# 其他东西

目录

[其他东西 1](#_Toc525659840)

[杜教多项式插值 1](#_Toc525659841)

[求x^2+y^2=n的(x,y)对数 2](#_Toc525659842)

[AC自动机 另一种写法 2](#_Toc525659843)

[多项式相关 3](#_Toc525659844)

[多项式除法 5](#_Toc525659845)

杜教多项式插值

#include<stdio.h>

#include<string.h>

#include<algorithm>

#include<assert.h>

using namespace std;

typedef long long ll;

const int mod = 1e9+7;

namespace polysum {

#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--)

const int D=2010;//最高幂次, 只需要扔这么多项进来

ll a[D],f[D],g[D],p[D],p1[D],p2[D],b[D],h[D][2],C[D];

ll powmod(ll a,ll b){ll res=1;a%=mod;assert(b>=0);for(;b;b>>=1){if(b&1)res=res\*a%mod;a=a\*a%mod;}return res;}

ll calcn(int d,ll \*a,ll n) { // a[0].. a[d] a[n]

if (n<=d) return a[n];

p1[0]=p2[0]=1;

rep(i,0,d+1) {

ll t=(n-i+mod)%mod;

p1[i+1]=p1[i]\*t%mod;

}

rep(i,0,d+1) {

ll t=(n-d+i+mod)%mod;

p2[i+1]=p2[i]\*t%mod;

}

ll ans=0;

rep(i,0,d+1) {

ll t=g[i]\*g[d-i]%mod\*p1[i]%mod\*p2[d-i]%mod\*a[i]%mod;

if ((d-i)&1) ans=(ans-t+mod)%mod;

else ans=(ans+t)%mod;

}

return ans;

}

void init(int M) {//最高幂次

f[0]=f[1]=g[0]=g[1]=1;

rep(i,2,M+5) f[i]=f[i-1]\*i%mod;

g[M+4]=powmod(f[M+4],mod-2);

per(i,1,M+4) g[i]=g[i+1]\*(i+1)%mod;

}

ll polysum(ll m,ll \*a,ll n) { // a[0].. a[m] \sum\_{i=0}^{n-1} a[i]

ll b[D];

for(int i=0;i<=m;i++) b[i]=a[i];

b[m+1]=calcn(m,b,m+1);

rep(i,1,m+2) b[i]=(b[i-1]+b[i])%mod;

return calcn(m+1,b,n-1);

}

ll qpolysum(ll R,ll n,ll \*a,ll m) { // a[0].. a[m] \sum\_{i=0}^{n-1} a[i]\*R^i

if (R==1) return polysum(n,a,m);

a[m+1]=calcn(m,a,m+1);

ll r=powmod(R,mod-2),p3=0,p4=0,c,ans;

h[0][0]=0;h[0][1]=1;

rep(i,1,m+2) {

h[i][0]=(h[i-1][0]+a[i-1])\*r%mod;

h[i][1]=h[i-1][1]\*r%mod;

}

rep(i,0,m+2) {

ll t=g[i]\*g[m+1-i]%mod;

if (i&1) p3=((p3-h[i][0]\*t)%mod+mod)%mod,p4=((p4-h[i][1]\*t)%mod+mod)%mod;

else p3=(p3+h[i][0]\*t)%mod,p4=(p4+h[i][1]\*t)%mod;

}

c=powmod(p4,mod-2)\*(mod-p3)%mod;

rep(i,0,m+2) h[i][0]=(h[i][0]+h[i][1]\*c)%mod;

rep(i,0,m+2) C[i]=h[i][0];

ans=(calcn(m,C,n)\*powmod(R,n)-c)%mod;

if (ans<0) ans+=mod;

return ans;

}

} // polysum::init();

求x^2+y^2=n的(x,y)对数

typedef long long ll;

const ll inf = 1e9+7;

const ll maxn = 2e5+7;

int solve(int n){

int sum=0;

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

if(n%i==0){

if(i%4==1)sum++;

else if(i%4==3)sum--;

if(i\*i!=n){

if(n/i%4==1)sum++;

else if(n/i%4==3)sum--;

}

}

}

return sum\*4;

}

int solve2(int n){

while(n%2==0)n/=2;

int res=4;

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

if(n%i==0){

int sum=0;

while(n%i==0)n/=i,sum++;

if(i%4==1)

res=res\*(sum+1);

else if(i%4==3&&sum%2==1)

return 0;

}

}

if(n>1){

if(n%4==1)

res=res\*2;

}

return res;

}

AC自动机 另一种写法

**// 2016南宁D**

**// 复杂度是所有串的len和**

**// 题意: 是否存在一个排列, 使得能一一对应**

**// 做法: 求每个点前相同val的len差, 然后直接AC自动机**

**// 修改fail的写法**

namespace ACM {

const int maxn=1e6+7;

map<int,int> next[maxn];

int fail[maxn],len[maxn],tot;

bool mark[maxn];

void init() {

tot=0; len[0]=0; fail[0]=0; mark[0]=0; next[0].clear();

}

void insert(int s[],int n) {

int i,p=0;

REP(i,n) {

int c=s[i];

if (!next[p].count(c)) {

next[p][c]=++tot; len[tot]=len[p]+1;

fail[tot]=0; mark[tot]=0;

next[tot].clear();

} p=next[p][c];

} mark[p]=1;

}

int Q[maxn],ST,ED;

inline int getnext(int x,int c){

for (;;x=fail[x]){

if (len[x]+1<=c) c=0;

if (!x||next[x].count(c)) break;

} if (next[x].count(c)) return next[x][c];

return x;

}

void buildAC() {

ST=0; ED=-1; Q[++ED]=0;

while (ST<=ED) {

int p=Q[ST++];

for (auto now:next[p]){

int c=now.first,nxt=now.second;

if (p) fail[nxt]=getnext(fail[p],c);

else fail[nxt]=0;

Q[++ED]=nxt;

} mark[p]|=mark[fail[p]];

}

}

bool query(int a[],int n) {

int p=0,have=0,i;

REP(i,n) {

int c=a[i]; p=getnext(p,c);

have|=mark[p];

} return have;

}

}

多项式相关

// 主要思路不是这个裸的乘法啥的啊!

// from picks' blog

// 对G(F(x))=0进行泰勒展开

// G'(F\_{t+1}(x))=G(F\_t(x))+G'(F\_t(x))/1\*(F\_{t+1}-F\_t(x))^1+....

// 后方的系数在mod x^2^t+1的意义下全是0!(因为减的那里的系数是2^t)

// F\_{t+1}(x)=F\_t(x)-G(F\_t(x))/G'(F\_t(x))

// 所以手动求个导数即可!

// 注意这个G(F(t))就是满足的那个式子! 注意要有常数项(否则可以全是0 =\_=!)

// 三角函数需要利用虚数来做, e^{iF(x)}=cos(F(x))+isin(F(x))

// exp(x): F\_{t+1}(x)=F\_t(x)-F\_t(x)\*((ln(F\_t(x))-P(x))\*F\_t(x))

// ln(x) : ln(F(x))=\int(积分) F'(x)/F(x)

// 注意F[0]要是0, 因为求导的时候会去掉这个贡献, 积分回来

// 当且仅当常数项有逆元, 可以多项式求逆

// 求逆:C\*A≡1(mod x^n)

// B\*A≡1(mod x^(n/2))

// (B\*A-1)\*(B\*A-1)≡0(mod x^(n/2))

// B\*B\*A\*A-2\*A\*B+1≡0(mod x^n)

// B\*B\*A-2\*B+C≡0(mod x^n)

// C≡B\*(2-A\*B)(mod x^n)

// 求根:C\*C≡A(mod x^n)

// B\*B≡A(mod x^n/2)

// (B\*B-A)\*(B\*B-A)≡0(mod x^n)

// B\*B\*B\*B-2\*C\*C\*B\*B+C\*C\*C\*C≡0(mod x^n)

// (B\*B+C\*C)\*(B\*B+C\*C)≡4\*C\*C\*B\*B(mod x^n)

// B\*B+A≡2\*C\*B(mod x^n)

// C=(B\*B+A)/(2\*B)

namespace FFT {

const int maxn=1<<18|7;

struct complex {

double a,b;

complex(double \_a=.0,double \_b=.0):a(\_a),b(\_b) {}

complex operator+(const complex x)const {return complex(a+x.a,b+x.b);}

complex operator-(const complex x)const {return complex(a-x.a,b-x.b);}

complex operator\*(const complex x)const {return complex(a\*x.a-b\*x.b,a\*x.b+b\*x.a);}

};

complex wn[maxn];

void initwn(int l) {

static int len=0; int i;

if (len==l) return; else len=l;

REP(i,len) wn[i]=complex(cos(2\*pi\*i/l),sin(2\*pi\*i/l));

}

void fft(complex \*A,int len,int inv) {

int i,j,k; initwn(len);

for (i=1,j=len/2; i<len-1; i++) {

if (i<j) swap(A[i],A[j]);

k=len/2;

while (j>=k) j-=k,k/=2;

if (j<k) j+=k;

} for (i=2; i<=len; i<<=1) {

for (j=0; j<len; j+=i) {

for (k=j; k<(j+i/2); k++) {

complex a,b; a=A[k];

b=A[k+i/2]\*wn[(ll)(k-j)\*len/i];

A[k]=a+b; A[k+i/2]=a-b;

}

}

} if (inv==-1) REP(i,len) A[i]=complex(A[i].a/len,A[i].b/len);

}

inline complex conj(complex &A) {return complex(A.a,-A.b);}

void mul(int \*A,int \*B,int \*ans,int len,int mod) { //ans=A\*B

static complex x1[maxn],x2[maxn];

static complex x3[maxn],x4[maxn];

static const int S=1<<15 ; int i;

REP(i,len) x1[i]=complex(A[i]/S,A[i]%S);

REP(i,len) x2[i]=complex(B[i]/S,B[i]%S);

fft(x1,len,1); fft(x2,len,1);

REP(i,len) {//这个k1, b1就是前面的, 这就减掉了一半常数

int j=(len-i)&(len-1);

complex k1=(conj(x1[i])+x1[j])\*complex(0.5,0);//dft k1

complex b1=(conj(x1[i])-x1[j])\*complex(0,0.5);//dft b1

complex k2=(conj(x2[i])+x2[j])\*complex(0.5,0);//dft k2

complex b2=(conj(x2[i])-x2[j])\*complex(0,0.5);//dft b2

x3[i]=k1\*k2+k1\*b2\*complex(0,1);

x4[i]=b1\*k2+b1\*b2\*complex(0,1);

} fft(x3,len,-1); fft(x4,len,-1);

REP(i,len) {

ll kk=x3[i].a+0.5,kb=x3[i].b+0.5;

ll bk=x4[i].a+0.5,bb=x4[i].b+0.5;

ans[i]=((kk%mod\*S%mod+kb+bk)%mod\*S%mod+bb)%mod;

}

}

const ll Mod=19260817;

// 下方的东西和ntt就根本无关, 这个模数是可以改的, 是多项式相关的东西

// 也就是说, 这个模数完全可以取其他的, 然后高精度的mtt来求, 不过可能会T到死

int eInv[maxn];

void initinv(int l) {

int i; eInv[0]=eInv[1]=1;

rep(i,2,l) eInv[i]=(Mod-Mod/i)\*eInv[Mod%i]%Mod;

}

void Ftof(int \*A,int \*B,int l) {//derivative 求导

int i;

FOR(i,1,l) B[i-1]=(ll)A[i]\*i%Mod;

}

void ftoF(int \*A,int \*B,int l) {//integral 积分

int i; // todo:get B[0], getinv

rFOR(i,1,l) B[i]=(ll)A[i-1]\*eInv[i]%Mod;

B[0]=0;

}

void inv(int \*A,int \*B,int l) { //B=inv(A)

static int C[maxn],D[maxn];

B[0]=eInv[A[0]]; B[1]=0;

for (int len=2; len<=l; len<<=1) {

int i; fill(B+len,B+len+len,0);

copy(A,A+len,C); fill(C+len,C+len+len,0);

mul(C,B,D,len\*2,Mod); fill(D+len,D+len+len,0);

mul(D,B,D,len\*2,Mod);

REP(i,len) B[i]=(B[i]\*2-D[i]+Mod)%Mod;

fill(B+len,B+len+len,0);

}

}

void ln(int \*A,int \*B,int l) {

static int C[maxn];

inv(A,B,l); Ftof(A,C,l);

mul(B,C,B,l\*2,Mod);

ftoF(B,B,l);

}

void exp(int \*A,int \*B,int l) {

static int C[maxn],i;

B[0]=1; B[1]=0;

for (int len=2; len<=l; len<<=1) {

fill(B+len,B+len+len,0);

ln(B,C,len); fill(C+len,C+len+len,0);

REP(i,len) C[i]=(C[i]-A[i]+Mod)%Mod;

mul(B,C,C,len\*2,Mod);

REP(i,len) B[i]=(B[i]-C[i]+Mod)%Mod;

}

}

//这里是更高一层的东西

static int A[maxn],B[maxn];

void multiply(int \*a,int \*b,int \*ans,int n,int m) {//C=A\*B(actual)

int len=1,i;

while (len<n+m-2) len<<=1;

REP(i,n) A[i]=a[i]; rep(i,n,len) A[i]=0;

REP(i,m) B[i]=b[i]; rep(i,m,len) B[i]=0;

mul(A,B,ans,len,Mod);

}

void getexp(int \*a,int \*ans,int n) {

int len=1,i;

while (len<n) len<<=1;

REP(i,n) A[i]=a[i]; rep(i,n,len) A[i]=0;

exp(A,ans,len);

}

void solve(int \*a,int \*ans,int m) {

static int A[maxn];

int i,j;

FOR(i,1,m) {//无穷背包

int now=(ll)i\*a[i]%Mod;

for (j=i-1; j<=m; j+=i) A[j]=(now+A[j])%Mod;

} ftoF(A,A,m);

getexp(A,ans,m+1);

}

}

多项式除法

// http://codeforces.com/contest/438/problem/E

// 题意: 问你有多少个二叉树点权从c中取, 而且权值和是k

// 做法: 考虑多一个点, 所以f[x]=sigma{f[k]\*f[x-k-s],(s in c)}

// 所以 满足 F=F^2\*C+1, 左边是生成函数

// 所以 F=[1-sqrt(1-4C)]/2C=1/(1+sqrt(1-4C))

// 当且仅当常数项有逆元, 可以多项式求逆

// 求逆:C\*A≡1(mod x^n)

// B\*A≡1(mod x^(n/2))

// (B\*A-1)\*(B\*A-1)≡0(mod x^(n/2))

// B\*B\*A\*A-2\*A\*B+1≡0(mod x^n)

// B\*B\*A-2\*B+C≡0(mod x^n)

// C≡B\*(2-A\*B)(mod x^n)

// 求根:C\*C≡A(mod x^n)

// B\*B≡A(mod x^n/2)

// (B\*B-A)\*(B\*B-A)≡0(mod x^n)

// B\*B\*B\*B-2\*C\*C\*B\*B+C\*C\*C\*C≡0(mod x^n)

// (B\*B+C\*C)\*(B\*B+C\*C)≡4\*C\*C\*B\*B(mod x^n)

// B\*B+A≡2\*C\*B(mod x^n)

// C=(B\*B+A)/(2\*B)

namespace NTT {

const int maxn=1<<20|7;

const ll MOD=998244353;

const ll g=3;

int wn[maxn],invwn[maxn];

ll mul(ll x,ll y) {

return x\*y%MOD;

}

ll poww(ll a,ll b) {

ll ret=1;

for (; b; b>>=1ll,a=mul(a,a))

if (b&1) ret=mul(ret,a);

return ret;

}

void initwn(int l) {

static int len=0;

if (len==l) return; len=l;

ll w=poww(g,(MOD-1)/len); int i;

ll invw=poww(w,MOD-2); wn[0]=invwn[0]=1;

rep(i,1,len) {

wn[i]=mul(wn[i-1],w);

invwn[i]=mul(invw,invwn[i-1]);

}

}

void ntt(ll \*A,int len,int inv) {

int i,j,k; initwn(len);

for (i=1,j=len/2; i<len-1; i++) {

if (i<j) swap(A[i],A[j]);

k=len/2;

while (j>=k) j-=k,k/=2;

if (j<k) j+=k;

} for (i=2; i<=len; i<<=1) {

for (j=0; j<len; j+=i) {

for (k=j; k<(j+i/2); k++) {

ll a,b; a=A[k];

if (inv==-1) b=mul(A[k+i/2],invwn[(ll)(k-j)\*len/i]);

else b=mul(A[k+i/2],wn[(ll)(k-j)\*len/i]);

A[k]=(a+b); (A[k]>=MOD) &&(A[k]-=MOD);

A[k+i/2]=(a-b+MOD); (A[k+i/2]>=MOD) &&(A[k+i/2]-=MOD);

}

}

} if (inv==-1) {

ll vn=poww(len,MOD-2);

REP(i,len) A[i]=mul(A[i],vn);

}

}

void mul(ll \*A,ll \*B,ll \*C,int len) { //C=A\*B

int i;

ntt(A,len,1); ntt(B,len,1);

REP(i,len) C[i]=mul(A[i],B[i]);

ntt(C,len,-1);

}

void inv(ll \*A,ll \*B,int l) { //B=inv(A)

static ll C[maxn];

B[0]=poww(A[0],MOD-2); B[1]=0;

for (int len=2; len<=l; len<<=1) {

int i; fill(B+len,B+len+len,0);

copy(A,A+len,C); fill(C+len,C+len+len,0);

ntt(C,len\*2,1); ntt(B,len\*2,1);

REP(i,len\*2) B[i]=mul(B[i],(MOD+2-mul(C[i],B[i])));

ntt(B,len\*2,-1); fill(B+len,B+len+len,0);

}

}

void sqrt(ll \*A,ll \*B,int l) { //B=sqrt(A)

static ll C[maxn],\_B[maxn];

B[0]=1; B[1]=0;// 这里应该是个二次剩余

for (int len=2; len<=l; len<<=1) {

int i; ll inv2=poww(2,MOD-2);

inv(B,\_B,len); fill(B+len,B+len+len,0);

copy(A,A+len,C); fill(C+len,C+len+len,0);

ntt(C,len\*2,1); ntt(\_B,len\*2,1); ntt(B,len\*2,1);

REP(i,len\*2) B[i]=mul(inv2,B[i]+mul(C[i],\_B[i]));

ntt(B,len\*2,-1); fill(B+len,B+len+len,0);

}

}

static ll A[maxn],B[maxn];

void multiply(ll \*a,ll \*b,ll \*ans,int n,int m) {//C=A\*B(actual)

int len=1,i;

while (len<n+m-2) len<<=1;

REP(i,n) A[i]=a[i]; rep(i,n,len) A[i]=0;

REP(i,m) B[i]=b[i]; rep(i,m,len) B[i]=0;

mul(A,B,ans,len);

}

void inverse(ll \*a,ll \*ans,int n){

int len=1,i;

while (len<n) len<<=1;

REP(i,n) A[i]=a[i]; rep(i,n,len) A[i]=0;

inv(A,ans,len);

}

void getsqrt(ll \*a,ll \*ans,int n){

int len=1,i;

while (len<n) len<<=1;

REP(i,n) A[i]=a[i]; rep(i,n,len) A[i]=0;

sqrt(A,ans,len);

}

void divide(ll \*a,ll \*b,ll \*ans,int n,int m,int &l) {

if (n<m) {l=1; ans[0]=0; return;}

int len=1,i; l=n-m+1;

while (len<n-m+1) len<<=1;

REP(i,n) A[i]=a[i]; reverse(A,A+n); min\_(n,l);

REP(i,m) B[i]=b[i]; reverse(B,B+m); min\_(m,l);

rep(i,m,len) B[i]=0;

inv(B,ans,len);

multiply(A,ans,ans,len,n);

reverse(ans,ans+l);

}

//ans1:答案; ans2:余数

void delivery(ll \*a,ll \*b,ll \*ans1,ll \*ans2,int n,int m,int &l1,int &l2) {

divide(a,b,ans1,n,m,l1); l2=m-1;

multiply(b,ans1,ans2,m,l1);

int i; REP(i,l2) ans2[i]=(a[i]-ans2[i]+M)%M;

}

}

ll A[maxn],ans[maxn];

int main() {

int i,k;

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

FOR(i,1,n) scanf("%d",&k),A[k]++;

REP(i,m+1) A[i]=-4\*A[i]; A[0]++;

REP(i,m+1) mod\_(A[i]);

NTT::getsqrt(A,ans,m+1);

add\_(ans[0],1);

NTT::inverse(ans,ans,m+1);

FOR(i,1,m) mul\_(ans[i],2);

FOR(i,1,m) printf("%lld\n",ans[i]);

}