《树上的孤独》解题报告

1.题目描述

1.1 题目大意

有两棵树,小 A 有 n 个节点,小 B 有 m 个节点.每一个节点都有一种颜色,小 A 和小 B 都以一号节点为根,接下来有 Q 次操作.

对于每一次操作先读入一个整数 P 表示操作类型

若 P=1,表示一次询问,那么接下来读入四个整数 P1,P2,P3,P4,我们令 lt 为上一次询问的答案,那么令 D1=P3 ⊕ lt,D2=P4 ⊕ lt,小 A 希望你能回答出小 A 的 P1 子树内离 P1 距离不超过 D1 和小 B 的 P2 子树内离 P2 距离不超过 D2 的节点中一共有多少种不同的颜色

若 P=2,表示小 A 对自己进行了一次修改,接下来读入两个整数 S1,S2,表示小 A 将自己的 S1 节点的颜色修改为了 S2

1.2 输入格式

第一行读入三个整数,n,m,q.

接下来读入 n-1 行，每行两个正整数 u,v，表示有一条边连接小 A 的 u 和 v

然后读入 m-1 行，每行两个正整数 u,v,表示有一条边连接小 B 的 u 和 v

紧接着读入 n 行，每行一个正整数 v1，第 i 行表示小 A 的第 i 号节点的颜色

还需要读入 m 行，每行一个正整数 v2，第 i 行表示小 B 的第 i 号节点的颜色

最后再读入 q 个询问，询问格式见题目描述

1.3 输出格式

对于每一个询问单独输出一行，每行一个整数表示答案

1.4 数据范围

n≤20,m≤2e5,q≤2e6.

P3,P4≤2e5

v1,v2,S2≤m

其中对于前 10%的数据满足 m,q≤2000

对于另外 20%的数据满足不存在 2 操作

还有 30%的数据满足 m,q≤100000

最后还有 10%的数据满足 n=1

2. 解题过程

算法 1

对于每次的询问，遍历两个点的子树，复杂度(n+m)*q,可以获得 10 分的好成绩

算法 2

因为第一棵树上的颜色是不带修的，所以第一棵树上的颜色最多就只有 n 种，所以我们只需要解决掉第二棵树上的答案，然后在特判第一棵树上的颜色是否出现过就可以了

所以我们只需要考虑一棵树的情况

首先我们发现我们要记得是有多少种不同的颜色，因为会有重复，所以颜色不好处理，我们考虑优先将颜色解决掉。

我们会发现，对于一个点，一种颜色，我们并不关心这个点子树里有多少个这种颜色，因为它在正确答案中只会被统计一次，所以我们只关心这个点子树里，这种颜色的最小深度是多少。

既然如此，我们就可以用虚树将它转化为一个简单的深度计数问题

假设我们对颜色为 C 的点建立虚树，虚树上相邻两个点之前的边就对应着原树上的一条链，这条链上的点，他们子树内颜色为 C 深度最浅的点的深度是一样的，也就是说，这种颜色对这条链的贡献是一样的。

而链的数量就是虚树上的点数-1，因为所有颜色虚树点数的总和是 O(N)级别的，所以链的总数是 O（N）级的，所以我们就可以使用数据结构暴力的更新每一条链。

std 使用的方法是线段树合并，首先对树上的每一个点建立以深度为下标的虚空线段树，然后对于一条链，我们在链底节点的深度线段树加上这种颜色的贡献，在链顶节点的父亲那里减去这种颜色的贡献。最后对于所有的点进行线段树合并，这样可以保证每一种颜色在一个节点只会被统计一次，与处理完，对于询问，我们只需要对一个节点的线段树区间求和就可以了。

算法 3

算法 3 比算法 2 多了修改，但是这个修改并没有起到很大的作用，因为我们发现第一棵树的大小只有 20，对于每一次询问，我们是可以暴力去考虑第一棵树的贡献的，所以问题就回到了算法二.

然后我们来考虑第一棵树，对于第一棵树的一个点，我们只关心它在第二棵树中是不是已经统计过了，也就是说，我们需要知道这个颜色在第二棵树子树中的最小深度。

这样得到了 n*q 个新的询问，这里可以把同一种颜色的点按照 dfs 序排序，然后我们询问的子树就是一段 dfs 序区间，对应到这种颜色上就是一段区间，然后就是一个区间求深度最小值，这里可以用 RMQ 实现 O（1）查询，但是我们查出这个点子树对应的区间的左右端点受在线的限制，还是只能带一个 log，也就是说在在线的限制下，出题人也只能以（n*q*log(m)+m*log(m)）的复杂度解决此题，这可以获得 70 分的好成绩

算法 4

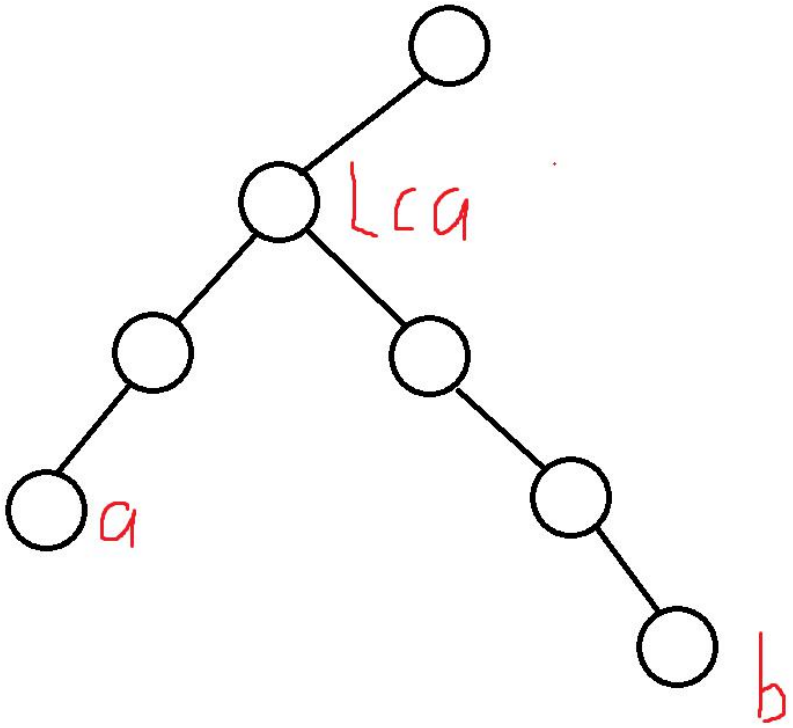
其实这个题并没有强制在线，它是一个假的强制在线，出题人只对每次询问的深度做出了强制在线的要求，也就是说每次询问的点是那个，你是知道的，可以从这里离线入手

那么我们用离线来解决算法三中的 log，我们回到算法三的询问中去，我们对于同一种颜色的点建立虚树，这里我们定义虚树上不是这种颜色的点的深度为它子树内这种颜色的点的最小深度

然后能发现一种性质，它子树区间内深度最小的点，其实就是虚树上在它子树内 dfs 序最小的点。

接下来用反证法给出证明:

设 dfs 序最小的点为 a,深度最小的点为 b，若 b!=a,那么它们必然存在着一个深度<=b 且 dfs 序更小的 LCA。与假设不符合



所以我们对询问的颜色，只需要找出 dfs 序>=询问的点的 dfs 序最小的点就可以了

所以我们把我们 n*q 个询问都挂在对应的点上，然后第二棵树的点按照 dfs 序排序然后一个个遍历过来，当遍历到一个点的时候，更新它有影响的所有颜色，然后 o(1)回答每一个询问

这样就可以用(n*q+m*log(n))的复杂度通过此题

3. 总结

比较繁琐的数据结构题，比较难想到的是题目对询问的部分内容进行了强制在线，但是题解做法仍然是离线。

参考资料

https://www.codechef.com/JUNE20A/problems/DIFVAL