# ACM模板

# 1数学知识

### 1.1数论

#### 1.1.1 FFT

//2e5要1s左右

#include<bits/stdc++.h>

using namespace std;

typedef complex<double> E;

const int N=4000005;

const double pi=acos(-1);

int n,m,L,R[N];

E A[N],B[N];

void fft(E \*a,int f){

for(int i=0;i<n;i++)if(i<R[i])swap(a[i],a[R[i]]);

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

E wn(cos(pi/i),f\*sin(pi/i));

for(int p=i<<1,j=0;j<n;j+=p){

E w(1,0);

for(int k=0;k<i;k++,w\*=wn){

E x=a[j+k],y=w\*a[j+k+i];

a[j+k]=x+y;a[j+k+i]=x-y;

}

}

}

}

signed main(){

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

for (int i=0;i<=n;i++)scanf("%lf",&A[i]);

for (int i=0;i<=m;i++)scanf("%lf",&B[i]);

m=n+m;for(n=1;n<=m;n<<=1)L++;

for (int i=0;i<n;i++)R[i]=(R[i>>1]>>1)|((i&1)<<(L-1));

fft(A,1);fft(B,1);

for (int i=0;i<n;i++)A[i]=A[i]\*B[i];

fft(A,-1);

for (int i=0;i<=m;i++)printf("%d ",(int)(A[i].real()/n+0.5));puts("");

return 0;

}

单位根反演

#### 1.1.2 NTT

∀i,j(1⩽i,j⩽p−1,i<j),g^i % p !=g^j %p 则g为p的原根

常见的998244353,1004535809,469762049的原根是3

ordn(t)为t的几次方%n为1，phi(n)=ordn(t)则t为原根

#include<bits/stdc++.h>

using namespace std;

const int N=4000005;

const int G=3,mod=998244353;//(119<<23)+1

int n,m,L,R[N];

int A[N],B[N];

int qpow(int x,int y){

int ans=1;

while (y){

if (y&1)ans=1ll\*ans\*x%mod;y>>=1;x=1ll\*x\*x%mod;

}

return ans;

}

void NTT(int\* a,int f){

for (int i=0;i<n;i++)if(i<R[i])swap(a[i],a[R[i]]);

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

int gn=qpow(G,(mod-1)/(i<<1));

for (int j=0;j<n;j+=(i<<1)){

int g=1;

for (int k=0;k<i;k++,g=1ll\*g\*gn%mod){

int x=a[j+k],y=1ll\*g\*a[j+k+i]%mod;

a[j+k]=(x+y)%mod;a[j+k+i]=(x-y+mod)%mod;

}

}

}

if (f==1)return;

int nv=qpow(n,mod-2);reverse(a+1,a+n);

for (int i=0;i<n;i++)a[i]=1ll\*a[i]\*nv%mod;

}

signed main(){

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

for (int i=0;i<=n;i++)scanf("%d",&A[i]);

for (int i=0;i<=m;i++)scanf("%d",&B[i]);

m=n+m;for(n=1;n<=m;n<<=1)L++;

for (int i=0;i<n;i++)R[i]=(R[i>>1]>>1)|((i&1)<<(L-1));

NTT(A,1);NTT(B,1);

for (int i=0;i<n;i++)A[i]=1ll\*A[i]\*B[i]%mod;

NTT(A,-1);

for (int i=0;i<=m;i++)printf("%d ",A[i]);puts("");

return 0;

}

#### 1.1.3 FWT

*Ci*​=∑*(i=j*⊕*k)* ​*Aj*​×*Bk*​

#include<bits/stdc++.h>

using namespace std;

const int N=(1<<17)+5;

typedef long long ll;

static int n,Len=1<<17;

const int mod=998244353;

inline int ad(int u,int v){return (u+=v)>=mod?u-mod:u;}

inline void FMT(int \*a){//or

for(register int z=1;z<Len;z<<=1)

for (int j=0;j<Len;j++)if(z&j)a[j]=ad(a[j],a[j^z]);

}

inline void IFMT(int \*a){

for(register int z=1;z<Len;z<<=1)

for (int j=0;j<Len;j++)if(z&j)a[j]=ad(a[j],mod-a[j^z]);

}

inline void FWTand(int \*a){

for(register int i=2,ii=1;i<=Len;i<<=1,ii<<=1)

for(register int j=0;j<Len;j+=i)

for(register int k=0;k<ii;++k)

a[j+k]=ad(a[j+k],a[j+k+ii]);

}

inline void IFWTand(int \*a){

for(register int i=2,ii=1;i<=Len;i<<=1,ii<<=1)

for(register int j=0;j<Len;j+=i)

for(register int k=0;k<ii;++k)

a[j+k]=ad(a[j+k],mod-a[j+k+ii]);

}

inline void FWTxor(int \*a){

static int t;

for(register int i=2,ii=1;i<=Len;i<<=1,ii<<=1)

for(register int j=0;j<Len;j+=i)

for(register int k=0;k<ii;++k){

t=a[j+k];

a[j+k]=ad(t,a[j+k+ii]);

a[j+k+ii]=ad(t,mod-a[j+k+ii]);

}

}

inline int div2(int x){return x&1?(mod+x)/2:x/2;}

inline void IFWTxor(int \*a){

static int t;

for(register int i=2,ii=1;i<=Len;i<<=1,ii<<=1)

for(register int j=0;j<Len;j+=i)

for(register int k=0;k<ii;++k){

t=a[j+k];

a[j+k]=div2(ad(t,a[j+k+ii]));

a[j+k+ii]=div2(ad(t,mod-a[j+k+ii]));

}

}

static int A[N],B[N],C[N];

inline void init(){

cin>>n;Len=1<<n;

for (int i=0;i<Len;i++)cin>>A[i];

for (int i=0;i<Len;i++)cin>>B[i];

}

inline void solve(){

FMT(A);FMT(B);

for (int i=0;i<Len;i++)C[i]=(unsigned long long)A[i]\*B[i]%mod;

IFMT(A);IFMT(B);IFMT(C);

for (int i=0;i<Len;i++)cout<<C[i]<<" ";putchar('\n');

FWTand(A);FWTand(B);

for (int i=0;i<Len;i++)C[i]=(unsigned long long)A[i]\*B[i]%mod;

IFWTand(A);IFWTand(B);IFWTand(C);

for (int i=0;i<Len;i++)cout<<C[i]<<" ";putchar('\n');

FWTxor(A);FWTxor(B);

for (int i=0;i<Len;i++)C[i]=(unsigned long long)A[i]\*B[i]%mod;

IFWTxor(C);

for (int i=0;i<Len;i++)cout<<C[i]<<" ";putchar('\n');

}

int main(){

init();

solve();

return 0;

}

#### 1.1.4整除分块

//n/i的和i=1~n

for(int l=1,r;l<=n;l=r+1){

r=n/(n/l);

ans+=(r-l+1)\*(n/l);

}

#### 1.1.5拉格朗日插值法

//O(klogk) 求1-n的k次方和

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int mod=1e9+7;

const int N=1e6+10;

ll n,k,fac[N],y[N];

ll qp(ll a,ll b) {

ll s=1;

while(b) {

if(b&1)s=s\*a%mod;

a=a\*a%mod,b>>=1;

}

return s;

}

ll sv(int n,int k) {

fac[0]=fac[1]=1,y[1]=1;

for(int i=2;i<=k+2;i++) fac[i]=fac[i-1]\*i%mod;///预处理阶乘

for(int i=2;i<=k+2;i++) y[i]=(y[i-1]+qp(i,k))%mod;///预处理求出每一项的结果

if(n<=k+2)return y[n];

ll sum=1,sig;

for (ll i=n-k-2;i<=n-1;i++) sum=sum\*i%mod;

ll ans=0;

for(ll i=1;i<=k+2;i++) {///k+1项的多项式有k+2项

ll fz=sum\*qp(n-i,mod-2)%mod;///分子

ll fm=qp(fac[i-1]\*fac[k+2-i]%mod,mod-2);///分母

if((k+2-i)%2==0) sig=1;///正负号

else sig=-1;

ans=(ans+mod+sig\*y[i]\*fz%mod\*fm%mod+mod)%mod;

}

return ans;

}

int main() {

cin>>n>>k;

cout<<sv(n,k)<<endl;

return 0;

}

### 1.2母函数、排列组合、群论、线代

#### 1.2.1母函数、生成函数

a0+a1x+a2x^2+a3x^3+...

组合

a0+a1x/1!+a2x^2/2!+a3x^3/3!+...

排列

#### 1.2.2康托展开

//O(n^2)排列是第几个

#include<bits/stdc++.h>

using namespace std;

int n,m,f[11],a[11];

char ka[11];

vector<int>v;

inline int cantor(){

int ret=0,x;

for(int i=0; i<n; ++i) {

x=0;

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

if( (ka[i]-ka[j])>0 ) x++;

ret+=x\*f[n-i-1];

}

return ret;

}

inline void incantor(int k){

int x;

while( !v.empty() ) v.erase(v.end());

for(int i=1; i<=n; ++i) v.push\_back(i);

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

a[i]=v[(x=k/f[n-i])];

v.erase(v.begin()+x);

k%=f[n-i];

}

a[n]=v[0];

}

int main(){

cin>>n;

for(int i=0; i<n; ++i) cin>>ka[i];

f[1]=1;

for(int i=2; i<=10; ++i) f[i]=f[i-1]\*i;

incantor(cantor()-1);

for(int i=1; i<=n; ++i) printf("%d ",a[i]);

return 0;

}

#### 1.2.3排列组合

ll C[2010][2010];//1

void get\_C(int maxn){

C[0][0]=1;

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

C[i][0]=1;

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

C[i][j]=(C[i-1][j]+C[i-1][j-1])%mod;

}

}

ll fac[N],inv[N];//2

ll pow\_mod(ll a,ll n){

ll ret=1;

while(n){

if(n&1) ret=ret\*a%mod;

a=a\*a%mod;

n>>=1;

}

return ret;

}

void init(){

fac[0]=1;

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

fac[i]=fac[i-1]\*i%mod;

}

}

ll C(ll x, ll y){

return fac[x]\*pow\_mod(fac[y]\*fac[x-y]%mod,mod-2)%mod;

}

//卢卡斯定理

ll F[100010];

void init(ll p){

F[0] = 1;

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

F[i] = F[i-1]\*i % (1000000007);

}

ll inv(ll a,ll m){

if(a == 1)return 1;

return inv(m%a,m)\*(m-m/a)%m;

}

ll Lucas(ll n,ll m,ll p){

ll ans = 1;

while(n&&m){

ll a=n%p;

ll b=m%p;

if(a<b)return 0;

ans=ans\*F[a]%p\*inv(F[b]\*F[a-b]%p,p)%p;

n/=p;m/=p;

}

return ans;

}

#### 1.2.4群论

2\*2的格子放2(m)种颜色，旋转相同的算同一种，问有几种方案。

burnside引理

|G|为轨迹数（旋转方法数，0，90，180，270）

则方案=1/|G|（不动点总数）=1/|G|∑(c1,...cg)(c1为轨迹一的不动点数)

polya定理

m种颜色，n个元素构成的环

L=1/|G|∗∑m^c(i) c(i)为第i种轨迹 循环节(循环)个数

m种颜色，n个元素构成的环，旋转置换一共有n种，旋转i\*2pie/n可看作旋转2pie/n的i次幂，可拆分成gcd(n,i)个轮换，总方案数为1~n∑m^gcd(n,i)=d|n∑m^d\*φ(n/d)

//对1,2,...,n按p置换规则置换k次得到q，已知k和q，求p

#include<bits/stdc++.h>

using namespace std;

#define int long long

const int N=1e6+10;

int n,k;

int a[N],vis[N],b[N];

vector<int>v[N];

int ans[N];

int mul(int x,int y,int m){

int ans=0;

while (y){

if (y&1)ans=(ans+x)%m;

x=(x+x)%m;y>>=1;

}return ans;

}

int gcd(int x,int y){

return y?gcd(y,x%y):x;

}

int top[N],stk[N];

signed main() {

scanf("%lld%lld",&n,&k);

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

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

if (vis[i])continue;

int cnt=1;

vis[i]=1;

for (int j=a[i];j!=i;j=a[j])vis[j]=1,++cnt;

++b[cnt],v[cnt].push\_back(i);

}

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

if (!b[i])continue;

int g=gcd(i,k);

if (b[i]%g)return puts("-1"),0;

for (int o=0;o<b[i];o+=g){

int x=k/g%i;

for (int j=0;j<g;j++)top[j]=v[i][o+j];

for (int p=0;p<i;p++)

for (int j=0;j<g;j++)

stk[p\*x%i\*g+j]=top[j],top[j]=a[top[j]];

for (int j=0;j<i\*g;j++)ans[stk[j]]=stk[(j+1)%(i\*g)];

}

}

for (int i=1;i<=n;i++)printf("%lld ",ans[i]);

return 0;

}

#### 1.2.5高斯消元

##### 1.2.5.1 求多元一次方程解

const double eps=1e-7;

double a[N][N],b[N];

void Gauss(int n){//b为增广矩阵，a为系数矩阵，得出b为结果

int r;

for(int i=0;i<n;++i){

//数据稳定性优化

r=i;

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

if(fabs(a[j][i])>fabs(a[r][i]))

r=j;

if(fabs(a[r][i])<eps)return;//无解

if(r!=i){

for(int j=0;j<n;++j)swap(a[r][j],a[i][j]);

swap(b[r],b[i]);

}

//消元

for(int j=n;j>=i;j--)

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

if(j==n) b[k]-=a[k][i]/a[i][i]\*b[i];

else a[k][j]-=a[k][i]/a[i][i]\*a[i][j];

}

//回代

for(int i=n-1;i>=0;--i){

if(fabs(a[i][i])<eps)continue;

for(int j=i+1;j<n;++j)b[i]-=a[i][j]\*b[j];

b[i]/=a[i][i];//最后b有0则无穷解，行列式答案就是b[i]之积

}

}

##### 1.2.5.2生成树计数（Matrix-tree定理）

度数矩阵-边矩阵(邻接矩阵)得到的矩阵求行列式就是生成树个数

n个点，n\*n的矩阵，度数矩阵是其对角线元素为点度数

边矩阵，若n与m有边，则矩阵(n,m)与(m,n)为1

int gauss(int n){//求行列式

int res=1;

for(int i=1;i<=n-1;i++){//枚举主对角线上第i个元素

for(int j=i+1;j<=n-1;j++){//枚举剩下的行

while(a[j][i]){//辗转相除

int t=a[i][i]/a[j][i];

for(int k=i;k<=n-1;k++)//转为倒三角

a[i][k]=(a[i][k]-t\*a[j][k]+mod)%mod;

swap(a[i],a[j]);//交换i、j两行

res=-res;//取负

}

}

res=(res\*K[i][i])%mod;

}

return (res+mod)%mod;

}

#### 1.2.6快速幂

//普通快速幂、大整数取模快速乘、矩阵运算、群置换、定长路径统计

##### 1.2.6.1 快速幂和快速乘

int qpow(int x,int y){

int ans=1;

while (y){

if (y&1)ans=ans\*x%mod;

y>>=1;x=x\*x%mod;

}return ans;

}

int qmul(int x,int y){

int ans=0;

while (y){

if (y&1)ans=(ans+x)%mod;

x=x\*2%mod;y>>=1;

}return ans;

}

##### 1.2.6.2 矩阵快速幂

Const int M=2;

struct Mat{

int m[M][M];

Mat(){

for (int i=0;i<M;i++){

for (int j=0;j<M;j++){

m[i][j]=0;

}

}

}

};

Mat operator \*(Mat &a, Mat &b){

Mat ans;

for(int i=0;i<M;i++){

for(int j=0;j<M;j++){

for(int k=0;k<M;k++){

ans.m[i][j]=(ans.m[i][j]+a.m[i][k]\*b.m[k][j]%mod)%mod;

}

}

}

return ans;

}

Mat qpow(Mat t,int p){

Mat ans;

for(int i=0;i<M;i++){

ans.m[i][i]=1;

}

while(p) {

if (p&1)ans=ans\*t;

p>>=1;t=t\*t;

}

return ans;

}

##### 1.2.6.3 定长路径统计

我们把该图的邻接矩阵 M 取 k 次幂，那么Mi,j就表示从i到j长度为k的**简单路径**的数目。时间复杂度为O(n^3logk)

#### 1.2.7线性基

const int maxn=63;//或62

ll p[maxn+1],d[maxn+1];

ll cnt=0;

//普通线性基

void get\_lb(ll x){

dep(i,maxn,0){

if (!(x>>i))continue;

if (!p[i]){

p[i]=x;

break;

}

x^=p[i];

}

}

//求最大异或和

ll getsum(){

ll ans=0;

dep(i,maxn,0)

if((ans^p[i])>ans)

ans^=p[i];

return ans;

}

//求第k小的值

void rebuild(){

dep(i,maxn,0){

if (p[i]){//优化，可有可无

rep(j,i+1,maxn){

if ((p[j]>>i)&1) p[j]^=p[i];

}

}

}

rep(i,0,maxn) if (p[i]) d[cnt++]=p[i];

return;

}

ll kth(ll k){

if(cnt!=n)--k;

if (k>=(1LL<<cnt)) return -1;

ll ans=0;

rep(i,0,cnt){

if ((k>>i)&1){

ans^=d[i];

}

}

return ans;

}

### 1.3博弈论

### 1.3.1 Nim game

##### 1.3.1.1 Anti-nim game

必赢条件

1.all ai=1,且n=2k

或2.exsit ai>1,异或和>0

##### 1.3.1.2 nim-k game

有n堆石子，每次可从k堆石子中拿走任意数量的石子。

必赢条件

n的二进制1的个数%(k+1)!=0

#### 1.3.2 k倍动态递减法

有一个整数 S(S ≥ 2)，两个人想让它变成 0。首先，第一个人需要把 S 减掉一个正数

x(0 < x < S)。之后双方轮流把 S 减掉一个正整数，但都不能超过先前一回合对方减掉的数的

K(1 ≤ K ≤ 100000) 倍，减到 0 的一方获胜。问谁会获得胜利，若胜利还要求先手第一步至少减去多少。

const int N=750000;

int i,j;

ll n,k,a[N],b[N],ans;

int main(){

scanf("%lld%lld",&n,&k);

a[1]=b[1]=1;

for(i=2,j=0;;i++){

a[i]=b[i&1]+1;

if(a[i]>=n)break;

while(a[j+1]\*k<a[i])j++;

b[i]=a[i]+b[j];

}

while(a[i]>n)i−−;

if(a[i]==n)puts("lose");

else{

while(n){

while(a[i]>n)i−−;

n+=a[i];

ans=a[i];

}

printf("%lld",ans);

}

}

### 3.1插头dp

### 3.2斜率优化和四边形不等式

### 3.3背包

当物体数量很多，体积较小时，可以贪心到背包剩余体积小于某个时间复杂度内的值，剩下来的背包，错误率较小。

### 3.4数位dp

//0<=x<=a,0<=y<=b,|x-y|<=k,x^y<=w hdoj6899

#include<bits/stdc++.h>

using namespace std;

#define int long long

const int N=1e5+10;

int a,b,k,w;

int dp[35][3][3][2][2][2];

int dfs(int n,int s1,int s2,int fa,int fb,int fw){

s1=min(s1,1ll);s2=min(s2,1ll);//>=1，之后无法减去，一直>0

if(s1<-1||s2<-1)return 0;//<-1，之后无法加回来，一直<0

if(n<0)return s1>=0&&s2>=0;

if (~dp[n][s1+1][s2+1][fa][fb][fw])return dp[n][s1+1][s2+1][fa][fb][fw];

int upa=fa?(a>>n&1):1;

int upb=fb?(b>>n&1):1;

int upw=fw?(w>>n&1):1;

int ans=0;

int t=(k>>n&1);

for (int i=0;i<=upa;i++){

for (int j=0;j<=upb;j++){

if ((i^j)>upw)continue;

ans+=dfs(n-1,s1\*2+i-j+t,s2\*2+j-i+t,fa&&i==upa,fb&&j==upb,fw&&(i^j)==upw);

//|x-y|<=k=>x-y+k>=0,y-x+k>=0

}

}

return dp[n][s1+1][s2+1][fa][fb][fw]=ans;

}

signed main(){

int T;

cin>>T;

while (T--){

cin>>a>>b>>k>>w;

memset(dp,-1,sizeof dp);

int ans=dfs(30,0,0,1,1,1);

cout<<ans<<endl;

}

return 0;

}

### 3.4思维dp

//atcoder ARC107D

//将k个1分成n个1/2^i(i=0,1,2...),有多少种分法,n>=k

//思路：所有可能的情况都可以由多一个1和整体都/2组成

//计数问题可以找怎么组成这个情况来dp

#include<bits/stdc++.h>

using namespace std;

const int N=3e3+10;

#define int long long

const int mod=998244353;

int n,k;

int dp[N][N];

signed main(){

cin>>n>>k;

dp[1][1]=1;

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

for (int j=n;j>=1;j--){

dp[i][j]=(dp[i][j]+dp[i-1][j-1])%mod;

if (j\*2<=n)dp[i][j]=(dp[i][j]+dp[i][j\*2])%mod;

}

}

cout<<dp[n][k]<<endl;

return 0;

}

## 4 图论

#### 4.1最小生成树

##### 4.1.1朱刘算法

//O(VE)

#include<bits/stdc++.h>

using namespace std;

const int inf=2000000000;

const int N=1000+10;

typedef long long ll;

struct edge{

int u,v;

ll w;

}e[10005];

int m,n,pre[N],id[N],visit[N],xroot;

ll in[N],sum;

//eCnt为图中的边数

//n为图中的顶点数

//pre[i]为顶点i的前驱节点

//id[i]为缩环，形成新图的中间量

//in[i]为点i的最小入边

//visit[i]遍历图时记录顶点是否被访问过

ll directedMST(int root,int nv,int ne){

ll ans=0;

while(1){

//1.找最小入边

for(int i=0;i<nv;i++) in[i]=inf;

for(int i=0;i<ne;i++){

int u=e[i].u;

int v=e[i].v;

if(u!=v&&e[i].w<in[v]){

if(u==root)//此处标记与源点相连的最小边

xroot=i;

in[v]=e[i].w;

pre[v]=u;

}

}

for(int i=0;i<nv;i++)//判断图是否连通

if(i!=root&&in[i]==inf)return -1;//除了跟以外有点没有入边,则根无法到达它

//2.找环

int nodeCnt=0;//图中环的数目

memset(id,-1,sizeof(id));

memset(visit,-1,sizeof(visit));

in[root]=0;

for(int i=0;i<nv;i++){

ans+=in[i];

int v=i;

while(visit[v]!=i&&id[v]==-1&&v!=root){//每个点寻找其前序点，要么最终寻找至根部，要么找到一个环

visit[v]=i;

v=pre[v];

}

if(v!=root&&id[v]==-1){//缩点

for(int u=pre[v];u!=v;u=pre[u])

id[u]=nodeCnt;

id[v]=nodeCnt++;

}

}

if(nodeCnt==0) break;//如果无环，跳出循环

for(int i=0; i<nv; i++)

if(id[i]==-1)

id[i]=nodeCnt++;

//3.缩点，重新标记

for(int i=0;i<ne;i++){

int v=e[i].v;

e[i].u=id[e[i].u];

e[i].v=id[e[i].v];

if(e[i].u!=e[i].v)

e[i].w-=in[v];

}

nv=nodeCnt;

root=id[root];

}

return ans;

}

int main(){

int m;

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

sum=0;

for(int i=0;i<m;i++){

scanf("%d%d%lld",&e[i].u,&e[i].v,&e[i].w);

e[i].u++;e[i].v++;//都++之后，把0设为超级源点，联通各点

sum+=e[i].w;

if(e[i].u==e[i].v)e[i].w=inf;//消除自环

}

sum++;//此处必须++，因为需要权值比总权值大，因为这个w几次，，，

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

e[i].u=0;

e[i].v=i-m+1;

e[i].w=sum;

}

ll ans=directedMST(0,n+1,m+n);//根 点数 边数

if(ans==-1||ans-sum>=sum) printf("impossible\n");//ans-sum是除去虚根的最小树形图的最短路径，如果这个距离比所有的边权值和sum还大，说明还有另外的边由虚点发出，故说明此图不连通

else printf("%lld %d\n",ans-sum,xroot-m);

printf("\n");

}

return 0;

}

##### 4.1.2最小异或生成树

const int N=2e5+10;

const int inf=0x3f3f3f3f;

int n;

vector<pair<int,int> >v[N];

int l[N\*40],r[N\*40],nx[N\*40][2];

ll exor[N];

int root,tot;

void dfs(int x,int fa){

for (pair<int,int> i:v[x]){

if (i.first==fa)continue;

exor[i.first]=exor[x]^i.second;

dfs(i.first,x);

}

}

void insert(int &now,int id,int dep){//插入第id大的数，这个数的bit位为字典树的节点

if(!now)now=++tot;

l[now]=min(l[now],id),r[now]=max(r[now],id);

if(dep<0)return;

ll bit=(exor[id]>>dep)&1;

insert(nx[now][bit],id,dep-1);

}

ll query(int now,ll val,int dep){

if(dep<0)return 0;

ll bit=(val>>dep)&1;

if(nx[now][bit]) return query(nx[now][bit],val,dep-1);

else return query(nx[now][bit^1],val,dep-1)+(1<<dep);//异或为1，加上该深度权值，因为存在左右2数，必定能往下搜

}

ll solve(int now,int dep){

int ls=nx[now][0],rs=nx[now][1];

if (dep<0)return 0;

if (r[ls]&&r[rs]){//存在左右的2数

ll mi=inf;

for (int i=l[ls];i<=r[ls];i++)mi=min(mi,query(rs,exor[i],dep-1));//找出左右异或最小的值，且必须要加上这个值，因为没有其他的异或值比这个更小

return solve(ls,dep-1)+solve(rs,dep-1)+mi+(1<<dep);

}

if (r[ls])return solve(ls,dep-1);

if (r[rs])return solve(rs,dep-1);

return 0;

}

int main(){

int x,y,z;

scanf("%d",&n);

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

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

x++;y++;

v[x].push\_back({y,z});v[y].push\_back({x,z});

}

exor[0]=0;dfs(1,0);

sort(exor+1,exor+1+n);

memset(l,inf,sizeof l);

for (int i=1;i<=n;i++)insert(root,i,30);

cout<<solve(root,30)<<endl;

return 0;

}

##### 4.1.3斯坦纳树

//给定特定点，建立最小生成树，不需要包含全部点 O(n\*3^k)k为特殊点个数

//luoguR38373870

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

typedef pair<int,int> P;

const int N=5e5+10;

const int inf=0x3f3f3f3f;

int n,m,k;

int a[N],inq[N],ans[11][11],dp[101][1111];

int d[4][2]={{0,1},{0,-1},{1,0},{-1,0}};

pair<P,int>pre[101][1111];

queue<P>q;

bool check(P u){

return u.first>=0&&u.second>=0&&u.first<n&&u.second<m;

}

int num(P u){

return u.first\*m+u.second;

}

void spfa(int now){

while(!q.empty()){

P u=q.front();q.pop();

inq[num(u)]=0;

for(int i=0;i<4;i++){

P v={u.first+d[i][0],u.second+d[i][1]};

int nu=num(u),nv=num(v);

if(check(v)&&dp[nv][now]>dp[nu][now]+a[nv]){

dp[nv][now]=dp[nu][now]+a[nv];

if(!inq[nv])inq[nv]=1,q.push(v);

pre[nv][now]={u,now};

}

}

}

}

void dfs(P u,int now){

if(!pre[num(u)][now].second) return;

ans[u.first][u.second] = 1;

int nu=num(u);

if(pre[nu][now].first==u)dfs(u,now^pre[nu][now].second);

dfs(pre[nu][now].first,pre[nu][now].second);

}

int main(){

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

memset(dp,inf,sizeof(dp));

int cnt=0,K=0,rt=0;

for (int i=0;i<n;i++){

for (int j=0;j<m;j++){

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

if (!a[cnt])dp[cnt][1<<(K++)]=0,rt=cnt;

cnt++;

}

}

for (int now=1;now<(1<<K);now++){

for (int i=0;i<n\*m;i++){

for (int s=now&(now-1);s;s=now&(s-1)){

if (dp[i][now]>dp[i][s]+dp[i][now^s]-a[i]){

dp[i][now]=dp[i][s]+dp[i][now^s]-a[i];

pre[i][now]={{i/m,i%m},s};

}

}

if(dp[i][now]<inf)q.push({i/m,i%m}),inq[i]=1;

}

spfa(now);

}

printf("%d\n",dp[rt][(1<<K)-1]);

dfs({rt/m,rt%m},(1<<K)-1);

cnt=0;

for (int i=0;i<n;i++){

for (int j=0;j<m;j++){

if(!a[cnt++]) putchar('x');

else putchar(ans[i][j]?'o':'\_');

}

puts("");

}

return 0;

}

#### 4.2 Cayley公式和prufer序列

Cayley公式是一个完全图K\_n有n^(n-2)棵生成树，换句话说n个节点的带标号的无根树

有n^(n-2)个。

无根树转化为Prufer序列：找到编号最小的叶节点，删除这个节点，然后与这个叶节点相连

的点计入序列，直到这棵树只剩下两个节点、一条边

Prufer序列转化为无根树：设点集V={1,2,3,...,n}，每次取出Prufer序列中最前面的元素u，

在V中找到编号最小的没有在Prufer序列中出现的元素v，给u，v连边然后从序列中删除v，

最后在V中剩下两个节点，给它们连边。最终得到的就是无根树。

具体实现也可以用一个set，维护Prufer序列中没有出现的编号。复杂度O(nlogn)。

#### 4.3二分匹配

##### 4.3.1 二分匹配

最大匹配：二分图中边集的数目最大的那个匹配；

最小顶点覆盖：用最少的点，让每条边都至少和其中一个点关联；

最小边覆盖：用尽量少的不相交的边覆盖所有顶点；

最小链覆盖：用尽量少的不相交简单路径覆盖有向无环图(DAG)G的所有顶点；

最长反链：DAG的一个点集，任意两点都不能从一个走到另一个。

最大独立集：在Ｎ个点的图G中选出m个点，使这m个点两两之间没有边的点中，m的最大值。

二分图的最小点覆盖 = 二分图的最大匹配（证明见König定理）

二分图的最少边覆盖 = 点数 - 二分图的最大匹配

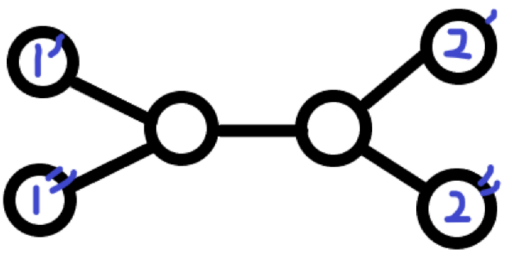
最小链覆盖 = 最长反链 = 点数 - 二分图的最大匹配（建图就是拆点i为i和i+n，按照二分图最大匹配的形式建图）

二分图的最大独立集 = 点数 - 二分图的最大匹配

无向图的最大团 = 无向图补图的最大独立集

##### 4.3.2 二分完全图匹配

//n个点，m条边，取其中一定的边满足每个点的度都是的d[i]，每个点的点数为度数的个数



#include<bits/stdc++.h>

#define ll long long

#define pb push\_back

using namespace std;

const int N=2333;

int n,m;

int cnt,match[N];//点数INIT

int fa[N];

vector<int>g[N];//INIT

int tp[N],pre[N];

queue<int>q;

int tim=0,tic[N];

void add(int u,int v){g[u].pb(v);g[v].pb(u);}

int ff(int x){if(x == fa[x]) return x; return fa[x]=ff(fa[x]);}

int lca(int x, int y){

tim++;x=ff(x);y=ff(y);

while(tic[x]!=tim){

tic[x]=tim;

x=ff(pre[match[x]]);

if(y)swap(x, y);

}

return x;

}

void go(int u,int v,int p){

while(ff(u)!=p){

pre[u]=v;v=match[u];

if(tp[v]==2) tp[v]=1,q.push(v);

if(ff(u)==u) fa[u]=p;

if(ff(v)==v) fa[v]=p;

u=pre[v];

}

}

bool aug(int s){

while(q.size())q.pop();

q.push(s);

for (int i=0;i<=cnt;i++)fa[i]=i,pre[i]=tp[i]=0;//

tp[s]=1;

while(q.size()){

int u=q.front();q.pop();

for(int v:g[u]){

if(tp[v]==2||ff(u)==ff(v)) continue;

if(!tp[v]){

tp[v]=2; pre[v]=u;

if(!match[v]){//找到增广点，一路返回修改

int now=v, tmp;

while(now){

tmp=match[pre[now]];

match[pre[now]]=now; match[now]=pre[now];

now=tmp;

}

return true;

}

tp[match[v]] = 1; q.push(match[v]);

}

else if(tp[v]==1){

int p=lca(u,v);

go(u,v,p); go(v,u,p);

}

}

}

return false;

}

//上面是模板部分

int d[N],sumd;

vector<int>to[N];

void init(){

sumd=0;

cnt=0;

for (int i=1;i<=n;i++)to[i].clear();

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

scanf("%d",&d[i]),sumd+=d[i];

for (int j=1;j<=d[i];j++) to[i].pb(++cnt),g[cnt].clear();

}

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

int u,v;

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

++cnt;g[cnt].clear();

for(int p:to[u])add(p,cnt);

++cnt;g[cnt].clear();add(cnt,cnt-1);

for(int p:to[v])add(p,cnt);

}

}

bool solve(){

int ans = 0;

for (int i=1;i<=cnt;i++) match[i]=0;

for (int i=1;i<=cnt;i++) ans+=(!match[i] && aug(i));

return ans\*2==cnt;

}

int main(){

while(scanf("%d%d",&n,&m)==2){

init();

if(solve())puts("Yes");

else puts("No");

}

}

#### 4.4最大团

int g[N][N],cnum[N],ans;

bool dfs(int step,int \*R,int Rlen){

int R2[N],R2len;

if(step > ans){ ans = step; return true;} //剪枝3

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

if(step + Rlen - i + 1 <= ans){ break;} //剪枝1

if(step + cnum[R[i]] <= ans){ break;} //剪枝2

R2len = 0;

for(int j = i + 1; j <= Rlen; ++j){

if(g[R[i]][R[j]]){

R2[++R2len] = R[j];

}

}

if(dfs(step + 1,R2,R2len))return true;

}

return false;

}

int maxclique(){

int R[N],Rlen = 0;

ans = 1; cnum[n] = 1;

for(int i = n - 1; i >= 1; --i){

Rlen = 0;

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

if(g[i][j])R[++Rlen] = j;

}

dfs(1,R,Rlen);

cnum[i] = ans;

}

return ans;

}

#### 4.5 Tarjan

##### 4.5.1强连通分量

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N=5e4+10;

int n,m,k;

int head[N];

int step,tot,gg;

int dfn[N],low[N];//dfn[x]标记x是在dfs中第几个遍历到的点

int color[N],cnt[N];

bool ff[N];

stack<int>s;

struct node{

int to,n;

}e[N<<1];

void add(int x,int y){

e[step].to=y;

e[step].n=head[x];

head[x]=step++;

}

void tarjan(int x){

dfn[x]=low[x]=++tot;//表示点i是第几个被遍历到的，把low[i]的值先赋值为dfn[i]的值

s.push(x);

ff[x]=1;

for(int i=head[x];i!=-1;i=e[i].n){

int v=e[i].to;

if(!dfn[v]){

tarjan(v);

low[x]=min(low[x],low[v]);

}

else if(ff[v])low[x]=min(low[x],dfn[v]);

}

int k;

if(low[x]==dfn[x]){

++gg;

do{

k=s.top();s.pop();

ff[k]=0;

color[k]=gg;cnt[gg]++;//将一个分量中的元素染成一色

}while(x!=k);

}

}

int main(){

cin>>n>>m;

int x,y;

step=0;

memset(head,-1,sizeof head);

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

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

add(x,y);

}

tot=0,gg=0;

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

if (!dfn[i])tarjan(i);

}

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

printf("%d ",cnt[i]);

}puts("");

return 0;

}

##### 4.5.2点双连通分量(割点)

//O(N+M)

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N=3e5+10;

int n,m,k;

int head[N];

int step,tot;

int dfn[N],low[N];

int ans[N];

struct node{

int to,n;

}e[N<<1];

void add(int x,int y){

e[step].to=y;

e[step].n=head[x];

head[x]=step++;

}

void tarjan(int x,int fa){

low[x]=dfn[x]=++tot;

for(int i=head[x];i!=-1;i=e[i].n){

int v=e[i].to;

if(!dfn[v]){

tarjan(v,x);

low[x]=min(low[v],low[x]);

if(low[v]>=dfn[x])ans[x]++;

//ans为x子树个数if (ans[x]>1||s!=x) x为割点，s为tarjan起始点if(dfn[v]<=dfn[t])代表该割点把s和t分开了

}

else if(v!=fa)low[x]=min(low[x],dfn[v]);

}

}

int main(){

cin>>n>>m;

int x,y;

step=0;

memset(head,-1,sizeof head);

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

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

add(x,y);add(y,x);

}

int num=0;

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

if (!dfn[i]){

num++;

tarjan(i,-1);

ans[i]--;

}

}

for (int i=1;i<=n;i++){//ans>0的为割点

cout<<ans[i]+num<<" ";//去除不同割点剩余连通块数目

}puts("");

return 0;

}

##### 4.5.3边双连通分量(割边)

void tarjan(int x,int fa){

low[x]=dfn[x]=++tot;

for(int i=head[x];i!=-1;i=e[i].n){

int v=e[i].to;

if(!dfn[v]){

tarjan(v,x);

low[x]=min(low[v],low[x]);

if(low[v]>dfn[x])ans++;//v-x为割的边

}

else if(v!=fa)low[x]=min(low[x],dfn[v]);

}

}

#### 4.6三元环

//方法一：tarjon强连通分量

//竞赛图为有向完全图，存在环则必存在三元环

//方法二：不存在环时，他的度数数组一定是0~n-1各出现一次

//三元环计数

#include<bits/stdc++.h>

#define ll long long

#define P pair<int,int>

using namespace std;

const int N = 1e5 + 10;

set<ll> g;

int deg[N];

vector<pair<int,int> > G[N];

int vi[N];

int X[N \* 2],Y[N \* 2],cnt[N],pos[N];

int main(){

int n,m,u,v,Sz;

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

Sz=sqrt(m);

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

vi[i] = deg[i] = pos[i] = 0;

G[i].clear();

}

g.clear();

int tot = 0;

for(int i = 0;i < m;i++){

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

u = X[i],v = Y[i];

deg[u]++,deg[v]++;

}

for(int i = 0;i < m;i++){

cnt[i] = 0;

if(deg[X[i]] < deg[Y[i]]) G[X[i]].push\_back(make\_pair(Y[i],i));

else if(deg[Y[i]] < deg[X[i]]) G[Y[i]].push\_back(P(X[i],i));

else{

if(X[i] < Y[i]) G[X[i]].push\_back(P(Y[i],i));

else G[Y[i]].push\_back(P(X[i],i));

}

}

ll ans = 0;

for(int i = 0;i < m;i++){

u = X[i],v = Y[i];

for(auto vp:G[u]) pos[vp.first] = vp.second,vi[vp.first] = i + 1;

for(auto vp:G[v]){

int vv = vp.first;

if(vi[vv] == i + 1){

cnt[i]++;

cnt[pos[vv]]++;

cnt[vp.second]++;

}

}

}

for(int i=0;i<m;i++)ans+=1ll\*cnt[i]\*(cnt[i]-1)/2;

printf("%lld\n",ans);

}

return 0;

}

#### 4.7树链剖分

#include<bits/stdc++.h>

using namespace std;

#define ll long long

const int N=2e5+10;

const int inf=0x3f3f3f3f;

const int mod=1e9+7;

int n,m,k,p;

int a[N];

int si[N],dep[N],fa[N],son[N];

int id[N],top[N],anew[N],cnt=0;//si:子树大小,dep:深度,fa:父亲,son:重儿子,top:重链的父亲

int head[N],step=0;

int res=0;

struct edge{

int to,n;

}e[N<<2];

void add(int x,int y){

e[step].to=y;

e[step].n=head[x];

head[x]=step++;

}

void dfs1(int x,int ff,int depth){//处理dep,si,fa,son

dep[x]=depth;

si[x]=1;

fa[x]=ff;

int mx=-1;

for (int i=head[x];~i;i=e[i].n){

int v=e[i].to;

if (v==ff)continue;

dfs1(v,x,depth+1);

si[x]+=si[v];

if (si[v]>mx)mx=si[v],son[x]=v;

}

}

void dfs2(int x,int topf){//x当前节点，topf当前链的最顶端的节点

id[x]=++cnt;//标记每个点的新编号

anew[cnt]=a[x];//把每个点的初始值赋到新编号上来

top[x]=topf;

if(!son[x])return;//如果没有儿子则返回

dfs2(son[x],topf);//按先处理重儿子，再处理轻儿子的顺序递归处理

for(int i=head[x];~i;i=e[i].n){

int v=e[i].to;

if(v==fa[x]||v==son[x])continue;

dfs2(v,v);

}

}

struct node{

int val,lazy;

int l,r;

}tr[N<<2];

void pushup(int rt){

tr[rt].val=(tr[rt<<1].val+tr[rt<<1|1].val)%p;//

}

void putdown(int rt){

if(tr[rt].lazy){

tr[rt<<1].lazy+=tr[rt].lazy;

tr[rt<<1].val+=(tr[rt<<1].r-tr[rt<<1].l+1)\*tr[rt].lazy;

tr[rt<<1|1].lazy+=tr[rt].lazy;

tr[rt<<1|1].val+=(tr[rt<<1|1].r-tr[rt<<1|1].l+1)\*tr[rt].lazy;

tr[rt<<1].val%=p;tr[rt<<1|1].val%=p;//

tr[rt].lazy=0;

}

}

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

tr[rt].l=l;

tr[rt].r=r;

tr[rt].lazy=0;

if (l==r){

tr[rt].val=anew[l]%p;//

return;

}

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

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

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

pushup(rt);

}

void update(int l,int r,int add,int rt){

if (l<=tr[rt].l&&r>=tr[rt].r){

tr[rt].lazy+=add;

tr[rt].val+=(tr[rt].r-tr[rt].l+1)\*add;

return;

}

putdown(rt);

if(l<=tr[rt<<1].r)

update(l,r,add,rt<<1);

if(r>=tr[rt<<1|1].l)

update(l,r,add,rt<<1|1);

pushup(rt);

}

int query(int l,int r,int rt){

if(l==tr[rt].l&&r==tr[rt].r)

return tr[rt].val%p;//

putdown(rt);

if(l>=tr[rt<<1|1].l)

return query(l,r,rt<<1|1);

else if(r<=tr[rt<<1].r)

return query(l,r,rt<<1);

else

return (query(l,tr[rt<<1].r,rt<<1)+query(tr[rt<<1|1].l,r,rt<<1|1))%p;//

}

void updateRange(int x,int y,int k){

while(top[x]!=top[y]){

if(dep[top[x]]<dep[top[y]])swap(x,y);

update(id[top[x]],id[x],k,1);

x=fa[top[x]];

}

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

update(id[x],id[y],k,1);

}

void updateSon(int x,int k){

update(id[x],id[x]+si[x]-1,k,1);

}

int qRange(int x,int y){

int ans=0;

while(top[x]!=top[y]){

if(dep[top[x]]<dep[top[y]])swap(x,y);

res=query(id[top[x]],id[x],1);

ans=(ans+res)%p;//

x=fa[top[x]];

}

//在同一链上时

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

res=query(id[x],id[y],1);

ans=(ans+res)%p;//

return ans;

}

int qSon(int x){

res=query(id[x],id[x]+si[x]-1,1);

return res;

}

int main(){

int x,y;

step=cnt=0;

memset(head,-1,sizeof head);

memset(son,0,sizeof son);

cin>>n>>m>>k>>p;//点数、询问数、根节点、取模数

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

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

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

add(x,y);add(y,x);

}

dfs1(k,0,1);

dfs2(k,k);

build(1,n,1);

while(m--){

int c,x,y,z;

scanf("%d",&c);

if(c==1){//x到y的最短路径+z

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

updateRange(x,y,z%p);//

}

else if(c==2){//询问x到y路径的权值和

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

printf("%d\n",qRange(x,y));

}

else if(c==3){//x的子树全+y

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

updateSon(x,y);

}

else{//询问x的子树权值和

scanf("%d",&x);

printf("%d\n",qSon(x));

}

}

return 0;

}

#### 4.8 LCA

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int inf=0x3f3f3f3f;

const int N=5e5+10;

const int mod=998244353;

int n,m,k;

int step,head[N];

int f[N][21],dep[N];

struct node{

int to,n;

}e[N<<1];

void add(int x,int y){

e[step].to=y;

e[step].n=head[x];

head[x]=step++;

}

inline void dfs(int u,int fa){

dep[u]=dep[fa]+1;

for(int i=0;i<=19;i++)//2^0 ~ 2^19

f[u][i+1]=f[f[u][i]][i];//递推公式，上面讲过了。

for(int i=head[u];i!=-1;i=e[i].n){

int v=e[i].to;

if(v==fa) continue;

f[v][0]=u;

dfs(v,u);

}

}

inline int LCA(int x,int y){

if(dep[x]<dep[y]) swap(x,y);//让x深度较大

//用“暴力”的思想：先让x,y跳到同一深度，然后一起往上跳

for(int i=20;i>=0;i--){//倒着for，x能多跳尽量多跳 ，才能优化时间

if(dep[f[x][i]]>=dep[y]) x=f[x][i];//先跳到同一层

if(x==y) return y;

}

for(int i=20;i>=0;i--){//此时x,y已跳到同一层

if(f[x][i]!=f[y][i]){//如果 f[x][i]和f[y][i]不同才跳

x=f[x][i];

y=f[y][i];

}

}

return f[x][0];//跳完上述步骤后，两点离LCA仅一步之遥，让x(或y)再向上跳一步就是LCA。

}

int main(){

int s,t;

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

cin>>n>>m>>s;

int x,y;

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

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

add(x,y);add(y,x);

}

dfs(s,-1);

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

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

printf("%d\n",LCA(x,y));

}

return 0;

}

#### 4.9网络流

最大流、最小割

struct Dinic{

int dep[N],head[N],cur[N];

int step,n;

struct node{

int to,n;

ll cap;

}e[N<<1];

void init(int x){

n=x;step=0;

for (int i=0;i<=n;i++)head[i]=-1;

}

void add(int x,int y,int v){

e[step].to=y;

e[step].cap=v;

e[step].n=head[x];

head[x]=step++;

e[step].to=x;

e[step].cap=0;

e[step].n=head[y];

head[y]=step++;

}

bool bfs(int s,int t){

for (int i=0;i<=n;i++)dep[i]=0;

queue<int>q;

q.push(s);

dep[s]=1;

cur[s]=head[s];

while(!q.empty()){

s=q.front();

q.pop();

for(int i=head[s];i!=-1;i=e[i].n){

int y=e[i].to;

if(e[i].cap>0&&dep[y]==0){

dep[y]=dep[s]+1;

cur[y]=head[y];

if(y==t) return true;

q.push(y);

}

}

}

return false;

}

ll dfs(int s,ll flow,int t){

if(s==t||flow<=0) return flow;

ll rest=flow;

for(int i=cur[s];i!=-1;i=e[i].n){

cur[s]=i;

int y=e[i].to;

if(e[i].cap>0&&dep[y]==dep[s]+1){

ll tmp=dfs(y,min(rest,e[i].cap),t);

if(tmp<=0) dep[y]=0;

rest-=tmp;

e[i].cap-=tmp;

e[i^1].cap+=tmp;

if(rest<=0) break;

}

}

return flow-rest;

}

ll solve(int s,int t){

ll ans=0;

while(bfs(s,t))

ans+=dfs(s,inf,t);

return ans;

}

}dinic;

#### 4.10费用流

struct zkw{

int Cost=0,Flow=0;

int vis[N],dis[N],head[N];

int step,n;

struct node{

int to,cap,cost,n;

}e[N<<1];

void add(int x,int y,int v,int c){

e[step].to=y;

e[step].cap=v;

e[step].cost=c;

e[step].n=head[x];

head[x]=step++;

e[step].to=x;

e[step].cap=0;

e[step].cost=-c;

e[step].n=head[y];

head[y]=step++;

}

bool spfa(int s,int t){

for(int i=0;i<=n;i++) vis[i]=0,dis[i]=inf;//要改

deque<int>q;

q.push\_back(t);

dis[t]=0;

vis[t]=1;

while(!q.empty()){

int u=q.front();

q.pop\_front();

for(int i=head[u];~i;i=e[i].n){

int v=e[i].to;

if(e[i^1].cap&&dis[v]>dis[u]+e[i^1].cost){

dis[v]=dis[u]+e[i^1].cost;

if(!vis[v]){

vis[v]=1;

if(!q.empty()&&dis[v]<dis[q.front()])//SLF优化

q.push\_front(v);

else q.push\_back(v);

}

}

}

vis[u]=0;

}

return dis[s]<inf;

}

int dfs(int s,int t,int flow){

vis[s]=1;

if(s==t||flow<=0) return flow;

int res,used=0;

for(int i=head[s];~i;i=e[i].n){

int v=e[i].to;

if(!vis[v]&&e[i].cap&&dis[s]-dis[v]==e[i].cost){

res=dfs(v,t,min(e[i].cap,flow-used));

if(res) {

e[i].cap-=res;

e[i^1].cap+=res;

Cost+=res\*e[i].cost;

used+=res;

}

if(used==flow)break;

}

}

return used;

}

void solve(int s,int t){

while(spfa(s,t)){

vis[t]=1;

while(vis[t]){

for(int i=0;i<=n;i++) vis[i]=0;//要改

Flow+=dfs(s,t,inf);

}

}

}

void init(int n){00

step=0;

Flow=Cost=0;

memset(head,-1,sizeof head);

this->n=n;

}

}zkw;

#### 4.11最短路

int n,m;

int c[N];

vector<pair<int,int>>v[N];

void dijkstra(int x){

priority\_queue< pair<int,int> , vector< pair<int,int>> , greater< pair<int,int>> > q;

fill(c,c+n+1,inf);

c[x]=0;

q.push({0,x});

while(!q.empty()){

pair<int,int> r=q.top();q.pop();

int u=r.second;

if (c[u]<r.first)continue;

for (auto i:v[u]){

int to=i.first,d=i.second;

if(c[u]+d<c[to]){

c[to]=c[u]+d;

q.push({c[to],to});

}

}

}

}

for(int k=1;k<=n;k++) //中转点

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

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

mp[i][j]=min(mp[i][j],mp[i][k]+mp[k][j]);

//floyed(传递闭包)O(n^3/32)

bitset<N>a[N];

for (int i=0;i<n;i++){

for (int j=0;j<n;j++){

if(a[j][i])a[j]|=a[i];

}

}

## 5 数据结构

#### 5.0.dsu on tree

#include<bits/stdc++.h>

using namespace std;

const int N=1e5+10;

int n,m,k;

int a[N],b[N];

int sz[N],son[N];

int ans[N];

vector<int>v[N];

void dfs1(int x,int fa){

sz[x]=1;

for (int i:v[x]){

if (i==fa)continue;

dfs1(i,x);

sz[x]+=sz[i];

if (sz[son[x]]<sz[i])son[x]=i;

}

}

int num[N],sum,nowson;

void cal(int x,int fa,int val){

if (num[a[x]]==k)sum--;

num[a[x]]+=val;

if (num[a[x]]==k)sum++;

for (int i:v[x]){

if (i==fa||i==nowson)continue;

cal(i,x,val);

}

}

void dfs2(int x,int fa,int is){

for(int i:v[x]){

if (i==fa||i==son[x])continue;

dfs2(i,x,0);

}

if(son[x])dfs2(son[x],x,1),nowson=son[x];

cal(x,fa,1);nowson=0;

ans[x]=sum;

if (!is)cal(x,fa,-1);

}

signed main(){

int T,q;

cin>>T;

for (int cas=1;cas<=T;cas++){

if (cas!=1)puts("");

cin>>n>>k;

int cnt=0;

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

scanf("%d",&a[i]);b[++cnt]=a[i];

}

sort(b+1,b+1+cnt);

int t=unique(b+1,b+1+cnt)-b;

for (int i=1;i<=n;i++)a[i]=lower\_bound(b+1,b+1+t,a[i])-b;

int x,y;

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

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

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

}

dfs1(1,0);

dfs2(1,0,1);

cin>>q;

printf("Case #%d:\n",cas);

while (q--){

scanf("%d",&x);

printf("%d\n",ans[x]);

}

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

v[i].clear();

son[i]=ans[i]=num[i]=0;

sum=0;nowson=0;

}

}

return 0;

}

#### 5.1树状数组

int c[N];

int lowbit(int x){

return x&(-x);

}

void add(int i,int k){

while(i <= n){

c[i]+=k;i+=lowbit(i);

}

}

int getsum(int i){

int res=0;

while(i>0){

res+=c[i];i-=lowbit(i);

}

return res;

}

#### 5.2线段树

##### 5.2.1区间和、区间加

struct SegmentTree{//

struct node{

int sum,lz;

int l,r;

}tr[N<<2];

void pushup(int rt){

tr[rt].sum=tr[rt<<1].sum+tr[rt<<1|1].sum;

}

void putdown(int rt){

if (tr[rt].lz){

tr[rt<<1].lz+=tr[rt].lz;

tr[rt<<1|1].lz+=tr[rt].lz;

tr[rt<<1].sum+=(tr[rt<<1].r-tr[rt<<1].l+1)\*tr[rt].lz;

tr[rt<<1|1].sum+=(tr[rt<<1|1].r-tr[rt<<1|1].l+1)\*tr[rt].lz;

tr[rt].lz=0;

}

}

void build(int l,int r,int rt=1){

tr[rt].l=l;tr[rt].r=r;

tr[rt].lz=0;

if (l==r){

tr[rt].sum=a[l];return;

}

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

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

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

pushup(rt);

}

void update(int l,int r,int add,int rt=1){

if (tr[rt].l>=l&&tr[rt].r<=r){

tr[rt].lz+=add;

tr[rt].sum+=(tr[rt].r-tr[rt].l+1)\*add;

return;

}

putdown(rt);

if (tr[rt<<1].r>=l)update(l,r,add,rt<<1);

if (tr[rt<<1|1].l<=r)update(l,r,add,rt<<1|1);

pushup(rt);

}

int query(int l,int r,int rt=1){

if (tr[rt].l==l&&tr[rt].r==r){

return tr[rt].sum;

}

putdown(rt);

if (tr[rt<<1].r>=r)return query(l,r,rt<<1);

if (tr[rt<<1|1].l<=l)return query(l,r,rt<<1|1);

return query(l,tr[rt<<1].r,rt<<1)+query(tr[rt<<1|1].l,r,rt<<1|1);

}

}tr;

##### 5.2.2 区间max，区间覆盖

struct SegmentTree{

struct node{

int mx,lz;

int l,r;

}tr[N<<2];

void pushup(int rt){

tr[rt].mx=max(tr[rt<<1].mx,tr[rt<<1|1].mx);

}

void putdown(int rt){

if (tr[rt].lz){

tr[rt<<1].lz=tr[rt<<1|1].lz=tr[rt].lz;

tr[rt<<1].mx=tr[rt<<1|1].mx=tr[rt].lz;

tr[rt].lz=0;

}

}

void build(int l,int r,int rt=1){

tr[rt].l=l;tr[rt].r=r;

tr[rt].lz=0;

if (l==r){

tr[rt].mx=a[l];return;

}

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

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

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

pushup(rt);

}

void update(int l,int r,int add,int rt=1){

if (tr[rt].l>=l&&tr[rt].r<=r){

tr[rt].lz=tr[rt].mx=add;

return;

}

putdown(rt);

if (tr[rt<<1].r>=l)update(l,r,add,rt<<1);

if (tr[rt<<1|1].l<=r)update(l,r,add,rt<<1|1);

pushup(rt);

}

int query(int l,int r,int rt=1){

if (tr[rt].l==l&&tr[rt].r==r)return tr[rt].mx;

putdown(rt);

if (tr[rt<<1].r>=r)return query(l,r,rt<<1);

if (tr[rt<<1|1].l<=l)return query(l,r,rt<<1|1);

return max(query(l,tr[rt<<1].r,rt<<1),query(tr[rt<<1|1].l,r,rt<<1|1));

}

}tr;

##### 5.2.3动态开点线段树

//可以处理树的合并，森林树上第k大

//求树上每个点对子树的逆序对

//要改的话，改pushup和l==r时的情况(update和merge)

#include<bits/stdc++.h>

using namespace std;

const int N=1e5+10;

struct Seg{

int cnt=0;

struct tree{

int l,r,val;

}tr[N\*31];//现在数据为1e9,也可以对数据进行离散化，只用\*21

void init(){

cnt=0;

}

void pushup(int rt){

tr[rt].val=tr[tr[rt].l].val+tr[tr[rt].r].val;

}

void update(int &rt,int l,int r,int x,int val){

if (!rt){

rt=++cnt;tr[rt].l=tr[rt].r=tr[rt].val=0;

}

if (l==r){

tr[rt].val+=val;return;

}

int mid=l+r>>1;

if (x<=mid)update(tr[rt].l,l,mid,x,val);

else update(tr[rt].r,mid+1,r,x,val);

pushup(rt);

}

int query(int rt,int l,int r,int L,int R){

if(!rt)return 0;

if(L<=l&&R>=r)return tr[rt].val;

int mid=l+r>>1;

if (L>mid)return query(tr[rt].r,mid+1,r,L,R);

if (R<=mid)return query(tr[rt].l,l,mid,L,R);

return query(tr[rt].r,mid+1,r,L,R)+query(tr[rt].l,l,mid,L,R);

}

int merge(int u,int v){

if(!u)return v;

if(!v)return u;

tr[u].val=tr[u].val+tr[v].val;

tr[u].l=merge(tr[u].l,tr[v].l);

tr[u].r=merge(tr[u].r,tr[v].r);

return u;

}

}tr;

int root[N];

vector<int>v[N];

int ans[N],p[N];

void dfs(int x){

for (int i:v[x]){

dfs(i);

tr.merge(root[x],root[i]);

}

ans[x]=tr.query(root[x],1,1e9,p[x]+1,1e9);

}

signed main(){

int n;

cin>>n;

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

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

tr.update(root[i],1,1e9,p[i],1);

}

int x;

for (int i=2;i<=n;i++){

scanf("%d",&x);

v[x].push\_back(i);

}

dfs(1);

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

printf("%d\n",ans[i]);

}

return 0;

}

//CF600E

#include<bits/stdc++.h>

using namespace std;

#define int long long

const int N=1e5+10;

int n,m;

struct Seg{

int cnt=0;

struct tree{

int l,r,val,ans;

}tr[N\*20];//也可以对数据进行离散化，只用\*21

void init(){

cnt=0;

}

void pushup(int rt){

if(tr[tr[rt].l].val==tr[tr[rt].r].val){

tr[rt].val=tr[tr[rt].l].val;

tr[rt].ans=tr[tr[rt].l].ans+tr[tr[rt].r].ans;

}

if(tr[tr[rt].l].val<tr[tr[rt].r].val){

tr[rt].val=tr[tr[rt].r].val;

tr[rt].ans=tr[tr[rt].r].ans;

}

if(tr[tr[rt].l].val>tr[tr[rt].r].val){

tr[rt].val=tr[tr[rt].l].val;

tr[rt].ans=tr[tr[rt].l].ans;

}

}

void update(int &rt,int l,int r,int x,int val){

if (!rt){

rt=++cnt;tr[rt].l=tr[rt].r=tr[rt].val=0;

}

if (l==r){

tr[rt].val+=val;tr[rt].ans=l;

return;

}

int mid=l+r>>1;

if (x<=mid)update(tr[rt].l,l,mid,x,val);

else update(tr[rt].r,mid+1,r,x,val);

pushup(rt);

}

int merge(int u,int v,int l,int r){

if(!u)return v;

if(!v)return u;

if (l==r){

tr[u].val+=tr[v].val;tr[u].ans=l;

return u;

}

int mid=l+r>>1;

tr[u].l=merge(tr[u].l,tr[v].l,l,mid);

tr[u].r=merge(tr[u].r,tr[v].r,mid+1,r);

pushup(u);

return u;

}

}tr;

int root[N];

vector<int>v[N];

int ans[N],p[N];

void dfs(int x,int fa){

for (int i:v[x]){

if (i==fa)continue;

dfs(i,x);

tr.merge(root[x],root[i],1,n);

}

ans[x]=tr.tr[root[x]].ans;

}

signed main(){

cin>>n;

m=1e5+10;

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

scanf("%lld",&p[i]);

tr.update(root[i],1,n,p[i],1);

}

int x,y;

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

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

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

}

dfs(1,0);

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

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

}

return 0;

}

##### 5.2.4 扫描线

#include<bits/stdc++.h>

using namespace std;

#define int long long

const int N=4e5+10;

int n;

int b[N];

int cnt1=0,cnt2=0;

int x\_1[N],x\_2[N],y\_1[N],y\_2[N];

struct node{

int l,r,h,op;

}a[N];

bool cmp(node a,node b){

return a.h<b.h;

}

struct Tree{

struct Tr{

int l,r,val,len;

}tr[N<<2];

void pushup(int rt) {

if(tr[rt].val)tr[rt].len=b[tr[rt].r+1]-b[tr[rt].l];

else tr[rt].len=tr[rt<<1].len+tr[rt<<1|1].len;

}

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

tr[rt].l=l;tr[rt].r=r;tr[rt].val=0;

if (l==r){

tr[rt].val=0;tr[rt].len=0;

return;

}

int mid=l+r>>1;

build(l,mid,rt<<1);build(mid+1,r,rt<<1|1);

pushup(rt);

}

void update(int l,int r,int val,int rt){

if (tr[rt].l>=l&&tr[rt].r<=r){

tr[rt].val+=val;

pushup(rt);

return;

}

if (l<=tr[rt<<1].r)update(l,r,val,rt<<1);

if (r>=tr[rt<<1|1].l)update(l,r,val,rt<<1|1);

pushup(rt);

}

}tr;

map<int,int>id;

signed main(){

cin>>n;

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

scanf("%lld%lld%lld%lld",&x\_1[i],&y\_1[i],&x\_2[i],&y\_2[i]);

b[++cnt1]=x\_1[i];b[++cnt1]=x\_2[i];

a[++cnt2].l=x\_1[i];a[cnt2].r=x\_2[i];a[cnt2].h=y\_1[i];a[cnt2].op=1;

a[++cnt2].l=x\_1[i];a[cnt2].r=x\_2[i];a[cnt2].h=y\_2[i];a[cnt2].op=-1;

}

sort(a+1,a+1+cnt2,cmp);

sort(b+1,b+1+cnt1);

int t=unique(b+1,b+1+cnt1)-b-1;

for (int i=1;i<=t;i++)id[b[i]]=i;

tr.build(1,t-1,1);

int ans=0;

for (int i=1;i<cnt2;i++){

tr.update(id[a[i].l],id[a[i].r]-1,a[i].op,1);

ans+=(a[i+1].h-a[i].h)\*tr.tr[1].len;

}

cout<<ans<<endl;

return 0;

}

#### 5.3主席树

##### 5.3.1可持久化线段树

#include<bits/stdc++.h>

using namespace std;

const int N=1000010;

int n,q,m;

int a[N];

struct ChairMan{

int T[N\*20],rt[N];

int L[N\*20],R[N\*20];

int cnt;

int build(int l,int r){

int rt=++cnt;

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

if (l==r){

T[rt]=a[l];

return rt;

}

L[rt]=build(l,mid);

R[rt]=build(mid+1,r);

return rt;

}

int update(int pre,int l,int r,int x,int add){

int rt=++cnt;

if (l==r){

T[rt]=add;

return rt;

}

L[rt]=L[pre];R[rt]=R[pre];

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

if (x<=mid)L[rt]=update(L[pre],l,mid,x,add);

else R[rt]=update(R[pre],mid+1,r,x,add);

return rt;

}

int query(int pre,int l,int r,int x){

if (l>=r)return T[pre];

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

if (x<=mid)return query(L[pre],l,mid,x);

else return query(R[pre],mid+1,r,x);

}

}hjt;

signed main(){

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

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

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

}

hjt.build(1,n);

hjt.rt[0]=1;

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

int v,cd,x,y;

scanf("%d%d%d",&v,&cd,&x);

if(cd==1){//修改v版本将a[x]改为y

scanf("%d",&y);

hjt.rt[i]=hjt.update(hjt.rt[v],1,n,x,y);

}

if(cd==2){//查询v版本的a[x]

hjt.rt[i]=hjt.rt[v];

cout<<hjt.query(hjt.rt[v],1,n,x)<<endl;

}

}

return 0;

}

##### 5.3.2静态第k大

//每棵树维护1-数的最大值，T维护数的下标

#include<bits/stdc++.h>

using namespace std;

const int N=200010;

int n,q,m;

int a[N],b[N],T[N];

struct Chairman{

int cnt=0;

int sum[N<<5],L[N<<5],R[N<<5];

void init(){

cnt=0;

}

void pushup(int rt){

sum[rt]=sum[L[rt]]+sum[R[rt]];

}

int build(int l,int r){

int rt=++cnt;

if (l==r){

sum[rt]=0;return rt;

}

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

L[rt]=build(l,mid);R[rt]=build(mid+1,r);

pushup(rt);

return rt;

}

int update(int pre,int l,int r,int x){

int rt=++cnt;

L[rt]=L[pre];R[rt]=R[pre];

if (l==r){

sum[rt]=sum[pre]+1;

return rt;

}

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

if (x<=mid)L[rt]=update(L[pre],l,mid,x);

else R[rt]=update(R[pre],mid+1,r,x);

pushup(rt);

return rt;

}

int query(int ll,int rr,int l,int r,int k){

if (l>=r)return l;

int x=sum[L[rr]]-sum[L[ll]];

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

if (x>=k)return query(L[ll],L[rr],l,mid,k);

else return query(R[ll],R[rr],mid+1,r,k-x);

}

}hjt;

signed main(){

int tt=1;

//cin>>tt;

while (tt--){

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

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

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

b[i]=a[i];

}

sort(b+1,b+1+n);

m=unique(b+1,b+1+n)-b-1;

hjt.init();

T[0]=hjt.build(1,m);

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

int t=lower\_bound(b+1,b+1+m,a[i])-b;

T[i]=hjt.update(T[i-1],1,m,t);

}

while (q--){

int x,y,z;

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

int t=hjt.query(T[x-1],T[y],1,m,z);

printf("%d\n",b[t]);

}

}

return 0;

}

#### 5.4左偏树

//多个优先队列合并

#include<bits/stdc++.h>

using namespace std;

const int N=1e5+10;

int n,m;

struct leftist\_tree{

int dis,val,lch,rch,f;

}lt[N];

inline int find(int x){

if(lt[x].f==x)return x;

return lt[x].f=find(lt[x].f);

}

inline int merge(int x,int y){//输入两个最大堆的根，返回合并后的根

if(!x||!y)return x+y;

if(lt[x].val<lt[y].val)swap(x,y);//<,>号修改可以改变最小堆还是最大堆，也可以重定义函数

if(x>y&&lt[x].val==lt[y].val)swap(x,y);//下标小的为x，可不要

int ls=lt[x].lch,rs=lt[x].rch;

rs=merge(rs,y);

lt[rs].f=x;

if(lt[ls].dis<lt[rs].dis)swap(ls,rs);//不左偏了swap回来

lt[x].lch=ls,lt[x].rch=rs;

if(lt[x].rch==0)lt[x].dis=0;

else lt[x].dis=lt[rs].dis+1;

return x;

}

int pop(int x){//返回删除该点后的根

int xx=find(x);

int ls=lt[xx].lch,rs=lt[xx].rch;

lt[ls].f=ls,lt[rs].f=rs;

lt[xx].lch=lt[xx].rch=lt[xx].dis=0;

return merge(ls,rs);

}

int main(){

int x,y;

while (scanf("%d",&n)==1){

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

scanf("%d",&lt[i].val);

lt[i].f=i;

lt[i].lch=lt[i].rch=lt[i].dis=0;

}

scanf("%d",&m);

while (m--){

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

int xx=find(x),yy=find(y);

if (xx==yy)puts("-1");

else{

lt[xx].val>>=1;lt[yy].val>>=1;

int l=pop(xx),r=pop(yy);

l=merge(l,xx),r=merge(r,yy);

printf("%d\n",lt[merge(l,r)].val);

}

}

}

return 0;

}

#### 5.5平衡树

##### 5.5.1Treap

//1.插入x

//2.删除一个x

//3.查询x的排名

//4.查询排名为x的数

//5.求小于x，最大的数

//6.求大于x，最小的数

#include<bits/stdc++.h>

using namespace std;

const int N=1e5+10;

const int inf=1e9;

int ch[N][2];//[i][0]代表i左儿子，[i][1]代表i右儿子

int val[N],dat[N];

int size[N],cnt[N];

int tot,root;

void clear(){

for (int i=0;i<=tot;i++)ch[i][0]=ch[i][1]=0;

fill(size,size+tot+1,0);

root=tot=0;

}

int New(int x){

val[++tot]=x;size[tot]=1;

dat[tot]=rand();//随机优先级

cnt[tot]=1;//副本数为1

return tot;

}

void pushup(int id){

size[id]=size[ch[id][0]]+size[ch[id][1]]+cnt[id];

}

void build(){

root=New(-inf),ch[root][1]=New(inf);//先加入正无穷和负无穷，便于之后操作(貌似不加也行)

pushup(root);

}

void Rotate(int &id,int d){//id是引用传递，0为左旋，1为右旋

int temp=ch[id][d^1];

ch[id][d^1]=ch[temp][d];

ch[temp][d]=id;

id=temp;

pushup(ch[id][d]),pushup(id);

}

void insert(int &id,int x){//id依然是引用，在新建节点时可以体现

if(!id){

id=New(x);return;//若节点为空，则新建一个节点

}

if(x==val[id])cnt[id]++;//若节点已存在，则副本数++;

else{//要满足BST性质，小于插到左边，大于插到右边

int d=x<val[id]?0:1;//这个d是方向的意思，按照BST的性质，小于本节点则向左，大于向右

insert(ch[id][d],x);//递归实现

if(dat[id]<dat[ch[id][d]])Rotate(id,d^1);//(参考一下图)与左节点交换右旋，与右节点交换左旋

}

pushup(id);

}

void Remove(int &id,int x){

if(!id)return;//到这了发现查不到这个节点，该点不存在，直接返回

if(x==val[id]){

if(cnt[id]>1){cnt[id]--,pushup(id);return;}

if(ch[id][0]||ch[id][1]){//发现只有一个值，且有儿子节点,我们只能把值旋转到底部删除

if(!ch[id][1]||dat[ch[id][0]]>dat[ch[id][1]]){//当前点被移走之后，会有一个新的点补上来(左儿子或右儿子)，按照优先级，优先级大的补上来

Rotate(id,1),Remove(ch[id][1],x);//我们会发现，右旋是与左儿子交换，当前点变成右节点；左旋则是与右儿子交换，当前点变为左节点

}

else Rotate(id,0),Remove(ch[id][0],x);

pushup(id);

}

else id=0;//发现本节点是叶子节点，直接删除

return;//这个return对应的是检索到值de所有情况

}

if (x<val[id])Remove(ch[id][0],x);

else Remove(ch[id][1],x);

pushup(id);

}

int get\_rk(int id,int x){

if(!id)return 0;//若查询值不存在，返回

if(x==val[id])return size[ch[id][0]] + 1;//查询到该值，由BST性质可知：该点左边值都比该点的值(查询值)小，故rank为左儿子大小 + 1

else if(x<val[id])return get\_rk(ch[id][0],x);//发现需查询的点在该点左边，往左边递归查询

else return size[ch[id][0]]+cnt[id]+get\_rk(ch[id][1],x);//若查询值大于该点值。说明询问点在当前点的右侧，且此点的值都小于查询值，所以要加上cnt[id]

}

int get\_ark(int id,int rk){

if(!id)return inf;

if(rk<=size[ch[id][0]])return get\_ark(ch[id][0],rk);//左边排名已经大于rank了，说明rank对应的值在左儿子那里

else if(rk<=size[ch[id][0]]+cnt[id])return val[id];//上一步排除了在左区间的情况，若是rank在左与中(目前节点)中，则直接返回目前节点(中区间)的值

else return get\_ark(ch[id][1],rk-size[ch[id][0]]-cnt[id]);//剩下只能在右区间找了，rank减去左区间大小和中区间，继续递归

}

int get\_pre(int x){

int id=root,pre;

while(id){

if(val[id]<x)pre=val[id],id=ch[id][1];

else id=ch[id][0];

}

return pre;

}

int get\_suf(int x){

int id=root,next;

while(id){

if(val[id]>x)next=val[id],id=ch[id][0];//同理，满足条件向左寻找更小解(也就是最优解)

else id=ch[id][1];//与上方同理

}

return next;

}

int main(){

build();

int m,cmd,x;scanf("%d",&m);

while(m--){

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

if(cmd==1)insert(root,x);

else if(cmd==2)Remove(root,x);

else if(cmd==3)printf("%d\n",get\_rk(root,x)-1);//注意：因为初始化时插入了INF和-INF,所以查询排名时要减1(-INF不是第一小，是“第零小”)

else if(cmd==4)printf("%d\n",get\_ark(root,x+1));//同理，用排名查询值得时候要查x+1名，因为第一名(其实不是)是-INF

else if(cmd==5)printf("%d\n",get\_pre(x));

else if(cmd==6)printf("%d\n",get\_suf(x));

}

return 0;

}

##### 5.5.2 Spaly

//hdoj1890

#include <bits/stdc++.h>

using namespace std;

const int N=1e5+10;

int n,pos[N];

struct node{

int x,id;

}a[N];

struct Splay\_Tree{

int fa[N],ch[N][2],sz[N];

int flip[N],rt;

void Flip(int x){

swap(ch[x][0],ch[x][1]);//要放这里，放下面会出问题

flip[x]^=1;

}

void pushdown(int x){

if(flip[x]){

Flip(ch[x][0]);Flip(ch[x][1]);flip[x]=0;

}

}

int pd[N];

void P(int x){

int cnt=0;

while(x){

pd[++cnt]=x;

x=fa[x];

}

while(cnt)pushdown(pd[cnt--]);

}

void pushup(int x){

sz[x]=sz[ch[x][0]]+sz[ch[x][1]]+1;

}

void Rotate(int x){

int y=fa[x],g=fa[y],c=ch[y][1]==x;

ch[y][c]=ch[x][c^1];fa[ch[x][c^1]]=y;

ch[x][c^1]=y;fa[y]=x;fa[x]=g;

if(g)ch[g][ch[g][1]==y]=x;

pushup(y);

}

void Splay(int x,int g=0){//x转到g的下面

P(x);

for(int y;(y=fa[x])!=g;Rotate(x))

if(fa[y]!=g)

Rotate((ch[fa[y]][1]==y)==(ch[y][1]==x)?y:x);

pushup(x);

if(!g)rt=x;

}

int Build(int f,int l,int r){

if(l>r)return 0;

int mid=(l+r)>>1;fa[mid]=f;flip[mid]=0;

ch[mid][0]=Build(mid,l,mid-1);

ch[mid][1]=Build(mid,mid+1,r);

sz[mid]=1;

pushup(mid);

return mid;

}

void solve(int i){

Splay(pos[1]);

Splay(pos[i],pos[1]);

printf("%d ",sz[ch[ch[rt][1]][0]]+1);

Splay(pos[i]);

int p=ch[pos[i]][1];

while(ch[p][0]){

pushdown(p);

p=ch[p][0];

}

pushdown(p);

Splay(pos[i-1]);Splay(p,pos[i-1]);//pos[i-1]+1到p-1的区间翻转

Flip(ch[ch[rt][1]][0]);

}

void init(int n){

rt=Build(0,1,n);

}

}sp;

bool cmp(node a,node b){

if(a.x!=b.x)

return a.x<b.x;

return a.id<b.id;

}

int main(){

while(~scanf("%d",&n)&&n){

sp.init(n+2);

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

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

a[i].id=i;

}

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

for(int i=1;i<=n;i++)pos[i+1]=a[i].id+1;

pos[1]=1;pos[n+2]=n+2;

for(int i=2,p;i<n+1;i++){

sp.solve(i);

}

printf("%d\n",n);

}

return 0;

}

#### 5.6 Kd Tree

#include<bits/stdc++.h>

using namespace std;

const int N=6e4+10,K=5;

const int inf=0x3f3f3f3f;

#define sqr(x) (x)\*(x)

int k,n,idx;

struct point{

int x[K];

bool operator < (const point &u) const{

return x[idx]<u.x[idx];

}

}po[N];

typedef pair<double,point>tp;

priority\_queue<tp>nq;

struct kdTree{

point pt[N<<2];

int son[N<<2];

void build(int l,int r,int rt=1,int dep=0){

if(l>r) return;

son[rt]=r-l;

son[rt\*2]=son[rt\*2+1]=-1;

idx=dep%k;

int mid=(l+r)/2;

nth\_element(po+l,po+mid,po+r+1);

pt[rt]=po[mid];

build(l,mid-1,rt\*2,dep+1);

build(mid+1,r,rt\*2+1,dep+1);

}

void query(point p,int m,int rt=1,int dep=0){

if(son[rt]==-1) return;

tp nd(0,pt[rt]);

for(int i=0;i<k;i++) nd.first+=sqr(nd.second.x[i]-p.x[i]);

int dim=dep%k,x=rt\*2,y=rt\*2+1,fg=0;

if(p.x[dim]>=pt[rt].x[dim]) swap(x,y);

if(~son[x]) query(p,m,x,dep+1);

if(nq.size()<m) nq.push(nd),fg=1;

else{

if(nd.first<nq.top().first) nq.pop(),nq.push(nd);

if(sqr(p.x[dim]-pt[rt].x[dim])<nq.top().first) fg=1;

}

if(~son[y]&&fg) query(p,m,y,dep+1);

}

}kd;

void print(point &p){

for(int j=0;j<k;j++) printf("%d%c",p.x[j],j==k-1?'\n':' ');

}

int main(){

while(scanf("%d%d",&n,&k)!=EOF){//点数，维数

for(int i=0;i<n;i++) for(int j=0;j<k;j++) scanf("%d",&po[i].x[j]);

kd.build(0,n-1);

int q;scanf("%d",&q);//询问次数

while (q--){

point ask;

for(int j=0;j<k;j++) scanf("%d",&ask.x[j]);

int m;scanf("%d",&m);//最近的m个点

kd.query(ask,m);

printf("the closest %d points are:\n", m);

point pt[20];

for(int j=1;!nq.empty();j++)pt[j]=nq.top().second,nq.pop();

for(int j=m;j>=1;j--)print(pt[j]);

}

}

return 0;

}

#### 5.7 LCT

#include<bits/stdc++.h>

using namespace std;

const int N=110005;

struct LCT{

int tr[N],val[N],fa[N],son[N][2],rev[N];

int top,sta[N];

bool isroot(int t){

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

}

bool isson(int t){return t==son[fa[t]][1];}

void update(int t){

tr[t]=tr[son[t][0]]^tr[son[t][1]]^val[t];

}

void pushdown(int t){

if (rev[t]){

rev[son[t][0]]^=1,rev[son[t][1]]^=1;

swap(son[t][0],son[t][1]);

rev[t]=0;

}

}

void rotate(int t){

int f=fa[t],g=fa[f],fis=isson(f),tis=isson(t);

fa[t]=g;

if (!isroot(f)) son[g][fis]=t;

fa[son[t][tis^1]]=f,son[f][tis]=son[t][tis^1];

fa[f]=t,son[t][tis^1]=f;

update(f),update(t);

}

void splay(int t){

sta[top=1]=t;

int x=t;

while (!isroot(x))x=fa[x],sta[++top]=x;

while (top)pushdown(sta[top--]);

while (!isroot(t)){

int f=fa[t];

if(isroot(f))rotate(t);

else if(isson(f)==isson(t))rotate(f),rotate(t);

else rotate(t),rotate(t);

}

update(t);

}

void access(int t){//访问

for(int i=0;t;i=t,t=fa[t])

splay(t),son[t][1]=i,update(t);

}

void makeroot(int t){//换根

access(t),splay(t),rev[t]^=1;

}

int findroot(int t){//找根（在真实的树中的）

access(t),splay(t);

while(son[t][0])pushdown(t),t=son[t][0];

splay(t);

return t;

}

void split(int u,int v){//提取路径

makeroot(v),access(u),splay(u);

}

void link(int u,int v){//连边

makeroot(u);

if (findroot(v)!=u)fa[u]=v;

}

void cut(int u,int v){//断边

split(u,v);

if (son[u][0]==v&&!son[v][1])son[u][0]=0,fa[v]=0;

}

}lct;

int main(){

int n,m,c,x,y;

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

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

scanf("%d",&lct.val[i]);

}

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

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

if (c==0){//询问x->y的xor值

lct.split(x,y);

printf("%d\n",lct.tr[x]);

}

else if (c==1){//连接x,y

lct.link(x,y);

}

else if (c==2){//删除x,y

lct.cut(x,y);

}

else{//将x的权值改为y

lct.splay(x),lct.val[x]=y;

}

}

return 0;

}

#### 5.8 并查集

##### 5.8.1 可撤销并查集

struct usf{

int f[N],ds[N],rk[N];

struct node{

int x,y,f,rk;

bool ds;

}st[N];

int top=0;

void init(int n){

top=0;

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

f[i]=i;ds[i]=0;rk[i]=0;

}

}

int unite(int x,int y){

bool dx=0,dy=0;

while (x!=f[x])dx^=ds[x],x=f[x];

while (y!=f[y])dy^=ds[y],y=f[y];

if (x==y)return dx^dy;//相同并查集的返回01，0有环，1无环

if (rk[x]<rk[y]){

swap(x,y);swap(dx,dy);

}

st[++top]={x,y,f[y],rk[x],ds[y]};

f[y]=x;ds[y]=dx^dy^1;

if (rk[x]==rk[y])rk[x]++;

return 2;

}

void undo(int t){

while (t--){

node o=st[top];--top;

f[o.y]=o.f;ds[o.y]=o.ds;

rk[o.x]=o.rk;

}

}

}usf;

##### 5.8.2 可持久化并查集

#include<bits/stdc++.h>

using namespace std;

const int N=2e5+10;

int n,q;

struct Sustainable\_DSU {

static const int N = 1e5 + 5, M = 2e6 + 5;

int root[N], lson[M], rson[M], fa[M], rnk[M], tot, n;

inline void build(int &rt, int l, int r) {

rt = ++tot;

if(l == r) {fa[rt] = l; return ;}

int m = l+r >> 1;

build(lson[rt], l, m);

build(rson[rt], m+1, r);

}

inline void init(int \_n) {

n = \_n;

tot = 0;

build(root[0], 1, n);

}

inline void Update(int old, int &rt, int p, int v, int l, int r) {

rt = ++tot;

lson[rt] = lson[old], rson[rt] = rson[old];

if(l == r) {

fa[rt] = v;

rnk[rt] = rnk[old];

return ;

}

int m = l+r >> 1;

if(p <= m) Update(lson[rt], lson[rt], p, v, l, m);

else Update(rson[rt], rson[rt], p, v, m+1, r);

}

inline void update(int rt, int p, int l, int r) {

if(l == r) { rnk[rt]++; return ;}

int m = l+r >> 1;

if(p <= m) update(lson[rt], p, l, m);

else update(rson[rt], p, m+1, r);

}

///返回rt版本p位置fa数组下标

inline int query(int rt, int p, int l, int r) {

if(l == r) return rt;

int m = l+r >> 1;

if(p <= m) return query(lson[rt], p, l, m);

else return query(rson[rt], p, m+1, r);

}

///返回rt版本p所在并查集fa数组下标

inline int Find(int rt, int p) {

int now = query(rt, p, 1, n);

if(fa[now] == p) return now;

else return Find(rt, fa[now]);

}

///在i时刻合并x和y所在并查集

inline void Merge(int i, int x, int y) {

root[i] = root[i-1];

int px = Find(root[i], x), py = Find(root[i], y);

if(fa[px] != fa[py]) {

if(rnk[px] > rnk[py]) swap(px, py);

Update(root[i-1], root[i], fa[px], fa[py], 1, n);

if(rnk[px] == rnk[py]) update(root[i], fa[py], 1, n);

}

}

}f;

signed main(){

int T,op,x,y,n,m;

cin>>n>>m;

f.init(n);

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

scanf("%d",&op);

if (op==1){//x,y合并

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

f.Merge(i,x,y);

}

else if (op==2){//回到x此操作后

scanf("%d",&x);

f.root[i]=f.root[x];

}

else{//询问x,y是否合并了

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

f.root[i]=f.root[i-1];

int xx=f.Find(f.root[i],x),yy=f.Find(f.root[i],y);

if (xx==yy)puts("1");

else puts("0");

}

}

return 0;

}

#### 5.9圆方树

#### 5.10 珂朵莉树

#include<bits/stdc++.h>

using namespace std;

#define int long long

const int N=1e5+10;

namespace Qpow{

long long pow(long long a, long long b, long long mod){

if (!a) return 0;

long long res = 1; a %= mod;

for ( ; b; (a \*= a) %= mod, b >>= 1ll)

if (b & 1) (res \*= a) %= mod;;

return res;

}

};

//build

struct Node{

int l, r;

mutable long long val;

Node(int a = -1, int b = -1, long long c = 0){

l = a, r = b, val = c;

}

bool operator < (const Node &a) const{

return l < a.l;

}

};

set<Node> st;

//modify

set<Node>::iterator split(int pos){

set<Node>::iterator it = st.lower\_bound(Node(pos));

if (it != st.end() && it->l == pos) return it;

--it; Node tmp = \*it; st.erase(it);

st.insert(Node(tmp.l, pos - 1, tmp.val));

return st.insert(Node(pos, tmp.r, tmp.val)).first; //first return iterator

}

void assign(int l, int r, long long val){

set<Node>::iterator itr = split(r + 1), itl = split(l);

st.erase(itl, itr);

st.insert((Node){l, r, val});

}

void add(int l, int r, long long val){

set<Node>::iterator itr = split(r + 1), itl = split(l);

for (set<Node>::iterator it = itl; it != itr; ++it)

it->val += val;

}

//query

long long querySum(int l, int r){

set<Node>::iterator itr = split(r + 1), itl = split(l); long long res = 0;

for (set<Node>::iterator it = itl; it != itr; ++it)

res += (it->r - it->l + 1) \* it->val;

return res;

}

long long querySumWithPow(int l, int r, long long x, long long mod){

set<Node>::iterator itr = split(r + 1), itl = split(l); long long res = 0;

for (set<Node>::iterator it = itl; it != itr; ++it)

(res += (it->r - it->l + 1) \* Qpow::pow(it->val, x, mod)) %= mod;

return res;

}

long long queryKth(int l, int r, int k){

vector< pair<long long, int> > vec(0);

set<Node>::iterator itr = split(r + 1), itl = split(l);

for (set<Node>::iterator it = itl; it != itr; ++it)

vec.push\_back(make\_pair(it->val, it->r - it->l + 1));

sort(vec.begin(), vec.end());

for (vector< pair<long long, int> >::iterator it = vec.begin(); it != vec.end(); ++it)

if ((k -= it->second) <= 0) return it->first;

return -1; //note:if there are negative numbers, return another impossible number.

}

int n,m,seed,mod;

int a[N];

int rnd(){

int ret=seed;

seed=(seed\*7+13)%1000000007;

return ret;

}

signed main(){

int T;

cin>>n>>m>>seed>>mod;

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

a[i]=(rnd()%mod)+1;

st.insert((Node){i,i,a[i]});//插入

}

while (m--){

int x,y,l,r,op;

op=(rnd()%4)+1;l=(rnd()%n)+1;r=(rnd()%n)+1;

if (l>r)swap(l,r);

if (op==3)x=(rnd()%(r-l+1))+1;

else x=(rnd()%mod)+1;

if (op==4)y=(rnd()%mod)+1;

if (op==1)add(l,r,x);//[l,r]区间全部+x

else if (op==2)assign(l,r,x);//[l,r]区间全部=x

else if (op==3)printf("%I64d\n",queryKth(l,r,x));//[l,r]区间第x小

else if (op==4)printf("%I64d\n",querySumWithPow(l,r,x,y));//[l,r]区间的x次方和%y

}

return 0;

}

## 6 各种操作

#pragma GCC optimize("Ofast,unroll-loops")

#pragma GCC target("avx,avx2,sse,sse2")

#pragma GCC optimize(2)

clock\_t start = clock();

clock\_t ends=clock();

cout <<"Running Time : "<<(double)(ends - start)/ CLOCKS\_PER\_SEC << endl;