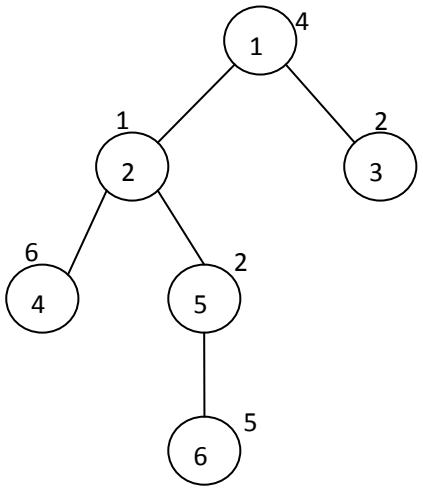
《网络管理》解题报告

by 龙凡

【问题简述】

给定一棵 N 个节点的带权树（权在节点上），如下图所示：



现在需要支持如下两种操作：

a. 修改某个点的权值。

b. 给定两个点 X 和 Y，以及一个整数 K。找出 X 到 Y 的路径上所经过的点中权值第 K 大的点。

操作的总数是 Q，需要对所有的 b 类操作进行正确的回答。

本题的数据范围：

100%数据满足$N \leq 80000$，$Q \leq 30000$。权值小于 10^8

40%数据满足$1 \leq k \leq 5$，且不存在改值操作。

10%数据满足$N, Q \leq 8000$

【考察点】

树的结构及其遍历

二分思想以及数据结构的灵活运用

【算法分析】

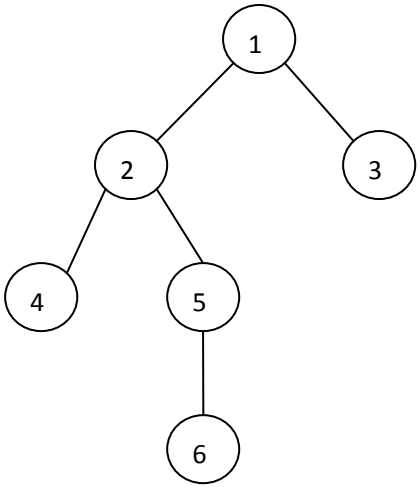
按照惯例，每年国家队选拔赛都会有一道题着重考察数据结构的设计和实现，标准算法使用的是平衡树加线段树的数据结构。并且在数据规模上兼顾了常见的各种其他算法，对于40%的数据可以用带有二分思想的动态规划来求解。10%的数据规模较小，可以直接用 $O(N^2)$的算法模拟计算得出。如果选手能够完成这两部分的程序，也能够拿到 50 分，而这已经是相当不错的分数了。

值得一提的是，标准算法中包含了平衡树、线段树和 LCA 问题求解等三个重要部分，要完整的在竞赛时间中编写正确这些代码对选手数据结构实现的基本功也是一种考验。

首先本题的难点之一就是本题的操作是在一颗树上执行的。而我们所熟悉的各类二分数据结构如线段树等都是针对序列的。所以我们先将树上的操作转化为序列上的操作才行。注意到，我们所需要关注的是两个节点 A 到 B 之间路径上经过的节点。我们知道，LCA 问题是可以有 O(N)-O(1)时间复杂度的在线算法的，所以 LCA 的求解肯定不会影响整个的时间复杂度。不妨我们认为我们已经能够求出 A 和 B 的最近公共祖先 C，那么 A 到 B 之间的路径可以做如下分解：

A 到 B 的路径 = 根到 A 的路径 + 根到 B 的路径 − 根到 C 的路径

现在我们需要将树中根到某个点 X 的路径中的点集用某种序列表示出来。即将树中的询问问题转化为序列中的询问问题。DFS 先序遍历序列拥有良好的性质，可以满足我们的需要。



在 DFS 先序遍历时，我们入栈和出栈时均记录一次这个节点的编号。那么左边的图遍历的序列是：

1 2 4 ***4*** 5 6 ***6 5 2*** 3 ***3 1***

上面的序列中粗体表示退栈。我们发现某个节点 A 到根所经过的点集刚好等于 A 进栈之后，已经进栈但尚未出栈的所有节点的集合。比如左图中的 6 号节点。当它进栈时，1、2、4、5 和 6 节点都已经进栈，且只有 4 号节点出栈了。那么 6 号节点到根所需要经过的路径上的点是 1、2、5 和 6。

于是：

根到节点 X 经过的点集 = X 入栈时已经入栈的节点 − X 入栈时已经出栈的节点

为了以下表述方便，我们记 X 入栈时已经入栈的节点为 Enter(X)，X 入栈时已经出栈的节点为 Leave(X)。注意到 Enter(X)和 Leave(X)实际上可以对应到入栈和出栈序列中的一段连续的区间。

总结来说：

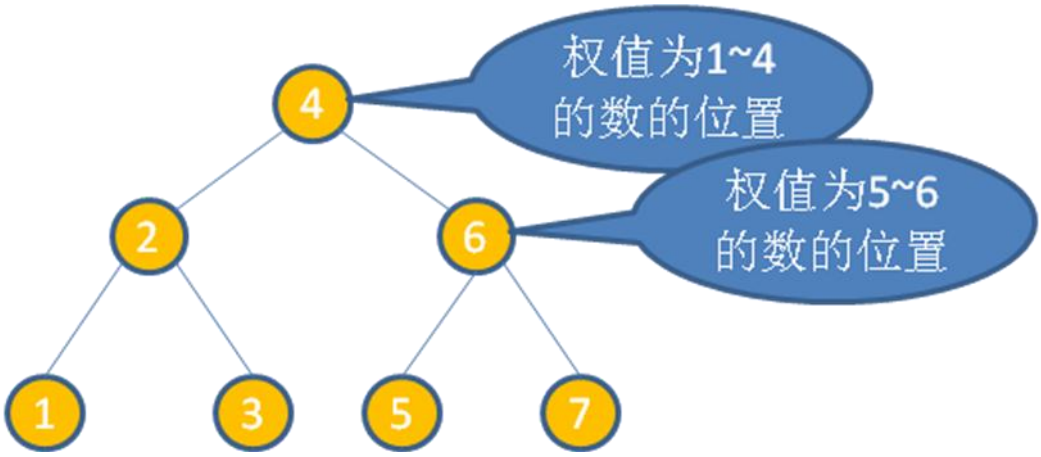
$$\begin{aligned}\text{Path\_Set}(A,B) &= \text{Enter}(A) - \text{Leave}(A) + \text{Enter}(B) - \text{Leave}(B) - (\text{Enter}(C) \\ &\quad - \text{Leave}(C))\end{aligned}$$

于是，我们先进行预处理将整个树 DFS 按先序遍历走一遍，将入栈和出栈的

序列生成出来。对于关于 A 点到 B 点路径中第 K 大的权值的提问，求出 A 和 B 的最近公共祖先 C。然后二分答案 Ans，分别计算 Enter(A)、Leave(A)、Enter(B)、Leave(B)、Enter(C)和 Leave(C)中权值比 Ans 大的节点有多少个。如果按照常规的线段树加平衡树的建树方法来实现上面操作，那么每次求解需要 $O(\log^2 N)$。加上二分 Ans 需要在 $O(\log^3 N)$的时间复杂度才能够完成一次询问。

上面的算法，已经能够通过绝大部分数据。但对于极端数据，仍然可能超时。事实上，我们可以利用题目并没有要求 Online 算法的特点，设计出每次询问 $O(\log^2(N+Q))$的算法的。其核心思想就是，将二分过程融入到数据结构的设计中。

由于我们是对最后回答的答案也就是权值进行二分，那么我们需要打破常规，在线段树层以权值进行建树。我们先对整个数据中出现过的权值进行离散化，权值最多可能有 N+Q 个不同的值。离散化之后，设权值为 1~7，那么我们按下图的结构分别对入栈和出栈序列建立两颗线段树：



上图中的每一个节点都是一颗平衡树。比如 4 号点中存储的就是由权值为 1~4 的所有的节点组成的平衡树，其 Key 值为节点在入栈（出栈）序列中的位置。这样在二分的过程中，就只需要顺着这棵树一直走下来就可以了。

比如我们先二分假设回答的权值 Ans 为 4（离散化后的值）。那么就询问 4 号点中存储的平衡树，得到权值为 1~4 的点中，有多少点的入栈（出栈）位置比 A、B、C 点在入栈位置靠前。如果下一步需要尝试更小的 Ans，则在树中向左走。否则我们将在 4 号点求出的节点数量累加进某个计数器，并向右走。按照这个过程下去，最终必将求出正确的 Ans。

在这个数据结构下，修改一个点的权值同样也是 $O(\log^2(N+Q))$。对于一次修改操作，我们需要在不超过 log(N+Q)个线段树节点中的平衡树中删除某个节点，并再另外的不超过 log(N+Q)个线段树节点中的平衡树重新插入这个节点。每次平衡树插入为 O(logN)，所以一次修改的时间复杂度为 $O(\log^2(N+Q))$。

于是按照上面的方案建树，每次询问的时间复杂度为 $O(\log^2(N+Q))$。求解最近公共祖先的算法只要使用每次询问小于 O(logN)的算法就不会影响时间复杂度。总共的询问次数为 Q，所以本算法的时间复杂度是 $O(Q\log^2(N+Q))$。在线段树加平衡树的结构中，每层线段树中的所有节点中的平衡树需要存储的节点数量为 N/2，层数为 $\log_2 N$，所以空间复杂度为 $O(N\log^2 N)$。

值得一提的是，本题相对开放，所以算法并不唯一，上面的算法仅供参考。在应对树上进行操作不方便的问题上，也可以使用树链划分的方法，将树合理的断成很多个链条进行处理。而最后的序列上的询问处理阶段，即使使用每次

$O(\log^3 N)$的传统建树方法，通过一定的优化也能在规定时间内出解。

【总结】

这是一道考察树结构的理解和二分数据结构的设计和实现的难题，最后的测试结果还是相当不错的。在 CTSC2008 的赛场上，总共有 5 位同学在这道题目上获得了满分，另有 1 名同学得到了 90 分，还有不少同学获得了 40～50 分。有些高分的同学虽然使用的是树链划分等理论时间复杂度稍差的方法，但是经过合理的优化，也能够在时限范围内出解。可以看出，在实际的赛场上，如何分配有限的时间，选择正确的算法和进行优化是值得研究的。