Fast Fourier Transform

Jean Baptiste Joseph Fourier (1768-1830)

Fast Fourier Transform

Applications.

- Perhaps single algorithmic discovery that has had the greatest practical impact in history.
- Optics, acoustics, quantum physics, telecommunications, systems theory, signal processing, speech recognition, data compression.
- Progress in these areas limited by lack of fast algorithms.

History.

- Cooley-Tukey (1965) revolutionized all of these areas.
- Danielson-Lanczos (1942) efficient algorithm.
- Runge-König (1924) laid theoretical groundwork.
- Gauss (1805, 1866) describes similar algorithm.
- Importance not realized until advent of digital computers.

Polynomials: Coefficient Representation

Degree n polynomial.

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

$$q(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_{n-1} x^{n-1}$$

Addition: O(n) ops.

$$p(x) + q(x) = (a_0 + b_0) + (a_1 + b_1) x + \cdots + (a_{n-1} + b_{n-1}) x^{n-1}$$

Evaluation: O(n) using Horner's method.

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

Multiplication (convolution): O(n^2).

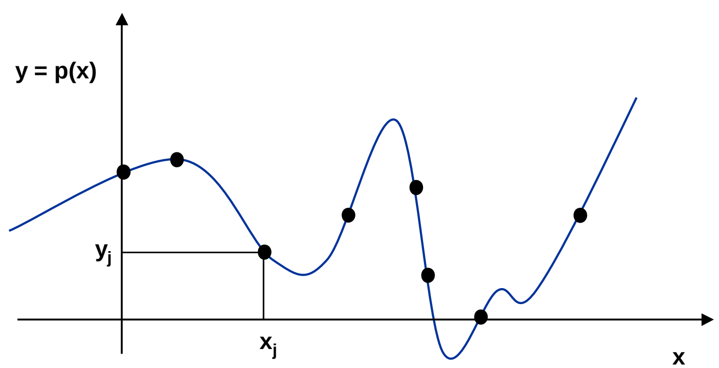
$$p(x) \times q(x) = (a_0 b_0) + (a_0 b_1 + a_1 b_0) x + \cdots + (\sum_{k=0}^{j} a_k b_{j-k}) x^j + \cdots + (a_{n-1} b_{n-1}) x^{2n-2}$$

Polynomials: Point-Value Representation

Degree n polynomial.

- Uniquely specified by knowing p(x) at n different values of x.

$$\{(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}, \text{ where } y_k = p(x_k)$$



Polynomials: Point-Value Representation

Degree n polynomial.

$$\{(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}, \quad \text{where } y_k = p(x_k)$$

$$\{(x_0, z_0), (x_1, z_1), \ldots, (x_{n-1}, z_{n-1})\}, \quad \text{where } z_k = q(x_k)$$

Addition: O(n).

$$\{(x_0,\ y_0 + z_0),\ (x_1,\ y_1 + z_1),\ \ldots,\ (x_{n-1},\ y_{n-1} + z_{n-1})\}$$

Multiplication: O(n), but need 2n points.

$$\{(x_0,\ y_0\, z_0),\ (x_1,\ y_1\, z_1), \ldots, (x_{2n-1},\ y_{2n-1}\, z_{2n-1})\}$$

Evaluation: O(n^2) using Lagrange's formula.

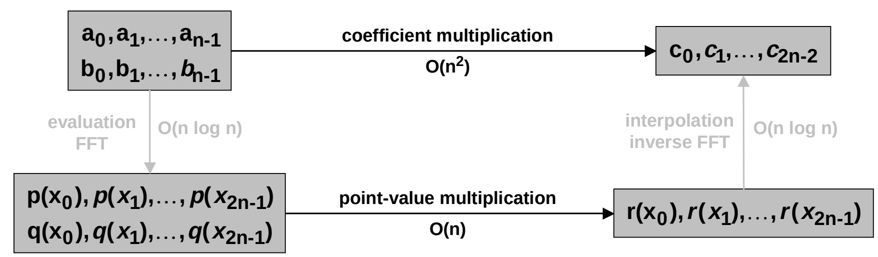
$$p(x) = \sum_{k=0}^{n-1} y_k \frac{\prod_{j \neq k} (x - x_j)}{\prod_{j \neq k} (x_k - x_j)}$$

Best of Both Worlds

Can we get "fast" multiplication and evaluation?

Representation	Multiplication	Evaluation
coefficient	O(n^2)	O(n)
point-value	O(n)	O(n^2)
FFT	O(n log n)	O(n log n)

✎ **Yes! Convert back and forth between two representations.**



Converting Between Representations: Naïve Solution

Evaluation (coefficient to point-value).

- Given a polynomial $p(x) = a_0 + a_2 x^1 + \ldots + a_{n-1} x^{n-1}$, choose n distinct points $\{x_0, x_1, \ldots, x_{n-1}\}$ and compute $y_k = p(x_k)$, for each k using Horner's method.
- O(n^2).

Interpolation (point-value to coefficient).

- Given n distinct points $\{x_0, x_1, \ldots, x_{n-1}\}$ and $y_k = p(x_k)$, compute the coefficients $\{a_0, a_1, \ldots, a_{n-1}\}$ by solving the following linear system of equations.
- Note Vandermonde matrix is invertible iff x_k are distinct.
- O(n^3).

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

Fast Interpolation: Key Idea

Key idea: choose $\{x_0, x_1, \ldots, x_{n-1}\}$ to make computation easier!

- Set $x_k = x_j$?

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_0^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

Fast Interpolation: Key Idea

Key idea: choose $\{x_0, x_1, \ldots, x_{n-1}\}$ to make computation easier!

- Set $x_k = x_j$?
- Use negative numbers: set $x_k = -x_j$ so that $(x_k)^2 = (x_j)^2$.
  - set $x_k = -x_{n/2+k}$

$$\begin{pmatrix} 1 & 17 & (17)^2 & (17)^3 & \vdots & (17)^{n-1} \\ 1 & 5 & (5)^2 & (5)^3 & \cdots & (5)^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \hline 1 & -17 & (-17)^2 & (-17)^3 & \vdots & (-17)^{n-1} \\ 1 & -5 & (-5)^2 & (-5)^3 & \cdots & (-5)^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

- $E = p_{even}(x^2) = a_0 + a_2 17^2 + a_4 17^4 + a_6 17^6 + \ldots + a_{n-2} 17^{n-2}$
- $O = x\, p_{odd}(x^2) = a_1 17 + a_3 17^3 + a_5 17^5 + a_7 17^7 + \ldots + a_{n-1} 17^{n-1}$
- $y_0 = E + O, \quad y_{n/2} = E - O$

Fast Interpolation: Key Idea

Key idea: choose $\{x_0, x_1, \ldots, x_{n-1}\}$ to make computation easier!

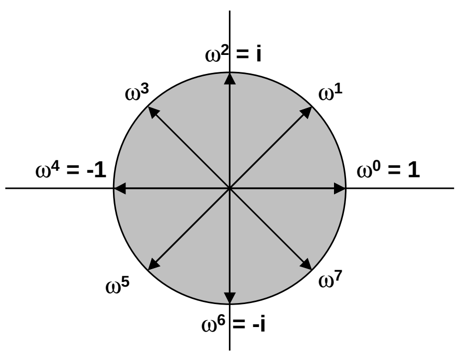
- Set $x_k = x_j$?
- Use negative numbers: set $x_k = -x_j$ so that $(x_k)^2 = (x_j)^2$.
  - set $x_k = -x_{n/2+k}$
- Use complex numbers: set $x_k = \omega^k$ where ω is n^{th} root of unity.
  - $(x_k)^2 = (-x_{n/2+k})^2$
  - $(x_k)^4 = (-x_{n/4+k})^4$
  - $(x_k)^8 = (-x_{n/8+k})^8$

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(n-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \omega^{3(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

Roots of Unity

An n^{th} root of unity is a complex number z such that $z^n = 1$.

- $\omega = e^{2\pi i/n}$ = principal n^{th} root of unity.
- $e^{it} = \cos t + i \sin t$.
- $i^2 = -1$.
- There are exactly n roots of unity: ω^k, $k = 0, 1, \ldots, n-1$.



Roots of Unity: Properties

L1: Let ω be the principal n^{th} root of unity. If $n > 0$, then $\omega^{n/2} = -1$.

- Proof: $\omega = e^{2\pi i/n} \Rightarrow \omega^{n/2} = e^{\pi i} = -1$. (Euler's formula)

L2: Let $n > 0$ be even, and let ω and ν be the principal n^{th} and $(n/2)^{th}$ roots of unity. Then $(\omega^k)^2 = \nu^k$.

- Proof: $(\omega^k)^2 = e^{(2k)2\pi i/n} = e^{(k)2\pi i/(n/2)} = \nu^k$.

L3: Let $n > 0$ be even. Then, the squares of the n complex n^{th} roots of unity are the n/2 complex $(n/2)^{th}$ roots of unity.

- Proof: If we square all of the n^{th} roots of unity, then each $(n/2)^{th}$ root is obtained exactly twice since:
  - L1 $\Rightarrow \omega^{k+n/2} = -\omega^k$
  - thus, $(\omega^{k+n/2})^2 = (\omega^k)^2$
  - L2 $\Rightarrow$ both of these $= \nu^k$
  - $\omega^{k+n/2}$ and ω^k have the same square

Divide-and-Conquer

Given degree n polynomial $p(x) = a_0 + a_1x^1 + a_2 x^2 + . . . + a_{n-1} x^{n-1}$.

- Assume n is a power of 2, and let ω be the principal n th root of unity.
- Define even and odd polynomials:
  - $p_{even}(x) := a_0 + a_2x^1 + a_4x^2 + a_6x^3 + . . . + a_{n-2} x^{n/2 - 1}$
  - $p_{odd}(x) := a_1 + a_3x^1 + a_5x^2 + a_7x^3 + . . . + a_{n-1} x^{n/2 - 1}$
  - $p(x) = p_{even}(x^2) + x\, p_{odd}(x^2)$
- Reduces problem of
  - evaluating degree n polynomial p(x) at $\omega^0, \omega^1, . . . , \omega^{n-1}$

  to
  - evaluating two degree n/2 polynomials at: $(\omega^0)^2, (\omega^1)^2, . . . , (\omega^{n-1})^2$.
- L3 $\Rightarrow$ $p_{even}(x)$ and $p_{odd}(x)$ only evaluated at n/2 complex (n/2) th roots of unity.

FFT Algorithm

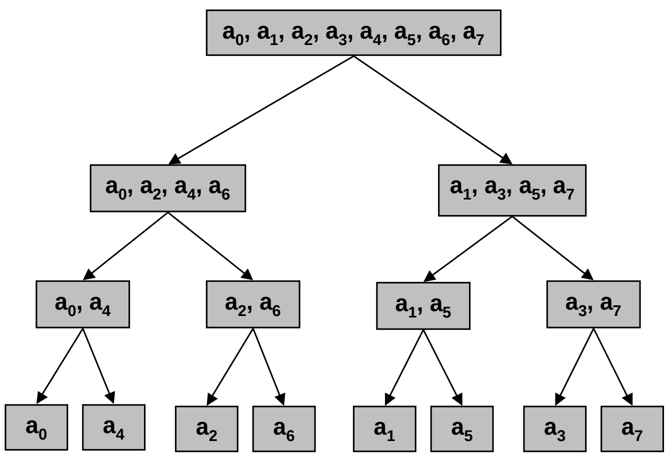
FFT (n, $a_0, a_1, a_2, . . . , a_{n-1}$)

```
if (n == 1)            // n is a power of 2
   return a0

ω ← e 2π i / n
(e0,e1,e2,...,en/2-1) ← FFT(n/2, a0,a2,a4,...,an-2)
(d0,d1,d2,...,dn/2-1) ← FFT(n/2, a1,a3,a5,...,an-1)

for k = 0 to n/2 - 1
      yk ← ek + ωk dk
   yk+n/2 ← ek - ωk dk

return (y0,y1,y2,...,yn-1)
```

O(n) complex multiplies if we pre-compute ω^k.

$$T(n) = 2T(n/2) + O(n) \;\Rightarrow\; T(n) = O(n \log n)$$

Recursion Tree



"bit-reversed" order

Proof of Correctness

Proof of correctness. Need to show $y_k = p(\omega^k)$ for each k = 0, . . . , n-1, where ω is the principal n th root of unity.

$$p(\omega^k) = \sum_{j=0}^{n-1} a_j\, \omega^{kj}$$

- Base case. n = 1 $\Rightarrow$ ω = 1. Algorithm returns $y_0 = a_0 = a_0\,\omega^0$.
- Induction step. Assume algorithm correct for n / 2.
  - let ν be the principal (n/2) th root of unity
  - $e_k = p_{even}(\nu^k) = p_{even}(\omega^{2k})$ by Lemma 2
  - $d_k = p_{odd}(\nu^k) = p_{odd}(\omega^{2k})$ by Lemma 2
  - recall $p(x) = p_{even}(x^2) + x\, p_{odd}(x^2)$

$$\begin{aligned} y_k &= e_k + \omega^k d_k \\ &= p_{even}(\omega^{2k}) + \omega^k p_{odd}(\omega^{2k}) \\ &= p(\omega^k) \end{aligned}$$

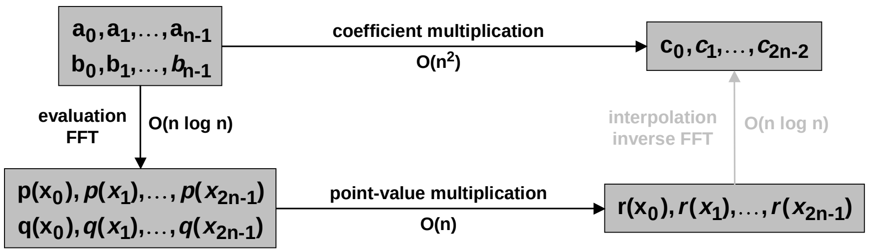
$$\begin{aligned} y_{k+n/2} &= e_k - \omega^k d_k \\ &= p_{even}(\omega^{2k}) - \omega^k p_{odd}(\omega^{2k}) \\ &= p_{even}(\omega^{2k}) + \omega^{k+n/2} p_{odd}(\omega^{2k}) \\ & p_{even}(\omega^{2k+n}) + \omega^{k+n/2} p_{odd}(\omega^{2k+n}) \\ & p(\omega^{k+n/2}) \end{aligned}$$

Best of Both Worlds

Can we get "fast" multiplication and evaluation?

Representation	Multiplication	Evaluation
coefficient	$O(n^2)$	$O(n)$
point-value	$O(n)$	$O(n^2)$
FFT	$O(n \log n)$	$O(n \log n)$

✎ Yes! Convert back and forth between two representations.



Inverse FFT

Forward FFT: given $\{a_0, a_1, \ldots, a_{n-1}\}$, compute $\{y_0, y_1, \ldots, y_{n-1}\}$.

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(n-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \omega^{3(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

Inverse FFT: given $\{y_0, y_1, \ldots, y_{n-1}\}$ compute $\{a_0, a_1, \ldots, a_{n-1}\}$.

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(n-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \omega^{3(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{pmatrix}^{-1} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

Inverse FFT

Great news: same algorithm as FFT, except use ω^{-1} as "principal" n^{th} root of unity (and divide by n).

$$F_n = \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 & \cdots & \omega^{(n-1)} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(n-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{(n-1)} & \omega^{2(n-1)} & \omega^{3(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{pmatrix}$$

$$F_n^{-1} = \frac{1}{n} \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \omega^{-3} & \cdots & \omega^{-(n-1)} \\ 1 & \omega^{-2} & \omega^{-4} & \omega^{-6} & \cdots & \omega^{-2(n-1)} \\ 1 & \omega^{-3} & \omega^{-6} & \omega^{-9} & \cdots & \omega^{-3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{-(n-1)} & \omega^{-2(n-1)} & \omega^{-3(n-1)} & \cdots & \omega^{-(n-1)(n-1)} \end{pmatrix}$$

Inverse FFT: Proof of Correctness

Summation lemma. Let ω be a primitive n^{th} root of unity. Then

$$\sum_{j=0}^{n-1} \omega^{kj} = \begin{cases} n & k \equiv 0 \bmod n \\ 0 & \text{otherwise} \end{cases}$$

- If k is a multiple of n then $\omega^k = 1$.
- Each n^{th} root of unity ω^k is a root of $x^n - 1 = (x - 1)(1 + x + x^2 + \ldots + x^{n-1})$, if $\omega^k \neq 1$ we have: $1 + \omega^k + \omega^{k(2)} + \ldots + \omega^{k(n-1)} = 0$.

Claim: F_n and F_n^{-1} are inverses.

$$\left(F_n F_n^{-1}\right)_{i\,i'} = \sum_{j=0}^{n-1} \omega^{ij} \frac{\omega^{-j\,i'}}{n} = \frac{1}{n} \sum_{j=0}^{n-1} \omega^{(i-i')\,j} = \begin{cases} 1 & \text{if } i = i' \\ 0 & \text{otherwise} \end{cases}$$

Inverse FFT: Algorithm

IFFT (n, a_0, a_1, a_2, . . . , a_{n-1})

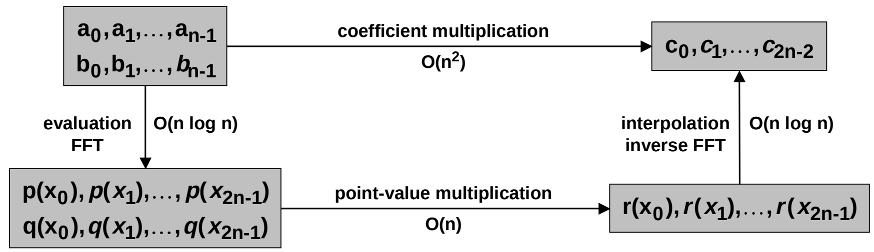
```
if (n == 1)          // n is a power of 2
   return a0

ω  ← e^(-2πi/n)
(e0,e1,e2,...,en/2-1) ← FFT(n/2, a0,a2,a4,...,an-2)
(d0,d1,d2,...,dn/2-1) ← FFT(n/2, a1,a3,a5,...,an-1)

for k = 0 to n/2 - 1
      yk ← (ek + ω^k dk ) / n
   yk+n/2 ← (ek - ω^k dk ) / n

return (y0,y1,y2,...,yn-1)
```


Best of Both Worlds

Can we get "fast" multiplication and evaluation?

Representation	Multiplication	Evaluation
coefficient	$O(n^2)$	O(n)
point-value	O(n)	$O(n^2)$
FFT	O(n log n)	O(n log n)

Yes! Convert back and forth between two representations.



Integer Arithmetic

Multiply two n-digit integers: $a = a_{n-1} \ldots a_1 a_0$ and $b = b_{n-1} \ldots b_1 b_0$.

- Form two degree n polynomials.
- Note: a = p(10), b = q(10).

$$p(x) = \sum_{j=0}^{n-1} a_j x^j \qquad q(x) = \sum_{j=0}^{n-1} b_j x^j$$

- Compute product using FFT in O(n log n) steps.
- Evaluate r(10) = a × b.
- Problem: O(n log n) complex arithmetic steps.

$$r(x) = p(x) \times q(x)$$

Solution.

- Strassen (1968): carry out arithmetic to suitable precision.
  - $T(n) = O(n\, T(\log n)) \Rightarrow T(n) = O(n \log n (\log \log n)^{1+\varepsilon})$
- Schönhage-Strassen (1971): use modular arithmetic.
  - $T(n) = O(n \log n \log \log n)$