树链剖分

是什么

一种树的处理办法，在进行过树链剖分过后，书上任意一条链上的节点都可以用$O(logn)$个连续的区间表示

一些定义

一个点的$size$为一个点子树中包括该节点本身的节点个数。

一个点的重儿子为一个点的儿子中$size$最大的一个。如果有多个，那么就任取一个作为重儿子。

我们定义重边为一个点和其重儿子之间的边，而轻边为除了重边之外的其他边。

重边构成的链为重链。

我们定义一个点的top为该点所在的重链深度最小的点。

怎么剖

所谓的树链剖分其实也就是将一颗树的所有边分为轻链和重链两个部分。剖分效果如图。

树链剖分要经过两次深搜，第一次深搜以某个节点为根开始深搜得到每个点的重儿子。第二次深搜要维护每个点的top其代码如下。

```
void dfs1(int u)
{
    siz[u]=1;
    for(int i=head[u];~i;i=Edges[i].next)
    {
        int v=Edges[i].to;
        if(v==father[u])continue;
        father[v]=u;dep[v]=dep[u]+1;
        dfs1(v);siz[u]+=siz[v];
        if(siz[v]>siz[son[u]])son[u]=v;
    }
}
void dfs2(int u,int t)
{
    sz++,pos[u]=sz;
    inv[sz]=u,top[u]=t;
    if(son[u])dfs2(son[u],t);
    for(int i=head[u];~i;i=Edges[i].next)
    {
        int v=Edges[i].to;
```

```
        if(v==father[u]||v==son[u])continue;
        dfs2(v,v);
    }
}
```

那么此时，我们对一条链(x,y)的询问，我们假设x的top较深，那么我们就让x跳转到x的父亲，并处理x到x的top之间这一段重链。一直重复这个过程直到x和y的top相同，然后对当前两点间的重链进行处理。

示例代码如下。

```
void update(int x,int y)
{
    while (top[x]!=top[y])
    {
        if (dep[top[x]]<dep[top[y]])swap(x,y);
        query(num[top[x]],num[x]);
        x = fa[top[x]];
    }
    if (dep[x]<dep[y])swap(x,y);
    query(num[y],num[x]);
}
```

那么这样剖分并查询为什么时间复杂度是$O(logn)$的呢，首先dfs的时候优先处理每个点的重儿子，那么一条重链在dfs序列中式一段连续的区间。我们在查询的时候，每次跳到top的父亲都会走一条轻边，可以发现fa_{top_x}的$size$至少是top_x的$size$的二倍，那么我们最多跳转了$log(n)$次就会结束查询。

通常树链剖分算法会和线段树相结合。

怎么用

普通用法

[BZOJ 2243][SDOI2011]染色 | 线段树

n个节点的无根树□m次操作，可以给a,b路径上的节点染色，或是查询a,b路径上的颜色段数。如1122222111是3段□$n,m\le 10^5$

题解：显然树剖，用线段树维护区间的颜色段数以及记录左右两端的颜色便于合并。

code:

```
#include<iostream>
#include<cstdio>
#include<cstring>
#define MAXN 100005
#define RG register
```

```
using namespace std;
int n,m,head[MAXN],cnt=0,ans,col[MAXN];
int
sz=0,pos[MAXN],top[MAXN],dep[MAXN],son[MAXN],father[MAXN],siz[MAXN],inv[MAX
N];
inline int read()
{
    int f=1,x=0;char c=getchar();
    while(c<'0'||x>'9'){if(c=='-')f=-1;c=getchar();}
    while(c>='0'&&c<='9'){x=x*10+c-'0';c=getchar();}
    return f*x;
}
struct Node1{
    int next,to;
}Edges[MAXN*2];
inline void addedge(int u,int v)
{
    Edges[++cnt].next=head[u];
    head[u]=cnt;
    Edges[cnt].to=v;
}
inline void dfs1(int u)
{
    siz[u]=1;
    for(int i=head[u];~i;i=Edges[i].next)
    {
        int v=Edges[i].to;
        if(v==father[u])continue;
        father[v]=u;dep[v]=dep[u]+1;
        dfs1(v);siz[u]+=siz[v];
        if(siz[v]>siz[son[u]])son[u]=v;
    }
}
inline void dfs2(int u,int t)
{
    sz++,pos[u]=sz;
    inv[sz]=u,top[u]=t;
    if(son[u])dfs2(son[u],t);
    for(int i=head[u];~i;i=Edges[i].next)
    {
        int v=Edges[i].to;
        if(v==father[u]||v==son[u])continue;
        dfs2(v,v);
    }
}
struct Node2{
    int lc,rc,num,lazy;
}t[MAXN*3];
inline void pushdown(int idx,int l,int r)
{
    if(l!=r&&t[idx].lazy)
```

```
    {
        t[idx<<1].lazy=t[idx<<1|1].lazy=t[idx<<1].num=t[idx<<1|1].num=1;
        t[idx].lazy=0;
        t[idx<<1].lc=t[idx<<1|1].lc=t[idx<<1].rc=t[idx<<1|1].rc=t[idx].lc;
    }
}
inline void update(int idx)
{
    t[idx].lc=t[idx<<1].lc;
    t[idx].rc=t[idx<<1|1].rc;
    t[idx].num=t[idx<<1].num+t[idx<<1|1].num-
(t[idx<<1].rc==t[idx<<1|1].lc);
}
inline void build(int idx,int l,int r)
{
    t[idx].lazy=0;
    if(l==r)
    {
        t[idx].lc=t[idx].rc=col[inv[l]];
        t[idx].num=1;
        return;
    }
    int mid=(l+r)>>1;
    build(idx<<1,l,mid),build(idx<<1|1,mid+1,r);
    update(idx);
}
inline void color(int idx,int l,int r,int a,int b,int c)
{
    if(l==a&&r==b)
    {
        t[idx].num=t[idx].lazy=1;
        t[idx].lc=t[idx].rc=c;
        return;
    }
    pushdown(idx,l,r);
    int mid=(l+r)>>1;
    if(a>mid)color(idx<<1|1,mid+1,r,a,b,c);
    else if(b<=mid)color(idx<<1,l,mid,a,b,c);
    else
    {
        color(idx<<1|1,mid+1,r,mid+1,b,c);
        color(idx<<1,l,mid,a,mid,c);
    }
    update(idx);
}
inline int query(int idx,int l,int r,int a,int b)
{
    if((l==a&&r==b)||t[idx].lazy)return t[idx].num;
    pushdown(idx,l,r);
    int mid=(l+r)>>1,ans=0;
```

```
    if(a>mid)return query(idx<<1|1,mid+1,r,a,b);
    else if(b<=mid)return query(idx<<1,l,mid,a,b);
    else
    {
        ans+=query(idx<<1,l,mid,a,mid);
        ans+=query(idx<<1|1,mid+1,r,mid+1,b);
        ans-=t[idx<<1].rc==t[idx<<1|1].lc;
    }
    return ans;
}
inline int getcolor(int idx,int l,int r,int a)
{
    if(l==r||t[idx].lazy)return t[idx].lc;
    int mid=(l+r)>>1;
    if(a<=mid)return getcolor(idx<<1,l,mid,a);
    else return getcolor(idx<<1|1,mid+1,r,a);
}
inline void solve1(int x,int y,int z)
{
    while(top[x]!=top[y])
    {
        if(dep[top[x]]<dep[top[y]])swap(x,y);
        color(1,1,sz,pos[top[x]],pos[x],z);
        x=father[top[x]];
    }
    if(pos[x]<pos[y])swap(x,y);
    color(1,1,sz,pos[y],pos[x],z);
}
inline void solve2(int x,int y)
{
    ans=0;
    while(top[x]!=top[y])
    {
        if(dep[top[x]]<dep[top[y]])swap(x,y);
        ans+=query(1,1,sz,pos[top[x]],pos[x]);
if(getcolor(1,1,sz,pos[top[x]])==getcolor(1,1,sz,pos[father[top[x]]]))ans--
;
        x=father[top[x]];
    }
    if(pos[x]<pos[y])swap(x,y);
    ans+=query(1,1,sz,pos[y],pos[x]);
    printf("%d\n",ans);
}
int main()
{
    memset(head,-1,sizeof(head));
    n=read(),m=read();
    for(RG int i=1;i<=n;++i)col[i]=read();
    int x,y,z;
    for(RG int i=1;i<n;++i)
    {
```

```
        x=read(),y=read();
        addedge(x,y),addedge(y,x);
    }
    dfs1(1),dfs2(1,1);
    build(1,1,sz);
    char opt;
    for(RG int i=1;i<=m;++i)
    {
        opt=getchar();
        while(opt!='C'&&opt!='Q')opt=getchar();
        if(opt=='C')
        x=read(),y=read(),z=read(),solve1(x,y,z);
        else if(opt=='Q')
        x=read(),y=read(),solve2(x,y);
    }
    return 0;
}
```

[BZOJ 3531][Sdoi2014]旅行 | 动态开点线段树

给定N个节点的树，每个节点为一个城市，有自己的宗教和评级c_i,w_i旅行者只会留宿信仰相同的城市，并且旅行终点是他们信仰相同的城市。有四种操作，CC x c：城市x的居民全体改信了c教；CW x w：城市x的评级调整为wQS x y：查询x到y旅行途中留宿过的城市的评级总和；QM x y：查询x到y旅行途中留宿过城市的评级最大值$N,Q\le10^5,C\le10^5$

题解：没有宗教的话就是一道模板题。不过有了宗教我们也可以对每个宗教建一棵线段树，空间问题动态开点即可。

code:

```
#include<iostream>
#include<cstdlib>
#include<cstdio>
#include<cstring>
#define MAXN 100005
using namespace std;
int n,q,w[MAXN],c[MAXN],head[MAXN],cnt=0;
int father[MAXN],top[MAXN],maxv[MAXN],siz[MAXN],deep[MAXN],pos[MAXN],sz=0;
int tot=0,rt[MAXN],ls[MAXN*100],rs[MAXN*100],maxn[MAXN*100],sum[MAXN*100];
int read()
{
    int x=0,f=1;char c=getchar();
    while(c<'0'||c>'9'){if(c=='-')f=-1;c=getchar();}
    while(c>='0'&&c<='9'){x=x*10+c-'0';c=getchar();}
    return x*f;
}
```

```
struct Node1
{
    int next,to;
}Edges[MAXN*2];
void addedge(int u,int v)
{
    Edges[++cnt].next=head[u];
    head[u]=cnt;
    Edges[cnt].to=v;
}
void dfs1(int u)
{
    siz[u]=1;
    for(int i=head[u];~i;i=Edges[i].next)
    {
        int v=Edges[i].to;
        if(v==father[u])continue;
        father[v]=u,deep[v]=deep[u]+1;
        dfs1(v);
        siz[u]+=siz[v];
        if(siz[v]>siz[maxv[u]])maxv[u]=v;
    }
}
void dfs2(int u,int t)
{
    top[u]=t;
    ++sz,pos[u]=sz;
    if(maxv[u])dfs2(maxv[u],t);
    for(int i=head[u];~i;i=Edges[i].next)
    {
        int v=Edges[i].to;
        if(v==maxv[u]||v==father[u])continue;
        dfs2(v,v);
    }
}
void update(int idx)
{
    sum[idx]=sum[ls[idx]]+sum[rs[idx]];
    maxn[idx]=max(maxn[ls[idx]],maxn[rs[idx]]);
}
void change(int &idx,int l,int r,int p,int w)
{
    if(!idx)++tot,idx=tot;
    if(l==r){maxn[idx]=sum[idx]=w;return;}
    int mid=(l+r)>>1;
    if(p<=mid)change(ls[idx],l,mid,p,w);
    else change(rs[idx],mid+1,r,p,w);
    update(idx);
}
int querysum(int idx,int l,int r,int L,int R)
{
```

```
    if(!idx)return 0;
    if(L==l&&R==r)return sum[idx];
    int mid=(l+r)>>1;
    if(R<=mid)return querysum(ls[idx],l,mid,L,R);
    else if(L>mid)return querysum(rs[idx],mid+1,r,L,R);
    else return
querysum(ls[idx],l,mid,L,mid)+querysum(rs[idx],mid+1,r,mid+1,R);
}
int querymax(int idx,int l,int r,int L,int R)
{
    if(!idx)return 0;
    if(L==l&&R==r)return maxn[idx];
    int mid=(l+r)>>1;
    if(R<=mid)return querymax(ls[idx],l,mid,L,R);
    else if(L>mid)return querymax(rs[idx],mid+1,r,L,R);
    else return
max(querymax(ls[idx],l,mid,L,mid),querymax(rs[idx],mid+1,r,mid+1,R));
}
int QS(int x,int y)
{
    int t=c[x],res=0;
    while(top[x]!=top[y])
    {
        if(deep[top[x]]<deep[top[y]])swap(x,y);
        res+=querysum(rt[t],1,sz,pos[top[x]],pos[x]);
        x=father[top[x]];
    }
    if(deep[x]<deep[y])swap(x,y);
    res+=querysum(rt[t],1,sz,pos[y],pos[x]);
    return res;
}

int QM(int x,int y)
{
    int t=c[x],res=0;
    while(top[x]!=top[y])
    {
        if(deep[top[x]]<deep[top[y]])swap(x,y);
        res=max(res,querymax(rt[t],1,sz,pos[top[x]],pos[x]));
        x=father[top[x]];
    }
    if(deep[x]<deep[y])swap(x,y);
    res=max(res,querymax(rt[t],1,sz,pos[y],pos[x]));
    return res;
}
int main()
{
    memset(head,-1,sizeof(head));
    n=read(),q=read();
    for(int i=1;i<=n;i++)w[i]=read(),c[i]=read();
```

```
    for(int i=1;i<n;i++)
    {
        int x=read(),y=read();
        addedge(x,y),addedge(y,x);
    }
    deep[1]=1;dfs1(1),dfs2(1,1);
    for(int i=1;i<=n;i++)change(rt[c[i]],1,sz,pos[i],w[i]);
    for(int i=1;i<=q;i++)
    {
        char opt[5];
        scanf("%s",opt);
        int x=read(),y=read();
        if(opt[1]=='C')
        {
            change(rt[c[x]],1,sz,pos[x],0);
            change(rt[y],1,sz,pos[x],w[x]);
            c[x]=y;
        }
        else if(opt[1]=='W')
        {
            change(rt[c[x]],1,sz,pos[x],y);
            w[x]=y;
        }
        else if(opt[1]=='S')
        printf("%d\n",QS(x,y));
        else printf("%d\n",QM(x,y));
    }
    return 0;
}
```

进阶用法

感觉这两道题有一点点共性但又不是显式的，无论如何挺值得两连做的。建议思考，仿佛有智力开发的效果

[BZOJ 3626][LNOI2014]LCA | 线段树

给出n个点的有根树□q次询问，每次询问给出l,r,z□求$\sum\limits_{l\le i\le r}\text{dep}[\text{LCA}(i,z)]$□$1\le n,q\le50000$

题解：定义询问(x,z)为$\sum\limits_{1\le i\le x}\text{dep}[\text{LCA}(i,z)]$□则可以将题中询问拆为询问$(r,z)$和$(l-1,z)$的差。于是可以离线，从$1$到$n$逐个加入，加入时将该节点到根节点路径上点的值各加一，线段树维护。而对于每个z可以在加入x刚好完成时查询z到根节点路径上点的值来得到答案。

code:

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<algorithm>
#define MAXN 50005
#define Mod 201314
using namespace std;
int n,q,deep[MAXN],head[MAXN],cnt=0,tot=0;
int sz=0,maxv[MAXN],father[MAXN],siz[MAXN],top[MAXN],pos[MAXN],res[MAXN*2];
struct Node1
{
    int next,to;
}Edges[MAXN*2];
struct Node2
{
    int l,r,sum,lazy;
}t[MAXN*4];
struct Node3
{
    int x,z,id;
    Node3(int x=0,int z=0,int id=0):x(x),z(z),id(id){}
}Q[MAXN*2];
bool cmp(Node3 a,Node3 b){return a.x<b.x;}
int read()
{
    int x=0,f=1;char c=getchar();
    while(c<'0'||c>'9'){if(c=='-')f=-1;c=getchar();}
    while(c>='0'&&c<='9'){x=x*10+c-'0';c=getchar();}
    return x*f;
}
void addedge(int u,int v)
{
    Edges[++cnt].next=head[u];
    head[u]=cnt;
    Edges[cnt].to=v;
}
void dfs1(int u)
{
    siz[u]=1;int k=-1;
    for(int i=head[u];~i;i=Edges[i].next)
    {
        int v=Edges[i].to;
        deep[v]=deep[u]+1,dfs1(v);
        siz[u]+=siz[v];
        if(k==-1||siz[v]>siz[k])k=v;
    }
    maxv[u]=k;
}
```

```
void dfs2(int u,int t)
{
    ++sz,pos[u]=sz;
    top[u]=t;
    if(maxv[u]!=-1)dfs2(maxv[u],t);
    for(int i=head[u];~i;i=Edges[i].next)
    if(maxv[u]!=Edges[i].to)dfs2(Edges[i].to,Edges[i].to);
}
void build(int idx,int l,int r)
{
    t[idx].l=l,t[idx].r=r;
    if(l==r)return;
    int mid=(l+r)>>1;
    build(idx<<1,l,mid),build(idx<<1|1,mid+1,r);
}
void pushdown(int idx)
{
    if(t[idx].l<t[idx].r&&t[idx].lazy)
    {
        t[idx<<1].lazy+=t[idx].lazy;
        t[idx<<1|1].lazy+=t[idx].lazy;
        t[idx<<1].sum+=t[idx].lazy*(t[idx<<1].r-t[idx<<1].l+1);
        t[idx<<1].sum%=Mod;
        t[idx<<1|1].sum+=t[idx].lazy*(t[idx<<1|1].r-t[idx<<1|1].l+1);
        t[idx<<1|1].sum%=Mod;
        t[idx].lazy=0;
    }
}
void add(int idx,int l,int r)
{
    if(l<=t[idx].l&&r>=t[idx].r)
    {
        t[idx].sum+=t[idx].r-t[idx].l+1;
        t[idx].lazy++;return;
    }
    pushdown(idx);
    int mid=(t[idx].l+t[idx].r)>>1;
    if(r<=mid)add(idx<<1,l,r);
    else if(l>mid)add(idx<<1|1,l,r);
    else add(idx<<1,l,r),add(idx<<1|1,l,r);
    t[idx].sum=(t[idx<<1].sum+t[idx<<1|1].sum)%Mod;
}
int query(int idx,int l,int r)
{
    if(l<=t[idx].l&&r>=t[idx].r)return t[idx].sum;
    pushdown(idx);
    int mid=(t[idx].l+t[idx].r)>>1;
    if(r<=mid)return query(idx<<1,l,r);
    else if(l>mid)return query(idx<<1|1,l,r);
    else return (query(idx<<1,l,r)+query(idx<<1|1,l,r))%Mod;
}
```

```
void change(int x)
{
    while(top[x])
    {
        add(1,pos[top[x]],pos[x]);
        x=father[top[x]];
    }
    add(1,pos[top[x]],pos[x]);
}
int ask(int x)
{
    int res=0;
    while(top[x])
    {
        res=(res+query(1,pos[top[x]],pos[x]))%Mod;
        x=father[top[x]];
    }
    res=(res+query(1,pos[top[x]],pos[x]))%Mod;
    return res;
}
int main()
{
    memset(head,-1,sizeof(head));
    n=read(),q=read();
    for(int i=1;i<n;i++){father[i]=read();addedge(father[i],i);}
    deep[0]=1,dfs1(0);
    dfs2(0,0);build(1,1,sz);
    for(int i=1;i<=q;i++)
    {
        int l=read(),r=read(),z=read();
        ++tot,Q[tot]=Node3(l-1,z,tot);
        ++tot,Q[tot]=Node3(r,z,tot);
    }
    sort(Q+1,Q+1+tot,cmp);
    int now=0;
    for(int i=1;i<=tot;i++)
    {
        while(now<=Q[i].x)change(now),++now;
        res[Q[i].id]=ask(Q[i].z);
    }
    for(int i=1;i<=tot;i+=2)
    printf("%d\n",(res[i+1]-res[i]+Mod)%Mod);
    return 0;
}
```

[BZOJ 4012][HNOI2015]开店 | 主席树

给出n个点的带权树，节点i的妖怪有年龄x_i，Q个询问，给出u,l,r询问年龄在$[l,r]$的妖怪到u点的距离之和，强制在线，$n\le150000,Q\le200000$，年龄上限$A\le10^9$。

题解：首先点u,v间的距离可以转换为$\text{dis}(u,\text{root})+\text{dis}(v,\text{root})-\text{dis}(\text{lca}(u,v),\text{root})$，对于每个询问$\text{dis}(u,\text{root})$和$\sum\limits_{x_v\in[l,r]}\text{dis}(v,\text{root})$都是可以通过提前处理快速得到的，那么剩下需要的就是$\sum\limits_{x_v\in[l,r]}\text{dis}(\text{lca}(u,v),\text{root})$，这里就是和上一题有点像的地方了，我们可以将年龄离散化然后维护一个**主席树**从小到大按顺序加入结点，加入时将该结点到根的边权加到主席树里，之后对于每个询问就也可以直接在$l-1$和r查询u到根得到结果。但主席树的具体过程是有一点玄妙的，还有lazy tag什么的想象不出可以见代码，$w[i]$维护主席树中结点的父边边权的前缀和。

code:

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<algorithm>
#define N 150005
#define Q 200005
using namespace std;
typedef long long LL;
int n,q,A,x[N],y[N],head[N],cnt=0;
int father[N],deep[N],pos[N],top[N],siz[N],maxv[N],sz=0;
int rt[N],ls[N*120],rs[N*120],tot=0;
LL sum[N*120],lazy[N*120],dis[N],d[N],w[N],p1[N],p2[N];
LL read()
{
    LL x=0,f=1;char c=getchar();
    while(c<'0'||c>'9'){if(c=='-')f=-1;c=getchar();}
    while(c>='0'&&c<='9'){x=x*10+c-'0';c=getchar();}
    return x*f;
}
struct Node1
{
    int next,to,w;
}Edges[N*2];
struct Node2
{
    int age,id;
    Node2(int age=0,int id=0):age(age),id(id){}
    bool operator < (const Node2& t) const
    {return age<t.age;}
}data[N];
void addedge(int u,int v,int w)
{
    Edges[++cnt].next=head[u];
```

```
    head[u]=cnt;
    Edges[cnt].to=v;
    Edges[cnt].w=w;
}
void dfs1(int u)
{
    siz[u]=1;
    for(int i=head[u];~i;i=Edges[i].next)
    {
        int v=Edges[i].to;
        if(v==father[u])continue;
        father[v]=u;
        deep[v]=deep[u]+1,d[v]=Edges[i].w,dis[v]=dis[u]+Edges[i].w;
        dfs1(v);siz[u]+=siz[v];
        if(!maxv[u]||siz[v]>siz[maxv[u]])maxv[u]=v;
    }
}
void dfs2(int u,int t)
{
    top[u]=t;
    ++sz,pos[u]=sz;
    w[sz]=w[sz-1]+d[u],p1[x[u]]++,p2[x[u]]+=dis[u];
    if(maxv[u])dfs2(maxv[u],t);
    for(int i=head[u];~i;i=Edges[i].next)
    {
        int v=Edges[i].to;
        if(v==maxv[u]||v==father[u])continue;
        dfs2(v,v);
    }
}
void insert(int &idx,int last,int l,int r,int L,int R)
{
    ++tot,idx=tot;
sum[idx]=sum[last],lazy[idx]=lazy[last],ls[idx]=ls[last],rs[idx]=rs[last];
    if(L==l&&R==r){lazy[idx]++;sum[idx]+=w[r]-w[l-1];return;}
    int mid=(l+r)>>1;
    if(R<=mid)insert(ls[idx],ls[last],l,mid,L,R);
    else if(L>mid)insert(rs[idx],rs[last],mid+1,r,L,R);
    else
insert(ls[idx],ls[last],l,mid,L,mid),insert(rs[idx],rs[last],mid+1,r,mid+1,
R);
    sum[idx]=sum[ls[idx]]+sum[rs[idx]]+lazy[idx]*(w[r]-w[l-1]);
}
LL query(int s,int t,int l,int r,int L,int R,int tag)
{
    if(L==l&&R==r)return sum[s]-sum[t]+1LL*tag*(w[r]-w[l-1]);
    int mid=(l+r)>>1;tag+=lazy[s]-lazy[t];
    if(R<=mid)return query(ls[s],ls[t],l,mid,L,R,tag);
    else if(L>mid)return query(rs[s],rs[t],mid+1,r,L,R,tag);
    else return
```

```
query(ls[s],ls[t],l,mid,L,mid,tag)+query(rs[s],rs[t],mid+1,r,mid+1,R,tag);
}
void add(int i,int x)
{
    while(x)
    {
        insert(rt[data[i].age],rt[data[i].age],1,sz,pos[top[x]],pos[x]);
        x=father[top[x]];
    }
}
LL ask(int l,int r,int u)
{
    LL res=0;
    while(u)
    {
        res+=query(rt[r],rt[l-1],1,sz,pos[top[u]],pos[u],0);
        u=father[top[u]];
    }
    return res;
}
int main()
{
    memset(head,-1,sizeof(head));
    n=read(),q=read(),A=read();
    for(int i=1;i<=n;i++)y[i]=x[i]=read();
    sort(y+1,y+1+n);
    int t=unique(y+1,y+1+n)-y-1;
    for(int i=1;i<=n;i++)x[i]=lower_bound(y+1,y+1+t,x[i])-y;
    for(int i=1;i<n;i++)
    {
        int a=read(),b=read(),c=read();
        addedge(a,b,c),addedge(b,a,c);
    }
    dfs1(1);dfs2(1,1);
    for(int i=1;i<=n;i++)data[i]=Node2(x[i],i);
    sort(data+1,data+1+n);
    for(int i=1;i<=n;i++)
    {
        if(!rt[data[i].age])
        rt[data[i].age]=rt[data[i].age-1];
        add(i,data[i].id);
    }
    for(int i=1;i<=t;i++)p1[i]+=p1[i-1],p2[i]+=p2[i-1];
    LL last=0;
    for(int i=1;i<=q;i++)
    {
        LL u=read(),a=read(),b=read();
        int l=min((a+last)%A,(b+last)%A),r=max((a+last)%A,(b+last)%A);
        l=lower_bound(y+1,y+1+t,l)-y,r=upper_bound(y+1,y+1+t,r)-y-1;
        if(l>r||l>t||r<1){last=0;printf("0\n");continue;}
        last=dis[u]*(p1[r]-p1[l-1])+p2[r]-p2[l-1]-2*ask(l,r,u);
```

```
            printf("%lld\n",last);
        }
        return 0;
}
```