

针对ntt无法解决的mod一个任意值的问题，可以用mtt来解决，方法是拆系数fft，因为不常考到，仅作为一个板子作为参考

#include <cstdio>

#include <complex>

#define debug(...) fprintf(stderr, \_\_VA\_ARGS\_\_)

typedef long long lolong;

typedef std::complex<double> complex;

inline int input() { int x; scanf("%d", &x); return x; }

inline lolong linput() { lolong x; scanf("%lld", &x); return x; }

const int maxn = 400005, maxk = 20;

const complex I(0, 1);

int R[maxn];

complex Wn[maxn];

void FFT(complex \*A, int n, int t) {

if(t == -1)

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

if(i < (n - i))

std::swap(A[i], A[n - i]);

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

if(i < R[i])

std::swap(A[i], A[R[i]]);

for(int m = 1, l = 0; m < n; m <<= 1, l ++) {

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

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

complex W = Wn[1ll \* (k - i) \* n / m];

complex a0 = A[k], a1 = A[k + m] \* W;

A[k] = a0 + a1;

A[k + m] = a0 - a1;

}

}

}

if(t == -1)

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

A[i] /= n;

}

int mod;

inline lolong num(complex x) {

double d = x.real();

return d < 0 ? lolong(d - 0.5) % mod : lolong(d + 0.5) % mod;

}

inline void FFTFFT(complex \*a, complex \*b, int len, int t) {

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

a[i] = a[i] + I \* b[i];

FFT(a, len, t);

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

b[i] = std::conj(a[i ? len - i : 0]);

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

complex p = a[i], q = b[i];

a[i] = (p + q) \* 0.5;

b[i] = (q - p) \* 0.5 \* I;

}

}

complex a0[maxn], a1[maxn], b0[maxn], b1[maxn];

complex p[maxn], q[maxn];

int main() {

int n = input(), m = input();

mod = input();

int M = int(sqrt(mod) + 1);

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

int x = input() % mod;

a0[i] = x / M;

a1[i] = x % M;

}

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

int x = input() % mod;

b0[i] = x / M;

b1[i] = x % M;

}

int len = 1;

while(len < n + m + 1)

len <<= 1;

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

R[i] = R[i >> 1] >> 1 | ((i & 1) \* (len >> 1));

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

Wn[i] = complex(cos(M\_PI / len \* i), sin(M\_PI / len \* i));

FFTFFT(a0, a1, len, 1);

FFTFFT(b0, b1, len, 1);

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

p[i] = a0[i] \* b0[i] + I \* a1[i] \* b0[i];

q[i] = a0[i] \* b1[i] + I \* a1[i] \* b1[i];

}

FFT(p, len, -1);

FFT(q, len, -1);

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

printf("%lld ", (M \* M \* num(p[i].real()) % mod +

M \* (num(p[i].imag()) + num(q[i].real())) % mod +

num(q[i].imag())) % mod);

puts("");

}