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:

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

. . . .

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 $O(n \log n)$.

FFT Outline	**Running time**
1. Set m to be a power of 2, $m > 2n$.	
2. Compute $p(a_0), p(a_1), p(a_2), \ldots, p(a_{m-1})$.	$O(n \log n)$
3. Compute $q(a_0), q(a_1), q(a_2), \ldots, q(a_{m-1})$.	$O(n \log n)$
4. Compute $r(a_0), r(a_1), \ldots, r(a_{m-1})$.	$O(n) : r(a_j) = p(a_j) \cdot q(a_j)$
5. Compute $r(X)$.	$O(n \log n)$

Given:

$$p(X) = p_0 + p_1 X + \ldots + p_m X^m$$

Compute:

$$p(a^0), p(a^1), p(a^2), \ldots, p(a^{m-1})$$

Divide and Conquer Algorithm:

1. Write $p(X) = p_e(X^2) + X \cdot p_o(X^2)$, where
   $p_e(Y) = p_0 + p_2 Y + p_4 Y^2 + \ldots$ and
   $p_o(Y) = p_1 + p_3 Y + p_5 Y^2 + \ldots$
2. Recursively evaluate $p_e(a^0), p_e(a^2), \ldots, p_e(a^{2(m-1)})$ and
   $p_o(a^0), p_o(a^2), \ldots, p(a^{2(m-1)})$.
3. Combine the results to compute $p(a^0), p(a^1), \ldots, p(a^{m-1})$, by
   setting $p(a^j) = p_e(a^{2j}) + a \cdot p_o(a^{2j})$

Running time

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

so running time is

$$O(n \log n)$$

Given:

$$p(X) = p_0 + p_1 X + \ldots + p_m X^m$$

Compute:

$$p(a^0), p(a^1), p(a^2), \ldots, p(a^{m-1})$$

Divide and Conquer Algorithm:

1. Write $p(X) = p_e(X^2) + X \cdot p_o(X^2)$, where
   $p_e(Y) = p_0 + p_2 Y + p_4 Y^2 + \ldots$ and
   $p_o(Y) = p_1 + p_3 Y + p_5 Y^2 + \ldots$
2. Recursively evaluate $p_e(a^0), p_e(a^2), \ldots, p_e(a^{2(m-1)})$ and
   $p_o(a^0), p_o(a^2), \ldots, p(a^{2(m-1)})$.
3. Combine the results to compute $p(a^0), p(a^1), \ldots, p(a^{m-1})$, by
   setting $p(a^j) = p_e(a^{2j}) + a \cdot p_o(a^{2j})$

Problem:
The polys are smaller, but the number of points is the same!

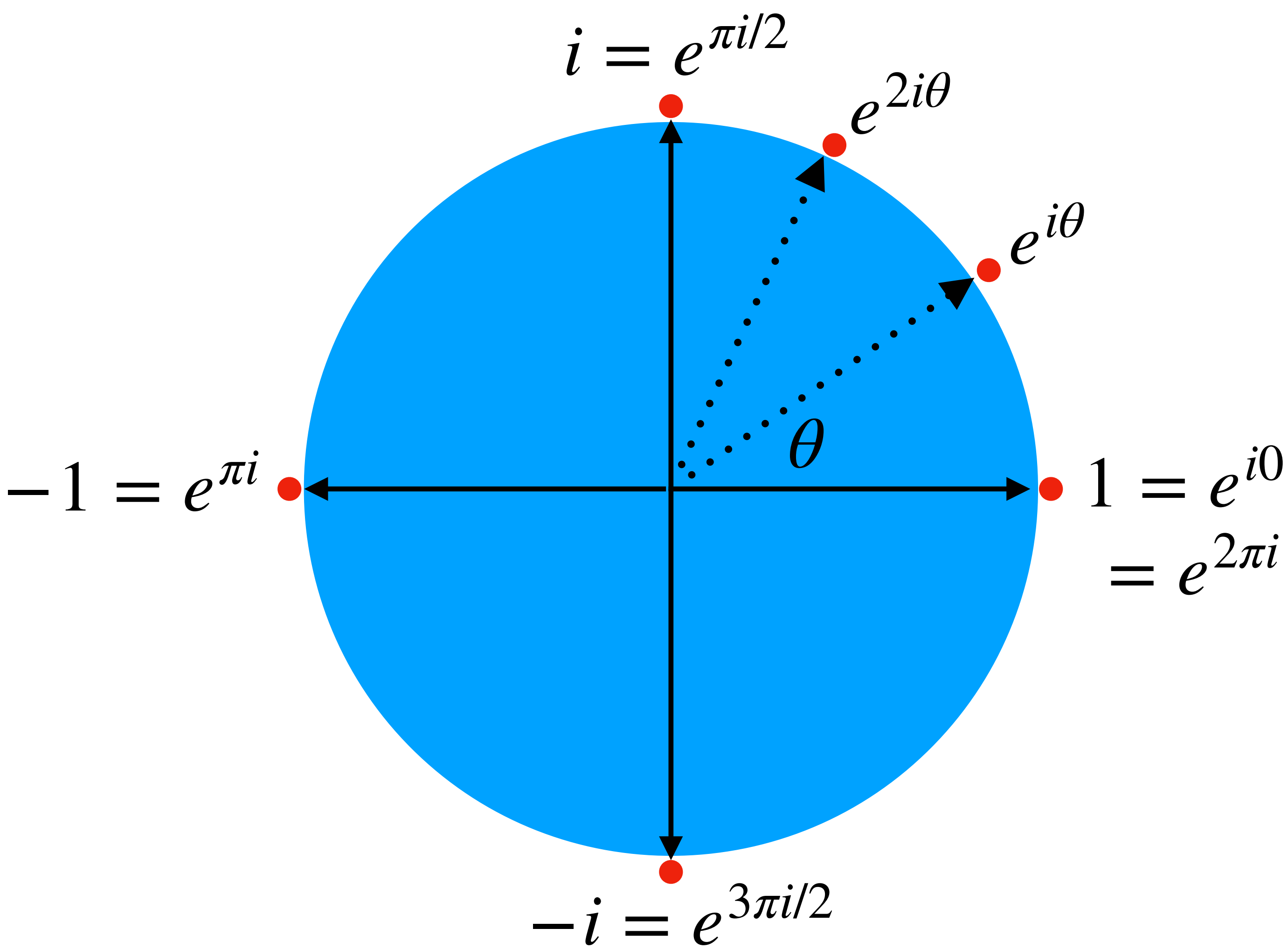
Running time

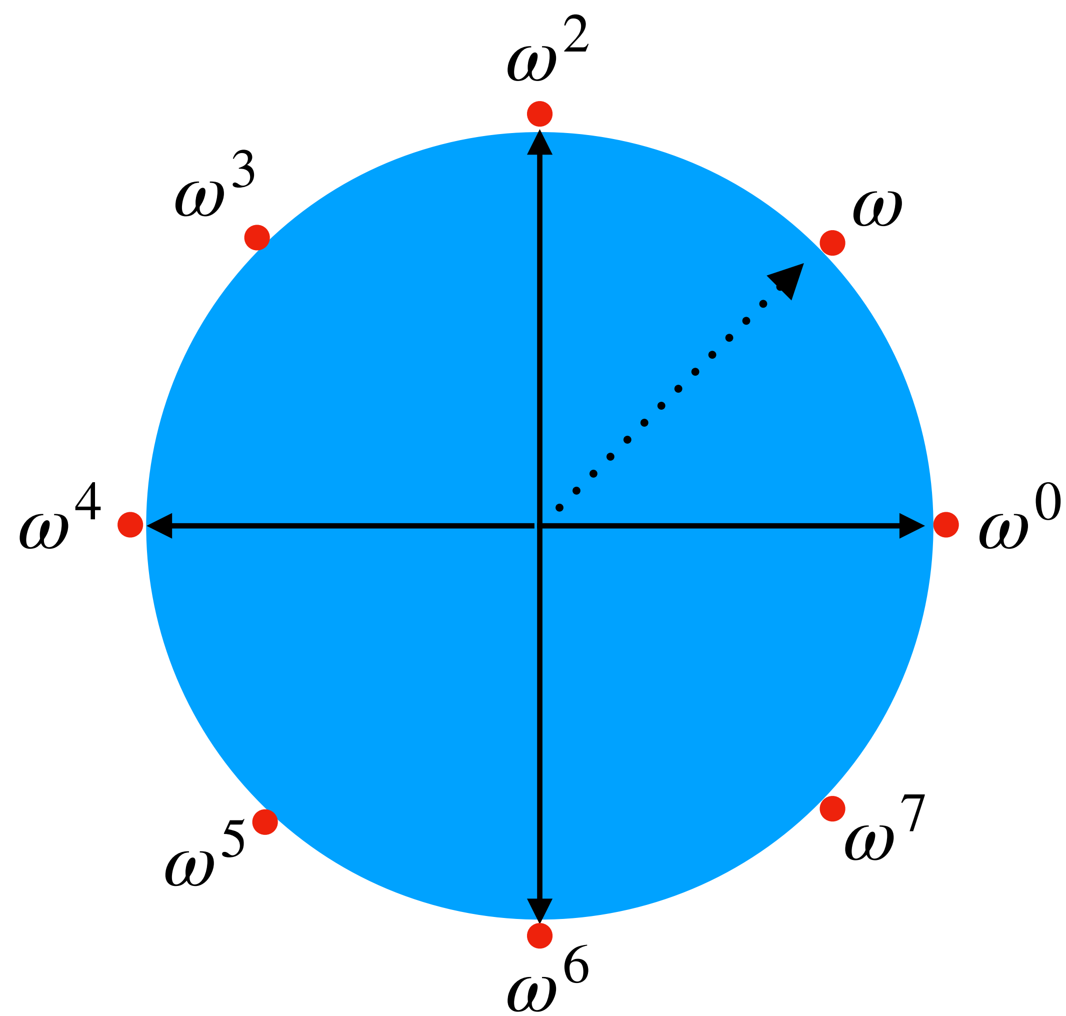
$$T(n) \leq 2T(n/2) + O(n)$$

so running time is

$$O(n \log n)$$

ω, a root of unity





$\omega = e^{2\pi i/m}$

$\omega^{jm} = (\omega^m)^j = 1^j = 1,$
So $1, \omega, \omega^2, \ldots, \omega^{m-1}$ are the m roots of unity, solutions to $X^m = 1$.

ω, a root of unity

Key properties

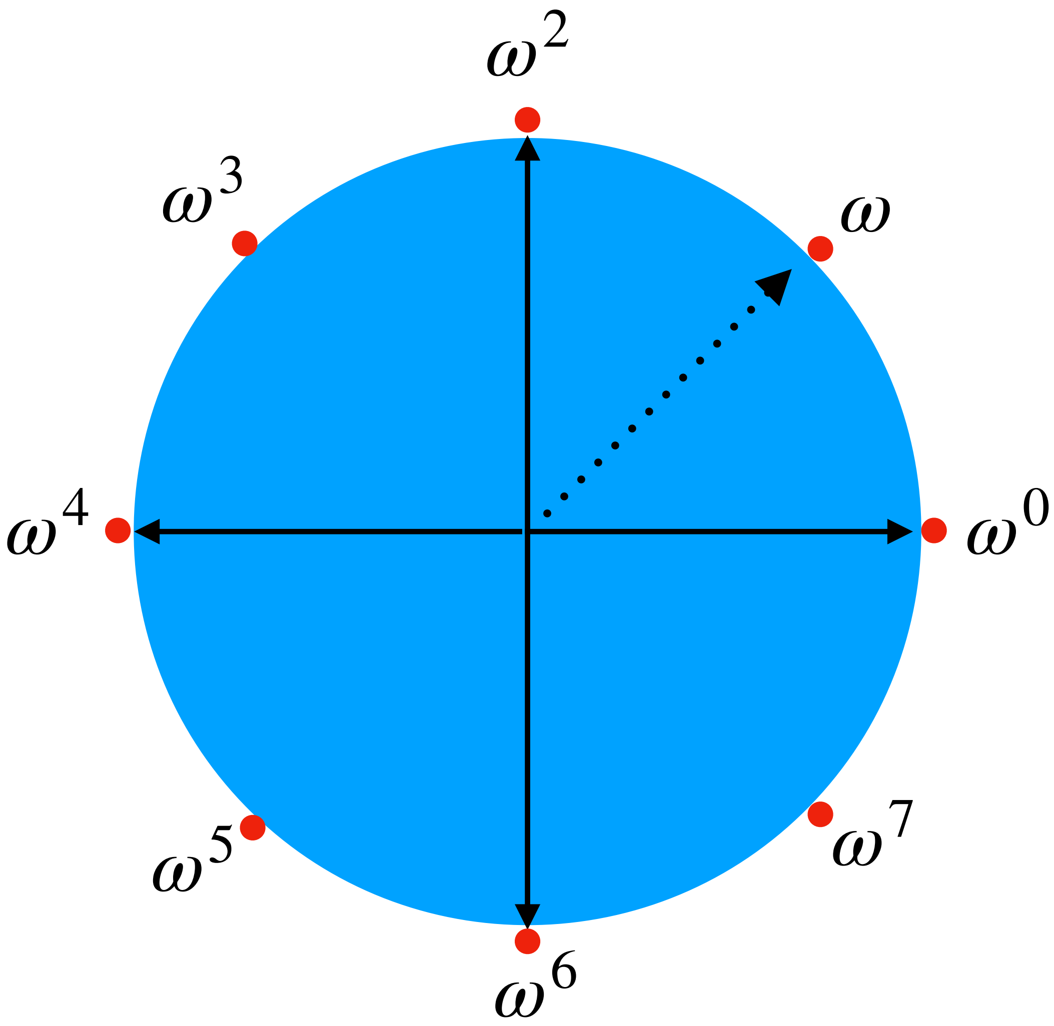
$$\omega^{-1} = \omega^{m-1}$$

$$1 + \omega^{j} + \omega^{2j} + \ldots + \omega^{(m-1)j} = 0$$

if

$$j = 1,2,\ldots, m-1.$$

If $j = 0$, it is m .



$\omega^{jm} = (\omega^m)^j = 1^j = 1,$
So $1, \omega, \omega^2, \ldots, \omega^{m-1}$ are the m roots of unity, solutions to $X^m = 1.$

Given:

$$p(X) = p_0 + p_1 X + \ldots + p_m X^m$$

Compute:

$$p(1), p(\omega), p(\omega^2), \ldots, p(\omega^{m-1})$$

Divide and Conquer Algorithm:

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

If m is even, we are evaluating each polynomial on only $m/2$ points!

Running time

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

so running time is

$$O(n \log n)$$

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>Running time</u>**
1. Set m to be a power of 2, $m > 2n$.	
2. Compute $p(1), p(\omega), p(\omega^2), \ldots, p(\omega^{m-1})$.	$O(n \log n)$
3. Compute $q(1), q(\omega), q(\omega^2), \ldots, q(\omega^{m-1})$.	$O(n \log n)$
4. Compute $r(1), r(\omega), \ldots, r(\omega^{m-1})$.	$O(n) : r(\omega^j) = p(\omega^j) \cdot q(\omega^j)$
5. Compute $r(X)$.	$O(n \log n)$

The catch: ω is a complex number!

Given:

$$r(1), r(\omega), \ldots, r(\omega^{m-1})$$

Compute:

$$r(X) = r_0 + r_1 X + \ldots + r_{m-1} X^{m-1}$$

Algorithm:

1. Let $q(Y) = r(1) + r(\omega) \cdot Y + \ldots + r(\omega^{m-1}) \cdot Y^{m-1}$.
2. Compute $q(1), q(\omega), \ldots, q(\omega^{m-1})$ using divide and conquer algorithm.
3. Set $r_j = q(\omega^{m-j})/m$.

Observe

$$q(\omega^{-t}) = \sum_{k=0}^{m-1} r(\omega^k) \cdot \omega^{-tk}$$
$$= \sum_{k=0}^{m-1} \sum_{\ell=0}^{m-1} r_\ell \cdot \omega^{\ell k} \cdot \omega^{-tk}$$
$$= \sum_{\ell=0}^{m-1} r_\ell \cdot \sum_{k=0}^{m-1} \omega^{(\ell - t)k}$$
$$= m \cdot r_t .$$

Running time

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

so running time is

$$O(n \log n)$$

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 $O(n \log n)$.

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

The catch: ω is a complex number!