# 《追逐》解题报告

天津英华实验学校 朱鹏睿

November 15, 2025

## Contents

# 1 题目

## 1.1 题面描述

给定一棵 $n$ 个点的树，接下来会有 $m$ 个情景：

一个情景中有 $k$ 个人，这些人一开始分别在 $x_1, x_2, \cdots, x_k$，保证 $x_i$ 互不相同。

第 $i$ 个人每个时刻会以 1 的速度向第 $(i \bmod k) + 1$ 个人的位置移动，注意运动是**连续**的。

这个情景会有 $q$ 次询问，每次询问给定 $p, t$，求第 $p$ 个人 $t$ 时刻之后的位置（可能在边上，要求输出在哪个点上或者在哪条边上）。

## 1.2 数据范围

$2 \le n \le 2 \times 10^5$，$2 \le k \le n, 1 \le q \le 2 \times 10^5$，$\sum k, \sum q \le 4 \times 10^5$

4s，1GB。

subtask 1（13 pts）：$n \le 3000, \sum k \le 6000$

subtask 2（8 pts）：树的形态随机。

subtask 3（14 pts）：保证树是一条链。

subtask 4（8 pts）：$k = 3$

subtask 5（8 pts）：$k = 4$

subtask 6（17 pts）：$k \le 30$

subtask 7（16 pts）：$n \le 10^5, \sum n \le 2 \times 10^5$

subtask 8（16 pts）：无特殊限制。

## 2 做法

这是赛时通过的选手的做法。

观察一件事情，所有人会停在直径中点，这是因为所有叶子上的人移动方向一定是向内，因此每一时刻直径大小都会 $-2$。

因此，直径端点的方向我们可以确定，它们一定是一直向直径中点运动的，设其分别为 $x, y$。

同时，每一个点 $i$ 的轨迹一定是从初始位置，一直向某一个点的方向运动，最后在这个位置上遇到 $i+1$，之后一直跟随 $i+1$ 运动，因此所有点的运动轨迹可以用 $O(k)$ 的信息表示。

那么考虑如果已经维护出了 $i$ 的路径，如何求出 $i-1$ 的路径。

$i$ 的路径会并入之前的一些点，我们用一个单调栈维护出每一段路径是向哪个点走，此时考虑 $i-1$ 时，我们先让它一直向 $i$ 走，如果它能够在第一段就追上 $i$，那么直接加入即可，否则将栈内最后一段路径弹出，然后继续和倒数第二段比较，可以发现这样我们就能求出 $i-1$ 和 $i$ 相遇的位置了。

如果使用 $O(1)$ LCA 可以做到 $O(k+q\log k)$ 的复杂度。

## 3 做法 - archived

这是出这道题的时候原来的做法，赛时选手发现了更为简单优美的做法。将原做法放在这里，仅供作参考。

不难发现所有人最后一定会停在一个点不动，可以进一步发现这个点在直径中心，但是这可能没有什么用处。

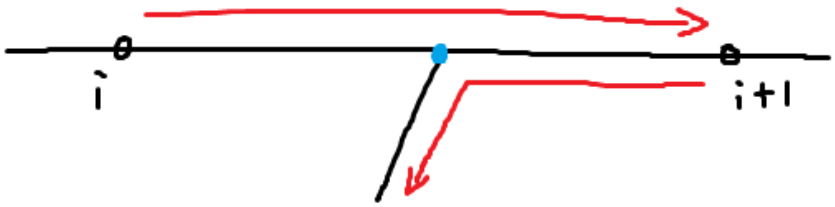
为了实现方便，可以在树上每条边再加入一个节点，这样所有事件都发生在整点处。

直接暴力模拟可以得到 $O(nk)$ 的复杂度。

考虑优化这件事情，首先可以观察到每个人 $i$ 一定是若干时间之后追上 $i+1$，并且在这之后一直跟着 $i+1$ 走；并且在追上 $i+1$ 之前，走的路径一定是一条链，也就是说所有人的轨迹其实是可以用 $O(k)$ 的信息描述的。

先从链上的情况入手：我们保留所有 $i$ 和 $i+1$ 相向而行的事件，每次取出最早的一次相遇事件 $u, u+1$，然后将它们合并在一起，并对 $u$ 和 $u-1$ 的相遇事件进行更新，即可合理地模拟出所有人的轨迹。

那么考虑扩展到树上，扩展到树上会产生一些问题：首要的就是我们无法定义“相向而行”。扩展原来方向的定义，我们给每一个点 $u$ 赋一个目标点 $dir(u)$，表示我们认为 $u$ 会一直向着 $dir(u)$ 移动。一开始我们将 $dir(u)$ 设为 $u+1$ 的初始位置，过程中我们可能会对这个 $dir(u)$ 进行更新。




我们找到 $i+1$ 到 $dir(i+1)$ 第一次离开 $pos_i - pos_{i+1}$ 路径的位置，也即图中的蓝点。如果 $i$ 比 $i+1$ 先到达蓝点，那么它们就会相遇；否则 $i+1$ 就会先离开，然后 $i$ 会在可预见的未来内追不上 $i+1$ 了。我们称这种状态为 $i$ **倒闭**了。

因此我们加入两个事件：一个事件是 $i$ 在 $dis/2$ 时刻之后，在 $pos_i$ 和 $pos_{i+1}$ 的中点处和 $i+1$ 相遇；第二个事件是 $i+1$ 抵达蓝点。这两个事件只有可能发生一个，我们在事件堆中先取出哪个，就把另一个删掉。

最后人的倒闭结构会变为：有若干个段，每一段的头是没有倒闭的，后面跟着一堆倒闭的人；我们可以认为这些倒闭的人的 $dir$ 即为段头的人的 $dir$。

可以用链表维护还没有追上的所有人，那么我们事件堆中将会有以下两种事件：

- $u$ 追上 $nxt(u)$：取消 $u$ 的倒闭事件，在链表上把 $u$ 删除；更新 $prv(u)$ 的 $dir$，加入和 $nxt(u)$ 的两种事件。
- $u$ 倒闭了：更新 $u$ 的 $dir$，取消 $u$ 追上 $nxt(u)$ 的事件，将 $u$ 的段接入 $nxt(u)$ 的段内；此时整个倒闭段的 $dir$ 都会被更新，但是影响的事件只有找到倒闭段尾部 $v$ 之后 $prv(v)$ 和 $v$ 的事件，更新即可。

为了询问方便，可以把询问也实现成一种事件。

最后实现需要：可删堆维护所有事件，环形链表维护所有人的追及关系，并查集维护每个人现在追上了在跟着谁走，以及另一个数据结构维护所有倒闭段（笔者使用了并查集）。

分析复杂度可以数一下上述两种事件分别对堆大小的影响，追上的事件会让堆大小 $+1$，这样的事件最多会发生 $k$ 次，而倒闭的时间会让堆大小 $-1$，因此总事件数是线性的，总时间复杂度为 $O((k+q)\log(k+q))$。