直径、重心、树剖、虚树（提高 + 班）

刘洋

2026

Table of Contents

1. 树的直径
2. 树的重心
3. 树链剖分
4. 虚树

Table of Contents

1. 树的直径
2. 树的重心
3. 树链剖分
4. 虚树

P4408 [NOI2003] 逃学的小孩

- 给定带边权的树，三点 A, B, C 未知。父母从 C 出发，先去离 C 近的那个，再去另一个；求最坏情况下走过的总路程的最大值。即最大化 $\mathrm{dis}(C, X) + \mathrm{dis}(X, Y)$，其中 $\mathrm{dis}(C, X) \leq \mathrm{dis}(C, Y)$。
- 数据范围：$3 \leq N \leq 2 \times 10^5$，边权 $0 \leq T_i \leq 10^9$。

Solution

Intuition: 最优解一定与直径相关。

- 设直径端点为 u, v。答案中较长的那段 $\mathrm{dis}(X, Y)$ 取直径 $\mathrm{dis}(u, v)$ 时最优。

Solution

Intuition: 最优解一定与直径相关。

- 设直径端点为 u, v。答案中较长的那段 $\mathrm{dis}(X, Y)$ 取直径 $\mathrm{dis}(u, v)$ 时最优。
- 第三点 C 要让较短段 $\min(\mathrm{dis}(C, u), \mathrm{dis}(C, v))$ 最大，枚举每个点 C，答案为 $\mathrm{dis}(u, v) + \max_C \min(\mathrm{dis}(C, u), \mathrm{dis}(C, v))$。

Solution

Intuition: 最优解一定与直径相关。

- 设直径端点为 u, v。答案中较长的那段 $\mathrm{dis}(X, Y)$ 取直径 $\mathrm{dis}(u, v)$ 时最优。
- 第三点 C 要让较短段 $\min(\mathrm{dis}(C, u), \mathrm{dis}(C, v))$ 最大，枚举每个点 C，答案为 $\mathrm{dis}(u, v) + \max_C \min(\mathrm{dis}(C, u), \mathrm{dis}(C, v))$。
- 两次 BFS/DFS 求直径端点，再各 DFS 一次求所有点到 u, v 的距离。复杂度 $O(N)$。

CF1783G Weighed Tree Radius

- 无权树 n 点，点权 a_i。带权距离 $w_v(u) = d(v,u) + a_u$（只加终点权）；偏心率 $e(v) = \max_u w_v(u)$，半径 $r = \min_v e(v)$。
- m 次修改 $a_v \leftarrow x$，每次修改后输出当前半径 r。
- 数据范围：$n \le 2 \times 10^5$，$m \le 10^5$，$0 \le a_i \le 10^6$。

Solution

Intuition: 半径由带权直径决定，$r = \lceil D/2 \rceil$；动态维护带权直径 = 线段树维护「点集直径端点」，合并靠点集直径合并引理，距离用欧拉序 $O(1)$ LCA。

- 带权直径 $D = \max_{u,v} \left(a_u + d(u,v) + a_v\right)$。性质：对任意 x，$e(x) = \max(d(x,A) + a_A,\ d(x,B) + a_B)$（$A, B$ 为直径端点），中心取中点得 $r = \lceil D/2 \rceil$。

Solution

Intuition: 半径由带权直径决定，$r = \lceil D/2 \rceil$；动态维护带权直径 = 线段树维护「点集直径端点」，合并靠点集直径合并引理，距离用欧拉序 $O(1)$ LCA。

- 带权直径 $D = \max_{u,v} \left(a_u + d(u,v) + a_v\right)$。性质：对任意 x，$e(x) = \max(d(x,A) + a_A,\ d(x,B) + a_B)$（$A, B$ 为直径端点），中心取中点得 $r = \lceil D/2 \rceil$。
- **点集直径合并引理：**两个点集 S, T 合并后，新直径的两端点一定落在 S 的直径端点与 T 的直径端点这 4 个点之中。故合并只需在这 4 点里枚举 $\binom{4}{2} = 6$ 对，取带权距离最大者。

Solution

Intuition: 半径由带权直径决定，$r = \lceil D/2 \rceil$；动态维护带权直径 = 线段树维护「点集直径端点」，合并靠点集直径合并引理，距离用欧拉序 $O(1)$ LCA。

- 带权直径 $D = \max_{u,v} \left(a_u + d(u,v) + a_v\right)$。性质：对任意 x，$e(x) = \max(d(x,A) + a_A,\ d(x,B) + a_B)$（$A, B$ 为直径端点），中心取中点得 $r = \lceil D/2 \rceil$。
- **点集直径合并引理：**两个点集 S, T 合并后，新直径的两端点一定落在 S 的直径端点与 T 的直径端点这 4 个点之中。故合并只需在这 4 点里枚举 $\binom{4}{2} = 6$ 对，取带权距离最大者。
- 线段树按顶点顺序分段，每个结点维护其对应点集的**直径端点对**；合并左右孩子即用上引理（6 候选）。每对候选要算 $d(u,v) = dep_u + dep_v - 2dep_{\mathrm{lca}}$，其中 LCA 由**欧拉序 + ST 表** $O(1)$ **求出**，故单次合并 $O(1)$。

Solution

Intuition: 半径由带权直径决定，$r = \lceil D/2 \rceil$；动态维护带权直径 = 线段树维护「点集直径端点」，合并靠点集直径合并引理，距离用欧拉序 $O(1)$ LCA。

- 带权直径 $D = \max_{u,v} \left(a_u + d(u,v) + a_v\right)$。性质：对任意 x，$e(x) = \max(d(x,A) + a_A,\ d(x,B) + a_B)$（$A, B$ 为直径端点），中心取中点得 $r = \lceil D/2 \rceil$。
- **点集直径合并引理：**两个点集 S, T 合并后，新直径的两端点一定落在 S 的直径端点与 T 的直径端点这 4 个点之中。故合并只需在这 4 点里枚举 $\binom{4}{2} = 6$ 对，取带权距离最大者。
- 线段树按顶点顺序分段，每个结点维护其对应点集的**直径端点对**；合并左右孩子即用上引理（6 候选）。每对候选要算 $d(u,v) = dep_u + dep_v - 2dep_{\mathrm{lca}}$，其中 LCA 由**欧拉序 + ST 表** $O(1)$ **求出**，故单次合并 $O(1)$。
- 修改点权 = 线段树单点更新，自底向上 $O(\log n)$ 次合并；根结点直径即当前 D，$r = \lceil D/2 \rceil$。总复杂度 $O((n+m)\log n)$。

Table of Contents

1. 树的直径
2. 树的重心
3. 树链剖分
4. 虚树

ABC291Ex Balanced Tree

- 给定 N 个点的树 T，构造一棵有根树 R：任意 x, y 在 R 中的 LCA z 必须落在 T 中 x 到 y 的简单路径上；且 R 中每个非根点 v 满足 $2 \cdot \mathrm{size}(v) \leq \mathrm{size}(\mathrm{parent}(v))$。输出每个点的父亲 p_i（根为 -1）。
- 数据范围：$1 \leq N \leq 10^5$。

Solution

Intuition: 这正是点分治的分治结构（点分树）。

- 对 T 做点分治：每层取当前连通块的重心作为 R 中该块的根，重心删去后各子块的重心作为其儿子，递归建出点分树。

Solution

Intuition: 这正是点分治的分治结构（点分树）。

- 对 T 做点分治：每层取当前连通块的重心作为 R 中该块的根，重心删去后各子块的重心作为其儿子，递归建出点分树。
- LCA 条件成立：R 中 x, y 的 LCA 是分治过程中第一个把它们分开的重心，而该重心必在 T 中 x–y 的路径上。

Solution

Intuition: 这正是点分治的分治结构（点分树）。

- 对 T 做点分治：每层取当前连通块的重心作为 R 中该块的根，重心删去后各子块的重心作为其儿子，递归建出点分树。
- LCA 条件成立：R 中 x, y 的 LCA 是分治过程中第一个把它们分开的重心，而该重心必在 T 中 x–y 的路径上。
- 平衡条件成立：取重心保证每个子块大小不超过父块的一半，故 $2 \cdot \mathrm{size}(v) \leq \mathrm{size}(\mathrm{parent}(v))$，树高 $O(\log N)$。

Solution

Intuition: 这正是点分治的分治结构（点分树）。

- 对 T 做点分治：每层取当前连通块的重心作为 R 中该块的根，重心删去后各子块的重心作为其儿子，递归建出点分树。
- LCA 条件成立：R 中 x, y 的 LCA 是分治过程中第一个把它们分开的重心，而该重心必在 T 中 x–y 的路径上。
- 平衡条件成立：取重心保证每个子块大小不超过父块的一半，故 $2 \cdot \mathrm{size}(v) \leq \mathrm{size}(\mathrm{parent}(v))$，树高 $O(\log N)$。
- 复杂度 $O(N \log N)$。

Table of Contents

1 树的直径

2 树的重心

3 树链剖分

4 虚树

LOJ6669 Nauuo and Binary Tree

- 交互题：隐藏的以 1 为根的二叉树，每次可询问 $\mathrm{dist}(u, v)$（两点间边数）。
- 在 $O(n \log n)$ 次询问内还原每个点的父亲。

Solution

Intuition: 树剖的非典型用法——用重链把交互询问次数压到 $O(n \log n)$，根基是「到根轻边 $\leq \log n$」。

- 先各问一次 $\mathrm{dist}(1, i)$ 得深度，按深度从小到大逐个插入；插入 u 时其父亲已在树中。

Solution

Intuition: 树剖的非典型用法——用重链把交互询问次数压到 $O(n \log n)$，根基是「到根轻边 $\leq \log n$」。

- 先各问一次 $\mathrm{dist}(1, i)$ 得深度，按深度从小到大逐个插入；插入 u 时其父亲已在树中。
- 定位分叉点：对当前子树根 x 取重链底端 b，问一次 $\mathrm{dist}(u, b)$，由 $dep_{lca} = (dep_u + dep_b - \mathrm{dist}(u, b))/2$ 一步算出 LCA 在重链上的位置，一条重链只花 $O(1)$ 次询问。

Solution

Intuition: 树剖的非典型用法——用重链把交互询问次数压到 $O(n \log n)$，根基是「到根轻边 $\leq \log n$」。

- 先各问一次 $\mathrm{dist}(1, i)$ 得深度，按深度从小到大逐个插入；插入 u 时其父亲已在树中。
- 定位分叉点：对当前子树根 x 取重链底端 b，问一次 $\mathrm{dist}(u, b)$，由 $dep_{lca} = (dep_u + dep_b - \mathrm{dist}(u, b))/2$ 一步算出 LCA 在重链上的位置，一条重链只花 $O(1)$ 次询问。
- 每次只向轻儿子方向走 $O(\log n)$ 步（轻边 $\leq \log n$），故总询问 $O(n \log n)$；二叉树每点至多两子，插入后增量维护 $size$。

Table of Contents

1. 树的直径
2. 树的重心
3. 树链剖分
4. 虚树

P2495 [SDOI2011] 消耗战

- n 个点的带权树，根为 1。m 次询问，每次给定 k 个关键点，求断开「根与所有关键点」连通所需的最小割边权和。
- 数据范围：$2 \le n \le 2.5 \times 10^5$，$1 \le m \le 5 \times 10^5$，$\sum k_i \le 5 \times 10^5$，边权 $1 \le w \le 10^5$。

Solution

Intuition: 单次树形 DP 是 $O(n)$，但 $\sum k_i$ 远小于 nm，对关键点建虚树只在 $O(\sum k_i)$ 规模上 DP。

- 预处理：DFS 求 dfn、倍增 LCA，以及每点到根路径上的最小边权 $minw_u = \min(minw_{fa}, w_{(u,fa)})$。

Solution

Intuition: 单次树形 DP 是 $O(n)$，但 $\sum k_i$ 远小于 nm，对关键点建虚树只在 $O(\sum k_i)$ 规模上 DP。

- 预处理：DFS 求 dfn、倍增 LCA，以及每点到根路径上的最小边权 $minw_u = \min(minw_{fa}, w_{(u,fa)})$。
- 每次询问把关键点按 dfn 排序，相邻点的 LCA 一并加入，用单调栈建出虚树（虚边权取两端在原树上的 $minw$）。

Solution

Intuition: 单次树形 DP 是 $O(n)$，但 $\sum k_i$ 远小于 nm，对关键点建虚树只在 $O(\sum k_i)$ 规模上 DP。

- 预处理：DFS 求 dfn、倍增 LCA，以及每点到根路径上的最小边权 $minw_u = \min(minw_{fa}, w_{(u,fa)})$。
- 每次询问把关键点按 dfn 排序，相邻点的 LCA 一并加入，用单调栈建出虚树（虚边权取两端在原树上的 $minw$）。
- 在虚树上树形 DP：$dp_u = \sum_v \big(v \text{ 为关键点? } minw_v : \min(minw_v, dp_v)\big)$，答案为 dp_{root}。

Solution

Intuition: 单次树形 DP 是 $O(n)$，但 $\sum k_i$ 远小于 nm，对关键点建虚树只在 $O(\sum k_i)$ 规模上 DP。

- 预处理：DFS 求 dfn、倍增 LCA，以及每点到根路径上的最小边权 $minw_u = \min(minw_{fa}, w_{(u,fa)})$。
- 每次询问把关键点按 dfn 排序，相邻点的 LCA 一并加入，用单调栈建出虚树（虚边权取两端在原树上的 $minw$）。
- 在虚树上树形 DP：$dp_u = \sum_v \big(v \text{ 为关键点? } minw_v : \min(minw_v, dp_v)\big)$，答案为 dp_{root}。
- 单次复杂度 $O(k \log n)$，总复杂度 $O\big(\sum k_i \log n\big)$。

P3233 [HNOI2014] 世界树

- n 个点的树，q 次询问。每次给 m 个关键点，每个点归属于距离最近的关键点（距离相同取编号最小）。
- 对每次询问，输出每个关键点管辖的点数（含自身）。
- 数据范围：$n \le 3 \times 10^5$，$\sum m \le 3 \times 10^5$。

Solution (1/4)：建虚树 + 两遍 DFS 定「归属」

为什么要虚树： $\sum m \le 3 \times 10^5$ 但 n 也很大，不能对每次询问扫全树。虚树只保留**关键点** + **它们两两的 LCA**，规模 $O(m)$。

- **建法**：关键点按 *dfn* 排序，相邻两点求 LCA 一并加入，去重后再按 *dfn* 排序，用**单调栈**连边（栈顶不是当前点祖先就弹栈连边）。点数 $\le 2m-1$。

Solution (1/4)：建虚树 + 两遍 DFS 定「归属」

为什么要虚树： $\sum m \le 3 \times 10^5$ 但 n 也很大，不能对每次询问扫全树。虚树只保留**关键点** + **它们两两的 LCA**，规模 $O(m)$。

- **建法**：关键点按 *dfn* 排序，相邻两点求 LCA 一并加入，去重后再按 *dfn* 排序，用**单调栈**连边（栈顶不是当前点祖先就弹栈连边）。点数 $\le 2m-1$。
- **第一遍 DFS（自底向上）**：(d_u, k_u) =「u 子树内最近的关键点及其编号」，用儿子的结果 + 边长更新。

Solution (1/4)：建虚树 + 两遍 DFS 定「归属」

为什么要虚树： $\sum m \le 3 \times 10^5$ 但 n 也很大，不能对每次询问扫全树。虚树只保留**关键点 + 它们两两的 LCA**，规模 $O(m)$。

- **建法**：关键点按 dfn 排序，相邻两点求 LCA 一并加入，去重后再按 dfn 排序，用**单调栈**连边（栈顶不是当前点祖先就弹栈连边）。点数 $\le 2m-1$。
- **第一遍 DFS（自底向上）**：(d_u, k_u) =「u 子树内最近的关键点及其编号」，用儿子的结果 + 边长更新。
- **第二遍 DFS（自顶向下 / 换根）**：用父亲的答案回灌，$(d_u, k_u) \leftarrow \min\big((d_u, k_u),\ (d_{fa} + w,\ k_{fa})\big)$，得到全局最近关键点。

Solution (1/4)：建虚树 + 两遍 DFS 定「归属」

为什么要虚树： $\sum m \le 3 \times 10^5$ 但 n 也很大，不能对每次询问扫全树。虚树只保留**关键点 + 它们两两的 LCA**，规模 $O(m)$。

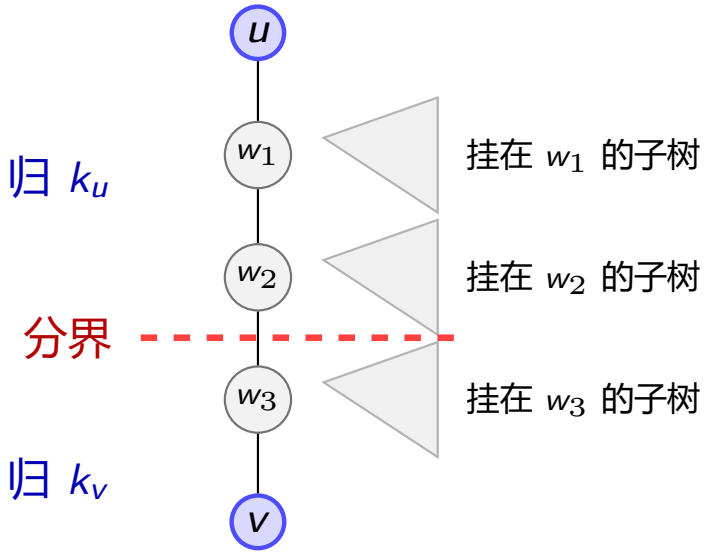
- **建法**：关键点按 *dfn* 排序，相邻两点求 LCA 一并加入，去重后再按 *dfn* 排序，用**单调栈**连边（栈顶不是当前点祖先就弹栈连边）。点数 $\le 2m-1$。
- **第一遍 DFS（自底向上）**：(d_u, k_u) = 「u 子树内最近的关键点及其编号」，用儿子的结果 + 边长更新。
- **第二遍 DFS（自顶向下 / 换根）**：用父亲的答案回灌，$(d_u, k_u) \leftarrow \min\big((d_u, k_u),\ (d_{fa} + w,\ k_{fa})\big)$，得到全局最近关键点。
- 平局规则要贯彻到底：比较时用**二元组** (距离，关键点编号) 的字典序，距离相同取编号小。两遍 DFS、后面的链上分界全都用这个比较，写成一个函数最稳。

Solution (1/4)：建虚树 + 两遍 DFS 定「归属」

为什么要虚树： $\sum m \le 3 \times 10^5$ 但 n 也很大，不能对每次询问扫全树。虚树只保留**关键点 + 它们两两的 LCA**，规模 $O(m)$。

- **建法**：关键点按 dfn 排序，相邻两点求 LCA 一并加入，去重后再按 dfn 排序，用**单调栈**连边（栈顶不是当前点祖先就弹栈连边）。点数 $\le 2m-1$。
- **第一遍 DFS（自底向上）**：(d_u, k_u) = 「u 子树内最近的关键点及其编号」，用儿子的结果 + 边长更新。
- **第二遍 DFS（自顶向下 / 换根）**：用父亲的答案回灌，$(d_u, k_u) \leftarrow \min\big((d_u, k_u),\ (d_{fa} + w,\ k_{fa})\big)$，得到全局最近关键点。
- 平局规则要贯彻到底：比较时用**二元组** $(\text{距离},\ \text{关键点编号})$ 的字典序，距离相同取编号小。两遍 DFS、后面的链上分界全都用这个比较，写成一个函数最稳。
- 求出每个虚树点的归属后，剩下的任务是**把 n 个点分配到各关键点名下**，分两类数（下页）。

Solution (2/4)：两类点——挂着的子树与边内的链



虚树边 (u, v) 对应原树一条链

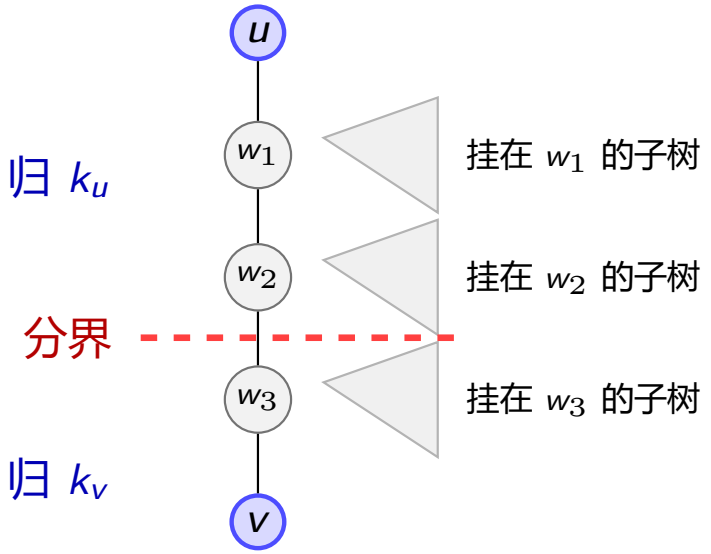
$w_1..w_{L-1}$ 不在虚树里

每个 w_i 连同它挂的子树

整块跟着 w_i 的归属走

- **第一类**：虚树点 u 自己 + 直接挂在 u 上（不通向任何虚树儿子）的真实子树 $\Rightarrow$ 整块归 k_u。大小 $= size_u - \sum_{\text{虚树儿子}v} size_{w_1(v)}$，其中 $w_1(v)$ 是 u 到 v 方向上的第一个点。

Solution (2/4)：两类点——挂着的子树与边内的链



虚树边 (u, v) 对应原树一条链

$w_1..w_{L-1}$ 不在虚树里

每个 w_i 连同它挂的子树

整块跟着 w_i 的归属走

- **第一类**：虚树点 u 自己 + 直接挂在 u 上（不通向任何虚树儿子）的真实子树 $\Rightarrow$ 整块归 k_u。大小 $= size_u - \sum_{\text{虚树儿子}v} size_{w_1(v)}$，其中 $w_1(v)$ 是 u 到 v 方向上的第一个点。
- **第二类**：每条虚树边内部的链 $w_1..w_{L-1}$，被一个分界点切成两段，上段归 k_u、下段归 k_v。

Solution (3/4)：分界点的位置与「差分」计数

设虚树边 (u, v)（u 是 v 的祖先），原树链为 $u = w_0, w_1, \ldots, w_L = v$，$L = dep_v - dep_u$。对中间点 w_i：

$$\text{走}u \text{ 侧的距离} = d_u + i, \qquad \text{走}v \text{ 侧的距离} = d_v + (L - i).$$

- 归 k_u 当且仅当 $d_u + i < d_v + L - i$，即

$$\boxed{i < \frac{d_v + L - d_u}{2}}$$

距离随 i 单调，故恰有一个分界：$w_1..w_b$ 归 k_u、$w_{b+1}..w_{L-1}$ 归 k_v。取整时若出现 $2i = d_v + L - d_u$ 的**平局**，按关键点编号比较 k_u 与 k_v 决定这一个点的归属。

Solution (3/4)：分界点的位置与「差分」计数

设虚树边 (u, v)（u 是 v 的祖先），原树链为 $u = w_0, w_1, \ldots, w_L = v$，$L = dep_v - dep_u$。对中间点 w_i：

$$\text{走}u \text{ 侧的距离} = d_u + i, \qquad \text{走}v \text{ 侧的距离} = d_v + (L - i).$$

- 归 k_u 当且仅当 $d_u + i < d_v + L - i$，即

$$\boxed{i < \frac{d_v + L - d_u}{2}}$$

  距离随 i 单调，故恰有一个分界：$w_1..w_b$ 归 k_u、$w_{b+1}..w_{L-1}$ 归 k_v。取整时若出现 $2i = d_v + L - d_u$ 的**平局**，按关键点编号比较 k_u 与 k_v 决定这一个点的归属。
- **计数用差分，不必逐点走**：w_i 连同它挂的子树共 $size_{w_i} - size_{w_{i+1}}$ 个点，求和望远镜相消：

$$k_u \text{ 得到}: \quad size_{w_1} - size_{w_{b+1}}$$
$$k_v \text{ 得到}: \quad size_{w_{b+1}} - size_v \quad (\text{再加上}v \text{ 自己那一类})$$

Solution (3/4)：分界点的位置与「差分」计数

设虚树边 (u, v)（u 是 v 的祖先），原树链为 $u = w_0, w_1, \ldots, w_L = v$，$L = dep_v - dep_u$。对中间点 w_i：

$$\text{走}u\text{ 侧的距离} = d_u + i, \qquad \text{走}v\text{ 侧的距离} = d_v + (L - i).$$

- 归 k_u 当且仅当 $d_u + i < d_v + L - i$，即

$$\boxed{i < \frac{d_v + L - d_u}{2}}$$

  距离随 i 单调，故恰有一个分界：$w_1..w_b$ 归 k_u、$w_{b+1}..w_{L-1}$ 归 k_v。取整时若出现 $2i = d_v + L - d_u$ 的**平局**，按关键点编号比较 k_u 与 k_v 决定这一个点的归属。
- **计数用差分，不必逐点走**：w_i 连同它挂的子树共 $size_{w_i} - size_{w_{i+1}}$ 个点，求和望远镜相消：

$$k_u \text{ 得到：} \quad size_{w_1} - size_{w_{b+1}}$$
$$k_v \text{ 得到：} \quad size_{w_{b+1}} - size_v \quad (\text{再加上}v\text{ 自己那一类})$$

- **与 ... 怎么找**：都是「 的第 i 级祖先」，**倍增**（或长链剖

Solution (3/4)：分界点的位置与「差分」计数

设虚树边 (u, v)（u 是 v 的祖先），原树链为 $u = w_0, w_1, \ldots, w_L = v$，$L = dep_v - dep_u$。对中间点 w_i：

$$\text{走}u\text{ 侧的距离} = d_u + i, \qquad \text{走}v\text{ 侧的距离} = d_v + (L - i).$$

- 归 k_u 当且仅当 $d_u + i < d_v + L - i$，即

$$\boxed{i < \frac{d_v + L - d_u}{2}}$$

  距离随 i 单调，故恰有一个分界：$w_1..w_b$ 归 k_u、$w_{b+1}..w_{L-1}$ 归 k_v。取整时若出现 $2i = d_v + L - d_u$ 的**平局**，按关键点编号比较 k_u 与 k_v 决定这一个点的归属。
- **计数用差分，不必逐点走**：w_i 连同它挂的子树共 $size_{w_i} - size_{w_{i+1}}$ 个点，求和望远镜相消：

$$k_u \text{ 得到：} \quad size_{w_1} - size_{w_{b+1}}$$
$$k_v \text{ 得到：} \quad size_{w_{b+1}} - size_v \quad (\text{再加上}v\text{ 自己那一类})$$

- **与 ... 怎么找**：都是「 的第 i 级祖先」，**倍增**（或长链剖

Solution (4/4)：易错点与自检

- 易错一（最常见）：**多组询问必须「局部清空」**——只重置这次用到的 $O(m)$ 个虚树点的 d, k、邻接表，不能整体 `memset`，否则单次 $O(n)$ 直接超时。

Solution (4/4)：易错点与自检

- 易错一（最常见）：**多组询问必须「局部清空」**——只重置这次用到的 $O(m)$ 个虚树点的 d, k、邻接表，不能整体 `memset`，否则单次 $O(n)$ 直接超时。
- 易错二：平局取编号小这条规则要用在所有比较处（两遍 DFS、链上分界）。只在一处漏掉就会 WA 且极难定位——建议写成统一的 `better(d1,k1,d2,k2)` 比较函数。

Solution (4/4)：易错点与自检

- 易错一（最常见）：**多组询问必须「局部清空」**——只重置这次用到的 $O(m)$ 个虚树点的 d, k、邻接表，不能整体 `memset`，否则单次 $O(n)$ 直接超时。
- 易错二：平局取编号小这条规则要用在所有比较处（两遍 DFS、链上分界）。只在一处漏掉就会 WA 且极难定位——建议写成统一的 `better(d1,k1,d2,k2)` 比较函数。
- 易错三：第一类的「挂着的子树」大小要扣掉每个虚树儿子方向的 $size_{w_1}$，注意是 w_1（链上第一个点）而不是虚树儿子 v 本身。

Solution (4/4)：易错点与自检

- 易错一（最常见）：**多组询问必须「局部清空」**——只重置这次用到的 $O(m)$ 个虚树点的 d, k、邻接表，不能整体 `memset`，否则单次 $O(n)$ 直接超时。
- 易错二：平局取编号小这条规则要用在所有比较处（两遍 DFS、链上分界）。只在一处漏掉就会 WA 且极难定位——建议写成统一的 `better(d1,k1,d2,k2)` 比较函数。
- 易错三：第一类的「挂着的子树」大小要扣掉每个虚树儿子方向的 $size_{w_1}$，注意是 w_1（链上第一个点）而不是虚树儿子 v 本身。
- 易错四：虚树建好后边权是 $dep_v - dep_u$（原树距离），不是 1。

Solution (4/4)：易错点与自检

- 易错一（最常见）：**多组询问必须「局部清空」**——只重置这次用到的 $O(m)$ 个虚树点的 d, k、邻接表，不能整体 `memset`，否则单次 $O(n)$ 直接超时。
- 易错二：平局取编号小这条规则要用在所有比较处（两遍 DFS、链上分界）。只在一处漏掉就会 WA 且极难定位——建议写成统一的 `better(d1,k1,d2,k2)` 比较函数。
- 易错三：第一类的「挂着的子树」大小要扣掉每个虚树儿子方向的 $size_{w_1}$，注意是 w_1（链上第一个点）而不是虚树儿子 v 本身。
- 易错四：虚树建好后边权是 $dep_v - dep_u$（原树距离），不是 1。
- **自检**：所有关键点的答案**之和必须恰为** n。这是一行代码的断言，能抓住绝大多数分类漏算/重算——务必让学生写上。