Sectumsempra模板——xzj部分

树链剖分

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
using namespace std;
typedef long long LL;
const LL INF=2e9;
const LL N=2e5 + 100;
struct ones
{
    LL max_l,max_r,max_sum,sum,tag;
    ones(){}
    ones(LL max_l,LL max_r,LL max_sum,LL sum,LL tag):max_l(max_l),max_r(max_r),max_sum(max_sum),sum(sum),tag(tag){}
}tree[N*4];
struct Edge
{
    LL v,next;
    Edge(){}
    Edge(LL v,LL next):v(v),next(next){}
}edge[N*2];
LL edn,p[N],dep[N],fa[N],son[N],size[N],place[N],n,m,value[N],now,top[N],which[N];
LL MAX(LL a,LL b,LL c){
    return max(a,max(b,c));
}
ones operator + (const ones &a,const ones &b){
    return ones(max(a.max_l,a.sum + b.max_l),max(b.max_r,b.sum + a.max_r),MAX(a.max_sum,b.max_sum,a.max_r + b.max_l),a.sum + b.sum
}
void dfs1(LL x){
    dep[x]=dep[fa[x]] + 1;
    size[x]=1;
    for(LL i=p[x];~i;i=edge[i].next)
    {
        LL y=edge[i].v;
        if(y!=fa[x])
        {
            fa[y]=x;
            dfs1(y);
            size[x]+=size[y];
            if(size[y]>size[son[x]]) son[x]=y;
        }
    }
}
void dfs2(LL x){
    place[x]=++now;
    which[now]=x;
    if(son[x]==0) return;
    top[son[x]]=top[x];
    dfs2(son[x]);
    for(LL i=p[x];~i;i=edge[i].next)
    {
        LL y=edge[i].v;
        if(y!=fa[x] && y!=son[x])
        {
            top[y]=y;
            dfs2(y);
        }
    }
}
void modify(LL x,LL len,LL c)
{
```

```
    if(c>=0) tree[x]=ones(len * c,len * c,len * c,len * c,c);
    else tree[x]=ones(c,c,c,len * c,c);
}
void build (LL x,LL l,LL r){
    if(l+1==r) modify(x,1,value[which[l]]);else
    {
        LL mid=(l+r)/2;
        build(x*2,l,mid);
        build(x*2+1,mid,r);
        tree[x]=tree[x*2]+tree[x*2+1];
    }
}
void downtag(LL x,LL len)
{
    if(tree[x].tag==INF) return;
    modify(x * 2,len/2,tree[x].tag);
    modify(x*2+1,len-len/2,tree[x].tag);
    tree[x].tag=INF;
}
void change(LL x,LL l,LL r,LL ll,LL rr,LL c)
{
    if(l>=ll && r<=rr) modify(x,r-l,c);
    else
    {
        downtag(x,r-l);
        LL mid=(l+r)/2;
        if(ll<mid) change(x*2,l,mid,ll,rr,c);
        if(rr>mid) change(x*2+1,mid,r,ll,rr,c);
        tree[x]=tree[x*2]+tree[x*2+1];
    }
}
ones get(LL x,LL l,LL r,LL ll,LL rr)
{
    if(l>=ll && r<=rr) return tree[x];
    downtag(x,r-l);
    LL mid=(l+r)/2;
    ones tmp=ones(-INF,-INF,-INF,0,INF);
    if(ll<mid) tmp=get(x*2,l,mid,ll,rr) + tmp;
    if(rr>mid) tmp=tmp + get(x*2+1,mid,r,ll,rr);
    return tmp;
}
int main()
{
    cin>>n>>m;
    for(LL i=1;i<=n;i++)
        scanf("%lld",&value[i]);
    edn=0;memset(p,-1,sizeof(p));
    for(LL i=1;i<n;i++)
    {
        LL u,v;
        scanf("%lld%lld",&u,&v);
        edge[edn]=Edge(v,p[u]);p[u]=edn++;
        edge[edn]=Edge(u,p[v]);p[v]=edn++;
    }
    dfs1(1);
    top[1]=1;
    dfs2(1);
    build(1,1,n+1);
    while (m--)
    {
        LL type,u,v,c;
        scanf("%lld%lld%lld%lld",&type,&u,&v,&c);
        if(type==2)
        {
            ones ansl=ones(-INF,-INF,-INF,0,INF),ansr=ones(-INF,-INF,-INF,0,INF);
            while(true)
            {
```

```
                if(top[u]==top[v])
                {
                    if(dep[u]<dep[v])
                    {
                        swap(u,v);
                        swap(ansl,ansr);
                    }
                    ansl=get(1,1,n+1,place[v],place[u]+1) + ansl;
                    LL ans=MAX(ansl.max_sum,ansr.max_sum,ansl.max_l+ansr.max_l);
                    printf("%lld\n",ans);
                    break;
                }
                if(dep[top[u]]<dep[top[v]])
                {
                    swap(u,v);
                    swap(ansl,ansr);
                }
                ansl=get(1,1,n+1,place[top[u]],place[u]+1) + ansl;
                u=fa[top[u]];
            }
        }else
        {
            while (true)
            {
                if(top[u]==top[v])
                {
                    if(dep[u]<dep[v]) swap(u,v);
                    change(1,1,n+1,place[v],place[u]+1,c);
                    break;
                }
                if(dep[top[u]]<dep[top[v]]) swap(u,v);
                change(1,1,n+1,place[top[u]],place[u]+1,c);
                u=fa[top[u]];
            }
        }
    }
}
```