

C++算法

快快编程1834

快快编程
kkcoding.net

树上问询经典思路有哪些

序列化+数据结构

树链剖分+数据结构

点分治

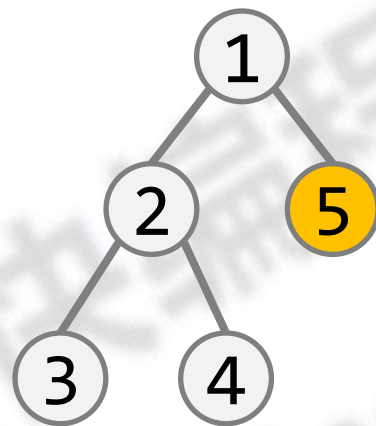
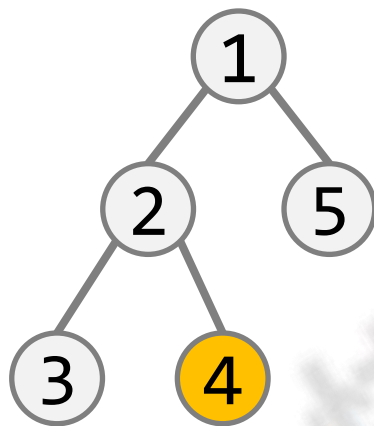
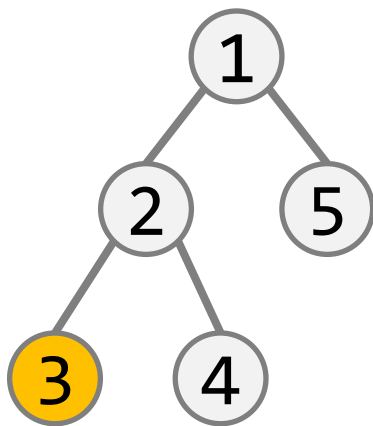
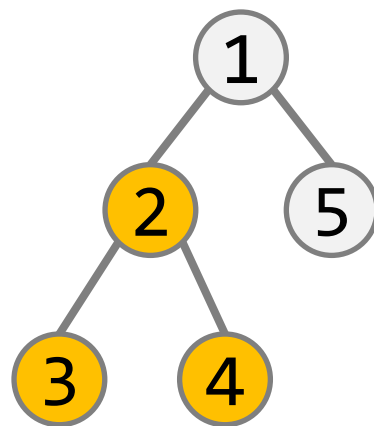
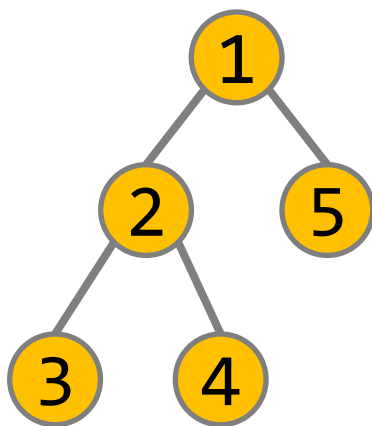
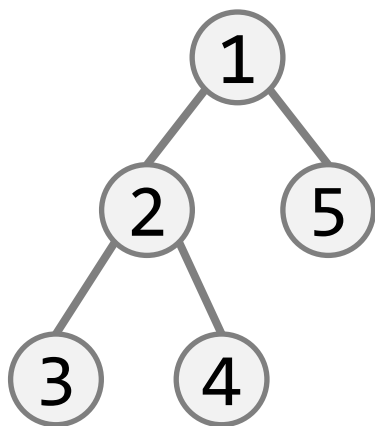
LCT

2种思维路径

暴力+优化

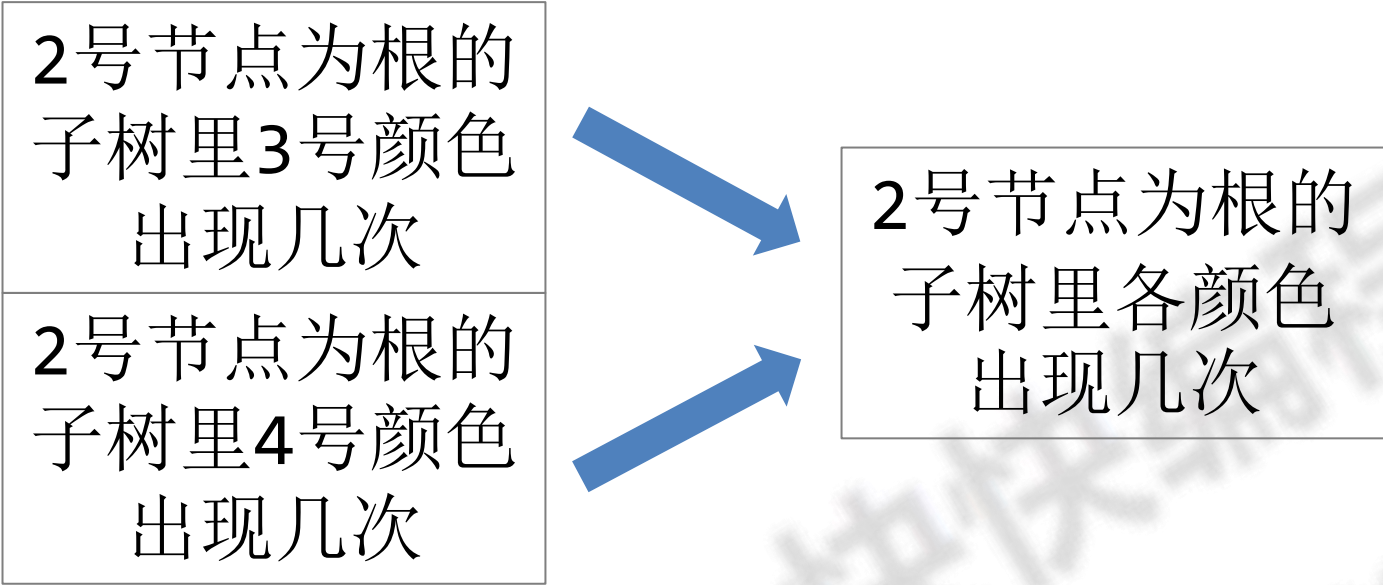
树简化为链

从暴力解法开始
识别计算重复部分
识别时间浪费在哪



快快1834
kkcoding.net

树上离线询问可以根据节点分类



树上离线询问可以根据节点分类

```
7 struct query{int id,c};
8 vector<query> q[N];

48     scanf("%d",&m);
49     for(int i=1;i<=m;++i){
50         int u,c;
51         scanf("%d %d",&u,&c);
52         q[u].push_back((query){i,c});
53     }
```


addSubTree()三个参数什么含义
为什么需要删除信息?

```
30 void dfs(int u, int fa){
31     if(q[u].size()){
32         addSubTree(u, fa, 1);
33         for(int i=0; i<q[u].size(); ++i)
34             ans[q[u][i].id] = cnt[q[u][i].c];
35         addSubTree(u, fa, 0);
36     }
37     for(int i=hd[u]; i; i=nxt[i])
38         if(to[i] != fa) dfs(to[i], u);
39 }
```

37, 38行能否和31-36交换位置?
可以。两种顺序有没有优劣之分

暴力 离线 询问

addSubTree()递归遍历子树

addNode()处理单点

快快1834

dfs套dfs
实现暴力
60分

```
21 void addNode(int u, bool tag){
22     if(tag)++cnt[clr[u]];
23     else --cnt[clr[u]];
24 }
25 void addSubTree(int u, int fa, bool tag){
26     addNode(u, tag);
27     for(int i=hd[u]; i; i=nxt[i])
28         if(to[i]!=fa) addSubTree(to[i], u, tag);
29 }
```

暴力
离线
询问

addSubTree()序列中遍历子树
addNode()处理单点

快快1834

序列化
实现暴力
70分

```
21 void addNode(int u, bool tag){
22     if(tag)++cnt[clr[u]];
23     else --cnt[clr[u]];
24 }
25 void addSubTree(int u, bool tag){
26     for(int i=tI[u];i<=tO[u];++i)
27         addNode(id[i],tag);
28 }
```

暴力 离线 询问

tI[u]是u子树在dfs序开始位置
tO[u]是u子树在dfs序结束位置

快快1834

序列化
实现暴力
70分

```
12 void dfs_tIO(int u, int fa){
13     id[tI[u]=++timer]=u;
14     for(int i=hd[u]; i; i=nxt[i]){
15         int v=to[i];
16         if(v==fa) continue;
17         dfs_tIO(v, u);
18     }
19     tO[u]=timer;
20 }
```

暴力解法
复杂度分析

对于随机树
高度平均 $O(\log n)$
暴力平均复杂度 $O(n \log n)$


最差情况是链状
暴力复杂度 $O(n^2)$

树简化为链
目标降到 $O(n)$ 或 $O(n \log n)$
或 $O(n \sqrt{n})$

编程
kkcoding.net

目标降到 $O(n)$ 或 $O(n\log n)$
或 $O(n\sqrt{n})$

思路路径

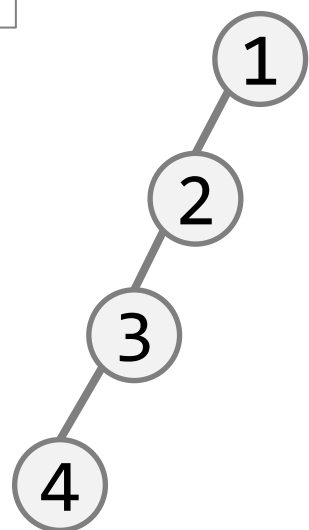


树简化为链
黑白两色

树上 黑白两色
链上 多种颜色

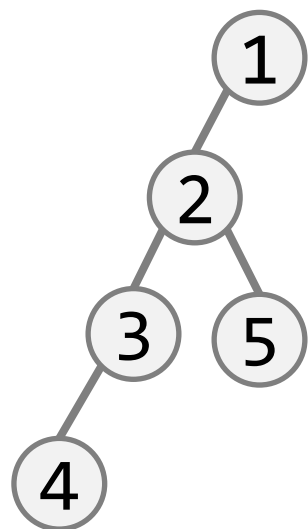
树上
多种颜色

快快编程
kkcoding.net

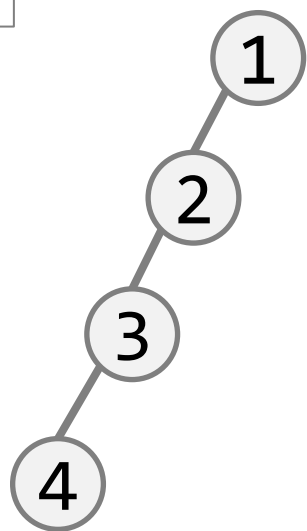


发现:从叶到根计算顺序
可以累积信息

能否推广到树形?
加一点点树形



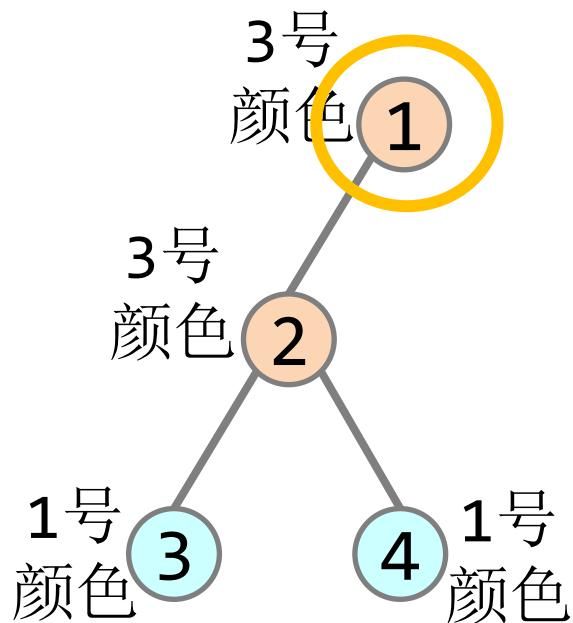
涉及到两条链的信息合并
这个思路能够走通
留给同学之后思考
回到这个思路起点
识别思路盲区



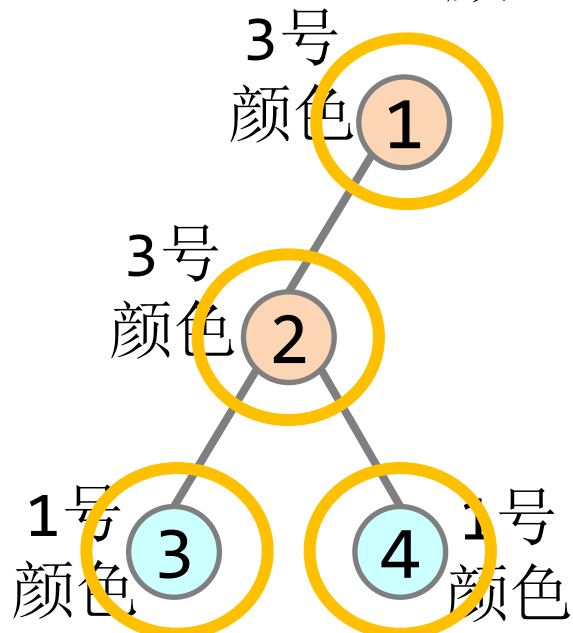
发现:从叶到根计算顺序
可以累积信息

这个发现引导我们思考 叶到根的顺序	DFS 回溯序
同时也让我们忽略了 根到叶的顺序	
根到叶的访问 能否带来有用信息	DFS 访问序

关于 $u=2$ 的所有问题,需要参考两个状态的信息
做差就是以 $u=2$ 为根的子树信息



cnt[1]	0
cnt[2]	0
cnt[3]	1
cnt[4]	0



cnt[1]	2
cnt[2]	0
cnt[3]	2
cnt[4]	0

进出子树的
两个状态"做差"

```
12 void dfs(int u, int fa){  
13     for(int i=0; i<q[u].size(); ++i)  
14         ans[q[u][i].id] = -cnt[q[u][i].c];  
15  
16  
17  
18     for(int i=0; i<q[u].size(); ++i)  
19         ans[q[u][i].id] += cnt[q[u][i].c];  
20 }
```

复杂度 $O(n)$

讨论

`cnt[]`是个共享数组

`cnt[]`数组为什么用一维数组
而不采用二维数组`cnt[u][c]`

快快编程1835

快快编程
kkcoding.net

有根树上 n 个节点
节点 u 的颜色为 $clr[u]$
共 m 个询问：
节点 u 子树内共有几种颜色

2种思维路径

暴力+优化

树简化为链

2种思维路径

暴力对于大部分
随机数据 $O(n\log n)$

树简化为链
发现叶到根累积信息
需要合并子树

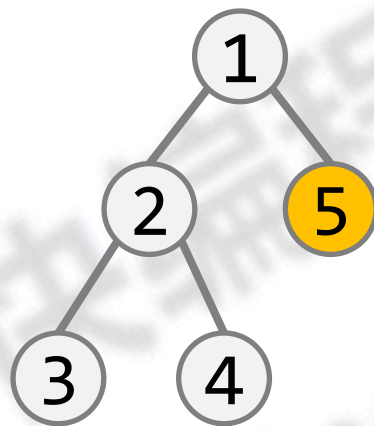
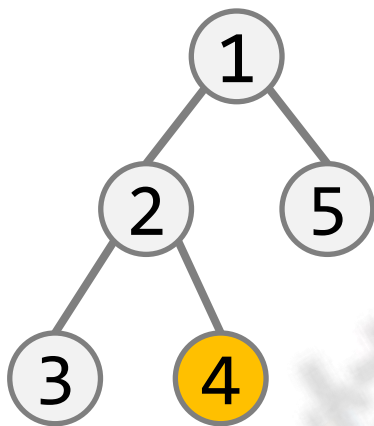
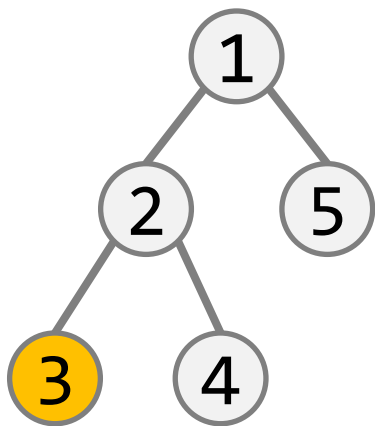
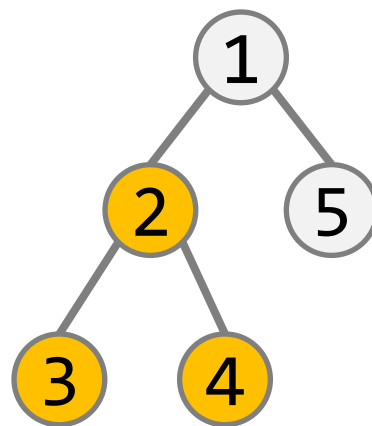
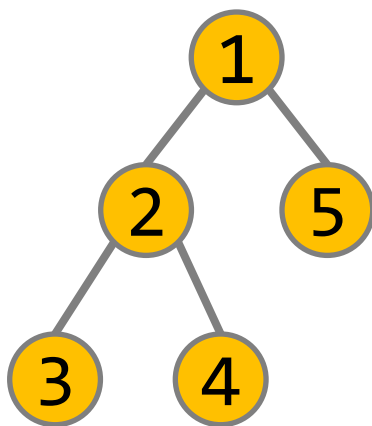
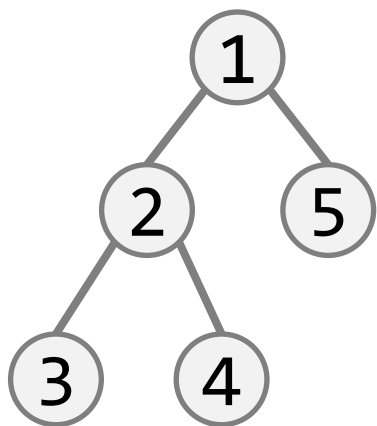
2种思维路径

暴力

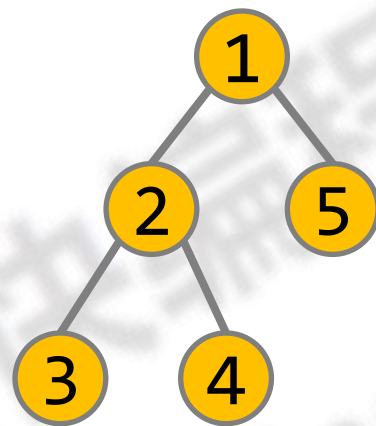
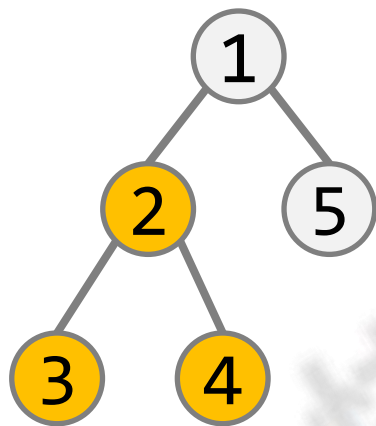
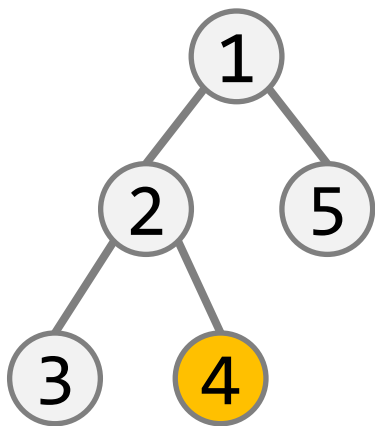
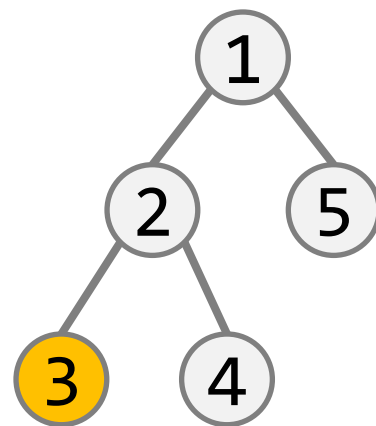
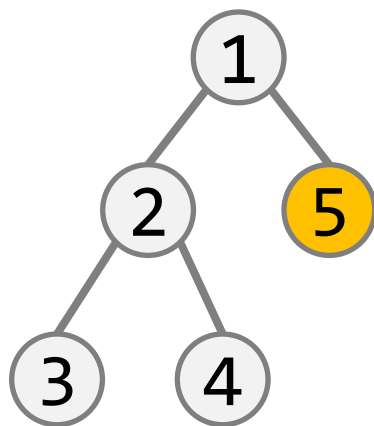
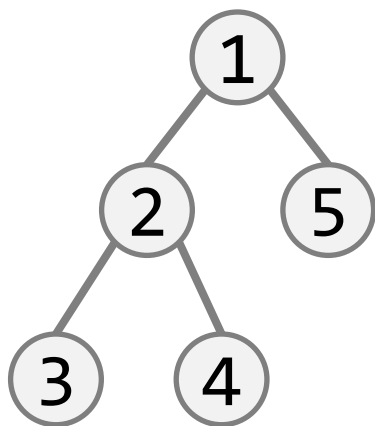


小并大

快快编程
kkcoding.net



快快1835
kkcoding.net



贪心	轻儿子们和根合并入重儿子	51并入234
----	--------------	---------

对于u节点的子树问询

先将u所有轻儿子们的 子树问询完成解答	因内存不够 解答后需清除
------------------------	-----------------

再将u的唯一重儿子的 子树问询完成解答	解答后 无需清除
------------------------	-------------

保留u的唯一重儿子的子树信息 将u所有轻儿子们的子树信息并入
添加u节点的信息 形成u节点子树的完整信息

kkcoding.net

```
48 void input(){
49     scanf("%d",&n);
50     for(int i=1;i<=n-1;++i){
51         int u,v;
52         scanf("%d %d",&u,&v);
53         addedge(u,v);
54         addedge(v,u);
55     }
56     for(int u=1;u<=n;++u)scanf("%d",&clr[u]);
57     scanf("%d",&m);
58     for(int i=1;i<=m;++i)scanf("%d",&q[i]);
59 }
60 void solve(){
61     dfs_sz_son(1,0);
62     dfs(1,0,1);
63     for(int i=1;i<m;++i)printf("%d ",ans[q[i]]);
64     printf("%d\n",ans[q[m]]);
65 }
```

```
10 void dfs_sz_son(int u,int fa){
11     sz[u]=1;
12     son[u]=0;
13     id[tI[u]=++timer]=u;
14     for(int i=hd[u];i;i=nxt[i]){
15         int v=to[i];
16         if(v==fa)continue;
17         dfs_sz_son(v,u);
18         sz[u]+=sz[v];
19         if(sz[son[u]]<sz[v])son[u]=v;
20     }
21     t0[u]=timer;
22 }
```

```
31 void dfs(int u, int fa, bool hvy){
32     for(int i=hd[u]; i; i=nxt[i]){
33         int v=to[i];
34         if(son[u]==v || fa==v) continue;
35         dfs(v, u, 0);
36     }
37     if(son[u]) dfs(son[u], u, 1);
38     for(int i=hd[u]; i; i=nxt[i]){
39         int v=to[i];
40         if(son[u]==v || fa==v) continue;
41         aTree(v, 1);
42     }
43     aNode(u, 1);
44     ans[u]=nClr;
45     if(hvy) return;
46     aTree(u, 0);
47 }
```

完成轻儿子们
子树询问

完成重儿子
子树询问
保留信息

轻儿子们
子树信息都并入

u单点信息并入

回答u的询问

若u是轻儿子,清除
u子树所有信息

```
23 void aNode(int u, bool tag){  
24     if(tag) nClr+=(++cnt[clr[u]]==1);  
25     else    nClr--=(--cnt[clr[u]]==0);  
26 }  
27 void aTree(int u, bool tag){  
28     for(int i=tI[u]; i<=tO[u]; ++i)  
29         aNode(id[i], tag);  
30 }
```

共m个询问：
节点u子树内共有几种颜色

推荐名称	小并大
中文名称	启发式合并
英文名称	DSU on tree


```

8 void dfs(ll u, ll fa, bool hvy){
9     //回答u的所有轻儿子子树询问, 递归调用dfs()
10
11     //完成u的重儿子子树询问, 保留重儿子子树信息
12
13     //将轻儿子们的子树信息都并入aTree(增加)
14
15     //将u的单点信息都并入aNode()
16
17     //回答u的询问
18
19     //若u是轻儿子, 清除u子树所有信息aTree(清除)
20 }

```

小并大/启发式合并

暴力
优化

发现可重复利用的信息

暴力顺序的调整

贪心：尽量多保留信息
选重儿子最后访问

小并大/启发式合并

最差情况 $O(n \log n)$

链状

最优情况 $O(n)$

针对朴素暴力
缺点的改进

请写出证明

点 v 在每次并入他人时访问1次
 v 到根路径上轻边数 $O(\log n)$
即 v 并入他人最多 $O(\log n)$ 次