，且 u 在 T 中是 v 的祖先。现在要将 T 上的每一条边染为黑色或白色。一个染色方案是合法的，如果对于任何 $(u,v) \in E$，在 u 到 v 的路径上至少存在一条边被染为了白色。求合法的染色方案数对某个给定的数取模的结果，$|T| \leq 3 \times 10^5$。

1.2 $O(n^2)$ 解法

定义关于 u 的部分染色方案为一个以 u 为根的子树内的边尚未确定染色、而其他边均已确定染色的方案；称一个关于 u 的部分染色方案是合法的，如果所有完全在 u 子树外的返祖边对应的路径上都存在一个白色边。定义一个部分染色方案的补全是对子树内边的染色；称补全是合法的，如果补全所得到的完整染色方案是合法的。

对于一个关于 u 的部分染色方案 $\mathcal{C}_u$，记 $p(\mathcal{C}_u)$ 为最小的 k，使得 u 的第 k 个祖先和第 $k+1$ 个祖先之间的边被染成白色。

定理 1.1. *令 $\mathcal{C}_u, \mathcal{C}'_u$ 是两个关于结点 u 的合法部分染色方案，且 $p(\mathcal{C}_u) = p(\mathcal{C}'_u)$，那么任何一个对于 $\mathcal{C}_u$ 合法的补全对于 $\mathcal{C}'_u$ 也是合法的。更进一步，对于它们的合法补全方案数是相等的。*

证明. 第二部分是第一部分的直接推论，仅证第一部分。如果对于 $\mathcal{C}_u$ 的补全是合法的，为证这一补全对于 $\mathcal{C}'_u$ 也是合法的，仅需考察三种返祖边对应的限制都被满足。

1. 完全在 u 为根之外的返祖边对应的限制：由于部分染色方案 $\mathcal{C}'_u$ 是合法的，这类边对应的限制已经被满足。
2. 在 u 为根的子树之内的返祖边对应的限制：由于在 u 为根的子树内的返祖边对应的限制是否被满足仅与子树内的染色方案有关，既然其在 $\mathcal{C}_u$ 的补全中被满足，那么在 $\mathcal{C}'_u$ 的补全中也被满足。
3. 跨过 u 的返祖边对应的限制：如果这一返祖边在 $\mathcal{C}_u$ 中已经被满足，由于 $p(\mathcal{C}_u) = p(\mathcal{C}'_u)$，其在 $\mathcal{C}'_u$ 中也已经被满足，因而在 $\mathcal{C}'_u$ 补全中已经被满足；如果其在 $\mathcal{C}_u$ 中未被满足，既然其在 $\mathcal{C}_u$ 的补全中被满足，其在子树内的部分在补全中已经有一条边染为了白色，因而也在 $\mathcal{C}'_u$ 的补全中被满足。

□

*lijt19@mails.tsinghua.edu.cn
†yangtq19@mails.tsinghua.edu.cn

设 $dp[u][k]$ 表示关于结点 u 的、满足 $p(\mathcal{C}_u, u) = k$ 的部分染色方案 $\mathcal{C}_u$ 的合法补全方案数。根据以上定理，$dp[u][k]$ 是良定义的。如果存在一条 u 到其祖先且长度小于等于 k 的返祖边，那么显而易见 $dp[u][k] = 0$，因为这条返祖边对应的路径上没有任何白色边。否则我们容易说明

$$dp[u][k] = \prod_{v \in \mathrm{chl}(u)} (dp[v][k+1] + dp[v][0]),$$

其中 $\mathrm{chl}(u)$ 是结点 u 的儿子集合，对于叶节点自然有 $dp[u][k] = 1$。这是由于 u 到其每一个儿子的边的染色方案是独立的，且对于其儿子 $v \in \mathrm{chl}(u)$，若其被染成白色，那么方案数是 $dp[u][0]$，否则为 $dp[u][k+1]$。最终答案即为 $dp[r][0]$，其中 r 为根节点。朴素实现这一做法的时间复杂度为 $O(n^2)$，因为每一条树上的边对应了一次 $O(n)$ 的计算。

1.3 $O(n \log n)$ 解法

分析动态规划状态间的依赖关系。注意到 $dp[u][k]$ 的计算依赖 $dp[v][k+1]$ 和 $dp[v][0]$，其中 $dp[v][0]$ 同 k 无关，并且 $(k+1) - \mathrm{depth}(v) = k - \mathrm{depth}(u)$。设 $f[u][k] = dp[u][k + \mathrm{depth}(u)] + dp[u][0]$，其中 $-n \le k \le n$，整理动态转移方程为

$$\begin{aligned} f[u][k] &= \prod_{v \in \mathrm{chl}(u)} f[v][k] + dp[u][0] \\ &= \prod_{v \in \mathrm{chl}(u)} f[v][k] + \frac{1}{2} f[u][-\mathrm{depth}(u)]. \end{aligned}$$

考虑下面的算法。对于每一个结点 u，维护一个值域在 $[-n, n]$ 的动态开点线段树（支持区间加、区间赋值、单点查询），其中第 k 位记录 $f[u][k]$。如果 u 的所有孩子对应的线段树均已完成计算，执行下面的算法。

1. 将所有线段树合并，得到一棵新树，其中第 k 位

$$T_u(k) = \prod_{v \in \mathrm{chl}(u)} f[v][k].$$

2. 根据状态转移方程，$f[u][-\mathrm{depth}(u)] = 2T_u(k) = 2dp[u][0]$，设 $d = T_u(k)$；
3. 设 u 到其祖先的最短返祖边长度为 l，区间赋值 $T_u(l - \mathrm{depth}(u)), \ldots, T_u(n) = 0$；
4. 将整棵线段树加上 d。

而对于叶子，仅需用一次线段树区间赋值将 $T_u(l - \mathrm{depth}(u)), \ldots, T_u(n)$ 赋值为 1，其余部分赋值为 0 即可。我们可以在线段树合并的同时计算逐点乘积。注意到每合并一次所有线段树的结点个数减一，而每一次区间操作会增加 $O(\log n)$ 个结点，因此总的时间复杂度为 $O(n \log n)$。

注 1.1. 这一线段树仅起到维护标记的作用，标记就是数据本身。一种可能的实现是在线段树的每个结点维护 $x \mapsto ax + b$ 的标记，其中 a, b 是常数。区间赋值对应于标记 $x \mapsto 0 \cdot x + d$，区间加对应于标记 $x \mapsto 1 \cdot x + c$，初始时整个区间为 0。在线段树 r_1, r_2 合并时，如果其中一棵树仅有一个结点，不失一般性设 r_1 仅有一个结点，其上有标记 $x \mapsto ax + b$，那么 r_1 对应的整个区间中所有元素均为 $a \cdot 0 + b = b$，因此合并（即两棵线段树做点乘）可以在 r_2 上打一个区间乘标记。若两棵线段树均不对应叶子节点，则分别做标记下传后递归合并。

2 超现实树 Surreal

2.1 题目描述

为描述的简洁，下文中所述的树均为非空、无标号、有根二叉树；称两个树相等，如果他们在考虑儿子左右的情况下同构。仅有一个结点的树的高度为 1。

定义树 S 关于叶子 u 和另一棵树 T 的替换为将 S 中的 u 结点删去，并将 T 接入到原先 u 所在的位置，得到新的树 S'，记作 $S \xrightarrow{u,T} S'$。定义树 S 生长为树 S'，如果将其所有叶子任意独立地替换为新树，可以得到 S'，树 S 生长得到的集合称为其生长集，记作 $\mathrm{grow}(S)$。

设 $\mathcal{A}$ 是树的集合，定义其生长集 $\mathrm{grow}(\mathcal{A})$ 为其中所有树生长集的并。定义一个树的集合 $\mathcal{A}$ 是完备的，如果仅有有限棵树不在 $\mathrm{grow}(\mathcal{A})$ 中。

给定一个树的集合，判定其是否是完备的。数据满足集合中所有树的点数之和为 $m = 1 \times 10^6$，无额外限制。

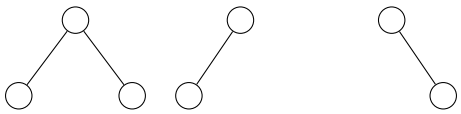

图 1: 一个完备而不包含所有树的集合。

2.2 $O(m \log m)$ 算法

定义一个树枝是由一条链加上（可能的）链上结点的儿子构成的树，除去所有叶子结点的部分称为树枝的主链，树枝的长为其主链包含的结点个数，记作 $|*|$。特别的，单个结点构成的树是长度为 0 的树枝。

引理 2.1. *任何一个树枝 C 仅能由长度小于等于 $|C|$ 的树枝生成；进一步，生成 C 的长度为 $|C|$ 的树枝仅有 C 本身。*

引理 2.2. *生成关系是传递的。*

证明. 两个引理的证明是直接的。 □

定理 2.1. *树的集合 $\mathcal{A}$ 是完备的，当且仅当其可以生成所有长度为 h 的树枝，其中 h 为 $\mathcal{A}$ 中树高的最大值。*

证明. 一方面，如果任何长度为 h 的树枝均可以被生成，我们将证明所有高度大于等于 $h+1$ 的树均可以被生成，从而仅有高度小于等于 h 的树可能无法被生成，因此 $\mathcal{A}$ 是完备的。对于任何高度大于等于 $h+1$ 的树 T，取其最长链上最高的 h 个结点作为主链，将主链之外的所有子树缩为一个结点，构成一个树枝 C。显然 C 可以生成原树，由于 C 可以被 $\mathcal{A}$ 生成，由引理 2.2 知原树可以被生成。

反过来，如果存在一个长度为 h 的树枝 C 无法被生成，取其主链末端结点的一个子结点 u。容易说明，将 u 替换为任意树所生成的树，均不能被 $\mathcal{A}$ 生成。这意味着存在无穷多不可被生成的树，从而 $\mathcal{A}$ 不是完备的。 □

由以上定理，原问题转化为判定是否存在一个长度为 $h+1$ 的树枝不可被生成，朴素实现可以得到一个 $O(4^h \mathrm{poly}(m))$ 的做法。更进一步，由引理 2.1 我们可以删去原集合中所有不是树枝的树。

现在考虑 $\mathcal{A}$ 仅包含树枝的情景。注意到一个长度为 h 的树枝可以用二进制对 (c,l) 表示，其中 $|c|=h-1$ 记录主链的形态（即从第二个结点开始，每个结点的左右儿子属性），$|l|=h+1$ 从上到下记录主链上叶子的存在性（特别的，在主链末端，可以先记左儿子再记右儿子并置于串末尾）。c 和 l 分别称为主链编码和叶子编码。为方便起见，用 0 表示不存在/左，用 1 表示存在/右。

定理 2.2. *设树枝 C 的表示为 (c,l)，其中 $c=c_1c_2\dots c_{h-1}$，$l=l_1l_2\dots l_{h-1}xy$。长度为 $k<h$ 的树枝 (c',l') 生成 C，当且仅当以下两个条件均满足。*

1. *$c'=c_1c_2\dots c_k$；*
2. *$l'=l_1l_2\dots l_{k-1}x'y'$。如果 C 上第 $k+1$ 个结点是第 k 个结点的左孩子，那么 $x'=1,y'=l_k$；反之，$x'=l_k,y'=1$。*

证明. 条件的充分性是显然的，仅证必要性。如果长度小于 h 的树枝 $C'=(c',l')$ 生成 C，而其主链不是 C 主链的一个前缀，那么 C' 主链末端及末端连接的一个叶子超出了 C，这与 C' 生成 C 矛盾，因此第一个条件必然满足。对于第二个条件的证明是类似的。 □

对于给定的树枝集合 $\mathcal{A}=\{(c^1,l^1),(c^2,l^2),\dots,(c^n,c^l)\}$，考虑建立 $\{c^1,\dots,c^n\}$ 的字典树 T_1 和 $\{l^1,\dots,l^n\}$ 的字典树 T_2。现在判定是否存在一个长度为 h 的树枝 (c,l) 无法被生成。注意到 c 和 l 仅与在字典树 T_1 和 T_2 中的部分（即主链长度比 T_1 中最深主链长 1）有关，仅需枚举字典树中的部分即可。

考虑下面的算法：在 T_1 上 DFS，并在 T_2 上维护那些“可被生成的”叶子编码。更严谨的，当 DFS 到结点 u、进入一个子树 v 时，在 T_2 中更新主链末端为 u 的树枝可以覆盖到的位置。很明显，对于一个树枝，其在 T_2 中可以覆盖到的位置构成一个子树，因此可以用线段树维护。当 DFS 退出一个结点时，可以用撤销法（或一次数据结构更新）取消对应树枝的贡献。当 DFS 到达 T_1 外侧时，在 T_2 的数据结构中查询是否有一个未被覆盖的位置，这显然可以用线段树最小值来实现。

注意到任何一个树枝都只会对应两次数据结构更新，总的更新开销不会超过 $O(|\mathcal{A}|\log|T_2|)=O(m\log m)$；每次 DFS 到 T_1 外侧会产生一次数据结构查询，总的查询开销不会超过 $O(|T_1|\log|T_2|)=O(m\log m)$。综上所述，总的时间开销是 $O(m\log m)$。

2.3 $O(m)$ 算法

定理 2.3. *令 $C=(c^i,l^i)$ 为一个树枝，其中 $c=c_1c_2\dots c_k$，$l=l_1l_2\dots l_{k-1}xy$，那么*

1. *如果 $x=1$ 且 $y=0$，那么所有主链长度大于 k 的可被其生成的树枝 (c',l') 一定满足 c' 以 $c_1c_2\dots c_k0$ 为前缀，且 l' 以 $l_1l_2\dots l_{k-1}0$ 为前缀；*
2. *如果 $x=0$ 且 $y=1$，那么所有主链长度大于 k 的可被生成的树枝 (c',l') 一定满足 c' 以 $c_1c_2\dots c_k1$ 为前缀，且 l' 以 $l_1l_2\dots l_{k-1}0$ 为前缀；*
3. *如果 $x=1$ 且 $y=1$，那么所有主链长度大于 k 的可被生成的树枝 (c',l') 一定满足一下两者之一：要么 c' 以 $c_1c_2\dots c_k0$ 为前缀，且 l' 以 $l_1l_2\dots l_{k-1}1$ 为前缀；要么 c' 以 $c_1c_2\dots c_k1$ 为前缀，且 l' 以 $l_1l_2\dots l_{k-1}0$ 为前缀。*

证明. 证明是容易的。 □

根据上面的定理，由于长度小于等于常数的可被生成树枝在完备性的意义下是无用的，我们容易将原问题转化为下面的问题：给定两棵 0/1-Trie T_1 和 T_2，以及若干对结点

$(u_i, v_i) \in T_1 \times T_2$。为 T_1 和 T_2 的所有结点补齐左右孩子，判定是否存在一对结点 (u, v)，满足

1. u 是叶子结点，且 $\operatorname{depth} v = \operatorname{depth} u + 1$；
2. 对于任何的 (c^i, l^l)，要么 c 不在 c^i 为根的子树中，要么 l 不在 l^i 为根的子树中。

换言之，判定是否所有的点对都被 $\{(c^i, l^i)\}$ 所覆盖。更进一步，这一转化后的问题还具有如下的性质：$\operatorname{depth} c^i = \operatorname{depth} l^i + 1$。考虑下面的算法。

1. 令 $T[i, j]$ 为哈希表，其中 $1 \le i \le |T_1|, 1 \le j \le |T_2|$；
2. 将所有给定的结点对按深度从深到浅分组装入桶中，并设 $T[u_i, v_i] = 1$；
3. 按照从深到浅的顺序遍历所有点对 (u, v)，设 u', v' 分别为 u, v 的兄弟结点，p, q 分别为 u, v 的父节点，使用哈希表查询：如果 $(u, v), (u, v'), (u', v), (u', v')$ 都在给定的结点对中，那么将 (p, q) 加入结点对（深度为 $\operatorname{depth} u - 1$ 的桶中，并设 $T[p, q] = 1$）。
4. 设 r_1, r_2 分别为 T_1 和 T_2 的根节点，如果算法结束后 (r_1, r_2) 存在，那么原树的集合是完备的。

算法的证明可以考虑对树高施归纳。若算法终止且 (r_1, r_2) 存在显然两棵树被覆盖；反过来，如果算法终止且 (r_1, r_2) 不存在，那么一定存在一个 r_1 的孩子 r_1' 和一个 r_2 的孩子 r_2' 满足 (r_1', r_2') 不存在，因此可以递归地构造一个无法被覆盖的结点对。注意到每个四个结点对消失会增加一个新的结点对，因此算法的时间复杂度是 $O(m)$ 的。

2.4 另一种线性算法

引理 2.3. *总点数为 m 的树枝集合最多只有 $4m/\log m + \sqrt{m}$ 条不同的树枝。*

证明. 注意到深度小于等于 h 的树枝仅有 $2^{2(h-1)}$ 种；深度大于 h 的树枝至少有 h 个结点，因而至多有 m/h 种。取 $h = \log m/4$，那么总的树枝数不会超过

$$2^{1/2 \cdot \log m} + \frac{m}{1/4 \cdot \log m} = 4m/\log m + \sqrt{m}.$$

□

由于可以用数组在线性时间内将深度小于等于 $\log m/4$ 的树枝去重，在去重之后，即使使用平衡树代替哈希表也仍然可以获得线性时间算法。此外，$O(m \log m)$ 的线段树算法也可以通过构建虚树得到线性时间算法。

2.5 有用树枝深度的界

定理 2.4. *定义一个完备树枝集合中的树枝为有用的，如果删去这个树枝得到的树枝集是不完备的。设所有树枝的结点个数之和为 m，那么*

1. *有用树枝深度的上界是 $O(\sqrt{m})$；*
2. *存在一个树枝集和其中一个深度为 $\Omega(\sqrt{m})$ 的有用树枝。*

证明. 对于上界，令 h 为有用树枝 C 的深度，考虑深度为 $1, 2, \ldots, h-1$ 的树枝 $C_1, C_2, \ldots, C_{h-1}$，满足 C_i 和 C 的 LCA 深度为 i，且 C_i 的主链长度充分长。根据 C 的有用性，可以将它们分别对应到主链长度至少为 $1, 2, \ldots, h-1$ 的、存在于给定树枝集合中的不同树枝。这样来看，$1 + 2 + \cdots + (h-1) \le m$，因此 $h = O(\sqrt{m})$。

对于下界，考虑图 2 给出的例子，换言之，所有主链为一个向右侧偏、上端没有叶子且末端要么有两个叶子、要么仅有左侧的叶子的树枝。所有主链长度为 h 的这样的树枝加上一个长度为 $h+1$ 的向右偏的主链构成了一个完备集合，且长度为 $h+1$ 的主链构成了深度为 $\Omega(\sqrt{m})$ 的有用树枝。 □

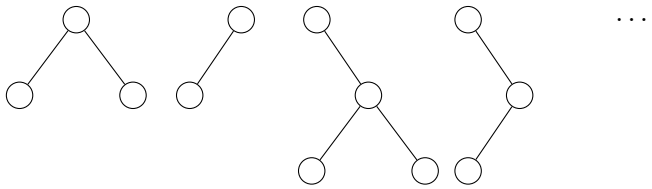

图 2: 深度的下界。

这一定理说明，任何深度超过 $O(\sqrt{m})$ 的树枝都不可能是有用的。在构造数据时，不应大量使用深度超过 $O(\sqrt{m})$ 的树枝。