T1 - Output Only 题解

题目简述

题目大意：
给定一棵 N 个节点的有根树，根为 1。每个节点有一个初始颜色 $c_i \in [0, k-1]$。
操作：选择一个节点 u 和颜色 p，将 u 及其子树内所有节点的颜色 $c_v \leftarrow (c_v + p) \pmod k$。
目标：通过若干次操作，使所有节点颜色变为 0。
题目要求在发生 Q 次单点颜色修改 (w_i, x_i) 后，分别求出达成目标所需的最少操作次数。

- $N, Q, k \leq 2 \times 10^5$。

The Key：
利用 **差分** 的思想，将子树修改转化为对“父子颜色关系”的维护。这是一个典型的树上差分或转化贡献维度的思维题。

子任务

Subtask 1：树是一条链

当树是一条链（例如 $1 \to 2 \to \cdots \to N$）时，我们可以利用序列差分的思想。
定义 $d_i = (c_i - c_{i-1}) \pmod k$，其中 $c_0 = 0$。
对于以 u 为根的子树（链上的后缀）进行操作，只会改变 d_u 的值，而不会影响 $d_{u+1} \dots d_N$。
为了让所有 $c_i = 0$，等价于让所有差分值 $d_i = 0$。
如果某位置 $d_i \neq 0$，我们就必须在 i 处进行一次操作来修正它。
因此答案为 $\sum_{i=1}^{N} [d_i \neq 0]$。

时间复杂度：$\mathcal{O}(N + Q)$

Subtask 2：$k = 2$

当只有两种颜色时，问题可以简化为统计**坏边**。
考虑一条边 (u, v)，其中 u 是 v 的父节点。
如果在 v 处不进行任何操作（指以 v 为根的操作），那么 v 的颜色变化量将完全取决于 u 的变化量。即操作后 $c'_v - c'_u \equiv c_v - c_u \pmod k$。

要使最终 $c'_v = c'_u = 0$，必须满足 $c_v - c_u \equiv 0$。
如果初始 $c_v \neq c_u$，我们就必须在 v 处进行一次操作（修正 v 相对于 u 的颜色偏差）。
对于 $k = 2$，这等价于统计两端颜色不同的边数。注意根节点需要单独判断（视作父节点颜色为 0）。

时间复杂度：$\mathcal{O}(N + Q)$

Subtask 4：$N, Q \leq 5 \times 10^4$

对于一般情况，如果不使用高级数据结构，可以考虑 **根号分治**。
按照节点的度数 $\deg$ 分为大点和小点。

- 修改小点时，直接暴力遍历其所有相邻边，更新答案。
- 修改大点时，更新大点的信息。
- 维护大点与大点之间的关系。

时间复杂度：$\mathcal{O}(Q\sqrt{N})$，可通过此子任务。

正解

根据我们对上面的性质的探索，我们可以显然的得到这样的性质：

> 本题等价于维护树上**两端颜色不同的边数**。
>
> $$Ans = [c_{\text{root}} \neq 0] + \sum_{(u,v)\in E} [c_u \neq c_v]$$

但是我们需要动态维护这个值，对于一个点 u 来说：

- **父边贡献**：当其 $c_u \neq c_{fa}$ 的时候，贡献加 1
- **子边贡献**：我们用 $cnt[u][col]$ 来维护 u 中颜色为 col 的数量，则 u 对子边的贡献就是 $S_u - cnt[u][c_u]$

动态的维护即可。

时间复杂度：$\mathcal{O}(N \times \log N + Q \times \log N)$

```cpp
//官方 std
#include <bits/stdc++.h>
#define uwu return 0;

```

```cpp
using namespace std;

const int SIZE = 2e5 + 5;

#define fs first
#define sc second

map <int, int> amnt_by_color[SIZE];

vector<int> graph[SIZE];

int color[SIZE], pa[SIZE], son_amnt[SIZE];

void dfs(int nd, int rt){
    pa[nd] = rt;
    for(auto i:graph[nd]){
        if(i == rt)
            continue;
        amnt_by_color[nd][color[i]]++;
        son_amnt[nd]++;
        dfs(i, nd);
    }
    return;
}

int main(){
    cin.tie(0), ios::sync_with_stdio(0);
    int N, k, Q;
    cin >> N >> k >> Q;
    for (int i = 1; i ≤ N; i++){
        cin >> color[i];
    }
    for (int i = 1, u, v; i < N; i++){
        cin >> u >> v;
        graph[u].push_back(v);
        graph[v].push_back(u);
    }
    dfs(1, 0);
    int different_edge = (color[1] == 0 ? 0 : 1);
    for (int i = 1; i ≤ N; i++){
        for(auto j:amnt_by_color[i]){
            if(j.fs ≠ color[i])
                different_edge += j.sc;
```

```
        }
    }
    for (int w, x; Q--;){
        cin >> w >> x;
        different_edge -= (color[pa[w]] ≠ color[w]);
        different_edge -= (son_amnt[w] - amnt_by_color[w]
[color[w]]);
        amnt_by_color[pa[w]][color[w]]--;
        color[w] = x;
        different_edge += (color[pa[w]] ≠ color[w]);
        different_edge += (son_amnt[w] - amnt_by_color[w]
[color[w]]);
        amnt_by_color[pa[w]][color[w]]++;
        cout << different_edge << '\n';
    }
    uwu;
}
```