**[SPOJ QTREE 1-7 浅谈树上分治问题](http://www.cnblogs.com/fullpower/p/3869547.html)**

 375.Query on a tree  【QTREE】

　　有两个操作：

　　（1）修改第i条边的边权

　　（2）询问a到b路径上的边权最大值。

　　树链剖分入门题。树链剖分+线段树维护最大值。修改/查询均为O(log^2)。

很懒，没有写。

913.Query on a tree II 【QTREE2】

　　有两个操作：

　　（1）询问a到b的距离。

　　（2）询问a到b路径上的第k个点。

　　很容易想到倍增LCA。

　　路径上的第k个点，要么是a的第k个父亲，要么是b的第k'个父亲，同样可以倍增实现。

　　询问都是O(logn)。

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<cmath>

#include<algorithm>

#define MAXN 10010

#define MAXD 16

using namespace std;

int v[MAXN\*2],e[MAXN\*2],next[MAXN\*2],first[MAXN];

int d[MAXN],p[MAXN][MAXD],dis[MAXN][MAXD];

int n,tot;

char op[5];

void addedge(int x,int y,int z){

v[++tot]=y;e[tot]=z;next[tot]=first[x];first[x]=tot;

v[++tot]=x;e[tot]=z;next[tot]=first[y];first[y]=tot;

}

void dfs(int u,int fa,int dist){

d[u]=d[fa]+1,p[u][0]=fa,dis[u][0]=dist;

for(int i=1;i<MAXD;i++){

p[u][i]=p[p[u][i-1]][i-1];

dis[u][i]=dis[u][i-1]+dis[p[u][i-1]][i-1];

}

for(int i=first[u];i;i=next[i])

if(v[i]!=fa)

dfs(v[i],u,e[i]);

}

int lca(int x,int y){

if(d[x]>d[y]) swap(x,y);

int delta=d[y]-d[x];

for(int i=0;i<MAXD;i++)

if(delta&(1<<i))

y=p[y][i];

if(x==y) return x;

for(int i=MAXD-1;i>=0;i--)

if(p[x][i]!=p[y][i])

x=p[x][i],y=p[y][i];

return p[x][0];

}

int getdis(int x,int y){

if(d[x]>d[y]) swap(x,y);

int ret=0,delta=d[y]-d[x];

for(int i=0;i<MAXD;i++)

if(delta&(1<<i))

ret+=dis[y][i],y=p[y][i];

if(x==y) return ret;

for(int i=MAXD-1;i>=0;i--)

if(p[x][i]!=p[y][i]){

ret+=dis[x][i],x=p[x][i];

ret+=dis[y][i],y=p[y][i];

}

ret=ret+dis[x][0]+dis[y][0];

return ret;

}

int getkthfa(int x,int k){

for(int i=0;i<MAXD;i++)

if(k&(1<<i))

x=p[x][i];

return x;

}

int getkth(int x,int y,int k){

int t=lca(x,y);

if(d[x]-d[t]>=k-1)

return getkthfa(x,k-1);

else

return getkthfa(y,d[x]-d[t]+d[y]-d[t]+1-k);

}

int main(){

freopen("input.txt","r",stdin);freopen("output.txt","w",stdout);

int t;scanf("%d",&t);

while(t--){

scanf("%d",&n);

memset(first,0,sizeof(first));

tot=0;

for(int i=1;i<n;i++){

int x,y,z;

scanf("%d%d%d",&x,&y,&z);

addedge(x,y,z);

}

dfs(1,0,0);

while(scanf("%s",op)!=EOF && op[1]!='O')

if(op[0]=='D'){

int a,b;scanf("%d%d",&a,&b);

printf("%d\n",getdis(a,b));

}else{

int a,b,k;scanf("%d%d%d",&a,&b,&k);

printf("%d\n",getkth(a,b,k));

}

}

return 0;

}

QTREE2http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif QTREE2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **DATE** | **PROBLEM** | **RESULT** | **TIME** | **MEM** | **LANG** |
| [12000885](http://www.spoj.com/files/src/12000885/" \o "View source code) | 2014-07-23 | [Query on a tree II](http://www.spoj.com/problems/QTREE2/" \o "QTREE2) | accepted | [0.61](http://www.spoj.com/ranks/QTREE2/" \o "See the best solutions) | 4.4M | C++4.3.2 |
|  |  |  |  |  |  |  |

2789.Query on a tree again 【QTREE3】

 　　现在有一颗树，其所有的节点都为白色。有两个操作：

　　（1）单点颜色修改，黑变白，或者白变黑。

　　（2）询问1到v路径上，最先出现黑点的位置。

  　树链剖分肯定能做，我用的是LCT。

　　我们需要维护一段区间1-v上最先出现的黑点位置，即这条链中最上面的黑点。

　　等价于splay中最左边出现的黑点。很裸吧。

　　理论的修改/询问均为O(logn)，最好常数优化，我加了fread 25-->100！

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<cmath>

#include<algorithm>

#define MAXN 200000

using namespace std;

int n,q;

struct LinkCutTree{

int ch[MAXN][2],col[MAXN],front[MAXN],f[MAXN],rev[MAXN];

bool isroot(int x){

return (!f[x]||ch[f[x]][0]!=x&&ch[f[x]][1]!=x);

}

void push\_up(int x){

if(!x) return;

int lc=ch[x][0],rc=ch[x][1];

if(~front[lc]) front[x]=front[lc];

else

if(col[x]) front[x]=x;

else

if(~front[rc]) front[x]=front[rc];

else

front[x]=-1;

}

void push\_down(int x){

if(!x) return;

int lc=ch[x][0],rc=ch[x][1];

if(rev[x]){

swap(ch[x][0],ch[x][1]);

rev[lc]^=1;rev[rc]^=1;

rev[x]=0;

}

}

void rotate(int x,int t){

if(isroot(x)) return;

int y=f[x];

ch[y][t]=ch[x][!t];if(ch[x][!t]) f[ch[x][!t]]=y;

f[x]=f[y];if(f[y]){

if(ch[f[y]][0]==y) ch[f[y]][0]=x;

if(ch[f[y]][1]==y) ch[f[y]][1]=x;

}

f[ch[x][!t]=y]=x;push\_up(y);

}

void splay(int x){

push\_down(x);

while(!isroot(x)){

int y=f[x];push\_down(y);push\_down(x);

rotate(x,ch[y][1]==x);

}

push\_up(x);

}

void init(){

memset(front,255,sizeof(front));

}

void access(int v){

for(int u=v,v=0;u;v=u,u=f[u]) splay(u),ch[u][1]=v;

}

void evert(int v){

access(v);splay(v);rev[v]^=1;

}

int find\_root(int v){

access(v);splay(v);for(;ch[v][0];v=ch[v][0]);splay(v);return v;

}

void link(int u,int v){

evert(u);f[u]=v;

}

void cut(int u,int v){

evert(u);access(v);splay(v);f[u]=ch[v][0]=0;

}

void modify(int v){

access(v);splay(v);col[v]^=1;

}

int query(int v){

evert(1);access(v);splay(v);return front[v];

}

}LCT;

char buf[8000000],\*pt = buf,\*o = buf;

inline int getint(){

int f = 1,x = 0;

while((\*pt != '-') && (\*pt < '0' || \*pt > '9')) pt ++;

if(\*pt == '-') f = -1,pt ++; else x = \*pt++ - 48;

while(\*pt >= '0' && \*pt <= '9') x = x \* 10 + \*pt ++ - 48;

return x \* f;

}

int main(){

freopen("input.txt","r",stdin);freopen("output.txt","w",stdout);

fread(buf,1,8000000,stdin);

n=getint(),q=getint();

LCT.init();

for(int i=1;i<n;i++){

int x=getint(),y=getint();

LCT.link(x,y);

}

while(q--){

int op=getint(),v=getint();

if(op==0) LCT.modify(v);

else printf("%d\n",LCT.query(v));

}

return 0;

}

QTREE3http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif QTREE3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **DATE** | **PROBLEM** | **RESULT** | **TIME** | **MEM** | **LANG** |
| [12002267](http://www.spoj.com/files/src/12002267/" \o "View source code) | 2014-07-23 | [Query on a tree again!](http://www.spoj.com/problems/QTREE3/" \o "QTREE3) | 100 | [11.63](http://www.spoj.com/ranks/QTREE3/" \o "See the best solutions) | 15M | C++4.3.2 |

2666.Query on a tree IV 【QTREE4】

　　一开始所有的点都是白色的。同样是两个操作：

　　（1）单点颜色修改

　　（2）询问整棵树中，最远的白色两点的距离 <==>max{dist(a,b)},color[a]=color[b]=white.

　　公认的QTREE1-5中最繁琐的题目。我想练练手边分，树的边分应该是最难写的。

　　请教神犇，都称“珍惜生命，请用点分治” T T

　　这道题目用边分是很直观的。

　　每层找出一条中心边，删边，可以分成左右两个子树。

　　左右各维护一个大根堆，分别记录白点到子树根的距离。

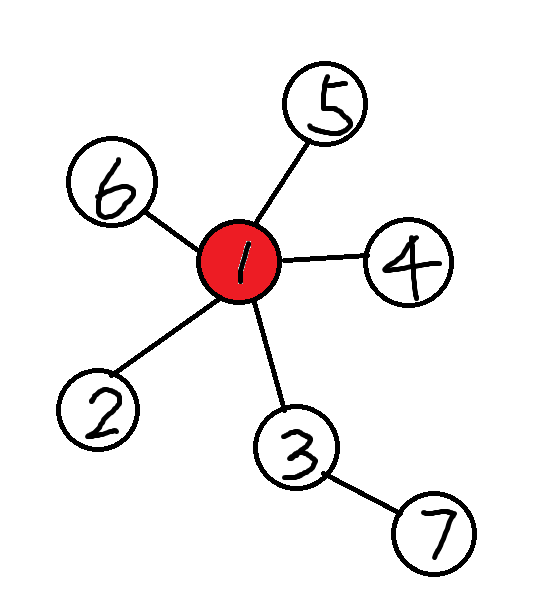
　　ans=max{lc.ans,rc.ans,leftheap.top()+rightheap.top()+midlen}

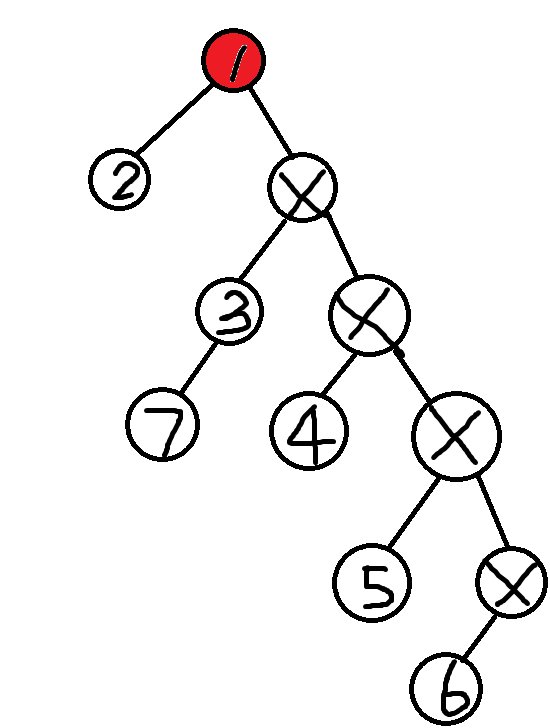
　　（极限数据我用SET3.8s，用优先队列2.5s）

　　询问是O(1)，修改是O(log^2)。

　　补充边分的知识：

　　①树是菊花状，边分会退化。

　　可以试着加虚点。（这里是向jcvb学习的）遵循第二个儿子就加虚点的原则。



②有些有趣的性质。不难发现每个节点的度不大于3，或者说这就是棵二叉树！

　　实节点只在左儿子。

　　最重要的是，他不再是菊花了！

　③dfs记录答案信息。点分/边分维护路径长度是可行的。

　　比如记下每个点到该层子树根的距离。总的空间是nlogn。

　④拆掉每条边的估价和左右子树的大小有关。

　　dfs\_size(),min{max{size[u],n-size[u]}}，可以找到中心边了。

　⑤递归左右子树，回到③。

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<cmath>

#include<algorithm>

#include<set>

#include<queue>

using namespace std;

#define MAXN 210000

#define MAXM 4000000

#define WHITE 1

#define BLACK 0

int V[MAXM],E[MAXM],NEXT[MAXM],FIR[MAXN]; int N;

int v[MAXN\*2],e[MAXN\*2],next[MAXN\*2],pre[MAXN\*2],fir[MAXN],ed[MAXN]; int n;

int col[MAXN],size[MAXN];

int p,mi,midedge,tot;

char buf[8000000],\*pt = buf,\*o = buf;

int getint(){

int f = 1,x = 0;

while((\*pt != '-') && (\*pt < '0' || \*pt > '9')) pt ++;

if(\*pt == '-') f = -1,pt ++; else x = \*pt++ - 48;

while(\*pt >= '0' && \*pt <= '9') x = x \* 10 + \*pt ++ - 48;

return x \* f;

}

char getch(){

char ch;

while(\*pt < 'A' || \*pt > 'Z') pt ++;

ch=\*pt;pt++;

return ch;

}

void ADD1(int x,int y,int z){

V[tot]=y;E[tot]=z;NEXT[tot]=FIR[x];FIR[x]=tot;tot++;

}

void ADD(int x,int y,int z){

V[tot]=y;E[tot]=z;NEXT[tot]=FIR[x];FIR[x]=tot;tot++;

V[tot]=x;E[tot]=z;NEXT[tot]=FIR[y];FIR[y]=tot;tot++;

}

void add(int x,int y,int z){

v[tot]=y;e[tot]=z;next[tot]=fir[x];fir[x]=tot;tot++;

v[tot]=x;e[tot]=z;next[tot]=fir[y];fir[y]=tot;tot++;

}

void getpre(){

memset(ed,255,sizeof(ed));

for(int i=1;i<=n;i++)

for(int j=fir[i];~j;j=next[j]){

pre[j]=ed[i];

ed[i]=j;

}

}

void \_delete(int x,int i){

if(fir[x]==i) fir[x]=next[i]; else next[pre[i]]=next[i];

if(ed[x]==i) ed[x]=pre[i]; else pre[next[i]]=pre[i];

}

void init(){

memset(FIR,255,sizeof(FIR));tot=0;

N=getint();

for(int i=1;i<N;i++){

int x=getint(),y=getint(),z=getint();

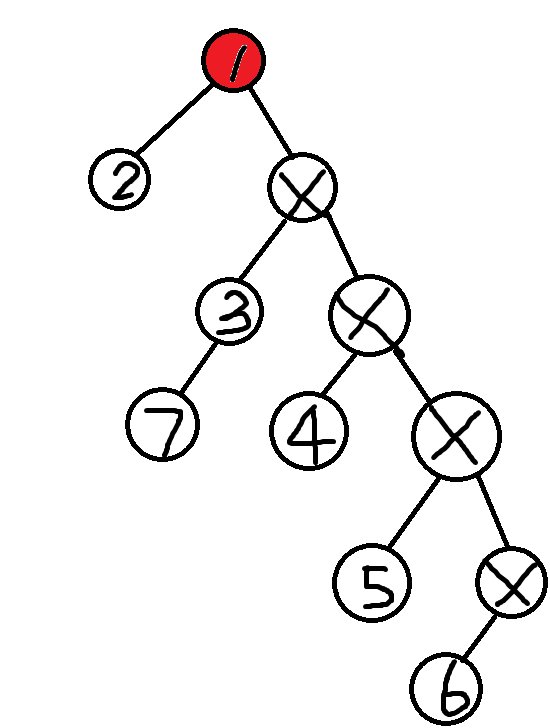
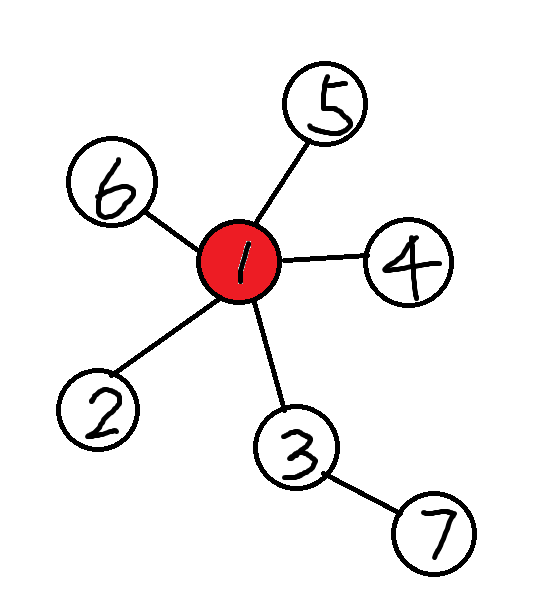
ADD(x,y,z);

}

}

void check(int u,int fa){

int father=0;

 for(int i=FIR[u];~i;i=NEXT[i])

if(V[i]!=fa)

if(father==0){//第1条边

add(u,V[i],E[i]);

father=u;

check(V[i],u);

}else{//非第1条边

++n;col[n]=BLACK;

add(father,n,0);add(n,V[i],E[i]);

father=n;

check(V[i],u);

}

}

void rebuild(){

memset(fir,255,sizeof(fir));tot=0;

n=N;

for(int i=1;i<=n;i++) col[i]=WHITE;

check(1,0);

getpre();

memset(FIR,255,sizeof(FIR));tot=0;

}

struct point{

int dist,id;

bool operator<(const point&b)const{

return dist<b.dist;

}

};

struct node{

int rt,midlen,ans;

int lc,rc;

priority\_queue<point>Q;

}T[MAXN\*4];

int cnt=0;

void dfs\_size(int u,int fa,int dist){

ADD1(u,p,dist);

if(col[u])T[p].Q.push((point){dist,u});

size[u]=1;

for(int i=fir[u];~i;i=next[i])

if(v[i]!=fa){

dfs\_size(v[i],u,dist+e[i]);

size[u]+=size[v[i]];

}

}

void dfs\_midedge(int u,int code){

if(max(size[u],size[T[p].rt]-size[u])<mi){

mi=max(size[u],size[T[p].rt]-size[u]);

midedge=code;

}

for(int i=fir[u];~i;i=next[i])

if(i!=(code^1))

dfs\_midedge(v[i],i);

}

void push\_up(int p){

T[p].ans=-1;

while(!T[p].Q.empty()&&(col[T[p].Q.top().id]==0)) T[p].Q.pop();//去掉黑色点

int lc=T[p].lc,rc=T[p].rc;

if(lc==0&&rc==0){//没有左右子树

if(col[T[p].rt]) T[p].ans=0;

}else{

if(T[lc].ans>T[p].ans) T[p].ans=T[lc].ans;

if(T[rc].ans>T[p].ans) T[p].ans=T[rc].ans;

if(!T[lc].Q.empty()&&!T[rc].Q.empty())//有左右子树

T[p].ans=max(T[lc].Q.top().dist+T[rc].Q.top().dist+T[p].midlen,T[p].ans);

}

}

void dfs(int pt,int u){

T[pt].rt=u;

p=pt;

dfs\_size(u,0,0);//计算树pt的各子树的大小

midedge=-1;mi=n;

dfs\_midedge(u,-1);//找中心边

if(~midedge){

int p1=v[midedge],p2=v[midedge^1];//边的两个顶点

T[pt].midlen=e[midedge];//中心边长度

\_delete(p1,midedge^1);//删除边

\_delete(p2,midedge);

dfs(T[pt].lc=++cnt,p1);//递归子树

dfs(T[pt].rc=++cnt,p2);

}

push\_up(pt);

}

void change(int x){

col[x]^=1;

if(col[x]==BLACK)

for(int i=FIR[x];~i;i=NEXT[i])

push\_up(V[i]);

else//WHITE

for(int i=FIR[x];~i;i=NEXT[i]){//枚举x到达的所有根节点V[i]，E[i]为x到V[i]的距离

T[V[i]].Q.push((point){E[i],x});

push\_up(V[i]);//更新

}

}

void solve(){

int x;char op;

int q=getint();

while(q--){

op=getch();

if(op=='A')

if(~T[1].ans)

printf("%d\n",T[1].ans);

else

puts("They have disappeared.");

else{

x=getint();

change(x);

}

}

}

int main(){

fread(buf,1,8000000,stdin);

init();

rebuild();

dfs(cnt=1,1);

solve();

return 0;

}

QTREE4http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif QTREE4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **DATE** | **PROBLEM** | **RESULT** | **TIME** | **MEM** |
| 12016831 | 2014-07-25 | [Query on a tree IV](http://www.spoj.com/problems/QTREE4/" \o "QTREE4) | accepted | [10.42](http://www.spoj.com/ranks/QTREE4/" \o "See the best solutions) | 101M |

2939.Query on a tree V 【QTREE5】

给定一棵n个点的树，点从1到n编号。每个点可能有两种颜色：黑或白。我们定义dist(a,b)为点a至点b路径上的边个数。

一开始所有的点都是黑色的。

两个操作：

（1）单点颜色修改

（2）给定点v，询问最近的白点（可能包括本身）的距离。

实际上是QTREE4的弱化版，于是这题就写点分了。

重心分解后，对每个子树root维护一个优先队列que，维护子树中各白点到root的距离。这样染黑色时可以快速从que中找到仍然是白色的第一个点。如果不用que，只用一个元素Path的话，染白色固然没问题，染黑色就麻烦了。

查询x时，遍历x所属子树root，按如下方式更新答案

ans = min(ans, dist(x, root) + dist\_white(root)); // x到根节点+根节点到白点

另外需注意查询次数可能非常多，需要预处理出所有的重心，而不要来一个查询求一次重心

 #include<iostream>

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<cmath>

#include<algorithm>

#include<queue>

using namespace std;

#define INF 0x3f3f3f3f

#define MAXN 400010

#define MAXM 3000010

#define BLACK 0

#define WHITE 1

int V[MAXM],E[MAXM],NEXT[MAXM],FIR[MAXN];

int v[MAXN],e[MAXN],next[MAXN],fir[MAXN];

int col[MAXN],size[MAXN],ma[MAXN],vis[MAXN];

int n,tot;int mi,root,rt;

struct point{

int dist,id;

bool operator<(const point&b)const{

return dist>b.dist;

}

};

priority\_queue<point>Q[MAXN];

char buf[10000000],\*pt = buf,\*o = buf;

inline int getint(){

int f = 1,x = 0;

while((\*pt != '-') && (\*pt < '0' || \*pt > '9')) pt ++;

if(\*pt == '-') f = -1,pt ++; else x = \*pt++ - 48;

while(\*pt >= '0' && \*pt <= '9') x = x \* 10 + \*pt ++ - 48;

return x \* f;

}

void ADD(int x,int y,int z){

V[tot]=y;E[tot]=z;NEXT[tot]=FIR[x];FIR[x]=tot;tot++;

}

void addedge(int x,int y){

v[tot]=y;next[tot]=fir[x];fir[x]=tot;tot++;

v[tot]=x;next[tot]=fir[y];fir[y]=tot;tot++;

}

void init(){

memset(fir,255,sizeof(fir));tot=0;

n=getint();

for(int i=1;i<n;i++){

int x=getint(),y=getint();

addedge(x,y);

}

memset(FIR,255,sizeof(FIR));tot=0;

}

void dfssize(int u,int fa){

size[u]=1;ma[u]=0;

for(int i=fir[u];~i;i=next[i])

if(v[i]!=fa&&!vis[v[i]]){

dfssize(v[i],u);

size[u]+=size[v[i]];

if(size[v[i]]>ma[u]) ma[u]=size[v[i]];

}

}

void dfsroot(int u,int fa){

if(size[rt]-size[u]>ma[u]) ma[u]=size[rt]-size[u];

if(ma[u]<mi) mi=ma[u],root=u;

for(int i=fir[u];~i;i=next[i])

if(v[i]!=fa&&!vis[v[i]])

dfsroot(v[i],u);

}

void dfsdis(int u,int fa,int dis){

ADD(u,root,dis);

for(int i=fir[u];~i;i=next[i])

if(v[i]!=fa&&!vis[v[i]])

dfsdis(v[i],u,dis+1);

}

void dfs(int u){

mi=n,rt=u;dfssize(u,0);dfsroot(u,0);

dfsdis(root,0,0);

vis[root]=1;

for(int i=fir[root];~i;i=next[i])

if(!vis[v[i]])

dfs(v[i]);

}

void modify(int x){

col[x]^=1;

if(col[x]==WHITE)

for(int i=FIR[x];~i;i=NEXT[i])//枚举x到所有根节点

Q[V[i]].push((point){E[i],x});

}

int query(int x){

int ret=INF;

for(int i=FIR[x];~i;i=NEXT[i]){

while(!Q[V[i]].empty()&&(col[Q[V[i]].top().id]==BLACK)) Q[V[i]].pop();

if(!Q[V[i]].empty()) ret=min(Q[V[i]].top().dist+E[i],ret);

}

return ret<INF?ret:-1;

}

void solve(){

int q=getint();

int op,x;

while(q--){

op=getint(),x=getint();

if(op==0)

modify(x);

else

printf("%d\n",query(x));

}

}

int main(){

fread(buf,1,10000000,stdin);

init();

dfs(1);

solve();

return 0;

}

QTREE5http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif QTREE5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **DATE** | **PROBLEM** | **RESULT** | **TIME** | **MEM** |
| [12018865](http://www.spoj.com/files/src/12018865/" \o "View source code) | 2014-07-25 | [Query on a tree V](http://www.spoj.com/problems/QTREE5/" \o "QTREE5) | accepted | [6.36](http://www.spoj.com/ranks/QTREE5/" \o "See the best solutions) | 95M |

16549.Query on a tree VI【QTREE6】

　　题目描述（翻译）

给你一棵n个点的树，编号1至n。每个点可以是黑色，可以是白色。初始时所有点都是黑色。下面有两种操作请你操作给我们看：

0 u：询问有多少个节点v满足路径u到v上所有节点（包括）都拥有相同的颜色

1 u：翻转u的颜色

输入输出格式

输入格式

第一行一个整数n

接下来n−1行，每行两个整数表示一条边

接下来一行一个整数m表示操作次数

接下来m行，每行两个整数分别表示操作类型和被操作节点

输出格式

对每个询问操作输出相应的结果

输入输出样例

输入样例

5

1 2

1 3

1 4

1 5

3

0 1

1 1

0 1

1

2

3

4

5

6

7

8

9

输出样例

5

1

1

2

**题解**

简单来说，就是维护同色联通块的大小

干脆直接暴力linkcut好了（T上天）

然后来介绍一下一种维护染色联通块的较为常用的模型

**很多与树有关的题目，当边权不好处理时，可以把边权转化为儿子的点权来做**

**然后这里恰恰相反，要把点权给转化为边权**

我们把每一个节点的颜色赋给与它父亲相连的边

弄两个LCT，对应两种颜色，一种颜色的边只有在对应的LCT中才会相连

于是，同色的联通块，就转化为了减去顶部节点后的边的连通块（因为顶部节点在这个LCT是没有向上连边的，说明顶部节点并不是这个颜色，那么他的所有子树并不连通，要断开才行）

然后可以发现，修改点的颜色之后，只要在原来的LCT中cut，在新的LCT上link就行啦

查询怎么做呢？先access，然后findroot，再输出root的实子树大小就行了

ps：**1本来没有父亲，但为了方便，可以建一个虚点n+1，令他作为1的父亲**就好了

//minamoto

#include<iostream>

#include<cstdio>

#define getc() (p1==p2&&(p2=(p1=buf)+fread(buf,1,1<<21,stdin),p1==p2)?EOF:\*p1++)

char buf[1<<21],\*p1=buf,\*p2=buf;

inline int read(){

#define num ch-'0'

char ch;bool flag=0;int res;

while(!isdigit(ch=getc()))

(ch=='-')&&(flag=true);

for(res=num;isdigit(ch=getc());res=res\*10+num);

(flag)&&(res=-res);

#undef num

return res;

}

char sr[1<<21],z[20];int C=-1,Z;

inline void Ot(){fwrite(sr,1,C+1,stdout),C=-1;}

inline void print(int x){

if(C>1<<20)Ot();if(x<0)sr[++C]=45,x=-x;

while(z[++Z]=x%10+48,x/=10);

while(sr[++C]=z[Z],--Z);sr[++C]='\n';

}

const int N=100005;

int f[N],Next[N<<1],head[N],ver[N<<1],tot,col[N],n,m;

inline void add(int u,int v){

ver[++tot]=v,Next[tot]=head[u],head[u]=tot;

ver[++tot]=u,Next[tot]=head[v],head[v]=tot;

}

struct LCT{

int fa[N],ch[N][2],si[N],sum[N];

LCT(){for(int i=1;i<=n+1;++i) sum[i]=1;}

inline bool isroot(int x){return ch[fa[x]][0]!=x&&ch[fa[x]][1]!=x;}

#define lc ch[x][0]

#define rc ch[x][1]

inline void pushup(int x){sum[x]=sum[lc]+sum[rc]+si[x]+1;}

void rotate(int x){

int y=fa[x],z=fa[y],d=ch[y][1]==x;

if(!isroot(y)) ch[z][ch[z][1]==y]=x;

fa[x]=z,fa[y]=x,fa[ch[x][d^1]]=y,ch[y][d]=ch[x][d^1],ch[x][d^1]=y,pushup(y);

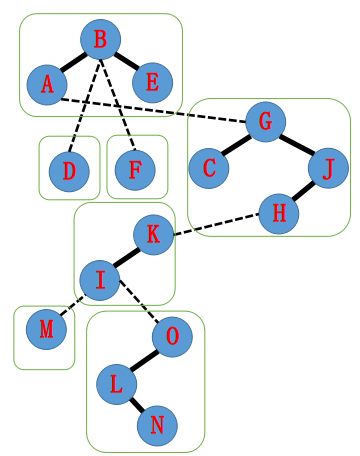
}

void splay(int x){

for(int y=fa[x],z=fa[y];!isroot(x);y=fa[x],z=fa[y]){

if(!isroot(y))

((ch[y][1]==x)^(ch[z][1]==y))?rotate(x):rotate(y);

 rotate(x);

}

pushup(x);

}

void access(int x){

for(int y=0;x;x=fa[y=x]){

splay(x);

si[x]+=sum[rc];//si为其它孩子节点数

si[x]-=sum[rc=y];

}

}

int findroot(int x){

access(x),splay(x);

while(lc) x=lc;

splay(x);

return x;

}

void link(int x){

access(x),splay(x);

int y=fa[x]=f[x];//f[x]为x在原树中的父亲

access(y),splay(y);

si[y]+=sum[x],sum[y]+=sum[x];

}

void cut(int x){

access(x),splay(x);

lc=fa[lc]=0;

pushup(x);

}

}lct[2];

void dfs(int u){

for(int i=head[u];i;i=Next[i]){

int v=ver[i];

if(v==f[u]) continue;

f[v]=u,dfs(v),lct[0].link(v);

}

}

int main(){

n=read();

for(int i=1;i<n;++i){

int u=read(),v=read();

add(u,v);

}

dfs(1);

f[1]=n+1,lct[0].link(1);//lct[1]?

m=read();

while(m--){

int op=read(),u=read();

if(op) lct[col[u]].cut(u),lct[col[u]^=1].link(u);

else{

int v=lct[col[u]].findroot(u);

print(lct[col[u]].sum[lct[col[u]].ch[v][1]]);

}

}

Ot();

return 0;

}

# **[SPOJ QTREE6 Query on a tree VI 树链剖分](https://www.cnblogs.com/AOQNRMGYXLMV/p/5260337.html)**

#### **题意：**

给出一棵含有n(1≤n≤105)个节点的树，每个顶点只有两种颜色：黑色和白色。  
一开始所有的点都是黑色，下面有两种共m(1≤n≤105)次操作：

* 0 u表示查询u所在的连通块的大小，相邻两个点颜色相同则属于一个连通块。
* 1 u表示翻转u的颜色，即黑点变白点，白点变黑点。

#### **分析：**

[参考CodeChef上的题解](https://discuss.codechef.com/questions/31833/qtree6-editorial)

首先将这棵树剖分成轻重链。  
然后我们维护两个值：White(u)和Black(u)。  
White(u)表示当u是白点时(这里我们不关心u真正的颜色)，以u为根的子树中，u所在的连通块的大小。  
同理，Black(u)表示u是黑点时(这里我们不关心u真正的颜色)，以u为根的子树中，u所在的连通块的大小。

* 对于查询操作0 u：

从u往上走，走到深度最小的与u同色的节点v，那么答案就是White(v)或Black(v)。

* 对于修改操作1,u：

由于对称性，不妨假设u从白点变为黑点。

从u的父节点往上走，走到第一个黑点v1，设路径path1为fa(u)→v，然后将path1上所有点的White值**减去**White(u)。

同样地，从u的父节点往上走，走到第一个白点v2，设路径path2为fa(u)→v2，然后将path2上所有点的Black值**加上**Black(u)。

这里修改操作是成段更新的，所以要用线段树维护一下。

接下来还要解决一个问题：如何快速找到上面说的深度最浅的同色点和遇到的第一个黑/白点  
继续用线段树维护一个fir0和fir1，表示该区间从右往左遇到的第一个白点和黑点，对应到树上的链就是从下往上。  
这样就解决了第二个问题，其实第一个问题的答案也可以间接得到。  
要找深度最前的同色点，就是第一个异色点的那个子节点。  
父节点只有一个，但子节点又如何确定呢？  
注意到，我们查询时是在剖出来的链上一条一条往上**“跳”**的，所以在同一条链上一个点的子节点就在线段树中它位置的右边相邻的那个点。

#include <cstdio>#include <cstring>#include <algorithm>

using namespace std;

const int maxn=100000+10;const int maxnode=maxn\*4;

struct Edge{

int v, nxt;

Edge() {}

Edge(int v, int nxt): v(v), nxt(nxt) {}

};

int ecnt, head[maxn];

Edge edges[maxn \* 2];

void AddEdge(int u, int v) {

edges[ecnt] = Edge(v, head[u]);

head[u] = ecnt++;

}

int n;

int fa[maxn], sz[maxn], son[maxn], dep[maxn];

void dfs(int u) {

sz[u] = 1; son[u] = 0;

for(int i = head[u]; ~i; i = edges[i].nxt) {

int v = edges[i].v;

if(v == fa[u]) continue;

fa[v] = u;

dep[v] = dep[u] + 1;

dfs(v);

sz[u] += sz[v];

if(sz[v] > sz[son[u]]) son[u] = v;

}

}

int tot, top[maxn], id[maxn], pos[maxn];

void dfs2(int u, int tp) {

top[u] = tp;

id[u] = ++tot;

pos[tot] = u;

if(!son[u]) return;

dfs2(son[u], tp);

for(int i = head[u]; ~i; i = edges[i].nxt) {

int v = edges[i].v;

if(v == fa[u] || v == son[u]) continue;

dfs2(v, v);

}

}

int color[maxn], addv[2][maxnode];int fir[2][maxnode];

void build(int o, int L, int R) {

if(L == R) {

fir[0][o] = 0; fir[1][o] = L;

addv[0][o] = 1; addv[1][o] = sz[pos[L]];

return;

}

int M = (L + R) / 2;

build(o<<1, L, M);

build(o<<1|1, M+1, R);

fir[0][o] = 0;

fir[1][o] = R;

}

void pushdown(int o) {

for(int i = 0; i < 2; i++) {

int& t = addv[i][o];

if(!t) continue;

addv[i][o<<1] += t;

addv[i][o<<1|1] += t;

t = 0;

}

}

void update(int o, int L, int R, int qL, int qR, int col, int v) {

if(qL <= L && R <= qR) {

addv[col][o] += v;

return;

}

pushdown(o);

int M = (L + R) / 2;

if(qL <= M) update(o<<1, L, M, qL, qR, col, v);

if(qR > M) update(o<<1|1, M+1, R, qL, qR, col, v);

}

void UPDATE(int u, int v, int col, int val) {

while(top[u] != top[v]) {

update(1, 1, n, id[top[u]], id[u], col, val);

u = fa[top[u]];

}

update(1, 1, n, id[v], id[u], col, val);

}

int querysize(int o, int L, int R, int p, int col) {

if(L == R) return addv[col][o];

pushdown(o);

int M = (L + R) / 2;

if(p <= M) return querysize(o<<1, L, M, p, col);

else return querysize(o<<1|1, M+1, R, p, col);

}

int queryfir(int o, int L, int R, int qL, int qR, int col) {

if(qL <= L && R <= qR) return fir[col][o];

int ans = 0;

int M = (L + R) / 2;

if(qR > M) ans = queryfir(o<<1|1, M+1, R, qL, qR, col);

if(ans) return ans;

if(qL <= M) ans = queryfir(o<<1, L, M, qL, qR, col);

return ans;

}

int QueryFir(int u, int col) {

int ans = 0;

int t = top[u];

while(t != 1) {

ans = queryfir(1, 1, n, id[t], id[u], col);

if(ans) return ans;

u = fa[t]; t = top[u];

}

return queryfir(1, 1, n, 1, id[u], col);

}

int QuerySuf(int u, int col) {

int ans = id[u];

while(top[u] != 1) {

int t = queryfir(1, 1, n, id[top[u]], id[u], col ^ 1);

if(t) return t == id[u] ? ans : t + 1;

ans = id[top[u]];

u = fa[top[u]];

}

int t = queryfir(1, 1, n, 1, id[u], col ^ 1);

if(!t) return 1;

return t == id[u] ? ans : t + 1;

}

void change(int o, int L, int R, int p) {

if(L == R) {

int u = pos[L];

int& c = color[u];

c ^= 1;

fir[c][o] = L;

fir[c ^ 1][o] = 0;

return;

}

int M = (L + R) / 2, lenr = R - M;

if(p <= M) change(o<<1, L, M, p);

else change(o<<1|1, M+1, R, p);

fir[0][o] = fir[0][o<<1|1] ? fir[0][o<<1|1] : fir[0][o<<1];

fir[1][o] = fir[1][o<<1|1] ? fir[1][o<<1|1] : fir[1][o<<1];

}

int main(){

scanf("%d", &n);

for(int i = 1; i <= n; i++) color[i] = 1;

ecnt = 0;

memset(head, -1, sizeof(head));

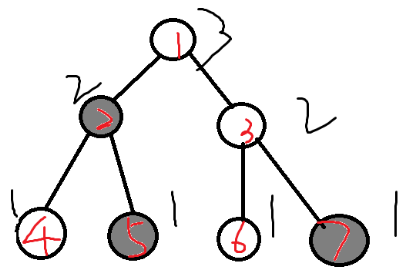
for(int i = 1; i < n; i++) {

int u, v; scanf("%d%d", &u, &v);

AddEdge(u, v);

AddEdge(v, u);

}

 dfs(1);

tot = 0;

dfs2(1, 1);

build(1, 1, n);

int \_; scanf("%d", &\_);

while(\_--) {

int op, u; scanf("%d%d", &op, &u);

if(op == 0) {

int v = pos[QuerySuf(u, color[u])];

int ans = querysize(1, 1, n, id[v], color[v]);

printf("%d\n", ans);

} else {

if(u != 1) {

int v = pos[QueryFir(fa[u], color[u]^1)];

if(!v) v = 1;

int sub = querysize(1, 1, n, id[u], color[u]);

UPDATE(fa[u], v, color[u], -sub);

v = pos[QueryFir(fa[u], color[u])];

if(!v) v = 1;

int add = querysize(1, 1, n, id[u], color[u] ^ 1);

UPDATE(fa[u], v, color[u] ^ 1, add);

}

change(1, 1, n, id[u]);

}

}

return 0;

}

# BZOJ 3637: Query on a tree VI (树链剖分+树状数组)：

<https://blog.csdn.net/the_useless/article/details/53953973>

16580.Query on a tree VII 【QTREE7】

　　有三个操作：

　　（1）给定点u，在满足条件（u,v）路径上所有点都与u同色 的点v中，询问点权最大值。包括u,v。

　　（2）单点改色。

　　（3）单点点权修改

　　xiaodao的神题啊，我琢磨了半天std，LCT的神应用。

　　弄黑白两棵LCT。

　　某个点，用set维护轻边上的最大值，splay上维护轻边与重边总的最大值。

　　单点改色，需要删除再插入。

　　单点删除怎么办？

　　在黑树上有边u→v，我们保证深度较深的点即儿子点v一定是黑点。

　　若v变为白色，那么删边u→v，而v的儿子边都不用删。

　　查询的时候，可能根是白色，但儿子一定是黑色，所以特判一下。

　　白树同理。时间复杂度O(nlog^2)，bzoj rank1！（暂时）spoj rank2.

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<cmath>

#include<algorithm>

#include<set>

#define MAXN 200010

using namespace std;

int col[MAXN],fa[MAXN];

int n;

char buf[10000000],\*pt = buf,\*o = buf;

inline int getint(){

int f = 1,x = 0;

while((\*pt != '-') && (\*pt < '0' || \*pt > '9')) pt ++;

if(\*pt == '-') f = -1,pt ++; else x = \*pt++ - 48;

while(\*pt >= '0' && \*pt <= '9') x = x \* 10 + \*pt ++ - 48;

return x \* f;

}

struct LinkCutTree{

int ch[MAXN][2],key[MAXN],ma[MAXN],f[MAXN];

multiset<int>S[MAXN];

bool isroot(int x){

return (!f[x]||ch[f[x]][0]!=x&&ch[f[x]][1]!=x);

}

void push\_up(int x){

ma[x]=key[x];

if(S[x].size()) ma[x]=max(\*S[x].rbegin(),ma[x]);

if(ch[x][0]) ma[x]=max(ma[ch[x][0]],ma[x]);

if(ch[x][1]) ma[x]=max(ma[ch[x][1]],ma[x]);

}

void rotate(int x,int t){

if(isroot(x)) return;

int y=f[x];

ch[y][t]=ch[x][!t];if(ch[x][!t]) f[ch[x][!t]]=y;

f[x]=f[y];

if(f[y]){

if(ch[f[y]][0]==y) ch[f[y]][0]=x;

if(ch[f[y]][1]==y) ch[f[y]][1]=x;

}

f[ch[x][!t]=y]=x;

push\_up(y);

}

void splay(int x){

while(!isroot(x)){

int y=f[x];

rotate(x,ch[y][1]==x);

}

push\_up(x);

}

void access(int v){

for(int u=v,v=0;u;v=u,u=f[u]){

splay(u);

if(ch[u][1]) S[u].insert(ma[ch[u][1]]);

if(ch[u][1]==v) S[u].erase(S[u].find(ma[ch[u][1]]));

}

}

int find\_root(int v){

access(v);splay(v);for(;ch[v][0];v=ch[v][0]);return v;

}

int query(int v){

int root=find\_root(v);splay(root);

return (col[root]==col[v])?ma[root]:ma[ch[root][1]];

}

void link(int v){//fa[v]，v在原树的父亲；f[v]，v在伸展树的父亲

access(fa[v]);splay(fa[v]);splay(v);ch[f[v]=fa[v]][1]=v;

}

void cut(int v){

access(v);splay(v);f[ch[v][0]]=0;ch[v][0]=0;

}

void modify(int v,int x){

access(v);splay(v);key[v]=x;

}

}LCT[2];

int v[MAXN],next[MAXN],first[MAXN];

int tot=0;

void addedge(int x,int y){

v[++tot]=y;next[tot]=first[x];first[x]=tot;

v[++tot]=x;next[tot]=first[y];first[y]=tot;

}

void dfs(int u){

for(int i=first[u];i;i=next[i])

if(v[i]!=fa[u]){//fa[v]，v在原树的父亲；f[v]，v在伸展树的父亲

fa[v[i]]=u;LCT[col[v[i]]].f[v[i]]=u;

dfs(v[i]);

LCT[col[v[i]]].S[u].insert(LCT[col[v[i]]].ma[v[i]]);

}

LCT[0].push\_up(u);

LCT[1].push\_up(u);

}

void init(){

n=getint();

for(int i=1;i<n;i++){

int x=getint(),y=getint();

addedge(x,y);

}

for(int i=1;i<=n;i++) col[i]=getint();

for(int i=1;i<=n;i++)

LCT[0].key[i]=LCT[0].ma[i]=LCT[1].key[i]=LCT[1].ma[i]=getint();

dfs(1);

}

void solve(){

int q=getint();

int op,v,x;

while(q--){

op=getint(),v=getint();

if(op==0) printf("%d\n",LCT[col[v]].query(v));

else if(op==1){

if(fa[v]) LCT[col[v]].cut(v);

col[v]^=1;

if(fa[v]) LCT[col[v]].link(v);

}else{

x=getint();

LCT[0].modify(v,x);

LCT[1].modify(v,x);

}

}

}

int main(){

fread(buf,1,10000000,stdin);

init();

solve();

return 0;

}

QTREE7http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif QTREE7

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **DATE** | **PROBLEM** | **RESULT** | **TIME** | **MEM** |
| [12057230](http://www.spoj.com/files/src/12057230/" \o "View source code) | 2014-07-31 | [Query on a tree VII](http://www.spoj.com/problems/QTREE7/" \o "QTREE7) | accepted | [1.30](http://www.spoj.com/ranks/QTREE7/" \o "See the best solutions) | 35M |

标签: [树分治](http://www.cnblogs.com/fullpower/tag/%E6%A0%91%E5%88%86%E6%B2%BB/)