

# 快速数论变换

## Number Theory Transfrom

**Idea:** 只需将 **FFT** 与 **NTT** 之间建立起映射关系即可。

**FFT** 的关键在于单位复数根  $\omega$ ，它定义为  $\omega^n = 1$ ，其中主  $n$  次复数根定义为  $\omega_n = e^{2\pi i/n}$ ，满足消去、折半、求和引理。

那么在模  $p$  意义下，考虑  $p$  的原根  $g$ ，与  $\omega_n$  对应的便是  $g^{\frac{p-1}{n}}$ ，满足  $\left(g^{\frac{p-1}{n}}\right)^n \equiv g^{p-1} \equiv 1 \pmod{p}$ ，当然这里要求  $n$  是  $p-1$  的因子。下面证明  $g^{\frac{p-1}{n}}$  也满足消去、折半、求和引理：

- 消去引理：  $\left(g^{\frac{p-1}{n}}\right)^{dk} = \left(g^{\frac{p-1}{n}}\right)^k$ ，证明显然；
- 折半引理：  $\left(g^{\frac{p-1}{n}}\right)^2 = g^{\frac{p-1}{n/2}}$ ，证明显然；
- 求和引理：  $\sum_{j=0}^{n-1} \left(g^{\frac{p-1}{n}}\right)^{kj} \equiv \frac{\left(g^{\frac{p-1}{n}}\right)^{kn} - 1}{\left(g^{\frac{p-1}{n}}\right)^k - 1} \equiv \frac{g^{(p-1)k} - 1}{g^{(p-1)k/n} - 1} \equiv 0 \pmod{p}$ ，证明显然。

于是乎，关于 **FFT** 的一切也成立于 **NTT**，只需将  $\omega_n$  换成  $g^{\frac{p-1}{n}}$  即可。

由于  $n$  是 2 的幂次，又是  $p-1$  的因子，所以  $p$  是形如  $c \cdot 2^k + 1$  形式的素数，常用：

$p$	$g$	$\text{inv}(g)$	$n$ 的上界
$998244353 = 7 \times 17 \times 2^{23} + 1$	3	332748118	$n \leq 2^{23} = 8388608$
$1004535809 = 479 \times 2^{21} + 1$	3	334845270	$n \leq 2^{21} = 2097152$
$469762049 = 7 \times 2^{26} + 1$	3	156587350	$n \leq 2^{26} = 67108864$

**Code:**

```
1  const LL MOD = 998244353;
2  const LL G = 3;
3  const LL invG = 332748118;
4
5  namespace NTT{
6      int n;
7      vector<int> rev;
8      inline void preprocess(int _n, int _m){
9          int cntBit = 0;
10         for(n = 1; n <= _n + _m; n <<= 1, cntBit++);
11         // n == 2^cntBit is a upper bound of lenf+leng
12         rev.resize(n);
13         for(int i = 0; i < n; i++)
14             rev[i] = (rev[i>>1]>>1) | ((i&1) << (cntBit-1));
15         // rev[k] is bit-reversal permutation of k
16     }
17     inline void ntt(vector<LL> &A, int flag){
18         // flag == 1: NTT; flag == -1: INTT
19         A.resize(n);
20         for(int i = 0; i < n; i++) if(i < rev[i]) swap(A[i], A[rev[i]]);
21         for(int m = 2; m <= n; m <<= 1){
```

```

22     LL wm = flag == 1 ? fpow(G, (MOD-1)/m) : fpow(invG, (MOD-1)/m);
23     for(int k = 0; k < n; k += m){
24         LL w = 1;
25         for(int j = 0; j < m / 2; j++){
26             LL t = w * A[k+j+m/2] % MOD, u = A[k+j];
27             A[k+j] = (u + t) % MOD;
28             A[k+j+m/2] = (u - t + MOD) % MOD;
29             w = w * wm % MOD;
30         }
31     }
32 }
33 if(flag == -1){
34     LL inv = fpow(n, MOD-2);
35     for(int i = 0; i < n; i++)
36         (A[i] *= inv) %= MOD;
37 }
38 }
39 }
40
41 int main(){
42     // ... input
43     NTT::preprocess(n, m);
44     NTT::ntt(f, 1); // f used to be coefficients, now they're point-values
45     NTT::ntt(g, 1); // g used to be coefficients, now they're point-values
46     for(int i = 0; i < NTT::n; i++) f[i] = f[i] * g[i];
47     NTT::ntt(f, -1); // f used to be point-values, now they're coefficients
48     // ... output
49     return 0;
50 }

```