3.7 快速傅里叶变换

问题背景

多项式：$A(x) = \sum_{j=0}^{n-1} a_j x^j$ 度界限为n的多项式，度为$0,1,\ldots,n-1$

常用操作：加法、乘法

多项式表示方法：系数表示、点值表示

系数表示：$a = (a_0, a_1, \ldots, a_{n-1})$

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

操作/表示方法	系数表示	点值表示
加法	$\theta(n)$	$\theta(n)$
乘法	$\theta(n^2)$	$\theta(n)$

问题背景

系数表示到点值表示的转换：求值

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

单次求值 $y = \sum_{j=0}^{n-1} a_j x^j$ 的代价：$\theta(n)$

n次求值代价：$\theta(n^2)$

问题背景

$$V=\begin{pmatrix}1 & x_0 & x_0^2 & \cdots & x_0^{n-1}\\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1}\end{pmatrix}\qquad \det(V)=\prod_{0\le j<k\le n-1}(x_k-x_j)$$

点值表示到系数表示的转换：插值

$$\begin{pmatrix}a_0\\ a_1\\ \vdots\\ a_{n-1}\end{pmatrix}=V^{-1}\begin{pmatrix}y_0\\ y_1\\ \vdots\\ y_{n-1}\end{pmatrix}$$

n个点(x_k,y_k)上插值，利用拉格朗日公式可得：

$$A(x)=\sum_{k=0}^{n-1}y_k\frac{\prod_{j\ne k}(x-x_j)}{\prod_{j\ne k}(x_k-x_j)}$$

计算代价：$\theta(n^2)$

能否更快捷地在两种表示方法之间转换？

问题背景

$a_0, a_1, \dots, a_{n-1}$
$b_0, b_1, \dots, b_{n-1}$

直观计算$\theta(n^2)$

乘法结果：
$c_0, c_1, \dots, c_{2n-2}$

能否在$o(n^2)$内完成?

$A(x_0), B(x_0)$
$A(x_1), B(x_1)$
$\dots \dots$
$A(x_{2n-1}), B(x_{2n-1})$

$\theta(n)$

乘法结果：
$C(x_0)$
$C(x_1)$
$\dots \dots$
$C(x_{2n-1})$

能否在$o(n^2)$内完成?

在度界限为n的多项式上对n个不同的输入进行求值

假设n是2的整数次幂

问题背景

n个点$x_0,\ldots,x_{n-1}$上面的求值操作，尝试分治：

$$A(x)=a_0+a_1x+\cdots+a_{n-1}x^{n-1}$$
$$=a_0+a_2x^2+a_4x^4+\cdots+x(a_1+a_3x^2+a_5x^4+\cdots)$$

$$B(x)=a_0+a_2x+a_4x^2+\cdots+a_{n-2}x^{n/2-1}$$
$$C(x)=a_1+a_3x+a_5x^2+\cdots+a_{n-1}x^{n/2-1}$$
$$A(x)=B(x^2)+xC(x^2)$$ 在$x_0^2,\ldots,x_{n-1}^2$上为多项式B和C求值

多项式的度界限折半，需要计算的点的数量不变

- $T(n,n)=2T\left(\frac{n}{2},n\right)+\theta(n),T(2,n)=\theta(n)$
- $T(n,n)=\theta(n^2)$ 多项式的度界限折半，需要计算的数量也需要折半

问题背景

n个数字，求平方后数量折半，再次求平方后继续折半…

1个数字，开平方后数量翻倍，再次开平方后继续翻倍…

单位1的复数n次方根：$\omega^n = 1$

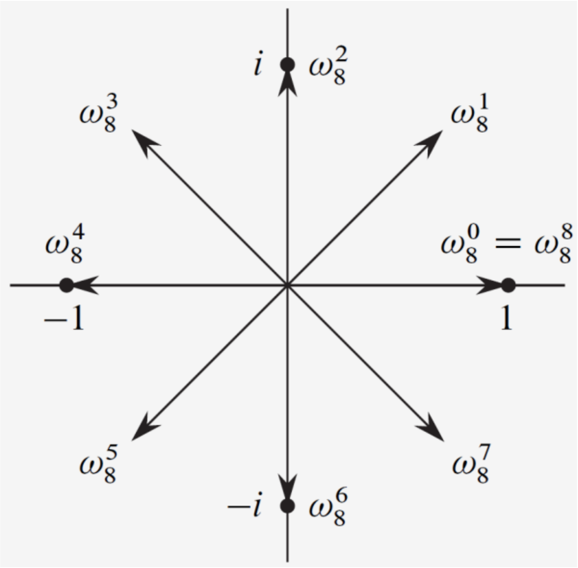
$i = \sqrt{-1}$, $e^{iu} = \cos(u) + i\sin(u)$

$\omega = e^{\frac{2\pi ik}{n}}$, $k = 0,1,\dots n-1$

$\omega_n = e^{\frac{2\pi i}{n}}$, 1的$n$次方根$\omega_n^0, \omega_n^1, \dots \omega_n^{n-1}$

$\omega_n^k = e^{\frac{2\pi ik}{n}}$

$\omega_n^2 = \omega_{n/2}, \omega_n^{n/2} = -1$



1的8次方根

$$A_j = A\left(\omega_n^j\right) = a_0 + a_1\omega_n^j + a_2\omega_n^{2j} + \cdots + a_k\omega_n^{kj} + .. + a_{n-1}\omega_n^{(n-1)j}$$

$$\omega_n^k = e^{\frac{2\pi ik}{n}}$$

问题定义

输入：$a_0, a_1, \ldots, a_{n-1}$，$a_i$是实数，$(0 \le i \le n-1)$

输出：$A_0, A_1, \ldots, A_{n-1}$，使得，

$$A_j = \sum_{k=0}^{n-1} a_k e^{\frac{2\pi ijk}{n}}$$

其中：

(1) $0 \le j \le n-1$

(2) e是自然对数的底数

(3) $i = \sqrt{-1}$ 是虚数单位

蛮力法利用定义计算每个A_j，时间复杂度为$\Theta(n^2)$

算法的数学基础

$$A_j = \sum_{k=0}^{n-1} a_k e^{\frac{2\pi ijk}{n}}$$ 令 $w_n = e^{\frac{2\pi i}{n}}$， 有： $A_j = \sum_{k=0}^{n-1} a_k w_n^{jk}$

$$A_j = a_0 + a_1 w_n^{j} + a_2 w_n^{2j} + a_3 w_n^{3j} + a_4 w_n^{4j} + \ldots\ldots + a_{n-2} w_n^{(n-2)j} + a_{n-1} w_n^{(n-1)j}$$

$$= (a_0 + a_2 w_n^{2j} + a_4 w_n^{4j} + \ldots\ldots + a_{n-2} w_n^{(n-2)j})$$

$$+ (a_1 w_n^{j} + a_3 w_n^{3j} + a_5 w_n^{5j} + \ldots\ldots + a_{n-1} w_n^{(n-1)j})$$

$$= (a_0 + a_2 w_n^{2j} + a_4 w_n^{4j} + \ldots\ldots + a_{n-2} w_n^{(n-2)j})$$

$$+ w_n^{j} (a_1 + a_3 w_n^{2j} + a_5 w_n^{4j} + \ldots\ldots + a_{n-1} w_n^{(n-2)j})$$

算法的数学基础

$$A_j = (a_0 + a_2 w_n^{2j} + a_4 w_n^{4j} + \ldots\ldots + a_{n-2} w_n^{(n-2)j})$$

奇数输入项的FFT

$$+ w_n^j (a_1 + a_3 w_n^{2j} + a_5 w_n^{4j} + \ldots\ldots + a_{n-1} w_n^{(n-2)j})$$

偶数输入项的FFT

由于$w_n^2 = w_{n/2}$,以及$w_n^{n+k} = w_n^k$,

$$A_j = \underbrace{(a_0 + a_2 w_{n/2}^{j} + a_4 w_{n/2}^{2j} + \ldots\ldots + a_{n-2} w_{n/2}^{\frac{(n-2)j}{2}})}_{B_j}$$

将问题划分为两个子问题

$$+ w_n^j \underbrace{(a_1 + a_3 w_{n/2}^{j} + a_5 w_{n/2}^{2j} + \ldots\ldots + a_{n-1} w_{n/2}^{\frac{(n-2)j}{2}})}_{C_j}$$

于是，$A_j = B_j + w_n^j C_j$ 还可证明，$A_{j+n/2} = B_j + w_n^{j+n/2} C_j$

$0 \le j < n/2$

$\omega_{n/2}^{j} = \omega_{n/2}^{j+n/2}, B_j = B_{j+n/2}, C_j = C_{j+n/2}$

分治算法过程

划分：将输入折分成$a_0,a_2,\ldots,a_{n-2}$和$a_1,a_3,\ldots,a_{n-1}$.

递归求解：递归计算$a_0,a_2,\ldots,a_{n-2}$的变换$B_0,B_1,\ldots,B_{n/2-1}$

递归计算$a_1,a_3\ldots,a_{n-1}$的变换$C_0,C_1,\ldots,C_{n/2-1}$

合并：根据$A_j= B_j+C_j\cdot W_n^{\ j}\,(j<n/2)$ 和$A_j=B_{j-n/2}+C_{j-n/2}\cdot W_n^{\ j}$ $(n/2\le j<n-1)$依次求得$A_0, A_1, \ldots, A_{n-1}$.

分治算法过程

例如：计算8个点$a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7$的FFT.

计算4个点 a_0, a_2, a_4, a_6的FFT.

计算4个点 a_1, a_3, a_5, a_7的FFT.

计算2个点 a_0, a_4的FFT.

计算2个点 a_2, a_6的FFT.

计算2个点 a_1, a_5的FFT.

计算2个点 a_3, a_7的FFT.

算法及复杂性分析

算法FFT($a_0,a_2,\ldots,a_{n-1}, n$)

输入：$a_0,a_1,\ldots,a_{n-1}$， $n=2^k$

输出：$a_0,a_1,\ldots,a_{n-1}$的傅里叶变换$A_0,\ldots,A_{n-1}$

1. $W \leftarrow \exp(2\pi i/n)$;
2. If (n=2) Then
3、 $A_0 \leftarrow a_0+a_1$;
4. $A_1 \leftarrow a_0-a_1$;
5. 输出A_0,A_1,算法结束；
6. $B_0,B_1,\ldots,B_{n/2-1} \leftarrow$FFT($a_0,a_2,\ldots,a_{n-2}, n/2$)；
7. $C_0,C_1,\ldots,C_{n/2-1} \leftarrow$FFT($a_1,a_3,\ldots,a_{n-1}, n/2$)；
8. For j=0 To $n/2-1$
9. $A_j \leftarrow B_j+C_j\cdot W^j$;
10. $A_{j+n/2} \leftarrow B_j+C_j\cdot W^{j+n/2}$;
11. 输出$A_0,A_1,\ldots,A_{n-1}$,算法结束；

$T(n)=\theta(1)$ If $n=2$

$T(n)=2T(n/2)+\theta(n)$ If $n>2$

$T(n)=\theta(n\log n)$