Counting on a Tree 解题报告

雅礼中学 刘剑成

1 试题来源

Codechef March Challenge 2015 - TREECNT2

2 试题大意

现在有一棵n个节点的树，每条边有一个边权。我们记一条长度大于1的路径的权值为路径中所有边权的最大公约数，输出一共有多少条路径的权值为1。

接下来会有Q次操作，每次操作会修改一条边的边权。对于每次修改你需要回答出在修改之后有多少条路径的权值为1。

$1 \leq N \leq 10^5$，$0 \leq Q \leq 100$，在任意时刻最大边权Z不会超过10^6，且边权均为正整数。

时间限制：2sec

空间限制：无

3 算法介绍

3.1 算法1

依照题意，在每次分别枚举链的两个端点，顺便求出当前链的最大公约数并计算答案。

在每次修改之后重新计算一次即可。

时间复杂度：$O(QN^2 \log Z)$

空间复杂度：$O(N)$

3.2 算法2

由于题目中需要求的是权值为1的链有多少条，所以当一条边包含多个相同的质因子时，我们可以将重复的质因子去掉，即对于一条边的边权为$Z = p_1^{a_1}p_2^{a_2}...p_k^{a_k}$，我们可以将它修改为$Z' = p_1p_2...p_k$。可以发现如果在修改前，若干个数的最大公约数不为1，那么在修改之后，若干个数的最大公约数同样不为1。

使用补集转化，则原问题可以看作有多少条链的每条边都被1整除减去有多少条链的权值是1的倍数。

对于第一部分很好办，答案即为$\binom{N}{2}$。

对于第二部分，可以套用相同的方法，假设当前需要求权值为g的链共有多少条，则可以转化为：有多少条链的每条边都被g整除减去有多少条链的权值是g的倍数且不等于g。

其中，计算有多少条链的每条边都被g整除，我们可以只保留所有被g整除的边，若一条链的每条边均被保留下来，那么它就是合法的。即若一个节点能通过被g整除的边走到另一个节点，那么它们之间的链即为合法的链。

我们用并查集来维护保留下来的边，两个点若在相同的连通块，那么它们之间的路径必定是合法的，所以若一个连通块大小为x，则它对答案的贡献为$\binom{x}{2}$。由于在10^6以内一个数不同的质因数个数不超过7个，所以合并的总次数同样不会超过$2^7 \times N$。

对于每个g都处理完答案之后，我们只需要套用容斥原理即可求出最终答案。

时间复杂度：$O(Q(ND\,\alpha(Z) + Z))$[1]

空间复杂度：$O(ND + Z)$

3.3 算法3

可以发现，每次修改仅仅只会修改一条边的边权，所以我们考虑在询问上是否可以更进一步的优化。

[1]其中D指修改后边权的约数个数

3.3.1 算法3.1

沿着**算法2**的思路，由于每次只会修改一条边的边权，而一个数的约数个数不超过D，所以我们每次无需将所有的森林均重构一遍，只需要将有边修改的森林重构即可。

算法3.1虽然在瓶颈复杂度上并没有任何优化，但却节省了不少常数。

时间复杂度：$O(ND\,\alpha(Z) + Z + QND\,\alpha(Z))$
空间复杂度：$O(ND + Z)$

3.3.2 算法3.2

注意到每次修改受到影响的只有经过这条边的链，所以只需要考虑经过这条边的链在修改之前、修改之后分别有多少条链的权值为1。

对于一条链，我们从当前修改的边将它分为三段。假设三段的最大公约数分别为a, b, c（其中b为当前所选边的权值），由于$gcd(a, b, c)= gcd(gcd(a, b), gcd(b, c))$[2]，那么只需要让$gcd(a, b)$与$gcd(b, c)$互质，整条链的权值即为1。

所以我们对于这条边的两个端点分别进行一次DFS，求出端点在不同的点时$gcd(a, b)$与$gcd(b, c)$的值分别为多少。接下来再枚举$gcd(a, b)$与$gcd(b, c)$的值，当$gcd(a, b)$与$gcd(b, c)$互质时将方案数记入答案即可。

时间复杂度：$O(ND\,\alpha(Z) + Z + Q(N\log Z + D^2))$
空间复杂度：$O(ND + Z)$

对于在Codechef上的数据，我们只需要将**算法3.1**和**算法3.2**结合，当**算法3.1**所需要重构的边数过多时使用**算法3.2**即可通过所有的测试点。

3.4 算法4

虽然**算法3**能够通过Codechef的数据，但这并不是最完美的做法。

观察**算法3.1**，可以发现，我们每次修改一条边的权值，都会将所有的边重新插入并查集，实际上这个过程的大部分操作是多余的。

[2] gcd代表若干个数的最大公约数

在枚举g的时候，对于一个相同的g，它在修改时最多被修改Q条边。所以除了这Q条边以外的所有边都没必要每次重构，只需要在一开始将这些边所连接的点并成一个块，之后每次重构时只需要加入被修改过的边就可以了。

由于修改的总边数是$O(QD)$级别的，所以我们重构Q次的总时间复杂度为$O(Q^2D)$。

时间复杂度：$O(ND\,\alpha(Z) + Z + Q^2D\,\alpha(Z))$
空间复杂度：$O(ND + Z)$

至此，我们已经有了一个靠谱的时间复杂度可以通过此题。

3.5 算法4的再优化

假设原问题中Q的范围继续加大，到了$Q \le 1000$甚至$Q \le 10000$的范围，我们该如何处理？

注意到在时间复杂度中，与Q相关的只有第二部分——即修改时的重构（废话）。观察修改的操作，每次要么是删除一条边，要么是添加一条边，且在添/删边时，整个图一直保持在森林的状态，而查询仅仅只是查询一个连通块的大小。可以发现，这些操作均可以由动态树实现，所以我们在预处理的缩点之后，直接用动态树来做即可。

时间复杂度：$O(ND\,\alpha(Z) + Z + QD\log Q)$
空间复杂度：$O(ND + Z)$