Fast Fourier Transform

Karatsuba's algorithm for multiplication

Given:

$p = p_0 + p_1 \cdot 10 + \ldots + p_{n-1} \cdot 10^{n-1}, \quad q = q_0 + q_1 \cdot 10 + \ldots + q_{n-1} \cdot 10^{n-1}$

Algorithm:

1. Let $p = a + 10^{n/2} \cdot b$ and $q = c + 10^{n/2} \cdot d$, with
   $a = p_0 + \ldots + p_{n/2-1} \cdot 10^{n/2-1}, b = p_{n/2} + \ldots + p_{n-1} \cdot 10^{n/2-1},$
   $c = q_0 + \ldots + q_{n/2-1} \cdot 10^{n/2-1}, d = q_{n/2} + \ldots + q_{n-1} \cdot 10^{n/2-1}$
2. Then $p \cdot q = ac + 10^{n/2} \cdot (bc + ad) + 10^n \cdot bd$
3. Recursively compute $ac, (a+b)(c+d), bd$.
4. Output $p \cdot q = ac + 10^{n/2} \cdot ((a+b)(c+d) - ac - bd) + 10^n \cdot bd$.

Running time:

$T(n) = 3T(n/2) + O(n), T(n) = O(n^{\log_2 3}) = O(n^{1.58})$.

Karatsuba's algorithm for multiplication

Given:

$$p(X) = p_0 + p_1 \cdot X + \ldots + p_{n-1} \cdot X^{n-1}, \qquad q(X) = q_0 + q_1 \cdot X + \ldots + q_{n-1} \cdot X^{n-1}$$

Algorithm:

1. Let $p(X) = a(X) + X^{n/2} \cdot b(X)$ and $q(X) = c(X) + X^{n/2} \cdot d(X)$, with
   $a(X) = p_0 + \ldots + p_{n/2-1} \cdot X^{n/2-1}$, $b(X) = p_{n/2} + \ldots + p_{n-1} \cdot X^{n/2-1}$,
   $c(X) = q_0 + \ldots + q_{n/2-1} \cdot X^{n/2-1}$, $d(X) = q_{n/2} + \ldots + q_{n-1} \cdot X^{n/2-1}$
2. Then $p \cdot q = ac + X^{n/2} \cdot (bc + ad) + X^n \cdot bd$
3. Recursively compute $ac, (a+b)(c+d), bd$.
4. Output $p \cdot q = ac + X^{n/2} \cdot ((a+b)(c+d) - ac - bd) + X^n \cdot bd$.

Running time:

$$T(n) = 3T(n/2) + O(n), T(n) = O(n^{\log_2 3}) = O(n^{1.58}).$$

The problem:

Given: two polynomials

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

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

$$p(X) = p_0 + p_1 X + \ldots + p_{n-1} X^{n-1} \qquad q(X) = q_0 + q_1 X + \ldots + q_{n-1} X^{n-1}$$

Compute:

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

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

Assume: n is a power of 2

FFT Outline	**Running time**
1. Compute $p(a_0), p(a_1), p(a_2), \ldots, p(a_{2n-1})$.	$O(n \log n)$
2. Compute $q(a_0), q(a_1), q(a_2), \ldots, q(a_{2n-1})$.	$O(n \log n)$
3. Compute $r(a_0), r(a_1), \ldots, r(a_{2n-1})$.	$O(n) : r(a_j) = p(a_j) \cdot q(a_j)$
4. Compute $r(X)$.	$O(n \log n)$

Given:

$$p(X) = p_0 + p_1 X + \ldots + p_{n-1} X^{n-1}$$

Compute:

$$p(a_0), p(a_1), p(a_2), \ldots, p(a_{n-1})$$

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 $p_e(a_0^2), p_e(a_1^2), \ldots, p_e(a_{n-1}^2)$ and
   $p_o(a_0^2), p_o(a_1^2), \ldots, p(a_{n-1}^2)$.
3. Combine the results to compute $p(a_0), \ldots, p(a_{n-1})$, by setting
   $p(a_j) = p_e(a_j^2) + a_j \cdot p_o(a_j^2)$

Running time

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

so running time is

$$O(n \log n)$$

Given:

$$p(X) = p_0 + p_1 X + \ldots + p_{n-1} X^{n-1}$$

Compute:

$$p(a_0), p(a_1), p(a_2), \ldots, p(a_{n-1})$$

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 $p_e(a_0^2), p_e(a_1^2), \ldots, p_e(a_{n-1}^2)$ and
   $p_o(a_0^2), p_o(a_1^2), \ldots, p(a_{n-1}^2)$.
3. Combine the results to compute $p(a_0), \ldots, p(a_{n-1})$, by setting
   $p(a_j) = p_e(a_j^2) + a_j \cdot p_o(a_j^2)$

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

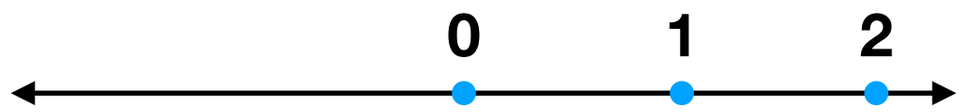
$$O(n \log n)$$

Two ways to think about numbers

Algebraic

Numbers are elements of a set, with rules for how to add and multiply

Geometric

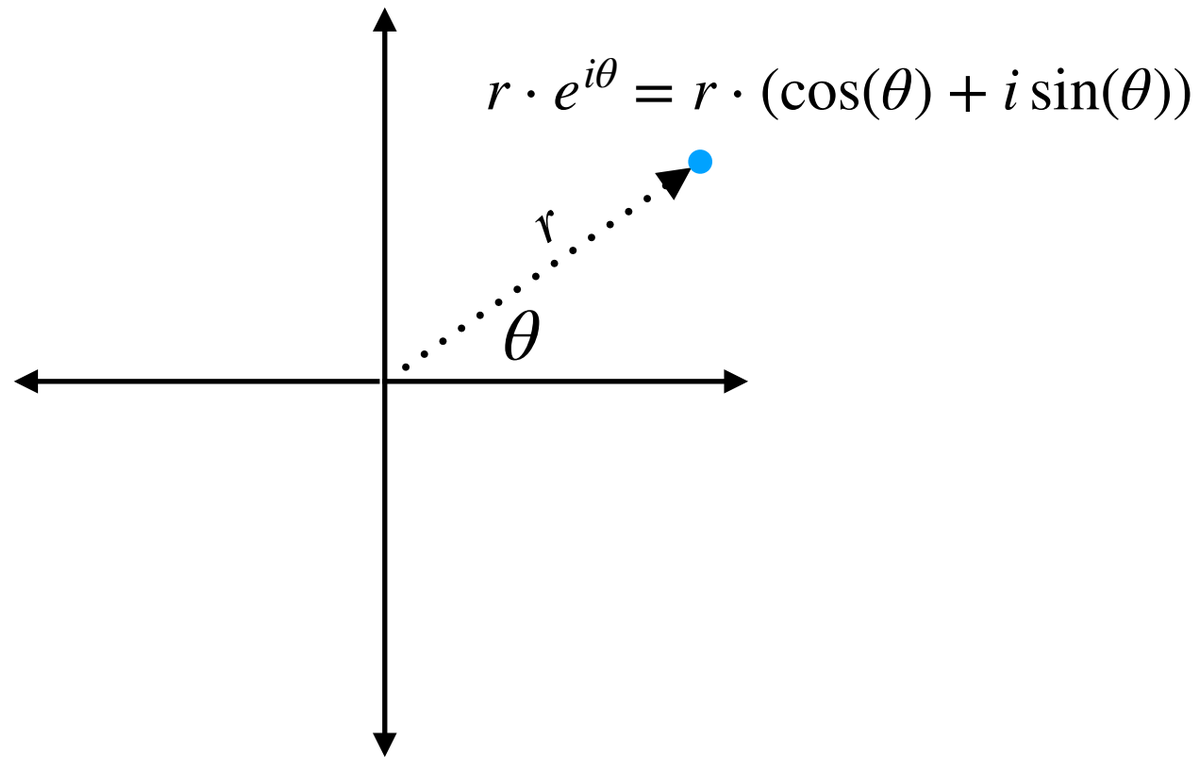


Complex Numbers

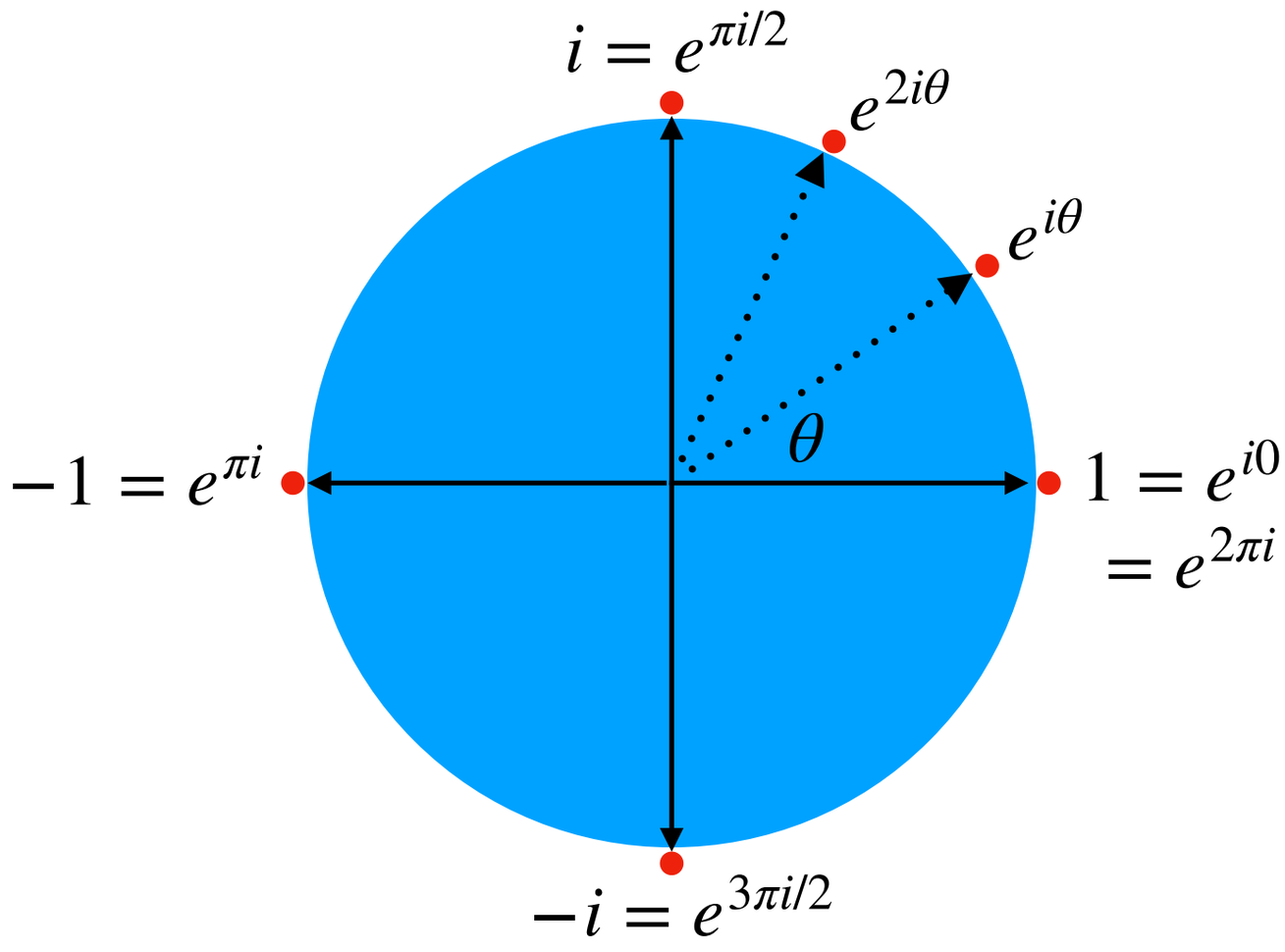
Algebraic

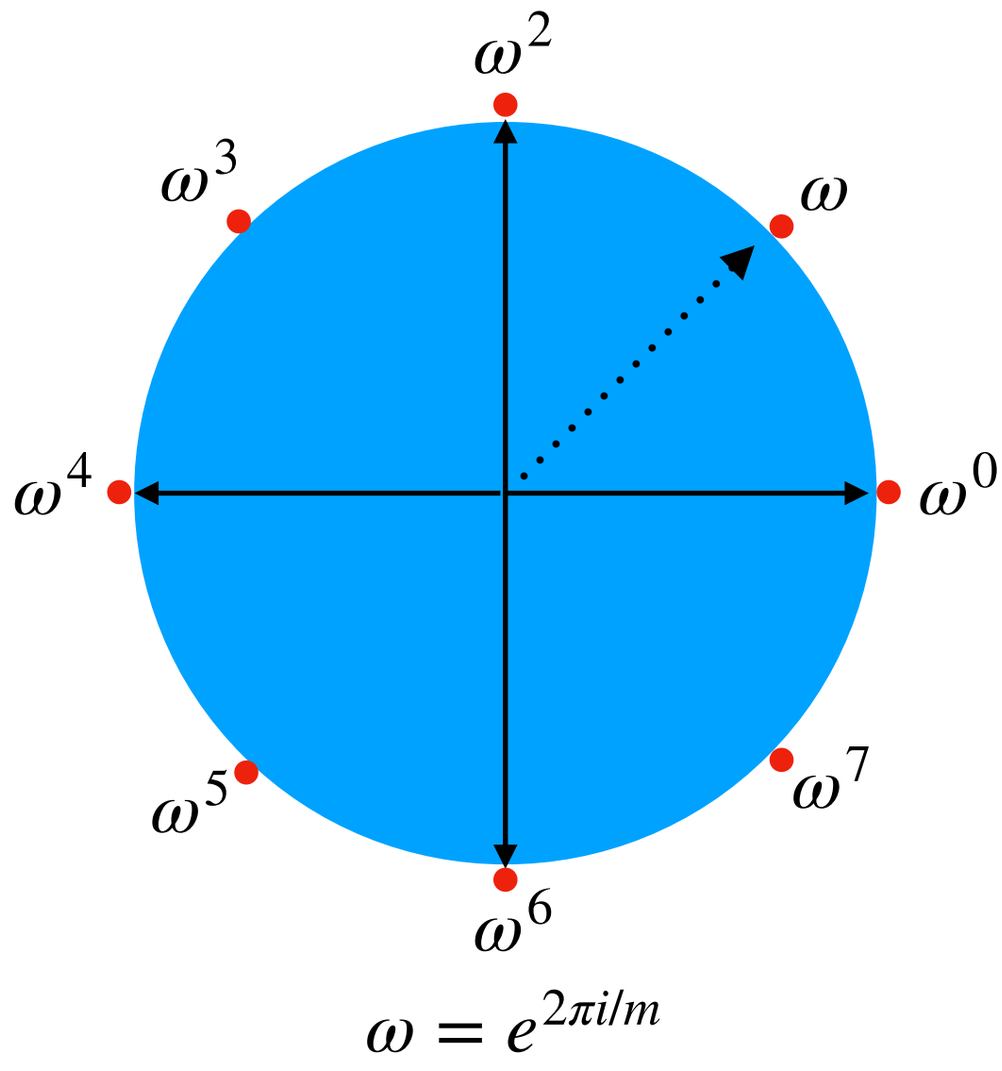
$a + i \cdot b$, where $i^2 = -1$,
and a, b are real numbers.

Geometric



ω, a root of unity





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

ω, a root of unity

Key properties

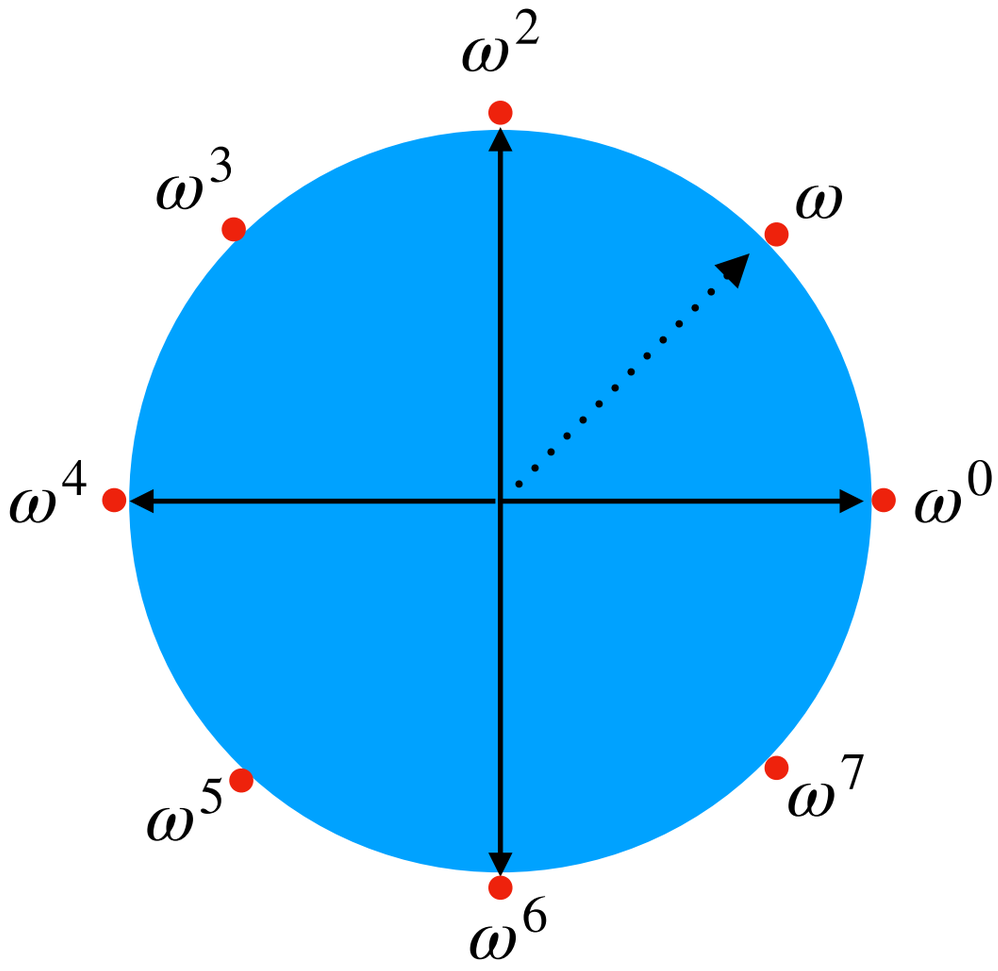
$$\omega^{-1} = \omega^{n-1}$$

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

if

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

If $j = 0$, it is n .



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

$\omega^{jn} = (\omega^n)^j = 1^j = 1,$
So $1,\omega^2, \omega^4, \ldots, \omega^{2(n-1)}$ are all $n/2$'th roots of unity!

Given:

$$p(X) = p_0 + p_1 X + \ldots + p_{n-1} X^{n-1}$$

Compute:

$$p(a_0), p(a_1), p(a_2), \ldots, p(a_{n-1})$$

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 $p_e(1), p_e(\omega^2), \ldots, p_e(\omega^{2(n-1)})$ and
   $p_o(1), p_o(\omega^2), \ldots, p(\omega^{2(n-1)})$.
3. Combine the results to compute $p(1), p(\omega), \ldots, p(\omega^{n-1})$, by
   setting $p(\omega^j) = p_e(\omega^{2j}) + \omega \cdot p_o(\omega^{2j})$

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

Running time

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

so running time is

$$O(n \log n)$$

Given: two polynomials

$$p(X) = p_0 + p_1 X + \ldots + p_{n-1} X^{n-1} \qquad q(X) = q_0 + q_1 X + \ldots + q_{n-1} X^{n-1}$$

Compute:

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

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

Assume: n is a power of 2

FFT Outline	**Running time**
1. Compute $p(a_0), p(a_1), p(a_2), \ldots, p(a_{2n-1})$.	$O(n \log n)$
2. Compute $q(a_0), q(a_1), q(a_2), \ldots, q(a_{2n-1})$.	$O(n \log n)$
3. Compute $r(a_0), r(a_1), \ldots, r(a_{2n-1})$.	$O(n) : r(a_j) = p(a_j) \cdot q(a_j)$
4. Compute $r(X)$.	$O(n \log n)$

Given:

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

Compute:

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

Algorithm:

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

Running time

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

so running time is

$$O(n \log n)$$

Observe

$$q(\omega^{-t}) = \sum_{k=0}^{n-1} r(\omega^k) \cdot \omega^{-tk}$$

$$= \sum_{k=0}^{n-1} \sum_{\ell=0}^{n-1} r_\ell \cdot \omega^{\ell k} \cdot \omega^{-tk}$$

$$= \sum_{\ell=0}^{n-1} r_\ell \cdot \sum_{k=0}^{n-1} \omega^{(\ell - t)k}$$

$$= n \cdot r_t .$$

Given: two polynomials

$$p(X) = p_0 + p_1 X + \ldots + p_{n-1} X^{n-1} \qquad q(X) = q_0 + q_1 X + \ldots + q_{n-1} X^{n-1}$$

Compute:

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

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

Assume: n is a power of 2

FFT Outline	**Running time**
1. Compute $p(a_0), p(a_1), p(a_2), \ldots, p(a_{2n-1})$.	$O(n \log n)$
2. Compute $q(a_0), q(a_1), q(a_2), \ldots, q(a_{2n-1})$.	$O(n \log n)$
3. Compute $r(a_0), r(a_1), \ldots, r(a_{2n-1})$.	$O(n) : r(a_j) = p(a_j) \cdot q(a_j)$
4. Compute $r(X)$.	$O(n \log n)$

To multiply integers

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

where

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

Running time:
$O(n \log n \log \log n)$

It is complicated because we have to handle arithmetic involving complex numbers.