《图像信息处理》

第十一讲 FFT、及编程漫谈

潘 纲
gpan@zju.edu.cn

2011春夏

2D离散傅立叶变换

$$F(u,v)=\frac{1}{\sqrt{MN}}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)\exp[-\mathrm{j}2\pi(ux/M+vy/N)]$$

$$f(x,y)=\frac{1}{\sqrt{MN}}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}F(u,v)\exp[\mathrm{j}2\pi(ux/M+vy/N)]$$

频谱（幅度） $|F(u,v)|=\left[R^2(u,v)+I^2(u,v)\right]^{1/2}$

相位角 $\phi(u,v)=\arctan[I(u,v)/R(u,v)]$

功率谱 $P(u,v)=|F(u,v)|^2=R^2(u,v)+I^2(u,v)$

快速傅里叶变换 (FFT)

$$F(u,v)=\frac{1}{\sqrt{MN}}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)\exp[-\mathrm{j}2\pi(ux/M+vy/N)]$$

直接进行一个$N \times N$的2-D傅里叶变换需要 ? 次复数乘法运算和 ? 次复数加法运算

1-D：复数乘法和加法的次数都正比于N^2

$$F(u)=\frac{1}{N}\sum_{x=0}^{N-1}f(x)\exp[-\mathrm{j}2\pi ux/N] \qquad u=0,\ 1,\ \cdots,\ N-1$$

快速傅里叶变换（FFT）：

将复数乘法和加法的次数减少为正比于$N \log_2 N$

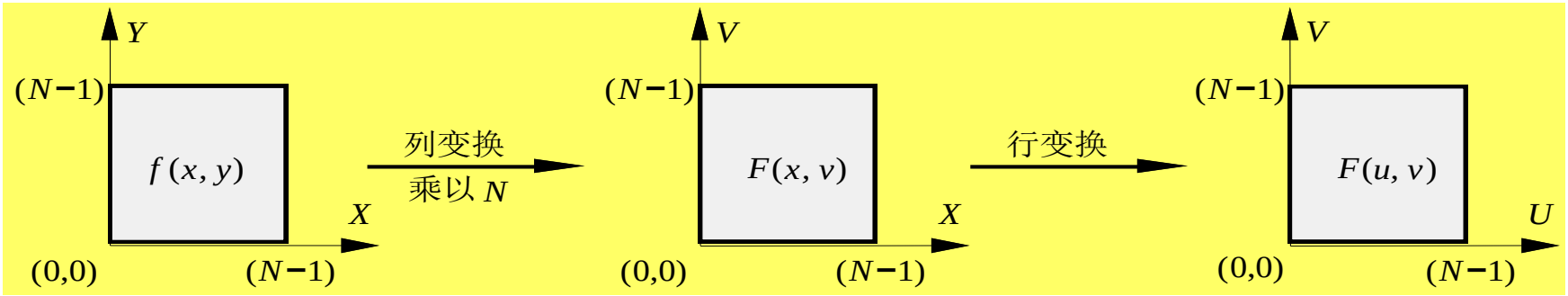
快速傅里叶变换 (FFT)

2D → 2次1D：

$$F(x,v)=N\left[\frac{1}{N}\sum_{y=0}^{N-1}f(x,y)\exp[-j2\pi vy/N]\right]$$

$$F(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}F(x,v)\exp[-j2\pi ux/N]$$

$O(N^4)$减为$O(N^2)$



FFT：复数乘法次数由N^2减少为$(N\log_2 N)/2$

复数加法次数由N^2减少为 $N\log_2 N$

快速傅里叶变换 (FFT)

- **"An algorithm for the machine calculation of complex Fourier series", Math. Comput. 19, 297–301, 1965.**
  - 后来发现，只是重新发明了高斯在1805年就已经提出的算法
  - John Wilder Tukey (1915-2000)
    - Princeton / Bell Labs
  - James William Cooley (1926-)
    - Columbia / IBM
- **Cooley-Tukey算法**
  - 最常见的FFT算法。这一方法以分治法为策略递归地将长度为N = N1N2的DFT分解为长度分别为N1和N2的两个较短序列的DFT

FFT算法的基本思想：

$$F(u)=N\left[\frac{1}{N}\sum_{x=0}^{N-1}f(x)\exp[-\mathrm{j}2\pi ux/N]\right]$$

两个特性：

1）系数 $w_N^{nk}=e^{-j\frac{2\pi}{N}nk}$ 是一个周期函数，它的周期性和对称性可用来改进运算，提高计算效率。

例

$$w_N^{n(N-k)}=w_N^{k(N-n)}=w_N^{-nk}$$

又如 $w_N^{N/2}=-1,$ 因此 $w_N^{(k+N/2)}=-w_N^k$

利用这些周期性和对称性，使DFT运算中有些项可合并；

2）利用 w_N^{nk} 的周期性和对称性，把长度为N点的大点数的DFT运算依次分解为若干个小点数的DFT。因为DFT的计算量正比于N^2，N小，计算量也就小。

FFT算法正是基于这样的基本思想发展起来的。它有多种形式，但基本上可分为两类：**时间抽取法**和**频率抽取法**。

按时间抽取的FFT（N点DFT运算的分解）

先从一个特殊情况开始，假定N是2的整数次方，

$N=2^M$，M：正整数

首先将序列$x(n)$分解为两组，一组为偶数项，一组为奇数项，

$$\begin{cases} x(2r)=x_1(r) \\ x(2r+1)=x_2(r) \end{cases} \qquad r=0,1,\ldots,N/2-1$$

DFT运算：

$$F(u)=N\left[\frac{1}{N}\sum_{x=0}^{N-1}f(x)\exp[-\mathrm{j}2\pi ux/N]\right]$$

$$x(k)=DFT[x(n)]=\sum_{n=0}^{N-1}x(n)w_N^{nk}$$

$$=\sum_{\text{偶数}n=0}^{N-2}x(n)w_N^{nk}+\sum_{\text{奇数}n=1}^{N-1}x(n)w_N^{nk}$$

$$=\sum_{r=0}^{N/2-1}x(2r)w_N^{2rk}+\sum_{r=0}^{N/2-1}x(2r+1)w_N^{(2r+1)k}$$

$$=\sum_{r=0}^{N/2-1}x\ (2r)w_N^{2rk}+w_N^k\sum_{r=0}^{N/2-1}x(2r+1)w_N^{2rk}$$

因为 $$w_N^{2n} = e^{-j\frac{2\pi}{N}2n} = e^{-j\frac{2\pi}{N/2}n} = w_{N/2}^{n}$$

故 $$X(k) = \sum_{r=0}^{N/2-1} x(2r) W_{\frac{N}{2}}^{rk} + W_N^k \sum_{r=0}^{N/2-1} x(2r+1) W_{\frac{N}{2}}^{rk}$$

$$= G(k) + W_N^k H(k)$$

其中

$$G(k) = \sum_{r=0}^{N/2-1} x(2r) W_{\frac{N}{2}}^{rk} \qquad H(k) = \sum_{r=0}^{N/2-1} x(2r+1) W_{\frac{N}{2}}^{rk}$$

注意到，H（k），G（k）有N/2个点，即k=0，1，…，N/2-1，还必须应用系数 w^k_N 的周期性和对称性

表示 X（k）的 N/2 ～N-1点：

由

$$W_{N/2}^{r(N/2+k)} = W_{N/2}^{rk}$$

得：

$$W_N^{(k+\frac{N}{2})} = -W_N^k$$

$$X(k+\frac{N}{2}) = G(k) - W_N^k H(k), \qquad k = 0,1,\cdots\frac{N}{2}-1$$

可见，一个N点的DFT被分解为两个N/2点的DFT，这两个N/2点的DFT再合成为一个N点DFT.

$$X(k)=G(k)+W_N^k H(k), \qquad k=0,1,\cdots,\frac{N}{2}-1$$

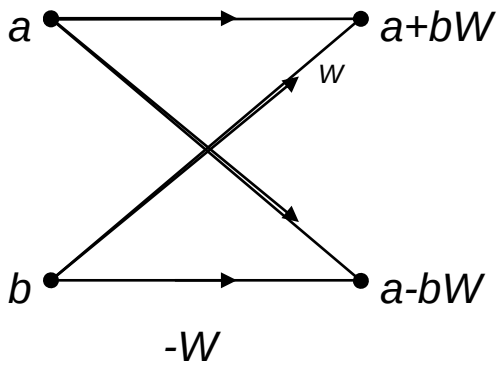
$$X(k+\frac{N}{2})=G(k)-W_N^k H(k), \qquad k=0,1,\cdots\frac{N}{2}-1$$

依此类推，G(k)和H(k)可以继续分下去，这种按时间抽取算法是在输入序列分成越来越小的子序列上执行DFT运算，最后再合成为 N点的DFT。

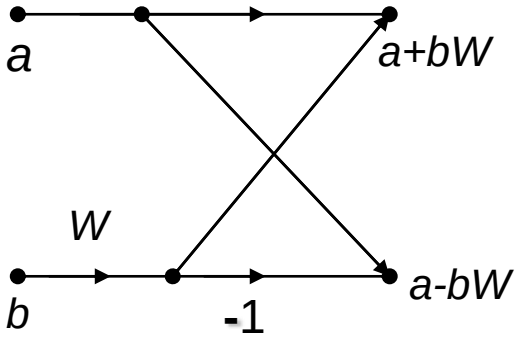
蝶形图

将$G(k)$和$H(k)$ 合成$X(k)$运算可归结为：

$$\begin{cases} a - bW \\ a + bW \end{cases}$$



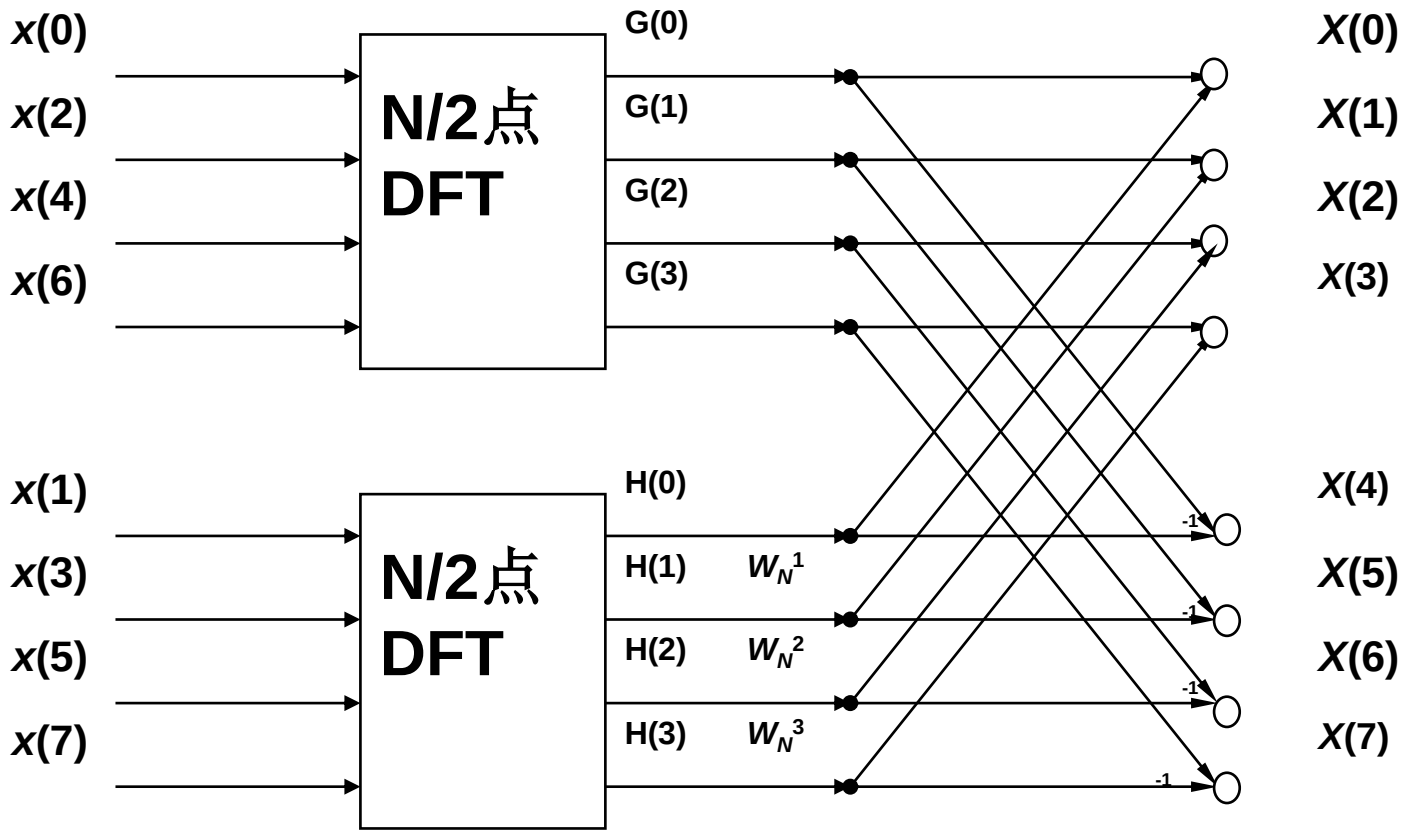
(a)



(b)

仅需一次乘法、两次加减法

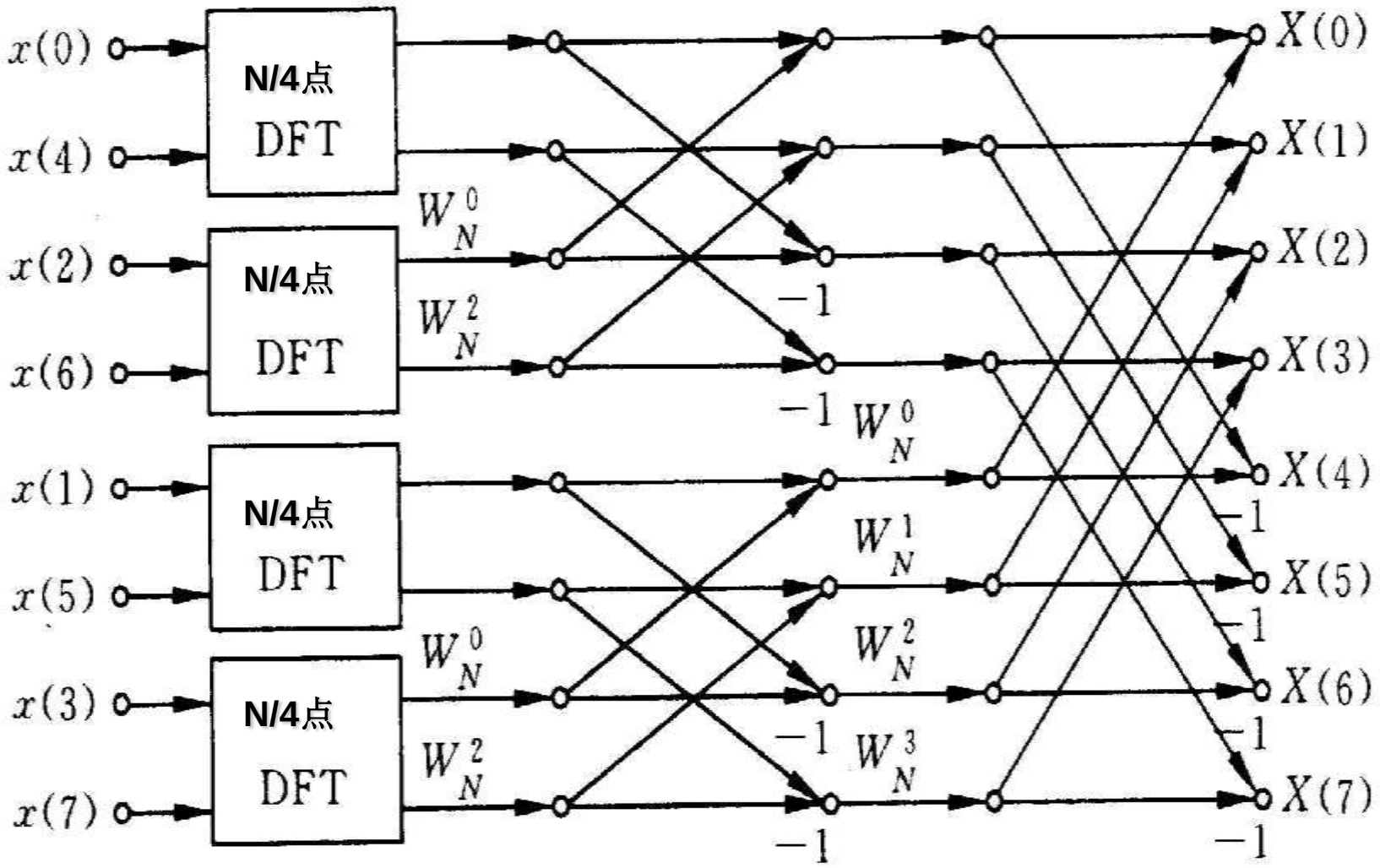
$N=2^3=8$ 的例子



两个4点DFT组成8点DFT

按照这个办法，继续把N/2用 2 除，由于$N=2^M$，仍然是偶数，可以被 2 整除，因此可以对两个N/2点的DFT再分别作进一步的分解。即对{$G(k)$}和{$H(k)$}的计算，又可以分别通过计算两个长度为N/4=2点的DFT，进一步节省计算量，见图。这样，一个 8 点的DFT就可以分解为四个 2 点的DFT。

由四个2点DFT组成8点DFT



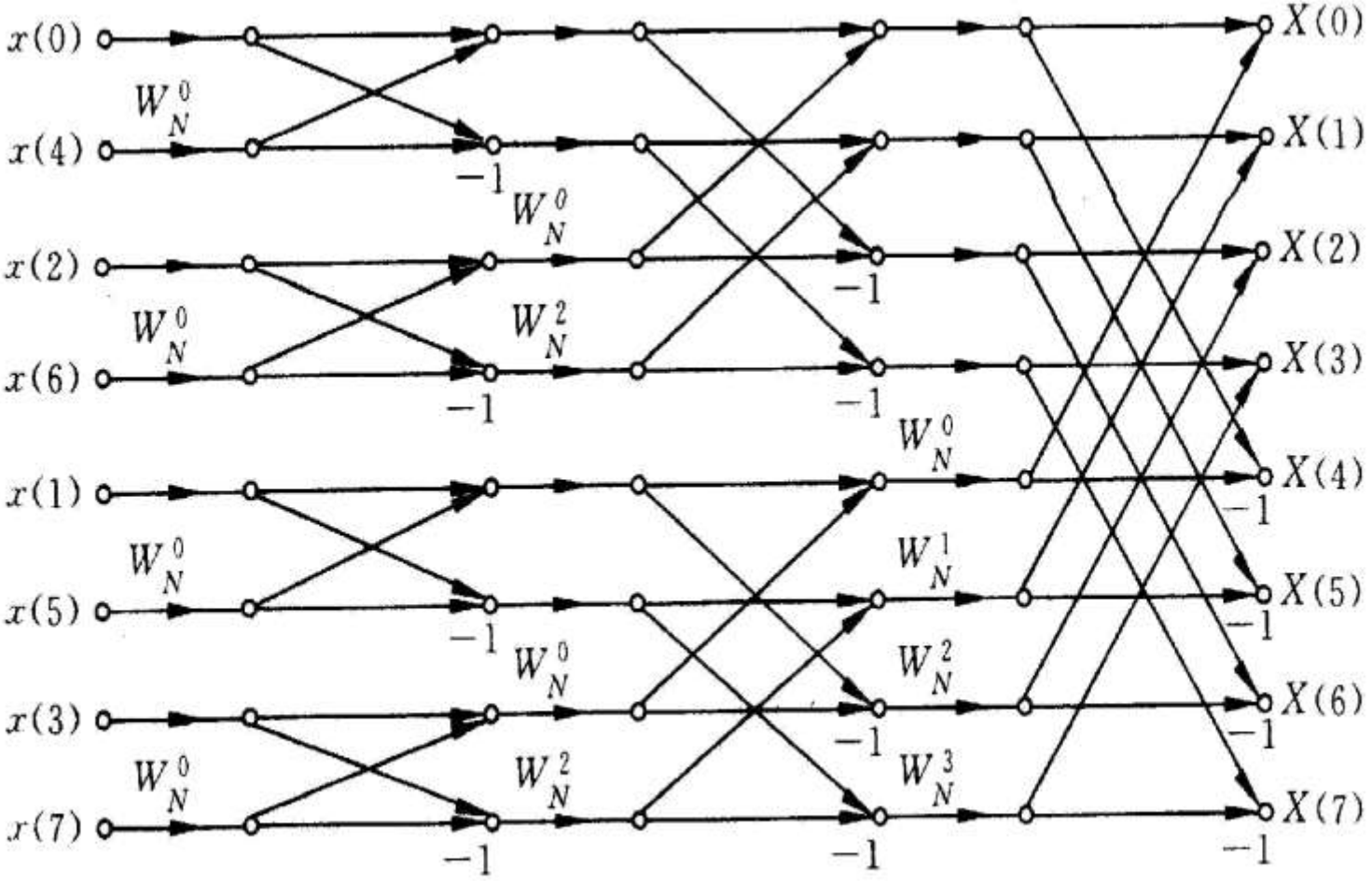
最后剩下的是2点DFT，它可以用一个蝶形结表示：

$$X(0)=x(0)+W_2^0x(1)=x(0)+W_N^ox(1)$$

$$X(1)=x(0)+W_2^1x(1)=x(0)-W_N^ox(1)$$

这样，一个8点的完整的按时间抽取运算的流图

由于这种方法每一步分解都是按输入时间序列是属于偶数还是奇数来抽取的，所以称为“**按时间抽取法**”或“**时间抽取法**”。



按时间抽取的8点FFT

蝶形运算复杂度分析

对于$N=2^M$，总是可以通过M次分解最后成为2点的DFT运算。这样构成从x(n)到X(k)的M级运算过程。从上面的流图可看到，每一级运算都由N/2个蝶形运算构成。因此每一级运算都需要N/2次复乘和N次复加，这样，经过时间抽取后M级运算总共需要的运算：

复乘 $\frac{N}{2} \bullet M = \frac{N}{2} \log_2 N$

复加 $N \bullet M = N \log_2 N$

而直接运算时则与N^2成正比。

其他问题

- **任意基数的FFT算法**
  - 补零
  - N为合数的FFT (i.e. N=PQ)

- **IFFT: IDFT的快速运算**
- **实数序列的FFT**