



# Chapter 1

# Divide and Conquer

## Part 2: Polynomial Multiplication

Algorithm Theory  
WS 2012/13

Fabian Kuhn

# Polynomials

---

**Real polynomial  $p$  in one variable  $x$ :**

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x^1 + a_0$$

Coefficients of  $p$ :  $a_0, a_1, \dots, a_n \in \mathbb{R}$

**Degree** of  $p$ : largest power of  $x$  in  $p$  ( $n$  in the above case)

**Example:**

$$p(x) = 3x^3 - 15x^2 + 18x$$

Set of all real-valued polynomials in  $x$ :  $\mathbb{R}[x]$  (polynomial ring)

# Operations: Addition

---

- Given: Polynomials  $p, q \in \mathbb{R}[x]$  of degree  $n$

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

$$q(x) = b_n x^n + b_{n-1} x^{n-1} + \dots + b_1 x + b_0$$

- Compute sum  $p(x) + q(x)$ :

$$\begin{aligned} p(x) + q(x) &= (a_n x^n + \dots + a_0) + (b_n x^n + \dots + b_0) \\ &= (a_n + b_n) x^n + \dots + (a_1 + b_1) x + (a_0 + b_0) \end{aligned}$$

# Operations: Multiplication

- Given: Polynomials  $p, q \in \mathbb{R}[x]$  of degree  $n$

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

$$q(x) = b_n x^n + b_{n-1} x^{n-1} + \dots + b_1 x + b_0$$

- Product  $p(x) \cdot q(x)$ :

$$\begin{aligned} p(x) \cdot q(x) &= (a_n x^n + \dots + a_0) \cdot (b_n x^n + \dots + b_0) \\ &= c_{2n} x^{2n} + c_{2n-1} x^{2n-1} + \dots + c_1 x + c_0 \end{aligned}$$

- Obtaining  $c_i$ : what products of monomials have degree  $i$ ?

$$\text{For } 0 \leq i \leq 2n: c_i = \sum_{j=0}^i a_j b_{i-j}$$

where  $a_i = b_i = 0$  for  $i > n$ .

# Operations: Evaluation

- Given: Polynomial  $p \in \mathbb{R}[x]$  of degree  $n$

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

- **Horner's method** for evaluation at specific value  $x_0$ :

$$p(x_0) = (\dots ((a_n x_0 + a_{n-1})x_0 + a_{n-2})x_0 + \dots + a_1)x_0 + a_0$$

- Pseudo-code:

```
 $p := a_n; i := n;$   
while ( $i > 0$ ) do  
     $i := i - 1;$   
     $p := p \cdot x_0 + a_i$   
end
```

- Running time:  $O(n)$

# Representation of Polynomials

---

## Coefficient representation:

- Polynomial  $p(x) \in \mathbb{R}[x]$  of degree  $n$  is given by its  $n + 1$  coefficients  $a_0, \dots, a_n$ :

$$p(x) = a_n x^n + \dots + a_1 x + a_0$$

- Example:

$$p(x) = 3x^3 - 15x^2 + 18x$$

- The most typical (and probably most natural) representation of polynomials

# Representation of Polynomials

## Product of linear factors:

- Polynomial  $p(x) \in \mathbb{C}[x]$  of degree  $n$  is given by its  $n$  roots

$$p(x) = a_n \cdot (x - x_1) \cdot (x - x_2) \cdot \dots \cdot (x - x_n)$$

- Example:

$$p(x) = 3x(x - 2)(x - 3)$$

- Every polynomial has exactly  $n$  roots  $x_i \in \mathbb{C}$  for which  $p(x_i) = 0$ 
  - Polynomial is uniquely defined by the  $n$  roots and  $a_n$
- We will not use this representation...

# Representation of Polynomials

## Point-value representation:

- Polynomial  $p(x) \in \mathbb{R}[x]$  of degree  $n$  is given by  $n + 1$  point-value pairs:

$$p = \{(x_0, p(x_0)), (x_1, p(x_1)), \dots, (x_n, p(x_n))\}$$

where  $x_i \neq x_j$  for  $i \neq j$ .

- Example: The polynomial

$$p(x) = 3x(x - 2)(x - 3)$$

is uniquely defined by the four point-value pairs  $(0,0), (1,6), (2,0), (3,0)$ .



# Operations: Coefficient Representation



Deg.- $n$  polynomials  $p(x) = a_n x^n + \dots + a_0$ ,  $q(x) = b_n x^n + \dots + b_0$

## Addition:

$$p(x) + q(x) = (a_n + b_n)x^n + \dots + (a_0 + b_0)$$

- Time:  $O(n)$

## Multiplication:

$$p(x) \cdot q(x) = c_{2n} x^{2n} + \dots + c_0, \quad \text{where } c_i = \sum_{j=0}^i a_j b_{i-j}$$

- Naive solution: Need to compute product  $a_i b_j$  for all  $0 \leq i, j \leq n$
- Time:  $O(n^2)$

# Operations Point-Value Representation



Degree- $n$  polynomials

$$p = \{(x_0, p(x_0)), \dots, (x_n, p(x_n))\}, q = \{(x_0, q(x_0)), \dots, (x_n, q(x_n))\}$$

- Note: we use the same points  $x_0, \dots, x_n$  for both polynomials

**Addition:**

$$p + q = \{(x_0, p(x_0) + q(x_0)), \dots, (x_n, p(x_n) + q(x_n))\}$$

- Time:  $O(n)$

**Multiplication:**

$$p \cdot q = \{(x_0, p(x_0) \cdot q(x_0)), \dots, (x_n, p(x_n) \cdot q(x_n))\}$$

- Time:  $O(n)$

# Faster Multiplication?

- Multiplication is slow ( $\Theta(n^2)$ ) when using the standard coefficient representation
- Try **divide-and-conquer** to get a faster algorithm
- Assume: degree is  $n - 1$ ,  $n$  is even
- **Divide polynomial  $p(x) = a_{n-1}x^{n-1} + \dots + a_0$  into 2 polynomials of degree  $n/2 - 1$ :**

$$p_0(x) = a_{n/2-1}x^{n/2-1} + \dots + a_0$$

$$p_1(x) = a_{n-1}x^{n/2-1} + \dots + a_{n/2}$$

$$p(x) = p_1(x) \cdot x^{n/2} + p_0(x)$$

- Similarly:  $q(x) = q_1(x) \cdot x^{n/2} + q_0(x)$

# Use Divide-And-Conquer

- **Divide:**

$$p(x) = p_1(x) \cdot x^{n/2} + p_0(x), \quad q(x) = q_1(x) \cdot x^{n/2} + q_0(x)$$

- **Multiplication:**

$$p(x)q(x) = p_1(x)q_1(x) \cdot x^n + (p_0(x)q_1(x) + p_1(x)q_0(x)) \cdot x^{n/2} + p_0(x)q_0(x)$$

- 4 multiplications of degree  $n/2 - 1$  polynomials:

$$T(n) = 4T(n/2) + O(n)$$

- Leads to  $T(n) = \Theta(n^2)$  like the naive algorithm... (see exercises)

# More Clever Recursive Solution

---

- **Recall that**

$$p(x)q(x) = p_1(x)q_1(x) \cdot x^n + (p_0(x)q_1(x) + p_1(x)q_0(x)) \cdot x^{n/2} + p_0(x)q_0(x)$$

- Compute  $r(x) = (p_0(x) + p_1(x)) \cdot (q_0(x) + q_1(x))$ :

# Karatsuba Algorithm

- Recursive multiplication:

$$r(x) = (p_0(x) + p_1(x)) \cdot (q_0(x) + q_1(x))$$

$$p(x)q(x) = p_1(x)q_1(x) \cdot x^n$$

$$+ (r(x) - p_0(x)q_0(x) + p_1(x)q_1(x)) \cdot x^{n/2}$$

$$+ p_0(x)q_0(x)$$

- Recursively do **3 multiplications of degr.  $(n/2 - 1)$ -polynomials**

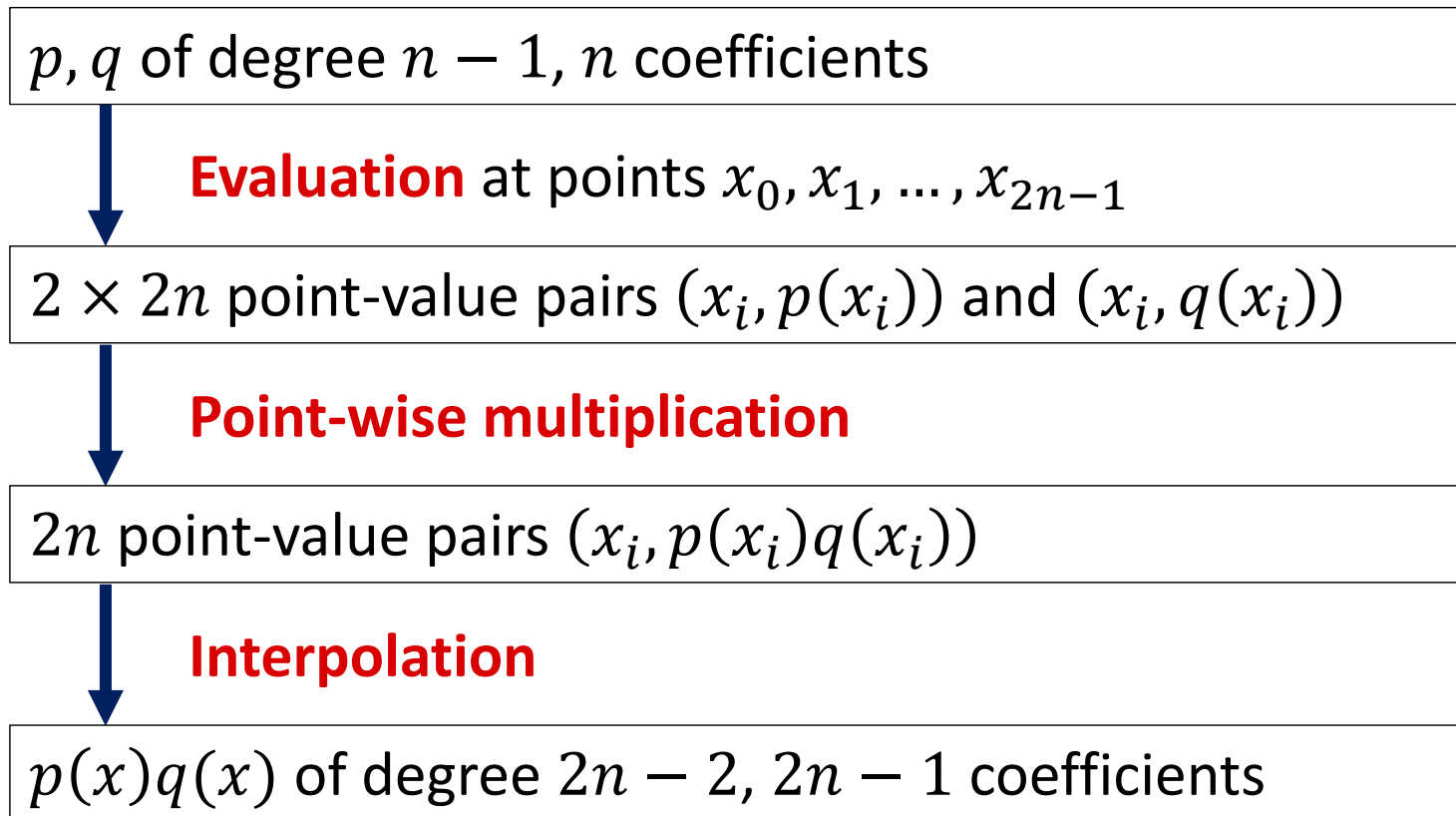
$$T(n) = 3T(n/2) + O(n)$$

- Gives:  $T(n) = O(n^{1.59})$  (see exercises)

# Faster Polynomial Multiplication?

Multiplication is fast when using the **point-value representation**

**Idea** to compute  $p(x) \cdot q(x)$  (for polynomials of degree  $< n$ ):



# Point-Value Representation of $p, q$

- Select points  $x_0, x_1, \dots, x_{N-1}$  to evaluate  $p$  and  $q$  in a clever way

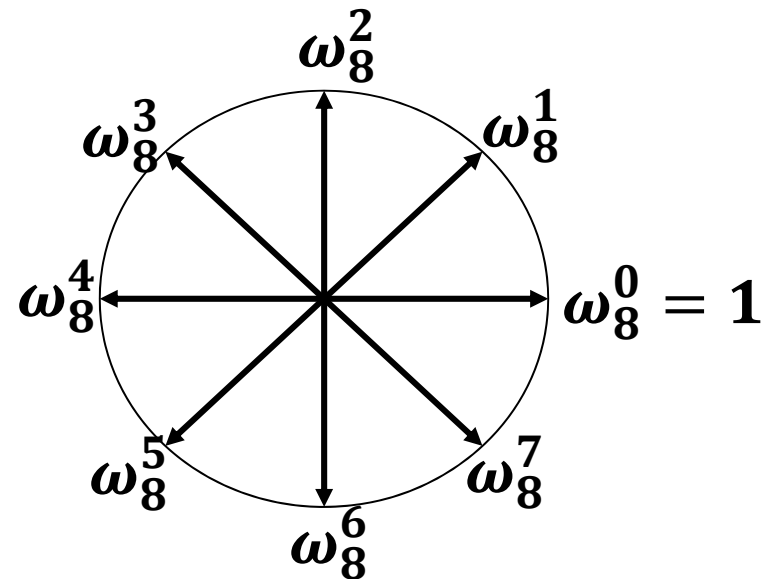
Consider the  $N$  powers of the principle  $N$ th root of unity:

**Principle root of unity:**  $\omega_N = e^{2\pi i/N}$

$$(i = \sqrt{-1}, \quad e^{2\pi i} = 1)$$

**Powers of  $\omega_n$  (roots of unity):**

$$1 = \omega_N^0, \omega_N^1, \dots, \omega_N^{N-1}$$



Note:  $\omega_N^k = e^{2\pi i k/N} = \cos \frac{2\pi k}{N} + i \cdot \sin \frac{2\pi k}{N}$



# Discrete Fourier Transform

---

- The values  $p(\omega_N^i)$  for  $i = 0, \dots, N - 1$  uniquely define a polynomial  $p$  of degree  $< N$ .

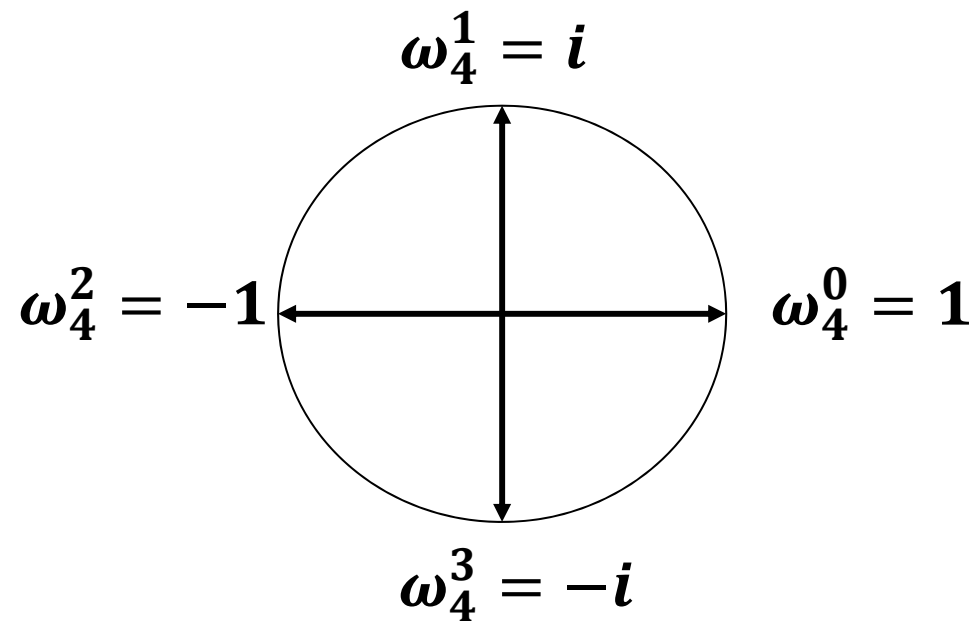
## Discrete Fourier Transform (DFT):

- Assume  $a = (a_0, \dots, a_{N-1})$  is the coefficient vector of poly.  $p$   
$$(p(x) = a_{N-1}x^{N-1} + \dots + a_1x + a_0)$$

$$\text{DFT}_N(a) := \left( p(\omega_N^0), p(\omega_N^1), \dots, p(\omega_N^{N-1}) \right)$$

# Example

- Consider polynomial  $p(x) = 3x^3 - 15x^2 + 18x$
- Choose  $N = 4$
- Roots of unity:



# Example

- Consider polynomial  $p(x) = 3x^3 - 15x^2 + 18x$
- $N = 4$ , roots of unity:  $\omega_4^0 = 1, \omega_4^1 = i, \omega_4^2 = -1, \omega_4^3 = -i$
- Evaluate  $p(x)$  at  $\omega_4^k$ :

$$\left(\omega_4^0, p(\omega_4^0)\right) = (1, p(1)) = (1, 6)$$

$$\left(\omega_4^1, p(\omega_4^1)\right) = (i, p(i)) = (i, 15 + 15i)$$

$$\left(\omega_4^2, p(\omega_4^2)\right) = (-1, p(-1)) = (-1, -36)$$

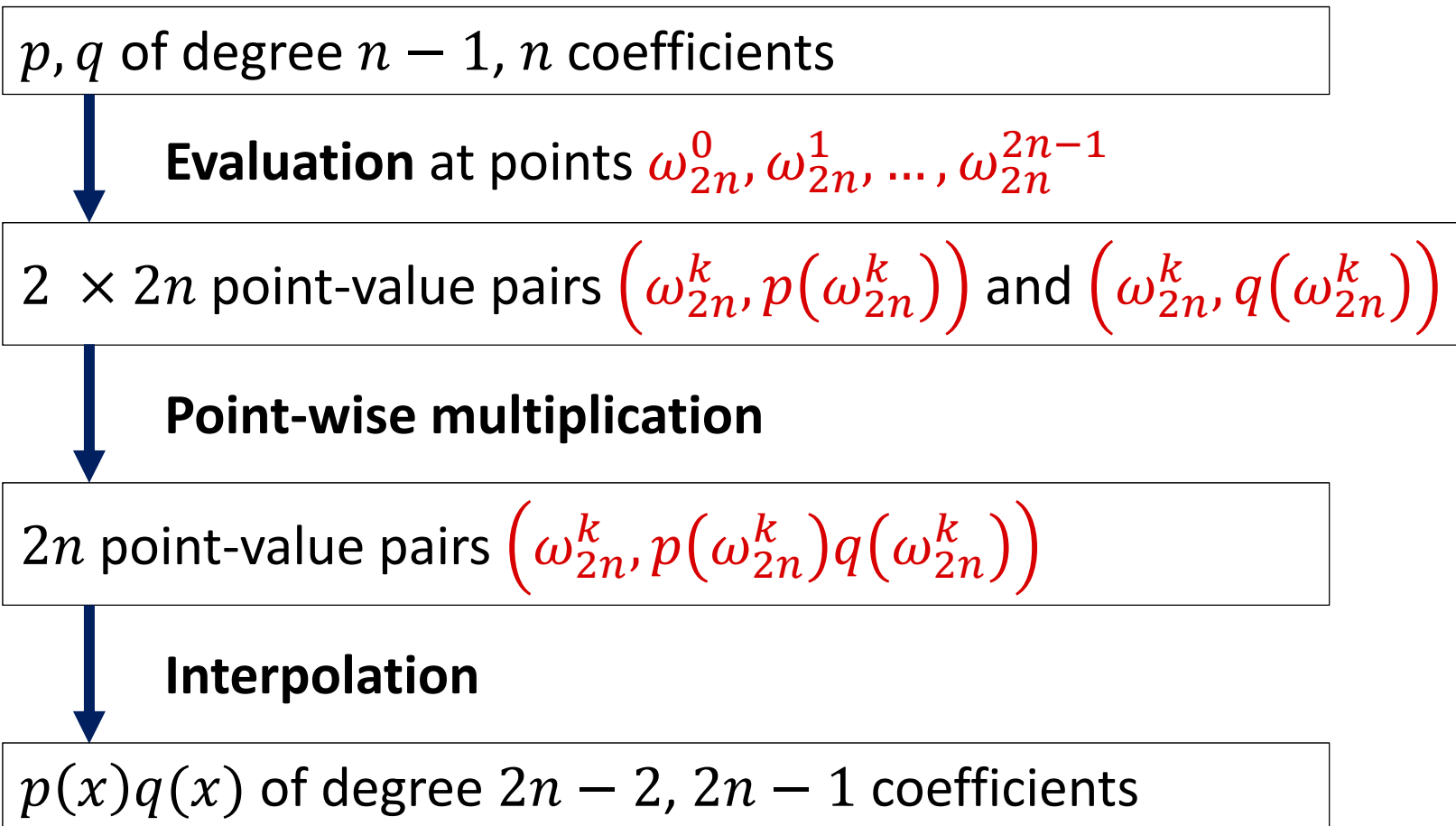
$$\left(\omega_4^3, p(\omega_4^3)\right) = (-i, p(-i)) = (-i, 15 - 15i)$$

- For  $a = (3, -15, 18, 0)$ :

$$\mathbf{DFT}_4(a) = (6, 15 + 15i, -36, 15 - 15i)$$

# Faster Polynomial Multiplication?

Idea to compute  $p(x) \cdot q(x)$  (for polynomials of degree  $< n$ ):



# Properties of the Roots of Unity

---

- **Cancellation Lemma:**

For all integers  $n > 0$ ,  $k \geq 0$ , and  $d > 0$ , we have:

$$\omega_{dn}^{dk} = \omega_n^k, \quad \omega_n^{k+n} = \omega_n^k$$

- **Proof:**

# Divide-and-Conquer Approach

---

- Divide  $p(x)$  of degree  $N - 1$  ( $N$  is even) into 2 polynomials of degree  $N/2 - 1$  differently than in Karatsuba's algorithm
- $p_0(x) = a_0 + a_2x + a_4x^2 + \dots + a_{N-2}x^{N/2-1}$  (even coeff.)  
 $p_1(x) = a_1 + a_3x + a_5x^2 + \dots + a_{N-1}x^{N/2-1}$  (odd coeff.)

# Discrete Fourier Transform

Evaluation for  $k = 0, \dots, N - 1$ :

$$\begin{aligned}
 p(\omega_N^k) &= p_0((\omega_N^k)^2) + \omega_N^k \cdot p_1((\omega_N^k)^2) \\
 &= \begin{cases} p_0(\omega_{N/2}^k) + \omega_N^k \cdot p_1(\omega_{N/2}^k) & \text{if } k < N/2 \\ p_0(\omega_{N/2}^{k-N/2}) + \omega_N^k \cdot p_1(\omega_{N/2}^{k-N/2}) & \text{if } k \geq N/2 \end{cases}
 \end{aligned}$$

For the coefficient vector  $a$  of  $p(x)$ :

$$\begin{aligned}
 \text{DFT}_N(a) &= \left( p_0(\omega_{N/2}^0), \dots, p_0(\omega_{N/2}^0), p_0(\omega_{N/2}^0), \dots, p_0(\omega_{N/2}^{N/2-1}) \right) \\
 &\quad + \left( \omega_N^0 p_0(\omega_{N/2}^0), \dots, \omega_N^{N/2-1} p_0(\omega_{N/2}^0), \omega_N^{N/2} p_0(\omega_{N/2}^0), \dots, \omega_N^{N-1} p_0(\omega_{N/2}^{N/2-1}) \right)
 \end{aligned}$$

# Example

For the coefficient vector  $a$  of  $p(x)$ :

$$\begin{aligned} \text{DFT}_N(a) = & \left( p_0(\omega_{N/2}^0), \dots, p_0(\omega_{N/2}^0), p_0(\omega_{N/2}^0), \dots, p_0(\omega_{N/2}^{N/2-1}) \right) \\ & + \left( \omega_N^0 p_0(\omega_{N/2}^0), \dots, \omega_N^{N/2-1} p_0(\omega_{N/2}^0), \omega_N^{N/2} p_0(\omega_{N/2}^0), \dots, \omega_N^{N-1} p_0(\omega_{N/2}^{N/2-1}) \right) \end{aligned}$$

$N = 4$ :

$$\begin{aligned} p(\omega_4^0) &= p_0(\omega_2^0) + \omega_4^0 p_1(\omega_2^0) \\ p(\omega_4^1) &= p_0(\omega_2^1) + \omega_4^1 p_1(\omega_2^1) \\ p(\omega_4^2) &= p_0(\omega_2^0) + \omega_4^2 p_1(\omega_2^0) \\ p(\omega_4^3) &= p_0(\omega_2^1) + \omega_4^3 p_1(\omega_2^1) \end{aligned}$$

Need:  $(p_0(\omega_2^0), p_0(\omega_2^1))$  and  $(p_1(\omega_2^0), p_1(\omega_2^1))$

(DFTs of coefficient vectors of  $p_0$  and  $p_1$ )



# Recursive Structure

For simplicity, we **abuse notation** in the following:

- Poly.  $p(x) = a_{N-1}x^{N-1} + \dots + a_0$  with coefficient vector  $a$

Let  $\text{DFT}_N(p) := \text{DFT}_N(a)$

## Recursive structure:

- For  $N = 4$ :

$$\begin{aligned} (\text{DFT}_4(p))_k &= p(\omega_4^k) \\ &= (\text{DFT}_2(p_0))_{k \bmod 4} + \omega_4^k \cdot (\text{DFT}_2(p_1))_{k \bmod 4} \end{aligned}$$

- General  $N$  (assume  $N$  is even):

$$\begin{aligned} (\text{DFT}_N(p))_k &= p(\omega_N^k) \\ &= (\text{DFT}_{N/2}(p_0))_{k \bmod N} + \omega_N^k \cdot (\text{DFT}_{N/2}(p_1))_{k \bmod N} \end{aligned}$$

# Computation of $DFT_N$

- Divide-and-conquer algorithm for  $DFT_N(p)$ :

## 1. Divide

$N \leq 1$ :  $DFT_1(p) = a_0$

$N > 1$ : Divide  $p$  into  $p_0$  (even coeff.) and  $p_1$  (odd coeff.).

## 2. Conquer

Solve  $DFT_{N/2}(p_0)$  and  $DFT_{N/2}(p_1)$  recursively

## 3. Combine

Compute  $DFT_N(p)$  based on  $DFT_{N/2}(p_0)$  and  $DFT_{N/2}(p_1)$

# Analysis

---

- $T(N)$ : max. time to compute  $\text{DFT}_N(p)$ :

$$T(N) = 2T(N/2) + O(N), \quad T(1) = O(1)$$

- As for mergesort, comparing orders, closest pair of points:

$$T(N) = O(N \cdot \log N)$$

# Small Improvement

Polynomial  $p$  of degree  $N - 1$ :

$$\begin{aligned}
 p(\omega_N^k) &= \begin{cases} p_0(\omega_{N/2}^k) + \omega_N^k \cdot p_1(\omega_{N/2}^k) & \text{if } k < N/2 \\ p_0(\omega_{N/2}^{k-N/2}) + \omega_N^k \cdot p_1(\omega_{N/2}^{k-N/2}) & \text{if } k \geq N/2 \end{cases} \\
 &= \begin{cases} p_0(\omega_{N/2}^k) + \omega_N^k \cdot p_1(\omega_{N/2}^k) & \text{if } k < N/2 \\ p_0(\omega_{N/2}^{k-N/2}) - \omega_N^{k-N/2} \cdot p_1(\omega_{N/2}^{k-N/2}) & \text{if } k \geq N/2 \end{cases}
 \end{aligned}$$

Need to compute  $p_0(\omega_{N/2}^k)$  and  $\omega_N^k \cdot p_1(\omega_{N/2}^k)$  for  $0 \leq k < N/2$ .

# Example

---



$$p(\omega_4^0) = p_0(\omega_2^0) + \omega_4^0 \cdot p_1(\omega_2^0)$$

$$p(\omega_4^1) = p_0(\omega_2^1) + \omega_4^1 \cdot p_1(\omega_2^0)$$

$$p(\omega_4^2) = p_0(\omega_2^0) - \omega_4^0 \cdot p_1(\omega_2^0)$$

$$p(\omega_4^3) = p_0(\omega_2^1) - \omega_4^1 \cdot p_1(\omega_2^1)$$

# Fast Fourier Transform (FFT) Algorithm



## Algorithm FFT(a)

- Input: Array  $a$  of length  $N$ , where  $N$  is a power of 2
- Output:  $\text{DFT}_N(a)$

**if**  $n = 1$  **then return**  $a_0$ ; //  $a = [a_0]$

$d^{[0]} := \text{FFT}([a_0, a_2, \dots, a_{N-2}]);$

$d^{[1]} := \text{FFT}([a_1, a_3, \dots, a_{N-1}]);$

$\omega_N := e^{2\pi i/N}$ ;  $\omega := 1$ ;

**for**  $k = 0$  **to**  $N/2 - 1$  **do** //  $\omega = \omega_N^k$

$x := \omega \cdot d_k^{[1]}$ ;

$d_k := d_k^{[0]} + x$ ;  $d_{k+N/2} := d_k^{[0]} - x$ ;

$\omega := \omega \cdot \omega_N$

**end**;

**return**  $d = [d_0, d_1, \dots, d_{N-1}]$ ;

# Example

---



- $p(x) = 3x^3 - 15x^2 + 18x + 0, a = [0, 18, -15, 3]$

# Faster Polynomial Multiplication?

Idea to compute  $p(x) \cdot q(x)$  (for polynomials of degree  $< n$ ):

$p, q$  of degree  $n - 1$ ,  $n$  coefficients

**Evaluation** at  $\omega_{2n}^0, \omega_{2n}^1, \dots, \omega_{2n}^{2n-1}$  using **FFT**

$2 \times 2n$  point-value pairs  $(\omega_{2n}^k, p(\omega_{2n}^k))$  and  $(\omega_{2n}^k, q(\omega_{2n}^k))$

**Point-wise multiplication**

$2n$  point-value pairs  $(\omega_{2n}^k, p(\omega_{2n}^k)q(\omega_{2n}^k))$

**Interpolation**

$p(x)q(x)$  of degree  $2n - 2$ ,  $2n - 1$  coefficients