Math 416

Worksheet 11. The Fast Fourier Transform

Remember that we want to evaluate a polynomial $A(x)$ at the n^{th} roots of unity $1, \zeta, \ldots, \zeta^{n-1}$. The idea: to evaluate $A(\zeta^k)$, we recursively evaluate $A_{\text{even}}(\zeta^{2k})$ and $A_{\text{odd}}(\zeta^{2k})$ and combine as follows:

$$\begin{aligned} A(\zeta^k) &= A_{\text{even}}(\zeta^{2k}) + \zeta^k A_{\text{odd}}(\zeta^{2k}) \\ A(\zeta^{k+n/2}) &= A_{\text{even}}(\zeta^{2k}) - \zeta^k A_{\text{odd}}(\zeta^{2k}) \end{aligned}$$

Problem 1. Why does the second equation give the correct value for $A(\zeta^{k+n/2})$?

Here is the algorithm.

Algorithm 1: Fast Fourier Transform

```
FFT(a, ζ)
    Input: A sequence a = (a_0, ..., a_{n-1}), n a power of 2, a primitive n^th root of unity ζ
    Output: M_n(ζ) · a
if ζ = 1 then
    return a
set (E_0, E_1, ..., E_{n/2-1}) = FFT((a_0, a_2, ..., a_{n-2}), ζ^2);
set (O_0, O_1, ..., O_{n/2-1}) = FFT((a_1, a_3, ..., a_{n-1}), ζ^2);
foreach k = 0 to n/2 - 1 do
    set c_k = E_k + ζ^k O_k ;
    set c_{k+n/2} = E_k - ζ^k O_k;
return (c_0, c_1, ..., c_{n-1})
```

Problem 2.

(a) Run $\text{FFT}((x,y),-1)$ to see that FFT works correctly on sequences of size 2.

(b) Verify (using either i or $-i$) that the FFT algorithm works correctly on input sequences of size 4.

We want to prove that the FFT algorithm is correct, i.e., that

$$\begin{aligned} \text{FFT}(z_\bullet, \zeta^{-1}) &= M(\zeta^{-1}) z_\bullet = \text{DFT}(z_\bullet) \text{ and} \\ \tfrac{1}{n}\,\text{FFT}(c_\bullet, \zeta) &= \tfrac{1}{n} M(\zeta) c_\bullet \;= \text{IFT}(c_\bullet), \end{aligned}$$

where here by ζ we mean $e^{2\pi i/n}$.

This boils down to the following fact.

Proposition. Suppose that $c_\bullet = M_n(\zeta)(z_\bullet)$ and write

$$\begin{aligned} E_\bullet &= M_{n/2}(\zeta^2) \cdot \begin{bmatrix} z_0 & z_2 & \cdots & z_{n-2} \end{bmatrix}^\top \quad \text{and} \\ O_\bullet &= M_{n/2}(\zeta^2) \cdot \begin{bmatrix} z_1 & z_3 & \cdots & z_{n-1} \end{bmatrix}^\top . \end{aligned}$$

Then $c_0, \ldots, c_{n-1}$ are given by the following formula, for $k = 0, 1, \ldots, n/2 - 1$.

$$c_k = E_k + \zeta^k O_k$$

$$c_{k+n/2} = E_k - \zeta^k O_k$$

Problem 3. Write down exactly what the Proposition is asserting in the case $n = 4$, $\zeta = i$.

Problem 4. Prove the Proposition, and explain why the correctness of the FFT algorithm follows.

Here's another way to look at it:

$$M(\zeta^{-1})z_\bullet = \begin{bmatrix} I_{n/2} & D_{n/2} \\ I_{n/2} & -D_{n/2} \end{bmatrix} \begin{bmatrix} M(\zeta^{-2})z_{\text{even}} \\ M(\zeta^{-2})z_{\text{odd}} \end{bmatrix}. \tag{$\star$}$$

Problem 5. In Equation ($\star$), $I_{n/2}$ is the $(n/2) \times (n/2)$ identity matrix. What is $D_{n/2}$?

Running time The FFT Algorithm, on an input sequence of length n, makes two recursive calls to itself on input sequences of length $n/2$, and also does some variable reassignment, etc., that takes $\Theta(n)$ time.

Problem 6. Write $T(n)$ for the worst-case running time of the FFT Algorithm on input sequences of size n.

(a) What is the recurrence that $T(n)$ satisfies?
(b) Which case of the Master Theorem does this fall under?
(c) What can we conclude about the asymptotics of $T(n)$ from the Master Theorem?

Polynomial multiplication Our original goal was to multiply polynomials $f(x)$ and $g(x)$ efficiently. The idea is to use FFT to pass from the coefficient forms of f and g to their point-value forms: $(f(1), f(\zeta), f(\zeta^2), \ldots, f(\zeta^{n-1}))$ and similarly for g. Then it is easy to multiply in point-value form: e.g. $(f \cdot g)(\zeta) = f(\zeta) \cdot g(\zeta)$. Then we use FFT to convert back to coefficient form.

Problem 7. Write out this polynomial-multiplication procedure in pseudocode, calling the FFT subroutine as necessary.

Remark. Base-b notation (e.g. in base $b = 10$, $753 = 7 \cdot 10^2 + 5 \cdot 10 + 3 \cdot 10^0$) expresses integers as polynomials evaluated at b, so a fast algorithm for polynomial multiplication gives a fast algorithm for integer multiplication.

Culture: ***for the interested reader***

- FFT crucial for signal-processing. See Wikipedia.
- FFT credited to Cooley–Tukey (1965), but the main ideas go back to Gauss 1805.
- Shor's quantum algorithm to factor into primes uses a quantum FFT.
- Can $O(n \log n)$ be improved? Open question!

Fourier analysis: ***for the interested reader*** What the heck does this have to do with Fourier analysis?

If $f\colon \mathbb{R} \to \mathbb{R}$ is 2π-periodic and 'reasonable' (bounded derivative, differentiable at most points, ...), then there are real numbers $a_0, a_1, b_1, a_2, b_2, \dots$ such that

$$f(x) = a_0 + a_1\cos(x) + b_1\sin(x) + a_2\cos(2x) + b_2\sin(2x) + \cdots \tag{1}$$

(and in particular the expression on the right converges). This is called the **Fourier series** of f. (Compare to the Taylor series: $\sum a_n x^n$.)

A 2π-periodic function is better thought of as a function $S^1 \to \mathbb{R}$, or even better $S^1 \to \mathbb{C}$. (This S^1 is the unit circle.) Try again with functions $\theta \mapsto e^{i\pi\theta}$:

$$\begin{aligned} f(\theta) &= \cdots + c_{-2}e^{-2i\theta} + c_{-1}e^{-i\theta} + c_0 + c_1e^{i\theta} + c_2e^{2i\theta} + \cdots \qquad (2)\\ &= \sum_{-\infty}^{\infty} c_k e^{ik\theta} \\ &= \sum_{-\infty}^{\infty} c_k(\cos(k\theta) + i\sin(k\theta)) \\ &= c_0 + \sum_{k=1}^{\infty} c_k(\cos(k\theta) + i\sin(k\theta)) + c_{-k}(\cos(k\theta) - i\sin(k\theta)) \\ &= c_0 + \sum_{k=1}^{\infty}(c_k + c_{-k})\cos(k\theta) + i(c_k - c_{-k})\sin(k\theta). \end{aligned}$$

Set $a_0 = c_0$ and $a_k = c_{-k} + c_k$, $b_k = i(c_k - c_{-k})$ for $k > 0$ to get the first expression (1).

Exercise. Assuming $f(\theta)$ equals a series as in (2) above, and that integration of infinite series can be done term by term, show that

$$c_k = \frac{1}{2\pi}\int_0^{2\pi} f(\theta)e^{-ik\theta}\, d\theta.$$

The Fourier Transform sends f to $(c_k : k \in \mathbb{Z})$; its inverse sends the sequence $\vec{c}$ to f.

Now observe that the Riemann sum of $\int_0^{2\pi} \frac{1}{2\pi} f(\theta)e^{-ik\theta}\, d\theta$ with n sample points $\theta = 2l\pi/n$, $l = 0, 1, \dots, n-1$, is

$$\begin{aligned} \frac{1}{2\pi}\sum_{l=0}^{n-1} f(2l\pi/n)e^{-ik\theta} \cdot \frac{2\pi}{n} &= \frac{1}{n}\sum_{l=0}^{n-1} f(2l\pi/n)e^{-2\pi ikl/n} \\ &= \frac{1}{n}\,\mathrm{DFT}(f(0), f(2\pi/n), f(4\pi/n), \dots, f(2(n-1)/n\pi)), \end{aligned}$$

a 'uniform sample' from f.

(Notice that our $\frac{1}{n}$ shows up in IFT instead. You can do it either way.)