Ichimei's blog (http://www.gatevin.moe/)

ねえ、あなたは何色になりたい？

FFT算法学习笔记

Posted on July 14, 2015 (http://www.gatevin.moe/acm/fft%e7%ae%97%e6%b3%95%e5%ad%a6%e4%b9%a0%e7%ac%94%e8%ae%b0/) by gatevin (http://www.gatevin.moe/author/admin/)

写在前面

关于学习FFT算法的资料个人最推荐的还是算法导论上的第30章(第三版), 多项式与快速傅里叶变换, 基础知识都讲得很全面

这篇文章即是博主在阅读FFT算法的相关学习资料之后的一些总结, 作为日后复习查阅的材料来源, 同时也可以作为读者学习FFT算法的一份资料

FFT算法基本概念

$FFT(FastFourierTransformation)$ 即快速傅里叶变换, 是离散傅里叶变换的加速算法, 可以在$O(nlogn)$的时间里完成DFT, 利用相似性也可以在同样复杂度的时间里完成逆DFT

$DFT(DiscreteFourierTransform)$ 即离散傅里叶变换, 这里主要就是多项式的系数向量转换成点值表示的过程

在ACM-ICPC竞赛中, FFT算法常被用来为多项式乘法加速, 即在$O(nlogn)$复杂度内完成多项式乘法, 当然实际应用不仅仅限于这些, 时常出现需要构造多项式相乘来进

行计数的问题, 也需要用FFT算法来解决, 相关的几个问题在本文中也会提及

FFT算法需要的基础数学知识

多项式相关：

多项式相关的定义：一个以x为变量的**多项式**定义在代数域F上, 将函数$A(x)$表示为形式和

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

, 称a_j为系数, 所有系数属于代数域F, 如果一个多项式的最高非零系数是a_k, 那么成这个多项式的**次数**是k, 记做$degree(A) = k$, 任何一个严格大于一个多项式次数的整数都是该多项式的**次数界**

关于多项式的加法和乘法相信看到这篇博客的读者都会最基本的中学的算法, 在计算的时候, 如果采用传统的中学的计算方法, 多项式加法的时间复杂度是$O(n)$, 乘法的时间复杂度是$O(n^2)$ (n是两个多项式A和B的次数).

多项式的表示

在平常的学习中, 最常见的是多项式的**系数表达方式**,

次数界为n的多项式$A(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$的系数表达为一个由系数组成的向量$a = (a_1, a_2, \ldots a_{n-1})$

但是多项式还有一个比较常用的表示方法, 即多项式的**点值表达方式**

一个次数界为n的多项式$A(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$的点值表达就是一个由n个点对组成的集合

$$\{(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}$$

使得对任意的整数$k = 0, 1, .. n-1$, x_k各不相同, 且$y_k = A(x_k)$

关于点值表达的正确性证明：

对于任意n个点值对组成的集合$\{(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}$, 如果存在一个次数界为n的多项式$A(x)$过这n个点, 那么

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix}$$

最左边这个n*n的矩阵称为范德蒙德矩阵, 可以用数学归纳法证明它的行列式值为

$$\prod_{0 \le j < k \le n-1} (x_k - x_j)$$

当x_k两两不相同时明显这个行列式的值不为0, 该矩阵可逆, 于是存在唯一解, 所以多项式的点值表达是合理的

相应的通过n个点的坐标直接确定多项式各个系数的值的方法是存在的, 感兴趣的读者可以查询拉格朗日公式的相关资料, 利用拉格朗日插值公式可以在$O(n^2)$的时间复杂度内得到多项式的系数表达, 这个也是算法导论中的一个习题

点值表达方式下多项式的乘法, 不难发现, 如果多项式$A(x), B(x)$的点值表示分别是

$$\{(x_0, y_{a_0}), (x_1, y_{a_1}), \ldots, (x_{n-1}, y_{a_{n-1}})\}$$

和

$$\{(x_0, y_{b_0}), (x_1, y_{b_1}), \ldots, (x_{n-1}, y_{b_{n-1}})\}$$

那么如果多项式$C(x) = A(x)B(x)$, 那么$C(x)$的点值表达是

$$\{(x_0, y_{a_0}y_{b_0}), (x_1, y_{a_1}y_{b_1}), \ldots, (x_{n-1}, y_{a_{n-1}}y_{b_{n-1}})\}$$

卷积

对于两个多项式的系数向量$a = (a_0, a_1, .., a_{n-1})$和$b = (b_0, b_1, .., b_{n-1})$, 两个多项式相乘得到的多项式的系数向量$c = (c_0, c_1, .., c_{2n-2})$满足$c_j = \Sigma_{k=0}^{j} a_k b_{j-k}$, 称系数向量c是输入向量a和b的卷积, 记作$c = a \otimes b$

在简单的多项式乘法的计算方法中, 每一个多项式的系数都通过系数表示方式下卷积的方式来进行计算, 时间复杂度是$O(n^2)$, 但是FFT是先将多项式的从系数表示法转换成点值表示法(可以在$O(nlogn)$的时间复杂度下完成, 也就是加速的DFT变换, 然后在点值表示法下进行乘积计算, 在$O(n)$的时间复杂度内得到结果的点值表示法, 然后进行逆DFT变换, 在$O(nlogn)$的时间复杂度下完成逆DFT变换得到系数表示法

而要理解DFT, 则需要一定复数上的数学知识：

复数相关的基础知识：

单位复数根：n次单位复数根指的是满足$\omega^n = 1$的所有复数ω, n次单位复数根刚好有n个, 他们是$e^{\frac{2k\pi}{n}i}$, 其中i是复数单位, $k = 0, 1, 2...n-1$, 在复平面上这n个根均匀的分布在半径为1的圆上, 关于复数指数的定义如下：

$$e^{ui} = cos(u) + sin(u)i$$

其中$\omega_n = e^{\frac{2i\pi}{n}}$倍称为主n次单位根(这个定义好像接下来没用到= =)

关于复数根的几个定理和引理：

消去引理： 对任何整数 $n \geq 0, k \geq 0, d \geq 0$有$\omega_{dn}^{dk} = \omega_n^k$

证明： $\omega_{dn}^{dk} = (e^{\frac{2i\pi}{dn}})^{dk} = (e^{\frac{2i\pi}{n}})^k = \omega_n^k$

一个推论： 对任意偶数 n > 0 有 $\omega_n^{\frac{n}{2}} = \omega_2 = -1$

证明：设n = 2*k那么 $\omega_n^{\frac{n}{2}} = \omega_{2k}^k = \omega_2 = e^{\pi} = cos(\pi) + sin(\pi)i = -1$

折半引理：如果n > 0是偶数, 那么n个n次单位复数根的平方的集合就是n/2个n/2次单位复数根的集合

证明：实际上这个引理就是证明了$(\omega_n^{k+\frac{n}{2}})^2 = \omega_n^{2k+n} = \omega_n^{2k}\omega_n^n = (\omega_n^k)^2$

折半引理对于采用分治对多项式系数向点值表达的转换有很大作用, 保证了递归的子问题是原问题规模的一半

求和引理：对任意整数$n \geq 1$和不能被n整除的非负整数k, 有

$$\sum_{j=0}^{n-1}(\omega_n^k)^j = 0$$

这个问题通过等比数列求和公式就可以得到：$\Sigma_{j=0}^{n-1}(\omega_n^k)^j = \frac{(\omega_n^k)^k - 1}{\omega_n^k - 1} = \frac{1^k - 1}{\omega_n^k - 1} = 0$

DFT与FFT, 以及逆DFT

DFT

在DFT变换中, 希望计算多项式A(x)在复数根$\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$处的值, 也就是求

$$y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{kj}$$

称向量$y = (y_0, y_1, \ldots, y_{n-1})$是系数向量$a = (a_0, a_1, \ldots, a_{n-1})$的离散傅里叶变换, 记为$y = DFT_n(a)$

FFT

直接计算DFT的复杂度是$O(n^2)$, 而利用复数根的特殊性质的话, 可以在$O(nlogn)$的时间内完成, 这个方法就是FFT方法, 在FFT方法中采用分治策略来进行操作, 主要利用了消去引理之后的那个推论

在FFT的策略中, 多项式的次数是2的整数次幂, 不足的话再前面补0, 每一步将当前的多项式A(x), 次数是2的倍数, 分成两个部分：

$$A^{[0]}(x) = a_0 + a_2x + a_4x^2 + \ldots + a_{n-2}x^{\frac{n}{2}-1}$$

$$A^{[1]}(x) = a_1 + a_3x^1 + a_5x^2 + \ldots + a_{n-1}x^{\frac{n}{2}-1}$$

于是就有了

$$A(x) = A^{[0]}(x^2) + xA^{[1]}(x^2)$$

那么我们如果能求出次数界是$\frac{n}{2}$的多项式$A^{[0]}(x)$和$A^{[1]}(x)$在n个n次单位复数根的平方处的取值就可以了, 即在

$$(\omega_n^0)^2, (\omega_n^1)^2, (\omega_n^2)^2, \ldots, (\omega_n^{n-1})^2$$

处的值, 那么根据折半引理, 这n个数其实只有$\frac{n}{2}$个不同的值, 也就是说, 对于每次分出的两个次数界$\frac{n}{2}$的多项式, 只需要求出其$\frac{n}{2}$个不同的值即可, 那么问题就递归到了原来规模的一半, 也就是说如果知道了两个子问题的结果, 当前问题可以在两个子问题次数之和的复杂度内解决, 那么这样递归问题的复杂度将会是O(nlogn)的,
用$a = (a_0, a_1, \ldots, a_{n-1})$表示系数向量, $y = (y_0, y_1, \ldots, y_{n-1})$表示离散变换之后的向量, 这里给出将算导上的代码翻译出来的C++代码实现(以解决算法第三版大论第三十章的一个习题, 求(0, 1, 2, 3)的DFT为例) :

```
Click To Expand Code
```

可以得到系数向量(0, 1, 2, 3)经过DFT之后是(6, -2-2i, -2, -2+2i)

在上面这个视线中需要注意的就是RecursiveFFT中的y[k] = y0[k] + w*y1[k];和y[k + (n >> 1)] = y0[k] - w*y1[k];的变化, 正是利用了对于偶数的n， 有$\omega_n^{\frac{n}{2}} = -1$成立, 得到数组y的所有值

另外有一个概念性的东西： $y_k = y_k^{[0]} + \omega_n^k y_k^{[1]}$和$y_{k+\frac{n}{2}} = y_k^{[0]} - \omega_n^k y_k^{[1]}$都成立, 在这其中将$\omega_n^k$称为**旋转因子**

在进行了DFT操作之后, 将多项式的系数表达成功转为了点值表达 接下来是如何在$O(nlogn)$的时间内完成逆DFT的问题, 通过逆DFT将点值表达还原为系数表达

逆DFT

根据DFT得到的向量y和系数向量a之间的关系, 可以用矩阵乘积的形式来表达他们之间的关系, 即$y = V_n a$, 也就是

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \omega_n^3 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \omega_n^6 & \cdots & \omega_n^{2(n-1)} \\ 1 & \omega_n^3 & \omega_n^6 & \omega_n^9 & \cdots & \omega_n^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \omega_n^{3(n-1)} & \cdots & \omega_n^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \end{bmatrix}$$

那么要将DFT变化得到的向量y还原成向量a的话, 只需要用V_n的逆矩阵V_n^{-1}乘上向量y即可, 这里需要用到一个定理

在矩阵V_n中, 不难发现对于任意的$j, k = 0, 1, \ldots, n-1$, $V_n(k, j) = \omega_n^{kj}$, 那么可以找到这样一个矩阵$V_n^{-1}$, 对于任意的$j, k = 0, 1, \ldots, n-1$, V_n^{-1}的(j, k)出的元素为$\frac{\omega_n^{-kj}}{n}$, V_n^{-1}是矩阵V_n的逆矩阵

证明如下：要证明这两个矩阵互逆, 证明其积为单位矩阵即可, 考虑两个矩阵的乘积在(j, j')出的元素, 可以发现这个元素是

$\Sigma_{k=0}^{n-1}(\frac{\omega_n^{-kj}}{n})(\omega_n^{kj'}) = \Sigma_{k=0}^{n-1}\frac{\omega_n^{k(j'-j)}}{n}$

当$j' = j$时, 这个和是1, 否则根据求和引理, 这个和是0, 故这两个矩阵互逆

那么根据这个逆矩阵可以发现要计算$DFT_n^{-1}(y)$的话, 有关系式$a_j = \frac{1}{n}\Sigma_{k=0}^{n-1} y_k\omega_n^{-kj}$比较这个式子和之前DFT里面y和a的关系式子, 可以发现只需要用ω_n^{-1}替换掉ω_n即可, 最后结果需要除以n, 所以计算逆DFT的方法和计算DFT和相似, 都可以在$O(nlogn)$的时间复杂度内解决

卷积定理

对任意两个长度为n的向量a和b, 其中n是2的幂, 有

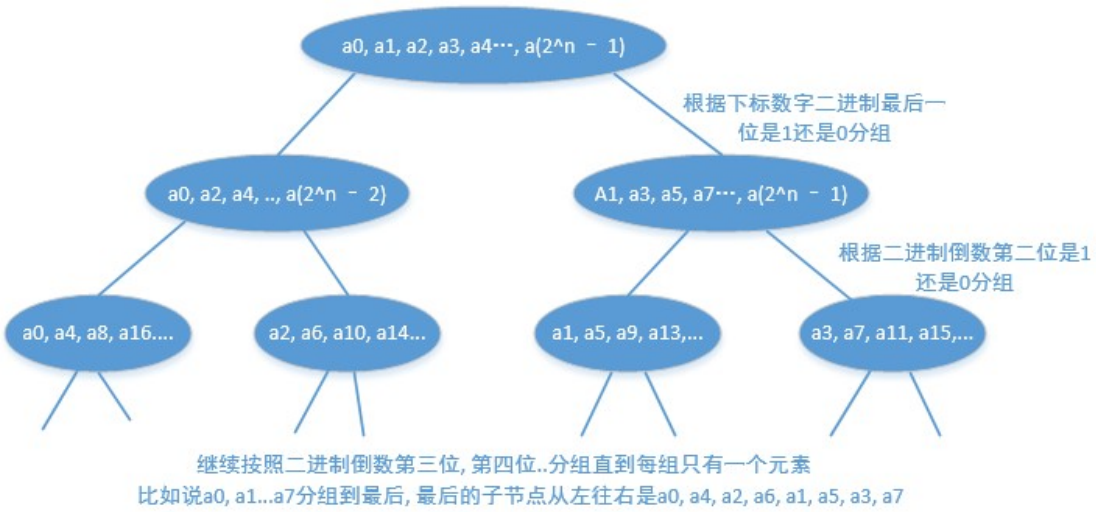
$$a \otimes b = DFT_{2n}^{-1}(DFT_{2n}(a) \cdot DFT_{2n}(b))$$

其中向量a和b用0填充, 使其长度达到2n, 并用·表示两个2n个元素组成向量的点乘(也就是每一维上的数相乘)

这个式子实际上就是多项式的系数表达在乘法时进行的卷积运算得到的结果, 等同于通过将其系数进行DFT变换变成点值表达之后相乘再换回来的过程

关于FFT算法的迭代实现

在递归实现DFT过程的FFT算法中, 我们每次将系数向量a分成两个部分利用折半引理来降低计算的规模, 可以发现在每次分组当中他们满足这样一个完全二叉树的分组(n是2的幂):



(http://www.gatevin.moe/wp-content/uploads/2015/07/FFT.png)

FFT递归系数分组

通过上图的流程可以看出, 最后一层的子节点下标的顺序实际上就是其下标转换成二进制串的倒序的字符串按照字典序排列的顺序

比如a0~a7得到的序列a0, a4, a2, a6, a1, a5, a3, a7下标的二进制是000, 100, 010, 110, 001, 101, 011, 111对应的串的倒序是000, 001, 010, 011, 100, 101, 110, 111这个

倒序的二进制刚好是0, 1, 2, 3, 4, 5, 6, 7的二进制表示, 于是我们可以在$O(nlogn)$的复杂度内得到做下面一层的下标顺序, 然后可以根据子节点的结果向上迭代得到父亲结点的值, 这样计算的话直接避免了递归, 如果直接递归的话在一些OJ上可能会造成爆栈的错误, 所以还是采用迭代的方式进行比较好.

关于迭代形式的FFT算法, C++代码实现如下(同样以求(0, 1, 2, 3)的DFT变换为例):

这段代码的话同时也进行了逆DFT, DFT和逆DFT的过程相似, 加上一个标记判断当前执行的是哪一种就行了

```
Click To Expand Code
```

通过上面这个代码的示例, FFT算法的实现基本没有什么问题了, 另外算法导论中的习题有一些很不错, 便于熟悉这一算法的很多细节, 这里就不一一提及了

经过这些学习之后, 进行一些实战演练是很有必要的, 接下来是相关习题的练习部分

相关练习:

HDU 1402 A*B Problem Plus 大整数乘法, 我的解答:戳我 (http://blog.csdn.net/gatevin/article/details/46880667)

HDU 4609 3-idiots FFT计数, 解答:戳我 (http://blog.csdn.net/gatevin/article/details/46891061)

UVALive 4671 K-neighbor Substrings FFT算字符串Hamming距离 解答:戳我 (http://blog.csdn.net/gatevin/article/details/46892383)

UVA 12298 Super Poker II FFT计数, long double, 解答: 戳我 (http://blog.csdn.net/gatevin/article/details/46897251)

URAL 1996 Cipher Massage 3 FFT + KMP 解答:戳我 (http://blog.csdn.net/gatevin/article/details/46912871)

CodeChef COUNTARI Arithmetic Progressions FFT + 分块, 解答:戳我 (http://blog.csdn.net/gatevin/article/details/46911323)

ZOJ 3856 Goldbach FFT计数, 解答:戳我 (http://blog.csdn.net/gatevin/article/details/46917303)

UVALive 6886 Golf Bot FFT模板题, 解答: 戳我 (http://blog.csdn.net/gatevin/article/details/46927033)

HDU 4093 Xavier is Learning to Count 容斥原理 + FFT,(2011年上海现场赛C题) 解答:戳我 (http://blog.csdn.net/gatevin/article/details/47048653)

Posted in ACM (http://www.gatevin.moe/category/acm/) Tagged FFT (http://www.gatevin.moe/tag/fft/)

2016.6.27 Diary → (http://www.gatevin.moe/diary/2016-6-27-diary/)

6 thoughts on "FFT算法学习笔记"

oilover says:
November 13, 2015 at 11:27 am (http://www.gatevin.moe/acm/fft%e7%ae%97%e6%b3%95%e5%ad%a6%e4%b9%a0%e7%ac%94%e8%ae%b0/#comment-14)

Mark~~~

Reply (http://www.gatevin.moe/acm/fft%e7%ae%97%e6%b3%95%e5%ad%a6%e4%b9%a0%e7%ac%94%e8%ae%b0/?replytocom=14#respond)

peeapl (http://cnblogs.com/get-an-ac-everyday) says:
March 22, 2016 at 4:41 pm (http://www.gatevin.moe/acm/fft%e7%ae%97%e6%b3%95%e5%ad%a6%e4%b9%a0%e7%ac%94%e8%ae%b0/#comment-38)

###nice

Reply (http://www.gatevin.moe/acm/fft%e7%ae%97%e6%b3%95%e5%ad%a6%e4%b9%a0%e7%ac%94%e8%ae%b0/?replytocom=38#respond)

NUM_24 says:
April 15, 2016 at 6:10 pm (http://www.gatevin.moe/acm/fft%e7%ae%97%e6%b3%95%e5%ad%a6%e4%b9%a0%e7%ac%94%e8%ae%b0/#comment-42)

好文章，滋磁
^_^

Reply (http://www.gatevin.moe/acm/fft%e7%ae%97%e6%b3%95%e5%ad%a6%e4%b9%a0%e7%ac%94%e8%ae%b0/?replytocom=42#respond)

Pingback: HDU 1402 A * B Problem Plus FFT | Sasuke (http://smgui.net/hdu-1402-a-b-problem-plus-fft/)

Naeioi says:
August 12, 2016 at 12:30 am (http://www.gatevin.moe/acm/fft%e7%ae%97%e6%b3%95%e5%ad%a6%e4%b9%a0%e7%ac%94%e8%ae%b0/#comment-58)

非常详细的总结。另提醒一下，代码里的动态内存没有释放，数据稍大可能导致泄露

Reply (http://www.gatevin.moe/acm/fft%e7%ae%97%e6%b3%95%e5%ad%a6%e4%b9%a0%e7%ac%94%e8%ae%b0/?replytocom=58#respond)

gatevin says:
August 20, 2016 at 9:40 pm (http://www.gatevin.moe/acm/fft%e7%ae%97%e6%b3%95%e5%ad%a6%e4%b9%a0%e7%ac%94%e8%ae%b0/#comment-59)

已经更正动态内存释放的问题，感谢~

Reply (http://www.gatevin.moe/acm/fft%e7%ae%97%e6%b3%95%e5%ad%a6%e4%b9%a0%e7%ac%94%e8%ae%b0/?replytocom=59#respond)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

POST COMMENT