多项式全家桶

模板

```
//#pragma GCC optimize("Ofast", "inline", "-ffast-math")
//#pragma GCC target("avx,sse2,sse3,sse4,mmx")
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
#define int long long
#define rep(i,x,y) for(int i=x;i<=y;++i)
#define repd(i,x,y) for(int i=x;i>=y;--i)
using namespace std;
const int N=2e6+7;
const int mod=998244353,g0=3,inv3=332748118;
int n,m,p,len,rev[N],inv[N],inv2;
int lim,a[N],b[N],invb[N],c[N],d[N],F[N],G[N],A0[N],A[N],B[N]; //多项式要用到的数组

int read(){ int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}
void write(int x){if(x>9) write(x/10);putchar(x%10+'0');}

void revlim(){rep(i,0,lim-1) rev[i]=((rev[i>>1]>>1)|((i&1)*(lim>>1)));}
void up(int x){lim=1;for(;lim<=x;lim<<=1);}
void cle(int *f){rep(i,0,lim-1) f[i]=0;}

inline int fpow(int a,int b=mod-2){  //快速幂
    int res=1;
    while(b){
        if(b&1) res=res*a%mod;
        a=a*a%mod;
        b>>=1;
    }
    return res;
}

inline int Inv(int x){
    if(inv[x]) return inv[x];
    return inv[x]=fpow(x,mod-2);
}

void NTT(int a[],int len,int type){   //多项式卷积
```

```
    rep(i,0,len-1) rev[i]=(rev[i>>1]>>1)|((i&1)*(len>>1));
    rep(i,0,len-1) if(i<rev[i]) swap(a[i],a[rev[i]]);
    for(int i=1;i<len;i<<=1){
        int wn=fpow(3,(mod-1)/(i<<1));
        if(type==-1) wn=fpow(wn);
        for(int j=0;j<len;j+=(i<<1)){
            int w=1,x,y;
            rep(k,0,i-1){
                x=a[j+k],y=a[i+j+k]*w%mod,w=w*wn%mod;
                a[j+k]=(x+y)%mod,a[i+j+k]=(x-y+mod)%mod;
            }
        }
    }
    if(type==-1){
        int x=fpow(len,mod-2);
        rep(i,0,len-1) a[i]=a[i]*x%mod;
    }
}

void get_der(int *f,int *g,int len){  //多项式求导
    for(int i=1;i<len;++i) g[i-1]=i*f[i]%mod;
    g[len-1]=0;
}

void get_int(int *f,int *g,int len){  //多项式积分
    for(int i=1;i<len;++i) g[i]=f[i-1]*Inv(i)%mod;
    g[0]=0;
}

void get_inv(int *f, int *g, int len){   //多项式求逆
    if(len==1){
        g[0]=fpow(f[0]);
        return;
    }
    get_inv(f,g,(len+1)>>1);
    up(len<<1),cle(c),copy(f,f+len,c),revlim(),NTT(c,lim,1),NTT(g,lim,1);
    rep(i,0,lim-1) g[i]=g[i]*(2ll+mod-g[i]*c[i]%mod)%mod;
    NTT(g,lim,-1),fill(g+len,g+lim,0);
}

void get_ln(int *f,int *g,int len){  //多项式对数函数
    get_der(f,A,n);get_inv(f,B,n);
    up(len<<1);revlim();
    NTT(A,lim,1);NTT(B,lim,1);
    rep(i,0,lim-1) A[i]=A[i]*B[i]%mod;
    NTT(A,lim,-1);
    get_int(A,g,len-1);
}

void get_sqrt(int a[],int b[],int n){  //多项式开根
    if(n==1){
        b[0]=1;
        return;
    }
    get_sqrt(a,b,n>>1);
    rep(i,0,n-1) A[i]=a[i];
    rep(i,n,2*n-1) A[i]=0;
    rep(i,0,2*n-1) B[i]=0;
```

```
    get_inv(b,B,n);
    NTT(A,n<<1,1);NTT(B,n<<1,1);
    rep(i,0,2*n-1) A[i]=A[i]*B[i]%mod;
    NTT(A,n<<1,-1);
    rep(i,0,n-1) b[i]=(b[i]+A[i])*Inv(2)%mod;
    rep(i,n,2*n-1) b[i]=0;
}

void get_mod(int a[],int b[],int n,int m){  //多项式取模
    rep(i,0,n) A[i]=a[n-i];
    rep(i,0,m) B[i]=b[m-i];
    rep(i,n-m+1,n) A[i]=0;
    rep(i,n-m+1,m) B[i]=0;
    for(len=1;len<=n-m;len<<=1);
    get_inv(B,invb,len),len<<=1;
    NTT(A,len,1);NTT(invb,len,1);
    rep(i,0,len-1) A[i]=A[i]*invb[i]%mod;
    NTT(A,len,-1);
    rep(i,n-m+1,len-1) A[i]=0;
    rep(i,0,(n-m)/2) swap(A[i],A[n-m-i]);
    for(len=1;len<=n;len<<=1);
    NTT(b,len,1);NTT(A,len,1);
    rep(i,0,len-1) b[i]=b[i]*A[i]%mod;
    NTT(b,len,-1);
    rep(i,0,m-1) c[i]=(a[i]-b[i]+mod)%mod;
}

void get_div(int *f,int *g,int *p,int *q,int n,int m){  //多项式除法
    up(n<<1),cle(a),cle(b),copy(g,g+m,a),reverse(a,a+m);
    get_inv(a,b,n),up(n<<1),revlim(),copy(f,f+n,a),reverse(a,a+n);
    NTT(a,lim,1);NTT(b,lim,1);
    rep(i,0,lim-1) p[i]=a[i]*b[i]%mod;
    NTT(p,lim,-1);
    fill(p+n-m+1,p+lim,0),reverse(p,p+n-m+1);
    cle(a),cle(d),copy(p,p+n-m+1,d),copy(g,g+m,a),NTT(d,lim,1),NTT(a,lim,1);
    rep(i,0,lim-1) d[i]=d[i]*a[i]%mod;
    NTT(d,lim,-1);
    rep(i,0,m-2) q[i]=(f[i]+mod-d[i])%mod;
}

void Solve(){
    n=read();m=read();p=read();
    rep(i,0,n-1) F[i]=Int(read()%p);
    rep(i,0,m-1) G[i]=read()%mod;
    up(n+m);revlim();
    NTT(F,lim,1);NTT(G,lim,1);
    rep(i,0,lim-1) F[i]= F[i]*G[i];
    NTT(A,lim,-1);
    rep(i,0,n+m-2){
        printf(A[i].get());
    }
}

signed main(){
//  ios::sync_with_stdio(0);
//  cin.tie(0);cout.tie(0);
//  freopen("in.cpp","r",stdin);
//  freopen("out.cpp","w",stdout);
```

```
    int T=1;
//  cin>>T;
//  clock_t start,finish;
//  start=clock();
    while(T--){
        solve();
    }
//  finish=clock();
//  cerr<<((double)finish-start)/CLOCKS_PER_SEC<<endl;
    return 0;
}
```

例题

2022牛客多校1 H-Fly(NTT优化背包)

给出 $a_1, a_2, \ldots, a_n$，以及 k 个限制 $(b_1, c_1), (b_2, c_2), \ldots, (b_k, c_k)$，$(b_i, c_i)$ 表示 x_{b_i} 的第 c_i 位（从低位向高位数，最低位为第 0 位）必须为 0。

给定整数 M，求满足 $a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq M$且 满足上述 k个限制的 $(x_1, x_2, \ldots, x_n)$ 的方案数。

```
#include <bits/stdc++.h>
#define int long long
using namespace std;
const int mod=998244353;
const int N=1e5+5e4;
const int G=3,Gi=332748118;
const int lim=40001;

int fpow(int a,int b=mod-2){
    int res=1;
    while(b){
        if(b&1) res=res*a%mod;
        a=a*a%mod;
        b>>=1;
    }
    return res;
}

int a[N],A[N],F[N],g[N];
int n,m,K,cnt[N],r[N<<1];
vector<int>had[62];
int vis[N], tag;

inline int get(int x){
    int len=1;
    while (len<=x) len<<=1;
    return len;
}

void NTT(int *A, int len, int tag){
    for(int i=1;i<=len;i++) r[i]=(r[i>>1]>>1)|((i&1)?(len>>1):0);
    for(int i=0;i<len;i++){
```

```
        if (i<r[i]) swap(A[i],A[r[i]]);
    }
    for(int mid=1;mid<len;mid<<=1){
        int wn=fpow(tag?G:Gi,(mod-1)/(mid<<1));
        for(int j=0;j<len;j+=(mid<<1)){
            for (int k=0,w=1;k<mid;k++,w=w*wn%mod){
                int x=A[j+k],y=w*A[j+k+mid]%mod;
                A[j+k]=(x+y)%mod;
                A[j+k+mid]=(x+mod-y)%mod;
            }
        }
    }
    if(tag) return;
    int invlen=fpow(len);
    for(int i=0;i<len;i++) A[i]=A[i]*invlen%mod;
}

signed main() {
    scanf("%d%lld%lld",&n,&m,&K);
    a[0]=1;
    ++cnt[1];
    for (int i=1;i<=n;i++) scanf("%d",&a[i]),++cnt[a[i]];
    for (int i=1,x,y;i<=K;i++){
        scanf("%d%d",&x,&y);
        had[y].push_back(x);
    }
    int len=get(lim);
    int w=fpow(3,(mod-1)/len);
    A[0]=1;
    for(int i=1;i<len;i++) A[i]=A[i-1]*w%mod;
    for(int i=0;i<len;i++) F[i]=1;
    for(int i=0;i<=lim;i++){
        if(cnt[i]){
            for (int j=0;j<len;j++){
                F[j]=F[j]*fpow((A[(j*i)&(len-1)]+1)%mod,cnt[i])%mod;
            }
        }
    }
    NTT(F,len,0);
    len=get(2*lim);
    memset(A,0,sizeof(A));
    A[0]=1;
    for(int i=0;(i<=60)&&m;i++){
        memcpy(g,F,sizeof(F));
        ++tag;
        for(auto x:had[i]){
            if(vis[x]!=tag){
                vis[x]=tag;
                for(int j=a[x];j<=lim;j++){
                    g[j]=(g[j]+mod-g[j-a[x]])%mod;
                }
            }
        }
        NTT(A,len,1);
        NTT(g,len,1);
        for (int j=0;j<len;j++) g[j]=g[j]*A[j]%mod;
        NTT(g,len,0);
        memset(A,0,sizeof(A));
```

```
        for(int j=m%2;j<=2*lim;j+=2) A[j>>1]=g[j];
        m>>=1;
    }
    printf("%d", A[0]);
    return 0;
}
```

2022牛客多校2 E - Falfa with Substring

求字符串中有多少个子序列为“bit”。

设$G_{n,k}$表示长度为n的字符串至少有k个bit子串，方案数为$G_{n,k}=C_{n-2k}^{k}*26^{n-3k}$，利用容斥推除长度为n的字符恰好有k个bit子串方案数为$F_{n,k}=\sum_{j\ge k}(-1)^{j-k}C_j^k$。经典地多项式卷积，可化为$k!F_{n,k}=\sum_{j\ge k}H_{n,n-j+k}j!G_{n,j}$,套一个NTT卷积即可。

```
//#pragma GCC optimize("Ofast", "inline", "-ffast-math")
//#pragma GCC target("avx,sse2,sse3,sse4,mmx")
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
#define int long long
using namespace std;
const int N=2e6+7;
const int mod=998244353,g=3,gi=332748118;
//998244353的一个原根为3,3在模此数意义下逆元为332748118

int read(){ int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}
void write(int x){if(x>9) write(x/10);putchar(x%10+'0');}

int n,m,lim,ln,pw[N],F[N],G[N],rev[N],frac[N],invf[N],ans[N];

inline int fpow(int a,int b){
    int res=1;
    while(b){
        if(b&1) res=res*a%mod;
        a=a*a%mod;
        b>>=1;
    }
    return res;
}

inline int inv(int x){
    return fpow(x,mod-2);
}

inline int C(int n,int m){
    if(n<m) return 0;
    return frac[n]*invf[m]%mod*invf[n-m]%mod;
}=
void NTT(int *a,int op) {
    for (int i=0;i<lim;i++){
        if (i<rev[i]) swap(a[i],a[rev[i]]);
    }
    for (int i=1;i<lim;i<<=1) {
```

```
        int w=fpow(3,(mod-1)/(i<<1));
        if (op==-1) w=fpow(w,mod-2);
        for (int j=0;j<lim;j+=(i<<1)){
            int wn=1;
            for (int k=j;k<i+j;k++) {
                int t=a[i+k]*wn%mod;
                a[i+k]=(a[k]+mod-t)%mod;
                a[k]=(a[k]+t)%mod;
                wn=wn*w%mod;
            }
        }
    }
    if(op==-1){
        int invL=fpow(lim,mod-2);
        for(int i=0;i<lim;i++){
            a[i]=a[i]*invL%mod;
        }
    }
}

void Solve(){
    cin>>n;
    m=n/3;
    pw[0]=1;frac[0]=1;
    for(int i=1;i<=n;++i) pw[i]=pw[i-1]*26%mod;
    for(int i=1;i<=n;++i) frac[i]=frac[i-1]*i%mod;
    invf[n]=inv(frac[n]);
    for(int i=n;i>=1;--i) invf[i-1]=invf[i]*i%mod;
    for(int i=0;i<=m;++i) F[i]=C(n-2*i,i)*pw[n-3*i]%mod*frac[i]%mod;
    for(int i=0;i<=m;++i) G[m-i]=invf[i]*fpow(mod-1,i)%mod;
    for(lim=1,ln=0;lim<=m+m;lim<<=1,ln++);
    for(int i=0;i<lim;++i) rev[i]=(rev[i>>1]>>1)|((i&1)<<(ln-1));
    NTT(F,1);NTT(G,1);
    for(int i=0;i<lim;++i) F[i]=F[i]*G[i]%mod;
    NTT(F,-1);
    for(int i=0;i<=m;++i) ans[i]=F[i+m]*invf[i]%mod;
    for(int i=0;i<=n;++i) cout<<ans[i]<<" ";
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
//  freopen("in.cpp","r",stdin);
//  freopen("out.cpp","w",stdout);
    int T=1;
//  cin>>T;
//  clock_t start,finish;
//  start=clock();
    while(T--){
        Solve();
    }
//  finish=clock();
//  cerr<<((double)finish-start)/CLOCKS_PER_SEC<<endl;
    return 0;
}
```