树上启发式合并

启发式合并大致的思路就是小的被并入大的里面。

树上启发式合并引入了树剖中重儿子的思想，可以想到的是如果一个子树中有更多的节点那么别的子树并入他那么时间复杂度更加低。

具体的操作方式可以这样说。首先预处理得到每个节点的重儿子，接下来分别遍历所有轻儿子并对其子树求出对应的解，返回的时候需要将轻儿子的信息删除，然后是再遍历重儿子并求出对应的解但不清空数据。

最后再遍历轻儿子并将所有信息暴力加入。

例题（$Codeforces\ 600E$）

题目大意

一个树每个节点有一种颜色，将每个节点为根节点的子树中出现最多的颜色的编号相加，并逐个输出。

题解

维护每个子树中出现最多的颜色出现的次数就行了，再套树上启发式合并板子就行了。

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
signed main(){
#ifdef ACM_LOCAL
    freopen("data.in", "r", stdin);
    freopen("data.out", "w", stdout);
#endif
    ios_base::sync_with_stdio(false);
    cin.tie(0), cout.tie(0);

    int n; cin >> n;
    vector<int> c(n + 1);
    for(int i = 1;i <= n;i ++) cin >> c[i];
    vector<vector<int>> g(n + 1);
    for(int i = 1;i < n;i ++){
        int u, v;
        cin >> u >> v;
        g[u].push_back(v);
        g[v].push_back(u);
    }

    vector<int> maxson(n + 1), siz(n + 1);
    auto dfs1 = [&](auto self, int fa, int x) -> void {
        siz[x] = 1;
        for(auto v : g[x]){
            if(v == fa) continue;
            self(self, x, v);
            siz[x] += siz[v];
            if(siz[maxson[x]] < siz[v]) maxson[x] = v;
        }
        return ;
    };
    dfs1(dfs1, 0, 1);

    vector<int> res(n + 1);
    vector<int> cnt(n + 1); int ans = 0, maxi = 0;
```

```
    auto dfs2 = [&](auto self, int fa, int x, int op) -> void {
        for(auto v : g[x]){
            if(v == fa || v == maxson[x]) continue;
            self(self, x, v, 0);
        }
        if(maxson[x]){
            self(self, x, maxson[x], 1);
        }
        // 计算其本身
        cnt[c[x]] ++;
        if(cnt[c[x]] > maxi){
            maxi = cnt[c[x]];
            ans = c[x];
        }else if(cnt[c[x]] == maxi){
            ans += c[x];
        }
        for(auto v : g[x]){
            if(v == fa || v == maxson[x]) continue;
            auto add = [&](auto self, int fa, int node, int x) -> void {
                cnt[c[node]] ++;
                if(cnt[c[node]] > maxi){
                    maxi = cnt[c[node]];
                    ans = c[node];
                }else if(cnt[c[node]] == maxi){
                    ans += c[node];
                }
                for(auto v : g[node]){
                    if(v == fa) continue;
                    self(self, node, v, x);
                }
                return ;
            };
            add(add, x, v, x);
        }
        res[x] = ans;
        if(op) return ;
        ans = maxi = 0;
        auto del = [&](auto self, int fa, int x) -> void {
            cnt[c[x]] --;
            for(auto v : g[x]){
                if(v == fa) continue;
                self(self, x, v);
            }
        };
        del(del, fa, x);
        return ;
    };
    dfs2(dfs2, 0, 1, 0);
    for(int i = 1;i <= n;i ++) cout << res[i] << " ";

    return 0;
}
```