快速数论变换(NTT)

原理

OI Wiki-快速数论变换(NTT)

例题

例 1

题意

P3803 【模板】多项式乘法(FFT)

给定一个 n 次多项式 $F(x)$,和一个 m 次多项式 $G(x)$,求出 $F(x)$ 和 $G(x)$ 的卷积,$1\le n,m\le 10^6$

题解

找到模数 p,使得 $p=qn+1,(n=2^k)$,对于模 p 的原根 g,$\displaystyle g_n\equiv g^q\equiv g^{\frac{p-1}{n}}$,可以看作 FFT 中 n 次单位根 ω_n 的等价，因为它们都是各自所在群的生成元。所以 NTT 的代码只需把 FFT 中的 ω_n 都用 $g^{\frac{p-1}{n}}$ 替换掉就好了。

评价

NTT 模板题

代码

```
#include<bits/stdc++.h>
using namespace std;
const int N=3*1e6+10,P=998244353,G=3,Gi=332748118;//原根3及其逆元
typedef long long ll;
int rev[N];
ll a[N],b[N];
ll fastpow(ll x,ll y){//快速幂
    ll ret=1;
    for(;y;y>>=1,x=x*x%P)
        if(y&1) ret=ret*x%P;
    return ret;
}
void NTT(ll *y,int len,int on){//与 FFT 的代码类似
```

```
    for(int i=0;i<len;i++)
        if(i<rev[i])
            swap(y[i],y[rev[i]]);
    for(int mid=1;mid<len;mid<<=1){
        ll wn=fastpow(on==1?G:Gi,(P-1)/(mid<<1));
        for(int j=0;j<len;j+=(mid<<1)){
            ll w=1;
            for(int k=0;k<mid;k++,w=(w*wn)%P){
                int u=y[j+k],t=w*y[j+k+mid]%P;
                y[j+k]=(u+t)%P;
                y[j+k+mid]=(u-t+P)%P;
            }
        }
    }
}
int main(){
    int n,m;
    scanf("%d%d",&n,&m);
    for(int i=0;i<=n;i++) scanf("%lld",&a[i]),a[i]%=P;
    for(int i=0;i<=m;i++) scanf("%lld",&b[i]),b[i]%=P;
    int len=1,l=0;
    while(len<=n+m) len<<=1,l++;//向上凑成2的幂次
    for(int i=0;i<len;i++) rev[i]=(rev[i>>1]>>1)|((i&1)<<(l-1));//蝴蝶变换
    NTT(a,len,1);
    NTT(b,len,1);
    for(int i=0;i<len;i++) a[i]=(a[i]*b[i])%P;
    NTT(a,len,-1);
    ll inv=fastpow(len,P-2);//最后要除以长度
    for(int i=0;i<=n+m;i++)
        printf("%lld ",(a[i]*inv)%P);

    return 0;
}
```

例 2

题意

P1919 【模板】A*B Problem升级版（FFT快速傅里叶）

给你两个正整数 a,b，求 $a\times b$。$1\le a,b\le 10^{1000000}$

题解

使用 NTT 求解本题

评价

NTT 模板题

代码

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=1e7+5;
const ll P=998244353,G=3,Gi=332748118;
int rev[N];
char s1[N],s2[N];
ll a[N],b[N];
int c[N];
ll fastpow(ll x,ll y){
    ll ret=1;
    for(;y;y>>=1,x=x*x%P)
        if(y&1) ret=ret*x%P;
    return ret;
}
void NTT(ll *y,int len,int on){
    for(int i=0;i<len;i++)
        if(i<rev[i])
            swap(y[i],y[rev[i]]);
    for(int mid=1;mid<len;mid<<=1){
        ll wn=fastpow(on==1?G:Gi,(P-1)/(mid<<1));
        for(int j=0;j<len;j+=(mid<<1)){
            ll w=1;
            for(int k=0;k<mid;k++,w=w*wn%P){
                ll u=y[j+k],t=w*y[j+k+mid]%P;
                y[j+k]=(u+t)%P;
                y[j+k+mid]=(u-t+P)%P;
            }
        }
    }
}
int main(){
    scanf("%s%s",s1,s2);
    int n=strlen(s1),m=strlen(s2);
    n--,m--;
    for(int i=0;i<=n;i++) a[i]=s1[n-i]-'0';
    for(int i=0;i<=m;i++) b[i]=s2[m-i]-'0';
    int len=1,l=0;
    while(len<=n+m) len<<=1,l++;
    for(int i=0;i<len;i++) rev[i]=(rev[i>>1]>>1)|((i&1)<<(l-1));
    NTT(a,len,1);
    NTT(b,len,1);
    for(int i=0;i<len;i++) a[i]=a[i]*b[i]%P;
```

```
    NTT(a,len,-1);
    ll inv=fastpow(len,P-2);
    for(int i=0;i<=n+m;i++){
        a[i]=a[i]*inv%P;
        c[i]+=a[i];
        c[i+1]+=c[i]/10;
        c[i]%=10;
    }
    if(c[n+m+1]!=0) n++;
    for(int i=n+m;i>=0;i--) printf("%d",c[i]);
    return 0;
}
```

例 3

题意

P4245 【模板】任意模数多项式乘法

给定 2 个多项式 $F(x),G(x)$，请求出 $F(x)*G(x)$ 系数对 p 取模，且不保证 p 可以分解成 $p=a\cdot 2^k+1$ 的形式，$1\le n,m\le 10^5,0\le a_i,b_i\le 10^9,2\le p\le 10^9+9$。

题解

若不对 p 取模，则系数最大值可达到 $np^2=10^{23}$，考虑使用 NTT 求解，选取 3 个模数 p_1,p_2,p_3，使得 $p_1p_2p_3>np^2$，分别求出这 3 个模下的系数 x，然后通过中国剩余定理合并答案，最后再对 p 取模得到最终答案。过程中注意防止 long long 溢出。

评价

MTT 模板题

代码

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e5+5;
typedef long long ll;
int rev[N<<2];
ll n,m;
ll a[N<<2],b[N<<2],c[N<<2];
ll a1[N<<2],b1[N<<2];
ll x[4][N<<2];
const ll p[4]={0,998244353,2281701377,469762049},G=3;//3个都有原根3的大模数
const ll p_12=p[1]*p[2];
```

```
ll fastpow(ll x,ll y,ll mod){
    ll ret=1;
    for(;y;y>>=1,x=x*x%mod)
        if(y&1) ret=ret*x%mod;
    return ret;
}
const ll
inv1=fastpow(p[1],p[2]-2,p[2]),inv2=fastpow(p_12%p[3],p[3]-2,p[3]);
void NTT(ll *y,int len,int on,ll P){
    ll Gi=fastpow(G,P-2,P);
    for(int i=0;i<len;i++)
        if(i<rev[i])
            swap(y[i],y[rev[i]]);
    for(int mid=1;mid<len;mid<<=1){
        ll wn=fastpow(on==1?G:Gi,(P-1)/(mid<<1),P);
        for(int j=0;j<len;j+=(mid<<1)){
            ll w=1;
            for(int k=0;k<mid;k++,w=w*wn%P){
                ll u=y[j+k],t=w*y[j+k+mid]%P;
                y[j+k]=(u+t)%P;
                y[j+k+mid]=(u-t+P)%P;
            }
        }
    }
}
void calc(ll *A,ll *B,int len,ll P,ll *ans){//计算某个模数 P 下的 NTT
    NTT(A,len,1,P);
    NTT(B,len,1,P);
    for(int i=0;i<len;i++) A[i]=A[i]*B[i]%P;
    NTT(A,len,-1,P);
    ll inv=fastpow(len,P-2,P);
    for(int i=0;i<=n+m;i++) ans[i]=A[i]*inv%P;
}
void MTT(ll *A,ll *B,int len,ll P,ll *C){//进行3次不同模数下的 NTT□然后通过中国剩余定理合并得到答案
    for(int i=1;i<=3;i++){
        memcpy(a1,a,sizeof(a1));
        memcpy(b1,b,sizeof(b1));
        calc(a1,b1,len,p[i],x[i]);
    }
    for(int i=0;i<=n+m;i++){
        ll ret=(x[2][i]-x[1][i]+p[2])%p[2]*inv1%p[2]*p[1]+x[1][i];
        C[i]=((x[3][i]-ret%p[3]+p[3])%p[3]*inv2%p[3]*(p_12%P)%P+ret)%P;
    }
}
int main(){
    ll P;
    scanf("%lld%lld%lld",&n,&m,&P);
    for(int i=0;i<=n;i++) scanf("%lld",&a[i]),a[i]%=P;
    for(int i=0;i<=m;i++) scanf("%lld",&b[i]),b[i]%=P;
    int len=1,l=0;
```

```
    while(len<=n+m) len<<=1,l++;
    for(int i=0;i<len;i++) rev[i]=(rev[i>>1]>>1)|((i&1)<<(l-1));
    MTT(a,b,len,P,c);
    for(int i=0;i<=n+m;i++)
        printf("%lld ",c[i]);
    return 0;
}
```