Fast Fourier Transform

The problem:

Given: two polynomials

$$p(X) = p_0 + p_1 X + \ldots + p_n X^n \qquad q(X) = q_0 + q_1 X + \ldots + q_n X^n$$

Compute:

$$r(X) = p(X) \cdot q(X)$$

The problem:

Given: two polynomials

$$p(X) = p_0 + p_1 X + \ldots + p_n X^n \qquad q(X) = q_0 + q_1 X + \ldots + q_n X^n$$

Compute:

$$r(X) = p(X) \cdot q(X)$$

....

Aside: If we can do this, we can multiply integers (in almost the same time)!

$$12345 \times 54321 = r(10) = p(10) \times q(10),$$

where

$$p(X) = 5 + 4X + 3X^2 + 2X^3 + X^4 \qquad q(X) = 1 + 2X + 3X^2 + 4X^3 + 5X^4$$

Fast Fourier Transform

Given: two polynomials

$$p(X) = p_0 + p_1 X + \ldots + p_n X^n \qquad q(X) = q_0 + q_1 X + \ldots + q_n X^n$$

Compute:

$$r(X) = p(X) \cdot q(X)$$

FFT: A divide and conquer algorithm to do this in time $O(n \log n)$.

Given: two polynomials

$$p(X) = p_0 + p_1 X + \ldots + p_n X^n \qquad q(X) = q_0 + q_1 X + \ldots + q_n X^n$$

Compute:

$$r(X) = p(X) \cdot q(X)$$

FFT: A divide and conquer algorithm to do this in time $O(n \log n)$.

Given: two polynomials

$$p(X) = p_0 + p_1 X + \ldots + p_n X^n \qquad q(X) = q_0 + q_1 X + \ldots + q_n X^n$$

Compute:

$$r(X) = p(X) \cdot q(X)$$

FFT: A divide and conquer algorithm to do this in time $O(n \log n)$.

<u>FFT Outline</u>

1. Compute $p(1), p(\omega), p(\omega^2), \ldots, p(\omega^{2n})$.
2. Compute $q(1), q(\omega), q(\omega^2), \ldots, q(\omega^{2n})$.
3. Compute $r(\omega), r(\omega^2), \ldots, r(\omega^{2n})$.
4. Compute $r(X)$.

Given: two polynomials

$$p(X) = p_0 + p_1 X + \ldots + p_n X^n \qquad q(X) = q_0 + q_1 X + \ldots + q_n X^n$$

Compute:

$$r(X) = p(X) \cdot q(X)$$

FFT: A divide and conquer algorithm to do this in time $O(n \log n)$.

<u>FFT Outline</u>	**<u>Running time</u>**
1. Compute $p(1), p(\omega), p(\omega^2), \ldots, p(\omega^{2n})$.	$O(n \log n)$
2. Compute $q(1), q(\omega), q(\omega^2), \ldots, q(\omega^{2n})$.	$O(n \log n)$
3. Compute $r(\omega), r(\omega^2), \ldots, r(\omega^{2n})$.	$O(n) : r(\omega^j) = p(\omega^j) \cdot q(\omega^j)$
4. Compute $r(X)$.	$O(n \log n)$

Given: two polynomials

$$p(X) = p_0 + p_1 X + \ldots + p_n X^n \qquad q(X) = q_0 + q_1 X + \ldots + q_n X^n$$

Compute:

$$r(X) = p(X) \cdot q(X)$$

FFT: A divide and conquer algorithm to do this in time $O(n \log n)$.

FFT Outline	**Running time**
1. Compute $p(1), p(\omega), p(\omega^2), \ldots, p(\omega^{2n})$.	$O(n \log n)$
2. Compute $q(1), q(\omega), q(\omega^2), \ldots, q(\omega^{2n})$.	$O(n \log n)$
3. Compute $r(\omega), r(\omega^2), \ldots, r(\omega^{2n})$.	$O(n) : r(\omega^j) = p(\omega^j) \cdot q(\omega^j)$
4. Compute $r(X)$.	$O(n \log n)$

The catch: ω is a complex number!

Key step

Given:

$$p(X) = p_0 + p_1 X + \ldots + p_n X^n$$

Compute:

$$p(1), p(\omega), p(\omega^2), \ldots, p(\omega^n)$$

Observe:

$$V \cdot \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega & \omega^2 & \ldots & \omega^n \\ 1 & \omega^2 & \omega^4 & \ldots & \omega^{2\cdot n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^n & \omega^{2\cdot n} & \ldots & \omega^{n\cdot n} \end{bmatrix} \cdot \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} p(1) \\ p(\omega) \\ p(\omega^2) \\ \vdots \\ p(\omega^n) \end{bmatrix}$$

where $V_{i,j} = \omega^{(i-1)(j-1)}$

We use divide and conquer to do this in time $O(n \log n)$

What is ω?

Without loss of generality, assume that $n + 1 = 2^t$.

$$\omega = e^{2\pi i/(n+1)}$$

primitive $n + 1$st root of unity.

Properties:

1. The numbers $1, \omega, \omega^2, \omega^3, \ldots, \omega^n$ are all the $n + 1$'st roots of unity (meaning $x^{n+1} = 1$, for $x = 1, \omega, \ldots, \omega^n$).

2. $V \cdot V = (n + 1) \cdot I$

3. The numbers $1, \omega^2, \omega^4, \omega^6, \ldots, \omega^{2(n+1)}$ are all $(n + 1)/2$'th roots of unity.

$$V \cdot \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega & \omega^2 & \ldots & \omega^n \\ 1 & \omega^2 & \omega^4 & \ldots & \omega^{2\cdot n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^n & \omega^{2\cdot n} & \ldots & \omega^{n\cdot n} \end{bmatrix} \cdot \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} p(1) \\ p(\omega) \\ p(\omega^2) \\ \vdots \\ p(\omega^n) \end{bmatrix}$$

Key step

Given:

$$p(X) = p_0 + p_1 X + \ldots + p_n X^n$$

Compute:

$$p(1), p(\omega), p(\omega^2), \ldots, p(\omega^n) = V \cdot \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{bmatrix}$$

Divide and Conquer:

1. Write $p(X) = p_e(X^2) + X \cdot p_o(X^2)$, where
   $p_e(X^2) = p_0 + p_2 X^2 + p_4 X^4 + \ldots$ and
   $p_o(X^2) = p_1 + p_3 X^2 + p_5 X^4 + \ldots$
2. Recursively evaluate $p_e(1), p_e(\omega^2), \ldots, p_e(\omega^{2n})$ and
   $p_o(1), p_o(\omega^2), \ldots, p(\omega^{2n})$.
3. Combine the results to compute $p(1), p(\omega), \ldots, p(\omega^n)$.

Running time

$T(n) \leq 2T(n/2) + O(n)$

so running time is

$O(n \log n)$

Final step

Given:

$$r(1), r(\omega), \ldots, r(\omega^n)$$

Compute:

$$r_0 + r_1 X + \ldots + r_n X^n$$

Observation:

$$\begin{bmatrix} r_0 \\ r_1 \\ \vdots \\ r_n \end{bmatrix} = \frac{1}{n+1} \cdot V \cdot V \cdot \begin{bmatrix} r_0 \\ r_1 \\ \vdots \\ r_n \end{bmatrix} = \frac{1}{n+1} \cdot V \cdot \begin{bmatrix} r(1) \\ r(\omega) \\ \vdots \\ r(\omega^n) \end{bmatrix}$$

So, we just use the same divide and conquer algorithm to compute

$$V \begin{bmatrix} r(1) \\ r(\omega) \\ \vdots \\ r(\omega^n) \end{bmatrix}.$$

Given: two polynomials

$$p(X) = p_0 + p_1 X + \ldots + p_n X^n \qquad q(X) = q_0 + q_1 X + \ldots + q_n X^n$$

Compute:

$$r(X) = p(X) \cdot q(X)$$

FFT: A divide and conquer algorithm to do this in time $O(n \log n)$.

Given: two polynomials

$$p(X) = p_0 + p_1 X + \ldots + p_n X^n \qquad q(X) = q_0 + q_1 X + \ldots + q_n X^n$$

Compute:

$$r(X) = p(X) \cdot q(X)$$

FFT: A divide and conquer algorithm to do this in time $O(n \log n)$.

<u>FFT Outline</u>

1. Compute $p(1), p(\omega), p(\omega^2), \ldots, p(\omega^{2n})$.
2. Compute $q(1), q(\omega), q(\omega^2), \ldots, q(\omega^{2n})$.
3. Compute $r(\omega), r(\omega^2), \ldots, r(\omega^{2n})$.
4. Compute $r(X)$.

Given: two polynomials

$$p(X) = p_0 + p_1X + \ldots + p_nX^n \qquad q(X) = q_0 + q_1X + \ldots + q_nX^n$$

Compute:

$$r(X) = p(X) \cdot q(X)$$

FFT: A divide and conquer algorithm to do this in time $O(n \log n)$.

FFT Outline	**Compute**
1. Compute $p(1), p(\omega), p(\omega^2), \ldots, p(\omega^{2n})$.	$O(n \log n)$
2. Compute $q(1), q(\omega), q(\omega^2), \ldots, q(\omega^{2n})$.	$O(n \log n)$
3. Compute $r(\omega), r(\omega^2), \ldots, r(\omega^{2n})$.	$O(n) : r(\omega^j) = p(\omega^j) \cdot q(\omega^j)$
4. Compute $r(X)$.	$O(n \log n)$

Given: two polynomials

$$p(X) = p_0 + p_1 X + \ldots + p_n X^n \qquad q(X) = q_0 + q_1 X + \ldots + q_n X^n$$

Compute:

$$r(X) = p(X) \cdot q(X)$$

FFT: A divide and conquer algorithm to do this in time $O(n \log n)$.

<u>**FFT Outline**</u>	<u>**Compute**</u>
1. Compute $p(1), p(\omega), p(\omega^2), \ldots, p(\omega^{2n})$.	$O(n \log n)$
2. Compute $q(1), q(\omega), q(\omega^2), \ldots, q(\omega^{2n})$.	$O(n \log n)$
3. Compute $r(\omega), r(\omega^2), \ldots, r(\omega^{2n})$.	$O(n) : r(\omega^j) = p(\omega^j) \cdot q(\omega^j)$
4. Compute $r(X)$.	$O(n \log n)$

The catch: ω is a complex number!