多项式 3

分治 FFT

算法简介

$O\left(n\log^2 n\right)$ 时间解决一些难以直接使用 FFT 解决的问题。

算法例题

洛谷p4721

题意

给定 $g_0,g_1\cdots g_{n-2}$，

已知 $f_0=1,f_{i+1}=\sum_{j=0}^{i} f_jg_{i-j}$，求 $f_0,f_1\cdots f_{n-1}$。

题解

发现转移过程可以用 CDQ 分治优化，区间 $[lef,mid]$ 对区间 $[mid,rig]$ 的贡献为

$$f_{i+1}\gets \sum_{j=lef}^{mid} f_jg_{i-j}$$

套用 NTT 可以 $O(n\log n)$ 求出 $\sum_{j=lef}^{mid} f_jg_{i-j},(mid\le i\lt rig)$，于是总时间复杂度为 $O\left(n\log^2 n\right)$。

```
const int MAXN=1e5+5,Mod=998244353,G=3,Inv_G=332748118;
int quick_pow(int a,int b){
    int ans=1;
    while(b){
        if(b&1)
        ans=1LL*ans*a%Mod;
        a=1LL*a*a%Mod;
        b>>=1;
    }
    return ans;
}
int rev[MAXN<<2];
int build(int k){
    int n,pos=0;
    while((1<<pos)<=k)pos++;
    n=1<<pos;
    _for(i,0,n)rev[i]=(rev[i>>1]>>1)|((i&1)<<(pos-1));
```

```
    return n;
}
void NTT(int *f,int n,int type){
    _for(i,0,n)if(i<rev[i])
    swap(f[i],f[rev[i]]);
    int t1,t2;
    for(int i=1;i<n;i<<=1){
        int w=quick_pow(type==1?G:Inv_G,(Mod-1)/(i<<1));
        for(int j=0;j<n;j+=(i<<1)){
            int cur=1;
            _for(k,j,j+i){
                t1=f[k],t2=1LL*cur*f[k+i]%Mod;
                f[k]=(t1+t2)%Mod,f[k+i]=(t1-t2)%Mod;
                cur=1LL*cur*w%Mod;
            }
        }
    }
    if(type==-1){
        int div=quick_pow(n,Mod-2);
        _for(i,0,n)
        f[i]=(1LL*f[i]*div%Mod+Mod)%Mod;
    }
}
int f[MAXN],g[MAXN],t1[MAXN<<2],t2[MAXN<<2];
void solve(int lef,int rig){
    if(lef==rig)return;
    int mid=lef+rig>>1;
    solve(lef,mid);
    int n1=mid-lef,n2=rig-lef-1,n=build(n1+n2);
    _rep(i,0,n1)t1[i]=f[i+lef];_for(i,n1+1,n)t1[i]=0;
    _rep(i,0,n2)t2[i]=g[i];_for(i,n2+1,n)t2[i]=0;
    NTT(t1,n,1);NTT(t2,n,1);
    _for(i,0,n)t1[i]=1LL*t1[i]*t2[i]%Mod;
    NTT(t1,n,-1);
    _for(i,mid,rig)f[i+1]=(f[i+1]+t1[i-lef])%Mod;
    solve(mid+1,rig);
}
int main()
{
    int n=read_int();
    _for(i,0,n-1)
    g[i]=read_int();
    f[0]=1;
    solve(0,n-1);
    _for(i,0,n)
    space(f[i]);
    return 0;
}
```

多项式求逆

算法简介

给定 $f(x)$□求 $f(x)f^{-1}(x)\equiv 1\pmod {x^n}$□时间复杂度 $O(n\log n)$□

算法实现

假设已知 $f(x)f_0^{-1}(x)\equiv 1\pmod {x^{\lceil \frac n2\rceil}}$□

由于 $f(x)f^{-1}(x)\equiv 1\pmod {x^n}$□显然有 $f(x)f^{-1}(x)\equiv 1\pmod {x^{\lceil \frac n2\rceil}}$□

于是 $f^{-1}(x)-f_0^{-1}(x)\equiv 0\pmod {x^{\lceil \frac n2\rceil}}$□

两倍同时平方，有 $f^{-2}(x)-2f^{-1}(x)f_0^{-1}(x)+f_0^{-2}(x)\equiv 0\pmod {x^n}$□

两边同时乘以 $f(x)$□有 $f^{-1}(x)\equiv f_0^{-1}(x)(2-f(x)f_0^{-1}(x))\pmod {x^n}$□

现在考虑逆元存在条件，发现只要 $[x^0]f(x)$ 的逆元存在，就可以递推出 $f(x)$ 的逆元。

于是 $f^{-1}(x)$ 存在等价于 $\left([x^0]f(x)\right)^{-1}$ 存在。

时间复杂度有 $T(n)=T\left(\frac n2\right)+O(n\log n)$□于是 $T(n)=O(n\log n)$□

递归版与递推版效率相差不大。

```
//递归版
int temp[MAXN<<2];
void ployinv(int *f,int *g,int n){
    if(n==1)
    return g[0]=quick_pow(f[0],Mod-2),void();
    ployinv(f,g,(n+1)>>1);
    int m=build(n<<1);
    _for(i,0,n)temp[i]=f[i];_for(i,n,m)temp[i]=0;
    NTT(temp,m,1);NTT(g,m,1);
    _for(i,0,m)g[i]=(2-1LL*temp[i]*g[i]%Mod)*g[i]%Mod;
    NTT(g,m,-1);
    _for(i,n,m)g[i]=0;
}
//递推版
int temp[MAXN<<2];
void ployinv(int *f,int *g,int n){
    g[0]=quick_pow(f[0],Mod-2);
    int n1=2,n2=4,pos=2;
    while((n1>>1)<n){
        _for(i,0,n2)rev[i]=(rev[i>>1]>>1)|((i&1)<<(pos-1));
        _for(i,0,n1)temp[i]=f[i];_for(i,n1,n2)temp[i]=0;
        NTT(temp,n2,1);NTT(g,n2,1);
        _for(i,0,n2)g[i]=(2-1LL*temp[i]*g[i]%Mod)*g[i]%Mod;
```

```
        NTT(g,n2,-1);
        _for(i,n1,n2)g[i]=0;
        n1<<=1,n2<<=1,pos++;
    }
    n1>>=1;
    _for(i,n,n1)g[i]=0;
}
```

多项式开方

算法简介

给定 $g(x)$，求 $f^2(x)\equiv g(x)\pmod {x^n}$，时间复杂度 $O(n\log n)$。

算法实现

假设已知 $f_0^2(x)\equiv g(x)\pmod {x^{\lceil \frac n2\rceil}}$。

两边平方，有 $\left(f_0^2(x)-g(x)\right)\equiv 0\pmod {x^n}$。

两边加上 $4f_0^2(x)g(x)$，有 $\left(f_0^2(x)+g(x)\right)^2 \equiv 4f_0^2(x)g(x)\pmod {x^n}$。

两边除以 $4f_0^2(x)$，有 $\left(\cfrac {f_0^2(x)+g(x)}{2f_0^2(x)}\right)^2 \equiv g(x)\pmod {x^n}$。

于是有 $f(x) \equiv \cfrac {f_0^2(x)+g(x)}{2f_0^2(x)} \equiv \cfrac {f_0(x)+f_0^{-1}(x)g(x)}2\pmod {x^n}$。

现在考虑 $f(x)$ 存在条件，发现只要 $([x^0]f(x))^2 \equiv [x^0]g(x)\pmod p$ 有解即可。

考虑 BSGS 求出 $[x^0]g(x)$ 对应原根的幂次，即可得到 $[x^0]f(x)$。

```
HASH_Table<int,int> H;
int bsgs(int a,int b){
    H.clear();
    int m=sqrt(Mod)+1,t=b,base;
    for(int i=1;i<=m;i++){
        t=1LL*t*a%Mod;
        H.insert(t,i);
    }
    t=1,base=quick_pow(a,m);
    for(int i=1;i<=m;i++){
        t=1LL*t*base%Mod;
        if(H.find(t)!=-1)return m*i-H.find(t);
    }
    return -1;
}
int temp[MAXN<<2],inv_f[MAXN<<2];
```

```
void ploysqrt(int *f,int *g,int n){
    f[0]=quick_pow(3,bsgs(3,g[0])/2);
    int n1=2,n2=4,pos=2,inv2=quick_pow(2,Mod-2);
    while((n1>>1)<n){
        _for(i,0,n2)rev[i]=(rev[i>>1]>>1)|((i&1)<<(pos-1));
        _for(i,0,n2)inv_f[i]=0;
        ployinv(f,inv_f,n1);
        _for(i,0,n1)temp[i]=g[i];_for(i,n1,n2)temp[i]=0;
        NTT(inv_f,n2,1);NTT(temp,n2,1);
        _for(i,0,n2)temp[i]=1LL*temp[i]*inv_f[i]%Mod;
        NTT(temp,n2,-1);
        _for(i,0,n1)f[i]=1LL*(f[i]+temp[i])*inv2%Mod;
        n1<<=1,n2<<=1,pos++;
    }
    n1>>=1;
    _for(i,n,n1)f[i]=0;
}
```

多项式对数函数

算法简介

给定 $f(x)$，模 x^n 意义下的 $\ln f(x)$，时间复杂度 $O(n\log n)$。

算法实现

$$\mathrm{d}(\ln f(x))\equiv \frac {f^{\prime}(x)}{f(x)}\mathrm{d}x\pmod {x^n}$$

$$\ln f(x)-\ln f(0)\equiv \int_0^x f^{\prime}(t)f^{-1}(t)\mathrm{d}t\pmod {x^n}$$

由于一般只考虑 $f(0)=1$ 的情况，同时易知 $\int f^{\prime}(x)f^{-1}(x)$ 常数项为 0，于是有

$$\ln f(x)\equiv \int f^{\prime}(x)f^{-1}(x)\mathrm{d}x\pmod {x^n}$$

```
int inv_f[MAXN<<2];
void ployln(int *f,int n){
    int m=build(n<<1);
    _for(i,0,m)inv_f[i]=0;
    ployinv(f,inv_f,n);
    _rep(i,1,n)f[i-1]=1LL*f[i]*i%Mod;_for(i,n,m)f[i]=0;
    NTT(f,m,1);NTT(inv_f,m,1);
    _for(i,0,m)f[i]=1LL*f[i]*inv_f[i]%Mod;
    NTT(f,m,-1);
    for(int i=n-1;i>=0;i--)f[i]=1LL*f[i-1]*quick_pow(i,Mod-2)%Mod;
    f[0]=0;
    _for(i,n,m)f[i]=0;
}
```

多项式牛顿迭代法

算法简介

给定多项式 $g(x)$□求 $f(x)$ 满足 $g(f(x))\equiv 0\pmod {x^n}$□时间复杂度 $O(n\log n)$□

算法实现

首先单独求出 $[x^0]g(f(x))\equiv 0\pmod x$□假设已知 $g(f_0(x))\equiv 0\pmod {x^{\lceil \frac n2\rceil}}$□

将 $g(x)$ 在 $f_0(x)$ 处泰勒展开，有

$$\sum_{i=0}^{\infty} \cfrac {g^{(i)}(f_0(x))}{i!}(f(x)-f_0(x))^i\equiv 0\pmod {x^n}$$

同时有 $x^{\lceil \frac n2\rceil}\mid (f(x)-f_0(x))$□于是有 $(f(x)-f_0(x))^i\equiv 0\pmod {x^n}(i\ge 2)$□

$$\sum_{i=0}^{\infty} \cfrac {g^{(i)}(f_0(x))}{i!}(f(x)-f_0(x))^i\equiv g(f_0(x))+g^{\prime}(f_0(x))(f(x)-f_0(x))\equiv 0\pmod {x^n}$$

$$f(x)\equiv f_0(x)-\frac {g(f_0(x))}{g^{\prime}(f_0(x))}\pmod {x^n}$$

准确来说这里把 $f_0(x)$ 当成了变元 y□$g^{\prime}(f_0(x))=\cfrac {\partial g}{\partial y}(y,x)$□

举个例子□$g(f_0(x))=g(y,x)=xy+x^2+x=\cfrac x{f_0(x)}+x^2+x$□$g^{\prime}(f_0(x))=\cfrac {\partial g}{\partial y}(y,x)=-\cfrac x{y^2}=-\cfrac x{f_0^2(x)}$□