

太戈编程
etiger.vip

信奥算法

1824

判断u是不是v的直系祖先

区间
包含

判断 区间 $[tI[u], tO[u]]$
是否包含 区间 $[tI[v], tO[v]]$

```
15 bool up(int u, int v){  
16     return tI[u] <= tI[v] && tO[v] <= tO[u];  
17 }
```

u==v情况也包含了
能应对特殊易错点

```
21 cin>>n;
22 for(int i=1;i<n;++i){
23     int u,v;
24     cin>>u>>v;
25     to[u].push_back(v);
26     to[v].push_back(u);
27 }
28 for(int u=1;u<=n;++u)
29     sort(to[u].begin(),to[u].end());
30 timer=0;
31 dfs(1,0);
```

```
3  const int N=1009;
4  vector<int> to[N];
5  int n,m;
6  int tI[N],tO[N],euler[N*2],timer;
7  void dfs(int u,int fa){
8      tI[u]=++timer;
9      euler[timer]=u;
10     for(int i=0;i<to[u].size();++i)
11         if(to[u][i]!=fa) dfs(to[u][i],u);
12     tO[u]=++timer;
13     euler[timer]=u;
14 }
```

1795

单源距离总和

树上找**1**个节点，到其他点距离总和最大

算法1

对于每个节点u作为根
DFS计算其他节点的深度 $d_u[]$
深度减**1**就是到根的距离

复杂度
 $O(n^2)$

算法2

先让**1**号节点作为根
算**1**号到其他点距离总和 $f[1]$

目标复杂度
 $O(n)$

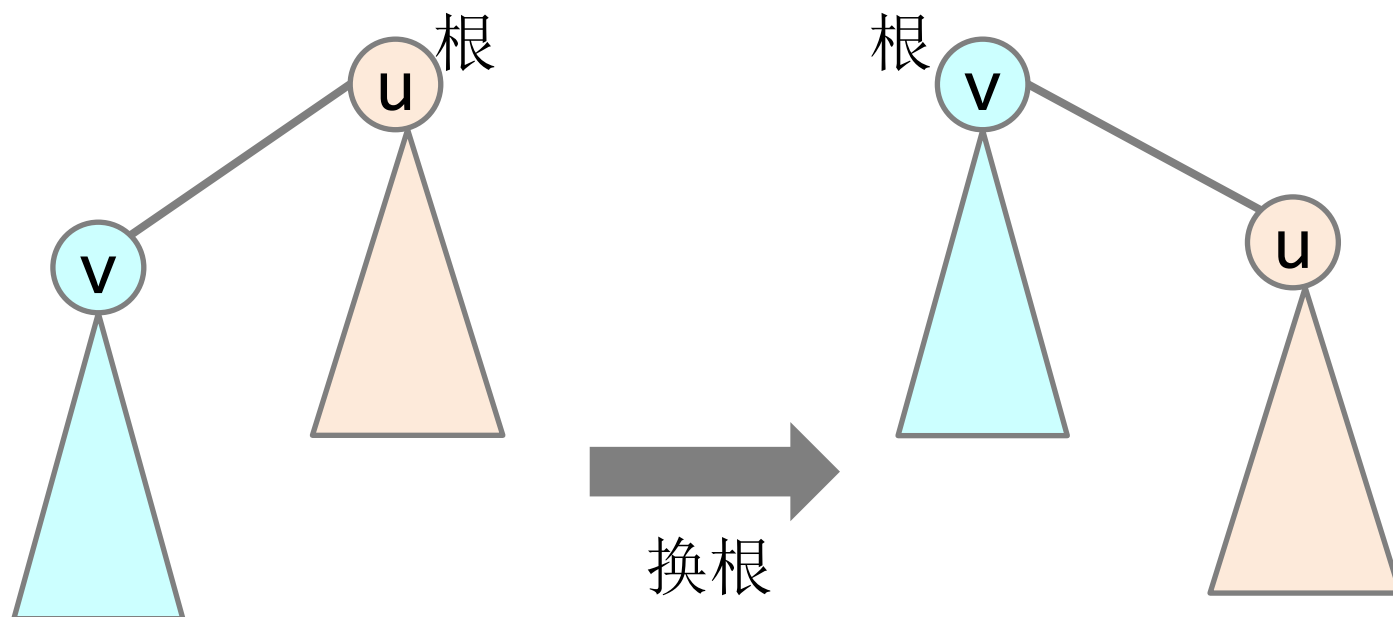
快速换根

记笔记

根节点换成其他节点u时
希望不用重新DFS
而是利用已知的基础信息
快速推导 $f[u]$

$f[u]$ 代表以 u 为起点到其他点距离的总和

$f[v]$ 代表以 v 为起点到其他点距离的总和

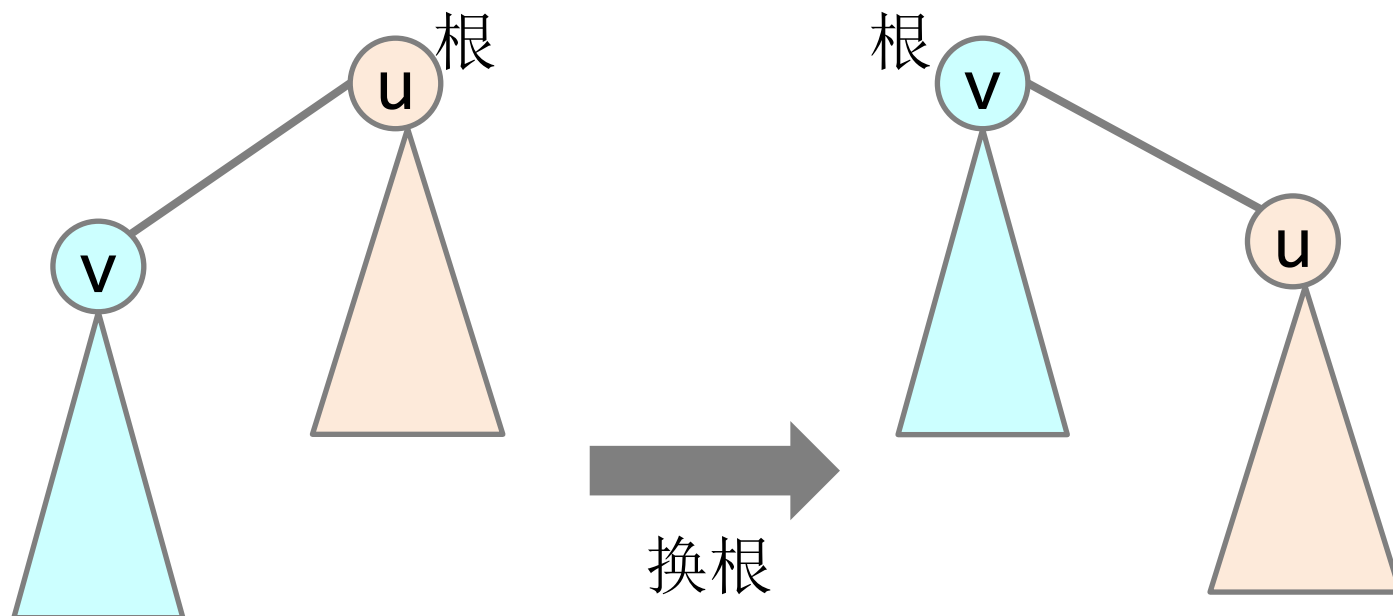


已知 $f[u]$ ，能否推导出 $f[v]$

蓝节点	到根距离都减少1	共 $sz[v]$ 个
红节点	到根距离都增加1	共 $n - sz[v]$ 个

$f[u]$ 代表以 u 为起点到其他点距离的总和

$f[v]$ 代表以 v 为起点到其他点距离的总和



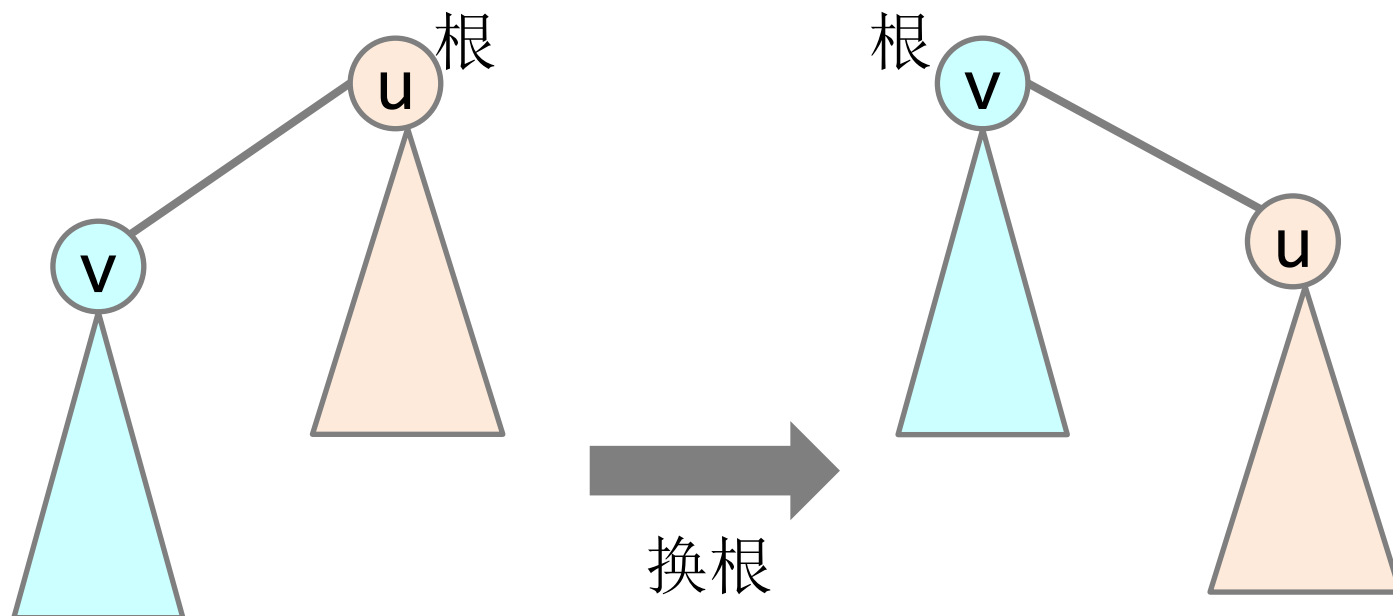
$$f[v] = f[u] - sz[v] + (n - sz[v])$$

$sz[v]$ 代表的子树大小是换根前的还是换根后的?

换根后 $sz[]$ 数组是否要重新计算?

$f[u]$ 代表以 u 为起点到其他点距离的总和

$f[v]$ 代表以 v 为起点到其他点距离的总和



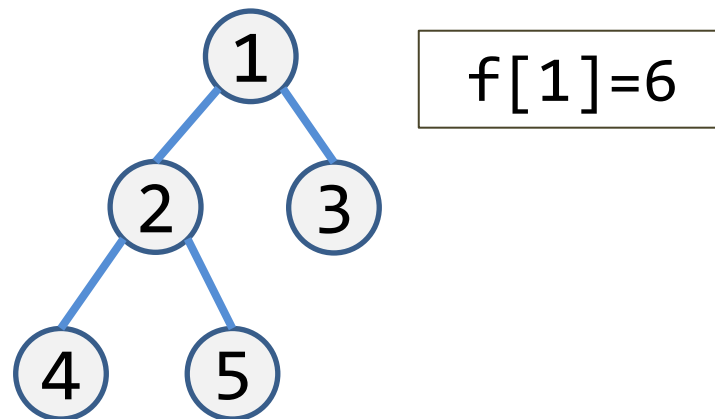
$$f[v] = f[u] - sz[v] + (n - sz[v])$$

$sz[v]$ 代表的子树大小是换根前的

换根后 $sz[]$ 数组不需要重新计算

DFS顺序换根:总是父亲 u 换给儿子 v , $sz[v]$ 准确

假想"换根"效果
程序并不用真的修改父子关系

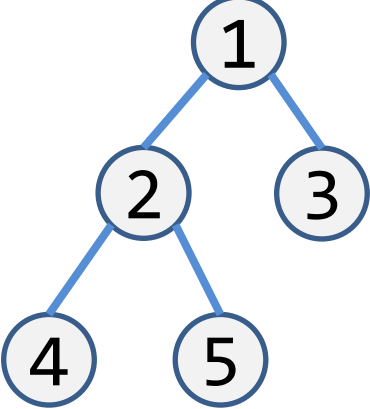


垃圾桶的位置 按照原树DFS顺序依次摆放	按照原树DFS顺序 依次会出现的"换根"事件
递推 $f[2]$ 依赖 $f[1]$ 和 $sz[2]$	1作根换成2作根
递推 $f[4]$ 依赖 $f[2]$ 和 $sz[4]$	2作根换成4作根
递推 $f[5]$ 依赖 $f[2]$ 和 $sz[5]$	2作根换成5作根
递推 $f[3]$ 依赖 $f[1]$ 和 $sz[3]$	1作根换成3作根

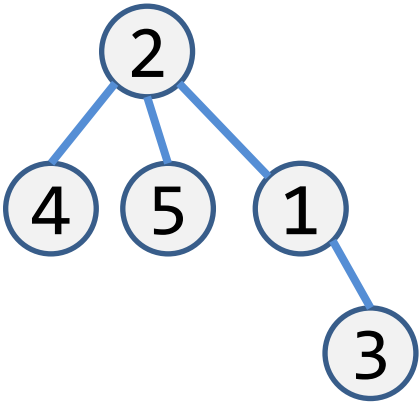
所需要的 $sz[]$ 就是
1作根时的计算结果

换根步骤易错点
"子树"概念变化,变量含义混淆

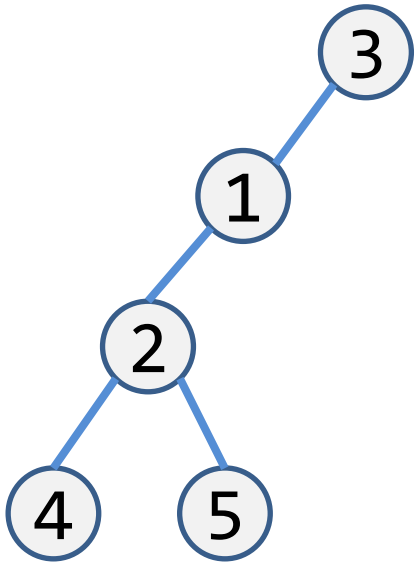
$f[1]=6$



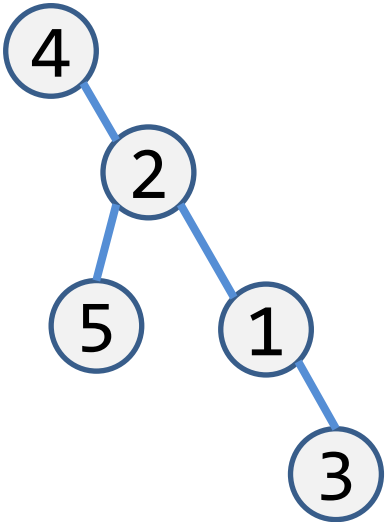
$f[2]=5$



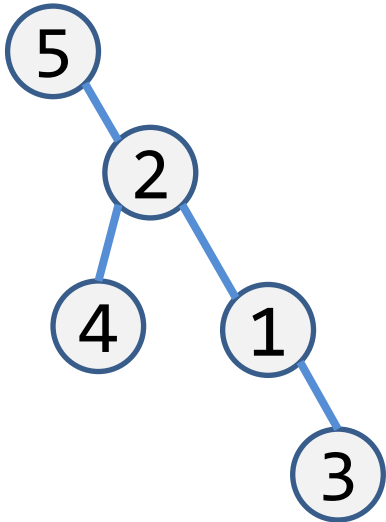
$f[3]=9$



$f[4]=8$



$f[5]=8$






$f[u]$ 代表以 u 为起点到其他点距离的总和

```
32 void solve(){
33     dfs_d_sz(1,0); ←
34     for(int u=1;u<=n;++u) f[1]+=d[u]-1;
35     dfs_f(1,0); ←
36     int id=max_element(f+1,f+1+n)-f;
37     cout<<id<<endl;
38 }
```

$f[v]$ 代表以 v 为起点到其他点距离的总和

$f[u]$ 代表以 u 为起点到其他点距离的总和

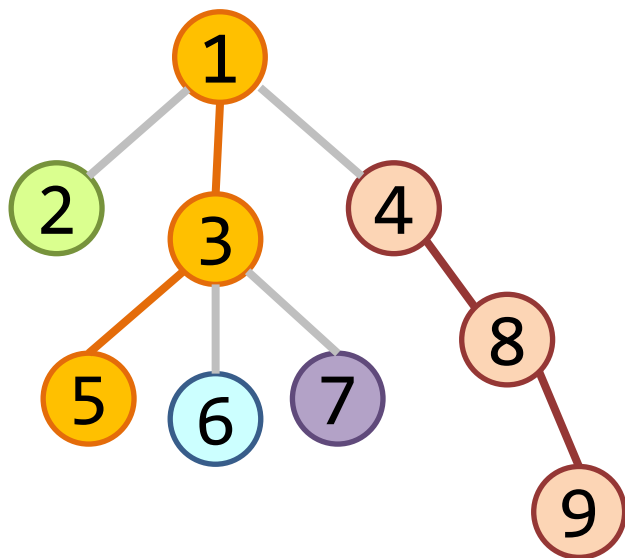
```
16 void dfs_f(int u, int fa){
17     for(int i=0; i<to[u].size(); ++i){
18         int v=to[u][i];
19         if(v==fa) continue;
20         f[v]=
21         
22         
23     }
```



计算顺序从上往下
算完 $f[v]$ 再递归

1709

重链剖分



1号有3个儿子

$sz[2]=1$

$sz[3]=4$

$sz[4]=3$

1号的重儿子是3号

2号和4号是轻儿子

从根出发的重链

$1 \rightarrow 3 \rightarrow 5$

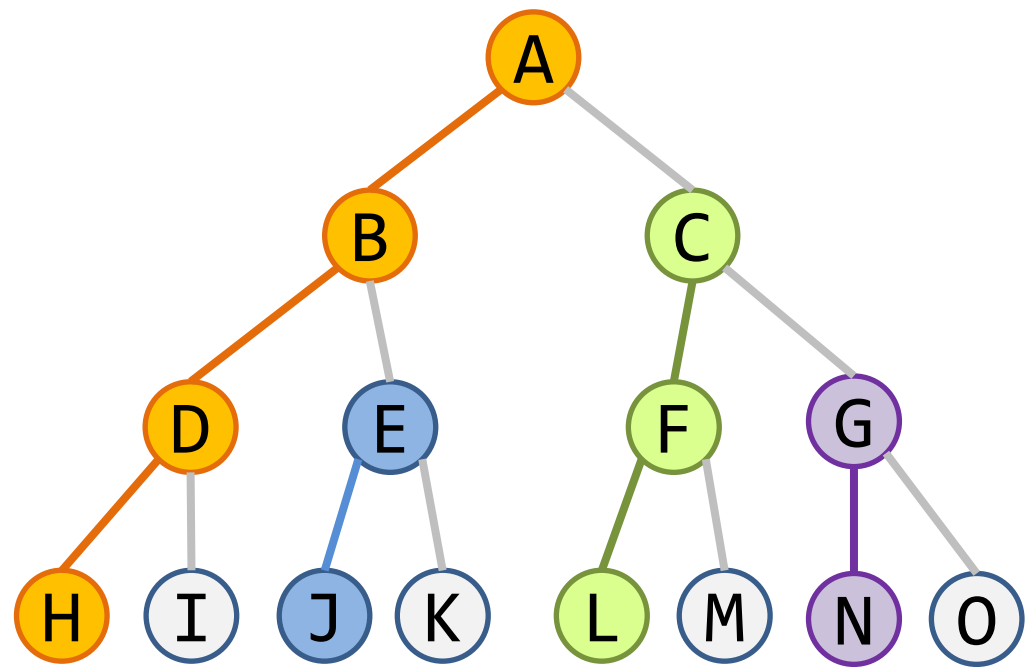
剩下一片森林

对每棵树继续

重链剖分

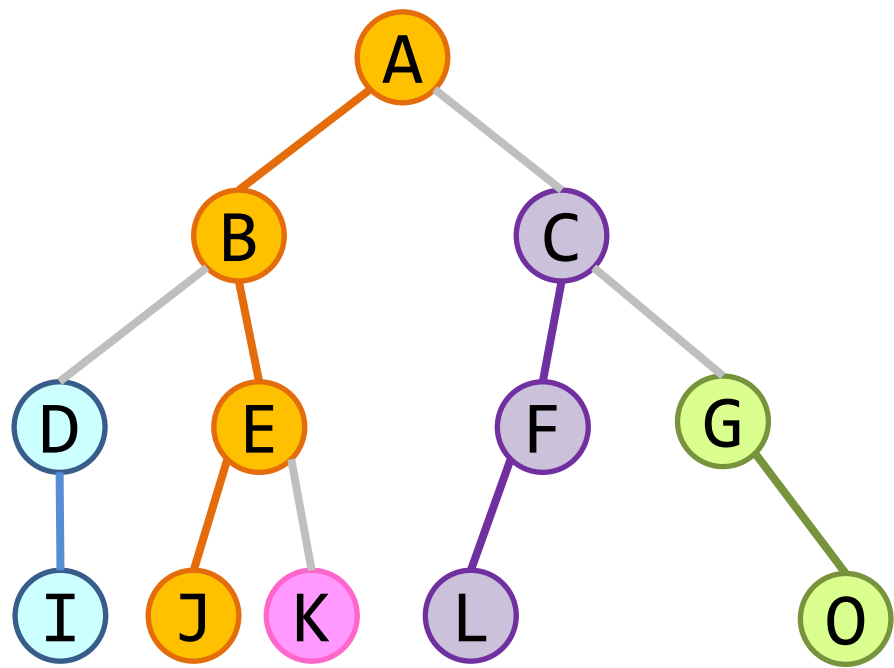
共5条链

重链剖分



重链剖分
共分成8条链

重链剖分



重链剖分
共分成5条链

重链剖分	son[u]代表u的重儿子	儿子们选sz最大的 若平局选编号最小的
	L[u]代表从u往下的重链节点数	

```
6 bool better(int u,int v){
7     return sz[u]>sz[v]||sz[u]==sz[v]&&u<v;
8 }
9 void dfs(int u,int fa) {
10     sz[u]=1;
11     son[u]=0;
12     for(int i=0;i<to[u].size();++i){
13         int v=to[u][i];
14         if(v==fa) continue;
15         dfs(v,u);
16         sz[u]+=sz[v];
17         if(better(v,son[u])) son[u]=v;
18     }
19     L[u]=L[son[u]]+1;
20 }
```

重链剖分

son[u]代表u的重儿子

儿子们选sz最大的
若平局选编号最小的

L[u]代表从u往下的重链节点数

```
29 void solve(){
30     for(int u=1;u<=n;++u){
31         dfs(u,0);
32         cout<<L[u]<<" ";
33     }
34     cout<<endl;
35 }
```

时间复杂度 $O(n^2)$

能否更快?

DFS多次调用
son[],L[]需要每次清空吗

重链剖分

将树上问题拆分成若干链上问题

重链上相邻的都是直系父子

任何两条重链不相交

任何节点属于唯一重链

链底部都是叶节点

重链和重心有关系吗?

太戈编程

1824

1795

拓展题

1709

只需80分