《Fibonacci Numbers on Tree》解题报告

厦门双十中学 汪文潇

1 题目来源

Codechef September Challenge 2014

2 题目大意

给定一棵 N 个结点的树，要求在线处理 M 个操作，操作有 4 种：

1、将 x 到 y 路径上第 k 个结点的权值增加 F[k]（指 Fibonacci 数列第 k 项）

2、询问以 x 为根时，以 y 结点为根的子树中所有结点的权值和模 1000000009。

3、询问 x 到 y 路径上所有结点的权值和模 1000000009。

4、将所有结点的权值还原到第 x 个操作后的状态。

3 数据规模及约定

$1 \leq \mathrm{N}, \mathrm{M} \leq 10^5$

4 关键字

数据结构 可持久化 数学

5 解题思路

考虑题目要求，树的形态固定，需要对一条树链进行修改、询问子树和、询问一条树链上的和，以及回到指定过去版本，并且保持在线。

这意味着也许需要实现一个可持久化的树链剖分（或者类似的结构）。

我们先暂时忽略信息的设计、维护这部分，实际上也许这些应该优先考虑。

为了同时支持子树操作和链操作，一个一般做法就是维护这棵树的 dfs 序列，并且在 dfs 时优先访问重边。这样每一条重链就会对应到 dfs 序上的连续的一段区间。

而我们要对树快速地进行子树、链操作，只需要用数据结构维护其 dfs 序。

不失一般性地，这里选用线段树维护。

在这种情况下，可持久化是容易做到的，dfs 序不会改变，那么直接将线段树可持久化即可。这个部分比较经典、常见，就不详细描述了。

那么现在开始关注如何设计、维护线段树上的信息。

这棵线段树需要快速地支持对一个区间内所有数从左往右或从右往左依次加上 Fibonacci 数列从某项开始的连续若干项以及区间求和。

那么怎么描述一个区间上若干次从左往右修改的整体效果呢？（反向同理）

显然，可以描述成从左往右依次加上一个首项为 a，第 2 项为 b，且满足每一项是前 2 项之和的数列。

那么对于线段树上每个区间，记录从左往右、从右往左的各一对 a、b，以及区间内的总和即可。

然而，对于一个修改的整体效果，还要计算它对整个区间内的总和的影响以及对子区间的效果，也就是说，要计算首项为 a，第 2 项为 b，且满足每一项是前 2 项之和的数列的某项以及前若干项之和。

显然，这是一个线性齐次常系数递推，可以用矩阵快速幂在$O(\log n)$的复杂度内求出,但是算上树链剖分的部分之后复杂度将会达到$O(n(\log n)^3)$，应该是不能接受的。

仔细审视这个问题：$F(1)=a, F(2)=b, F(n)=F(n-1)+F(n-2)$。

先考虑是否能求出$F[n]$的通项，下面是一种方法。

考虑递推数列的特征方程$X^2=X+1$,可以解出 2 个解：

$$X_1=\frac{1+\sqrt{5}}{2}, X_2=\frac{1-\sqrt{5}}{2}$$

那么设$F[n]=c{X_1}^n+d{X_2}^n$，则有：

$$\begin{cases} F(1)=cX_1+dX_2=a \\ F(2)=c{X_1}^2+d{X_2}^2=b \end{cases}$$

由此可以解出 c 和 d，然后通过预处理X_1和X_2的次幂，就可以$O(1)$求出要求的$F(n)$。

不难发现，要计算前若干项的和也即$\sum_{i=1}^{n} F(i)$实际上是只需要计算 2 个公比分别为X_1和X_2的等比数列的和，可以通过预处理这 2 个等比数列的前缀和（即X_1和X_2的前若干次幂的和）来$O(1)$得出。

现在唯一的问题就是这题是在模 1000000009 的意义下的，幸运的是，在模 1000000009 的意义下 5 的平方根是存在的，所以可以找对应的X_1和X_2，后续运算也可以顺利在模意义下完成。在模意义下的正确性也是有保证的。

至此，此题得到解决，时空复杂度均为$O(n(\log n)^2)$

6 数据制作

数据制作方面大致与一般同类型的题目相同。

树的形态采取了简单构造加随机的模式。

不同操作种类在不同数据中按不同概率随机出现。

询问方面，由于树的形态保证了随机情况下的期望链长和期望子树大小都不会太小，直接采用了完全随机的方式。

比较特殊的地方是回到指定版本的操作，实际上是影响了版本树，为了防止回到在树上过浅的版本，限定了回到版本的编号，并据此调整了这个操作出现的概率。

特别的是，树链剖分对于非特殊构造的树结构复杂度常数较小。这题的解法中由于可持久化其常数对实际使用的时间和空间都有不小的影响。

经过思考，我认为造一些比较均衡的二叉树来把其复杂度卡到上界以增加实际使用的时间和空间对检验其代码的优劣并无必要，所以并没有加入这样的数据(实际大约能增加约 1 倍内存消耗)。