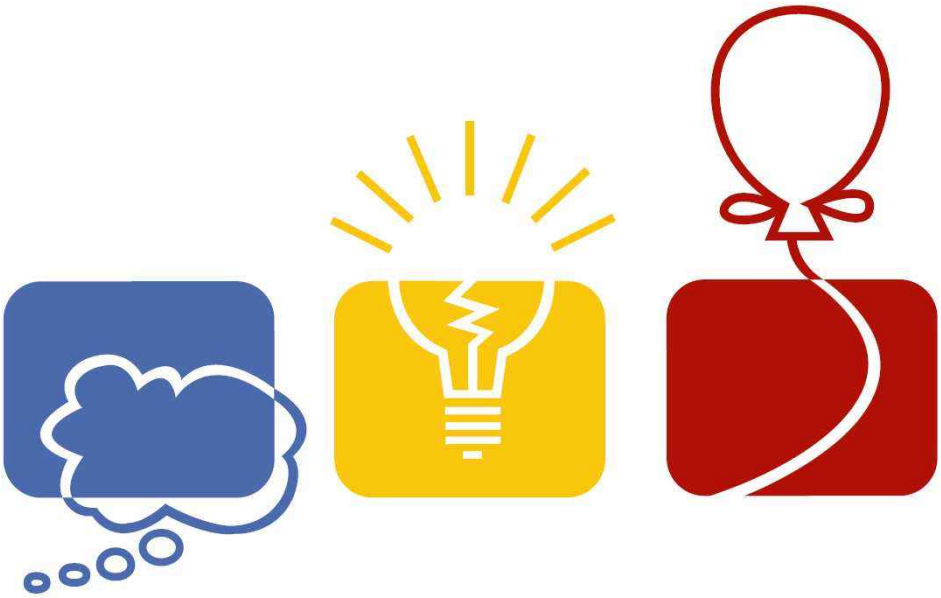

ACM Template of Gipsy

Ocean University of China

Gipsy

October 4, 2019

Contents

0 Vim

```
syntax on
set cindent
set nu
set tabstop=4
set shiftwidth=4
set background=dark
map <C-A> ggVG"+y
map <F5> :call Run()<CR>
func! Run()
    exec "w"
    exec "!g++ -Wall % -o %<"
    exec "!./%<"
endfunc
set autoindent
set cindent
inoremap ( ()<ESC>i
inoremap [ []<ESC>i
inoremap { {}<ESC>i
inoremap ' ''<ESC>i
inoremap " ""<ESC>i
```

1 Math

1.1 Number Theory

1.1.1 CRT

r 为数组长度
m 是 mod 数，需要相互互质
a 是 mod 后的余数
M 最小共倍数
ans 答案

```
void ChinaRT (int r, int *m, int *a, int &M, int &ans) {
    M = 1, ans = 0;
    int x, y;
    for (int i = 1; i <= r; ++i) M *= m[i];
    for (int i = 1; i <= r; ++i) {
        exgcd (M / m[i], m[i], x, y);
        ans = (ans + M / m[i] * x * a[i]) % M;
    }
    if (ans < 0) ans += M;
}
```

1.1.2 Du Sieve

计算欧拉函数和莫比乌斯函数的前缀和
处理的数据范围 1e12

```
const int N = 4e6;
int prime [N + 10], tot;
int v [N + 10];
ll phi [N + 10];
ll miu [N + 10];
map <ll, ll> _phi, _miu;
ll calcmiu (ll n) {
    if (n < N) return miu [n];
    if (_miu.count (n) ) return _miu [n];
    ll x = 2, ans = 1;
    while (x <= n) {
        ll y = n / (n / x);
        ans -= calcmiu (n / x) * ( y - x + 1);
        x = y + 1;
    }
    return _miu [n] = ans;
}
ll calcphi (ll n) {
    if (n < N) return phi [n];
    if (_phi.count (n) ) return _phi [n];
    ll x = 2, ans = n * (n + 1) / 2;
    while (x <= n){
        ll y = n / (n / x);
        ans -= calcphi (n / x) * ( y - x + 1);
        x = y + 1;
    }
    return _phi [n] = ans;
}

int main(){
    // 计算出phi和miu
```

```
    for (int i = 2; i <= N; ++i) {
        phi [ i ] += phi [ i - 1 ];
        miu [ i ] += miu [ i - 1 ];
    }
    ll x;
    scanf ("%lld", &x);
    cout << calcphi (x) << " " << calcmiu (x) << endl;
}
```

1.1.3 ExBSGS

```
ll BSGS (ll a, ll b, ll p) {
    if (a %= p, b %= p, b == 1) return 0;
    ll t = 1; ll f, g, delta = 0, m = sqrt (p) + 1, i;
    for (g = gcd (a, p) ; g != 1; g = gcd (a , p) ){
        if (b % g) return -1;
        b /= g, p /= g, t = t * (a / g) % p, delta ++;
        if (b == t) return delta;
    }
    map <ll, ll> Hash;
    for (i = 0; i < m; i++, b = b * a % p) Hash [b] = i;
    for (i = 1, f = quickmod (a, m, p); i <= m + 1; i++)
    if (t = t * f % p, Hash.count (t) ) return i * m - Hash [t] + delta;
    return -1;
}
```

1.1.4 ExCRT

a mod 数，r 余数，n 长度，不要求 mod 数彼此互质
需要函数 exgcd
ExCRT 返回答案

```
ll a [100005], r [100005]; int n;
ll ExCRT () {
    ll M = a [1], R = r [1], x, y, d;
    for (int i = 2; i <= n; i++) {
        d = exgcd (M, a [i], x, y);
        if ( (R - r [i]) % d != 0) return -1;
        x = (R - r [i]) / d * x % a [i];
        R -= x * M;
        M = M / d * a [i];
        R %= M;
    }
    return (R % M + M) % M;
}
```

1.1.5 ExGCD

```
int exgcd (int a, int b, int &x, int &y){
    if (b == 0) {
    x = 1, y = 0;
    return a;
    }
    int q = exgcd (b, a % b, y, x);
    y -= a / b * x;
    return q;
}
```

1.1.6 Inv

线性处理逆元

```
inv [0] = 0;
inv [1] = 1;
for (int i = 2; i < n; i ++) {
    inv [i] = (M-1ll * M / i * inv [M % i] % M) % M;
}
```

1.1.7 Mobius

```
for (int i = 1; i <= N; ++i) U[i] = 1;
for (int i = 2; i <= N; ++i){
    if (v[i]) continue;
    for (int j = i; j <= N; j += i){
        v[j] = 1;
        if (j % (i * i) == 0) U[j] = 0;
        U[j] *= -1;
    }
}
u[1] = 1;
for (int i = 2; i <= N; ++i){
    if (! v[i]) u[i] = -1, p[++cnt] = i;
    for(int j = 1; j <= cnt && i * p[j] <= N; ++j){
        int k = i * p[j]; v[k] = 1;
        if(i % p[j]) u[k] = -u[i];
        else{
            u[k] = 0;
            break;
        }
    }
}
```

1.1.8 Pollard Rho

MR(次数 (3), 需要判断的质数) 是质数返回 1
find(需要分解的数, 次数 (100)) 返回在 a 数组中
需要大数 mul,gcd, 大数快速幂 quickmod

```
vector <ll> a;
bool TD(ll a,ll n){
    ll m=n-1,x,y; int num=0;
    while (!(m&1)) m>>=1,num++;
    x=quickmod(a,m,n);
    for (int i=1;i<=num;x=y,i++){
        y=quickmod(x,2,n);
        if ((y==1)&&(x!=1)&&(x!=n-1))
            return true;
    }
    return y!=1;
}
bool MR(int t,ll x){
    if (x==1) return false;
    if (x==2) return true;
    if (!(x&1)) return false;
    while (t--)
```

```
        if (TD(rand()%(x-1)+1,x))
            return false;
    return true;
}
ll PR(ll n,int t){
    ll i=1,k=2,x=rand()%n,y=x,d;
    while (1){
        i++,x=(mul(x,x,n)+t)%n,d=gcd(y-x,n);
        if (d>1&&d<n) return d;
        if (y==x) return n;
        if (i==k) y=x,k<<=1;
    }
}
void find(ll x,int k){
    if (x==1) return;
    if (MR(3,x)) return (void)a.push_back(x);
    ll t=x;
    while (t==x) t=PR(x,k--);
    find(t,k),find(x/t,k);
}
```

1.1.9 Primitive Root

求出质数 p 的首个原根，复杂度约为 $p * log^2(p)/phi(p-1)$
前 78494 个质数 (1 1000000) 当中，原根的平均值约为 4.88，可以忽略为常数。
因此复杂度只有 $log^2(p)$
solve 函数返回原根

```
#define N 1000000
int top=0;
ll st[40];
void init (ll m) {
    top = 0;
    memset (st, 0, sizeof st);
    for (ll i = 2; i <= sqrt (m) + 1; i++) {
        if ( m % i == 0) {
            st [top++] = i;
            while (m % i == 0) m /= i;
        }
    }
    if (m>1) st[top++]=m;
}
ll solve (ll p) {
    init (p-1);
    for (ll g = 1; g <= p - 1; g++) {
        bool bb = true;
        for (int j = 0; j < top; j++) {
            ll mod = power (g, (p - 1) / st[j], p); /// 快速幂
            if (mod == 1) {
                bb = false;
                break;
            }
        }
        if (bb) {
            return g;
        }
    }
}
```

1.1.10 Square Mod

二次剩余

```
struct T{
    ll p, d;
};
ll w;
//二次域乘法
T multi_er(T a, T b, ll m){
    T ans;
    ans.p = (a.p * b.p % m + a.d * b.d % m * w % m) % m;
    ans.d = (a.p * b.d % m + a.d * b.p % m) % m;
    return ans;
}
//二次域上快速幂
T power(T a, ll b, ll m){
    T ans;
    ans.p = 1;
    ans.d = 0;
    while(b){
        if(b & 1){
            ans = multi_er(ans, a, m);
            b--;
        }
        b >>= 1;
        a = multi_er(a, a, m);
    }
    return ans;
}
//求勒让德符号
ll Legendre(ll a, ll p){
    return quick_mod(a, (p-1)>>1, p);
}
ll mod(ll a, ll m){
    a %= m;
    if(a < 0) a += m;
    return a;
}
ll Solve(ll n,ll p){
    if(p == 2) return 1;
    if (Legendre(n, p) + 1 == p)
        return -1;
    ll a = -1, t;
    while(true){
        a = rand() % p;
        t = a * a - n;
        w = mod(t, p);
        if(Legendre(w, p) + 1 == p) break;
    }
    T tmp;
    tmp.p = a;
    tmp.d = 1;
    T ans = power(tmp, (p + 1)>>1, p);
    return ans.p;
}
int main()
    int n, p;
    scanf("%d %d",&n,&p);
```

```
    n %= p;
    int a = Solve(n, p);
    if(a == -1) puts("No root");
    int b = p - a;
    if(a > b) swap(a, b);
    if(a == b) printf("%d\n",a);
    else printf("%d %d\n",a,b);
}
```

1.2 Fast Transformation

1.2.1 FFT

```
const double PI = acos(-1.0);
//复数结构体
struct Complex{
    double x,y;//实部和虚部 x+yi
    Complex(double _x = 0.0,double _y = 0.0){
        x = _x;
        y = _y;
    }
    Complex operator -(const Complex &b)const{
        return Complex(x-b.x,y-b.y);
    }
    Complex operator +(const Complex &b)const{
        return Complex(x+b.x,y+b.y);
    }
    Complex operator *(const Complex &b)const{
        return Complex(x*b.x-y*b.y,x*b.y+y*b.x);
    }
};
/*
* 进行FFT和IFFT前的反转变换。
* 位置i和 （i二进制反转后位置）互换
* len必须去2的幂
*/
void change(Complex y[],int len)
{
    int i,j,k;
    for(i = 1, j = len/2; i <len-1; i++)
    {
        if(i < j)swap(y[i],y[j]);
//交换互为小标反转的元素， i<j保证交换一次
//i做正常的+1， j左反转类型的+1,始终保持i和j是反转的
        k = len/2;
        while(j >= k)
        {
            j -= k;
            k /= 2;
        }
        if(j < k)j += k;
    }
}
/*
* 做FFT
* len必须为2^k形式,
* on==1时是DFT， on==-1时是IDFT
*/
void fft(Complex y[],int len,int on)
```

```
{
    change(y,len);
    for(int h = 2; h <= len; h <<= 1)
    {
        Complex wn(cos(-on*2*PI/h),sin(-on*2*PI/h));
        for(int j = 0; j < len; j+=h)
        {
            Complex w(1,0);
            for(int k = j; k < j+h/2; k++)
            {
                Complex u = y[k];
                Complex t = w*y[k+h/2];
                y[k] = u+t;
                y[k+h/2] = u-t;
                w = w*wn;
            }
        }
    }
    if(on == -1)
        for(int i = 0; i < len; i++)
            y[i].x /= len;
}
```

1.2.2 NTT

```
const int mod = (479<<21)+1;
const int g = 3;  //原根
long long quick_mod(long long a,long long b)
{
    long long ans=1;
    while(b){
        if(b&1)
            ans=ans*a%mod;
        a=a*a%mod;
        b>>=1;
    }
    return ans;
}
void NTT(int n,long long a[],bool on=false) //长度为N (2的次数),默认正变换, 逆变换加true
{
    int r=0;
    while((1<<++r)!=n);
    for(int i=0; i<n; i++){
        int tmp=0;
        for(int j=0; j<r; j++)//蝴蝶操作
            if(i&(1<<j))
                tmp+=1<<(r-j-1);
        if(i<tmp)
            swap(a[i],a[tmp]);
    }
    for(int i=1; i<=r; i++){
        int m=1<<i;
        long long wn=quick_mod(g,(mod-1)/m);
        for(int k=0; k<n; k+=m){
            long long w=1;
            for(int j=0; j<(m>>1); j++){
                long long t,u;
                t=w*(a[k+j+(m>>1)]%mod)%mod;
```

```
                u=a[k+j]%mod;
                a[k+j]=(u+t)%mod;
                a[k+j+(m>>1)]=((u-t)%mod+mod)%mod;
                w=w*wn%mod;
            }
        }
    }
    if(on){
        for(int i=1; i<n>>1; i++)
            swap(a[i],a[n-i]);
        long long inv=quick_mod(n,mod-2);
        for(int i=0; i<n; i++)
            a[i]=a[i]%mod*inv%mod;
    }
}
```

1.2.3 FWT

```
const int mod = 1e9+7;
int inv2 = 500000004;
void FWT(int a[], int n)//请确保n为2的整数次幂
{
    for(int d = 1; d < n; d <<= 1){
        for(int m=d<<1,i=0;i<n;i+=m){
            for(int j=0;j<d;j++){
                int x=a[i+j],y=a[i+j+d];
                //xor_MOD:a[i+j]=(x+y)%mod,a[i+j+d]=(x-y+mod)%mod;
                //xor:a[i+j]=x+y,a[i+j+d]=x-y;
                //and:a[i+j]=x+y;
                //or:a[i+j+d]=x+y;
            }
        }
    }
}

void UFWT(int a[],int n)//请确保n为2的整数次幂
{
    for(int d=1;d<n;d<<=1){
        for(int m=d<<1,i=0;i<n;i+=m){
            for(int j=0;j<d;j++){
                int x=a[i+j],y=a[i+j+d];
                //xor_MOD:a[i+j]=1LL*(x+y)*inv2%mod,a[i+j+d]=(1LL*(x-y)*inv2%mod+mod)%
    mod;
                //xor:a[i+j]=(x+y)/2,a[i+j+d]=(x-y)/2;
                //and:a[i+j]=x-y;
                //or:a[i+j+d]=y-x;
            }
        }
    }
}
```

1.2.4 NTT INV

```
#include<bits/stdc++.h>
using namespace std;
#define RI register int
const int mod=998244353,G=3,N=2100000;
int n;
```

```
int a[N],b[N],c[N],rev[N];
int ksm(int x,int y) {
    int re=1;
    for(;y;y>>=1,x=1LL*x*x%mod) if(y&1) re=1LL*re*x%mod;
    return re;
}
void NTT(int *a,int n,int x) {
    for(RI i=0;i<n;++i) if(i<rev[i]) swap(a[i],a[rev[i]]);
    for(RI i=1;i<n;i<<=1) {
        RI gn=ksm(G,(mod-1)/(i<<1));
        for(RI j=0;j<n;j+=(i<<1)) {
            RI t1,t2,g=1;
            for(RI k=0;k<i;++k,g=1LL*g*gn%mod) {
                t1=a[j+k],t2=1LL*g*a[j+k+i]%mod;
                a[j+k]=(t1+t2)%mod,a[j+k+i]=(t1-t2+mod)%mod;
            }
        }
    }
    if(x==1) return;
    int ny=ksm(n,mod-2); reverse(a+1,a+n);
    for(RI i=0;i<n;++i) a[i]=1LL*a[i]*ny%mod;
}
void work(int deg,int *a,int *b) {
    if(deg==1) {b[0]=ksm(a[0],mod-2);return;}
    work((deg+1)>>1,a,b);
    RI len=0,orz=1;
    while(orz<(deg<<1)) orz<<=1,++len;
    for(RI i=1;i<orz;++i) rev[i]=(rev[i>>1]>>1)|((i&1)<<(len-1));
    for(RI i=0;i<deg;++i) c[i]=a[i];
    for(RI i=deg;i<orz;++i) c[i]=0;
    NTT(c,orz,1),NTT(b,orz,1);
    for(RI i=0;i<orz;++i)
        b[i]=1LL*(2-1LL*c[i]*b[i]%mod+mod)%mod*b[i]%mod;
    NTT(b,orz,-1);
    for(RI i=deg;i<orz;++i) b[i]=0;
}
int main(){
    n=read();
    for(RI i=0;i<n;++i) a[i]=read();
    work(n,a,b);
    for(RI i=0;i<n;++i) printf("%d ",b[i]);
    return 0;
}
```

1.3 Combinatorics

1.3.1 ExLucas

```

ll n,m,MOD,ans;

ll fast_pow(ll a,ll p,ll Mod)
{
    ll ans=1ll;
    for (;p;p>>=1,a=a*a%Mod)
        if (p&1)
            ans=ans*a%Mod;
    return ans;
}
```

```
void exgcd(ll a,ll b,ll &x,ll &y)
{
    if (!b) x=1ll,y=0ll;
    else exgcd(b,a%b,y,x),y-=a/b*x;
}
ll inv(ll A,ll Mod)
{
    if (!A) return 0ll;
    ll a=A,b=Mod,x=0ll,y=0ll;
    exgcd(a,b,x,y);
    x=((x%b)+b)%b;
    if (!x) x+=b;
    return x;
}
ll Mul(ll n,ll pi,ll pk)
{
    if (!n) return 1ll;
    ll ans=1ll;
    if (n/pk)
    {
        for (ll i=2;i<=pk;++i)
            if (i%pi) ans=ans*i%pk;
        ans=fast_pow(ans,n/pk,pk);
    }
    for (ll i=2;i<=n%pk;++i)
        if (i%pi) ans=ans*i%pk;
    return ans*Mul(n/pi,pi,pk)%pk;
}
ll C(ll n,ll m,ll Mod,ll pi,ll pk)
{
    if (m>n) return 0ll;
    ll a=Mul(n,pi,pk),b=Mul(m,pi,pk),c=Mul(n-m,pi,pk);
    ll k=0ll,ans;
    for (ll i=n;i;i/=pi) k+=i/pi;
    for (ll i=m;i;i/=pi) k-=i/pi;
    for (ll i=n-m;i;i/=pi) k-=i/pi;
    ans=a*inv(b,pk)%pk*inv(c,pk)%pk*fast_pow(pi,k,pk)%pk;
    return ans*(Mod/pk)%Mod*inv(Mod/pk,pk)%Mod;
}
int main()
{
    scanf("%I64d%I64d%I64d",&n,&m,&MOD);
    for (ll x=MOD,i=2;i<=MOD;++i)
        if (x%i==0)
        {
            ll pk=1ll;
            while (x%i==0) pk*=i,x/=i;
            ans=(ans+C(n,m,MOD,i,pk))%MOD;
        }
    printf("%I64d\n",ans);
}
```

1.3.2 Lucas

Caculate C(n,m)mod p p must be a prime

```
ll C(ll n,ll m){
    return n<m||m<0?0:JC[n]*INV[m]%M*INV[n-m]%M;
```

```
}
ll Lucas(ll n,ll m){
    return m?C(n%M,m%M)*Lucas(n/M,m/M)%M:1;
}
```

1.4 Matrix

1.4.1 Matrix FastPow

```
typedef vector<ll> vec;
typedef vector<vec> mat;
mat mul(mat& A, mat& B){
    mat C(A.size(), vec(B[0].size()));
    for (int i = 0; i < A.size(); i++)
        for (int k = 0; k < B.size(); k++)
            if (A[i][k]) // 对稀疏矩阵的优化
                for (int j = 0; j < B[0].size(); j++)
                    C[i][j] = (C[i][j] + A[i][k] * B[k][j]) % mod;
    return C;
}
mat Pow(mat A, ll n){
    mat B(A.size(), vec(A.size()));
    for (int i = 0; i < A.size(); i++) B[i][i] = 1;
    for (; n; n >>= 1, A = mul(A, A))
        if (n & 1) B = mul(B, A);
    return B;
}
```

1.4.2 Gauss

```
const double eps=1e-8;
double a[N][N];
void gauss(int n){
    for(int i=1;i<=n;++i){
        int mx=i;
        for(int j=i+1;j<=n;j++){
            if(fabs(a[j][i])>fabs(a[mx][i]))mx=j;
        }
        if(mx!=i)for(int j=1;j<=n;j++)swap(a[i][j],a[mx][j]);
        for(int j=1;j<=n;++j){
            if(i==j)continue;
            double rate=a[j][i]/a[i][i];
            for(int k=i;k<=n;++k)a[j][k]-=a[i][k]*rate;
        }
    }
}
```

1.4.3 Linear Basis

```
struct Linear_Basis {
    ll d [63], p [63], tot;
    void init () {
        tot = 0;
        memset (d, 0, sizeof (d) );memset (p, 0, sizeof (p) );
    }
    bool ins (ll x) {
        for (int i = 62; i >= 0; i--)
```

```
            if (x & (1ll << i) ) {
                if (! d[i]) {d [i] = x; break;}
                x ^= d [i];
            }
        return x>0;
    }
} LB;
```

1.4.4 Exgcd Gauss

```
for(int i=2;i<=n;++i)
    for(int j=i+1;j<=n;++j)
        while(a[j][i]){
            int t=a[i][i]/a[j][i];
            for(int k=i;k<=n;++k)a[i][k]=(a[i][k]-1ll*a[j][k]*t%M+M)%M,swap(a[i][k],a[j
    ][k]);
        }
```

1.4.5 BM

```
#include<bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef vector<int> VI;
typedef long long ll;
typedef pair<int,int> PII;
const ll mod=1000000007;
// ll powmod(ll a,ll b)
int _,n;
namespace linear_seq {
    const int N=10010;
    ll res[N],base[N],_c[N],_md[N];

    vector<int> Md;
    void mul(ll *a,ll *b,int k) {
        rep(i,0,k+k) _c[i]=0;
        rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
        for (int i=k+k-1;i>=k;i--) if (_c[i])
            rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
        rep(i,0,k) a[i]=_c[i];
    }
    int solve(ll n,VI a,VI b) {
        ll ans=0,pnt=0;
        int k=SZ(a);
        assert(SZ(a)==SZ(b));
        rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
        Md.clear();
        rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
        rep(i,0,k) res[i]=base[i]=0;
        res[0]=1;
        while ((1ll<<pnt)<=n) pnt++;
```

```
        for (int p=pnt;p>=0;p--) {
            mul(res,res,k);
            if ((n>>p)&1) {
                for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
                rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
            }
        }
        rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
        if (ans<0) ans+=mod;
        return ans;
    }
    VI BM(VI s) {
        VI C(1,1),B(1,1);
        int L=0,m=1,b=1;
        rep(n,0,SZ(s)) {
            ll d=0;
            rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
            if (d==0) ++m;
            else if (2*L<=n) {
                VI T=C;
                ll c=mod-d*powmod(b,mod-2)%mod;
                while (SZ(C)<SZ(B)+m) C.pb(0);
                rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                L=n+1-L; B=T; b=d; m=1;
            } else {
                ll c=mod-d*powmod(b,mod-2)%mod;
                while (SZ(C)<SZ(B)+m) C.pb(0);
                rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                ++m;
            }
        }
        return C;
    }
    int gao(VI a,ll n) {
        VI c=BM(a);
        c.erase(c.begin());
        rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
        return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
    }
};
int main(){
    VI vec;
    int a[10]={0,1,1,2,3,5,8};
    for(int i=1;i<7;i++)
        vec.pb(a[i]);
    printf("%d\n",linear_seq::gao(vec,n-1));
}
```

1.5 Others

1.5.1 博弈问题

1. 有 N 堆石子，每堆石子个数都不少于前一堆的石子个数。两人轮流操作每次操作可以从一堆石子中移走任意多石子，但是要保证操作后仍然满足初始时的条件谁没有石子可移时输掉游戏。问先手是否必胜。
   做差值阶梯博弈：偶数堆不用考虑，奇数堆异或。
2. 反 NIM 博弈：先手胜当且仅当：1，所有堆的石子数都为 1 且游戏的 SG 值为 0；2，有些堆的石子数大于 1 且游戏的 SG 值不为 0。
3. GCD 博弈：有两堆石子, 两个人轮流去取. 每次取的时候, 只能从较多的那堆石子里取, 并且取的数目必须是较少

的那堆石子数目的整数倍. 最后谁能够把一堆石子取空谁就算赢.
设两个数字 $n, m(n > m)$ 当 $n/m >= 2 || n\%m == 0$ 先手获胜，否则不断递归下去

1.5.2 等差数列异或和

```
ll f(ll a,ll b,ll c,ll n){
    if (n==0) return 0;
    ll ans=(b/c)*n*(n-1)/2+(a/c)*n;
    ans=ans+f((b*n+a)%c,c,b%c,((b%c)*n+(a%c))/c);
    return ans;
}
ll F(ll x,ll y,ll z){
    ll ans=0;
    for (ll u=0;u<40;u++)
        ans=ans|((f(x,z,(1ll<<u),(y-x)/z+1)&1)<<u);
    return ans;
}
```

1.5.3 快速乘

```
inline ll qmul(ll x, ll y, ll mod){
    ll t = (x * y - (ll)((long double)x / mod * y + 1.0e-8) * mod);
    return t < 0 ? t + mod : t;
}
```

1.5.4 数位 DP

```
int a[20];
ll dp[20][state];
ll dfs(int pos,,bool lead,bool limit){
    if(pos==-1) return 1;
    if(!limit && !lead && dp[pos][state]!=-1) return dp[pos][state];
    int up=limit?a[pos]:9;
    ll ans=0;
    for(int i=0;i<=up;i++){
        ans+=dfs(pos-1,,lead && i==0,limit && i==a[pos])
    }
    if(!limit && !lead) dp[pos][state]=ans;
    return ans;
}
ll solve(ll x){
    int pos=0;
    while(x){
        a[pos++]=x%10;
        x/=10;
    }
    return dfs(pos-1, ,true,true);
}
int main(){
    ll l,r;
    while(~scanf("%lld%lld",&l,&r)){
        memset(dp,-1,sizeof dp);
        printf("%lld\n",solve(r)-solve(l-1));
    }
}
```

1.5.5 Formula

1. $\sum_{i=1}^{n} i = \sum_{i=1}^{n} \varphi(i)\lfloor n/i \rfloor$
2. $a^n \mod (a^k * y) = a^k(a^{n-k} \mod y)$
3. 小于 n 且互素的数之和为 $n\varphi(n)/2$
4. 错排公式：$D(n) = (n-1)(D(n-2)+D(n-1)) = \sum_{i=2}^{n} \frac{(-1)^k n!}{k!} = [\frac{n!}{e} + 0.5]$
5. 欧拉定理推广：$\gcd(n,p) = 1 \Rightarrow a^n \equiv a^{n\%\varphi(p)} \pmod p$
6. 模的幂公式：$a^n \pmod m = \begin{cases} a^n \mod m & n < \varphi(m) \\ a^{n\%\varphi(m)+\varphi(m)} \mod m & n \geq \varphi(m) \end{cases}$
7. 皮克定理：$S = a + b/2 - 1$ S：面积，a：内部格点数，b：边上格点数
8. 约瑟夫环：$F[1] = 0, F[i] = (F[i-1] + m)\%i$
9. 位数公式：正整数 x 的位数 $N = \log_{10}(n) + 1$
10. 斯特灵公式 $n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$
11. 设 $a > 1, m, n > 0$, 则 $\gcd(a^m - 1, a^n - 1) = a^{\gcd(m,n)} - 1$
12. 设 $a > b, \gcd(a,b) = 1$, 则 $\gcd(a^m - b^m, a^n - b^n) = a^{\gcd(m,n)} - b^{\gcd(m,n)}$

$$G = \gcd(C_n^1, C_n^2, ..., C_n^{n-1}) = \begin{cases} n, & n \text{ is prime} \\ 1, & n \text{ has multy prime factors} \\ p, & n \text{ has single prime factor } p \end{cases}$$

$\gcd(Fib(m), Fib(n)) = Fib(\gcd(m,n))$

13. 若 $\gcd(m,n) = 1$, 则:
    (a) 最大不能组合的数为 $m * n - m - n$
    (b) 不能组合数个数 $N = \frac{(m-1)(n-1)}{2}$
14. $(n+1)lcm(C_n^0, C_n^1, ..., C_n^{n-1}, C_n^n) = lcm(1, 2, ..., n+1)$
15. 若 p 为素数，则 $(x + y + ... + w)^p \equiv x^p + y^p + ... + w^p \pmod p$
16. 卡特兰数：1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012
    $h(0) = h(1) = 1, h(n) = \frac{(4n-2)h(n-1)}{n+1} = \frac{C_{2n}^n}{n+1} = C_{2n}^n - C_{2n}^{n-1}$
17. 伯努利数：$B_n = -\frac{1}{n+1}\sum_{i=0}^{n-1} C_{n+1}^i B_i$

$$\sum_{i=1}^{n} i^k = \frac{1}{k+1}\sum_{i=1}^{k+1} C_{k+1}^i B_{k+1-i}(n+1)^i$$

18. 二项式反演：

$$f_n = \sum_{i=0}^{n}(-1)^i \binom{n}{i} g_i \Leftrightarrow g_n = \sum_{i=0}^{n}(-1)^i \binom{n}{i} f_i$$

$$f_n = \sum_{i=0}^{n}\binom{n}{i} g_i \Leftrightarrow g_n = \sum_{i=0}^{n}(-1)^{n-i}\binom{n}{i} f_i$$

19. 多重求和：对于第 K 重求和 $A[i] = A[i-1] * (K+i-1)/i$
20. 拉格朗日四平方和定理：每个正整数均可表示为 4 个整数的平方和
21. 图论欧拉公式：$V - E + F = 1 + k$,k 为连通分量
22. 球缺公式：$V = \frac{\pi h^2(3r-h)}{3}$
23. 矩阵树定理：度数矩阵减邻接矩阵的的任意一个代数余子式，有向图；外向图：度数改入度；内向图：度数改出度；有向图去掉的行列必须是根节点对应的那个。
24. 当 $x \geq \phi(p)$ 时有 $a^x \equiv a^{x \bmod \phi(p)+\phi(p)} \pmod p$
25. $\mu^2(n) = \sum_{d^2|n} \mu(d)$
26. $\sum_{d|n} \varphi(d) = n$

27. $\sum_{d|n} 2^{\omega(d)} = \sigma_0(n^2)$，其中 ω 是不同素因子个数
28. $\sum_{d|n} \mu^2(d) = 2^{\omega(d)}$
29. 杜教筛
    求 $S(n) = \sum_{i=1}^{n} f(i)$，其中 f 是一个积性函数。
    构造一个积性函数 g，那么由 $(f*g)(n) = \sum_{d|n} f(d)g(\frac{n}{d})$，得到 $f(n) = (f*g)(n) - \sum_{d|n,d<n} f(d)g(\frac{n}{d})$。

$$g(1)S(n) = \sum_{i=1}^{n}(f*g)(i) - \sum_{i=1}^{n}\sum_{d|i,d<i} f(d)g(\frac{n}{d}) \tag{1}$$

$$\overset{t=\frac{i}{d}}{=} \sum_{i=1}^{n}(f*g)(i) - \sum_{t=2}^{n} g(t)S(\lfloor\frac{n}{t}\rfloor) \tag{2}$$

    当然，要能够由此计算 $S(n)$，会对 f, g 提出一些要求：

    (a) $f*g$ 要能够快速求前缀和。
    (b) g 要能够快速求分段和（前缀和）。
    (c) 对于正常的积性函数 $g(1) = 1$，所以不会有什么问题。

    在预处理 $S(n)$ 前 $n^{\frac{2}{3}}$ 项的情况下复杂度是 $O(n^{\frac{2}{3}})$。
30. 数论函数求和
    (a) $\sum_{i=1}^{n} i[gcd(i,n) = 1] = \frac{n\varphi(n)+[n=1]}{2}$
    (b) $\sum_{i=1}^{n}\sum_{j=1}^{m}[gcd(i,j) = x] = \sum_{d}\mu(d)\lfloor\frac{n}{dx}\rfloor\lfloor\frac{m}{dx}\rfloor$
    (c) $\sum_{i=1}^{n}\sum_{j=1}^{m} gcd(i,j) = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{d|gcd(i,j)}\varphi(d) = \sum_{d}\varphi(d)\lfloor\frac{n}{d}\rfloor\lfloor\frac{m}{d}\rfloor$
    (d) $S(n) = \sum_{i=1}^{n}\mu(i) = 1 - \sum_{i=1}^{n}\sum_{d|i,d<i}\mu(d) \overset{t=\frac{i}{d}}{=} 1 - \sum_{t=2}^{n} S(\lfloor\frac{n}{t}\rfloor)$, 利用 $[n=1] = \sum_{d|n}\mu(d)$
    (e) $S(n) = \sum_{i=1}^{n}\varphi(i) = \sum_{i=1}^{n} i - \sum_{i=1}^{n}\sum_{d|i,d<i}\varphi(i) \overset{t=\frac{i}{d}}{=} \frac{i(i+1)}{2} - \sum_{t=2}^{n} S(\frac{n}{t})$, 利用 $n = \sum_{d|n}\varphi(d)$
    (f) $\sum_{i=1}^{n}\mu^2(i) = \sum_{i=1}^{n}\sum_{d^2|n}\mu(d) = \sum_{d=1}^{\lfloor\sqrt{n}\rfloor}\mu(d)\lfloor\frac{n}{d^2}\rfloor$
    (g) $\sum_{i=1}^{n}\sum_{j=1}^{n} gcd^2(i,j) = \sum_{d} d^2\sum_{t}\mu(t)\lfloor\frac{n}{dt}\rfloor^2$
    $\overset{x=dt}{=} \sum_{x}\lfloor\frac{n}{x}\rfloor^2\sum_{d|x} d^2\mu(\frac{t}{x})$
    (h) $\sum_{i=1}^{n}\varphi(i) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}[i\perp j] - 1 = \frac{1}{2}\sum_{i=1}^{n}\mu(i)\cdot\lfloor\frac{n}{i}\rfloor^2 - 1$
31. Fibonacci
    (a) $F_{a+b} = F_{a-1}\cdot F_b + F_a\cdot F_{b+1}$
    (b) $F_1 + F_3 + \cdots + F_{2n-1} = F_{2n}, F_2 + F_4 + \cdots + F_{2n} = F_{2n+1} - 1$
    (c) $\sum_{i=1}^{n} F_i = F_{n+2} - 1$
    (d) $\sum_{i=1}^{n} F_i^2 = F_n\cdot F_{n+1}$
    (e) $F_n^2 = (-1)^{n-1} + F_{n-1}\cdot F_{n+1}$
    (f) $gcd(F_a, F_b) = F_{gcd(a,b)}$
32. 生成函数
    (a) $(1+ax)^n = \sum_{k=0}^{n}\binom{n}{k}a^k x^k$
    (b) $\frac{1-x^{r+1}}{1-x} = \sum_{k=0}^{n} x^k$
    (c) $\frac{1}{1-ax} = \sum_{k=0}^{\infty} a^k x^k$
    (d) $\frac{1}{(1-x)^2} = \sum_{k=0}^{\infty}(k+1)x^k$
    (e) $\frac{1}{(1-x)^n} = \sum_{k=0}^{\infty}\binom{n+k-1}{k}x^k$

(f) $e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$

(g) $\ln(1+x) = \sum_{k=0}^{\infty} \frac{(-1)^{k+1}}{k} x^k$

33. 莫比乌斯反演

(a) $g(n) = \sum_{d|n} f(d) \Leftrightarrow f(n) = \sum_{d|n} \mu(d) g(\frac{n}{d})$

(b) $f(n) = \sum_{n|d} g(d) \Leftrightarrow g(n) = \sum_{n|d} \mu(\frac{d}{n}) f(d)$

34. 低阶等幂求和

(a) $\sum_{i=1}^{n} i^1 = \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n$

(b) $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n$

(c) $\sum_{i=1}^{n} i^3 = \left[\frac{n(n+1)}{2}\right]^2 = \frac{1}{4}n^4 + \frac{1}{2}n^3 + \frac{1}{4}n^2$

(d) $\sum_{i=1}^{n} i^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30} = \frac{1}{5}n^5 + \frac{1}{2}n^4 + \frac{1}{3}n^3 - \frac{1}{30}n$

(e) $\sum_{i=1}^{n} i^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12} = \frac{1}{6}n^6 + \frac{1}{2}n^5 + \frac{5}{12}n^4 - \frac{1}{12}n^2$

2 Data Structure

2.1 树状数组

```
const int N =;
int c[N+2][N+2];
int lowbit (int i) {
    return i & (-i);
}
void add (int x, int y, int value) {
    while (x <= N) {
        int i = y;
        while (i <= N) {
            c[x][i] += value;
            i += lowbit (i);
        }
        x += lowbit (x);
    }
}
int sum (int x, int y) {
    int sum = 0;
    while (x > 0) {
        int i = y;
        while (i > 0) {
            sum += c[x][i];
            i -= lowbit (i);
        }
        x -= lowbit (x);
    }
    return sum;
}
```

2.2 CDQ 分治

W*W 的矩阵，初始值 S. 修改操作数 M<=160000, 询问数 Q<=10000,W<=2000000
输入 1: 把 (x,y) 的格子增加 a
输入 2: 输出以左下角为 (x1,y1), 右上角为 (x2,y2) 的矩阵的权值和

```
#define maxn 200005
int s,w,n,cnt;
int ans[10005],sum[2000005];
struct data{int x,y,z,pos,id;}a[maxn],b[maxn];
inline bool cmp(data a,data b){
    if (a.x==b.x&&a.y==b.y) return a.pos<b.pos;
    else if (a.x==b.x) return a.y<b.y;
    else return a.x<b.x;
}
inline void add(int x,int v){
    for(int i=x;i<=w;i+=i&(-i)) sum[i]+=v;
}
inline int query(int x){
    int ret=0;
    for(int i=x;i;i-=i&(-i)) ret+=sum[i];
    return ret;
}
inline void solve(int l,int r){
    if (l==r) return;
    int mid=(l+r)/2,l1=l,l2=mid+1;
```

```
    for(int i=l;i<=r;++i){
        if (a[i].id<=mid&&!a[i].pos) add(a[i].y,a[i].z);
        if (a[i].id>mid&&a[i].pos) ans[a[i].pos]+=query(a[i].y)*a[i].z;
    }
    for(int i=l;i<=r;++i) if (a[i].id<=mid&&!a[i].pos) add(a[i].y,-a[i].z);
    for(int i=l;i<=r;++i){
        if (a[i].id<=mid) b[l1++]=a[i];
        else b[l2++]=a[i];
    }
    for(int i=l;i<=r;++i) a[i]=b[i];
    solve(l,mid);solve(mid+1,r);
}
int main(){
    read(s);read(w);
    while(1){
        int opt;read(opt);
        if (opt==1){
            int x,y,z;read(x),read(y),read(z);
            n++;a[n]=(data){x,y,z,0,n};
        }
        else if (opt==2){
            int x1,x2,y1,y2;read(x1),read(y1),read(x2),read(y2);--x1,--y1;
            ans[++cnt]=(x2-x1)*(y2-y1)*s;
            n++;a[n]=(data){x1,y1,1,cnt,n};
            n++;a[n]=(data){x1,y2,-1,cnt,n};
            n++;a[n]=(data){x2,y1,-1,cnt,n};
            n++;a[n]=(data){x2,y2,1,cnt,n};
        }
        else break;
    }
    sort(a+1,a+n+1,cmp);
    solve(1,n);
    for(int i=1;i<=cnt;i++) printf("%d\n",ans[i]);
}
```

2.3 主席树

2.3.1 区间静态第 k 大

```
const int N=1e5+10;
struct HisTree{
    int l,r,sum;
}T[N*40];
vector<int> V;
int R[N],A[N],cnt;
int getid(int x){
    return lower_bound(V.begin(),V.end(),x)-V.begin()+1;
}
void build(int l,int r,int &x,int y,int pos){
    T[++cnt]=T[y],T[cnt].sum++,x=cnt;
    if(l==r)return;
    int mid=(l+r)/2;
    if(pos<=mid)build(l,mid,T[x].l,T[y].l,pos);
    else build(mid+1,r,T[x].r,T[y].r,pos);
}
int ask(int l,int r,int x,int y,int k){
    if(l==r)return l;
    int sum=T[T[y].l].sum-T[T[x].l].sum;
    int mid=(l+r)/2;
```

```
    if(sum>=k)return ask(l,mid,T[x].l,T[y].l,k);
    else return ask(mid+1,r,T[x].r,T[y].r,k-sum);
}
int main(){
    int n,m,x,y,z;
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;++i)scanf("%d",&A[i]),V.push_back(A[i]);
    sort(V.begin(),V.end());
    V.erase(unique(V.begin(),V.end()),V.end());
    for(int i=1;i<=n;++i)build(1,n,R[i],R[i-1],getid(A[i]));
    while(m--){
        scanf("%d%d%d",&x,&y,&z);
        printf("%d\n",V[ask(1,n,R[x-1],R[y],z)-1]);
    }
}
```

2.3.2 树上静态第 k 大

```
const int N=3e5+10;
struct HisTree{
    int l,r,sum;
}T[N*20];
int tot,cnt,ver[N],head[N],Next[N],d[N],fa[N],n,m,R[N],A[N],f[N][20],t;
vector<int> V;
int getid(int x){
    return lower_bound(V.begin(),V.end(),x)-V.begin()+1;
}
void add(int x,int y){
    ver[++tot]=y;Next[tot]=head[x];head[x]=tot;
}
void build(int l,int r,int &x,int y,int pos){
    T[++cnt]=T[y],x=cnt,T[x].sum++;
    if(l==r)return;
    int mid=(l+r)/2;
    if(pos<=mid)build(l,mid,T[x].l,T[y].l,pos);
    else build(mid+1,r,T[x].r,T[y].r,pos);
}
void dfs(int x,int y){
    fa[x]=y;
    for(int i=head[x];i;i=Next[i]){
        if(ver[i]==y)continue;
        build(1,n,R[ver[i]],R[x],getid(A[ver[i]]));
        dfs(ver[i],x);
    }
}
void bfs(){
    queue<int> Q;
    Q.push(1);d[1]=1;
    while(Q.size()){
        int x=Q.front();Q.pop();
        for(int i=head[x];i;i=Next[i]){
            int y=ver[i];
            if(d[y])continue;
            d[y]=d[x]+1;
            f[y][0]=x;
            for(int j=1;j<=t;++j)
                f[y][j]=f[f[y][j-1]][j-1];
            Q.push(y);
```

```
        }
    }
}
int lca(int x,int y){
    if(d[x]>d[y])swap(x,y);
    for(int i=t;i>=0;--i)
        if(d[f[y][i]]>=d[x])y=f[y][i];
    if(x==y)return x;
    for(int i=t;i>=0;--i)
        if(f[x][i]!=f[y][i])x=f[x][i],y=f[y][i];
    return f[x][0];
}
int ask(int l,int r,int x,int y,int z,int Z,int k){
    if(l==r)return l;
    int sum=T[T[x].l].sum+T[T[y].l].sum-T[T[z].l].sum-T[T[Z].l].sum;
    int mid=(l+r)/2;
    if(sum>=k)return ask(l,mid,T[x].l,T[y].l,T[z].l,T[Z].l,k);
    else return ask(mid+1,r,T[x].r,T[y].r,T[z].r,T[Z].r,k-sum);
}
int main(){
    cin>>n>>m;
    t=(int)(log(n)/log(2))+1;
    for(int i=1;i<=n;++i)scanf("%d",&A[i]),V.push_back(A[i]);
    sort(V.begin(),V.end());
    V.erase(unique(V.begin(),V.end()),V.end());
    int x,y,z;
    for(int i=1;i<n;++i)scanf("%d%d",&x,&y),add(x,y),add(y,x);
    build(1,n,R[1],R[0],getid(A[1]));
    dfs(1,0);
    bfs();
    while(m--){
        scanf("%d%d%d",&x,&y,&z);
        printf("%d\n",V[ask(1,n,R[x],R[y],R[fa[lca(x,y)]],R[lca(x,y)],z)-1]);
    }
}
```

2.4 KD 树

2.4.1 K 维点上最近

k 维 n 个点，给定点的最近的 m 个点

```
#include<bits/stdc++.h>
#define SQ(x) (x)*(x)
using namespace std;
const int N=1e5+10;
int idx,k;
struct P{
    int x[5];
    bool operator <(const P &u)const{return x[idx]<u.x[idx];}
}A[N];
typedef pair<int,P> DIS;
priority_queue<DIS>Q;
vector<DIS> V;
struct KDTREE{
    int S[N<<2];P R[N<<2];
    void build(int p,int l,int r,int dep){
        if(l>r)return;
        S[p]=r-l;S[p*2]=S[p*2+1]=-1;
```

```
        idx=dep%k;int mid=(l+r)/2;
        nth_element(A+l,A+mid,A+r+1);
        R[p]=A[mid];
        build(p*2,l,mid-1,dep+1);
        build(p*2+1,mid+1,r,dep+1);
    }
    void query(int p,int m,int dep,P a){
        if(S[p]==-1)return;
        DIS tmp=DIS(0,R[p]);
        for(int i=0;i<k;++i)tmp.first+=SQ(tmp.second.x[i]-a.x[i]);
        int l=p*2,r=p*2+1,dim=dep%k,flag=0;
        if(a.x[dim]>=R[p].x[dim])swap(l,r);
        if(~S[l])query(l,m,dep+1,a);
        if(Q.size()<m)Q.push(tmp),flag=1;
        else{
            if(Q.top().first>tmp.first)Q.pop(),Q.push(tmp);
            if(SQ(R[p].x[dim]-a.x[dim])<Q.top().first)flag=1;
        }
        if(~S[r]&&flag)query(r,m,dep+1,a);
    }
}KDT;
int main(){
    ios::sync_with_stdio(0);cin.tie(0);
    int n,m;
    while(cin>>n>>k){
        for(int i=1;i<=n;++i)for(int j=0;j<k;++j)cin>>A[i].x[j];
        KDT.build(1,1,n,0);
        int T;
        cin>>T;
        while(T--){
            P a;
            for(int i=0;i<k;++i)cin>>a.x[i];
            cin>>m;
            KDT.query(1,m,0,a);
            V.clear();
            while(Q.size()){
                V.push_back(Q.top());
                Q.pop();
            }
            printf("the closest %d points are:\n",V.size());
            for(int i=V.size()-1;i>=0;--i){
                printf("%d",V[i].second.x[0]);
                for(int j=1;j<k;++j)printf(" %d",V[i].second.x[j]);
                printf("\n");
            }
        }
    }
}
```

2.4.2 查询直线上点数量

n 个点，m 个直线，问每条直线上有几个点

```
#include<cstdio>
#include<algorithm>
const int N=100010;
using namespace std;
typedef long long ll;
```

```
int Case,n,m,i,root,cmp_d,ans,A,B;
ll LB,LA,LC;
struct node{
    int d[2];
    int l,r;
    int Max[2],Min[2];
    int sum;
}t[N];
inline bool cmp(const node&a,const node&b){
    return a.d[cmp_d]<b.d[cmp_d];
}
inline void umax(int&a,int b){if(a<b)a=b;}
inline void umin(int&a,int b){if(a>b)a=b;}
inline void up(int x){
    if(t[x].l){
        umax(t[x].Max[0],t[t[x].l].Max[0]);
        umin(t[x].Min[0],t[t[x].l].Min[0]);
        umax(t[x].Max[1],t[t[x].l].Max[1]);
        umin(t[x].Min[1],t[t[x].l].Min[1]);
    }
    if(t[x].r){
        umax(t[x].Max[0],t[t[x].r].Max[0]);
        umin(t[x].Min[0],t[t[x].r].Min[0]);
        umax(t[x].Max[1],t[t[x].r].Max[1]);
        umin(t[x].Min[1],t[t[x].r].Min[1]);
    }
}
int build(int l,int r,int D){
    int mid=(l+r)>>1;
    cmp_d=D;
    nth_element(t+l+1,t+mid+1,t+r+1,cmp);
    t[mid].Max[0]=t[mid].Min[0]=t[mid].d[0];
    t[mid].Max[1]=t[mid].Min[1]=t[mid].d[1];
    t[mid].sum=1;
    if(l!=mid)t[mid].l=build(l,mid-1,!D);else t[mid].l=0;
    if(r!=mid)t[mid].r=build(mid+1,r,!D);else t[mid].r=0;
    up(mid);
    return mid;
}
inline bool check(int xl,int xr,int yl,int yr){
    ll t=-LB*xl+LC;
    if(LA*yl<=t&&t<=LA*yr)return 1;
    t=-LB*xr+LC;
    if(LA*yl<=t&&t<=LA*yr)return 1;
    t=-LA*yl+LC;
    if(LB*xl<=t&&t<=LB*xr)return 1;
    t=-LA*yr+LC;
    if(LB*xl<=t&&t<=LB*xr)return 1;
    return 0;
}
void ask(int x){
    if(!check(t[x].Min[0],t[x].Max[0],t[x].Min[1],t[x].Max[1]))return;
    if(LB*t[x].d[0]+LA*t[x].d[1]==LC)ans++;
    if(t[x].l)ask(t[x].l);
    if(t[x].r)ask(t[x].r);
}
int main(){
    scanf("%d",&Case);
    while(Case--){
```

```
65          scanf("%d%d",&n,&m);
66          for(i=1;i<=n;i++)scanf("%d%d",&t[i].d[0],&t[i].d[1]);
67          root=build(1,n,0);
68          while(m--){
69              scanf("%d%d",&A,&B);//(a,0) (0,b)
70              LA=A;
71              LB=B;
72              LC=LA*LB;
73              ans=0;
74              ask(root);
75              printf("%d\n",ans);
76          }
77      }
78  }
```

2.5 莫队算法

```
1   struct Quary{
2       int l,r,num;
3       bool operator < (const Quary node) const{//奇偶优化
4           return (pos[l] < pos[node.l])||(pos[l] == pos[node.l] && (pos[l] & 1 ? r < node
        .r : r > node.r));
5       }
6   }quary[N];
7   int pos[N],ans[N],block;
8   void add(int x)
9
10  void del(int x)
11
12  int main(){
13      int n,m;
14      while(scanf("%d%d", &n, &m) == 2){
15          block = sqrt(n);
16          for(int i = 1; i <= n; i ++){
17              scanf("%d", &ma[i]);
18              pos[i] = i / block;
19          }
20          for(int i = 1; i <= m; i ++){
21              int num, l, r;
22              scanf("%d%d",&quary[i].l,&quary[i].r);
23              quary[i].num = i;
24          }
25          sort(quary + 1, quary + m + 1);
26          l = 1;
27          r = 0;
28          for(int i = 1; i <= m; i ++){
29              while(l > quary[i].l){
30                  l --;
31                  add(l);
32              }
33              while(r < quary[i].r){
34                  r ++;
35                  add(r);
36              }
37              while(l < quary[i].l){
38                  del(l);
39                  l ++;
40              }
```

```
            while(r > quary[i].r){
                del(r);
                r --;
            }
            ans[quary[i].num] = ask(quary[i]);
        }
    }
}
```

2.6 DSU

计算每个点的所有子树中颜色出现最多的颜色
如果颜色出现次数一样，就颜色累加

```
#include<bits/stdc++.h>
using namespace std;
#define X first
#define Y second
typedef long long ll;
const int N=2e5+10;ll ans[N];
int A[N],ver[N],Next[N],head[N],dsu[N],mx[N],sz[N],tot,t;
unordered_map<int,int> a[N];
void add(int x,int y){
    ver[++tot]=y;Next[tot]=head[x];head[x]=tot;
}
void dfs1(int x=1,int y=-1){
    sz[x]=1;
    for(int i=head[x];i;i=Next[i])
        if(ver[i]!=y)dfs1(ver[i],x),sz[x]+=sz[ver[i]];
}
void relax(int x,int y,int z){
    if(z==mx[x])ans[x]+=y;
    if(z>mx[x])ans[x]=y,mx[x]=z;
}
void mrg(int x,int y,int z){
    for(auto& i : a[z]){
        a[y][i.X]+=i.Y;
        relax(x,i.X,a[y][i.X]);
    }
}
void dfs2(int x=1,int y=-1){
    if(sz[x]==1){
        dsu[x]=++t;a[t][A[x]]=1;mx[x]=1;ans[x]=A[x];return;
    }
    int big=0,ma=0;
    for(int i=head[x];i;i=Next[i]){
        if(ver[i]!=y){
            dfs2(ver[i],x);
            if(sz[ver[i]]>ma)ma=sz[ver[i]],big=ver[i];
        }
    }
    dsu[x]=dsu[big];mx[x]=mx[big];ans[x]=ans[big];
    for(int i=head[x];i;i=Next[i]){
        if(ver[i]!=big&&ver[i]!=y){
            mrg(x,dsu[x],dsu[ver[i]]);
        }
    }
    a[dsu[x]][A[x]]++;relax(x,A[x],a[dsu[x]][A[x]]);
```

```
}
int main(){
    ios::sync_with_stdio(0);
    int n,x,y;cin>>n;
    for(int i=1;i<=n;++i)cin>>A[i];
    for(int i=1;i<n;++i)cin>>x>>y,add(x,y),add(y,x);
    dfs1();dfs2();
    for(int i=1;i<=n;++i)cout<<ans[i]<<" ";
}
```

2.7 树链剖分

题意：给一棵树，并给定各个点权的值，然后有 3 种操作：
I D C1 C2 K: 把 C1 与 C2 的路径上的所有点权值加减 K
Q C：查询节点编号为 C 的权值

```
#include<bits/stdc++.h>
using namespace std;
const int N=3e5+100;
int head[N],Next[N],ver[N],siz[N],son[N],
c[N],dp[N],fa[N],dep[N],pos[N],top[N],A[N];
int tot,sz;
void add(int x,int y){
    ver[++tot]=y;
    Next[tot]=head[x];
    head[x]=tot;
}
void dfs1(int x){
    son[x]=0,siz[x]=1;
    for(int i=head[x];i;i=Next[i]){
        int y=ver[i];
        if(y==fa[x]) continue;
        fa[y]=x;
        dep[y]=dep[x]+1;
        dfs1(y);
        siz[x]+=siz[y];
        if(siz[y]>siz[son[x]]) son[x]=y;
    }
}
void dfs2(int x,int tp){
    pos[x]=++sz;
    top[x]=tp;
    if(son[x]) dfs2(son[x],top[x]);
    for(int i=head[x];i;i=Next[i]){
        int y=ver[i];
        if(y!=son[x]&&y!=fa[x])
            dfs2(y,y);
    }
}
int lowbit(int x){
    return x&(-x);
}
void Add(int x,int val){
    for(;x<=N;x+=lowbit(x)) c[x]+=val;
}
int Query(int x){
    int sum=0;
    for(;x;x-=lowbit(x)) sum+=c[x];
```

```
    return sum;
}
void update(int x,int y,int z){
    Add(x,z);
    Add(y+1,-z);
}
void change(int x,int y,int z){
    while(top[x]!=top[y]){
        if(dep[top[x]]<dep[top[y]]) swap(x,y);
        update(pos[top[x]],pos[x],z);
        x=fa[top[x]];
    }
    if(dep[x]>dep[y]) swap(x,y);
    update(pos[x],pos[y],z);
}
int main(){
    int n,m,q;
    while(cin>>n>>m>>q){
        for(int i=0;i<N;++i){
            head[i]=Next[i]=ver[i]=siz[i]=son[i]=
        c[i]=dp[i]=fa[i]=dep[i]=pos[i]=top[i]=A[i]=0;
        }
        tot=sz=0;
        for(int i=1;i<=n;++i) scanf("%d",&A[i]);
        int u,v,k;
        while(m--){
            scanf("%d%d",&u,&v);
            add(u,v);
            add(v,u);
        }
        dfs1(1);
        dfs2(1,1);
        string s;
        while(q--){
            cin>>s;
            if(s[0]=='I'){
                scanf("%d%d%d",&u,&v,&k);
                change(u,v,k);
            }else if(s[0]=='D'){
                scanf("%d%d%d",&u,&v,&k);
                change(u,v,-k);
            }else{
                scanf("%d",&k);
                cout<<Query(pos[k])+A[k]<<endl;
            }
        }
    }
}
```

2.8 笛卡尔树

```
const int maxn = "Edit";
int lson[maxn], rson[maxn], fa[maxn];
void build(int n){
    stack<int> s;
    for (int i = 0; i < n; i++){
        int last = -1;
        while (!s.empty() && a[i] > a[s.top()]) last = s.top(), s.pop();
```

```
        if (!s.empty()) rson[s.top()] = i, fa[i] = s.top();
        lson[i] = last;
        if (~last) fa[last] = i;
        s.push(i);
    }
}
```

2.9 字典树

```
struct Trie{
    int T[N*10][2],tot;
    void init(){
        tot=1;
        memset(T,0,sizeof T);
        insert(0);
    }
    void insert(int x){
        int p=1;
        for(int i=1<<30;i;i>>=1){
            int y=(x&i)?1:0;
            if(!T[p][y]) T[p][y]=++tot;
            p=T[p][y];
        }
    }
    int find(int x){
        int p=1,ans=0;
        for(int i=1<<30;i;i>>=1){
            int y=(x&i)?0:1;
            if(T[p][y])ans+=i,p=T[p][y];
            else p=T[p][!y];
        }
        return ans;
    }
}T;
```

2.10 Sparse Table

```
const int N=;
struct SparseTable{
    int f[N][20];
    void build(int n){
        for(int i=1;i<=n;++i)f[i][0]=A[i];
        for(int j=1;(1<<j)<=n;++j)
            for(int i=1;i+(1<<j)-1<=n;++i)
                f[i][j]=max(f[i][j-1],f[i+(1<<(j-1))][j-1]);
    }
    int ask(int l,int r){
        int k=31-__bulitin_clz(r-l+1);
        return max(f[l][k],f[r-(1<<k)+1][k]);
    }
}ST;
```

3 Computational Geometry

3.1 My Geo Template

```
#include<bits/stdc++.h>
using namespace std;
typedef double db;
const db eps = 1e-15, pi = acos(-1);
int sign(db x) {return x < -eps ? -1 : x > eps;}
int cmp(db x, db y) {return sign(x - y);}
int intersect(db l1, db r1, db l2, db r2) {
    if (l1 > r1) swap(l1, r1); if (l2 > r2) swap(l2, r2); return cmp(r1, l2) != -1 &&
    cmp(r2, l1) != -1;
}
int inmid(db k1, db k2, db k3) {return sign(k1 - k3) * sign(k2 - k3) <= 0;}//k3 in [k1,
    k2]?1:0
struct Point {
    db x, y;
    Point operator + (const Point & a)const {return Point{a.x + x, a.y + y};}
    Point operator - (const Point & a)const {return Point{x - a.x, y - a.y};}
    Point operator * (db a) const {return Point{x * a, y * a};}
    Point operator / (db a) const {return Point{x / a, y / a};}
    bool operator < (const Point p) const {int a = cmp(x, p.x); if (a) return a == -1;
    return cmp(y, p.y) == -1;}
    bool operator == (const Point & a) const {return cmp(x, a.x) == 0 && cmp(y, a.y) ==
     0;}
    db abs() {return sqrt(x * x + y * y);}
    db abs2() {return x * x + y * y;}
    db dis(Point p) {return ((*this) - p).abs();}
    db dis2(Point p) {return ((*this) - p).abs2();}
    Point turn90() {return (Point) { -y, x};}
    Point unit() {db w = abs(); return (Point) {x / w, y / w};}
    int getP() const {return sign(y) == 1 || (sign(y) == 0 && sign(x) == -1);}
    void input() {scanf("%lf%lf", &x, &y);}
};
typedef vector<Point> VP;
db cross(Point p1, Point p2) {return p1.x * p2.y - p1.y * p2.x;}
db cross(Point p0, Point p1, Point p2) {return cross(p1 - p0, p2 - p0);}
db dot(Point p1, Point p2) {return p1.x * p2.x + p1.y * p2.y;}
db rad(Point p1, Point p2) {return atan2(cross(p1, p2), dot(p1, p2));}
int inmid(Point k1, Point k2, Point k3) {return inmid(k1.x, k2.x, k3.x) && inmid(k1.y,
    k2.y, k3.y);}
bool compareangle(Point p1, Point p2) {//Polar Angle Sort
    return p1.getP() < p2.getP() || (p1.getP() == p2.getP() && sign(cross(p1, p2)) > 0)
    ;
}
int clockwise(Point p1, Point p2, Point Point3) { // p1 p2 Point3 anticlockwise:1
    clockwise:-1 others:0
    return sign(cross(p1, p2, Point3));
}
struct Line {
    Point s, e;
    void input() {scanf("%lf%lf%lf%lf", &s.x, &s.y, &e.x, &e.y);}
    Point vec() {return e - s;}
    Point unit() {return vec() / length();}
    db length() {return sqrt(dot(s - e, s - e));}
    db length2() {return dot(s - e, s - e);}
};
int onS(Line l, Point p) {// On Seg?
```

```
49      return inmid(l.s, l.e, p) && sign(cross(l.s - p, l.e - l.s)) == 0;
50  }
51  bool checkLL(Line l1, Line l2) {
52      return cmp(cross(l1.s, l2.s, l2.e), cross(l1.e, l2.s, l2.e)) != 0;
53  }
54  bool checkLS(Line l1, Line l2) {//Intersection of Line l1 and Seg l2?
55      return sign(cross(l2.s, l1.s, l1.e)) * sign(cross(l2.e, l1.s, l1.e)) <= 0;
56  }
57  int checkSS(Line l1, Line l2) {//Intersection of Two Seg?1:0
58      return intersect(l1.s.x, l1.e.x, l2.s.x, l2.e.x) && intersect(l1.s.y, l1.e.y, l2.s.
        y, l2.e.y) && checkLS(l1, l2) && checkLS(l2, l1);
59  }
60  Point project(Line l, Point p) {
61      return l.s + l.vec() * dot(p - l.s, l.vec()) / l.length2();
62  }
63  Point reflect(Line l, Point p) {//Mirror Point
64      return project(l, p) * 2 - p;
65  }
66  Point getLL(Line l1, Line l2) {//Intersection Point of Line l1,l2
67      db w1 = cross(l2.s, l1.s, l2.e), w2 = cross(l2.s, l2.e, l1.e); return (l1.s * w2 +
        l1.e * w1) / (w1 + w2);
68  }
69  db disSP(Line l, Point p) {
70      Point p2 = project(l, p);
71      if (inmid(l.s, l.e, p2)) return p.dis(p2); else return min(p.dis(l.s), p.dis(l.e));
72  }
73  db disSS(Line l1, Line l2) {
74      if (checkSS(l1, l2)) return 0;
75      return min(min(disSP(l1, l2.s), disSP(l1, l2.e)), min(disSP(l2, l1.s), disSP(l2, l1
        .e)));
76  }
77  db area(vector<Point> A) {//Anticlockwise
78      db ans = 0;
79      for (int i = 0; i < A.size(); i++) ans += cross(A[i], A[(i + 1) % A.size()]);
80      return ans / 2;
81  }
82  int contain(VP A, Point p) {//2:in 1:on 0:out
83      int ans = 0; A.push_back(A[0]);
84      for (int i = 1; i < A.size(); i++) {
85          Line l = {A[i - 1], A[i]};
86          if (onS(l, p)) return 1; if (cmp(l.s.y, l.e.y) > 0) swap(l.s, l.e);
87          if (cmp(l.s.y, p.y) >= 0 || cmp(l.e.y, p.y) < 0) continue;
88          if (sign(cross(l.e, l.s, p)) < 0)ans ^= 1;
89      }
90      return ans << 1;
91  }
92  bool checkConvex(VP A) { //anticlock
93      int n = A.size(); A.push_back(A[0]); A.push_back(A[1]);
94      for (int i = 0; i < n; i++) if (sign(cross(A[i], A[i + 1], A[i + 2])) == -1) return
         0;
95      return 1;
96  }
97  VP ConvexHull(VP A, int flag = 1) { // flag=0 不严格 flag=1 严格
98      int n = A.size(); VP ans(n * 2);
99      sort(A.begin(), A.end()); int now = 0; if (n <= 1)return A;
100     for (int i = 0; i < n; ans[now++] = A[i++])
101         while (now > 1 && sign(cross(ans[now - 2], ans[now - 1], A[i])) < flag)--now;
102     for (int i = n - 2, pre = now; i >= 0; ans[now++] = A[i--])
103         while (now > pre && sign(cross(ans[now - 2], ans[now - 1], A[i])) < flag)--now;
```

```
104     ans.resize(now - 1); return ans;
105 }
106 db ConvexDiameter(VP A) {
107     int n = A.size(); if (n <= 1)return 0;
108     int is = 0, js = 0; for (int k = 1; k <= n; ++k)is = A[k] < A[is] ? k : is, js = A[
        js] < A[k] ? k : js;
109     int i = is, j = js; db ret = A[i].dis(A[j]); do {
110         if (sign(cross(A[(i + 1) % n] - A[i], A[(j + 1) % n] - A[j])) >= 0)(++j) %= n;
111         else (++i) %= n;
112         ret = max(ret, A[i].dis(A[j]));
113     } while (i != is || j != js);
114     return ret;
115 }
116 VP ConvexCut(VP A, Line l) { // 保留 k1,k2,p 逆时针的所有点,判断n是否为0
117     int n = A.size(); A.push_back(A[0]); VP ans;
118     for (int i = 0; i < n; i++) {
119         int w1 = clockwise(l.s, l.e, A[i]), w2 = clockwise(l.s, l.e, A[i + 1]);
120         if (w1 >= 0) ans.push_back(A[i]);
121         if (w1 * w2 < 0) ans.push_back(getLL(l, Line{A[i], A[i + 1]}));
122     }
123     return ans;
124 }
125 struct Circle {
126     Point o; db r;
127     void input() {o.input(); scanf("%lf", &r);}
128     int inside(Point k) {return cmp(r, o.dis(k));}
129 };
130 int checkCC(Circle c1, Circle c2) {// 返回两个圆的公切线数量
131     db d = c1.o.dis(c2.o); if (cmp(d, c1.r + c2.r) == 1)return 4;
132     if (cmp(d, c1.r + c2.r) == 0)return 3; if (cmp(d, abs(c1.r - c2.r)) == 1)return 2;
133     if (cmp(d, abs(c1.r - c2.r)) == 0)return 1; return 0;
134 }
135 vector<Point> getCL(Circle c, Line l) {// 沿着 s->e 方向给出 , 相切给出两个
136     Point p = project(l, c.o); db d = c.r * c.r - p.dis2(c.o);
137     if (sign(d) == -1)return{};
138     Point del = l.vec() / l.length() * sqrt(max(db(0.0), d)); return {p - del, p + del
        };
139 }
140 vector<Point> getCC(Circle c1, Circle c2) { // 沿圆 c1 逆时针给出 , 相切给出两个
141     int pd = checkCC(c1, c2); if (pd == 0 || pd == 4) return {};
142     db a = c1.o.dis2(c2.o), cosA = (c1.r * c1.r + a - c2.r * c2.r) / (2 * c1.r * sqrt(
        max(a, (db)0.0)));
143     db b = c1.r * cosA, c = sqrt(max((db)0.0, c1.r * c1.r - b * b));
144     Point k = Line{c1.o, c2.o} .unit(), m = c1.o + k * b, del = k.turn90() * c;
145     return {m - del, m + del};
146 }
147 vector<Point> TangentCP(Circle c, Point p) {
148     db x = c.o.dis2(p), d = x - c.r * c.r; if (sign(d) < 0)return{};
149     Point p1 = c.o + (p - c.o) * (c.r * c.r / x), p2 = (p - c.o).turn90() * (c.r * sqrt
        (d) / x);
150     return {p1 - p2, p1 + p2};//counter clock-wise
151 }
152 db areaCT(Circle c, Line l) { // 圆与有向三角形的面积交
153     l.s = l.s - c.o, l.e = l.e-c.o, c.o = c.o - c.o;
154     int pd1 = c.inside(l.s), pd2 = c.inside(l.e);
155     vector<Point>A = getCL(c, l);
156     if (pd1 >= 0) {
157         if (pd2 >= 0) return cross(l.s, l.e) / 2;
158         return c.r * c.r * rad(A[1], l.e) / 2 + cross(l.s, A[1]) / 2;
```

```
159     } else if (pd2 >= 0) {
160         return c.r * c.r * rad(l.s, A[0]) / 2 + cross(A[0], l.e) / 2;
161     } else {
162         int pd = cmp(c.r, disSP(l, c.o));
163         if (pd <= 0) return c.r * c.r * rad(l.s, l.e) / 2;
164         return cross(A[0], A[1]) / 2 + c.r * c.r * (rad(l.s, A[0]) + rad(A[1], l.e)) /
    2;
165     }
166 }
167 Circle getcircle(Point k1, Point k2, Point k3) { //检查是否共线
168     db a1 = k2.x - k1.x, b1 = k2.y - k1.y, c1 = (a1 * a1 + b1 * b1) / 2;
169     db a2 = k3.x - k1.x, b2 = k3.y - k1.y, c2 = (a2 * a2 + b2 * b2) / 2;
170     db d = a1 * b2 - a2 * b1;
171     Point o = (Point) {k1.x + (c1 * b2 - c2 * b1) / d, k1.y + (a1 * c2 - a2 * c1) / d};
172     return (Circle) {o, k1.dis(o)};
173 }
174 Circle minC(vector<Point> A) {
175     random_shuffle(A.begin(), A.end()); Circle ans = (Circle) {A[0], 0};
176     for (int i = 1; i < A.size(); i++)
177         if (ans.inside(A[i]) == -1) {
178             ans = (Circle) {A[i], 0};
179             for (int j = 0; j < i; j++)
180                 if (ans.inside(A[j]) == -1) {
181                     ans.o = (A[i] + A[j]) / 2; ans.r = ans.o.dis(A[i]);
182                     for (int k = 0; k < j; k++)
183                         if (ans.inside(A[k]) == -1)ans = getcircle(A[i], A[j], A[k]);
184                 }
185         }
186     return ans;
187 }
188 struct Point3 {
189     db x, y, z;
190     Point3 operator + (Point3 a) {return (Point3) {x + a.x, y + a.y, z + a.z};}
191     Point3 operator - (Point3 a) {return (Point3) {x - a.x, y - a.y, z - a.z};}
192     Point3 operator * (db a) {return (Point3) {x*a, y*a, z*a};}
193     Point3 operator / (db a) {return (Point3) {x / a, y / a, z / a};}
194     db abs2() {return x * x + y * y + z * z;}
195     db abs() {return sqrt(x * x + y * y + z * z);}
196     db dis(Point3 a) {return ((*this) - a).abs();}
197     void input() {scanf("%lf%lf%lf", &x, &y, &z);}
198 };
199 Point3 cross(Point3 k1, Point3 k2) {
200     return (Point3) {k1.y*k2.z - k1.z*k2.y, k1.z*k2.x - k1.x*k2.z, k1.x*k2.y - k1.y*k2.
    x};
201 }
202 db dot(Point3 k1, Point3 k2) {return k1.x * k2.x + k1.y * k2.y + k1.z * k2.z;}
203 db rand_db() {return 1.0 * rand() / RAND_MAX;}
204 typedef vector<Point3> VP3; typedef vector<VP3> VVP3;
205 db minSphere(VP3 A) {
206     double step = 10000, ans = 1e30, mt;
207     Point3 p = Point3{0, 0, 0}; int s = 0;
208     while (step > eps) {
209         for (int i = 0; i < A.size(); i++)if (p.dis(A[s]) < p.dis(A[i]))s = i;
210         mt = p.dis(A[s]); ans = min(ans, mt);
211         p = p + (A[s] - p) / mt * step; step *= 0.98;
212     }
213     return ans;
214 }
215 db getV(Point3 k1, Point3 k2, Point3 k3, Point3 k4) { // get the Volume
```

```
216     return dot(cross(k2 - k1, k3 - k1), k4 - k1);
217 }
218 namespace CH3 {
219     VVP3 ret; set<pair<int, int> >e;
220     int n; VP3 p, q;
221     void wrap(int a, int b) {
222         if (e.find({a, b}) == e.end()) {
223             int c = -1;
224             for (int i = 0; i < n; i++) if (i != a && i != b) {
225                     if (c == -1 || sign(getV(q[c], q[a], q[b], q[i])) > 0) c = i;
226                 }
227             if (c != -1) {
228                 ret.push_back({p[a], p[b], p[c]});
229                 e.insert({a, b}); e.insert({b, c}); e.insert({c, a});
230                 wrap(c, b); wrap(a, c);
231             }
232         }
233     }
234     VVP3 ConvexHull3D(VP3 _p) {
235         p = q = _p; n = p.size();
236         ret.clear(); e.clear();
237         for (auto &i : q) i = i + (Point3) {rand_db() * 1e-4, rand_db() * 1e-4, rand_db
    () * 1e-4};
238         for (int i = 1; i < n; i++) if (q[i].x < q[0].x) swap(p[0], p[i]), swap(q[0], q
    [i]);
239         for (int i = 2; i < n; i++)
240             if ((q[i].x - q[0].x) * (q[1].y - q[0].y) > (q[i].y - q[0].y) * (q[1].x - q
    [0].x))
241                 swap(q[1], q[i]), swap(p[1], p[i]);
242         wrap(0, 1);
243         return ret;
244     }
245 }
246 db volume(VVP3 A) {
247     if (A.size() == 0) return 0; Point3 p = A[0][0]; db ans = 0;
248     for (VP3 nowF : A)
249         for (int i = 2; i < nowF.size(); i++)
250             ans += abs(getV(p, nowF[0], nowF[i - 1], nowF[i]));
251     return ans / 6;
252 }
253 db area(VP3 A){
254     if(A.size()<3)return 0;db ans=0;
255     for(int i=2;i<A.size();++i)ans+=cross(A[i-1]-A[0],A[i]-A[0]).abs();
256     return abs(ans/2);
257 }
258 db surface_area(VVP3 A){
259     if(A.size()==0)return 0;db ans=0;
260     for(VP3 i:A)ans+=area(i);return ans;
261 }
```

4 Graph Thoery

4.1 Network Flow

4.1.1 DINIC

```
#include<bits/stdc++.h>
using namespace std;
const int N=,M=,inf=1<<30;
struct DINIC{
    int Next[M],d[N],head[N],ver[M],edge[M],tot,S,T;
    queue<int> q;
    void add(int x,int y,int z){
        ver[++tot]=y,edge[tot]=z,Next[tot]=head[x],head[x]=tot;
        ver[++tot]=x,edge[tot]=0,Next[tot]=head[y],head[y]=tot;
    }
    void build(){
        tot=1,S=N-2,T=N-1,ans=0;
    }
    bool bfs(){
        memset(d,0,sizeof d);
        while(q.size())q.pop();
        q.push(S);d[S]=1;
        while(q.size()){
            int x=q.front();q.pop();
            for(int i=head[x];i;i=Next[i])
                if(edge[i]&&!d[ver[i]]){
                    q.push(ver[i]);
                    d[ver[i]]=d[x]+1;
                    if(ver[i]==T)return 1;
                }
        }
        return 0;
    }
    int find(int x,int flow){
        if(x==T)return flow;
        int rest=flow,k;
        for(int i=head[x];i&&rest;i=Next[i])
            if(edge[i]&&d[ver[i]]==d[x]+1){
                k=find(ver[i],min(rest,edge[i]));
                if(!k)d[ver[i]]=0;
                edge[i]-=k;
                edge[i^1]+=k;
                rest-=k;
            }
        return flow-rest;
    }
    int dinic(){
        int flow=0,maxflow=0;
        while(bfs())
            while(flow=find(S,inf))maxflow+=flow;
        return maxflow;
    }
}G;
```

5 String

5.1 Palindromic$_A$utomaton

```
struct Palindromic_Automaton {
    int ch[N][26], f[N], len[N], s[N], ok[N];
    int cnt[N]; // 结点表示的本质不同的回文串的个数(调用count()后)
    int num[N]; // 结点表示的最长回文串的最右端点为回文串结尾的回文串个数
    int last, sz, n;
    int newnode(int x) {
        memset(ch[sz], 0, sizeof(ch[sz]));
        cnt[sz] = num[sz] = 0, len[sz] = x;
        return sz++;
    }
    void init() {
        sz = 0; newnode(0), newnode(-1); last = n = 0, s[0] = -1, f[0] = 1;
    }
    int get(int u) {for (; s[n - len[u] - 1] != s[n]; u = f[u]); return u;}
    void add(int c) {
        s[++n] = c;
        int u = get(last);
        if (!ch[u][c]) {
            int np = newnode(len[u] + 2);
            f[np] = ch[get(f[u])][c];
            num[np] = num[f[np]] + 1;
            ch[u][c] = np;
            ok[np]=check(n-len[np]+1,n);
        }
        last = ch[u][c];
        cnt[last]++;
    }
    ll count(){
        ll ans=0;
        for (int i = sz - 1; ~i; i--) cnt[f[i]] += cnt[i];
        for(int i=sz-1;i>1;--i)ans=ans+1ll*cnt[i]*ok[i];
        return ans;
    }
}pam;
int main(){
    pam.init();
    for(int i=0;i<n;++i)pam.add(s[i]-'a');
    cout<<pam.count()<<endl;
}
```

5.2 Aho$_C$orasick$_A$utomaton

```
int ans[N];
struct Aho_Corasick_Automaton {
    int ch[N][26], f[N], val[N], sz, rt;
    int newnode() { memset(ch[sz], -1, sizeof(ch[sz])), val[sz] = 0; return sz++; }
    void init() { sz = 0, rt = newnode(); }
    inline int idx(char c) { return c - 'a'; };
    int ver[N],Next[N],head[N],tot;
    void add(int x,int y){ver[++tot]=y;Next[tot]=head[x];head[x]=tot;}
    void dfs(int x){
        for(int i=head[x];i;i=Next[i]){
            dfs(ver[i]);val[x]+=val[ver[i]];
        }
```

```
13      }
14      void insert(const char* s,int pos) {
15          int u = 0;
16          for (int i = 0; s[i]; i++) {
17              int c = idx(s[i]);
18              if (ch[u][c] == -1) ch[u][c] = newnode();
19              u = ch[u][c];
20          }
21          ans[pos]=u;
22      }
23      void build() {
24          queue<int> q; f[rt] = rt;
25          for (int c = 0; c < 26; c++) {
26              if (~ch[rt][c])f[ch[rt][c]] = rt, add(rt,ch[rt][c]), q.push(ch[rt][c]);
27              else ch[rt][c] = rt;
28          }
29          while (!q.empty()) {
30              int u = q.front();
31              q.pop();
32              for (int c = 0; c < 26; c++) {
33                  if (~ch[u][c]) f[ch[u][c]] = ch[f[u]][c], add(ch[f[u]][c],ch[u][c]), q.
    push(ch[u][c]);
34                  else ch[u][c] = ch[f[u]][c];
35              }
36          }
37      }
38      int query(const char* s) {
39          int u = rt;
40          for (int i = 0; s[i]; i++) {
41              int c = idx(s[i]); u = ch[u][c];
42              val[u]++;
43          }
44          return 0;
45      }
46
47  } aca;
48  char s[N];
49  int main(){
50      int n;while(cin>>n&&n){
51          aca.init();memset(ans,0,sizeof ans);
52          for(int i=1;i<=n;++i)scanf("%s",s),aca.insert(s,i);
53          aca.build();scanf("%s",s);aca.query(s);
54          aca.dfs(aca.rt);
55          for(int i=1;i<=n;++i)printf("%d\n",aca.val[ans[i]]);
56      }
57  }
```

6 Others

6.1 Language

6.1.1 Language

1. nth_element(first,nth,last) 将第 n_th 元素放到它该放的位置上，左边元素都小于它，右边元素都大于它.
2. Prev_permutation and Next_permutation(begin,end) 生成排列

6.1.2 Policy-Based Data Structures

红黑树

声明/头文件

```
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
typedef tree<pt, null_type, less<pt>, rb_tree_tag, tree_order_statistics_node_update>
    rbtree;
```

使用方法

```
pt                                  // 关键字类型
null_type                           // 无映射(低版本g++为null_mapped_type)
less<int>                           // 从小到大排序
rb_tree_tag                         // 红黑树 (splay_tree_tag)
tree_order_statistics_node_update   // 结点更新
T.insert(val);                      // 插入
T.erase(iterator);                  // 删除
T.order_of_key();                   // 查找有多少数比它小
T.find_by_order(k);                 // 有k个数比它小的数是多少
a.join(b);                          // b并入a 前提是两棵树的key的取值范围不相交
a.split(v, b);                      // key小于等于v的元素属于a，其余的属于b
T.lower_bound(x);                   // >=x的min的迭代器
T.upper_bound((x);                  // >x的min的迭代器
```

可并堆

声明/头文件

```
#include <ext/pb_ds/priority_queue.hpp>
using namespace __gnu_pbds;
typedef __gnu_pbds::priority_queue <int, greater<int>, pairing_heap_tag> Heap;
```

使用方法

```
erase(iterator) 根据迭代器删除元素
modify(iterator, val) 根据迭代器修改值
join(other), 使用之后另一个会被清空
其他同STL
```

可持久化平衡树

声明/头文件

```
#include <ext/rope>
using namespace __gnu_cxx;
```

使用方法

```
下标从0开始，不可以cin，可以cout
由于rope的底层实现，insert，erase，get都是logn的
reverse是O(n)的，所以构造两个rope来做
push_back(x)    在末尾添加x
insert(pos,x)   在pos插入x
erase(pos,x)    从pos开始删除x个
replace(pos,x)  从pos开始换成x
substr(pos,x)   提取pos开始x个
at(x)/[x]   访问第x个元素
rope<int>*his[maxn], his[i]=new rope<char>(*his[i-1]) 可持久化数组
```

6.2 Tricks

6.2.1 读入挂

```
inline void read(int &x){char ch;bool ok;
for(ok=0,ch=getchar();!isdigit(ch);ch=getchar()) if(ch=='-') ok=1;
for(x=0;isdigit(ch);x=x*10+ch-'0',ch=getchar());if(ok) x=-x;}
```

6.2.2 读入神挂

```
namespace IO {
    const int MX = 4e7; //1e7 占用内存 11000kb
    char buf[MX]; int c, sz;
    void Begin() {
        c = 0;
        sz = fread(buf, 1, MX, stdin);//一次性全部读入
    }
    inline bool Read(int &t) {
        while (c < sz && buf[c] != '-' && (buf[c] < '0' || buf[c] > '9')) c++;
        if (c >= sz) return false;//若读完整个缓冲块则退出
        bool flag = 0; if(buf[c] == '-') flag = 1, c++;
        for(t = 0; c < sz && '0' <= buf[c] && buf[c] <= '9'; c++) t = t * 10 + buf[c] -
     '0';
        if(flag) t = -t;
        return true;
    }
}
```

6.2.3 手动扩栈

```
#pragma comment(linker, "/STACK:1024000000,1024000000")
```

6.3 BigNum

```
// 加法 乘法 小于号 输出
struct bint
{
    int l;
    short int w[100];
    bint(int x = 0)
    {
        l = x == 0, memset(w, 0);
        while (x) w[l++] = x % 10, x /= 10;
    }
```

```
11      bool operator<(const bint& x) const
12      {
13          if (l != x.l) return l < x.l;
14          int i = l - 1;
15          while (i >= 0 && w[i] == x.w[i]) i--;
16          return (i >= 0 && w[i] < x.w[i]);
17      }
18      bint operator+(const bint& x) const
19      {
20          bint ans;
21          ans.l = l > x.l ? l : x.l;
22          for (int i = 0; i < ans.l; i++)
23          {
24              ans.w[i] += w[i] + x.w[i];
25              ans.w[i + 1] += ans.w[i] / 10;
26              ans.w[i] = ans.w[i] % 10;
27          }
28          if (ans.w[ans.l] != 0) ans.l++;
29          return ans;
30      }
31      bint operator*(const bint& x) const
32      {
33          bint res;
34          int up, tmp;
35          for (int i = 0; i < l; i++)
36          {
37              up = 0;
38              for (int j = 0; j < x.l; j++)
39              {
40                  tmp = w[i] * x.w[j] + res.w[i + j] + up;
41                  res.w[i + j] = tmp % 10;
42                  up = tmp / 10;
43              }
44              if (up != 0) res.w[i + x.l] = up;
45          }
46          res.l = l + x.l;
47          while (res.w[res.l - 1] == 0 && res.l > 1) res.l--;
48          return res;
49      }
50      void print()
51      {
52          for (int i = l - 1; ~i; i--) printf("%d", w[i]);
53          puts("");
54      }
55  };
```