Fast Fourier Transform (FFT) & Polynomials

15-750
4/3/19

FFT Applications

1) Signal processing
2) Image & Data Compression
3) Solving Partial Differenial Equations (PDE's)
4) Polynomial Multiplication

We start with mult Polynomials

Recall If $f(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$

$g(x) = b_0 + b_1 x + \cdots + b_{n-1} x^{n-1}$

Then $h(x) = f(x) \cdot g(x)$ if

$$h(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{2n-2} x^{2n-2}$$

Where $C_k = \sum_{i+j=k} a_i b_j$

As an algorithm this is $O(n^2)$ operations.

We can also view poly mult as a convolution

We view $b_0, b_1, \ldots$ as a possibly infinite discrete signal which we convolve with $a_0, \ldots, a_{n-1}$

$$\begin{array}{lcccccccc} \text{Signal} & b_0 & b_1 & b_2 & b_3 & \cdots & b & \cdots \\ & & & a_0 & a_1 & \cdots & a_{n-1} & \end{array}$$

Write in matrix form

$$\begin{pmatrix} a_0 & 0 & \cdots & & 0 \\ a_1 & a_0 & 0 & & \\ a_2 & a_1 & a_0 & & \\ \vdots & & & \ddots & \\ a_{n-1} & \cdots & & & a_0 \\ \vdots & \ddots & & & \vdots \\ 0 & & & & a_{n-1} \end{pmatrix} \begin{pmatrix} b_0 \\ \vdots \\ b_{n-1} \end{pmatrix} = \begin{pmatrix} a_0 b_0 \\ a_1 b_0 + a_0 b_1 \\ a_2 b_0 + a_1 b_1 + a_0 b_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \end{pmatrix}$$

Making the Impossible Work!

Idea:

1) Evaluate: $f(0), f(1), \cdots, f(2m-2), g(0), \cdots, g(2m-2)$

2) Mult: $r_0 = f(0)g(0), \cdots, r_{2m-2} = f(2m-2)\,g(2m-2)$

3) Interpolate:

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 2^2 & \cdots & \\ \vdots & & & & \\ 1 & 2m-2 & (2m-2)^2 & \cdots & \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{2m-2} \end{pmatrix} = \begin{pmatrix} r_0 \\ \vdots \\ \\ \\ r_{2m-2} \end{pmatrix}$$

ie solve

Steps 1) & 3) look at best $O(n^2)$ $n = 2m-1$

Only 2) is $O(n)$

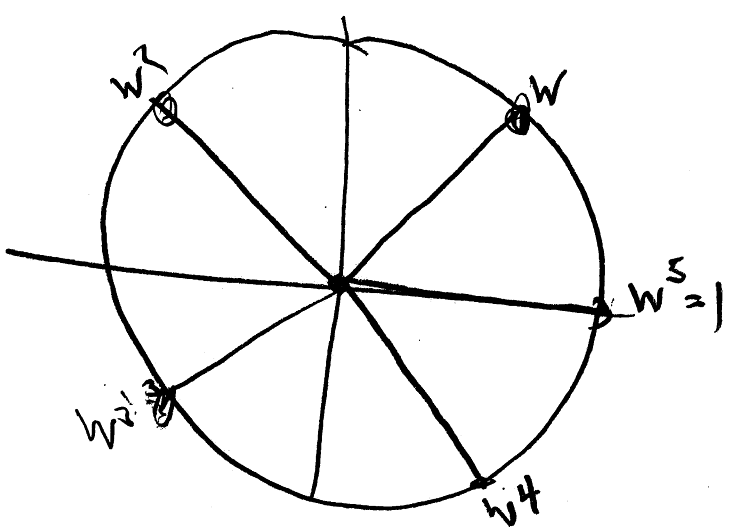
Idea: Pick nth-roots of unity to evaluate polynomials

The n-th roots of unity

Complex plane $a+ib$ $a,b\in\mathbb{R}$ $i^2=-1$

$a = \cos\left(\frac{2\pi}{n}\right)$ $b = \sin\left(\frac{2\pi}{n}\right)$ $w = a+ib$

eg $n=5$



Claim: $w = \cos\theta + i\sin\theta$ then

$$w^2 = \cos 2\theta + i\sin 2\theta$$

pf Let R_θ = CCW rotation of plane by θ degrees.

$$R_\theta = \begin{pmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{pmatrix} = M \qquad R_{2\theta} = M^2 = \begin{pmatrix}\cos^2\theta - \sin^2\theta & * \\ 2\cos\theta\sin\theta & *\end{pmatrix}$$

n-th roots of unity Definition

w is an n-th root if

1) $w^p \neq 1$ for $1 \leq p < n$

2) $w^n = 1$

Claims

3) $\sum_{j=0}^{n-1} w^j = 0$

pf $X = \sum_{j=0}^{n-1} w^j$

$$w \cdot X = \sum_{j=0}^{n-1} w^j = X \Rightarrow X = 0$$

---

4) $\sum_{j=0}^{n-1} w^{jp} = 0$ for $1 \leq p < n$

pf set $W = w^p$

and repeat proof in 3)

Def The Discrete Fourier Transform (DFT)

$$F_n \equiv \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{n-1} \\ 1 & w^2 & w^4 & \cdots & w^{2n-2} \\ 1 & w^3 & w^6 & \cdots & w^{3n-3} \\ \vdots & & & & \\ 1 & w^{n-1} & \cdots & & w^{(n-1)^2} \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ \vdots \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} v_0 \\ \vdots \\ \vdots \\ \vdots \\ v_{n-1} \end{pmatrix}$$

__Note__ row & column numering start at zero.

ie $(F_n)_{ij} = w^{ij}$ for $0 \le i,j \le n-1$ $\quad F_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

$$F_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} \begin{pmatrix} a_{00} \\ a_{01} \\ a_{10} \\ a_{11} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & i & -i \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -i & i \end{pmatrix} \begin{pmatrix} a_{00} \\ a_{10} \\ a_{01} \\ a_{11} \end{pmatrix} \begin{matrix} \text{even} \\ \\ \text{odd} \\ \end{matrix}$$

$$= \begin{pmatrix} F_2 & C \\ F_2 & -C \end{pmatrix} = \begin{pmatrix} F_2 & \begin{pmatrix} 0 & 0 \\ 0 & i \end{pmatrix} F_2 \\ F_2 & -\begin{pmatrix} 0 & 0 \\ 0 & i \end{pmatrix} F_2 \end{pmatrix}$$

<u>Thm</u> If we reorder columns of F even-then-odd

$$\tilde{F}_n = \left(\begin{array}{c|c} F_{n/2} & D_{n/2} F_{n/2} \\ \hline F_{n/2} & -D_{n/2} F_{n/2} \end{array}\right) \text{ where } D_{n/2} = \begin{pmatrix} 1 & & & & 0 \\ & w & & & \\ & & w^3 & & \\ & & & \ddots & \\ 0 & & & & w^{n/2-1} \end{pmatrix}$$

---

no proof.

thus $F_{2n}(a) = \begin{pmatrix} F_n & D_n F_n \\ F_n & -D_n F_n \end{pmatrix} \begin{pmatrix} a_{even} \\ a_{odd} \end{pmatrix}$

$$= \begin{pmatrix} F_n a_{even} + D_n F_n a_{odd} \\ F_n a_{even} - D_n F_n a_{odd} \end{pmatrix}$$

A recurence for number of ops

$$T(n) = 2T(n/2) + cn$$

Thus $T(n) = O(n \lg n)$

Back to Poly Multi

We will use n-th roots of unity

<u>step 1</u> $O(n\log n)$ time

<u>step 2</u> $O(n)$ time

<u>step 3</u> ie solve $F_n a = b$ for a

Trick: F_n^{-1} is just an FFT

<u>Claim</u> $\left(F_n^{-1}\right)_{ij} = \left(\frac{1}{n}\right) w^{-ij} = \frac{1}{n} F_n(w^{-1})$

$= \left(\frac{1}{n}\right) (w^{-1})^{ij}$

FFT inverse

Thm $\left(F_n^{-1}\right)_{ij} = \frac{1}{n} w^{-ij}$

ie $$\sum_{j=0}^{n-1} w^{ij} w^{-jk} = \begin{cases} n & \text{if } i=k \checkmark \\ 0 & \text{o.w} \end{cases}$$

$$LHS = \sum_{j=0}^{n-1} w^{j(i-k)}$$

if $i=k$ then $LHS = \sum_{j=0}^{n-1} 1 = n$

if $i \neq k$ then $LHS = \sum_{j=0}^{n-1} w^{jp} = 0$ prop 4)

$1 \leq p < n$

Thus step 3) is just an FFT using the root ω^{-1}.

Our timing is just

Step 1)	Two FFT's	$O(n\log n)$ ops
Step 2)	$2n$ multiples	$O(n)$ ops
Step 3)	One FFT	$O(n\log n)$ ops
Total		$O(n\log n)$ ops

<u>Thm</u> Two deg n polynomial can be multi in $O(n\log n)$ ops

<u>2-issues</u>

A) Do we have to work with complex numbers?

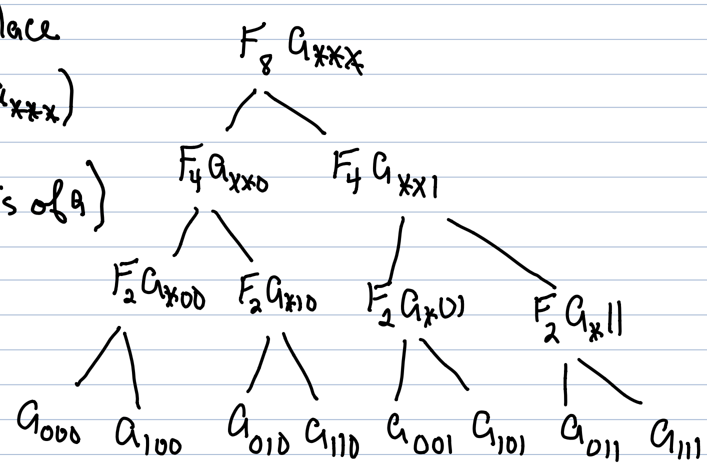
B) Can we preorder the columns?

A) One can work over quotient rings that have roots of unity.

B) We show after reordering columns we can do FFT in-place

Consider $F_8(a_{***})$

($a_{**0} \equiv$ even elements of a)



Lets write this as a permutation of elements a

$$
\begin{aligned}
a_{000} &\longrightarrow a_{000} \\
a_{001} &\longrightarrow a_{100} \\
a_{010} &\longrightarrow a_{010} \\
a_{011} &\longrightarrow a_{110} \\
a_{100} &\longrightarrow a_{001} \\
a_{101} &\longrightarrow a_{101} \\
a_{110} &\longrightarrow a_{011} \\
a_{111} &\longrightarrow a_{111}
\end{aligned}
$$

Thus a_* goes to the bit reversal of a_*.

<u>Note</u> After Bit Reversal FFT can be done inplace.

Additional notes

Alternate interpretation

The Discrete Fourier Transform on a finite sequence a of length n is

$$F\{a\}(k) = \sum_{j=0}^{n-1} a_j e^{\frac{i2\pi jk}{n}}.$$

Compare to the Continuous Fourier Transform on a function f which is given by

$$\mathcal{F}\{f\}(k) = \int_{-\infty}^{\infty} f(t)e^{-i2\pi kt}dt.$$

We derived the first formula by considering the evaluation of a polynomial with coefficients a at the nth roots of unity. Another interpretation is that when the input is a time-varying signal, the Fourier Transform represents the input's frequency spectrum. That is, if we decompose the input into a linear superposition of sinusoidal waves, $|F\{a\}(k)|$ is the amplitude of the wave of frequency k.

Convolution

To multiply two polynomials, the coefficients of the resulting polynomial is just the convolution of the coefficients of the orignal polynomials. The convolution of two finite sequences a, b is given by

$$(a * b)[k] = \sum_{j=0}^{k} a[j]b[k-j].$$

We solved polynomial multiplication, or convolution, in 3 steps:

1. Use FFT to compute the Discrete Fourier Transforms, $F\{a\}$ and $F\{b\}$.
2. Multiply pointwise $F\{a\}$ and $F\{b\}$.
3. Take the inverse DFT $F^{-1}\{F\{a\} \cdot F\{b\}\}$, also using FFT, to recover $a * b$.

Here, convolution of the original inputs corresponded to the simpler multiplication operation in the Fourier domain. This fact also holds for the Continuous Fourier Transform: $\mathcal{F}\{f(t) * g(t)\} = \mathcal{F}\{f(t)\} \cdot \mathcal{F}\{g(t)\}$. We call this the *Convolution Theorem*.