Speike & Tom 解题报告

华东师范大学第二附属中学 万成章

October 2021

1 题目大意

1.1 题目描述

一次 Tom 用鞭炮炸 Jerry 的老鼠洞时，不小心炸到了 Speike 的狗窝。

后院的道路构成了一棵 n 个点的**无向树**，Speike 与 Tom 之间的追逐以如下方式展开：

- Tom 和 Speike 一开始分别在 a, b 两个点；
- Tom 和 Speike 轮流行动，Tom 先行；
- 每次行动者可以选择不动，或是沿着一条边走到另一个端点；
- 如果一次行动后到达了 Speike 和 Tom 处于同一位置则 Speike 胜。

以 Tom 的智商足够知道这个游戏他是必败的，所以他趁 Speike 没反应过来之前建立了 m 条**额外边**，这些额外边**只能被 Tom 经过，而不能被 Speike 经过**。

现在 Tom 想要知道，对于所有的 $n \times (n-1)$ 组可能的 (a, b) $(a \neq b)$，有多少组追逐中 Tom 有策略使得 Speike 永远无法获胜。

1.2 数据范围

对于 100% 的数据：$1 \leq n, m \leq 10^5$，$1 \leq x, y \leq n$，保证 $x \neq y$ 且所有无序对 (x, y) 不同，但不保证 (x, y) 这条边不在树中。

Subtask 1（10 pts）：$n, m \leq 20$。

Subtask 2（15 pts）：$n, m \leq 300$。

Subtask 3（15 pts）：$n, m \leq 2000$。

Subtask 4（20 pts）：$n, m \leq 40000$。
Subtask 5（10 pts）：$m = 1$。
Subtask 6（30 pts）：无特殊限制。

1.3 题目来源

本题是一道原创题。

2 解题过程

2.1 搜索

可以采用搜索或其他暴力算法得到一定的分数，例如可以将 $O(n^2)$ 个状态全部列出后从最终状态倒推每个状态的胜负情况，这里不加赘述。

2.2 性质观察

称 Tom 获胜，当且仅当 Speike 永远无法获胜。

对于额外边 (x, y)，如果 x, y 的树上距离为 d，我们就说这条额外边是一条 d 级边。

- **观察 1**：只要 Tom 所在点是一条大于 2 级的边的端点，那么 Tom 获胜。
- 证明：Tom 可以始终位于 x, y 两点之一，若 Speike 行动后和 x 相邻则 Tom 到 y，否则 Tom 到 x，由于树上不存在 x, y 的公共邻居，所以 Tom 始终不会被 Speike 追到。

于是，我们可以将额外边进行这样的处理：将所有大于 2 级边的端点标记，然后删除这些边，同时也可以删除无用的 1 级边，只保留 2 级边。那么上面的观察说明了 Tom 到达标记点后就可以获胜，而我们还可以说明这一条件必要的。

- **观察 2**：不存在一种情况，使得追逐无限进行，且 Tom 任意时刻都不在标记点上。
- 证明：以 Speike 初始点 b 为树根，令 Speike 的策略为始终向 Tom 走近一步，可以归纳证明 Tom 永远在 Speike 所在点子树内，这是因为

如果某一步 Tom 走到 Speike 所在点 b' 子树外，那么由于不经过大于 2 级边，所以肯定是从原来 b' 的一个儿子走到 b' 的父亲，那么 Speike 一步后就可以追到 Tom，追逐就没有无限进行。

也就是 Tom 获胜当且仅当他能在不被追到的情况下走到一个标记点。通过上述证明过程，可知以 b 为根时 Tom 在到达标记点前始终位于 Speike 所在点子树中。

设 i,j 的树上距离为 $d_1(i,j)$，而考虑额外边时的最短路为 $d_2(i,j)$，则 Tom 可以在不被追到的情况下通过路径 $a = v_1 \to v_2 \to \ldots \to v_k$，当且仅当对于任意 v_i 有 $d_2(a,v_i) < d_1(b,v_i)$。

2.2.1 由此得到的一个算法

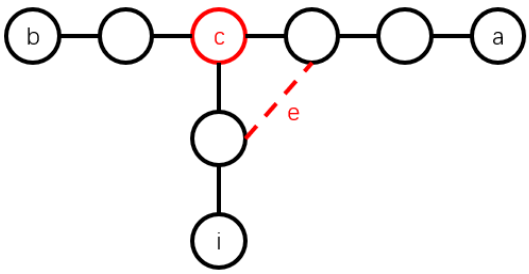
根据观察 1, 2 以及相关分析，我们立刻得到一个 $O(n+m)$ 的判断给定的一组 a,b 的胜负情况的算法：

首先求出所有标记点，然后对于一次询问 a,b，取所有满足 $d_2(a,i) < d_1(b,i)$ 的 i 构成的点导出子图（包含树边和额外边），求 a 所在的连通块是否有标记点，如果有则 Tom 获胜，否则 Speike 获胜。

对于每一对 (a,b) 进行上述判定，即可在 $O(n^2(n+m))$ 的复杂度内完成本题，该算法也是作者在对拍时使用的暴力算法。

2.3 结构分析

以 b 为树根，考虑 2.2.1 所述算法中的导出子图里 a 所在的连通块在树上有怎样的结构。



如图，我们要讨论 a 能否到达 i，首先可以找到 a,b,i 三个点的交汇点（到三个点距离和最小的点），即图中红点 c，那么 a 到 i 的路径要么需要经过 c，要么需要经过一条恰好跨过 c 的 2 级边（即图中边 e）。

由于 Tom 可以走树边，所以一旦经过 c 点后 Tom 的行动就不会比

Speike 慢了，所以整个过程的“瓶颈”实际上就在 c 这里，我们只需要讨论 Tom 是否可以在不被 Speike 追上的情况下走到 c（或者 e 的端点），而这即是比较 $d_2(a,c)$ 和 $d_1(b,c)$ 的大小。

注意到这里的单调性：可选的 c 是 $a \to b$ 链上靠近 a 的一个前缀，那么所有可能到达的 i 都在以 b 为根时 c 的子树里，我们可以首先通过树上倍增定位出离 a 最远的 c，然后询问这个子树里是否有标记点。

注意有一些细节是上面没有讨论的，例如边 e 的问题，所有关于 2 级边的处理细节将到正解部分再提及。

2.3.1 由此得到的一个算法

根据上述结构，我们可以 $O(\log n)$ 回答单次询问，需要树上倍增，如果要完成所有 $O(n^2)$ 次查询，则时间复杂度为 $O(n^2 \log n + m)$。

2.4 算法优化

如果要对 $n \times (n-1)$ 对 (a,b) 分别判定，上述算法已经接近极限，我们需要一个更好的方法来计数点对。而点对计数的有力工具是树分治，下面介绍一种通过点分治结合 2.3 所述结构的算法。

以重心 p 为根，只考虑 a,b 属于 p 不同子树的情况。

设 $a \to b$ 路径上依次经过 $a = v_1, v_2, \ldots, v_k = b$ 这些点，首先找出一个最小的 q 使得删去 v_{q+1} 后 v_q 所在连通块（下面将这个连通块记为 $C(v_{q+1}, v_q)$）有至少一个标记点，那么 Tom 至少需要经过 v_q 才能到达一个标记点，根据 2.3 中的结构，只需要比较 $d_2(a,v_q)$ 和 $d_1(b,v_q)$ 的大小即可。

接下来我们分成两种情况讨论：

- v_q 在 $a \to p$ 路径上（包括点 p）。

  这时可以发现 v_q 是与 b 无关的，枚举 a 后我们可以直接找出其祖先中最近的一个满足子树中有标记点的点，那就是 v_q。有了 v_q 也就有了 $d = d_2(a,v_q)$，那么我们只需要对 $d_1(b,v_q) > d$ 的 b 进行计数就可以了，开个桶记录当前分治到的连通块中所有点到重心 p 的距离即可。

- v_q 在 $p \to b$ 路径上（不包括点 p）。

  这时 v_q 与 a 是无关的，我们同样可以枚举 b 后算出 v_q，然后算出 $d = d_1(b,v_q)$，然后用桶计数满足 $d_2(a,v_q) < d$ 的 a 的数量。

算法的主要过程实际上已经叙述完了，但由于 2 级边的存在，使得其中有不少繁琐的细节，下面来对此进行一些解释。

2.4.1 算法细节

设当前分治重心为 p，首先判定是否所有标记点都在 p 的同一个子树中，如果是的话，将这个子树的根记为 mk，对它的处理比较特殊，我们首先只计算 b 不在 mk 子树中的情况。

对于第一种情况（v_q 在 $a \to p$ 上），首先对整个连通块 DFS 一次，将所有点 i 的 $d_1(i,p)$ 加到一个桶中，然后分别对每个 p 的子树（除了 mk）DFS，首先将这个子树中点的 $d_1(i,p)$ 从桶中删除（因为统计的是不同子树间的点对），然后枚举其中的点作为 a，找到 v_q，然后求出可行的 b 到 p 的最小距离，计算桶中数据的一个后缀和累加到答案。

对于这个最小距离需要分类讨论：

- 如果不存在 v_{q-1} 到另一个 v_q 的儿子的 2 级边，那么可以直接用 $d_2(a,v_q)$ 进行计算。
- 如果存在 v_{q-1} 到另一个 v_q 的儿子 x 的 2 级边，且 x 子树中有标记点，那么可以不经过 v_q，而是通过 $v_{q-1} \to x$ 走捷径，应该利用这个长度进行计算。
- 特别地，在上种情况中，如果 v_q 就是 p，并且 x 是唯一的，那么对于 x 子树中的 b 就无法使用 $v_{q-1} \to x$ 这条捷径，std 的实现方法是用 map 记录这样多算的部分，最后 DFS 一次减掉。

如果 mk 存在，那么还需要考虑第二种情况（v_q 在 $p \to b$ 上），显然只有 mk 子树中的点能成为这样的 b，通过 DFS 枚举 b，计算出 v_q，同样讨论是否存在 v_{q-1} 到 v_q 儿子的 2 级边，然后算出 $d_2(a,p)$ 的最大值，用桶的前缀和累加进答案。

2.4.2 复杂度

由于点分治中还需要进行一些较为复杂的计算，考虑到原本本题的细节就较多，代码并不好写，所以对于其中的一些计算采用了多 1 个 log 的代价进行维护，因此 std 的时间复杂度为 $O((n+m)\log^2 n)$，其中第二个 log 来自于树状数组，树上倍增，map。

如果精细实现，理论上应该可以做到 $O((n+m)\log n)$，但代码难度将进一步上升，因此本题并没有强求这种做法。

3 总结与出题背景

本题延续了作者去年集训队作业自选题 Tom & Jerry 的追逐模型，并且结合树上数据结构的常见思维方法，考察了选手的图论和树上问题的综合能力。

本题的思维要求其实并不高，但需要一环接一环的推导，而在代码实现上比较考察选手的细节处理能力和调试能力，并没有什么大段的模板可以仿照，总体来说是一道思维过程轻量级，而代码实现有一些难度的题。

这题的最初雏形与 AGC 005 E 比较相像，在发现这一点后我将问题修改为在树的基础上加上 m 条额外边的形式，同时将查询内容升级为了点对计数，这样一来就比那道 AGC 题多了很多不同的思维过程。

原本的标准解法除了点分治外还用到了长链剖分，但在后来的讨论中发现事实上只需要用到点分治即可解决。

4 参考资料

[1] [AGC 005 E] Sugigma: The Showndown
(https://atcoder.jp/contests/agc005/tasks/agc005_e)
[2] 与左骏驰同学、王又嘉同学、管晏如同学的讨论。