《Jumping Lights》解题报告

题目大意

给定 n 个结点的树 T，初始每个结点都未被标记。接下来有 q 次操作：

- `0 u`，表示去除结点 u 上的标记，若其未被标记则不操作；`1 u`，表示把结点 u 标记，若其已被标记则不操作。
- `2 u`，表示进行一次如下操作：对每个点 u，若操作前存在 u 的邻居 v 使得 v 被标记，则操作后将 u 标记；否则去除 u 上的标记。

每次操作后，输出树上被标记结点的个数。

数据范围

$1 \le n \le 3 \times 10^5$，$1 \le q \le 10^6$。

时间复杂度 2s，空间复杂度 512MB。

解题过程

以下记被标记为 1，未被标记为 0。记第 i 个 2 操作发生后，第 $i+1$ 个 2 操作发生前为 i 时刻。

不妨设原树 T 为以 1 为根的有根树，下称一个节点为叶子当且仅当其在以 1 为根的意义下没有子节点。

观察到若从某个状态开始一直进行 2 操作，最终整棵树将会进入一个周期为 2 的循环中。因此分开考虑奇数刻/偶数刻的树的形态。具体来说，我们令 T_i 为维护时刻 i 的树中每个结点的标记情况的数据结构（为简化说明，不妨定义 $\forall i \le 0, T_i = T_0$，即 i 可为负数），每次 0/1 操作直接在当前 T_i 上进行修改，2 操作时新的 T_{i+1} 由 T_{i-1} 进行某些修改得来。

考虑 2 操作中修改的具体形式。我们可将其等效为若干次标记 $0 \to 1$ 与 $1 \to 0$。注意到若一直进行 2 操作，每个数据结构中为 1 的结点的个数似乎是不降的，即使有修改也如此；但若有很多叶子同时被修改了，则不好处理。接下来将此转化为严谨的算法。我们记势能函数 $\Phi(T_i)$ 将数据结构 T_i 映射至 T_i 中为 1 **非叶子结点** 构成的集合，设当前时刻为 i，第 i 时刻发生的单点修改次数为 D_i。随后给出如下结论：

结论1

- $|\Phi(T_{i-1}) \backslash \Phi(T_{i+1})| \le D_i$。即在 T_{i-1} 中为 1，而在 T_{i+1} 中为 0 的结点个数不超过时刻 i 内的修改个数。

首先给出下述引理：记 $N(S)$ 表示树 T 上点集 S 的邻域的并集。对于非叶子节点集合 S，有 $|N(S)| \ge |S|$。

对引理的证明：对每个非叶子节点 u 任取一子节点，记为 ch_u。显然 $\forall u \in S$，$ch_u \in N(S)$ 且 ch_u 两两不同，因此 $|N(S)| \ge |\bigcup_u ch_u| = |S|$。

回到原证明，记 $S = \Phi(T_{i-1}) \backslash \Phi(T_{i+1})$。考虑若结点 u 在 T_{i-1} 中为 1，在一次 2 操作后 T_i 上其所有邻居必然均变成 1。同时其在 T_{i+1} 中为 0，说明其在 T_i 的所有邻居也都为 0。因此时刻 i 中 u 的所有邻居必然都被修改过一次。被修改的点集包含 $N(S)$，则有 $|S| \le |N(S)| \le D_i$，结论得证。

通过分析势能，我们立刻能够得到关于 $0 \to 1$ 的如下结论：

结论2

- $\sum|\Phi(T_{i+1})\backslash\Phi(T_{i-1})| \le 2n+q$。即对于所有 2 操作，从 T_{i-1} 到 T_{i+1} 发生 $0\to 1$ 的结点个数的总和不超过 $2(n+q)$。

证明是较为简单的。考察 $|\Phi(T_i)|$ 的变化，一开始 $|\Phi(T_{-1})| = |\Phi(T_0)| = 0$，且任意时刻集合大小至多为 n。从 T_{i-1} 到 T_{i+1} 的过程中，至多有 D_i 个数被从 T_{i-1} 集合中删去，而 T_i 在单点修改的过程中也至多失去了 D_i 个点，因此奇偶两部分整个过程中至多删去了 $\sum_i 2D_i = 2q$ 个数；而加入数的次数等于 $0\to 1$ 的修改个数。由聚合分析可知其不应超过 $2(n+q)$ 次。

至此我们证明了对于非叶子节点，每次 2 操作在对应数据结构上对结点标记的变动次数是线性的，因此可以暴力找到所有需要变动的结点。

当然我们还需要一种对叶子的处理方法：将所有叶子按照其邻居划分等价类，注意到一次操作后所有同一个等价类的叶子状态相同，因此只需关心叶子等价类的整体是否发生了变化，而无需对其中的每个点进行操作。我们依据这些观察设计算法。对每个 T_i 维护如下内容：

- 对每个结点 u，维护其子节点中所有非叶子 0 节点构成的集合 W_u，以及叶节点 $0/1$ 结点个数 $cnt_{0/1}$。
- 维护集合 S_{lst} 表示第 $i-1$ 时刻所有 $0\to 1$ 的非叶结点及第 i 时刻所有因单点修改变成 1 的点的集合，S_w 表示第 i 时刻内所有因单点修改变成 0 的点的集合。

这些集合需要支持两种操作：加入/删除一个点，以及访问所有集合内的点。这两种操作的复杂度分别需要为 $O(1)$ 和 $O(size)$。首先论证该数据结构的可实现性。我们维护所有原先集合内的点以及所有修改构成的集合，修改时将一个修改标记加入该集合，访问时 $O(size)$ 处理所有修改，判断哪些点仍旧属于当前集合。由于一次修改至多让该集合 $size$ 与所维护集合的 $size$ 的差增大 2，势能分析可知满足上述复杂度要求。

在进行 2 操作时，我们将其拆解为三个步骤：找到 $T_{i-1}\to T_{i+1}$ 中所有 $1\to 0$ 和 $0\to 1$ 的非叶节点并暴力更新，然后修改所有叶子的值。首先对于 $1\to 0$，根据前述结论其必然存在于某个被修改为 0 的点的邻域中。我们对 T_i 的 S_w 里每个点查询其父亲 u，判断其邻域是否包含于 S_w，然后暴力修改，复杂度为 $O(S_w)$。对于 $0\to 1$，考虑若点 u 在 T_{i-1} 中为 0 而在 T_{i+1} 中为 1，其必然满足以下二者之一：

- 其在 T_{i-2} 中周围所有点为 0，但其在第 i 时刻结束时周围有点为 1。这说明其存在一个邻居 v 要么在第 i 刻被修改了，要么在 $T_{i-2}\to T_i$ 的过程中被变成了 1。但如果是后者，v 必然不是叶子，否则与 u 在 T_{i-1} 中为 0 矛盾。此时枚举 v 即枚举 T_i 的 S_{lst}，对其内所有点暴力遍历 W 与父节点然后进行修改即可。
- 其在 T_{i-2} 变成 T_{i-1} 的 2 操作中变为了 1，但随后在第 $i-1$ 时刻被单点修改为了 0。此时可直接枚举 T_{i-1} 的 S_w 并判断其内所有点是否需要修改。

叶节点的情况是类似的，只需要在上面枚举 S_w 和枚举 S_{lst} 的过程中顺便更新其叶节点集合即可。此处我们需要支持全局赋值，维护 cnt 的操作，显然可以做到 $O(1)$。

最后是所有可能的单点修改。我们只需要修改该点父亲的 W_u 和 cnt，当前数据结构的 S_w 和 S_{lst}。这些均可以做到 $O(1)$。

综合分析一下所有的复杂度。对每个 S_w 我们分别在 i 时刻和 $i+1$ 枚举了一次，总复杂度 $O(\sum|S_w|) \le O(\sum D_i) \le O(q)$。对每个 S_{lst} 我们只在第 i 时刻枚举了一次，而 $\sum|S_{lst}| \le (\sum D_i) + \sum(|\Phi(T_{i+1})\backslash\Phi(T_{i-1})|) = O(n+q)$。对每个 W_u 我们枚举的复杂度之和即为 $0\to 1$ 节点数，且枚举完清空整个集合，因此复杂度可均摊至对 $0\to 1$ 个数的分析中。总时间复杂度 $O(n+q)$。

注意具体实现时，需要仅用两个数据结构 T_0 和 T_1 分别模拟奇数刻和偶数刻的情形。