# Xianduanshu

6133

#include<cstdio>

#include<cstring>

#include<iostream>

#include<algorithm>

#include<vector>

using namespace std;

const int M= 1e+5+123;

int cnt;

struct pos{

int l,r,sz;

long long sum,val;

pos(int l=0,int r=0,int sz=0,long long sum=0,long long val=0) : l(l),r(r),sz(sz),sum(sum),val(val){}

}tree[M\*18];

void build(int &node,int l,int r,int pos,int val){

node = ++cnt;

tree[node].sum = tree[node].val = val;

tree[node].sz = 1;

tree[node].l = tree[node].r = 0;

if(l == r) return;

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

if(pos <= mid) build(tree[node].l,l,mid,pos,val);

else build(tree[node].r,mid+1,r,pos,val);

}

int merg(int A,int B,int l,int r){

if(A==0||B==0) return A+B;

if(l==r){

tree[A].val=tree[A].val+tree[B].val;

tree[A].val=tree[A].val+tree[B].sz\*tree[A].sum;

tree[A].sz+=tree[B].sz;

tree[A].sum+=tree[B].sum;

return A;

}

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

tree[A].l=merg(tree[A].l,tree[B].l,l,mid);

tree[A].r=merg(tree[A].r,tree[B].r,mid+1,r);

tree[A].sum =tree[tree[A].l].sum + tree[tree[A].r].sum;

tree[A].sz = tree[tree[A].l].sz + tree[tree[A].r].sz;

tree[A].val =tree[tree[A].l].val + tree[tree[A].r].val;

tree[A].val += tree[tree[A].l].sum \* tree[tree[A].r].sz;

return A;

}

int n;

int rot[M],vis[M];

vector<int> G[M];

long long Ans[M];

int stk[M];

int fa[M];

int dfs2(int u){

int sz=1;

stk[1]=u;

fa[u]=0;

while(sz){

int u=stk[sz];

vis[u]++;

if(vis[u]==1){

for(int i=0;i<G[u].size();i++){

int v=G[u][i];

if(v != fa[u]){

fa[v]=u;

stk[++sz]=v;

}

}

}

if(vis[u]==2){

Ans[u] = tree[rot[u]].val;

if(fa[u])merg(rot[fa[u]],rot[u],1,n);

sz--;

}

//printf("%d %d %d\n",u,sz,vis[u]);

}

return 0;

}

int c[M],b[M];

int init(){

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

tree[i].val=0,tree[i].sum=0;

cnt=0;

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

G[i].clear();

Ans[i]=0;

fa[i]=0;

rot[i]=0;

vis[i]=0;

c[i]=0;

}

return 0;

}

struct value{

int x,y;

value(int x=0,int y=0):x(x),y(y){}

}a[M];

int cmp(value A,value B){

return A.x<B.x;

}

int main(){

// freopen("1001.in","r",stdin);

// freopen("1001.ans2","w",stdout);

int T;

scanf("%d",&T);

while(T--){

scanf("%d",&n);

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

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

a[i].y=i;

}

sort(1+a,1+a+n,cmp);

int ct=0,last=-1;

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

if(last!=a[i].x){

ct++;

last=a[i].x;

}

c[a[i].y]=ct;

b[ct]=a[i].x;

}

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

build(rot[i],1,n,c[i],b[c[i]]);

}

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

int x,y;

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

G[x].push\_back(y);

G[y].push\_back(x);

}

dfs2(1);

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

printf("%lld",Ans[i]);

if(i!=n) printf(" ");

else printf("\n");

}

init();

}

return 0;

}

#include<cstdio>

#include<cstring>

#include<algorithm>

#include<iostream>

using namespace std;

const int M=1e+5+123;

const int mod=1e+9+7;

struct matrix{

int r,c;

long long a[4][4];

}P[M\*4];

matrix init(){

matrix A;

for(int i=1;i<=3;i++)

for(int j=1;j<=3;j++)

A.a[i][j]=0;

return A;

}

int tag[M\*4];

/\*int prin(matrix A){

printf("ppppppppppppppp\n");

for(int i=1;i<=A.r;i++){

for(int j=1;j<=A.c;j++){

printf("%d ",A.a[i][j]);

}

printf("\n");

}

return 0;

}\*/

matrix operator \* (const matrix &A,const matrix &B){

matrix C=init();

C.r=A.r,C.c=B.c;

C.a[1][3]=0,C.a[2][3]=0,C.a[3][3]=1;

for(int k=1;k<=3;k++){

for(int i=1;i<=3;i++){

for(int j=1;j<=2;j++){

C.a[i][j]+=A.a[i][k]\*B.a[k][j];

if(C.a[i][j]>=mod)C.a[i][j]%=mod;

}

}

}

return C;

}

matrix set1(){

matrix C;

C.r=3,C.c=3;

C.a[1][1]=1,C.a[1][2]=0,C.a[1][3]=0;

C.a[2][1]=1,C.a[2][2]=1,C.a[2][3]=0;

C.a[3][1]=1,C.a[3][2]=0,C.a[3][3]=1;

return C;

}

matrix set0(){

matrix C;

C.r=3,C.c=3;

C.a[1][1]=1,C.a[1][2]=1,C.a[1][3]=0;

C.a[2][1]=0,C.a[2][2]=1,C.a[2][3]=0;

C.a[3][1]=0,C.a[3][2]=1,C.a[3][3]=1;

return C;

}

matrix set2(){

matrix C;

C.r=3,C.c=3;

C.a[1][1]=1,C.a[1][2]=0,C.a[1][3]=0;

C.a[2][1]=0,C.a[2][2]=1,C.a[2][3]=0;

C.a[3][1]=0,C.a[3][2]=0,C.a[3][3]=1;

return C;

}

char s[M];

int swp(int num){

/\*for(int i=1;i<=3;i++)

swap(P[num].a[1][i],P[num].a[2][i]);

for(int i=1;i<=3;i++)

swap(P[num].a[i][1],P[num].a[i][2]);

\*/

swap(P[num].a[1][1],P[num].a[2][2]);

swap(P[num].a[1][2],P[num].a[2][1]);

swap(P[num].a[3][1],P[num].a[3][2]);

return 0;

}

int pushdown(int num){

int ls=num<<1,rs=num<<1|1;

if(tag[num]){

tag[ls]=tag[ls]^1,

tag[rs]=tag[rs]^1,

swp(ls);

swp(rs);

tag[num]=0;

}

return 0;

}

int pushup(int num){

P[num]=P[num<<1]\*P[num<<1|1];

return 0;

}

int build(int num,int l,int r){

if(l==r){

if(s[l]=='1') P[num]=set1();

else P[num]=set0();

return 0;

}

P[num]=set2();

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

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

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

if(l!=r) pushup(num);

return 0;

}

int modefy(int num,int l,int r,int left,int right){

if(l>=left&&r<=right){

tag[num]^=1;

swp(num);

return 0;

}

pushdown(num);

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

if(left<=mid) modefy(num<<1,l,mid,left,right);

if(right>mid) modefy(num<<1|1,mid+1,r,left,right);

if(l!=r)pushup(num);

return 0;

}

matrix Ans;

int query(int num,int l,int r,int left,int right){

if(l>=left&&r<=right){

Ans=Ans\*P[num];

//prin(P[num]);

return 0;

}

pushdown(num);

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

if(left<=mid) query(num<<1,l,mid,left,right);

if(right>mid) query(num<<1|1,mid+1,r,left,right);

//if(l!=r)pushup(num);

return 0;

}

int input() {

int res = 0, ch, flag = 0;

if((ch = getchar()) == '-') flag = 1; ///正负

else if(ch >= '0' && ch <= '9') res = ch - '0';

while((ch = getchar()) >= '0' && ch <= '9' ) res = res \* 10 + ch - '0';

return flag ? -res : res;

}

int main(){

freopen("A.in","r",stdin);

int T=input();

P[0]=set2();

while(T--){

int n,m;

n=input();m=input();

memset(tag,0,sizeof(tag));

scanf("%s",s+1);

build(1,1,n);

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

int x,y,z;

z=input(),x=input(),y=input();

if(z==1)modefy(1,1,n,x,y);

if(z==2) {

Ans=set2();

query(1,1,n,x,y);

//prin(Ans);

printf("%lld\n",(Ans.a[3][1]+Ans.a[3][2])%mod);

}

}

}

return 0;

}

/\*

Ans2.r=1,Ans2.c=3;

if(s[x]=='1')Ans2.a[1][1]=1,Ans2.a[1][2]=0,Ans2.a[1][3]=1;

if(s[x]=='0')Ans2.a[1][1]=0,Ans2.a[1][2]=1,Ans2.a[1][3]=1;

\*/