树上搜索及相关应用

汪直方

宁波市镇海蛟川书院

2024 年 2 月 18 日

深度优先搜索是树上最常用的搜索。

深度优先搜索是树上最常用的搜索。

Problem

给定一棵 n 个结点的有根树的边集，求每个结点的父亲结点编号、深度、高度、子树大小。

要求时间复杂度 $O(n)$。

Solution

从根开始进行深度优先搜索：传入当前结点 u 与其父亲（对于根不妨传入 0），求出其深度为其父亲结点深度加一，枚举与其相邻的结点 v，若 v 不为 u 的父亲，则其为 u 的儿子，递归求出其所有信息，求出 u 的高度为所有儿子的高度的最大值加一，u 的子树大小为所有儿子的子树大小和加一。

Solution

从根开始进行深度优先搜索：传入当前结点 u 与其父亲（对于根不妨传入 0），求出其深度为其父亲结点深度加一，枚举与其相邻的结点 v，若 v 不为 u 的父亲，则其为 u 的儿子，递归求出其所有信息，求出 u 的高度为所有儿子的高度的最大值加一，u 的子树大小为所有儿子的子树大小和加一。

时空复杂度 $\Theta(n)$。

Problem (离线 k 级祖先)

给定一棵 n 个结点的树，q 次查询：给定结点 u 与整数 k，求 $\mathrm{par}_k(u)$。

要求时间复杂度 $O(n+q)$。

Problem (离线 k 级祖先)

给定一棵 n 个结点的树，q 次查询：给定结点 u 与整数 k，求 $\mathrm{par}_k(u)$。

要求时间复杂度 $O(n+q)$。

Solution

我们将每个询问存在对应的结点上。

Problem (离线 k 级祖先)

给定一棵 n 个结点的树，q 次查询：给定结点 u 与整数 k，求 $\mathrm{par}_k(u)$。

要求时间复杂度 $O(n+q)$。

Solution

我们将每个询问存在对应的结点上。

注意到我们可以在深度优先搜索时，记录递归栈即根到当前结点路径上的结点序列，我们进行深度优先搜索并在搜索到每个结点时处理其的所有询问即可。

Problem (离线 k 级祖先)

给定一棵 n 个结点的树，q 次查询：给定结点 u 与整数 k，求 $\mathrm{par}_k(u)$。

要求时间复杂度 $O(n+q)$。

Solution

我们将每个询问存在对应的结点上。

注意到我们可以在深度优先搜索时，记录递归栈即根到当前结点路径上的结点序列，我们进行深度优先搜索并在搜索到每个结点时处理其的所有询问即可。

时空复杂度 $\Theta(n+q)$。

类似深度优先搜索，我们也可以在树上使用广度优先搜索。

类似深度优先搜索，我们也可以在树上使用广度优先搜索。
广度优先搜索序的一些性质会在后面介绍。

对于一棵给定的 n 个结点的有根树 T，以根为起点深度优先搜索并维护一个初始为空的序列，每次递归访问一个结点时将其插入序列末尾，搜索结束后得到一个长度为 n 的序列，为 $[1,n]\cap\mathbb{N}$ 的置换，称为有根树 T 的深度优先搜索序（dfs 序），记为 $\mathrm{dfn}(T)$。称结点 u 在 T 中的 dfs 序为结点 u 在 T 的 dfs 序中的位置，记为 $\mathrm{dfn}_T(u)$，不引起歧义的情况下可以称为结点 u 的 dfs 序，记为 $\mathrm{dfn}(u)$。

对于一棵给定的 n 个结点的有根树 T，以根为起点深度优先搜索并维护一个初始为空的序列，每次递归访问一个结点时将其插入序列末尾，搜索结束后得到一个长度为 n 的序列，为 $[1,n]\cap\mathbb{N}$ 的置换，称为有根树 T 的深度优先搜索序（dfs 序），记为 $\mathrm{dfn}(T)$。称结点 u 在 T 中的 dfs 序为结点 u 在 T 的 dfs 序中的位置，记为 $\mathrm{dfn}_T(u)$，不引起歧义的情况下可以称为结点 u 的 dfs 序，记为 $\mathrm{dfn}(u)$。

此外，根据实际应用场景，还有两种常见的变种：

对于一棵给定的 n 个结点的有根树 T，以根为起点深度优先搜索并维护一个初始为空的序列，每次递归访问一个结点时将其插入序列末尾，搜索结束后得到一个长度为 n 的序列，为 $[1,n]\cap\mathbb{N}$ 的置换，称为有根树 T 的深度优先搜索序（dfs 序），记为 $\mathrm{dfn}(T)$。称结点 u 在 T 中的 dfs 序为结点 u 在 T 的 dfs 序中的位置，记为 $\mathrm{dfn}_T(u)$，不引起歧义的情况下可以称为结点 u 的 dfs 序，记为 $\mathrm{dfn}(u)$。

此外，根据实际应用场景，还有两种常见的变种：

1. 括号序：递归访问一个结点时将其对应左括号插入序列末尾，在回溯时将其对应右括号插入序列末尾，长度为 $2n$。

对于一棵给定的 n 个结点的有根树 T，以根为起点深度优先搜索并维护一个初始为空的序列，每次递归访问一个结点时将其插入序列末尾，搜索结束后得到一个长度为 n 的序列，为 $[1,n]\cap\mathbb{N}$ 的置换，称为有根树 T 的深度优先搜索序（dfs 序），记为 $\mathrm{dfn}(T)$。称结点 u 在 T 中的 dfs 序为结点 u 在 T 的 dfs 序中的位置，记为 $\mathrm{dfn}_T(u)$，不引起歧义的情况下可以称为结点 u 的 dfs 序，记为 $\mathrm{dfn}(u)$。

此外，根据实际应用场景，还有两种常见的变种：

1. 括号序：递归访问一个结点时将其对应左括号插入序列末尾，在回溯时将其对应右括号插入序列末尾，长度为 $2n$。
2. 欧拉序：递归访问一个结点时将其插入序列末尾，在非根结点回溯时将其父亲结点插入序列末尾，长度为 $2n-1$，相当于将树每条边拆成两条有向边后的欧拉回路上的点依序排列。

Property 1.1

对于一棵有根树，u 为其中的结点，则 $\mathrm{subtree}(u)$ 在树的 dfs 序中为一个区间的值域。

Problem

有两个集合 A, B，对于 $a \in A, b \in B$，可以在 $\Theta(1)$ 的复杂度内计算出 $b * a \in A$，B 上存在可以在 $\Theta(1)$ 复杂度内计算的二元运算 $\cdot$，满足 $\forall a \in A, b_1, b_2 \in B, b_2 * (b_1 * a) = (b_2 \cdot b_1) * a$，$A$ 上存在可以在 $\Theta(1)$ 复杂度计算的运算 $\oplus$，满足 $\forall a_1, a_2, a_3 \in A, (a_1 \oplus a_2) \oplus a_3 = a_1 \oplus (a_2 \oplus a_3)$ 且 $\forall a_1, a_2 \in A, b \in B, (b * a_1) \oplus (b * a_2) = b * (a_1 \oplus a_2)$。

给定一棵点集为 V 的有根树，对 $u \in V$ 给定 $a_u \in A$，q 次操作：

1. 给定 $u \in V, x \in A$，执行 $a_u \leftarrow x$。
2. 给定 $u \in V$，求 a_u。
3. 给定 $u \in V, x \in B$，对 $v \in \text{subtree}(u)$ 执行 $a_v \leftarrow x * a_v$。
4. 给定 $u \in V$，求 $\bigoplus_{v \in \text{subtree}(u)} a_v$，其中 $\oplus$ 可按 dfs 序进行。

$n = |V|$，要求时间复杂度 $O(n + q \log n)$。

Solution

我们将结点按 dfs 序重标号，将问题转化为区间操作区间查询使用线段树解决。

Solution

我们将结点按 dfs 序重标号，将问题转化为区间操作区间查询使用线段树解决。

时间复杂度 $\Theta(n + q \log n)$。

Solution

我们将结点按 dfs 序重标号，将问题转化为区间操作区间查询使用线段树解决。

时间复杂度 $\Theta(n + q \log n)$。

对于一些情况如操作为整数加查询为整数求和，同样可以使用树状数组减小常数。

Problem

有一个集合 A 关于可以在 $\Theta(1)$ 复杂度计算的其上二元运算 $\oplus$ 构成群，且对任意 $a \in A$，可以 $\Theta(1)$ 计算其逆元。

给定一棵点集为 V 的有根树，对 $u \in V$ 给定 $a_u \in A$，q 次操作：

1. 给定 $u \in V, x \in A$，执行 $a_u \leftarrow x$。
2. 给定 $u, v \in V$，求 $\bigoplus_{w \in \text{path}(u,v)} a_w$，其中 $\oplus$ 按路径顺序进行。

$n = |V|$，要求时间复杂度 $O(n + q \log n)$。

Solution

注意到对于树的括号序，两个点之间的简单路径边集可以用两点对应左括号间的左开右闭区间删去匹配括号对后的剩余位置表示，右括号表示对应结点向其父亲的有向边，左括号表示对应结点父亲向其的有向边。

Solution

注意到对于树的括号序，两个点之间的简单路径边集可以用两点对应左括号间的左开右闭区间删去匹配括号对后的剩余位置表示，右括号表示对应结点向其父亲的有向边，左括号表示对应结点父亲向其的有向边。

在维护点信息时，我们可以通过最近公共祖先将路径拆成端点为祖先后代关系的两条路径，左括号维护结点对应信息，右括号维护其逆，端点左括号间的左开右闭区间的信息之积即为祖先到后代的路径去除祖先后的信息之积，右括号间的左闭右开区间的信息之积即为后代到祖先的路径去除祖先后的信息之积的逆。

Solution

注意到对于树的括号序，两个点之间的简单路径边集可以用两点对应左括号间的左开右闭区间删去匹配括号对后的剩余位置表示，右括号表示对应结点向其父亲的有向边，左括号表示对应结点父亲向其的有向边。

在维护点信息时，我们可以通过最近公共祖先将路径拆成端点为祖先后代关系的两条路径，左括号维护结点对应信息，右括号维护其逆，端点左括号间的左开右闭区间的信息之积即为祖先到后代的路径去除祖先后的信息之积，右括号间的左闭右开区间的信息之积即为后代到祖先的路径去除祖先后的信息之积的逆。

查询求出最近公共祖先后使用线段树进行两次区间查询与一次单点查询（最近公共祖先）即可，单点修改是容易的。

Solution

注意到对于树的括号序，两个点之间的简单路径边集可以用两点对应左括号间的左开右闭区间删去匹配括号对后的剩余位置表示，右括号表示对应结点向其父亲的有向边，左括号表示对应结点父亲向其的有向边。

在维护点信息时，我们可以通过最近公共祖先将路径拆成端点为祖先后代关系的两条路径，左括号维护结点对应信息，右括号维护其逆，端点左括号间的左开右闭区间的信息之积即为祖先到后代的路径去除祖先后的信息之积，右括号间的左闭右开区间的信息之积即为后代到祖先的路径去除祖先后的信息之积的逆。

查询求出最近公共祖先后使用线段树进行两次区间查询与一次单点查询（最近公共祖先）即可，单点修改是容易的。

时间复杂度 $\Theta(n + q \log n)$。

Problem

有两个集合 A, B，对于 $a \in A, b \in B$，可以在 $\Theta(1)$ 的复杂度内计算出 $b * a \in A$，B 上存在可以在 $\Theta(1)$ 复杂度内计算的二元运算 $\cdot$，满足 $\forall a \in A, b_1, b_2 \in B, b_2 * (b_1 * a) = (b_2 \cdot b_1) * a$，$A$ 上存在可以在 $\Theta(1)$ 复杂度计算的运算 $\oplus$，满足 A 关于可以在 $\Theta(1)$ 复杂度计算的其上二元运算 $\oplus$ 构成交换群，且对任意 $a \in A$，可以 $\Theta(1)$ 计算其逆元，且 $\forall a_1, a_2 \in A, b \in B, (b * a_1) \oplus (b * a_2) = b * (a_1 \oplus a_2)$。

给定一棵点集为 V 的有根树，对 $u \in V$ 给定 $a_u \in A$，q 次操作：

1. 给定 $u \in V, x \in A$，执行 $a_u \leftarrow x$。
2. 给定 $u \in V, x \in B$，对 $v \in \text{subtree}(u)$ 执行 $a_v \leftarrow x * a_v$。
3. 给定 $u, v \in V$，求 $\bigoplus_{w \in \text{path}(u,v)} a_w$。

$n = |V|$，要求时间复杂度 $O(n + q \log n)$。

Solution

将左括号与右括号的区间和分开维护即可支持子树修改，其余与不带子树修改的版本相同。

Solution

将左括号与右括号的区间和分开维护即可支持子树修改，其余与不带子树修改的版本相同。

时间复杂度 $\Theta(n + q \log n)$。

Problem

有一个集合 A 关于可以在 $\Theta(1)$ 复杂度计算的其上二元运算 $\oplus$ 构成交换群，且对任意 $a \in A$，可以 $\Theta(1)$ 计算其逆元。

给定一棵点集为 V 的有根树，对 $u \in V$ 给定 $a_u \in A$，q 次操作：

1. 给定 $u \in V, x \in A$，对 $v \in \mathrm{subtree}(u)$ 执行 $a_v \leftarrow x \oplus a_v$。
2. 给定 $u, v \in V, x \in A$，对 $w \in \mathrm{path}(u, v)$ 执行 $a_w \leftarrow x \oplus a_w$。
3. 给定 $u \in V$，求 a_u。
4. 给定 $u \in V$，求 $\bigoplus_{v \in \mathrm{subtree}(u)} a_v$。

Solution

注意到 $\oplus$ 具有交换律，我们只需考虑链加与单点或子树查询。

Solution

注意到 $\oplus$ 具有交换律，我们只需考虑链加与单点或子树查询。

链加时，我们先求出 u, v 的最近公共祖先 $\mathrm{lca}(u, v)$，再对 $\mathrm{lca}(u, v)$ 与 u 的左括号的左闭右闭区间与 $\mathrm{lca}(u, v)$ 与 v 的左括号的左开右闭区间进行区间加，一个结点的值即为左括号值减右括号值。

Solution

注意到 $\oplus$ 具有交换律，我们只需考虑链加与单点或子树查询。

链加时，我们先求出 u, v 的最近公共祖先 $\mathrm{lca}(u, v)$，再对 $\mathrm{lca}(u, v)$ 与 u 的左括号的左闭右闭区间与 $\mathrm{lca}(u, v)$ 与 v 的左括号的左开右闭区间进行区间加，一个结点的值即为左括号值减右括号值。

我们分别维护区间内的左括号和与右括号和即可支持求子树和。

Solution

注意到 $\oplus$ 具有交换律，我们只需考虑链加与单点或子树查询。

链加时，我们先求出 u, v 的最近公共祖先 $\mathrm{lca}(u, v)$，再对 $\mathrm{lca}(u, v)$ 与 u 的左括号的左闭右闭区间与 $\mathrm{lca}(u, v)$ 与 v 的左括号的左开右闭区间进行区间加，一个结点的值即为左括号值减右括号值。

我们分别维护区间内的左括号和与右括号和即可支持求子树和。

时间复杂度 $\Theta(n + q \log n)$。

Solution

注意到 $\oplus$ 具有交换律，我们只需考虑链加与单点或子树查询。

链加时，我们先求出 u, v 的最近公共祖先 $\mathrm{lca}(u, v)$，再对 $\mathrm{lca}(u, v)$ 与 u 的左括号的左闭右闭区间与 $\mathrm{lca}(u, v)$ 与 v 的左括号的左开右闭区间进行区间加，一个结点的值即为左括号值减右括号值。

我们分别维护区间内的左括号和与右括号和即可支持求子树和。

时间复杂度 $\Theta(n + q \log n)$。

子树修改也可视为对区间左括号进行修改。

Problem (【ZJOI2007】捉迷藏)

给定一棵 n 个结点的树，有一个树点集的子集 S 初始为 $[1,n]\cap\mathbb{N}$。q 次操作：向 S 中插入或从 S 中删除一个结点后查询 S 中点对的最大距离。

$1\leqslant n\leqslant 10^5, 1\leqslant q\leqslant 5\times 10^5$。

Problem (【ZJOI2007】捉迷藏)

给定一棵 n 个结点的树，有一个树点集的子集 S 初始为 $[1,n]\cap\mathbb{N}$。q 次操作：向 S 中插入或从 S 中删除一个结点后查询 S 中点对的最大距离。

$1\leqslant n\leqslant 10^5, 1\leqslant q\leqslant 5\times 10^5$。

Solution

我们处理出树的括号序，对左括号维护 1，右括号维护 -1，两个结点对应右括号间的左闭右开区间的最大后缀和减最小前缀和即为两点距离。

Problem (【ZJOI2007】捉迷藏)

给定一棵 n 个结点的树，有一个树点集的子集 S 初始为 $[1,n]\cap\mathbb{N}$。q 次操作：向 S 中插入或从 S 中删除一个结点后查询 S 中点对的最大距离。

$1\leqslant n\leqslant 10^5, 1\leqslant q\leqslant 5\times 10^5$。

Solution

我们处理出树的括号序，对左括号维护 1，右括号维护 -1，两个结点对应右括号间的左闭右开区间的最大后缀和减最小前缀和即为两点距离。

用线段树维护即可，时间复杂度 $\Theta(n+q\log n)$。

Problem (【ZJOI2007】捉迷藏)

给定一棵 n 个结点的树，有一个树点集的子集 S 初始为 $[1,n]\cap\mathbb{N}$。q 次操作：向 S 中插入或从 S 中删除一个结点后查询 S 中点对的最大距离。

$1\leqslant n\leqslant 10^5, 1\leqslant q\leqslant 5\times 10^5$。

Solution

我们处理出树的括号序，对左括号维护 1，右括号维护 -1，两个结点对应右括号间的左闭右开区间的最大后缀和减最小前缀和即为两点距离。

用线段树维护即可，时间复杂度 $\Theta(n+q\log n)$。

将线段树改为平衡树甚至可以相当方便地在同复杂度内支持换父亲操作。

对于一棵给定的 n 个结点的有根树，以根为起点广度优先搜索并维护一个初始为空的序列，每次访问一个结点时将其插入序列末尾，搜索结束后得到一个长度为 n 的序列，为 $[1,n]\cap\mathbb{N}$ 的置换，称为给定有根树的广度优先搜索序（bfs 序）。称结点 u 在给定有根树中的 bfs 序为结点 u 在其 bfs 序中的位置，不引起歧义的情况下可以称为结点 u 的 bfs 序。

对于一棵给定的 n 个结点的有根树，以根为起点广度优先搜索并维护一个初始为空的序列，每次访问一个结点时将其插入序列末尾，搜索结束后得到一个长度为 n 的序列，为 $[1,n]\cap\mathbb{N}$ 的置换，称为给定有根树的广度优先搜索序（bfs 序）。称结点 u 在给定有根树中的 bfs 序为结点 u 在其 bfs 序中的位置，不引起歧义的情况下可以称为结点 u 的 bfs 序。

Property 2.1

对于一棵有根树，u 为其中的结点，则 $\mathrm{son}(u)$ 在树的 bfs 序中为一个区间的值域。

Problem (SOJ1374 会是·加强版)

有一棵 n 个结点的树，结点按 bfs 序编号。构造 $\max(n-2,0)$ 对结点对进行一次查询，交互库会返回每对结点对的距离，令 $f(u)=\max_{v\in \text{subtree}(u)} \text{dep}(v)$，其中 $\text{dep}(1)=1$，求 $\sum_{i=1}^{n} f(i)$。

$1\leqslant n\leqslant 5\times 10^6$。

Problem (SOJ1374 会是·加强版)

有一棵 n 个结点的树，结点按 bfs 序编号。构造 $\max(n-2,0)$ 对结点对进行一次查询，交互库会返回每对结点对的距离，令 $f(u)=\max_{v\in \text{subtree}(u)} \text{dep}(v)$，其中 $\text{dep}(1)=1$，求 $\sum_{i=1}^{n} f(i)$。

$1\leqslant n\leqslant 5\times 10^6$。

Solution

我们可以通过 $\{(i,i-1)|i\in[3,n]\cap\mathbb{N}\}$ 查询出 $i\in[2,n]\cap\mathbb{N}$ 的所有 $\text{dist}(i-1,i)$。

Problem (SOJ1374 会是·加强版)

有一棵 n 个结点的树，结点按 bfs 序编号。构造 $\max(n-2,0)$ 对结点对进行一次查询，交互库会返回每对结点对的距离，令 $f(u)=\max_{v\in \text{subtree}(u)} \text{dep}(v)$，其中 $\text{dep}(1)=1$，求 $\sum_{i=1}^{n} f(i)$。

$1\leqslant n\leqslant 5\times 10^6$。

Solution

我们可以通过 $\{(i,i-1)|i\in[3,n]\cap\mathbb{N}\}$ 查询出 $i\in[2,n]\cap\mathbb{N}$ 的所有 $\text{dist}(i-1,i)$。

注意到若 $\text{dist}(i-1,i)$ 为偶数，则 $\text{dep}(i)=\text{dep}(i-1)$，否则 $\text{dep}(i)=\text{dep}(i+1)$。

Solution

注意到 $n=1$ 时树形态确定答案为 1，我们考虑对于 $i\in[2,n]\cap\mathbb{N}$，根据 $[1,i)\cap\mathbb{N}$ 的导出子图的答案求出 $[1,i]\cap\mathbb{N}$ 的导出子图的答案。

Solution

注意到 $n=1$ 时树形态确定答案为 1，我们考虑对于 $i\in[2,n]\cap\mathbb{N}$，根据 $[1,i)\cap\mathbb{N}$ 的导出子图的答案求出 $[1,i]\cap\mathbb{N}$ 的导出子图的答案。

若 $\mathrm{dep}(i)=\mathrm{dep}(i-1)$ 则加入结点 i 后导出子图的 $\mathrm{anc}(i)\setminus\mathrm{anc}(i-1)$ 中的结点 f 会加一，否则 $\mathrm{anc}(i)$ 中的结点 f 会加一，两者的集合大小分别为 $\mathrm{dist}(i,i-1)/2$ 和 $\mathrm{dep}(i)$。此外答案还会增加 $f(i)$ 即 $\mathrm{dep}(i)$。

Solution

注意到 $n=1$ 时树形态确定答案为 1，我们考虑对于 $i\in[2,n]\cap\mathbb{N}$，根据 $[1,i)\cap\mathbb{N}$ 的导出子图的答案求出 $[1,i]\cap\mathbb{N}$ 的导出子图的答案。

若 $\mathrm{dep}(i)=\mathrm{dep}(i-1)$ 则加入结点 i 后导出子图的 $\mathrm{anc}(i)\setminus\mathrm{anc}(i-1)$ 中的结点 f 会加一，否则 $\mathrm{anc}(i)$ 中的结点 f 会加一，两者的集合大小分别为 $\mathrm{dist}(i,i-1)/2$ 和 $\mathrm{dep}(i)$。此外答案还会增加 $f(i)$ 即 $\mathrm{dep}(i)$。

时空复杂度 $\Theta(n)$。

Problem (【NOI2013】树的计数)

给定 n 个结点的带标号有根树的 dfs 序和 bfs 序，求可能的树的集合的平均树高。(两种搜索序中遍历儿子的顺序一致。)

$2 \leqslant n \leqslant 2 \times 10^5$。

Problem (【NOI2013】树的计数)

给定 n 个结点的带标号有根树的 dfs 序和 bfs 序，求可能的树的集合的平均树高。(两种搜索序中遍历儿子的顺序一致。)

$2 \leqslant n \leqslant 2 \times 10^5$。

Solution

我们可以将结点按照 dfs 序重标号，因此不妨设 dfs 序为 $1, 2, \ldots, n$。

Problem (【NOI2013】树的计数)

给定 n 个结点的带标号有根树的 dfs 序和 bfs 序，求可能的树的集合的平均树高。(两种搜索序中遍历儿子的顺序一致。)

$2 \leqslant n \leqslant 2 \times 10^5$。

Solution

我们可以将结点按照 dfs 序重标号，因此不妨设 dfs 序为 $1, 2, \ldots, n$。

此时 bfs 序即结点以深度为第一关键字，标号为第二关键字的排序结果，树高即 bfs 序最后一个结点的深度。

Problem (【NOI2013】树的计数)

给定 n 个结点的带标号有根树的 dfs 序和 bfs 序，求可能的树的集合的平均树高。(两种搜索序中遍历儿子的顺序一致。)

$2 \leqslant n \leqslant 2 \times 10^5$。

Solution

我们可以将结点按照 dfs 序重标号，因此不妨设 dfs 序为 $1, 2, \ldots, n$。

此时 bfs 序即结点以深度为第一关键字，标号为第二关键字的排序结果，树高即 bfs 序最后一个结点的深度。

假设我们已知了 $\mathrm{subtree}(l) = [l, r] \cap \mathbb{N}$，考察 l 的子树形态。

Problem (【NOI2013】树的计数)

给定 n 个结点的带标号有根树的 dfs 序和 bfs 序，求可能的树的集合的平均树高。(两种搜索序中遍历儿子的顺序一致。)

$2 \leqslant n \leqslant 2 \times 10^5$。

Solution

我们可以将结点按照 dfs 序重标号，因此不妨设 dfs 序为 $1, 2, \ldots, n$。

此时 bfs 序即结点以深度为第一关键字，标号为第二关键字的排序结果，树高即 bfs 序最后一个结点的深度。

假设我们已知了 $\mathrm{subtree}(l) = [l, r] \cap \mathbb{N}$，考察 l 的子树形态。

若 $l = r$，则 l 的子树由 l 单个结点构成。下面均不考虑该情况即 $l+1$ 为 l 的第一个儿子。

Solution

若 bfs 序中 $l+1$ 之后存在子树外的结点，设在 bfs 序中的第一个为 x，则 $\mathrm{par}(x)$ 一定在子树外。

Solution

若 bfs 序中 $l+1$ 之后存在子树外的结点，设在 bfs 序中的第一个为 x，则 $\mathrm{par}(x)$ 一定在子树外。

注意到 bfs 序之中 $\mathrm{par}(x)$ 之后 x 之前不可能存在一对父子结点，因此 bfs 序中从 $l+1$ 到 x 的前一个均为 l 的儿子，从而得到 l 每个儿子子树的结点范围，可以递归求解。

Solution

若 bfs 序中 $l+1$ 之后不存在子树外的结点，若 x 是最大的满足 $[l+1,x]\cap\mathbb{N}$ 在 bfs 序连续的结点，若 $x=n$ 则 x 没有儿子和右侧的兄弟，否则可以根据 $x+1$ 的位置求出 x 右侧的所有兄弟，并类似递归求解这些结点的子树。

Solution

若 bfs 序中 $l+1$ 之后不存在子树外的结点，若 x 是最大的满足 $[l+1,x]\cap\mathbb{N}$ 在 bfs 序连续的结点，若 $x=n$ 则 x 没有儿子和右侧的兄弟，否则可以根据 $x+1$ 的位置求出 x 右侧的所有兄弟，并类似递归求解这些结点的子树。

而对于 $i\in[l+2,x]\cap\mathbb{N}$，一种情况为 i 为 $i-1$ 的右兄弟且 $i-1$ 没有儿子，另一种情况为 i 为 $i-1$ 的第一个儿子且 $i-1$ 没有右侧的兄弟，且在这种叙述下所有选择相互独立，x 的期望深度为 $l+1$ 的期望深度加 $(x-l-1)/2$。

Solution

若 bfs 序中 $l+1$ 之后不存在子树外的结点，若 x 是最大的满足 $[l+1,x]\cap\mathbb{N}$ 在 bfs 序连续的结点，若 $x=n$ 则 x 没有儿子和右侧的兄弟，否则可以根据 $x+1$ 的位置求出 x 右侧的所有兄弟，并类似递归求解这些结点的子树。

而对于 $i\in[l+2,x]\cap\mathbb{N}$，一种情况为 i 为 $i-1$ 的右兄弟且 $i-1$ 没有儿子，另一种情况为 i 为 $i-1$ 的第一个儿子且 $i-1$ 没有右侧的兄弟，且在这种叙述下所有选择相互独立，x 的期望深度为 $l+1$ 的期望深度加 $(x-l-1)/2$。

时间复杂度 $\Theta(n)$。

Solution

若 bfs 序中 $l+1$ 之后不存在子树外的结点，若 x 是最大的满足 $[l+1,x]\cap\mathbb{N}$ 在 bfs 序连续的结点，若 $x=n$ 则 x 没有儿子和右侧的兄弟，否则可以根据 $x+1$ 的位置求出 x 右侧的所有兄弟，并类似递归求解这些结点的子树。

而对于 $i\in[l+2,x]\cap\mathbb{N}$，一种情况为 i 为 $i-1$ 的右兄弟且 $i-1$ 没有儿子，另一种情况为 i 为 $i-1$ 的第一个儿子且 $i-1$ 没有右侧的兄弟，且在这种叙述下所有选择相互独立，x 的期望深度为 $l+1$ 的期望深度加 $(x-l-1)/2$。

时间复杂度 $\Theta(n)$。

按 bfs 序重标号也可以类似分析，基于期望的线性性计算相邻结点深度差的概率可以得到此题的一个较简单写法。

虚树，即根据内在逻辑等非直接给定因素虚拟建立的树。

虚树，即根据内在逻辑等非直接给定因素虚拟建立的树。

在给定一棵树的情况下，对于一个给定点集，其点对间路径之并为原给定树的一个连通子图，对其中点集外的二度点进行 compress 操作，我们可以得到一个点集为给定点集及点对间路径之间的交点，边集对应原树上以其顶点为端点的链，这也是原树进行 rake 和 compress 的树收缩后得到的保留给定点集的最小树，称为给定点集在给定无根树上的虚树，在不引起歧义的情况下，可称为给定点集的虚树。

虚树，即根据内在逻辑等非直接给定因素虚拟建立的树。

在给定一棵树的情况下，对于一个给定点集，其点对间路径之并为原给定树的一个连通子图，对其中点集外的二度点进行 compress 操作，我们可以得到一个点集为给定点集及点对间路径之间的交点，边集对应原树上以其顶点为端点的链，这也是原树进行 rake 和 compress 的树收缩后得到的保留给定点集的最小树，称为给定点集在给定无根树上的虚树，在不引起歧义的情况下，可称为给定点集的虚树。

对于一个大小为 m 的点集，其虚树的结点个数为 $\Theta(m)$。求解其在原给定树上的问题时，我们常常可以将原树中的相关信息转移至其虚树的点与边上，再在虚树上用其它方案求解。

对于有根树的情况，我们在问题中常常需要树序的信息，这时与有根情况的 top tree 类似，所有结点的最近公共祖先即使不在点集中且只有两个儿子的子树内有给定点，依然不适合被作为 compress 操作的对象：保持簇的两个端点为祖先后代关系在一些题目的分析中是必要的。

对于有根树的情况，我们在问题中常常需要树序的信息，这时与有根情况的 top tree 类似，所有结点的最近公共祖先即使不在点集中且只有两个儿子的子树内有给定点，依然不适合被作为 compress 操作的对象：保持簇的两个端点为祖先后代关系在一些题目的分析中是必要的。

我们可以证明，一个给定点集在给定有根树作为无根树上的虚树的点集与仅包含点集最近公共祖先的集合的并为给定点集任意子集最近公共祖先构成的集合。

对于有根树的情况，我们在问题中常常需要树序的信息，这时与有根情况的 top tree 类似，所有结点的最近公共祖先即使不在点集中且只有两个儿子的子树内有给定点，依然不适合被作为 compress 操作的对象：保持簇的两个端点为祖先后代关系在一些题目的分析中是必要的。

我们可以证明，一个给定点集在给定有根树作为无根树上的虚树的点集与仅包含点集最近公共祖先的集合的并为给定点集任意子集最近公共祖先构成的集合。

我们将该集合在给定有根树作为无根树上的虚树称为原给定点集在给定有根树上的虚树，在不引起歧义的情况下，同样可称为给定点集的虚树。

对于有根树的情况，我们在问题中常常需要树序的信息，这时与有根情况的 top tree 类似，所有结点的最近公共祖先即使不在点集中且只有两个儿子的子树内有给定点，依然不适合被作为 compress 操作的对象：保持簇的两个端点为祖先后代关系在一些题目的分析中是必要的。

我们可以证明，一个给定点集在给定有根树作为无根树上的虚树的点集与仅包含点集最近公共祖先的集合的并为给定点集任意子集最近公共祖先构成的集合。

我们将该集合在给定有根树作为无根树上的虚树称为原给定点集在给定有根树上的虚树，在不引起歧义的情况下，同样可称为给定点集的虚树。

我们也可以证明，一个点集在给定有根树上的虚树的点集即为其任意子集最近公共祖先构成的集合。

这也直接地给出了给定点集在给定有根树上的虚树的一种建立方式：我们将给定点集按照 dfs 序枚举，用栈维护前缀最近公共祖先至当前结点的链中在前缀结点集的虚树中出现的结点，每次求当前结点与上一结点的最近公共祖先 l 并将栈中深度超过 l 的结点弹出并连边，若 l 不在栈中则插入栈，再将当前结点也插入栈。

这也直接地给出了给定点集在给定有根树上的虚树的一种建立方式：我们将给定点集按照 dfs 序枚举，用栈维护前缀最近公共祖先至当前结点的链中在前缀结点集的虚树中出现的结点，每次求当前结点与上一结点的最近公共祖先 l 并将栈中深度超过 l 的结点弹出并连边，若 l 不在栈中则插入栈，再将当前结点也插入栈。

对于无根树，我们自然可以随意指定一个根后按有根树建出虚树再对点集最近公共祖先进行判断，但实际上判断在多数题目中并不必要：基本不影响后续解法的正确性（可以基于有根树考虑后续解法），在后续算法复杂度关于虚树大小不超过指数级别的情况下均不会影响复杂度，甚至在后续算法复杂度近线性的情况下可能因为该判断而增加常数。

虚树也可以用来辅助思考一些树上相关结点为常数的性质。

虚树也可以用来辅助思考一些树上相关结点为常数的性质。

Property 1.1

对于一棵点集为 V 的树，若 $\{\text{path}(s,t)|s,t\in V\}$ 中的路径两两相交，则所有路径一定交于至少一点。

Problem (虚树大小问题)

给定一棵 n 个点的树，q 次询问：给定一个非空点集，求其点对间路径边集之并的大小。

$1 \leqslant n, q, s \leqslant 10^6$，其中 s 为给定点集的大小和。

Problem (虚树大小问题)

给定一棵 n 个点的树，q 次询问：给定一个非空点集，求其点对间路径边集之并的大小。

$1 \leqslant n, q, s \leqslant 10^6$，其中 s 为给定点集的大小和。

Solution

问题等价于令询问给定点集的虚树中边权为其对应链的边数，求虚树边权和。

Problem (虚树大小问题)

给定一棵 n 个点的树，q 次询问：给定一个非空点集，求其点对间路径边集之并的大小。

$1 \leqslant n, q, s \leqslant 10^6$，其中 s 为给定点集的大小和。

Solution

问题等价于令询问给定点集的虚树中边权为其对应链的边数，求虚树边权和。

注意到虚树中所有叶子均在点集中，考虑欧拉序，虚树边权和为所有给定点在虚树中按 dfs 序排列后相邻两者与首尾两者的带权距离和的一半。

Solution

注意到虚树中的带权距离即为原树距离，且两者 dfs 序中相对位置相同，我们可以根据原树中的 dfs 序将给定点集排序得到 $u_1, \ldots, u_k$，答案即为 $\sum_{i=1}^{k} \mathrm{dist}(u_i, u_{i \bmod k+1})/2$。

Solution

注意到虚树中的带权距离即为原树距离，且两者 dfs 序中相对位置相同，我们可以根据原树中的 dfs 序将给定点集排序得到 $u_1, \ldots, u_k$，答案即为 $\sum_{i=1}^{k} \operatorname{dist}(u_i, u_{i \bmod k+1})/2$。

时间复杂度为 q 组长度和为 s 值域为 $[1, n] \cap \mathbb{N}$ 的数排序的时间复杂度加上 $\Theta(n)$，在线朴素实现可做到 $\Theta(n + s \log s)$，离线桶排可做到 $\Theta(n + s)$。

Problem (【ZJOI2019】语言)

给定一棵 n 个点的树与 m 条链 $\mathrm{path}(s_1,t_1),\ldots,\mathrm{path}(s_m,t_m)$，求 $\left|\left\{\{u,v\}\in([1,n]\cap\mathbb{N})^{(2)}\middle|\exists i\in[1,m]\cap\mathbb{N},u,v\in\mathrm{path}(s_i,t_i)\right\}\right|$。

$1\leqslant n,m\leqslant 10^5$。

Problem (【ZJOI2019】语言)

给定一棵 n 个点的树与 m 条链 $\mathrm{path}(s_1,t_1),\ldots,\mathrm{path}(s_m,t_m)$，求 $\left|\left\{\{u,v\}\in([1,n]\cap\mathbb{N})^{(2)}\middle|\exists i\in[1,m]\cap\mathbb{N},u,v\in\mathrm{path}(s_i,t_i)\right\}\right|$。

$1\leqslant n,m\leqslant 10^5$。

Solution

我们考虑将统计二元集改为统计相同条件的有序对，二元集数即有序对数减 n 后除以 2。

Problem (【ZJOI2019】语言)

给定一棵 n 个点的树与 m 条链 $\mathrm{path}(s_1,t_1),\ldots,\mathrm{path}(s_m,t_m)$，求 $\left|\left\{\{u,v\}\in([1,n]\cap\mathbb{N})^{(2)}\middle|\exists i\in[1,m]\cap\mathbb{N},u,v\in\mathrm{path}(s_i,t_i)\right\}\right|$。

$1\leqslant n,m\leqslant 10^5$。

Solution

我们考虑将统计二元集改为统计相同条件的有序对，二元集数即有序对数减 n 后除以 2。

我们考虑枚举有序对的第一个结点，另一个结点的方案数即经过该结点的链的点集并的大小。

Problem (【ZJOI2019】语言)

给定一棵 n 个点的树与 m 条链 $\mathrm{path}(s_1,t_1),\ldots,\mathrm{path}(s_m,t_m)$，求 $\left|\left\{\{u,v\}\in([1,n]\cap\mathbb{N})^{(2)}\middle|\exists i\in[1,m]\cap\mathbb{N},u,v\in\mathrm{path}(s_i,t_i)\right\}\right|$。

$1\leqslant n,m\leqslant 10^5$。

Solution

我们考虑将统计二元集改为统计相同条件的有序对，二元集数即有序对数减 n 后除以 2。

我们考虑枚举有序对的第一个结点，另一个结点的方案数即经过该结点的链的点集并的大小。

我们容易通过双向的包含关系说明，该点集并即为经过该结点链端点的虚树在原树对应的连通子树的点集。

Solution

注意到经过一个结点的链为经过其儿子的链除去以其儿子为最近公共祖先的链再并上以其为端点的链。

Solution

注意到经过一个结点的链为经过其儿子的链除去以其儿子为最近公共祖先的链再并上以其为端点的链。

而点集虚树在原树对应的连通子树点集大小为按 dfs 序排序后相邻点对与首尾点对距离和的一半。

Solution

注意到经过一个结点的链为经过其儿子的链除去以其儿子为最近公共祖先的链再并上以其为端点的链。

而点集虚树在原树对应的连通子树点集大小为按 dfs 序排序后相邻点对与首尾点对距离和的一半。

我们可以按 dfs 序通过线段树合并维护，每个结点维护对应 dfs 序区间内第一个与最后一个结点以及 dfs 区间内相邻点对距离和。(为减小常数也可维护左子树最后一个与右子树第一个的距离。)

Solution

注意到经过一个结点的链为经过其儿子的链除去以其儿子为最近公共祖先的链再并上以其为端点的链。

而点集虚树在原树对应的连通子树点集大小为按 dfs 序排序后相邻点对与首尾点对距离和的一半。

我们可以按 dfs 序通过线段树合并维护，每个结点维护对应 dfs 序区间内第一个与最后一个结点以及 dfs 区间内相邻点对距离和。(为减小常数也可维护左子树最后一个与右子树第一个的距离。)

时间复杂度 $\Theta(n + m \log n)$。

Solution

注意到经过一个结点的链为经过其儿子的链除去以其儿子为最近公共祖先的链再并上以其为端点的链。

而点集虚树在原树对应的连通子树点集大小为按 dfs 序排序后相邻点对与首尾点对距离和的一半。

我们可以按 dfs 序通过线段树合并维护，每个结点维护对应 dfs 序区间内第一个与最后一个结点以及 dfs 区间内相邻点对距离和。(为减小常数也可维护左子树最后一个与右子树第一个的距离。)

时间复杂度 $\Theta(n + m \log n)$。

根据钱哥的国家集训队论文，使用压位 Trie 合并可能有更小的时空常数。

Problem (【HNOI2014】世界树)

给定一棵 n 个点的树，q 次查询：给定非空点集 V，对于每个 $x \in V$ 求 $|\{u \in [1,n] \cap \mathbb{N} | \forall y \in V, \mathrm{dist}(x,u) + [x > y] \leqslant \mathrm{dist}(y,u)\}|$。

$1 \leqslant n, q, \sum m_i \leqslant 3 \times 10^5$，其中 m_i 为第 i 次给定点集的大小。

Problem (【HNOI2014】世界树)

给定一棵 n 个点的树，q 次查询：给定非空点集 V，对于每个 $x \in V$ 求 $|\{u \in [1,n] \cap \mathbb{N} | \forall y \in V, \mathrm{dist}(x,u) + [x > y] \leqslant \mathrm{dist}(y,u)\}|$。

$1 \leqslant n, q, \sum m_i \leqslant 3 \times 10^5$，其中 m_i 为第 i 次给定点集的大小。

Solution

我们不妨随意指定一个根，对每次查询，我们先建立给定点集在给定有根树上的虚树。

Problem (【HNOI2014】世界树)

给定一棵 n 个点的树，q 次查询：给定非空点集 V，对于每个 $x \in V$ 求 $|\{u \in [1,n] \cap \mathbb{N} | \forall y \in V, \mathrm{dist}(x,u) + [x > y] \leqslant \mathrm{dist}(y,u)\}|$。

$1 \leqslant n, q, \sum m_i \leqslant 3 \times 10^5$，其中 m_i 为第 i 次给定点集的大小。

Solution

我们不妨随意指定一个根，对每次查询，我们先建立给定点集在给定有根树上的虚树。

我们可以在虚树上 dp 求出距离其上每个点最近的给定点及该距离。

Solution

注意到每个给定点统计答案的集合在原树上为连通子图。

Solution

注意到每个给定点统计答案的集合在原树上为连通子图。

我们可以通过倍增、重链剖分加二分等方式在 $\Theta(\log n)$ 的时间复杂度内求出一个不包含给定点的簇中分别有多少结点距离最近的给定点与簇端点相同（求出临界点后子树大小作差）。

Solution

注意到每个给定点统计答案的集合在原树上为连通子图。

我们可以通过倍增、重链剖分加二分等方式在 $\Theta(\log n)$ 的时间复杂度内求出一个不包含给定点的簇中分别有多少结点距离最近的给定点与簇端点相同（求出临界点后子树大小作差）。

时间复杂度 $\Theta(n + \sum m_i \log n)$。

Property 2.1

一棵直径为 d 的树，存在结点 u 满足 $\forall(s,t\in V)\wedge(\mathrm{dist}(s,t)=d)$，$u\in\mathrm{path}(s,t)$。

Property 2.1

一棵直径为 d 的树，存在结点 u 满足 $\forall(s,t\in V)\wedge(\mathrm{dist}(s,t)=d)$，$u\in\mathrm{path}(s,t)$。

Proof

根据性质 1.1，只要我们证明 $\forall(s_1,t_1,s_2,t_2\in V)\wedge(\mathrm{dist}(s_1,t_1)=\mathrm{dist}(s_2,t_2)=d),\mathrm{path}(s_1,t_1)\cap\mathrm{path}(s_2,t_2)\neq\varnothing$ 即可。

Property 2.1

一棵直径为 d 的树，存在结点 u 满足 $\forall(s,t\in V)\wedge(\mathrm{dist}(s,t)=d)$，$u\in\mathrm{path}(s,t)$。

Proof

根据性质 1.1，只要我们证明 $\forall(s_1,t_1,s_2,t_2\in V)\wedge(\mathrm{dist}(s_1,t_1)=\mathrm{dist}(s_2,t_2)=d),\mathrm{path}(s_1,t_1)\cap\mathrm{path}(s_2,t_2)\neq\varnothing$ 即可。

考虑反证法，

Property 2.1

一棵直径为 d 的树，存在结点 u 满足 $\forall(s,t\in V)\wedge(\mathrm{dist}(s,t)=d)$，$u\in\mathrm{path}(s,t)$。

Proof

根据性质 1.1，只要我们证明 $\forall(s_1,t_1,s_2,t_2\in V)\wedge(\mathrm{dist}(s_1,t_1)=\mathrm{dist}(s_2,t_2)=d),\mathrm{path}(s_1,t_1)\cap\mathrm{path}(s_2,t_2)\neq\varnothing$ 即可。

考虑反证法，考虑 s_1,s_2,t_1,t_2 的虚树，此时 $\mathrm{path}(s_1,t_1)$ 与 $\mathrm{path}(s_2,t_2)$ 通过路径 $\mathrm{path}(u,v)$ 连接。

Property 2.1

一棵直径为 d 的树，存在结点 u 满足 $\forall(s,t\in V)\wedge(\mathrm{dist}(s,t)=d)$，$u\in\mathrm{path}(s,t)$。

Proof

根据性质 1.1，只要我们证明 $\forall(s_1,t_1,s_2,t_2\in V)\wedge(\mathrm{dist}(s_1,t_1)=\mathrm{dist}(s_2,t_2)=d),\mathrm{path}(s_1,t_1)\cap\mathrm{path}(s_2,t_2)\neq\varnothing$ 即可。

考虑反证法，考虑 s_1,s_2,t_1,t_2 的虚树，此时 $\mathrm{path}(s_1,t_1)$ 与 $\mathrm{path}(s_2,t_2)$ 通过路径 $\mathrm{path}(u,v)$ 连接。

注意到我们选取 s_1,t_1 中距离 u 较远者与 s_2,t_2 中距离 v 较远者作为端点的路径长度 $\geqslant\frac{d}{2}+\mathrm{dist}(u,v)+\frac{d}{2}>d$，产生矛盾。

Property 2.1

一棵直径为 d 的树，存在结点 u 满足 $\forall(s,t\in V)\wedge(\mathrm{dist}(s,t)=d)$，$u\in\mathrm{path}(s,t)$。

Proof

根据性质 1.1，只要我们证明 $\forall(s_1,t_1,s_2,t_2\in V)\wedge(\mathrm{dist}(s_1,t_1)=\mathrm{dist}(s_2,t_2)=d),\mathrm{path}(s_1,t_1)\cap\mathrm{path}(s_2,t_2)\neq\varnothing$ 即可。

考虑反证法，考虑 s_1,s_2,t_1,t_2 的虚树，此时 $\mathrm{path}(s_1,t_1)$ 与 $\mathrm{path}(s_2,t_2)$ 通过路径 $\mathrm{path}(u,v)$ 连接。

注意到我们选取 s_1,t_1 中距离 u 较远者与 s_2,t_2 中距离 v 较远者作为端点的路径长度 $\geqslant\frac{d}{2}+\mathrm{dist}(u,v)+\frac{d}{2}>d$，产生矛盾。

直径的一些其它很多性质也可以类似地通过反证法加调整法证明。

我们也可以通过细致的讨论得到下述性质：

我们也可以通过细致的讨论得到下述性质：

Property 2.2

一棵直径为 d 的树，其中结点 s, t 满足 $\mathrm{dist}(s,t) = d$，则树的中心为 $\mathrm{path}(s,t)$ 中距离 s 或 t 为 $\lfloor d/2 \rfloor$ 的一个或两个结点，树的半径为 $\lceil d/2 \rceil$。

我们也可以通过细致的讨论得到下述性质：

Property 2.2

一棵直径为 d 的树，其中结点 s, t 满足 $\mathrm{dist}(s,t) = d$，则树的中心为 $\mathrm{path}(s,t)$ 中距离 s 或 t 为 $\lfloor d/2 \rfloor$ 的一个或两个结点，树的半径为 $\lceil d/2 \rceil$。

Property 2.3

一棵直径为 d 的树，c 为其中心，其中结点 s, t 满足 $\mathrm{dist}(s,t) = d$，则 c 在 $\mathrm{path}(s,t)$ 中。

我们也可以通过细致的讨论得到下述性质：

Property 2.2

一棵直径为 d 的树，其中结点 s, t 满足 $\mathrm{dist}(s, t) = d$，则树的中心为 $\mathrm{path}(s, t)$ 中距离 s 或 t 为 $\lfloor d/2 \rfloor$ 的一个或两个结点，树的半径为 $\lceil d/2 \rceil$。

Property 2.3

一棵直径为 d 的树，c 为其中心，其中结点 s, t 满足 $\mathrm{dist}(s, t) = d$，则 c 在 $\mathrm{path}(s, t)$ 中。

对于边带非负权的情况，中心为与 s, t 距离最大值最小的结点，半径为该距离，其余结论与证明不变。

Problem (树的直径)

给定一棵 n 个结点的无根树，求所有路径中长度最大的任意一条。要求时间复杂度 $O(n)$。

Problem (树的直径)

给定一棵 n 个结点的无根树，求所有路径中长度最大的任意一条。要求时间复杂度 $O(n)$。

Solution

我们可以通过 dfs 在 $\Theta(n)$ 的时间复杂度内求出距离一个结点最远的结点。

Problem (树的直径)

给定一棵 n 个结点的无根树，求所有路径中长度最大的任意一条。要求时间复杂度 $O(n)$。

Solution

我们可以通过 dfs 在 $\Theta(n)$ 的时间复杂度内求出距离一个结点最远的结点。

我们求出距离任意一个结点最远的结点，其一定可以作为答案路径的一个端点。

Problem (树的直径)

给定一棵 n 个结点的无根树，求所有路径中长度最大的任意一条。要求时间复杂度 $O(n)$。

Solution

我们可以通过 dfs 在 $\Theta(n)$ 的时间复杂度内求出距离一个结点最远的结点。

我们求出距离任意一个结点最远的结点，其一定可以作为答案路径的一个端点。

再求出距离其最远的结点即可。

Problem (树的直径)

给定一棵 n 个结点的无根树，求所有路径中长度最大的任意一条。

要求时间复杂度 $O(n)$。

Solution

我们可以通过 dfs 在 $\Theta(n)$ 的时间复杂度内求出距离一个结点最远的结点。

我们求出距离任意一个结点最远的结点，其一定可以作为答案路径的一个端点。

再求出距离其最远的结点即可。

时间复杂度 $\Theta(n)$。

Problem (Anton and Tree)

给定一棵 n 个结点的树，每个点给定黑或白色，一次操作可以选择任一同色连通块使其反色，求使整棵树颜色相同的最少操作次数。

$1 \leqslant n \leqslant 2 \times 10^5$。

Problem (Anton and Tree)

给定一棵 n 个结点的树，每个点给定黑或白色，一次操作可以选择任一同色连通块使其反色，求使整棵树颜色相同的最少操作次数。

$1 \leqslant n \leqslant 2 \times 10^5$。

Solution

注意到同色连通块在后续操作中一定同时改变颜色，我们可以使用并查集等方式缩点使得树任一相邻结点对颜色不同。

Problem (Anton and Tree)

给定一棵 n 个结点的树，每个点给定黑或白色，一次操作可以选择任一同色连通块使其反色，求使整棵树颜色相同的最少操作次数。

$1 \leqslant n \leqslant 2 \times 10^5$。

Solution

注意到同色连通块在后续操作中一定同时改变颜色，我们可以使用并查集等方式缩点使得树任一相邻结点对颜色不同。

注意到答案一定不低于直径的一半。

Problem (Anton and Tree)

给定一棵 n 个结点的树，每个点给定黑或白色，一次操作可以选择任一同色连通块使其反色，求使整棵树颜色相同的最少操作次数。

$1 \leqslant n \leqslant 2 \times 10^5$。

Solution

注意到同色连通块在后续操作中一定同时改变颜色，我们可以使用并查集等方式缩点使得树任一相邻结点对颜色不同。

注意到答案一定不低于直径的一半。

注意到我们一定可以通过适当的选择（中点或中心边的任意端点的所在同色连通块）在一次操作内使直径除以二上取整降低。

Problem (Anton and Tree)

给定一棵 n 个结点的树，每个点给定黑或白色，一次操作可以选择任一同色连通块使其反色，求使整棵树颜色相同的最少操作次数。

$1 \leqslant n \leqslant 2 \times 10^5$。

Solution

注意到同色连通块在后续操作中一定同时改变颜色，我们可以使用并查集等方式缩点使得树任一相邻结点对颜色不同。

注意到答案一定不低于直径的一半。

注意到我们一定可以通过适当的选择（中点或中心边的任意端点的所在同色连通块）在一次操作内使直径除以二上取整降低。

因此答案即直径除以二上取整。

Problem (Anton and Tree)

给定一棵 n 个结点的树，每个点给定黑或白色，一次操作可以选择任一同色连通块使其反色，求使整棵树颜色相同的最少操作次数。

$1 \leqslant n \leqslant 2 \times 10^5$。

Solution

注意到同色连通块在后续操作中一定同时改变颜色，我们可以使用并查集等方式缩点使得树任一相邻结点对颜色不同。

注意到答案一定不低于直径的一半。

注意到我们一定可以通过适当的选择（中点或中心边的任意端点的所在同色连通块）在一次操作内使直径除以二上取整降低。

因此答案即直径除以二上取整。

时间复杂度 $\Theta(n)$。

Problem (Tree Destruction)

给你一棵 n 个结点的树，$n-1$ 次操作：选择两个叶子，获得其间的距离的收益，然后将其中一个点删除，求最大收益及其方案。

$2 \leqslant n \leqslant 2 \times 10^5$。

Problem (Tree Destruction)

给你一棵 n 个结点的树，$n-1$ 次操作：选择两个叶子，获得其间的距离的收益，然后将其中一个点删除，求最大收益及其方案。

$2 \leqslant n \leqslant 2 \times 10^5$。

Solution

容易将问题转化为：选择一个叶子删去，获得当前树中以其为端点的最远距离的收益。

Problem (Tree Destruction)

给你一棵 n 个结点的树，$n-1$ 次操作：选择两个叶子，获得其间的距离的收益，然后将其中一个点删除，求最大收益及其方案。

$2 \leqslant n \leqslant 2 \times 10^5$。

Solution

容易将问题转化为：选择一个叶子删去，获得当前树中以其为端点的最远距离的收益。

注意到在进行若干操作后删除一个未删除结点时产生的最大收益即当前树中以其为端点的最远距离。

Problem (Tree Destruction)

给你一棵 n 个结点的树，$n-1$ 次操作：选择两个叶子，获得其间的距离的收益，然后将其中一个点删除，求最大收益及其方案。

$2 \leqslant n \leqslant 2 \times 10^5$。

Solution

容易将问题转化为：选择一个叶子删去，获得当前树中以其为端点的最远距离的收益。

注意到在进行若干操作后删除一个未删除结点时产生的最大收益即当前树中以其为端点的最远距离。

注意到在直径从 d 变为 $d-1$ 时至少有 $\lceil d/2 \rceil$ 个结点最大收益减小 1，且最后直径一定为 0。

Solution

注意到我们可以在求出一条长度为直径的路径后先删除路径外结点再删除该路径取到该上界。

Solution

注意到我们可以在求出一条长度为直径的路径后先删除路径外结点再删除该路径取到该上界。

时间复杂度 $\Theta(n)$。

Problem (【CTSC2017】网络)

给定一棵 n 个结点的树，边给定非负权，选择两个点将一条以它们为顶点 L 为边权的边加入边集，求图的最小直径。

$1 \leqslant n \leqslant 10^5$。

Problem (【CTSC2017】网络)

给定一棵 n 个结点的树，边给定非负权，选择两个点将一条以它们为顶点 L 为边权的边加入边集，求图的最小直径。

$1 \leqslant n \leqslant 10^5$。

我们先考虑一种特殊情况。

Problem (【CTSC2017】网络)

给定一棵 n 个结点的树，边给定非负权，选择两个点将一条以它们为顶点 L 为边权的边加入边集，求图的最小直径。

$1 \leqslant n \leqslant 10^5$。

我们先考虑一种特殊情况。

Problem (【IOI2016】shortcut)

给定一条 n 个点的链，边给定非负权，对每个链上的点给定长度，新建一个点并连一条该长度的边。选择两个初始链上的点将一条以它们为顶点 c 为长度的边加入边集，求图的最小直径。

$1 \leqslant n \leqslant 10^6$。

Solution (【IOI2016】shortcut)

首先求出链上点到第一个点的距离 L_i。

Solution (【IOI2016】shortcut)

首先求出链上点到第一个点的距离 L_i。

对两个点 $i < j$ 下方连的点，初始的简单路径长度为 $L_j - L_i + d_i + d_j$。

Solution (【IOI2016】shortcut)

首先求出链上点到第一个点的距离 L_i。

对两个点 $i<j$ 下方连的点，初始的简单路径长度为 $L_j-L_i+d_i+d_j$。

如果加入的边以 $a<b$ 为顶点，那么经过这条边的简单路径长度为 $|L_i-L_a|+|L_j-L_b|+d_i+d_j+c$。

Solution (【IOI2016】shortcut)

首先求出链上点到第一个点的距离 L_i。

对两个点 $i<j$ 下方连的点，初始的简单路径长度为 $L_j-L_i+d_i+d_j$。

如果加入的边以 $a<b$ 为顶点，那么经过这条边的简单路径长度为 $|L_i-L_a|+|L_j-L_b|+d_i+d_j+c$。

考虑二分答案，

Solution (【IOI2016】shortcut)

首先求出链上点到第一个点的距离 L_i。

对两个点 $i<j$ 下方连的点，初始的简单路径长度为 $L_j-L_i+d_i+d_j$。

如果加入的边以 $a<b$ 为顶点，那么经过这条边的简单路径长度为 $|L_i-L_a|+|L_j-L_b|+d_i+d_j+c$。

考虑二分答案，若答案 $\leqslant M$，那么对于满足 $L_j-L_i+d_i+d_j>M$ 时都应有 $|L_i-L_a|+|L_j-L_b|+d_i+d_j+c\leqslant M$。

Solution (【IOI2016】shortcut)

首先求出链上点到第一个点的距离 L_i。

对两个点 $i<j$ 下方连的点，初始的简单路径长度为 $L_j-L_i+d_i+d_j$。

如果加入的边以 $a<b$ 为顶点，那么经过这条边的简单路径长度为 $|L_i-L_a|+|L_j-L_b|+d_i+d_j+c$。

考虑二分答案，若答案 $\leqslant M$，那么对于满足 $L_j-L_i+d_i+d_j>M$ 时都应有 $|L_i-L_a|+|L_j-L_b|+d_i+d_j+c\leqslant M$。

注意到绝对值不超过一个数可以将所有的情况分别求最小值再求与根据 L_a+L_b 与 L_a-L_b 的取值范围判断是否存在 a,b 的解。

Solution (【IOI2016】shortcut)

首先求出链上点到第一个点的距离 L_i。

对两个点 $i<j$ 下方连的点，初始的简单路径长度为 $L_j-L_i+d_i+d_j$。

如果加入的边以 $a<b$ 为顶点，那么经过这条边的简单路径长度为 $|L_i-L_a|+|L_j-L_b|+d_i+d_j+c$。

考虑二分答案，若答案 $\leqslant M$，那么对于满足 $L_j-L_i+d_i+d_j>M$ 时都应有 $|L_i-L_a|+|L_j-L_b|+d_i+d_j+c\leqslant M$。

注意到绝对值不超过一个数可以将所有的情况分别求最小值再求与根据 L_a+L_b 与 L_a-L_b 的取值范围判断是否存在 a,b 的解。

以 $L_i-L_a+L_j-L_b+d_i+d_j+c\leqslant M$ 为例，整理可得 $(L_i+d_i)+(L_j+d_j)-(M-c)\leqslant L_a+L_b$。

Solution (【IOI2016】shortcut)

考虑枚举 j,

Solution (【IOI2016】shortcut)

考虑枚举 j，对于 $(d_i - L_i) + (d_j \pm L_j) - (M - c) \leqslant -L_a \pm L_b$，我们维护 $\max_{i=1}^{j-1}(d_i - L_i)$，显然若其不满足 $L_j - L_i + d_i + d_j > M$ 则不存在 i 满足。$(d_i + L_i) + (d_j + L_j) - (M - c) \leqslant L_a + L_b$ 可以类似枚举 i 维护 $\max_{j=i+1}^{n}(d_j + L_j)$ 解决。

Solution (【IOI2016】shortcut)

考虑枚举 j，对于 $(d_i - L_i) + (d_j \pm L_j) - (M - c) \leqslant -L_a \pm L_b$，我们维护 $\max_{i=1}^{j-1}(d_i - L_i)$，显然若其不满足 $L_j - L_i + d_i + d_j > M$ 则不存在 i 满足。$(d_i + L_i) + (d_j + L_j) - (M - c) \leqslant L_a + L_b$ 可以类似枚举 i 维护 $\max_{j=i+1}^{n}(d_j + L_j)$ 解决。

因此我们只需考虑 $(d_i + L_i) + (d_j - L_j) - (M - c) \leqslant L_a - L_b$。

Solution (【IOI2016】shortcut)

考虑枚举 j，对于 $(d_i - L_i) + (d_j \pm L_j) - (M - c) \leqslant -L_a \pm L_b$，我们维护 $\max_{i=1}^{j-1}(d_i - L_i)$，显然若其不满足 $L_j - L_i + d_i + d_j > M$ 则不存在 i 满足。$(d_i + L_i) + (d_j + L_j) - (M - c) \leqslant L_a + L_b$ 可以类似枚举 i 维护 $\max_{j=i+1}^{n}(d_j + L_j)$ 解决。

因此我们只需考虑 $(d_i + L_i) + (d_j - L_j) - (M - c) \leqslant L_a - L_b$。

依然枚举 j，

Solution (【IOI2016】shortcut)

考虑枚举 j，对于 $(d_i - L_i) + (d_j \pm L_j) - (M - c) \leqslant -L_a \pm L_b$，我们维护 $\max_{i=1}^{j-1}(d_i - L_i)$，显然若其不满足 $L_j - L_i + d_i + d_j > M$ 则不存在 i 满足。$(d_i + L_i) + (d_j + L_j) - (M - c) \leqslant L_a + L_b$ 可以类似枚举 i 维护 $\max_{j=i+1}^{n}(d_j + L_j)$ 解决。

因此我们只需考虑 $(d_i + L_i) + (d_j - L_j) - (M - c) \leqslant L_a - L_b$。

依然枚举 j，那么 i 的范围一定是将 $L_i - d_i$ 排序后 $L_i - d_i < L_j + d_j - M$ 的前缀中 $< j$ 的，所以显然可以线段树优化。但由于 $L_i + d_i$ 和 $L_i - d_i$ 都不是有序的所以似乎不能利用单调性优化。

Solution (【IOI2016】shortcut)

考虑枚举 j，对于 $(d_i - L_i) + (d_j \pm L_j) - (M - c) \leqslant -L_a \pm L_b$，我们维护 $\max_{i=1}^{j-1}(d_i - L_i)$，显然若其不满足 $L_j - L_i + d_i + d_j > M$ 则不存在 i 满足。$(d_i + L_i) + (d_j + L_j) - (M - c) \leqslant L_a + L_b$ 可以类似枚举 i 维护 $\max_{j=i+1}^{n}(d_j + L_j)$ 解决。

因此我们只需考虑 $(d_i + L_i) + (d_j - L_j) - (M - c) \leqslant L_a - L_b$。

依然枚举 j，那么 i 的范围一定是将 $L_i - d_i$ 排序后 $L_i - d_i < L_j + d_j - M$ 的前缀中 $< j$ 的，所以显然可以线段树优化。但由于 $L_i + d_i$ 和 $L_i - d_i$ 都不是有序的所以似乎不能利用单调性优化。

实际上，如果两个点 $i < j$ 满足 $L_i - d_i > L_j - d_j$，那么由 $L_i < L_j$ 得出 $-L_i - d_i > -L_j - d_j$，从而对于 $> j$ 的作为右端点，j 作为左端点的不等式约束强于 i 作为左端点的约束。

Solution (【IOI2016】shortcut)

因此我们可以使用单调栈，保持从栈底至栈顶 $L_i - d_i$ 单调递增 $L_i + d_i$ 单调递增，枚举至 j 时将 $L_i - d_i > L_j - d_j$ 的元素全部从栈中弹出，查询栈顶后将 j 压入栈。

Solution (【IOI2016】shortcut)

因此我们可以使用单调栈，保持从栈底至栈顶 $L_i - d_i$ 单调递增 $L_i + d_i$ 单调递增，枚举至 j 时将 $L_i - d_i > L_j - d_j$ 的元素全部从栈中弹出，查询栈顶后将 j 压入栈。

这样就做到了 $\Theta(n)$ 判断，总时间复杂度为 $\Theta(n \log(nV))$。

Solution (【IOI2016】shortcut)

因此我们可以使用单调栈，保持从栈底至栈顶 $L_i - d_i$ 单调递增 $L_i + d_i$ 单调递增，枚举至 j 时将 $L_i - d_i > L_j - d_j$ 的元素全部从栈中弹出，查询栈顶后将 j 压入栈。

这样就做到了 $\Theta(n)$ 判断，总时间复杂度为 $\Theta(n \log(nV))$。

我们考虑回到【CTSC2017】网络。

Solution (【IOI2016】shortcut)

因此我们可以使用单调栈，保持从栈底至栈顶 $L_i - d_i$ 单调递增 $L_i + d_i$ 单调递增，枚举至 j 时将 $L_i - d_i > L_j - d_j$ 的元素全部从栈中弹出，查询栈顶后将 j 压入栈。

这样就做到了 $\Theta(n)$ 判断，总时间复杂度为 $\Theta(n \log(nV))$。

我们考虑回到【CTSC2017】网络。

Solution (【CTSC2017】网络)

我们可以先求出长度为当前直径的一条路径，可以证明一定在其上加边最优（后述），

Solution (【IOI2016】shortcut)

因此我们可以使用单调栈，保持从栈底至栈顶 $L_i - d_i$ 单调递增 $L_i + d_i$ 单调递增，枚举至 j 时将 $L_i - d_i > L_j - d_j$ 的元素全部从栈中弹出，查询栈顶后将 j 压入栈。

这样就做到了 $\Theta(n)$ 判断，总时间复杂度为 $\Theta(n \log(nV))$。

我们考虑回到【CTSC2017】网络。

Solution (【CTSC2017】网络)

我们可以先求出长度为当前直径的一条路径，可以证明一定在其上加边最优（后述），因此求出该路径上每个点不经过路径上其它点可达的最远距离，转化为【IOI2016】shortcut。

Solution (【IOI2016】shortcut)

因此我们可以使用单调栈，保持从栈底至栈顶 $L_i - d_i$ 单调递增 $L_i + d_i$ 单调递增，枚举至 j 时将 $L_i - d_i > L_j - d_j$ 的元素全部从栈中弹出，查询栈顶后将 j 压入栈。

这样就做到了 $\Theta(n)$ 判断，总时间复杂度为 $\Theta(n \log(nV))$。

我们考虑回到【CTSC2017】网络。

Solution (【CTSC2017】网络)

我们可以先求出长度为当前直径的一条路径，可以证明一定在其上加边最优（后述），因此求出该路径上每个点不经过路径上其它点可达的最远距离，转化为【IOI2016】shortcut。

时间复杂度 $\Theta(n \log(nV))$。

Proof

我们设 dist 仍表示原树距离，最优方案在 u 与 v 之间加边，结点 S, T 间距离为直径，对任意树上结点 X，$f_X = \mathrm{minarg}_{u \in \mathrm{path}(S,T)} \mathrm{dist}(u, X)$。不妨设 $\mathrm{dist}(f_u, S) \leqslant \mathrm{dist}(f_v, S)$。

Proof

我们设 dist 仍表示原树距离，最优方案在 u 与 v 之间加边，结点 S, T 间距离为直径，对任意树上结点 X，$f_X = \mathrm{minarg}_{u \in \mathrm{path}(S,T)} \mathrm{dist}(u, X)$。不妨设 $\mathrm{dist}(f_u, S) \leqslant \mathrm{dist}(f_v, S)$。

我们考虑另一种方案（下称方案二）：在 f_u 和 f_v 上增加长度为 L 的边，我们接下来说明该方案不劣于在 u, v 间加边。

Proof

我们设 dist 仍表示原树距离，最优方案在 u 与 v 之间加边，结点 S, T 间距离为直径，对任意树上结点 X，$f_X = \text{minarg}_{u \in \text{path}(S,T)} \text{dist}(u, X)$。不妨设 $\text{dist}(f_u, S) \leqslant \text{dist}(f_v, S)$。

我们考虑另一种方案（下称方案二）：在 f_u 和 f_v 上增加长度为 L 的边，我们接下来说明该方案不劣于在 u, v 间加边。

设 a, b 为方案二距离最远的点对之一，若其中 f_a, f_b 均不在 $\{f_u, f_v\}$ 中，则当前方案 a, b 间距离一定不低于方案二，因此不妨设 $f_a = f_u$。

Proof

若 f_u 在 $\mathrm{path}(b,T)$ 上则 a,b 在方案二距离为 $\mathrm{dist}(a,b)$，因此不妨将 b 调整至 S 距离不降。若当前方案 a,S 距离为 $\mathrm{dist}(a,S)$ 则当前方案不优于方案二，否则 $\mathrm{dist}(u,f_u)>L+\mathrm{dist}(v,f_v)+\mathrm{dist}(f_u,f_v)$，此时当前方案 f_u,f_v 距离为 $\mathrm{dist}(f_u,f_v)$，S,T 距离为 $\mathrm{dist}(S,T)\geqslant\mathrm{dist}(a,S)$，同样不优于方案二。

Proof

若 f_u 在 $\mathrm{path}(b,T)$ 上则 a,b 在方案二距离为 $\mathrm{dist}(a,b)$，因此不妨将 b 调整至 S 距离不降。若当前方案 a,S 距离为 $\mathrm{dist}(a,S)$ 则当前方案不优于方案二，否则 $\mathrm{dist}(u,f_u)>L+\mathrm{dist}(v,f_v)+\mathrm{dist}(f_u,f_v)$，此时当前方案 f_u,f_v 距离为 $\mathrm{dist}(f_u,f_v)$，S,T 距离为 $\mathrm{dist}(S,T)\geqslant\mathrm{dist}(a,S)$，同样不优于方案二。

若 f_u 不在 $\mathrm{path}(b,T)$ 上则方案二中 a,b 距离不超过 S,b 距离，若 $f_b=f_v$ 再次调整为 S,T，转化为 f_a,f_b 均不在 $\{f_u,f_v\}$ 中。

Proof

若 f_u 在 $\mathrm{path}(b,T)$ 上则 a,b 在方案二距离为 $\mathrm{dist}(a,b)$，因此不妨将 b 调整至 S 距离不降。若当前方案 a,S 距离为 $\mathrm{dist}(a,S)$ 则当前方案不优于方案二，否则 $\mathrm{dist}(u,f_u)>L+\mathrm{dist}(v,f_v)+\mathrm{dist}(f_u,f_v)$，此时当前方案 f_u,f_v 距离为 $\mathrm{dist}(f_u,f_v)$，S,T 距离为 $\mathrm{dist}(S,T)\geqslant\mathrm{dist}(a,S)$，同样不优于方案二。

若 f_u 不在 $\mathrm{path}(b,T)$ 上则方案二中 a,b 距离不超过 S,b 距离，若 $f_b=f_v$ 再次调整为 S,T，转化为 f_a,f_b 均不在 $\{f_u,f_v\}$ 中。

事实上只要对于 S,T,u,v,a,b 的虚树结论成立即可，也可以使用程序验证所有情况。

Problem (【ICPC2023 杭州】F. Top Cluster)

给定一棵 n 个点的无根带权树，保证点权 a 互不相同，q 次询问：给定其中结点 u，$k \in \mathbb{N}$，求 $\operatorname{mex}\{a_v | v \in [1,n] \cap \mathbb{N}, \operatorname{dist}(u,v) \leqslant k\}$。

$1 \leqslant n, q \leqslant 5 \times 10^5$。

Problem (【ICPC2023 杭州】F. Top Cluster)

给定一棵 n 个点的无根带权树，保证点权 a 互不相同，q 次询问：给定其中结点 u，$k \in \mathbb{N}$，求 $\operatorname{mex}\{a_v | v \in [1,n] \cap \mathbb{N}, \operatorname{dist}(u,v) \leqslant k\}$。

$1 \leqslant n, q \leqslant 5 \times 10^5$。

Solution

注意到点权互不相同，我们可以将问题转化为不在点权中出现或距离 u 超过 k 的最小自然数。

Problem (【ICPC2023 杭州】F. Top Cluster)

给定一棵 n 个点的无根带权树，保证点权 a 互不相同，q 次询问：给定其中结点 u，$k \in \mathbb{N}$，求 $\operatorname{mex}\{a_v | v \in [1,n] \cap \mathbb{N}, \operatorname{dist}(u,v) \leqslant k\}$。

$1 \leqslant n, q \leqslant 5 \times 10^5$。

Solution

注意到点权互不相同，我们可以将问题转化为不在点权中出现或距离 u 超过 k 的最小自然数。

我们考虑进行预处理，使得每次询问可以二分答案判断是否存在不超过 x 的自然数不在点权中出现或距离 u 超过 k。

Problem (【ICPC2023 杭州】F. Top Cluster)

给定一棵 n 个点的无根带权树，保证点权 a 互不相同，q 次询问：给定其中结点 u，$k \in \mathbb{N}$，求 $\operatorname{mex}\{a_v | v \in [1,n] \cap \mathbb{N}, \operatorname{dist}(u,v) \leqslant k\}$。

$1 \leqslant n, q \leqslant 5 \times 10^5$。

Solution

注意到点权互不相同，我们可以将问题转化为不在点权中出现或距离 u 超过 k 的最小自然数。

我们考虑进行预处理，使得每次询问可以二分答案判断是否存在不超过 x 的自然数不在点权中出现或距离 u 超过 k。

不在点权中出现是容易处理的，我们考虑对 $x \in [0,n] \cap \mathbb{N}$ 维护点权不超过 x 的结点的集合，支持查询是否存在距离 u 超过 k 的点。

Solution

维护这些集合长度为虚树对应连通子图直径的一条路径即可，其中一个端点一定是距离原树任意点最远的集合中点。

Solution

维护这些集合长度为虚树对应连通子图直径的一条路径即可，其中一个端点一定是距离原树任意点最远的集合中点。

时间复杂度为 $\Theta(n+q\log n)$ 加上 n 个结点的树 $\Theta(n+q\log n)$ 次查询两点距离的时间复杂度。

我们定义一棵树的重心为 $\sum_{v\in V(T)} \mathrm{dist}(u,v)$ 最小的 $u \in V(T)$。

我们定义一棵树的重心为 $\sum_{v\in V(T)} \mathrm{dist}(u,v)$ 最小的 $u \in V(T)$。

Property 3.1

结点 c 是树 T 的重心与以下结论等价：

我们定义一棵树的重心为 $\sum_{v\in V(T)} \mathrm{dist}(u,v)$ 最小的 $u \in V(T)$。

Property 3.1

结点 c 是树 T 的重心与以下结论等价：

1. 在 T 的所有结点中，T 删去 c 后形成的连通块大小的最大值最小。

我们定义一棵树的重心为 $\sum_{v\in V(T)} \mathrm{dist}(u,v)$ 最小的 $u \in V(T)$。

Property 3.1

结点 c 是树 T 的重心与以下结论等价：

1. 在 T 的所有结点中，T 删去 c 后形成的连通块大小的最大值最小。
2. T 删去 c 后形成的连通块大小均不超过 $|V(T)|/2$。

我们定义一棵树的重心为 $\sum_{v\in V(T)} \mathrm{dist}(u,v)$ 最小的 $u \in V(T)$。

Property 3.1

结点 c 是树 T 的重心与以下结论等价：

1. 在 T 的所有结点中，T 删去 c 后形成的连通块大小的最大值最小。
2. T 删去 c 后形成的连通块大小均不超过 $|V(T)|/2$。

容易用调整法说明上述性质的正确性。

我们定义一棵树的重心为 $\sum_{v\in V(T)} \mathrm{dist}(u,v)$ 最小的 $u \in V(T)$。

Property 3.1

结点 c 是树 T 的重心与以下结论等价：

1. 在 T 的所有结点中，T 删去 c 后形成的连通块大小的最大值最小。
2. T 删去 c 后形成的连通块大小均不超过 $|V(T)|/2$。

容易用调整法说明上述性质的正确性。

Property 3.2

一棵树最多有两个重心，若有两个重心则一定有一条边以它们为端点，且此时删去这条边后两个连通块大小相同。

Property 3.3

将两棵树通过一条边连接形成的树的重心在原先两棵树重心的路径上。

Property 3.3

将两棵树通过一条边连接形成的树的重心在原先两棵树重心的路径上。

Property 3.4

一棵树加入或者删除一个叶子后树的重心与原先树的重心距离不超过 1。

Problem (树的重心)

给定一棵 n 个结点的树，求其重心。
要求时间复杂度 $O(n)$。

Problem (树的重心)

给定一棵 n 个结点的树，求其重心。
要求时间复杂度 $O(n)$。

Solution

我们可以使用删去 c 后形成的连通块大小均不超过 $n/2$ 判定结点 c 是否为重心。

Problem (树的重心)

给定一棵 n 个结点的树，求其重心。
要求时间复杂度 $O(n)$。

Solution

我们可以使用删去 c 后形成的连通块大小均不超过 $n/2$ 判定结点 c 是否为重心。

等价地，我们任意指定一个根求出子树大小后若 $\forall v \in \mathrm{son}(u), |\mathrm{subtree}(v)| \leqslant n/2$ 且 $|\mathrm{subtree}(u)| \geqslant n/2$ 则 u 是树的重心。

Problem (树的重心)

给定一棵 n 个结点的树，求其重心。
要求时间复杂度 $O(n)$。

Solution

我们可以使用删去 c 后形成的连通块大小均不超过 $n/2$ 判定结点 c 是否为重心。

等价地，我们任意指定一个根求出子树大小后若 $\forall v \in \mathrm{son}(u), |\mathrm{subtree}(v)| \leqslant n/2$ 且 $|\mathrm{subtree}(u)| \geqslant n/2$ 则 u 是树的重心。

时间复杂度 $\Theta(n)$。

Problem (【USACO10MAR】Great Cow Gathering G)

给定一棵 n 个结点的树，点集为 V，边给定非负权，对 $u \in V$ 给定非负权 a_u，令 $f(u) = \sum_{v \in V} a_v \operatorname{dist}(u, v)$，求 $\min_{u \in V} f(u)$。

$1 \leqslant n \leqslant 5 \times 10^5$。

满足 $f(u)$ 最小的结点 u 也常被称为给定带权树的重心。

Problem (【USACO10MAR】Great Cow Gathering G)

给定一棵 n 个结点的树，点集为 V，边给定非负权，对 $u \in V$ 给定非负权 a_u，令 $f(u) = \sum_{v \in V} a_v \operatorname{dist}(u, v)$，求 $\min_{u \in V} f(u)$。

$1 \leqslant n \leqslant 5 \times 10^5$。

满足 $f(u)$ 最小的结点 u 也常被称为给定带权树的重心。

Solution

我们可以类似通过调整法说明：若边权为正，则结点 u 为树的重心当且仅当删除 u 后每个连通块的点权和不超过原树点权和的一半。

Problem (【USACO10MAR】Great Cow Gathering G)

给定一棵 n 个结点的树，点集为 V，边给定非负权，对 $u \in V$ 给定非负权 a_u，令 $f(u) = \sum_{v \in V} a_v \operatorname{dist}(u, v)$，求 $\min_{u \in V} f(u)$。

$1 \leqslant n \leqslant 5 \times 10^5$。

满足 $f(u)$ 最小的结点 u 也常被称为给定带权树的重心。

Solution

我们可以类似通过调整法说明：若边权为正，则结点 u 为树的重心当且仅当删除 u 后每个连通块的点权和不超过原树点权和的一半。

我们可以直接将边权为 0 的边构成的连通块缩为一个点求出所有重心，也可以注意到将边权全部改为 1 后树的重心一定是原树的重心。

Problem (【USACO10MAR】Great Cow Gathering G)

给定一棵 n 个结点的树，点集为 V，边给定非负权，对 $u \in V$ 给定非负权 a_u，令 $f(u) = \sum_{v \in V} a_v \operatorname{dist}(u, v)$，求 $\min_{u \in V} f(u)$。

$1 \leqslant n \leqslant 5 \times 10^5$。

满足 $f(u)$ 最小的结点 u 也常被称为给定带权树的重心。

Solution

我们可以类似通过调整法说明：若边权为正，则结点 u 为树的重心当且仅当删除 u 后每个连通块的点权和不超过原树点权和的一半。

我们可以直接将边权为 0 的边构成的连通块缩为一个点求出所有重心，也可以注意到将边权全部改为 1 后树的重心一定是原树的重心。

与无权树的重心类似，将子树大小定义为点权和即可。

Problem (【USACO10MAR】Great Cow Gathering G)

给定一棵 n 个结点的树，点集为 V，边给定非负权，对 $u \in V$ 给定非负权 a_u，令 $f(u) = \sum_{v \in V} a_v \operatorname{dist}(u, v)$，求 $\min_{u \in V} f(u)$。

$1 \leqslant n \leqslant 5 \times 10^5$。

满足 $f(u)$ 最小的结点 u 也常被称为给定带权树的重心。

Solution

我们可以类似通过调整法说明：若边权为正，则结点 u 为树的重心当且仅当删除 u 后每个连通块的点权和不超过原树点权和的一半。

我们可以直接将边权为 0 的边构成的连通块缩为一个点求出所有重心，也可以注意到将边权全部改为 1 后树的重心一定是原树的重心。

与无权树的重心类似，将子树大小定义为点权和即可。

时间复杂度 $\Theta(n)$。

Property 4.1

对于一棵有根树及结点序列 $u_1, u_2, \ldots, u_k$，有性质 $\mathrm{lca}\{u_1, \ldots, u_k\}$ 为 $\mathrm{lca}(u_1, u_2), \ldots, \mathrm{lca}(u_{k-1}, u_k)$ 之中深度最小的去重后唯一的一个。

Property 4.1

对于一棵有根树及结点序列 $u_1, u_2, \ldots, u_k$，有性质 $\mathrm{lca}\{u_1, \ldots, u_k\}$ 为 $\mathrm{lca}(u_1, u_2), \ldots, \mathrm{lca}(u_{k-1}, u_k)$ 之中深度最小的去重后唯一的一个。

Property 4.2

对于一棵有根树及结点 u, v，满足 $\mathrm{dfn}(v) = \mathrm{dfn}(u) + 1$，则有 $\mathrm{lca}(u, v) = \mathrm{par}(v)$。

Problem (离线最近公共祖先)

给定一棵 n 个结点的有根树，q 次查询：给定结点 u, v，求 $\mathrm{lca}(u, v)$。要求时间复杂度 $O((n+q)\alpha(n))$。

Problem (离线最近公共祖先)

给定一棵 n 个结点的有根树，q 次查询：给定结点 u, v，求 $\mathrm{lca}(u, v)$。要求时间复杂度 $O((n+q)\alpha(n))$。

Solution (离线最近公共祖先的 Tarjan 算法)

我们先离线将询问 (u, v) 存储在结点 u, v 上（另一个结点与询问编号）。

Problem (离线最近公共祖先)

给定一棵 n 个结点的有根树，q 次查询：给定结点 u, v，求 $\mathrm{lca}(u, v)$。要求时间复杂度 $O((n+q)\alpha(n))$。

Solution (离线最近公共祖先的 Tarjan 算法)

我们先离线将询问 (u, v) 存储在结点 u, v 上（另一个结点与询问编号）。

在 dfs 过程中维护被遍历过的所有点与当前结点的最近公共祖先：用并查集维护不相交集合，并用数组对每个不相交集合维护其相同的值，回溯时将当前结点的集合与父亲结点的集合合并，并将合并后集合的答案设为父亲结点。

Problem (离线最近公共祖先)

给定一棵 n 个结点的有根树，q 次查询：给定结点 u, v，求 $\mathrm{lca}(u, v)$。要求时间复杂度 $O((n+q)\alpha(n))$。

Solution (离线最近公共祖先的 Tarjan 算法)

我们先离线将询问 (u, v) 存储在结点 u, v 上（另一个结点与询问编号）。

在 dfs 过程中维护被遍历过的所有点与当前结点的最近公共祖先：用并查集维护不相交集合，并用数组对每个不相交集合维护其相同的值，回溯时将当前结点的集合与父亲结点的集合合并，并将合并后集合的答案设为父亲结点。

dfs 至一个结点时遍历询问，若另一个结点已被遍历过即另一个结点的 dfs 序不超过当前结点，则直接在并查集上查询得到答案。

Solution

使用按秩合并与路径压缩时，时间复杂度 $\Theta((n+q)\alpha(n+q,n))$。

Solution

使用按秩合并与路径压缩时，时间复杂度 $\Theta((n+q)\alpha(n+q,n))$。
可以使用四毛子优化合并树已知的并查集做到 $\Theta(n+q)$。

Solution

使用按秩合并与路径压缩时，时间复杂度 $\Theta((n+q)\alpha(n+q,n))$。
可以使用四毛子优化合并树已知的并查集做到 $\Theta(n+q)$。

一般 OI 中离线最近公共祖先使用时间复杂度 $\Theta((n+q)\alpha(n+q,n))$ 的 Tarjan 算法。

Problem (在线最近公共祖先)

给定一棵 n 个结点的有根树，q 次查询：给定结点 u, v，求 $\mathrm{lca}(u, v)$。
强制在线，要求时间复杂度与静态区间最小值问题相同。

Problem (在线最近公共祖先)

给定一棵 n 个结点的有根树，q 次查询：给定结点 u, v，求 $\mathrm{lca}(u, v)$。
强制在线，要求时间复杂度与静态区间最小值问题相同。

Solution

处理出每个结点的 dfs 序与父亲，对非根结点 x 令 $f_{\mathrm{dfn}(x)} = \mathrm{par}(x)$，令 $\{l, r\} = \{\mathrm{dfn}(u), \mathrm{dfn}(v)\}$ 且满足 $l \leqslant r$。

Problem (在线最近公共祖先)

给定一棵 n 个结点的有根树，q 次查询：给定结点 u, v，求 $\mathrm{lca}(u, v)$。
强制在线，要求时间复杂度与静态区间最小值问题相同。

Solution

处理出每个结点的 dfs 序与父亲，对非根结点 x 令 $f_{\mathrm{dfn}(x)} = \mathrm{par}(x)$，令 $\{l, r\} = \{\mathrm{dfn}(u), \mathrm{dfn}(v)\}$ 且满足 $l \leqslant r$。

注意到 $\mathrm{dfn}(\mathrm{lca}(u, v)) = \min_{i=l+1}^{r} f_i$，我们在预处理 $\Theta(n)$ 单次查询 $\Theta(1)$ 的时间复杂度内将问题转化为了静态区间最小值问题。

一般此处静态区间最小值问题使用 ST 表求解，时空复杂度预处理 $\Theta(n \log n)$ 单次查询 $\Theta(1)$。

一般此处静态区间最小值问题使用 ST 表求解，时空复杂度预处理 $\Theta(n \log n)$ 单次查询 $\Theta(1)$。

由于树的遍历在任意给定边集的情况下常数较大，一般来说 n 的数据范围均可接受 ST 表的预处理时间，空间常数较小，一般也没有出题人特意限制。

一般此处静态区间最小值问题使用 ST 表求解，时空复杂度预处理 $\Theta(n \log n)$ 单次查询 $\Theta(1)$。

由于树的遍历在任意给定边集的情况下常数较大，一般来说 n 的数据范围均可接受 ST 表的预处理时间，空间常数较小，一般也没有出题人特意限制。

我们可以用四毛子的静态区间最小值问题解法做到时空复杂度预处理 $\Theta(n)$ 单次查询 $\Theta(1)$。

一般此处静态区间最小值问题使用 ST 表求解，时空复杂度预处理 $\Theta(n \log n)$ 单次查询 $\Theta(1)$。

由于树的遍历在任意给定边集的情况下常数较大，一般来说 n 的数据范围均可接受 ST 表的预处理时间，空间常数较小，一般也没有出题人特意限制。

我们可以用四毛子的静态区间最小值问题解法做到时空复杂度预处理 $\Theta(n)$ 单次查询 $\Theta(1)$。

在题目解法其它部分使用重链剖分时，在线最近公共祖先也常直接使用重链剖分求解，时间复杂度预处理 $\Theta(n)$ 单次查询 $O(\log n)$。

Problem (在线加叶子求最近公共祖先)

有根树 $T=(V,E)$ 以 1 为根，初始时 $V=\{1\},E=\varnothing$，q 次操作：

1. 给定 $p\in V,u=|V|+1$，执行 $V\leftarrow\{u\},E\leftarrow E\cup\{\{p,u\}\}$。
2. 给定 $u,v\in V$，查询 $\mathrm{lca}(u,v)$。

强制在线，要求时间复杂度 $O((q_1+q_2)\log q_1)$。

Problem (在线加叶子求最近公共祖先)

有根树 $T=(V,E)$ 以 1 为根，初始时 $V=\{1\},E=\varnothing$，q 次操作：

1. 给定 $p\in V,u=|V|+1$，执行 $V\leftarrow\{u\},E\leftarrow E\cup\{\{p,u\}\}$。
2. 给定 $u,v\in V$，查询 $\mathrm{lca}(u,v)$。

强制在线，要求时间复杂度 $O((q_1+q_2)\log q_1)$。

Solution

我们对每个非根结点 u 对 $i\in[0,\log_2\mathrm{dist}(1,u)]\cap\mathbb{N}$ 维护 $\mathrm{par}_{2^i}(u)$ 与 $\mathrm{dep}(u)$。

Problem (在线加叶子求最近公共祖先)

有根树 $T=(V,E)$ 以 1 为根，初始时 $V=\{1\},E=\varnothing$，q 次操作：

1. 给定 $p\in V,u=|V|+1$，执行 $V\leftarrow\{u\},E\leftarrow E\cup\{\{p,u\}\}$。
2. 给定 $u,v\in V$，查询 $\mathrm{lca}(u,v)$。

强制在线，要求时间复杂度 $O((q_1+q_2)\log q_1)$。

Solution

我们对每个非根结点 u 对 $i\in[0,\log_2\mathrm{dist}(1,u)]\cap\mathbb{N}$ 维护 $\mathrm{par}_{2^i}(u)$ 与 $\mathrm{dep}(u)$。

对于加叶子操作，注意到 $\mathrm{par}(u)=p,\mathrm{dep}(u)=\mathrm{dep}(p)+1$，对 $i\in[1,\log_2\mathrm{dist}(1,u)]\cap\mathbb{N}$ 有 $\mathrm{par}_{2^i}(u)=\mathrm{par}_{2^{i-1}}(\mathrm{par}_{2^{i-1}}(u))$，单次时间复杂度为 $\Theta(\log q_1)$。

Solution

对于查询操作，显然 $\mathrm{dep}(\mathrm{lca}(u,v)) \leqslant \min(\mathrm{dep}(u), \mathrm{dep}(v))$，因此对于 $\mathrm{dep}(u) < \mathrm{dep}(v)$，我们有 $\mathrm{lca}(u,v) = \mathrm{lca}(u, \mathrm{par}_{\mathrm{dep}(v)-\mathrm{dep}(u)}(v))$，我们可以用倍增（或二进制分组）在 $\Theta(1+\log k)$ 的时间复杂度内求出 $\mathrm{par}_k(u)$，因此我们可以在 $\Theta(\log q_1)$ 的时间复杂度内将查询转化为 $\mathrm{dep}(u) = \mathrm{dep}(v)$ 的情况。

Solution

对于查询操作，显然 $\mathrm{dep}(\mathrm{lca}(u,v)) \leqslant \min(\mathrm{dep}(u),\mathrm{dep}(v))$，因此对于 $\mathrm{dep}(u) < \mathrm{dep}(v)$，我们有 $\mathrm{lca}(u,v) = \mathrm{lca}(u, \mathrm{par}_{\mathrm{dep}(v)-\mathrm{dep}(u)}(v))$，我们可以用倍增（或二进制分组）在 $\Theta(1+\log k)$ 的时间复杂度内求出 $\mathrm{par}_k(u)$，因此我们可以在 $\Theta(\log q_1)$ 的时间复杂度内将查询转化为 $\mathrm{dep}(u) = \mathrm{dep}(v)$ 的情况。

此时若对于已知的 $k \in \mathbb{N}$ 有 $\mathrm{dep}(\mathrm{lca}(u,v)) - \mathrm{dep}(u) \leqslant 2^{k+1}$ 则我们可以比较 $u' = \mathrm{par}_{2^k}(u)$ 与 $v' = \mathrm{par}_{2^k}(v)$，若相同则 $\mathrm{dep}(\mathrm{lca}(u,v)) - \mathrm{dep}(u) \leqslant 2^k$ 否则 $\mathrm{lca}(u,v) = \mathrm{lca}(u',v')$ 且 $\mathrm{dep}(u') = \mathrm{dep}(v'), \mathrm{dep}(\mathrm{lca}(u',v')) - \mathrm{dep}(u') \leqslant 2^k$。

Solution

对于查询操作，显然 $\mathrm{dep}(\mathrm{lca}(u,v)) \leqslant \min(\mathrm{dep}(u), \mathrm{dep}(v))$，因此对于 $\mathrm{dep}(u) < \mathrm{dep}(v)$，我们有 $\mathrm{lca}(u,v) = \mathrm{lca}(u, \mathrm{par}_{\mathrm{dep}(v)-\mathrm{dep}(u)}(v))$，我们可以用倍增（或二进制分组）在 $\Theta(1+\log k)$ 的时间复杂度内求出 $\mathrm{par}_k(u)$，因此我们可以在 $\Theta(\log q_1)$ 的时间复杂度内将查询转化为 $\mathrm{dep}(u) = \mathrm{dep}(v)$ 的情况。

此时若对于已知的 $k \in \mathbb{N}$ 有 $\mathrm{dep}(\mathrm{lca}(u,v)) - \mathrm{dep}(u) \leqslant 2^{k+1}$ 则我们可以比较 $u' = \mathrm{par}_{2^k}(u)$ 与 $v' = \mathrm{par}_{2^k}(v)$，若相同则 $\mathrm{dep}(\mathrm{lca}(u,v)) - \mathrm{dep}(u) \leqslant 2^k$ 否则 $\mathrm{lca}(u,v) = \mathrm{lca}(u',v')$ 且 $\mathrm{dep}(u') = \mathrm{dep}(v'), \mathrm{dep}(\mathrm{lca}(u',v')) - \mathrm{dep}(u') \leqslant 2^k$。

因此我们可以使用倍增在 $\Theta(\log q_1)$ 的时间复杂度内求出两点最近公共祖先。

我们考虑如何在修改操作复杂度可接受的情况下进行 $\Theta(1)$ 询问。

我们考虑如何在修改操作复杂度可接受的情况下进行 $\Theta(1)$ 询问。

Solution

我们考虑加一个叶子对树 dfs 序产生的影响：如果 dfs 时先访问新加入的叶子，那么加一个叶子后的 dfs 序可以由原先 dfs 序在给定父亲结点后加入给定叶子得到。

我们考虑如何在修改操作复杂度可接受的情况下进行 $\Theta(1)$ 询问。

Solution

我们考虑加一个叶子对树 dfs 序产生的影响：如果 dfs 时先访问新加入的叶子，那么加一个叶子后的 dfs 序可以由原先 dfs 序在给定父亲结点后加入给定叶子得到。

因此我们可以用静态最近公共祖先转化为静态区间最值的相同方式将问题转化为：维护一条带点权的有向链，修改操作给定一个结点，将其出边分裂为两条并在中间插入另一个给定点权的新结点，保证点权不低于原先出边两个端点中任意一个的点权，查询操作给定两个结点，查询左开右闭的路径最小值。

我们考虑如何在修改操作复杂度可接受的情况下进行 $\Theta(1)$ 询问。

Solution

我们考虑加一个叶子对树 dfs 序产生的影响：如果 dfs 时先访问新加入的叶子，那么加一个叶子后的 dfs 序可以由原先 dfs 序在给定父亲结点后加入给定叶子得到。

因此我们可以用静态最近公共祖先转化为静态区间最值的相同方式将问题转化为：维护一条带点权的有向链，修改操作给定一个结点，将其出边分裂为两条并在中间插入另一个给定点权的新结点，保证点权不低于原先出边两个端点中任意一个的点权，查询操作给定两个结点，查询左开右闭的路径最小值。

注意到维护 n 个元素时，多数平衡树插入一个结点均摊会改变至多 $\Theta(\log n)$ 个结点的到根链情况，treap（含非旋的写法）为期望 $\Theta(\log n)$。

我们考虑如何在修改操作复杂度可接受的情况下进行 $\Theta(1)$ 询问。

Solution

我们考虑加一个叶子对树 dfs 序产生的影响：如果 dfs 时先访问新加入的叶子，那么加一个叶子后的 dfs 序可以由原先 dfs 序在给定父亲结点后加入给定叶子得到。

因此我们可以用静态最近公共祖先转化为静态区间最值的相同方式将问题转化为：维护一条带点权的有向链，修改操作给定一个结点，将其出边分裂为两条并在中间插入另一个给定点权的新结点，保证点权不低于原先出边两个端点中任意一个的点权，查询操作给定两个结点，查询左开右闭的路径最小值。

注意到维护 n 个元素时，多数平衡树插入一个结点均摊会改变至多 $\Theta(\log n)$ 个结点的到根链情况，treap（含非旋的写法）为期望 $\Theta(\log n)$。

我们考虑用平衡树维护这些链的结点。

Solution

每个结点用一个位数为 $\Theta(\log n)$ 的整数从低位到高位维护从根到它每一个结点分别为左儿子还是右儿子，求 lca 的深度可以异或后求低位端 0 的位数，时间复杂度 $\Theta(1)$。

Solution

每个结点用一个位数为 $\Theta(\log n)$ 的整数从低位到高位维护从根到它每一个结点分别为左儿子还是右儿子，求 lca 的深度可以异或后求低位端 0 的位数，时间复杂度 $\Theta(1)$。

类似猫树，对每个结点 u 和每个深度 d 维护 u 与其平衡树上深度为 d 的祖先结点（若为 leafy 的平衡树则为其左子树的最后一个链中结点）在链上的左开右闭区间的最小值。

Solution

每个结点用一个位数为 $\Theta(\log n)$ 的整数从低位到高位维护从根到它每一个结点分别为左儿子还是右儿子，求 lca 的深度可以异或后求低位端 0 的位数，时间复杂度 $\Theta(1)$。

类似猫树，对每个结点 u 和每个深度 d 维护 u 与其平衡树上深度为 d 的祖先结点（若为 leafy 的平衡树则为其左子树的最后一个链中结点）在链上的左开右闭区间的最小值。

我们可以在平衡树上做二区间合并，综上所述我们可以在严格 $\Theta(1)$ 的时间复杂度内完成 lca 的查询。

Solution

每个结点用一个位数为 $\Theta(\log n)$ 的整数从低位到高位维护从根到它每一个结点分别为左儿子还是右儿子，求 lca 的深度可以异或后求低位端 0 的位数，时间复杂度 $\Theta(1)$。

类似猫树，对每个结点 u 和每个深度 d 维护 u 与其平衡树上深度为 d 的祖先结点（若为 leafy 的平衡树则为其左子树的最后一个链中结点）在链上的左开右闭区间的最小值。

我们可以在平衡树上做二区间合并，综上所述我们可以在严格 $\Theta(1)$ 的时间复杂度内完成 lca 的查询。

我们考虑修改：对于到根链的情况不变的结点，由于新加的叶子不会对已有的区间最值产生影响，其维护的信息不发生变化；对于发生变化的结点，我们在多数平衡树中往往容易在 $\Theta(\log n)$ 的时间复杂度内暴力更新单个结点的信息，单次修改总时间复杂度均摊或期望 $\Theta(\log^2 n)$。

Solution

我们可以使用底层分块的思路优化修改操作的时间复杂度：我们将至多 $B = \Theta(\log^2 q_1)$ 个结点作为一个块，每个块的权值是块内所有结点的权值最小值，对每个块建一个结点按前述算法维护，每次在一个结点后插入一个新结点时若块内结点数不超过 B 则只需更新块内信息，若块内结点数超过 B 则将块内结点对半分为两个块。

Solution

我们可以使用底层分块的思路优化修改操作的时间复杂度：我们将至多 $B = \Theta(\log^2 q_1)$ 个结点作为一个块，每个块的权值是块内所有结点的权值最小值，对每个块建一个结点按前述算法维护，每次在一个结点后插入一个新结点时若块内结点数不超过 B 则只需更新块内信息，若块内结点数超过 B 则将块内结点对半分为两个块。

注意到每个分裂后的块至少再插入 $\lceil B/2 \rceil$ 个结点才会再次分裂，我们容易通过势能分析说明块间的信息修改时间复杂度均摊 $\Theta(1)$。

Solution

我们可以使用底层分块的思路优化修改操作的时间复杂度：我们将至多 $B = \Theta(\log^2 q_1)$ 个结点作为一个块，每个块的权值是块内所有结点的权值最小值，对每个块建一个结点按前述算法维护，每次在一个结点后插入一个新结点时若块内结点数不超过 B 则只需更新块内信息，若块内结点数超过 B 则将块内结点对半分为两个块。

注意到每个分裂后的块至少再插入 $\lceil B/2 \rceil$ 个结点才会再次分裂，我们容易通过势能分析说明块间的信息修改时间复杂度均摊 $\Theta(1)$。

注意到对于块内结点数为 l 时我们可以通过 $\Theta(l!l^2)$ 时空复杂度的预处理维护 $l!$ 种插入顺序 $\Theta(l^2)$ 中询问答案为第几次插入的数做到 $\Theta(1)$ 处理块内查询，我们可以在长度为 $\Theta(\log^2 q_1)$ 的块内再套一层底层分块，对于 $l \leqslant B_2 = \Theta(\log^2 \log q_1)$ 的情况进行预处理。

Solution

我们可以使用底层分块的思路优化修改操作的时间复杂度：我们将至多 $B = \Theta(\log^2 q_1)$ 个结点作为一个块，每个块的权值是块内所有结点的权值最小值，对每个块建一个结点按前述算法维护，每次在一个结点后插入一个新结点时若块内结点数不超过 B 则只需更新块内信息，若块内结点数超过 B 则将块内结点对半分为两个块。

注意到每个分裂后的块至少再插入 $\lceil B/2 \rceil$ 个结点才会再次分裂，我们容易通过势能分析说明块间的信息修改时间复杂度均摊 $\Theta(1)$。

注意到对于块内结点数为 l 时我们可以通过 $\Theta(l!l^2)$ 时空复杂度的预处理维护 $l!$ 种插入顺序 $\Theta(l^2)$ 中询问答案为第几次插入的数做到 $\Theta(1)$ 处理块内查询，我们可以在长度为 $\Theta(\log^2 q_1)$ 的块内再套一层底层分块，对于 $l \leqslant B_2 = \Theta(\log^2 \log q_1)$ 的情况进行预处理。

因此时空复杂度预处理 $O(q_1)$，修改均摊 $\Theta(1)$，查询严格 $\Theta(1)$。

Solution

如果需要避免预处理时间复杂度，我们可以令 B 和 B_2 在当前修改次数的对数翻倍时进行重构，按照翻倍后的情况进行制定块大小。

Solution

如果需要避免预处理时间复杂度，我们可以令 B 和 B_2 在当前修改次数的对数翻倍时进行重构，按照翻倍后的情况进行制定块大小。

时空复杂度预处理 $\Theta(1)$，修改均摊 $\Theta(1)$，查询严格 $\Theta(1)$。