

树上启发式合并

DSU on Tree

Idea: 如果暴力做, 我们会整体做一次 dfs 以遍历每个点, 然后对遍历的每个点 dfs 它的子树, 计算答案, 把答案记录抹去, 算下一个点。这样做是 $O(n^2)$ 的。为优化, 我们遍历到一个点时, 先遍历它的轻儿子, 计算轻儿子的答案并抹去记录, 最后计算它的重儿子的答案且不抹去记录, 如此在计算它本身的答案时, 只需要暴力算出轻儿子的答案, 合并上原本就存在的重儿子答案即可。

代码中的 `dsu()` 函数对应暴力算法的整体 dfs, `getData()` 函数对应暴力算法的“对遍历的每个点 dfs 计算子树答案”。

Complexity: $O(n \lg n)$

Code:

```
1  int fa[N], sz[N], son[N];
2  void dfs(int x, int f){
3      fa[x] = f, sz[x] = 1, son[x] = 0;
4      for(int i = head[x]; i; i = edge[i].nxt){
5          if(edge[i].to == f) continue;
6          dfs(edge[i].to, x);
7          sz[x] += sz[edge[i].to];
8          if(!son[x] || sz[edge[i].to] > sz[son[x]])
9              son[x] = edge[i].to;
10     }
11 }
12
13 LL ans[N], mx, sum, cnt[N]; // GLOBAL variants to store the answer
14 int mark; // mark the heavy son which needs to be ignored
15 void getData(int x, int val){ // get data with brute-force
16
17     cnt[c[x]] += val;
18     if(mx < cnt[c[x]]) mx = cnt[c[x]], sum = c[x];
19     else if(mx == cnt[c[x]]) sum += c[x];
20
21     for(int i = head[x]; i; i = edge[i].nxt){
22         if(edge[i].to == fa[x]) continue;
23         if(edge[i].to == mark) continue; // ignore the marked subtree
24         getData(edge[i].to, val);
25     }
26 }
27 void dsu(int x, bool opt){ // opt == true: answer needs to be erased
28     for(int i = head[x]; i; i = edge[i].nxt){
29         if(edge[i].to == fa[x] || edge[i].to == son[x]) continue;
30         dsu(edge[i].to, true); // solve for light sons first
31     }
32     if(son[x]) dsu(son[x], false), mark = son[x]; // solve for heavy son
33     // now the global variants have already stored heavy son's answer
34     getData(x, 1);
35     mark = 0;
36
37     // now the global variants store the answer for vertex x
38     ans[x] = sum;
39
40     if(opt){ // erase the answer
41         getData(x, -1);
42         mx = 0, sum = 0;
43     }
44 }
45
46 int main(){
47     scanf("%d", &n);
48     for(int i = 1; i <= n; i++)
49         scanf("%d", &c[i]);
50     for(int i = 1; i < n; i++){
51         int u, v; scanf("%d%d", &u, &v);
52         addEdge(u, v), addEdge(v, u);
53     }
54     dfs(1, 0);
55     dsu(1, true);
56     for(int i = 1; i <= n; i++)
57         printf("%lld ", ans[i]);
58     return 0;
```

