动态树就是很多splay组成的，每颗splay维护的是一条树链，splay之间用fa[]相连，如fa[x]=y表示以x为根的splay树的父亲是y，但是y的两个儿子（son[y][0]和son[y][1]）中没有一个是x

相对的，splay之间的父子关系也是通过fa[]和son[][]来维护的，不同的是，如果fa[x]=y,则必有son[y][0]或者son[y][1]等于x，这个一定要想明白！

对于无向图，就是先bfs建树，此时每一个点都是一颗splay，因为对于任意fa[x]=y,son[y][0]和son[y][1]一定不是x

isroot(x)：判断x是否为其所在splay树的根（依据上述关系）

splay(x)：将x旋转为其所在的splay树的根

access(x)：将x到root（即原树的根，不同于x所在的splay树的根）的路径变成实边，注意理解函数中y的意义

querysum(x,y)/querymax(x,y)：求x到y的路径上的点权和/最大点权，实质就是求包含x和y的splay的lca

　　先access(x)，此时从x到root变成一条实边，即在同一颗splay树种（此时x向下的实边已经断开了，也就是说x是其所在splay树种深度最大的点了）

　　再access(y)，此时从y到root变成一条实边，返回值就是lca（画个图挺好理解的，这里一定要想明白！）//此时x – lca – root 在x – lca 处 断开

其实动态树的精髓是link和cut，要加上旋转标记，就是将一颗splay树维护的链全部反向

以上是我的理解，应该是对的。。。嘿嘿。

代码觉得挺简单易懂的，可以看一下~

#include <iostream>

#include <algorithm>

#include <cstdlib>

#include <cstring>

#include <cstdio>

#define N 500000

#define INF 0x7f7f7f7f

using namespace std;

//http://www.cnblogs.com/proverbs/archive/2013/01/04/2845053.html

int n,m,cnt;

bool vis[N];

int to[N],next[N],head[N],q[N];

int fa[N],son[N][2],mx[N],val[N],sum[N];

inline void pushup(int x)

{

if(!x) return;

mx[x]=max(val[x],max(mx[son[x][0]],mx[son[x][1]]));

sum[x]=sum[son[x][0]]+sum[son[x][1]]+val[x];

}

inline bool isroot(int x)

{

return son[fa[x]][0]!=x&&son[fa[x]][1]!=x;

}

inline void link(int x,int y,int c)

{

fa[x]=y; son[y][c]=x;

}

inline void rotate(int x,int c)

{

int y=fa[x];

if(!isroot(y)) link(x,fa[y],son[fa[y]][1]==y);

else fa[x]=fa[y];

link(son[x][!c],y,c);

link(y,x,!c);

pushup(y);

}

inline void splay(int x)

{

while(!isroot(x))

{

int y=fa[x];

int cy=(son[fa[y]][1]==y),cx=(son[y][1]==x);

if(isroot(y)) rotate(x,cx);

else

{

if(cx==cy) rotate(y,cy);

else rotate(x,cx);

rotate(x,cy);

}

}

pushup(x);

}

inline int access(int x)

{

int y;

for(y=0;x;y=x,x=fa[x])

{

splay(x);

son[x][1]=y;

pushup(x);

}

return y;

}

inline int querysum(int x,int y)

{

access(x);

int lca=access(y);

splay(x);

if(lca==x) return val[lca]+sum[son[lca][1]];

else return val[lca]+sum[son[lca][1]]+sum[x];

}

inline int querymax(int x,int y)

{

access(x);

int lca=access(y);

splay(x);

if(lca==x) return max(val[lca],mx[son[lca][1]]);

else return max(val[lca],max(mx[son[lca][1]],mx[x]));

}

inline void change(int x,int w)

{

//access(x);

splay(x); val[x]=w; pushup(x);

}

inline void add(int u,int v)

{

to[cnt]=v; next[cnt]=head[u]; head[u]=cnt++;

}

inline void bfs()

{

memset(vis,0,sizeof vis);

vis[1]=true; q[1]=1;

int h=1,t=2,sta;

while(h<t)

{

sta=q[h++];

for(int i=head[sta];~i;i=next[i])

if(!vis[to[i]])

{

fa[to[i]]=sta;

vis[to[i]]=true;

q[t++]=to[i];

}

}

}

inline void go()

{

memset(head,-1,sizeof head); cnt=0;

memset(son,0,sizeof son);

memset(fa,0,sizeof fa);

mx[0]=-INF;

scanf("%d",&n);

for(int i=1,a,b;i<n;i++)

{

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

add(a,b); add(b,a);

}

for(int i=1;i<=n;i++) scanf("%d",&val[i]);

bfs();

scanf("%d",&m);

char str[10]; int a,b;

while(m--)

{

scanf("%s",str);

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

if(str[1]=='H') change(a,b);

else if(str[1]=='S') printf("%d\n",querysum(a,b));

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

}

}

int main()

{

go();

return 0;

}