§4.11 快速傅里叶变换(Fast Fourier Transform)

4.11.1 多项式的点值表示法

对一个 n 次多项式：

$$f(x)=\sum_{i=0}^{n-1}a_i*x^i$$

其系数表达是一个由系数组成的向量$(a_0,a_1,\dots,a_{n-1})$，其点值表达是一个有 n 个元素的集合$\{(x_0,f(x_0)),(x_1,f(x_1)),\dots,(x_{n-1},f(x_{n-1}))\}$，其中$x_0,x_1,\dots,x_{n-1}$的取值各不同。

事实上，一个系数表达对应的点值表达是有无限多种的，但是任意一个点值表达，最多对应着一个系数表达。

在很多多项式的运算中，不单可以用系数表达运算，还可以用点值表达运算。

例如一个 n-1 次多项式$f(x)$和一个 m-1 次多项式间$g(x)$的乘法运算$h(x)=f(x)*g(x)$，显然$h(x)$是一个 n+m-2 次多项式。如果把$f(x)$和$g(x)$都看成 n+m-2 次多项式，只是后面有很多项的系数为 0，那么如果已知$f(x)$和$g(x)$的点值表达:

$$\{(x_0,f(x_0)),(x_1,f(x_1)),\dots,(x_{n+m-2},f(x_{n+m-2}))\}$$
$$\{(x_0,g(x_0)),(x_1,g(x_1)),\dots,(x_{n+m-2},g(x_{n+m-2}))\}$$

那么可以很快求出$h(x_0),h(x_1),\dots,h(x_{n+m-2})$，只需要将对应的点值相乘就行了，而不需要先求出$h(x)$的系数表达后再计算，那么这样就可以求出$h(x)$的点值表达了。

有了点值表达，如果能够在较快的时间内在点值表达与系数表达之间相互转换，那么就能够较快的进行多项式乘法了。

4.11.2 单位复根的一些性质

在数学中，复数是很重要的一部分，其中单位复根的性质更是用处多多。

令 n 次单位复根为ω_n，在复平面上可以表示为$(cos\frac{2\pi}{n},sin\frac{2\pi}{n})$，可以看作一个绕 x 轴逆时针旋转$\frac{2\pi}{n}$的向量。

单位复根有很多性质，快速傅里叶变换中需要的有：

$$\omega_n^n=1$$
$$\omega_n^m=\omega_{2n}^{2m}$$
$$\omega_{2n}^m=-\omega_{2n}^{m+n}$$

上述性质都可以在复平面上证明。

考虑

$$\frac{1}{n}\sum_{i=0}^{n-1}\omega_n^{r*i}$$

当$r\equiv 0\ (mod\ n)$的时候，ω_n^r=1，上式的值为 1。

否则根据等比数列求和：

$$\frac{1}{n}\sum_{i=0}^{n-1}\omega_n^{r*i}=\frac{1}{n}*\frac{1-\omega_n^{r*n}}{1-\omega_n^r}=0$$

那么逻辑表达式

$$[r \equiv 0\ (mod\ n)] = \frac{1}{n}\sum_{i=0}^{n-1} \omega_n^{r*i}$$

4.11.3 离散傅里叶变换

离散傅里叶变换（Discrete Fourier Transform，缩写为DFT），是傅里叶变换在时域和频域上都呈离散的形式，将信号的时域采样变换为其DTFT的频域采样。在信息学竞赛中，离散傅里叶变换的最大意义就是加速多项式乘法。

离散傅里叶变换可以将多项式(函数)在点值表示和系数表示之间转换。

对于一个n-1次的多项式$A(x), A_0 = A(\omega_n^0), A_1 = A(\omega_n^1), \dots, A_{n-1} = A(\omega_n^{n-1})$，根据定义：

$$A_r = \sum_{k=0}^{n-1} \omega_n^{k*r} * a_k$$

$$a_r = \sum_{i=0}^{n-1} [i = r] * a_i = \sum_{i=0}^{n-1} [i - r \equiv 0\ (mod\ n)] * a_i = \sum_{i=0}^{n-1} \frac{1}{n} \sum_{k=0}^{n-1} \omega_n^{(i-r)*k} * a_i$$

$$= \frac{1}{n} \sum_{k=0}^{n-1} \omega_n^{-r*k} * \sum_{i=0}^{n-1} \omega_n^{i*k} * a_i = \frac{1}{n} \sum_{k=0}^{n-1} \omega_n^{-r*k} * A_k$$

所以就得到了离散傅里叶变换公式

$$A_r = \sum_{k=0}^{n-1} \omega_n^{k*r} * a_k$$

$$a_r = \frac{1}{n} \sum_{k=0}^{n-1} \omega_n^{-r*k} * A_k$$

其中$a_0, a_1, \dots, a_{n-1}$是多项式$A(x)$的系数表达，$A_0, A_1, \dots, A_{n-1}$是多项式$A(x)$的点值表达。

4.11.4 快速傅里叶变换

在4.11.3中，已经提到了利用离散傅里叶在点值表达和系数表达中间相互转换，但是按照公式，显然还是太慢了。这时候需要利用单位复根的性质来加速DFT，也就是快速傅里叶变换(Fast Fourier Transform，缩写为FFT)。

首先考虑点值表达的计算：

考虑n-1(n为偶数)次多项式$A(x) = a_0 + a_1 x + \dots + a_{n-1} x^{n-1}$，设$A_{[0]}(X) = a_0 + a_2 x + \dots + a_{n-2} x^{\frac{n}{2}-1}$，$A_{[1]}(X) = a_1 + a_3 x + \dots + a_{n-1} x^{\frac{n}{2}-1}$。

那么

$$A(\omega_n^m) = a_0 + a_2 * (\omega_n^m)^2 + \dots + a_{n-2} * (\omega_n^m)^{n-2} + a_1 * \omega_n^m + a_3 * (\omega_n^m)^3 + \dots + a_{n-1} * (\omega_n^m)^{n-1}$$

$$= A_{[0]}(\omega_n^{2m}) + A_{[1]}(\omega_n^{2m}) * \omega_n^m = A_{[0]}\left(\omega_{\frac{n}{2}}^m\right) + A_{[1]}\left(\omega_{\frac{n}{2}}^m\right) * \omega_n^m$$

同理

$$A\left(\omega_n^{m+\frac{n}{2}}\right) = A_{[0]}(\omega_n^{2m+n}) + A_{[1]}(\omega_n^{2m+n}) * \omega_n^{m+\frac{n}{2}}$$

$$= A_{[0]}(\omega_n^{2m} * \omega_n^n) + A_{[1]}(\omega_n^{2m} * \omega_n^n) * \omega_n^{m+\frac{n}{2}}$$

$$= A_{[0]}(\omega_n^{2m}) + A_{[1]}(\omega_n^{2m}) * \omega_n^{m+\frac{n}{2}} = A_{[0]}\left(\omega_{\frac{n}{2}}^m\right) + A_{[1]}\left(\omega_{\frac{n}{2}}^m\right) * \omega_n^{m+\frac{n}{2}}$$

$$= A_{[0]}\left(\omega_{\frac{n}{2}}^m\right) - A_{[1]}\left(\omega_{\frac{n}{2}}^m\right) * \omega_n^m$$

那么只要有了$A_{[0]}(x)$的点值表达：

$$\left\{\left(\omega_{\frac{n}{2}}^0, A_{[0]}\left(\omega_{\frac{n}{2}}^0\right)\right), \left(\omega_{\frac{n}{2}}^1, A_{[0]}\left(\omega_{\frac{n}{2}}^1\right)\right), \dots, \left(\omega_{\frac{n}{2}}^{\frac{n}{2}-1}, A_{[0]}\left(\omega_{\frac{n}{2}}^{\frac{n}{2}-1}\right)\right)\right\}$$

和$A_1(x)$的点值表达：

$$\left\{\left(\omega_{\frac{n}{2}}^0, A_{[1]}\left(\omega_{\frac{n}{2}}^0\right)\right), \left(\omega_{\frac{n}{2}}^1, A_{[1]}\left(\omega_{\frac{n}{2}}^1\right)\right), \dots, \left(\omega_{\frac{n}{2}}^{\frac{n}{2}-1}, A_{[1]}\left(\omega_{\frac{n}{2}}^{\frac{n}{2}-1}\right)\right)\right\}$$

那么就能在$O(n)$的时间里算出$A(x)$的点值表达：

$$\left\{\left(\omega_n^0, A(\omega_n^0)\right), \left(\omega_n^1, A(\omega_n^1)\right), \dots, \left(\omega_n^{n-1}, A_1(\omega_n^{n-1})\right)\right\}$$

通过不断递归，显然是可以通过上述方法将系数表达转换成点值表达。那么对于一个 n-1 次多项式$A(x)$，其规模为$T(n)$，$A_{[0]}(x)$和$A_{[1]}(x)$的规模都是$T(\frac{n}{2})$，所以

$$T(n) = 2 * T\left(\frac{n}{2}\right) + O(n)$$

根据主定理，易得时间复杂度为$O(nlog_2 n)$。

以上就是快速傅里叶变换的大致思想。

事实上，如果想要用快速傅里叶变换，必须满足 n 是 2 的幂，所以当所求的 n 不是 2 的幂时，必须将 n 变成 2 的幂。

所以现在能够在$O(nlog_2 n)$的时间里完成系数表达向点值表达的转换了，对于点值表达向系数表达转换，根据 DFT 的式子，发现和系数表达向点值表达转换十分相似，所以利用同样的方法也可以在$O(nlog_2 n)$的时间里完成转换。

我们发现，如果要完成上述算法，必须要做到能够把一个多项式的奇数项和偶数项分开，虽然在递归中是可以实现的，但是常数过大。

这时观察程序执行的时候每个函数的系数，以 8 次多项式为例：

$$(a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7)$$
$$(a_0, a_2, a_4, a_6)(a_1, a_3, a_5, a_7)$$
$$(a_0, a_4)(a_2, a_6)(a_1, a_5)(a_3, a_7)$$
$$(a_0)(a_4)(a_2)(a_6)(a_1)(a_5)(a_3)(a_7)$$

观察$(0,4,2,6,1,5,3,7)$的二进制表示为$(000,100,010,110,001,101,011,111)$，发现就是把$(0,1,2,3,4,5,6,7)$的二进制表示$(000,001,010,011,100,101,110,111)$中的每一位的二进制位翻转，称为“位逆序置换”，那么如果能够较快的求出位逆序置换之后的序列，那么在求解的时候就可以每次处理的都是一段连续的区间，这时实现起来就比较简单了。至于位逆序置换，

怎么做都是也可以的。

4.11.5 数论变换

实际应用中，复数运算的 FFT，精度不高，可能导致误差。如果当进行多项式乘法的两个多项式的系数都是整数并且运算定义在模意义下，并且模数 P 比较特殊的时候，是可以不用复数的。

这种情况下就可以使用数论变换(Number Theoretic Transform，缩写为 NTT)，不但效率高，而且不会产生精度误差。那么对于一个 n-1(因为要用 FFT，所以 n 只能为 2 的幂)次多项式，到底什么模数 P 才能进行 NTT 呢？只要 P 满足$2^k|P-1$并且$2^k>n$。

在 P 满足上述条件时，求出 P 的原根 g，在模 P 意义下，根据原根的性质有

$$g^{P-1}\equiv 1(mod\ P)$$

并且$g^0,g^1,\dots,g^{P-2}$在模 P 意义下的取遍$1\sim P-1$中的所有数。

例如$P=998244353=2^{23}*7*17+1$，原根为 3

那么对于一个 n-1 次多项式，在利用 FFT 求解时，把所有的ω_k都用$g^{\frac{P-1}{k}}$来表示，是等价的。具体为什么只需要带入 DFT/FFT 的推导就能够证明正确性了。

4.11.6 多项式求逆

首先对多项式$A(x)=a_0+a_1x^1+a_2x^2+\cdots$在模$x^n$意义下的乘法逆元$B_n(x)=b_0+b_1x^1+b_2x^2+\cdots+b_{n-1}x^{n-1}$定义如下：

$$A(x)*B_n(x)\equiv 1\ (mod\ x^n)$$

如果暴力计算$B_n(x)$，时间复杂度是$O(n^2)$，效率太低了，可以考虑如下方法。

显然$B_1(x)=a_0^{-1}$。

假设已经知道了$B_{\lfloor\frac{n}{2}\rfloor}(x)$ ，那么有：

$$A(x)*B_{\lfloor\frac{n}{2}\rfloor}(x)\equiv 1\ (mod\ x^{\lceil\frac{n}{2}\rceil})$$

$$A(x)*B_{\lfloor\frac{n}{2}\rfloor}(x)-1\equiv 0\ (mod\ x^{\lceil\frac{n}{2}\rceil})$$

$$\left(A(x)*B_{\lfloor\frac{n}{2}\rfloor}(x)-1\right)^2\equiv 0\ (mod\ x^n)$$

$$A(x)^2*B_{\lfloor\frac{n}{2}\rfloor}(x)^2-2*A(x)*B_{\lfloor\frac{n}{2}\rfloor}(x)+1\equiv 0\ (mod\ x^n)$$

$$2*A(x)*B_{\lfloor\frac{n}{2}\rfloor}(x)-A(x)^2*B_{\lfloor\frac{n}{2}\rfloor}(x)^2\equiv 1\ (mod\ x^n)$$

$$A(x)*\left(2*B_{\lfloor\frac{n}{2}\rfloor}(x)-A(x)*B_{\lfloor\frac{n}{2}\rfloor}(x)^2\right)\equiv 1\ (mod\ x^n)$$

所以根据$B_n(x)$的定义，$B_n(x)=2*B_{\lfloor\frac{n}{2}\rfloor}(x)-A(x)*B_{\lfloor\frac{n}{2}\rfloor}(x)^2$，所以如果知道了$B_{\lfloor\frac{n}{2}\rfloor}(x)$，可以通过 FFT 在$O(nlog_2n)$的时间里求出$B_n(x)$。

所以求出$B_n(x)$的时间复杂度为$T(n)=T\left(\frac{n}{2}\right)+O(nlog_2n)$，所以时间复杂度为$O(nlog_2n)$。

4.11.7 多项式求平方根

对多项式$A(x)=a_0+a_1x^1+a_2x^2+\cdots$在模$x^n$意义下的平方根$B_n(x)=b_0+b_1x^1+b_2x^2+\cdots+b_{n-1}x^{n-1}$定义如下：

$$B_n(x)^2=A(x)\ (mod\ x^n)$$

考虑采用与多项式求逆的相同方法。

如果已经知道了$B_{\lceil\frac{n}{2}\rceil}(x)$，那么有：

$$B_{\lceil\frac{n}{2}\rceil}(x)^2=A(x)\ (mod\ x^{\lceil\frac{n}{2}\rceil})$$

$$B_{\lceil\frac{n}{2}\rceil}(x)^2-A(x)=0\ (mod\ x^{\lceil\frac{n}{2}\rceil})$$

$$\left(B_{\lceil\frac{n}{2}\rceil}(x)^2-A(x)\right)^2=0\ (mod\ x^n)$$

$$\left(B_{\lceil\frac{n}{2}\rceil}(x)^2+A(x)\right)^2=4*B_{\lceil\frac{n}{2}\rceil}(x)^2*A(x)\ (mod\ x^n)$$

$$\left(\frac{B_{\lceil\frac{n}{2}\rceil}(x)^2+A(x)}{2*B_{\lceil\frac{n}{2}\rceil}(x)}\right)^2=A(x)\ (mod\ x^n)$$

所以$B_n(x)=\frac{B_{\lceil\frac{n}{2}\rceil}(x)^2+A(x)}{2*B_{\lceil\frac{n}{2}\rceil}(x)}$，那么可以通过多项式求逆和多项式乘法在$O(nlog_2n)$的时间内求出。同样时间复杂度是$O(nlog_2n)$。