常系数非齐次线性递推

求 $f_n = H(n) + \sum_{i=1}^{k} a_i \times f_{n-i}$，其中 $deg(H(x)) = m$。

仍然考虑齐次递推的方法，构造转移矩阵。

定义初始状态为 $St = \{f_{k-1}, f_{k-2}, \ldots, f_0, n^m, n^{m-1}, \ldots, n^0\}$。

$A = \begin{pmatrix} A_{1k\times k} & 0_{k\times(m+1)} \\ A_{2(m+1)\times k} & A_{3(m+1)\times(m+1)} \end{pmatrix}$，其中

$$A_{1k\times k} = \begin{pmatrix} a_1 & 1 & 0 & 0 & \ldots & 0 \\ a_2 & 0 & 1 & 0 & \ldots & 0 \\ a_3 & 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_k & 0 & 0 & 0 & \ldots & 0 \end{pmatrix}$$

$$A_{2(m+1)\times k} = \begin{pmatrix} h_m & 0 & 0 & \ldots & 0 \\ h_{m-1} & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_0 & 0 & 0 & \ldots & 0 \end{pmatrix}$$

$$A_{3(m+1)\times(m+1)} = \begin{pmatrix} \binom{m}{m} & 0 & 0 & \ldots & 0 \\ \binom{m}{m-1} & \binom{m-1}{m-1} & 0 & \ldots & 0 \\ \binom{m}{m-2} & \binom{m-1}{m-2} & \binom{m-2}{m-2} & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \binom{m}{0} & \binom{m-1}{0} & \binom{m-2}{0} & \ldots & \binom{0}{0} \end{pmatrix}$$

其中 A_3 利用了二项式展开，将 $n^m \to (n+1)^m$。易知最后要求的是 $(StA^n)_k$。

考虑特征多项式 g，由特征多项式的定义，以及分块矩阵的行列式特点，得 $g(x) = |xI - A_1||xI - A_3|$。有齐次递推得到的式子，有 $g_1(x) = -x^n + \sum_{i=0}^{k-1} a_{k-i}x^i$，而 $|xI - A_3|$ 只有主对角线上有值，故 $g_2(x) = (x-1)^{m+1}$，二项式系数求一下即可。

$g(x) = g_1(x) * g_2(x)$，卷一下既得特征多项式的系数，同时也知道了 $deg(g) = m + 1 + k$，故还需递推出 $f_k, f_{k+1}, \ldots, f_{k+m}$ 的值。

设 F 为 f_i 的 OGF，A 为递推系数 a_i 的 OGF，P 为剩余项的 OGF。于是有 $F = F * A + P \Rightarrow F = \frac{P}{1-A}$ 当 $i \in [k, k+m]$时，由递推式子的定义有 $[x^i]P(x) = H(i)$，多点求值即可。而当 $i \in [0, k-1]$，由于 **初始值是题目随机给的**，P 中要减去 $F * A$ 产生的项，即 $[x^i]P(x) = a_i - F * A(i)$，卷积一下求个逆得到$f_0, f_1, \ldots, f_{k+m}$。最后套个齐次线性递推的模板就出来了。

```
//code sourced from kinesis
#include<bits/stdc++.h>
using namespace std;

#define _REP(i,a,b) for(int i = (a) ; i >= (int)(b) ; --i )
#define REP(i,a,b) for(int i = (a) ; i <= (int)(b) ; ++i )
#define UREP(i,u) for(int i = p[(u)] ; i + 1 ; i = edge[i].next)
//iterator: for(int u:x),x is container
#define x(p) (p).first
#define y(p) (p).second
#define pii pair<int,int>
#define mp(x,y) make_pair((x), (y))
#define sign(x) (fabs(x) < eps ? 0 : ((x) > 0 ? 1 : -1))
#define ll long long
#define L7 __int128//1<<7 bit
#define ull unsigned long long
const int inf = 0x3f3f3f3f;
const ll inff = 0x3f3f3f3f3f3f3f3f;
const int mod = 998244353;
const double eps = 1e-9;
#define ri1(x) scanf("%d", &(x))
#define ri2(x,y) scanf("%d%d", &(x), &(y))
#define ri3(x,y,z) scanf("%d%d%d", &(x), &(y), &(z))
#define ri4(a,b,c,d) scanf("%d%d%d%d", &(a), &(b), &(c), &(d))
//#define Debug
#ifdef Debug
#endif // Debug

const int maxn = 131072;//由于倍增的时候有30000*4的多项式存在，故最长长度开4倍

namespace NTT//优化过的ntt，使用时记得初始化
{
    const int p = 998244353,g = 3;
    int w[maxn<<2],inv[maxn<<2],r[maxn<<2],last;
    int mod(int x){return x >= p ? x - p : x;}
    ll qp(ll base,ll n)
    {
        base %= p;
        ll res = 1;
        while(n){
            if(n&1) (res *= base) %= p;
            (base *= base) %= p;
            n >>= 1;
        }
        return res;
    }

    void init()
    {
        int lim = maxn << 1;//最长数组的两倍
        inv[1] = 1;
        for(int i=2;i<=lim;i++) inv[i] = mod(p - 1ll * (p / i) * inv[p%i] % p);
        for(int i=1;i<lim;i<<=1)
        {
            int wn = qp(g,(p - 1) / (i<<1));
            for(int j=0,ww=1;j<i;j++,ww=1ll*ww*wn%p) w[i+j] = ww;
        }
    }

    void ntt(vector<int> &f,int n,int op)
    {
        if(last!=n)
        {
            for(int i=1;i<n;i++) r[i] = (r[i>>1]>>1)|((i&1)?(n>>1):0);
            last=n;
        }
        for(int i=1;i<n;i++) if(i<r[i])swap(f[i],f[r[i]]);
        for(int i=1;i<n;i<<=1)
        for(int j=0;j<n;j+=i<<1)
            for(int k=0;k<i;k++)
            {
                int x=f[j+k],y=1ll*f[i+j+k]*w[i+k]%p;
                f[j+k]=mod(x+y);f[i+j+k]=mod(x-y+p);
            }
        if(op==-1)
        {
            reverse(&f[1],&f[n]);
            for(int i=0;i<n;i++)f[i]=1ll*f[i]*inv[n]%p;
        }
    }
}
using NTT::ntt;

ll qp(ll base,ll n)
{
    base %= mod;
    ll res = 1;
    while(n){
        if(n&1) (res *= base) %= mod;
        (base *= base) %= mod;
        n >>= 1;
    }
    return res;
}

void PolyInv(int deg,vector<int> &f,vector<int> &g)//蝴蝶变换的rev[maxn<<2],f.resize(n<<2),g.resi
{
    if(deg==1) {g[0] = qp(f[0],mod-2);return ;}
    PolyInv((deg+1)>>1,f,g);
    int lim = 1;
    while(lim<(2*deg)) lim <<= 1;
    vector<int> h(lim);
    REP(i,0,deg-1) h[i] = f[i];
    ntt(g,lim,1),ntt(h,lim,1);
    REP(i,0,lim-1) g[i] = 1LL * (2LL + mod - 1LL * g[i] * h[i] % mod) % mod * g[i] % mod;
    ntt(g,lim,-1);
    REP(i,deg,lim-1) g[i] = 0;
}

namespace Comb
{
    ll Finv[maxn+10],fac[maxn+10],inv[maxn+10];
    ll qp(ll base,ll n) {
        ll res = 1;
        base%=mod;
        while(n){
            if(n&1) (res *= base) %= mod;
            (base *= base) %= mod;
            n >>= 1;
        }
        return res;
    }
    void init()
    {
        int n = maxn;
        inv[1]=1;
        for(int i=2;i<=n;++i)inv[i]=((mod-mod/i)*inv[mod%i])%mod;
        fac[0]=Finv[0]=1;
        for(int i=1;i<=n;++i)fac[i]=fac[i-1]*i%mod,Finv[i]=Finv[i-1]*inv[i]%mod;
    }
    ll C(ll n,ll m)
    {
        if(m<0||m>n)return 0;
        return fac[n]*Finv[n-m]%mod*Finv[m]%mod;
    }
}

namespace _Rec
{
    namespace mul_Eva
    {
        int x[maxn<<1],y[maxn<<1];
        vector<int> f(maxn<<1),h(maxn<<1),g[maxn<<1],_g,g_(maxn<<1);
        void Get_g(int p,int l,int r)
        {
            if(l==r){
                g[p].resize(2);
                g[p][0] = 1,g[p][1] = (mod - x[l]) % mod;
                return ;
            }
            int mid = l + r >> 1;
            Get_g(p<<1,l,mid),Get_g(p<<1|1,mid+1,r);
            int lim = 1,deg = r-l+2;
            while(lim<deg) lim <<= 1;
            g[p<<1].resize(lim),g[p<<1|1].resize(lim),g[p].resize(lim);
            ntt(g[p<<1],lim,1),ntt(g[p<<1|1],lim,1);
            REP(i,0,lim-1) g[p][i] = 1LL * g[p<<1][i] * g[p<<1|1][i] % mod;
            ntt(g[p],lim,-1);//做完以后除了g[1]每一个g都是点值形式
            g[p].resize(deg);
        }
        vector<int> mulT(vector<int> a,vector<int> b,int n)//除去idft之外的卷积转置，需要保证a和b是
        {
            int lim=a.size();
            for(int i=0;i<lim;i++)a[i]=1ll*a[i]*b[i]%mod;
            ntt(a,lim,1);
            return vector<int>(&a[0],&a[n]);//取a[0]-a[n-1]
        }
        void solve(int p,int l,int r,vector<int> h)
        {
            if(l==r){y[l]=h[0];return;}//把y作为存答案的数组
            int mid=(l+r)>>1,lim=1;
            while(lim<(r-l+2))lim<<=1;
            h.resize(lim);
            ntt(h,lim,-1);
            solve(p<<1,l,mid,mulT(g[p<<1|1],h,mid-l+1));
            solve(p<<1|1,mid+1,r,mulT(g[p<<1],h,r-mid));
        }
        void Get_Y(int m)
        {
            Get_g(1,0,m-1);
            _g = g[1];
            PolyInv(m,_g,g_);
            int lim = 1;
            while(lim<((m+2)<<1)) lim <<= 1;
            g_.resize(lim),f.resize(lim);
            ntt(g_,lim,1);ntt(f,lim,-1);
            solve(1,0,m-1,mulT(g_,f,m));
        }
    }
    namespace Recurrence{
        vector<int> g(maxn<<1),gT(maxn<<1),_gT(maxn<<1);
        void PolyModulo(int n,int m,vector<int> &f,vector<int> &R)//特别修改的多项式取模
        {
            int lim = 1,deg = 2*n - m + 10;
            while(lim<(deg)) lim <<= 1;
            vector<int> fT(lim),Q(lim);
            REP(i,0,n-1) fT[i] = f[n-1-i];
            ntt(fT,lim,1);
            REP(i,0,lim-1) fT[i] = 1LL * fT[i] * _gT[i] % mod;
            ntt(fT,lim,-1);
            REP(i,0,n-m) Q[i] = fT[i];
            reverse(Q.begin(),Q.begin()+n-m+1);
            lim = 1,deg = n + 10;
            while(lim<(deg)) lim <<= 1;
            vector<int> f1 = f;
            f1.resize(lim),Q.resize(lim),R.resize(lim);
            ntt(f1,lim,1),ntt(Q,lim,1);
            REP(i,0,lim-1) R[i] = (1LL * f1[i] + mod - 1LL * g[i] * Q[i] % mod) % mod;
            ntt(R,lim,-1);
        }
        void get_g(int n,int m)
        {
            int lim = 1,deg = 2*n - m + 10;//这里的长度要与mod_poly一致
            while(lim<(deg)) lim <<= 1;
            REP(i,0,m-1) gT[i] = g[m-1-i];
            PolyInv(n-m+1,gT,_gT);
            ntt(_gT,lim,1);
            lim = 1,deg = n + 10;//这里的长度同样要与mod_poly一致
            while(lim<(deg)) lim <<= 1;
            ntt(g,lim,1);
        }
        vector<int> f(maxn<<1);
        void mul(int k)
        {
            int lim = 1;
            while(lim<(k<<1)) lim <<= 1;
            ntt(f,lim,1);
            REP(i,0,lim-1) f[i] = 1LL * f[i] * f[i] % mod;
            ntt(f,lim,-1);
            vector<int> R;
            PolyModulo(2*k+1,k+1,f,R);
            REP(i,0,k-1) f[i] = R[i];
            REP(i,k,lim-1) f[i] = 0;
        }
        void add(int k)
        {
            _REP(i,k,1) f[i] = f[i-1];f[0] = 0;
        }

        int st[maxn<<1];//递推系数，初始值，长度均为k

        int work(int n,int k)//a_n = sum 1 to k
        {
            get_g(2*k+1,k+1);
            f[1] = 1;
            int cur = 1,lg = log2(n),ok = 0;
            _REP(i,lg-1,0){
                if(k<=(cur<<1)) ok = 1;
                if(!ok) f[cur<<1] = f[cur],f[cur] = 0,cur<<=1;
                else mul(k);
                if(n&(1<<i)){
                    if(!ok) f[cur+1] = f[cur],f[cur] = 0,cur+=1;
                    else add(k);
                }
            }
```

```
            vector<int> R;
            PolyModulo(2*k+1,k+1,f,R);
            REP(i,0,k-1) f[i] = R[i];
            int ans = 0;
            REP(i,0,k-1) (ans += 1LL * f[i] * st[i] % mod) %= mod;
            return (ans + mod) % mod;
        }
    }
    vector<int> a(maxn<<1),P(maxn<<1);
    int m,k;
    void get_P()
    {
        int lim = 1;
        while(lim<(k<<1)+2) lim <<= 1;
        vector<int> _A(lim),B(lim);
        REP(i,0,k-1) _A[i] = Recurrence::st[i];
        REP(i,1,k) B[i] = a[i];
        ntt(_A,lim,1),ntt(B,lim,1);
        REP(i,0,lim-1) _A[i] = 1LL * _A[i] * B[i] % mod;
        ntt(_A,lim,-1);
        REP(i,0,k-1) P[i] = (1LL * Recurrence::st[i] - _A[i] + mod) % mod;
        REP(i,0,m) mul_Eva::x[i] = k + i;
        mul_Eva::Get_Y(m+1);
        REP(i,0,m) P[k+i] = mul_Eva::y[i];
    }
    void get_g()
    {
        int op = 1,lim = 1;
        while(lim<(m+k+3)) lim <<= 1;
        vector<int> h1(lim),h2(lim);
        REP(i,0,m+1) h1[i] = (1LL * op * Comb::C(m+1,i) + mod) % mod,op = -op;
        REP(i,0,k-1) h2[i] = a[k-i];h2[k] = mod - 1;
        ntt(h1,lim,1),ntt(h2,lim,1);
        REP(i,0,lim-1) Recurrence::g[i] = 1LL * h1[i] * h2[i] % mod;
        ntt(Recurrence::g,lim,-1);
    }
    void get_st()
    {
        int lim = 1;
        while(lim<(2*m+2*k+2)) lim <<= 1;
        vector<int> B(lim),_f(lim);
        REP(i,1,k) B[i] = (mod - a[i]) % mod;B[0] = 1;
        PolyInv(m+k+1,B,_f);
        ntt(_f,lim,1),ntt(P,lim,1);
        REP(i,0,lim-1) P[i] = 1LL * P[i] * _f[i] % mod;
        ntt(P,lim,-1);
        REP(i,0,m+k) Recurrence::st[i] = P[i];
    }
    int work(int n)
    {
        get_P();
        get_g();
        get_st();
        return Recurrence::work(n,m+k+1);
    }
}

int main()
{
    #ifndef ONLINE_JUDGE
    freopen("in.txt","r",stdin);//cf needn't delete this
    #endif // ONLINE_JUDGE
    NTT::init();
    Comb::init();
    int n;
    ri3(n,_Rec::m,_Rec::k);
    REP(i,0,_Rec::k-1) ri1(_Rec::Recurrence::st[i]);
    REP(i,1,_Rec::k) ri1(_Rec::a[i]);
    REP(i,0,_Rec::m) ri1(_Rec::mul_Eva::f[i]);
    printf("%d\n",_Rec::work(n));
    return 0;
}
```