Fast Fourier Transform

湖南师大附中：骆轩源

February 28, 2014

Contents

1 前言

选择讲这个东西主要是我最近也在想一些与FFT有关的问题，我还记得我高一刚刚学FFT所带的心态是，哇这个东西好高端，一定要学会这样就可以到处装酷了。但是一年以后当我再想写FFT的时候，对一些地方的正负号总是纠结不定。这使我感觉到理解一个算法为什么是对的，和弄清楚他什么会想到这些步骤，弄清楚他如何构建这样一个思路的差距是很大的。

今天我是本着与各位高手交流的目的，与大家一起交流一下对FFT理解，本文的所有证明都是我自己给出的，当局者迷，我在局中就难以发现漏洞，还望各位高手指教。

2 多项式

2.1 定义

多项式定义为某个代数域上，关于变量x的形式和：

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

2.2 次数界

若$A(x)$的最高次数的次数为k，则所有严格大于k的整数都是$A(x)$的次数界。

2.3 多项式的系数表示

对于次数界为n的多项式，可以用一个n维向量:

$$< a_0, a_1 .... a_{n-1} >$$

表示多项式：

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

2.4 多项式的点值表示

用n个点：

$$< (x_0, y_0), (x_1, y_1), ...(x_{n-1}, y_{n-1}) >$$

其中

$$y_i = \sum_{j=0}^{n-1} a_j x_i^j$$

来表示多项式

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

其中要求x_i互不相同。

2.5 不同表示的作用

对于一个多项式我们往往更想知道的是其系数表示，但是在运算中，往往会使用点值表示。我们先来讨论复数域的情况，比如两个次数界为n的多项式进行乘法：如果用系数表示，复杂度是$O(n^2)$的，但是如果使用点值表示，复杂度是$O(n)$的。

所以我们通常情况下是将多项式的系数表示转变成点值表示(DFT),然后在点值表示下进行高效的运算，然后再将点值表示转变成系数表示(逆DFT)

2.6 DFT与逆DFT的良定义

我们定义DFT为一种运算，它能把**任意**一个次数界为n的多项式系数表示A(x)，变成该多项式的点值表示D(x)。也即

$$DFT(A(x)) = D(x)$$

定义逆DFT(DFT^{-1})为一种运算，它能把**任意**一个次数界为n的点值表示D(x)变成次数界为n的系数表示A(x)。也即：

$$DFT^{-1}(D(x)) = A(x)$$

现在我们要关心的问题是：我们在使用点值表示进行运算后还要逆DFT回去。所以一个良定义的DFT与逆DFT运算往往要满足：

1. 对于任意的一个点值表示D(x),仅存在一个系数表示A(x)，使得

   $$DFT(A(x)) = D(x)$$

   而且需满足

   $$DFT^{-1}(D(x)) = A(x)$$

2. 假设对两个多项式$A1(x)A2(x)$的点值表示$D1(x)D2(x)$进行运算（此处用加法举例），则一定有：

   $$DFT^{-1}(D1(x) + D2(x)) = A1(x) + A2(x)$$

为了方便起见，这里提出一个条件1的充分条件:

当$DFT(x)$与$DFT^{-1}(x)$满足对于任意的系数表示$A(x)$，都有

$$DFT^{-1}(DFT(A(x))) = A(x)$$

的时候，该DFT与逆DFT函数满足条件1。

证明：

先证明对于任意的一个点值表示$D(x)$,都仅存在一个满足

$$DFT(A(x)) = D(x)\text{的}A(x)$$

不妨使用反证法，假设存在两个不同的系数表示$A1(x), A2(x)$,满足

$$DFT(A1(x)) = DFT(A2(x)) = D(x)$$

则由前提条件可得：

$$DFT^{-1}(DFT(A1(x))) = DFT^{-1}(D(x)) = A1(x)$$

,

$$DFT^{-1}(DFT(A2(x))) = DFT^{-1}(D(x)) = A2(x)$$

所以得到$A1(x) = A2(x)$，矛盾。

关于第二点$DFT^{-1}(D(x)) = A(x)$则显然成立。

3 复数域上的DFT与DFT^{-1}

DFT与逆DFT的函数设计，主要取决于选择的点值表示的横坐标x向量。我们设一个x向量为$< x_0, x_1, x_2....x_{n-1} >$。（其中x互不相同）

构造DFT过程为计算：

$$yi = \sum_{j=0}^{n-1} a_j x_i^j : 0 \le i \le n-1$$

构造逆DFT过程为求解方程组（a为未知数）：

$$yi = \sum_{j=0}^{n-1} a_j x_i^j : 0 \le i \le n-1$$

条件2是显然成立的,现在来说明条件1成立。

即证明对于任意的系数表示A(x)，都有

$$DFT(DFT^{-1}(A(x))) = A(x)$$

套用到这个具体实例上，就是证明逆DFT的方程组的解唯一（因为A(x)已经是一组解了，所以只要没有多解结论就成立）。

我们将方程组写成矩阵乘积的形式：

$$\begin{pmatrix} 1 & 1 & 1 & ... & ... & 1 \\ 1 & x_1 & x_1^2 & ... & ... & x_1^{n-1} \\ 1 & ... & ... & ... & ... & ... \\ 1 & x_{n-1} & x_{n-1}^2 & ... & ... & x_{n-1}^{n-1} \end{pmatrix} * \begin{pmatrix} a_0 \\ a_1 \\ ... \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ ... \\ y_{n-1} \end{pmatrix} \tag{1}$$

左边的矩阵为范德蒙德矩阵，由线性代数知识可知当x 互不相同的时候该矩阵的行列式值不为0，也就是说该矩阵是非奇异的。我们不妨假设有两组不同的解a,b都满足该方程，则必有(a-b)为该范德蒙德矩阵的一个空向量，最终得出该矩阵为奇异矩阵，与前提矛盾。所以在复数域上进行以上定义的DFT过程与逆DFT过程是合法的。

但是我们现在想高效的计算DFT与逆DFT。于是我们就需要在x的选择上做文章了。

3.1 复数域上$O(n\lg n)$的DFT

FFT算法的核心思想在于分治，所以x的选择也是为了分治服务的。

一种合法的x向量的选择为$< w_n^0, w_n^1, ...w_n^{n-1} >$ 其中w_n为n次单位复根。现在先来说明一下单位复根。

数学上一般定义$w_n = e^{\frac{2\pi i}{n}}$,其中$i$为$\sqrt{-1}$由欧拉公式可知：

$$e^{iu} = \cos u + i\sin u$$

3.1.1 定理1：相消定理

对于任意整数$n \geq 0, k \geq 0, d \geq 0$,有：

$$w_{dn}^{dk} = w_n^k$$

证明：

$$w_{dn}^{dk} = e^{\frac{2\pi dki}{dn}} = e^{\frac{2\pi ki}{n}} = w_n^k$$

3.1.2 推论：折半定理

对于任意偶数$n \geq 0$,和偶数$k \geq 0$，有

$$w_n^k = w_{\frac{n}{2}}^{\frac{k}{2}}$$

证明略去。（由定理1易得）

现在我们考虑如何使用这两个性质来进行DFT。

首先我们可以将次数界n变成某个2的次幂。

然后我们不妨令：

$$F_n = \begin{pmatrix} 1 & 1 & 1 & ... & ... & 1 \\ 1 & w_n^1 & w_n^2 & ... & ... & w_n^{n-1} \\ 1 & ... & ... & ... & ... & ... \\ 1 & w_n^{n-1} & w_n^{2(n-1)} & ... & ... & w_n^{(n-1)(n-1)} \end{pmatrix} \tag{2}$$

则DFT(A(x))实际上是求：$y = F_n * < a_0, a_1, a_2....a_{n-1} >$ 我们令：

$$A1 =< a_0, a_2, a_4....a_{n-2} >$$

$$A2 =< a_1, a_3, a_5....a_{n-1} >$$

$$Y1 = F_{\frac{n}{2}} * A1$$

$$Y2 = F_{\frac{n}{2}} * A2$$

如果我们能够做到O(n)的时间内合并Y1,Y2得到y，那么DFT的时间复杂度T(n)就满足：

$$T(n) = 2T(n/2) + O(n) = O(n\lg n)$$

进一步探究可得：

$$Y1[i] = a_0 + w_{\frac{n}{2}}^{i} a_2 + w_{\frac{n}{2}}^{2i} a_4 .....w_{\frac{n}{2}}^{(\frac{n}{2}-1)i} a_{n-2}$$

$$Y2[i] = a_1 + w_{\frac{n}{2}}^{i} a_3 + w_{\frac{n}{2}}^{2i} a_5 .....w_{\frac{n}{2}}^{(\frac{n}{2}-1)i} a_{n-1}$$

当$0 \le i \le n/2$时有:

$$y[i] = a_0 + w_n^i a_1 + w_n^{2i} a_2 .....w_n^{(n-1)i} a_{n-1}$$

分开计算：

$$y[i] = (a_0 + w_n^{2i} a_2 + w_n^{4i} a_4 .....w_n^{(n-2)i} a_{n-2}) + (w_n^i a_1 + w_n^{3i} a_3 + w_n^{5i} a_5 .....w_n^{(n-1)i} a_{n-1})$$

配合折半定理，最后可以得到：

$$y[i] = Y1[i] + w_n^i Y2[i]$$

当$\frac{n}{2} \le i \le n-1$时有：

$$y[i] = a_0 + w_n^i a_1 + w_n^{2i} a_2 .....w_n^{(n-1)i} a_{n-1}$$

同样可以分开计算：

$$y[i] = (a_0 + w_n^{2i} a_2 + w_n^{4i} a_4 .....w_n^{(n-2)i} a_{n-2}) + (w_n^i a_1 + w_n^{3i} a_3 + w_n^{5i} a_5 .....w_n^{(n-1)i} a_{n-1})$$

我们考虑如何将w_n^{ki}（其中k为偶数）变成$Y1$里面的东西：

令$j = i - \frac{n}{2}$ 则有

$$w_n^{ki} = w_n^{k(j+\frac{n}{2})} = w_n^{kj} w_n^{k\frac{n}{2}}$$

又因为$w_n^{\frac{n}{2}} = w_2 = -1$ 所以

$$w_n^{ki} = w_n^{kj} = w_n^{k(i-\frac{n}{2})}$$

同理可以推出当k为奇数的时候有：

$$w_n^{ki} = -w_n^{k(i-\frac{n}{2})}$$

所以有

$$Left = (a_0 + w_n^{2(i-\frac{n}{2})}a_2 + w_n^{4(i-\frac{n}{2})}a_4.....w_n^{(n-2)(i-\frac{n}{2})}a_{n-2}) = Y1[i - \frac{n}{2}]$$

$$Right = -w_n^{i-\frac{n}{2}}(a_1 + w_n^{2(i-\frac{n}{2})}a_3 + w_n^{4(i-\frac{n}{2})}a_5.....w_n^{(n-2)(i-\frac{n}{2})}a_{n-1}) = -w_n^{i-\frac{n}{2}}Y2[i - \frac{n}{2}]$$

所以综合看来：

$$y[i] = Y1[i - \frac{n}{2}] - w_n^{i-\frac{n}{2}}Y2[i - \frac{n}{2}]$$

这些式子看上去很复杂，但是当你看到这些式子的时候你应该感到高兴。因为可以使用Y1与Y2组合出y啦，就说明了DFT可以被分治计算。

如果说要用一个思路来归纳刚才的式子变形，那就是想尽一切办法将y的式子分成能由Y1，和Y2里面的东西组成的，比如当$i \geq \frac{n}{2}$时，我们就需要将w_n^i降次等等。

看似复杂的操作，实际上核心就只有一个“分治”，只需知道它是把原多项式分成了哪两个部分分开计算。剩下的完全可以由自己推出。所以我说理解一个东西并不是很难，但是真正要掌握它，就需要想尽办法理清思路，就好像打仗一样，总指挥只是制定计划要打下那几座城市，他知道打下这些城市就可以达到他的战略目标，但是如果连这些城市具体要怎么打他都去想的话，他的大脑一定会乱的，而且这样也是没有必要的。

3.2 复数域上$O(n\lg n)$的逆DFT

回顾我前面说的逆DFT：解n维方程组。

因为已经证明了该方程组的解唯一，所以我们只要构造出一个合法的解就行了。怎么求这个解呢？可以用高斯消元，恩，只是复杂度有点小高($O(n^3)$)。看来还是要用一下单位复根的特殊性质的，但是单纯从解方程来看确实单位复根也没什么好性质。

那么怎么办呢？假设你是一个失意的科学家，然后工作了这么多年都没想出一个靠谱的算法，好不容易有了一个灵感，怎么就会在逆DFT 这里退缩呢？既然解方程不行，为了生存，我手动构造一个Fn的逆也要把它解出来！

怎么构造F_n的逆F_n^{-1}呢？即构造F_n^{-1}让$F_nF_n^{-1}=I$

观察发现F_n的第i行(从0开始，下面的也是）为$<1,w_n^i,w_n^{2i}.....w_n^{(n-1)i}>$

为了让F_n的第i行乘F_n^{-1}的第i列为1, 首先我们应该想到的是1是个实数，w_n^{ix}是个不三不四的东西，那么我们先要把它变成实数再说。不妨构造F_n^{-1}的第i列为：

$<1,w_n^{-i},w_n^{-2i},w_n^{-3i}.....w_n^{-(n-1)i}>$

则两个向量相乘为n，不管了把F_n^{-1}的第i列的每个元素先乘个$\frac{1}{n}$再说，改进之后第i列变成：

$<\frac{1}{n},\frac{1}{n}w_n^{-i},\frac{1}{n}w_n^{-2i},\frac{1}{n}w_n^{-3i}.....\frac{1}{n}w_n^{-(n-1)i}>$

然后怀揣着忐忑的心情去试试用F_n的第i行乘一下F_n^{-1}的第j列(不为i)看看是不是等于0，设答案为ans

$$ans=\frac{1}{n}\sum_{k=0}^{n-1}w_n^{k(i-j)}$$

可以把ans看成一个等比数列求和,因为公比w_n^{i-j}显然不为1，而且等比数列求和公式对于复数也是适用的，所以

$$ans=\frac{1-w_n^{n(i-j)}}{1-w_n^{i-j}}=0$$

居然等于0！太好了！这样我就可以算出a了：

$$\begin{pmatrix}1 & 1 & 1 & ... & ... & 1\\ 1 & w_n^{-1} & w_n^{-2} & ... & ... & w_n^{-(n-1)}\\ 1 & ... & ... & ... & ... & ...\\ 1 & w_n^{-(n-1)} & w_n^{-2(n-1)} & ... & ... & w_n^{-(n-1)(n-1)}\end{pmatrix}*\begin{pmatrix}\frac{1}{n}y_0\\ \frac{1}{n}y_1\\ ...\\ \frac{1}{n}y_{n-1}\end{pmatrix}=\begin{pmatrix}a_0\\ a_1\\ ...\\ a_{n-1}\end{pmatrix}\tag{3}$$

惊喜还不止这些，我们发现逆DFT与DFT操作所乘的矩阵神一般的相似，于是我们就可以用类似DFT的方法分治计算啦。

3.3 复数域下FFT思想归纳

我们在FFT之前首先做的事情，就是构造DFT与逆DFT过程。回顾我们的构造：

DFT操作，代入n个变量然后求值（这个过程又称为插值）

逆DFT操作，根据代入的坐标反解系数。

然后我们证明了该DFT与逆DFT的良定义。用到的方法就是证明一个系数被DFT再逆DFT后能够回来。

这是在我们还没有想出使用n次单位复根的情况下制定的DFT与逆DFT。

在我们已经了解了传统的FFT是怎么操作以后，我们可以简单的将DFT与逆DFT表示为：

DFT操作即Fn乘系数向量a。逆DFT操作即Fn^{-1}乘点值纵坐标向量y,然后同样也可以证明DFT与逆DFT的良定义。

下面类似地，引入另一种代数域下的DFT与逆DFT做法，作为思想的回顾。

4 模p意义下的快速傅里叶变换

我们知道有一类组合问题是可以用生成函数（母函数）来解决的，这时候就需要涉及到多项式的乘法，而且往往我们只需要知道答案对一个数取模的结果（越说越像OI），有时候模意义下的运算可以使得减小单位复根运算带来的精度误差。

当然我这里引入这个方法，也不是为了要说它有多么多么实用，只是为了巩固思想。

假设我们在次数界n（n为2的幂，当然不是2的幂也有办法使它变成2的幂，而且得到数还是O(n)级别的）且模p的意义下进行多项式运算。而且$p = kn + 1$，其中$k = O(\lg n)$(注意这里只是说明k的大小级别)。然后我们依旧来探讨该代数域下的FFT。

首先要想到的就是构造DFT与逆DFT。我们尽量地将FFT做得类似于复数域下的FFT。

既然是这样，那么我们首先要找到w_n的替代品。不妨设模p的乘法群的生成元为g,那么我们选用g^k代替w_n作为主n次单位根。为什么呢？因为$k = \frac{(p-1)}{n}$

所以$g^k = g^{\frac{(p-1)}{n}}$,类似地我们在这里使用$g_n$来表示主n次单位根。

现在来证明原来w_n有的对FFT有用的性质它都有。

4.1 相消定理

对于任意的整数$n \geq 0, k \geq 0, d \geq 0$，有：

$$g_{dn}^{dk} = g_n^k$$

证明：

$$g_{dn}^{dk} = g^{\frac{(p-1)dk}{dn}} = g^{\frac{(p-1)k}{n}} = g_n^k$$

4.2 折半定理

对于任意偶数$n \geq 0$,和偶数$k \geq 0$，有

$$g_n^k = g_{\frac{n}{2}}^{\frac{k}{2}}$$

证明略去。（由相消定理易得）

4.3 一些特殊值(mod p 意义下)

$$g_2 = -1$$

$$g_n^0 = g_n^n = 1$$

剩下来就一个求和定理了（就那个等比数列求和的），如果也满足了，那么就跟复数域的没什么不同了。

如果所有的定理都要我来证明，那么你们就没事做了是吧。

我们定义DFT运算为（求y）

$$\begin{pmatrix} 1 & 1 & 1 & ... & ... & 1 \\ 1 & g_n^1 & g_n^2 & ... & ... & g_n^{n-1} \\ 1 & ... & ... & ... & ... & ... \\ 1 & g_n^{n-1} & g_n^{2(n-1)} & ... & ... & g_n^{(n-1)(n-1)} \end{pmatrix} * \begin{pmatrix} a_0 \\ a_1 \\ ... \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ ... \\ y_{n-1} \end{pmatrix} \tag{4}$$

这里的乘法加法都是模意义下的。

定义逆DFT运算为(求a)：

$$\begin{pmatrix} 1 & 1 & 1 & ... & ... & 1 \\ 1 & g_n^{-1} & g_n^{-2} & ... & ... & g_n^{-(n-1)} \\ 1 & ... & ... & ... & ... & ... \\ 1 & g_n^{-(n-1)} & g_n^{-2(n-1)} & ... & ... & g_n^{-(n-1)(n-1)} \end{pmatrix} * \begin{pmatrix} \frac{1}{n}y_0 \\ \frac{1}{n}y_1 \\ ... \\ \frac{1}{n}y_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ ... \\ y_{n-1} \end{pmatrix} \tag{5}$$

注意这里的$\frac{1}{n}$,表示n在 mod p意义下的逆元，因为有$\gcd(n,p) = 1$,所以逆元一定存在。因为这里的g_n同样满足相消定理，所以一样可以被分治计算。

5 总结

原来我也总是觉得FFT算法是那么的厉害，甚至还觉得有点复杂，其实主要是里面涉及到的一些高等一点数学知识的原因。抛开数学，它也不过是一个很普通的算法，它的精髓在于分治。

但是对于设计出这个算法的人，我们是要仰望的，要设计出这样一个算法，首先要坚定地认为一定存在一个DFT过程是可以被分治计算的，而且逆DFT也要能被分治计算，这也确实是一个很美妙的巧合，也许是设计者致力于寻找然后偶然发现，也许是有更加深层次的理论使得这样的巧合可以被轻松的分析出来。

总之，这里想表达的是主要是我对FFT算法思想的思考，而不是算法的实战意义。想揭示的是算法背后的线索。也有可能我想的是错的，也有可能背后还有更深层次的东西我不知道，但是不管怎么样，也是一个尝试吧。

人类总是在不断地归纳，不断地推翻自己的归纳中不断进步。