试题交流 解题报告

宁波市镇海中学 钱易

大鱼治水

简要题意

本题是一道交互题。

给定一个 n 个节点以 1 为根的树，其中有 $n-1$ 条边，每条边要么是轻边，要么是重边。任意时刻一个节点到儿子的边中最多有一条重边。

你一开始给出每条边是轻边还是重边，之后可以调整轻重边，每次操作可以删除一条重边将其改为轻边或者加入一条重边代替轻边。

给定 q 个要求，每个要求会给定一个节点 x，你需要调整轻重边使得 x 到根路径均为重边，达成要求之后可以继续调整，假设调整总次数最大值为 w，当 $w \leq 35$ 时即可获得满分，不然会根据 w 大小获得一定部分分。

数据范围

$1 \leq n \leq 50000, 1 \leq q \leq 500000$

题解

算法一

对于该问题，我们最容易想到的算法就是使用 Link Cut Tree 直接维护，在 access 过程中改变轻重链。

很可惜该算法的操作次数是均摊 $O(\log n)$ 的，其在最坏情况下可能可以达到 $O(n)$ 的切换次数，因此 Link Cut Tree 并不能直接应用在本题之中。

算法二

显然的是，一个点任意时刻到根轻边数量不能太多，我们考虑使用轻重链剖分。

对于初始情况，我们直接给出原树的一个轻重链剖分，然后对于每个要求，我们把这个点上方所有轻边改为重边，然后再把修改了的轻边修改回去。

在每一条轻边处，我们都要删除重边，加入重边两次，因此最大操作次数就可能达到 $4 \times (\lceil \log_2(n+1) \rceil - 1)$，当 $n = 50000$ 时会进行 56 次修改，可以获得 18 分的成绩。

算法三

我们来思考一下轻重链剖分为何会消耗如此之多的步数。一个典型的将轻重链剖分算法复杂度卡满的树结构是满二叉树，它的特点是一个点两个儿子的大小是一样的，这时候如果我们使用之前的方法，可能会将重边连到一个儿子再连回去，而连回去这一步是完全没有必要的，而何时我们没有必要连回去呢?

我们假设节点 x 的子树大小为 $size_x$。

我们假定一个常数 $0 \leq \alpha \leq 1$，如果一个节点 x 与其父亲 p 的子树大小满足 $size_x \geq size_p * \alpha$ 时，如果我们把 p 到儿子的重边连到了 x，我们在恢复时将不会把重边连回 p 的重儿子。

此时我们来分析一下时间复杂度：假设我们我们在从下往上调整轻重边时遇到了一条轻边，下面的节点是 x，上面的节点是 p，如果这条轻边改为重边后不用改回去，那么此时要删除一条重边，添加一条重边，于此同时所在节点子树大小至少变为了原来的 $\frac{1}{1-\alpha}$ 倍（因为此时调整前重边在另外一个大小不小于 $\alpha \times size_p$ 的子树），如果需要改回去，那么此时要删除两次重边，添加两次重边，所在子树大小至少变为了原来的 $1/\alpha$ 倍（因为 $size_x < size_p * \alpha$）。

假设当前子树大小是 p，那么我们可能会遇到两种情况:

1. p 变为 q，其中 $q \geq \frac{p}{1-\alpha}$，消耗了两步。

2. p 变为 q，其中 $q \geq \frac{p}{\alpha}$，消耗了四步。

p 的大小始终不会超过 n，我们可以解得最优情况下 $\alpha = \frac{3-\sqrt{5}}{2}$，此时步数是 $O(2\log_{\phi} n + O(1))$，其中 $\phi = \frac{\sqrt{5}+1}{2}$，大约 42 步左右，可以得到 52 分左右的成绩。

算法四

算法三使用了一种新的策略，然而阈值算法可能导致算法多出一些无谓的常数，我们考虑如何优化。

我们假设 dp_x 为子树 x 中，操作一个点最多花费几次添加删除重链。

我们考虑进行树形 DP 计算如何确定策略，假设节点 x 的儿子的 dp 值分别是 $a_i (1 \leq i \leq m)$，并且满足 $a_i \geq a_{i+1} (1 \leq i < m)$。

在算法三中，存在一些儿子，它们子树中要求过后我们不会把重边改到重儿子，显然的是这些儿子的 dp 值都要比其它的大，我们假设前 k 个为这些特殊的儿子。

那么节点 x 的 dp 值为 $\max(2 + \max_{i=1}^{k} a_i, 4 + \max_{i=k+1}^{m} a_i)$，可以发现 $k = m$ 时最优，因此 dp 值为 $2 + \max_{i=1}^{m} a_i$。

特别得，当 $k = 1$ 时，我们唯一的特殊儿子中要求永远不需要在这个节点切换，dp 值为 $max(a_1, 4 + \max_{i=2}^{m} a_i)$。

因此我们可以轻而易举求出每个节点的哪些儿子是特殊的，我们应该使用怎样的策略。

我们令 g_x 代表该算法中，最坏情况下要操作 x 次最小的树的节点个数。

经过计算可得: $g_2 = 3, g_4 = 5, g_{2k} = g_{2k-2} + g_{2k-4} + 1$，其中 $g_{40} = 35421, g_{42} = 57313$，因此该算法最坏情况下需要 40 步，可以获得 60 分的成绩。

算法五

算法四通过树形 DP 细化使用的策略，我们考虑继续使用树形 DP 的方式决定策略。

我们考虑添加一种新的策略: 如果对一个节点，对于它某个儿子内的要求，我们不进行任何恢复，对于其它儿子内的要求，我们恢复仅仅删除重边（此时这个节点处不存在任何重边）。

此时，重儿子在这个地方最多进行一次加入重边，轻儿子在这个地方最多删除两次重边，加入一次重边。

我们继续沿用算法四中的 DP，那么使用这个策略得到此时的 x 节点的 dp 值为 $\max(a_1+1, 3+\max_{i=2}^{m} a_i)$。

我们在这个节点处判断两个策略哪个更加优秀，并且使用优秀的策略进行修改。

我们令 g_x 代表该算法中，最坏情况下要操作 x 次最小的树的节点个数。

经过计算可得: $g_2 = 3, g_3 = 5, g_4 = 7, g_i = g_{i-2} + g_{i-3} + 1$，其中 $g_{35} = 47665, g_{36} = 63143$，因此该算法最坏情况下需要 35 步，可以获得满分的好成绩。

但是需要注意的是，本题数据范围较大，并不能以一步一步向上跳的形式直接实行给出的策略，我们可以使用类似轻重链剖分的方法，跳过连续的一段没必要的节点，仅仅在关键节点处停留判断，就可以达到一个较为优秀的时间复杂度。

总结

本题是一道本人原创的一道交互题，题意简单自然，切合实际，存在轻重链剖分等不同巧妙的策略，区分度良好，而最后我们使用了树形 DP 自然的将几种不同的策略结合在了一起，个人认为思路新颖巧妙，是一道不可多得的好题，故选择这题来试题交流。