暴力启发式合并 保留重儿子 优化空间

有一颗树，根节点为 1

有 n 个节点，每个节点都有不同的颜色

问每一个节点，以该节点为根的子树中出现最多次的颜色有哪些?

先计算出所有轻儿子的答案

然后计算并保留重儿子子树的信息

接着暴力遍历所有轻儿子的子树，将信息保留下来

如果遍历轻儿子的子树呢?

可以选择再一次 `dfs` 也可以选择树链剖分。

时间复杂度：$O(n\log(n))$

参考优质博主的题解

```
#include <bits/stdc++.h>

const long long N = 2e6;
long long n;
long long a[N];
long long g[N], e[N], ne[N], ant;
long long hson[N], size[N];
long long cnt_col[N];
long long sum_col, max_cnt;
long long ans[N];

void add(long long x, long long y)
{
    ant ++;
    e[ant] = y;
    ne[ant] = g[x];
    g[x] = ant;
}

void dfs_1(long long u, long long v)
{
    size[u] = 1;
    for (long long i = g[u]; i; i = ne[i])
    {
        if (e[i] == v) continue;
        dfs_1(e[i], u);
        size[u] += size[e[i]];
        if (hson[u] == 0 || size[hson[u]] < size[e[i]])
```

```
        {
            hson[u] = e[i];
        }
    }
}

// 将点 u 的颜色加进去会有什么影响?
void add(long long u)
{
    cnt_col[a[u]] ++;
    if (cnt_col[a[u]] > max_cnt)
    {
        max_cnt = cnt_col[a[u]];
        sum_col = a[u];
    }
    else if (cnt_col[a[u]] == max_cnt)
    {
        sum_col += a[u];
    }
}

// 将点 u 的颜色删除
void del(long long u)
{
    cnt_col[a[u]] --;
}

// 将以 u 为根的子树的所有颜色加进去
void add_tree(long long u, long long v)
{
    add(u);
    for (long long i = g[u]; i; i = ne[i])
    {
        if (e[i] == v) continue;
        add_tree(e[i], u);
    }
}

// 将以 u 为根的子树的所有颜色拿出来
void del_tree(long long u, long long v)
{
    del(u);
    for (long long i = g[u]; i; i = ne[i])
    {
        if (e[i] == v) continue;
        del_tree(e[i], u);
    }
}

void dfs_2(long long u, long long v, bool keep)
{
    for (long long i = g[u]; i; i = ne[i])
    {
        if (e[i] == v || e[i] == hson[u]) continue;
```

```
        dfs_2(e[i], u, 0);
    }

    if (hson[u]) dfs_2(hson[u], u, 1);
    add(u);

    for (long long i = g[u]; i; i = ne[i])
    {
        if (e[i] == v || e[i] == hson[u]) continue;
        add_tree(e[i], u);
    }

    ans[u] = sum_col;

    if (keep == false)
    {
        del_tree(u, v); max_cnt = 0; sum_col = 0;
    }
}

void solve()
{
    std::cin >> n;
    for (long long i = 1; i <= n; i ++) std::cin >> a[i];
    for (long long i = 1; i < n; i ++)
    {
        long long x, y; std::cin >> x >> y;
        add(x, y); add(y, x);
    }
    dfs_1(1, 0);
    dfs_2(1, 0, 1);
    for (long long i = 1; i <= n; i ++)
    {
        std::cout << ans[i] << ' ';
    }
    std::cout << '\n';

}

int main()
{
    std::ios::sync_with_stdio(0);
    std::cin.tie(0); std::cout.tie(0);

    solve();
    return 0;
}
```