# BZOJ-4034: [HAOI2015]树上操作 （树链剖分 入门题 子树整体修改 线段树 区间修改+查询）

## Description

有一棵点数为 N 的树，以点 1 为根，且树点有边权。然后有 M 个

操作，分为三种：

操作 1 ：把某个节点 x 的点权增加 a 。

操作 2 ：把某个节点 x 为根的子树中所有点的点权都增加 a 。

操作 3 ：询问某个节点 x 到根的路径中所有点的点权和。

## Input

第一行包含两个整数 N, M 。表示点数和操作数。接下来一行 N 个整数，表示树中节点的初始权值。接下来 N-1

行每行三个正整数 fr, to ， 表示该树中存在一条边 (fr, to) 。再接下来 M 行，每行分别表示一次操作。其中

第一个数表示该操作的种类（ 1-3 ） ，之后接这个操作的参数（ x 或者 x a ） 。

## Output

对于每个询问操作，输出该询问的答案。答案之间用换行隔开。

## Sample Input

5 5

1 2 3 4 5

1 2

1 4

2 3

2 5

3 3

1 2 1

3 5

2 1 2

3 3

## Sample Output 6 9 13

## HINT

 对于 100% 的数据， N,M<=100000 ，且所有输入数据的绝对值都不会超过 10^6 。

#include <bits/stdc++.h>

using namespace std;

#define maxn 100005

int a[maxn], sz[maxn], dep[maxn], id[maxn], val[maxn], son[maxn], mx[maxn], top[maxn], pre[maxn], tot;

vector<vector<int> >g(maxn);

struct tree{

long long sm, lazy;

}c[maxn \* 4];

void dfs(int x, int fa, int d){

sz[x] = 1;

pre[x] = fa;

dep[x] = d;

son[x] = 0;

int cur;

for(int i = 0; i < g[x].size(); ++i){

cur = g[x][i];

if(cur == fa) continue;

dfs(cur, x, d + 1);

sz[x] += sz[cur];

if(sz[son[x]] < sz[cur]){

son[x] = cur;

}

}

}

void dfs1(int x, int tp){

id[x] = ++tot;

top[x] = tp;

if(son[x]){

dfs1(son[x], tp);

}

int cur;

for(int i = 0; i < g[x].size(); ++i){

cur = g[x][i];

if(cur == pre[x] || cur == son[x]) continue;

dfs1(cur, cur);

}

mx[x] = tot;

}

void build(int o, int l, int r){

c[o].lazy = 0;

if(l == r){

c[o].sm = val[l];

return;

}

int mid = l + r >> 1;

build(o << 1, l, mid);

build(o << 1 | 1, mid + 1, r);

c[o].sm = c[o << 1].sm + c[o << 1 | 1].sm;

}

void pushdown(int o, int l, int r){

if(l == r || c[o].lazy == 0) return;

int mid = l + r >> 1;

c[o << 1].lazy += c[o].lazy;

c[o << 1 | 1].lazy += c[o].lazy;

c[o << 1].sm += c[o].lazy \* (mid - l + 1);

c[o << 1 | 1].sm += c[o].lazy \* (r - mid);

c[o].lazy = 0;

}

long long query(int o, int l, int r, int L, int R){

pushdown(o, l, r);

if(l >= L && r <= R){

return c[o].sm;

}

int mid = l + r >> 1;

long long ans = 0;

if(mid >= L) ans += query(o << 1, l, mid, L, R);

if(mid < R) ans += query(o << 1 | 1, mid + 1, r, L, R);

return ans;

}

long long getsm(int x){

int tp = top[x];

long long ans = 0;

while(tp != 1){

ans += query(1, 1, tot, id[tp], id[x]);

x = pre[tp];

tp = top[x];

}

ans += query(1, 1, tot, 1, id[x]);

return ans;

}

void add(int o, int l, int r, int L, int R, int v){

pushdown(o, l, r);

if(l >= L && r <= R){

c[o].lazy += v;

c[o].sm += 1LL \* v \* (r - l + 1);

return;

}

int mid = l + r >> 1;

if(mid >= L) add(o << 1, l, mid, L, R, v);

if(mid < R) add(o << 1 | 1, mid + 1, r, L, R, v);

c[o].sm = c[o << 1].sm + c[o << 1 | 1].sm;

}

int main(){

int n, m, u, v, op;

scanf("%d %d", &n, &m);

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

scanf("%d", &a[i]);

}

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

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

g[u].push\_back(v);

g[v].push\_back(u);

}

tot = 0;

dfs(1, 0, 1);

dfs1(1, 1);

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

val[id[i]] = a[i];

}

build(1, 1, tot);

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

scanf("%d", &op);

if(op == 1){

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

add(1, 1, tot, id[u], id[u], v);

}

else if(op == 2){

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

add(1, 1, tot, id[u], mx[u], v);

}

else{

scanf("%d", &u);

printf("%lld\n", getsm(u));

}

}

}

/\*

题意：一棵树，1e5个节点，每个节点有权值，1e5次操作，每次操作要么修改某节点的值，

要么修改节点以及其子树中所有点的值，要么询问某点到根路径上所有点的权值和。

思路：

树链剖分的裸题，在线段树上做区间的修改和查询操作，对于修改子树中所有点的操作，关键

在于怎么把子树上所有的点对应到线段树上，其实一颗子树中所有点所在区间就是其中节点的

最小标号到最大标号，dfs性质决定的，知道这个就简单了，维护一下最大标号即可，最小标号

就是根节点自己。单点修改和路径和查询就是基本操作了。

\*/

# // 树上的询问 树链剖分 区间修改 单点查询

给你一棵具有N个点（编号为1到N）M条边的树，并给定各个点权的值，然后有3种操作：

I C1 C2 K：把C1与C2的路径上的所有点权值加上K

D C1 C2 K：把C1与C2的路径上的所有点权值减去K

Q C：查询节点编号为C的权值

Input

对于每组测试数据的第一行包含三个整数N,M,P，分别表示N个结点(1<=N<=50000)，M条边(M=N-1)和P(1≤P≤100000)个操作。

接下来一行为N个整数A1,A2,...AN(0≤Ai≤1000),表示一开始每个结点的值。

接下来的M行，每行两个整数u和v，表示u到v有一条边。

接下来P行，表示相应的操作，其中0<=k<=1000。

Output对于每个询问输出一行，为询问结点的值。

Sample Input

3 2 5

1 2 3

2 1

2 3

I 1 3 5

Q 2

D 1 2 2

Q 1

Q 3

Sample Output 7 4 8

#include<iostream>

#include<cstdio>

#include<cstring>

#define L(x) (x<<1)

#define R(x) (x<<1|1)

using namespace std;

struct Edge

{

int to,next;

}w[300005];

int cnt=0,h[300005]={0};

struct SegmentTree

{

int l,r,sum,lazy;

}tree[300005];

int fa[300005]={0},tid[300005]={0},dep[300005]={0},son[300005]={0},size[300005]={0},top[300005]={0};

int rank[300005]={0}, st[300005]={0}, tim=0,n,q,m;

void add(int x,int y)

{ cnt++;w[cnt].to=y;w[cnt].next=h[x];h[x]=cnt;}

void DFS1(int x,int f,int d)

{

dep[x]=d;

fa[x]=f;

size[x]=1;

for(int i=h[x];i;i=w[i].next)

{

int to=w[i].to;

if(to!=f)

{

DFS1(to,x,d+1);

size[x]+=size[to];

if(son[x]==0||size[to]>size[son[x]])son[x]=to;

}

}

}

void DFS2(int x,int tp)

{

top[x]=tp;

tid[x]=++tim;

rank[tim]=x;//反向映射

if(son[x]==0)return;

DFS2(son[x],tp);

for(int i=h[x];i;i=w[i].next)

{

int to=w[i].to;

if(to!=son[x]&&to!=fa[x])DFS2(to,to);

}

}

void Pushup(int x)

{ tree[x].sum=tree[x\*2].sum+tree[x\*2+1].sum;}

void Pushdown(int x)

{

if(tree[x].lazy)

{

int d=tree[x].lazy;

tree[L(x)].sum+=d;

tree[R(x)].sum+=d;

tree[L(x)].lazy+=d;

tree[R(x)].lazy+=d;

tree[x].lazy=0;//清零哦。

}

}

void Build(int l,int r,int root)

{

tree[root].l=l;

tree[root].r=r;

tree[root].sum=0;tree[root].lazy=0;

if(l==r)//l=r时就直接给他赋值

{

tree[root].sum=st[rank[l]];

return;

}

int mid=(l+r)>>1;

Build(l,mid,L(root));

Build(mid+1,r,R(root));

Pushup(root);//儿子变了，合并标记给父亲节点

}

void update(int l,int r,int root,int delta)

{

if(l<=tree[root].l&&tree[root].r<=r)

{

tree[root].sum+=delta;

tree[root].lazy+=delta;//懒惰标记一起加哦。

return;

}

Pushdown(root);

int mid=(tree[root].l+tree[root].r)>>1;

if(l<=mid)update(l,r,L(root),delta);

if(r>mid)update(l,r,R(root),delta);

}

void change(int x,int y,int v)

{

int t1=top[x],t2=top[y];

while(t1!=t2)

{

if(dep[t1]<dep[t2]){swap(t1,t2);swap(x,y);}

update(tid[t1],tid[x],1,v);

x=fa[t1];t1=top[x];

}

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

update(tid[x],tid[y],1,v);

}

int ask(int x,int root)

{

if(tree[root].l==x&&tree[root].r==x)return tree[root].sum;

Pushdown(root);//一定要relax一下，更新下层标记

int mid=(tree[root].l+tree[root].r)>>1;

if(x<=mid)return ask(x,L(root));

else return ask(x,R(root));

}

void pre()

{

memset(w,0,sizeof(w));

memset(h,0,sizeof(h));

memset(son,0,sizeof(son));

memset(size,0,sizeof(size));

memset(rank,0,sizeof(rank));

memset(tid,0,sizeof(tid));

memset(dep,0,sizeof(dep));

memset(top,0,sizeof(top));

memset(fa,0,sizeof(fa));

tim=0;

cnt=0;

}

int main()

{

while(scanf("%d%d%d",&n,&m,&q) != EOF)

{

pre();

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

int a,b;

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

{

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

add(a,b);

add(b,a);

}

DFS1(1,0,0);DFS2(1,1);

Build(1,n,1);

while(q--)

{

char cmd;

scanf("%s",&cmd);

if(cmd=='Q')

{

int a;

scanf("%d",&a);

printf("%d\n",ask(tid[a],1));

}

if(cmd=='I')

{

int a,b,c;

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

change(a,b,c);

}

if(cmd=='D')

{

int a,b,c;

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

change(a,b,-c);

}

}

}

}