The Fast Fourier Transform

October 4, 2023 5:25 PM

[Cooley & Tukey 1965]

- No class next Tuesday Oct. 10th.
- Assignment #2 due Tuesday Oct 10th @ 11pm
- Mondays office hour moved to Tuesday.

Let $a(x) = \sum_{i=0}^{n-1} a_i x^i$ with $a_i \in F$ a field and $\alpha_1 \neq \alpha_2 \neq \cdots \neq \alpha_n \in F$.

How fast can we evaluate $a(x)$ at $x = \alpha_i$ for $1 \le i \le n$?
Given $a(\alpha_i)$ for $1 \le i \le n$ how fast can we interpolate $a(x)$?

Evaluating $a(x)$ using Horner's method: $\underset{\#\alpha_i\text{'s}}{n}\,\underset{\text{mults in } F}{(n-1)} \in O(n^2)$.
Newton's interpolation costs $O(n^2)$ mults in F.

$$\underset{\text{coefficients}}{\overset{F^n \,\ni}{a = [a_0, a_1, \dots, a_{n-1}]}} \;\underset{\text{interpolation}}{\overset{\text{evaluation}}{\rightleftarrows}}\; \underset{\text{values}}{\overset{F^n \,\ni}{[a(\alpha_1), a(\alpha_2), \dots, a(\alpha_n)]}}$$

Idea of the FFT? Suppose $2 \mid n$.

$$a(x) = (a_0 + a_2 x^2 + \cdots + a_{n-2} x^{n-2}) + x(a_1 + a_3 x^2 + \cdots + a_{n-1} x^{n-2})$$

$$a(x) = b(x^2) + x\,c(x^2)$$

where $b(x) = a_0 + a_2 x + \cdots + a_{n-2} x^{(n-2)/2}$
$c(x) = a_1 + a_3 x + \cdots + a_{n-1} x^{(n-2)/2}$.

Evaluate $a(x)$ at $\pm 1, \pm 2, \pm 3, \dots, \pm \frac{n}{2}$

$$a(\pm 2) = b(4) \pm 2 \cdot c(4) \qquad a(2) = b(4) + 2c(4)$$
$$a(-2) = b(4) - 2 \cdot c(4).$$

This saves almost $\frac{1}{2}$ the work.

Def. An element $\omega \in F$ is an n'th root of unity if $\omega^n = 1$. ω is a primitive n'th root of unity (pnru) if $\omega^n = 1$ and $\omega^k \neq 1$ for $1 \le k \le n-1$.

Example. In $\mathbb{C}$ i is a primitive 4th root of unity.
$i^4 = (i^2)^2 = (-1)^2 = 1.$ $\quad i^1 = i \quad i^2 = -1 \quad i^3 = -i$

Example. In $\mathbb{Z}_{13}$ $\omega = 5$ is a p4ru.

ω^0	ω^1	ω^2	ω^3	$\omega^4 = -5 \cdot 5 = -25 = 1.$
1	5	-1	-5	1

Lemma 1. Let ω be a pnru.

me (i) $\omega^j = -\omega^{j+n/2}$ or $\omega^{j+n/2} = -\omega^j$ for $0 \le j < \frac{n}{2}$

you (ii) ω^2 is a p$\frac{n}{2}$ru.

(iii) $\omega^0 + \omega^1 + \omega^2 + \cdots + \omega^{n-1} = 0.$

Proof (i) $(\omega^{j+n/2} - \omega^j)(\omega^{j+\frac{n}{2}} + \omega^j)$

one of these $= 0$.

$= \omega^{2j+n} - \omega^{2j}$

$= \omega^n \cdot \omega^{2j} - \omega^{2j}$

$= 0.$

for any j $\quad \neq 0 \quad \neq 1$

Suppose $\omega^{j+n/2} - \omega^j = 0 \Rightarrow \omega^j(\omega^{n/2} - 1) = 0$ $\boxtimes$

So $\omega^{j+n/2} + \omega^j = 0 \Rightarrow \omega^{j+n/2} = -\omega^j$ $\square$.

Def. Let ω be a pnru in F a field.
Let $a(x) \in F[x]$ & degree $\leq n-1$. Then

$$B = [a(1), a(\omega), a(\omega^2), \ldots, a(\omega^{n-1})] \in F^n$$

is the (discrete) Fourier Transform of $a(x)$.

How fast can we compute B?

Algorithm DFFT (Discrete FFT)

Inputs $n = 2^k$, $A = [a_0, a_1, \ldots, a_{n-1}] \in F^n$; $\omega \in F$ is a pnru.
where $a(x) = \sum_{i=0}^{n-1} a_i x^i$.

Output $[a(1), a(\omega), a(\omega^2), \ldots, a(\omega^{n-1})] \in F^n$.

if $n = 1$ $\begin{Bmatrix} A = [a_0] & a(1) = a_0 \\ a(x) = a_0 & [a_0] \end{Bmatrix}$ return A.

$b \leftarrow [a_0, a_2, a_4, \ldots, a_{n-2}]$ // $b(x) = a_0 + a_2 x + \cdots + a_{n-2} x^{n-2}$

$c \leftarrow [a_1, a_3, a_5, \ldots, a_{n-1}]$ // $c(x) = a_1 + a_3 x + \cdots + a_{n-1} x^{n-1}$

// $a(x) = b(x^2) + x\, c(x^2)$.

$B \leftarrow DFFT(\frac{n}{2}, b, \omega^2)$ // $\underline{B} = [b(1), b(\omega^2), \overset{i}{b}(\omega^4), \ldots, b(\omega^{n-2})]$

$C \leftarrow DFFT(\frac{n}{2}, c, \omega^2)$ // $C = [c(1), c(\omega^2), c(\omega^4), \ldots, c(\omega^{n-2})]$

[$Y \leftarrow 1$

for $i = 0, 1, \ldots, \frac{n}{2} - 1$ do

[$T \leftarrow Y \cdot C_i$ // $Y = \omega^i$

$A_i \leftarrow B_i + \omega^i C_i = T$

$A_{i+\frac{n}{2}} \leftarrow B_i - \omega^i C_i = T$

[$Y \leftarrow \omega \cdot Y$

end for.

return $[A_0, A_1, \ldots, A_{n-1}]$

$a(x) = b(x^2) + x\, c(x^2)$

$= b(\omega^{2i}) + \omega^i c(\omega^{2i}) = \underline{a(\omega^i)}$

$= b(\omega^{2i}) - \omega^i \cdot c(\omega^{2i})$

$= b((\omega^{i+n/2})^2) - \omega^i c((\omega^{i+n/2})^2) = a(\omega^{i+n/2})$.

$\underline{+\omega^{i+n/2}}$

Cost? Let $T(n)$ be the # mults in F.

$n=1 \quad T(1)=0$

$n>1 \quad T(n) = 2T(\frac{n}{2}) + 1 + \frac{n}{2}\cdot 2$ ← for ω^2

↑ two recursive calls.

Solving this recurrence we get

$$T(n) = 1\cdot n\cdot \log_2 n + n - 1 \in O(n \log n).$$

Optimization. Precompute $1, \omega, \omega^2, \ldots, \omega^{n/2-1}$ using $\frac{n}{2}-1$ mults in F.

Set $W = [\underbrace{1, \omega, \omega^2, \ldots, \omega^{\frac{n}{2}-1}}_{n/2}, \underbrace{1, \omega^2, \omega^4, \ldots, \omega^{n-2}}_{n/4}, \underbrace{1, \omega^4, \omega^8, \ldots, \omega^{n-4}}_{n/8}, \ldots, 1, 0] \in F^n$.

This means we can eliminate $y \leftarrow \omega \cdot y$ from the loop.

And we have.

$$T(n) = 2T(n/2) + \frac{n}{2} \text{ multiplications in } F.$$

$$T(1) = 0$$

$$\Rightarrow \quad T(n) = \frac{1}{2} n \log_2 n \in O(n \log n).$$