

ALGORITHMS AND DATA STRUCTURES · LECTURE 5

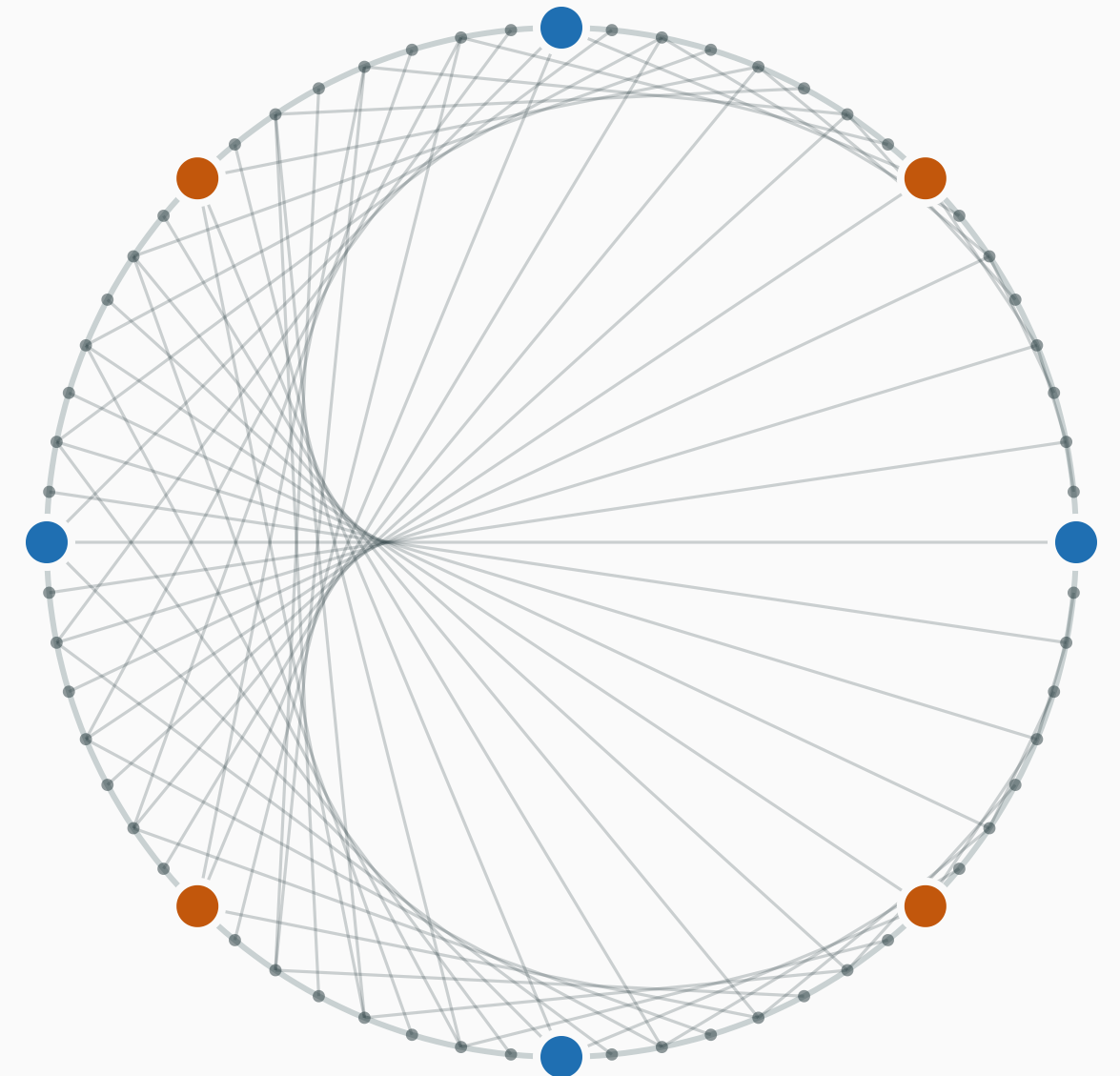
Multiplying polynomials

with the Fast Fourier Transform

Heng Guo

School of Informatics, University of Edinburgh

Monday 5 October 2026 · 16:10–17:00



Multiply $A(x) = 2 + x$ and $B(x) = 1 + 2x$.

Every term of A times every term of B :

$$A(x) B(x) = 2 + 4x + x + 2x^2$$

Add up equal powers of x :

$$A(x) B(x) = 2 + 5x + 2x^2$$

Only the coefficients matter:

$a = (2, 1)$ and $b = (1, 2)$ give $c = (2, 5, 2)$.

$$c_1 = a_0 b_1 + a_1 b_0 = 2 \cdot 2 + 1 \cdot 1 = 5$$

	1	2x
2	2	4x
x	x	2x ²

In general, the coefficient of x^k is the sum of all products $a_i b_j$ with $i + j = k$.

Input: the coefficients of $A(x) = \sum_{j=0}^{n-1} a_j x^j$ and $B(x) = \sum_{j=0}^{n-1} b_j x^j$.

Output: the coefficients of $C(x) = A(x) B(x)$, the **convolution** of the two coefficient vectors:

$$c_k = \sum_{j=0}^k a_j b_{k-j} \quad (0 \leq k \leq 2n-2).$$

Here $a_j = b_j = 0$ for $j \geq n$.

	b_0	b_1	b_2	b_3			
a_0	$a_0 b_0$	$a_0 b_1$	$a_0 b_2$	$a_0 b_3$			
a_1	$a_1 b_0$	$a_1 b_1$	$a_1 b_2$	$a_1 b_3$			
a_2	$a_2 b_0$	$a_2 b_1$	$a_2 b_2$	$a_2 b_3$			
a_3	$a_3 b_0$	$a_3 b_1$	$a_3 b_2$	$a_3 b_3$			
	c_0	c_1	c_2	c_3	c_4	c_5	c_6

$c_3 = a_0 b_3 + a_1 b_2 + a_2 b_1 + a_3 b_0$

Schoolbook method: all n^2 products, $\Theta(n^2)$ time.

🔍 *Can we do better?*

TODAY the **Fast Fourier Transform**: $O(n \log n)$ time

Coefficients: $(a_0, a_1, \dots, a_{n-1})$.

Values at n distinct points $x_0, \dots, x_{n-1}$:
 $(A(x_0), A(x_1), \dots, A(x_{n-1}))$.

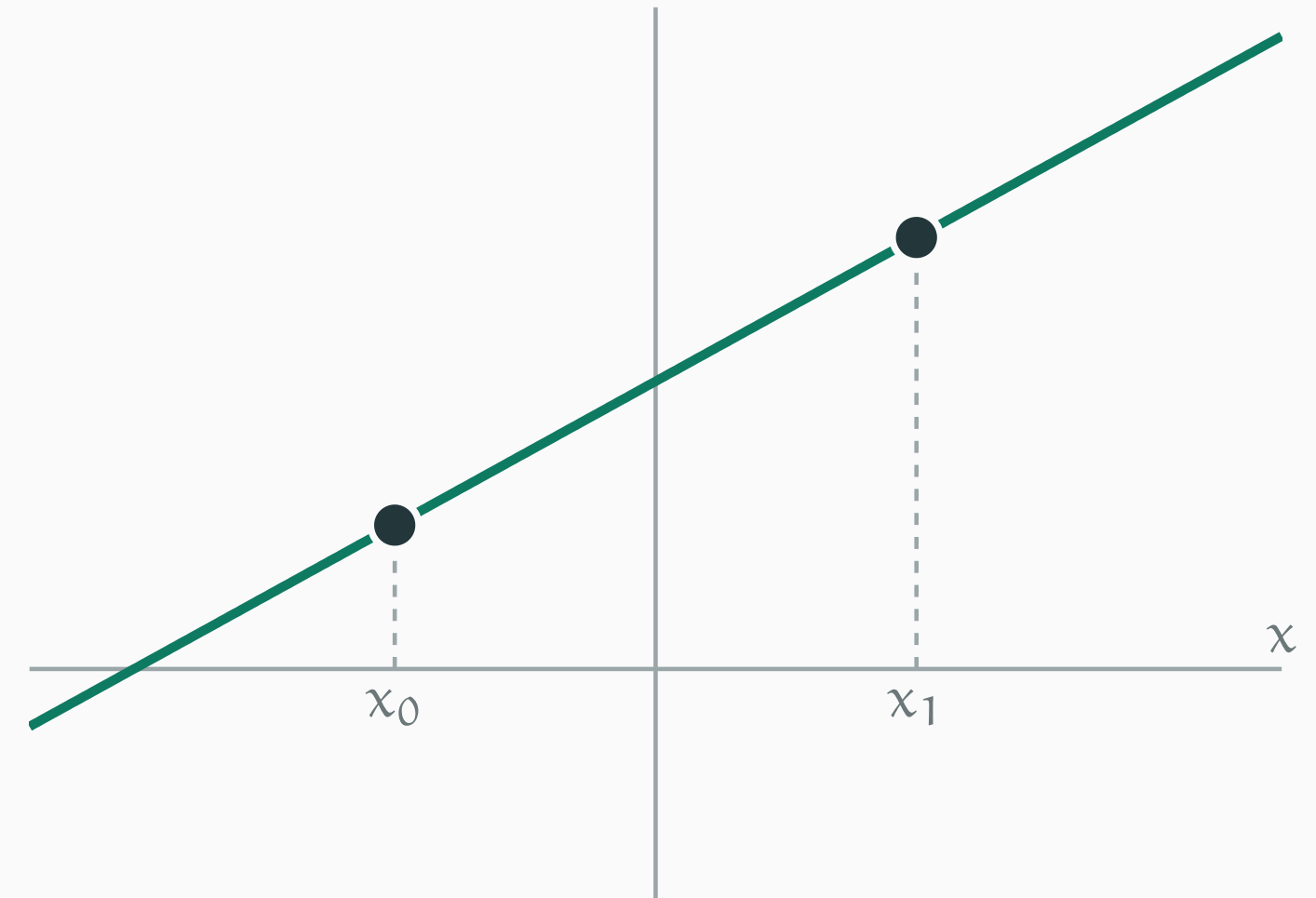
FACT. For distinct $x_0, \dots, x_{n-1}$ and any $y_0, \dots, y_{n-1}$, there is **exactly one** polynomial A of degree $\leq n-1$ with $A(x_j) = y_j$ for all j .

Unique: if A and A' both fit, then $A - A'$ has degree $\leq n-1$ but n roots $x_0, \dots, x_{n-1}$, so $A - A' = 0$.

Exists: Lagrange interpolation gives

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

where the j -th product is 1 at x_j and 0 at every other x_k .



Given A and B at the same points $x_0, \dots, x_{m-1}$:

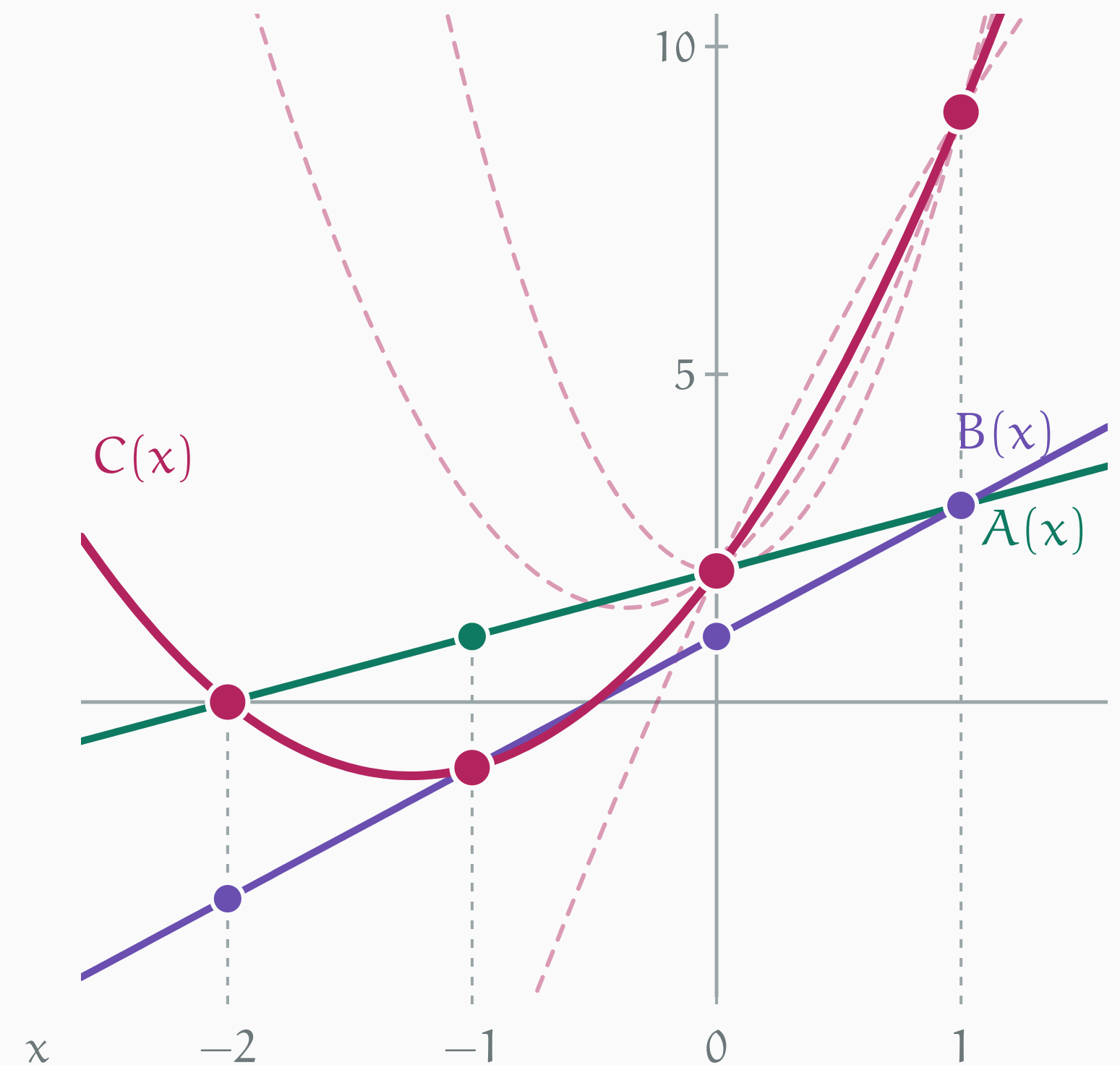
$$C(x_j) = A(x_j) \cdot B(x_j) \quad (m \text{ multiplications}).$$

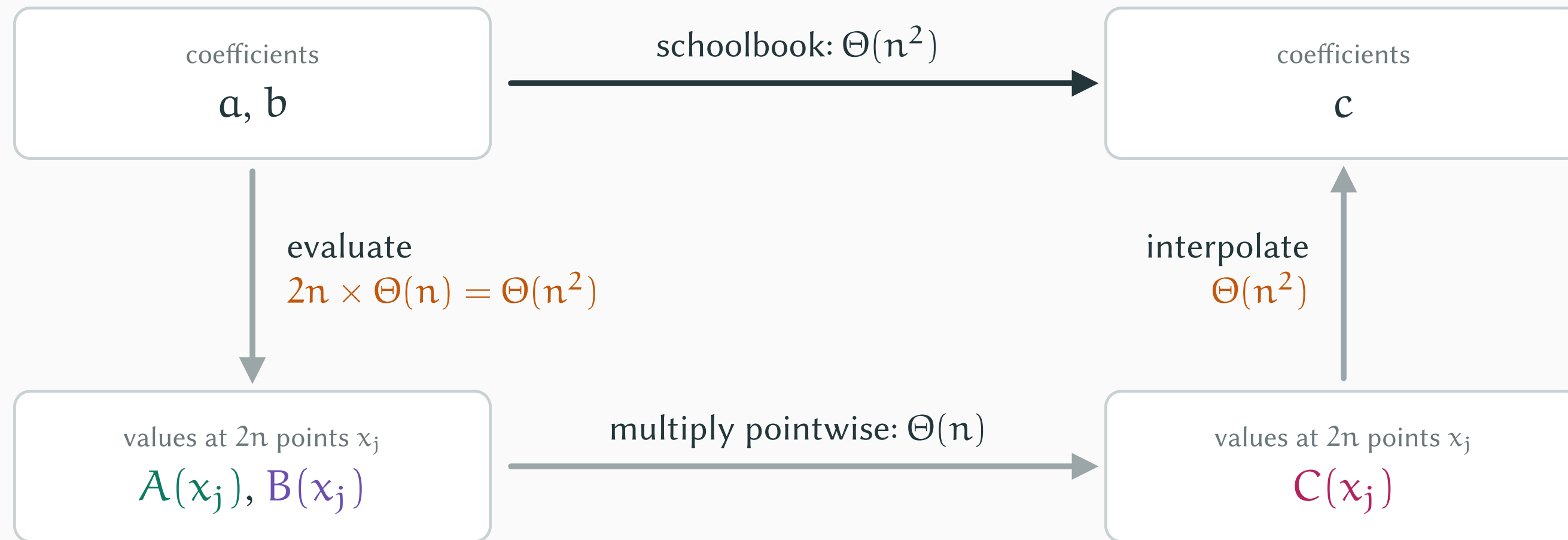
Example: $A(x) = 2 + x$ and $B(x) = 1 + 2x$:

x_j	-2	-1	0	1
$A(x_j)$	0	1	2	3
$B(x_j)$	-3	-1	1	3
$C(x_j)$	0	-1	2	9

The values determine $C(x) = 2 + 5x + 2x^2$.

Catch: C has degree up to $2n - 2$, so we need $2n - 1$ points, not just n .



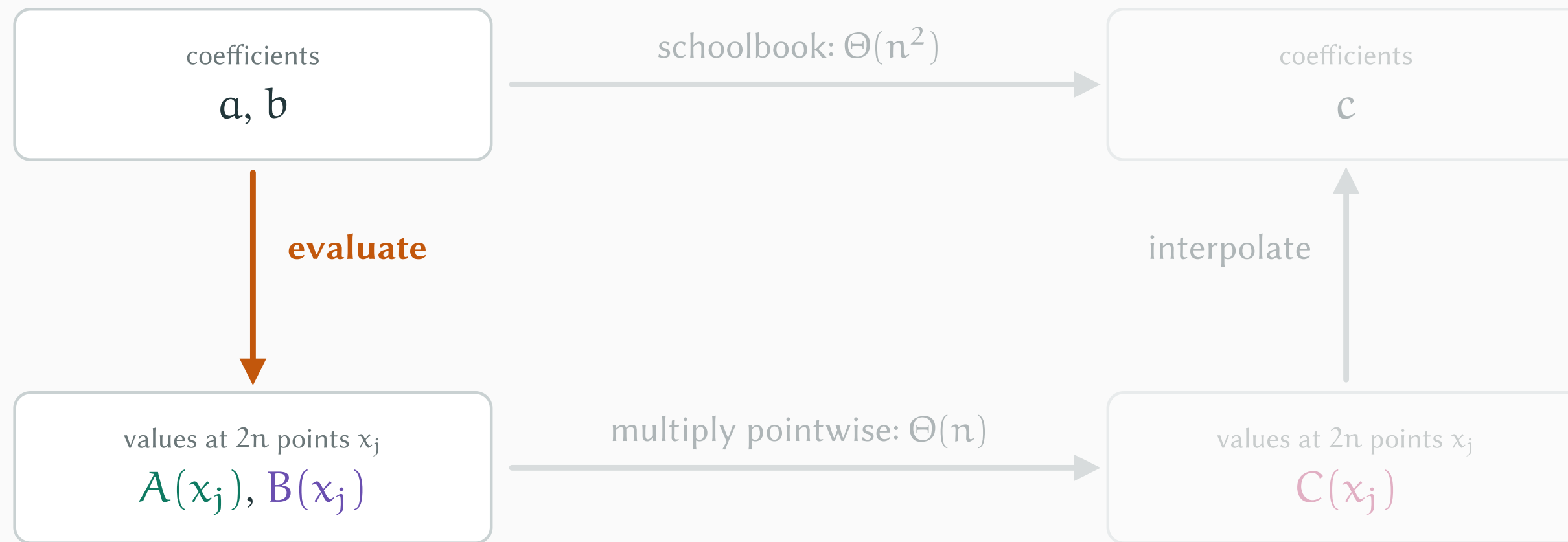


$A(x) = a_0 + x(a_1 + x(a_2 + \dots))$: $\Theta(n)$ time per point.

No gain yet.

But we can choose the points x_j !

Next: points at which evaluating, and interpolating, takes only $O(n \log n)$ time.



Evaluate A (with n coefficients) at n points of our choice
in $O(n \log n)$ time, instead of $\Theta(n^2)$.

We assume n is a power of 2.

For the product, we apply this to A and B padded with zeros to $2n$ coefficients.

? *Which points?*

$$\begin{aligned} A(x) &= a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6 + a_7x^7 \\ &= (a_0 + a_2x^2 + a_4x^4 + a_6x^6) \text{ even powers} \\ &\quad + x(a_1 + a_3x^2 + a_5x^4 + a_7x^6) \text{ odd powers, } x \text{ factored out} \end{aligned}$$

With y in place of x^2 : two polynomials with $n/2 = 4$ coefficients each,

$$A_{\text{even}}(y) = a_0 + a_2y + a_4y^2 + a_6y^3, \quad A_{\text{odd}}(y) = a_1 + a_3y + a_5y^2 + a_7y^3.$$

$$A(x) = A_{\text{even}}(x^2) + x \cdot A_{\text{odd}}(x^2)$$

for every even n , with A_{even} and A_{odd} of $n/2$ coefficients each

$$A(x) = A_{\text{even}}(x^2) + x A_{\text{odd}}(x^2)$$

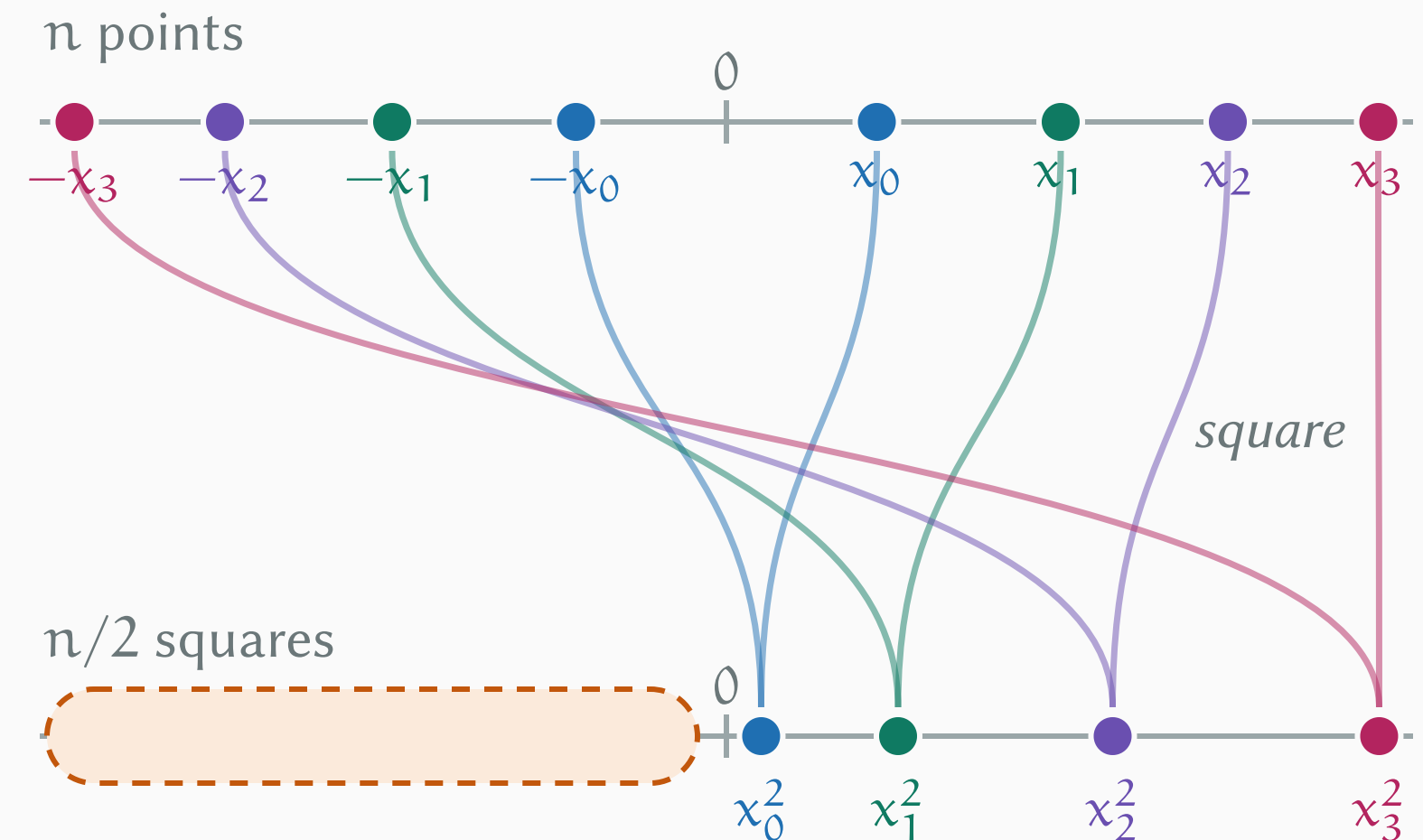
$$A(-x) = A_{\text{even}}(x^2) - x A_{\text{odd}}(x^2)$$

The same two values give $A(x)$ and $A(-x)$.

Choose the n points in pairs $\pm x_0, \dots, \pm x_{n/2-1}$.

Evaluating A at these n points reduces to evaluating A_{even} and A_{odd} at the $n/2$ points $x_0^2, \dots, x_{n/2-1}^2$, plus $O(n)$ work.

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



*no negative squares
so no pairs to recurse on*

Catch: to recurse, the squares x_j^2 must again be $\pm$ pairs. But real squares are ≥ 0 !

We need a set of n points that is made of $\pm$ pairs and whose squares form such a set again, of $n/2$ points, ... down to 1 point.

Work backwards, taking square roots:

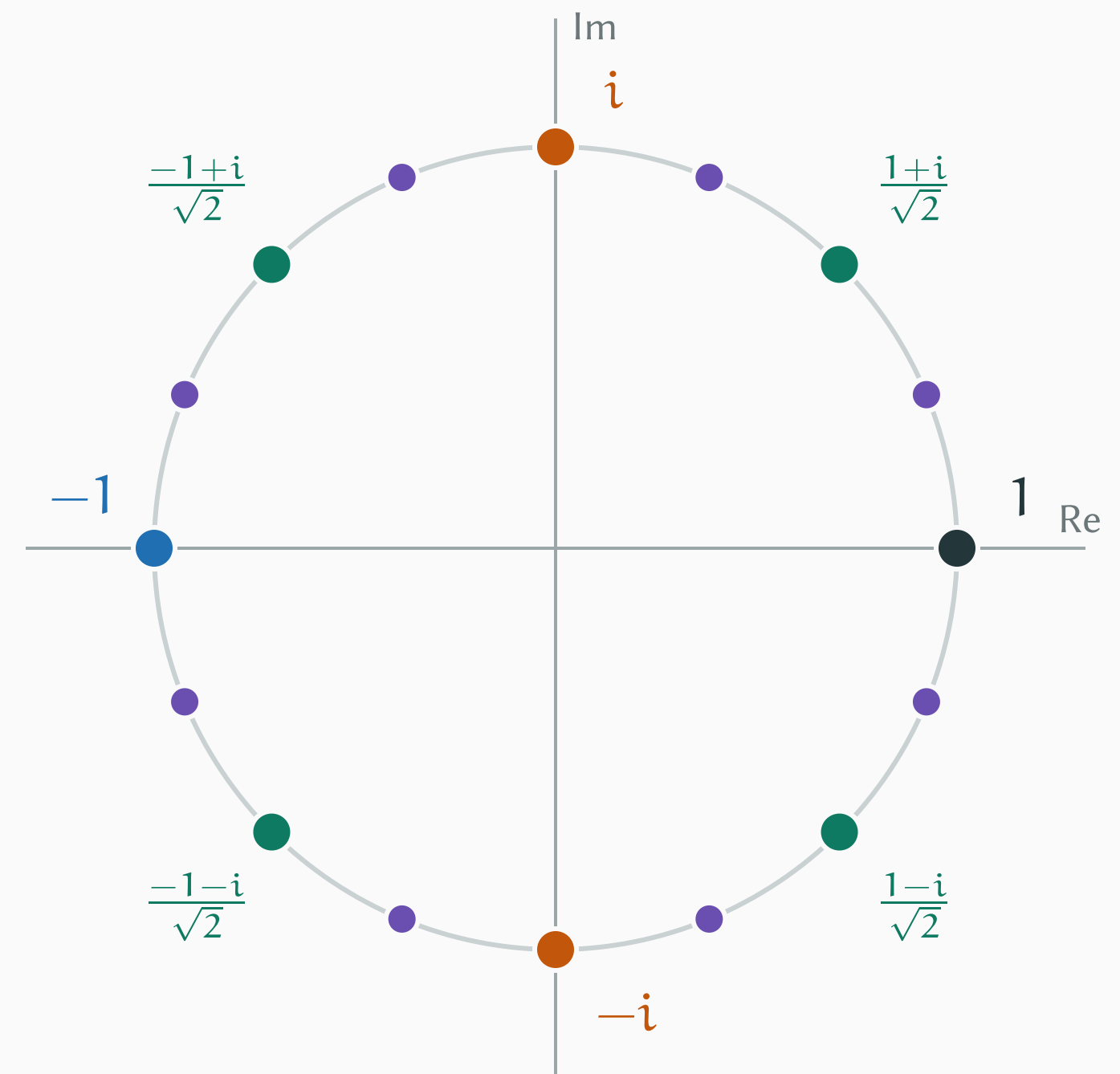
$$n = 1: \quad 1$$

$$n = 2: \quad \pm 1 \text{ (square roots of 1)}$$

$$n = 4: \quad \pm 1, \pm i \text{ (square roots of 1 and } -1\text{)}$$

$$n = 8: \quad \pm 1, \pm i, \pm \frac{1+i}{\sqrt{2}}, \pm \frac{1-i}{\sqrt{2}} \text{ (and so on)}$$

These are the **complex n -th roots of unity**: the n solutions of $z^n = 1$, evenly spaced on the unit circle.



DEFINITION. The **principal n -th root of unity** is

$$\omega_n = e^{2\pi i/n} = \cos \frac{2\pi}{n} + i \sin \frac{2\pi}{n}.$$

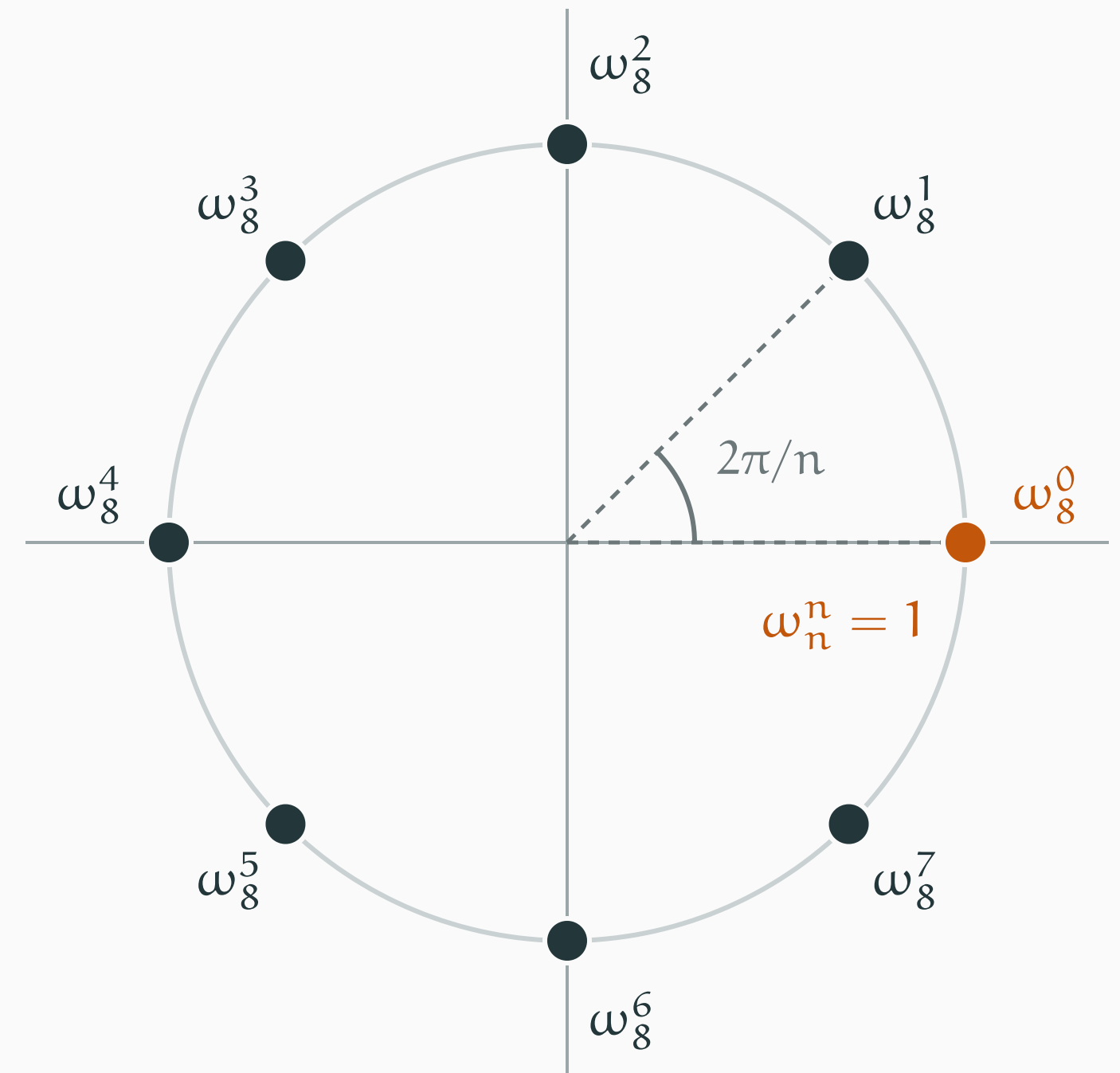
The n -th roots of unity are $\omega_n^0 = 1, \omega_n^1, \dots, \omega_n^{n-1}$.

Recall $e^{i\theta} = \cos \theta + i \sin \theta$, and $e^{i\alpha} e^{i\beta} = e^{i(\alpha+\beta)}$:
multiplying complex numbers on the unit circle adds their angles.

So ω_n^k is the point at angle $2\pi k/n$:
each factor ω_n turns by $2\pi/n$.

$\omega_n^n = e^{2\pi i} = 1$, so exponents count modulo n :

$$\omega_n^{j+n} = \omega_n^j.$$



Let n be even.

PLUS-MINUS PAIRS. $\omega_n^{k+n/2} = -\omega_n^k$.

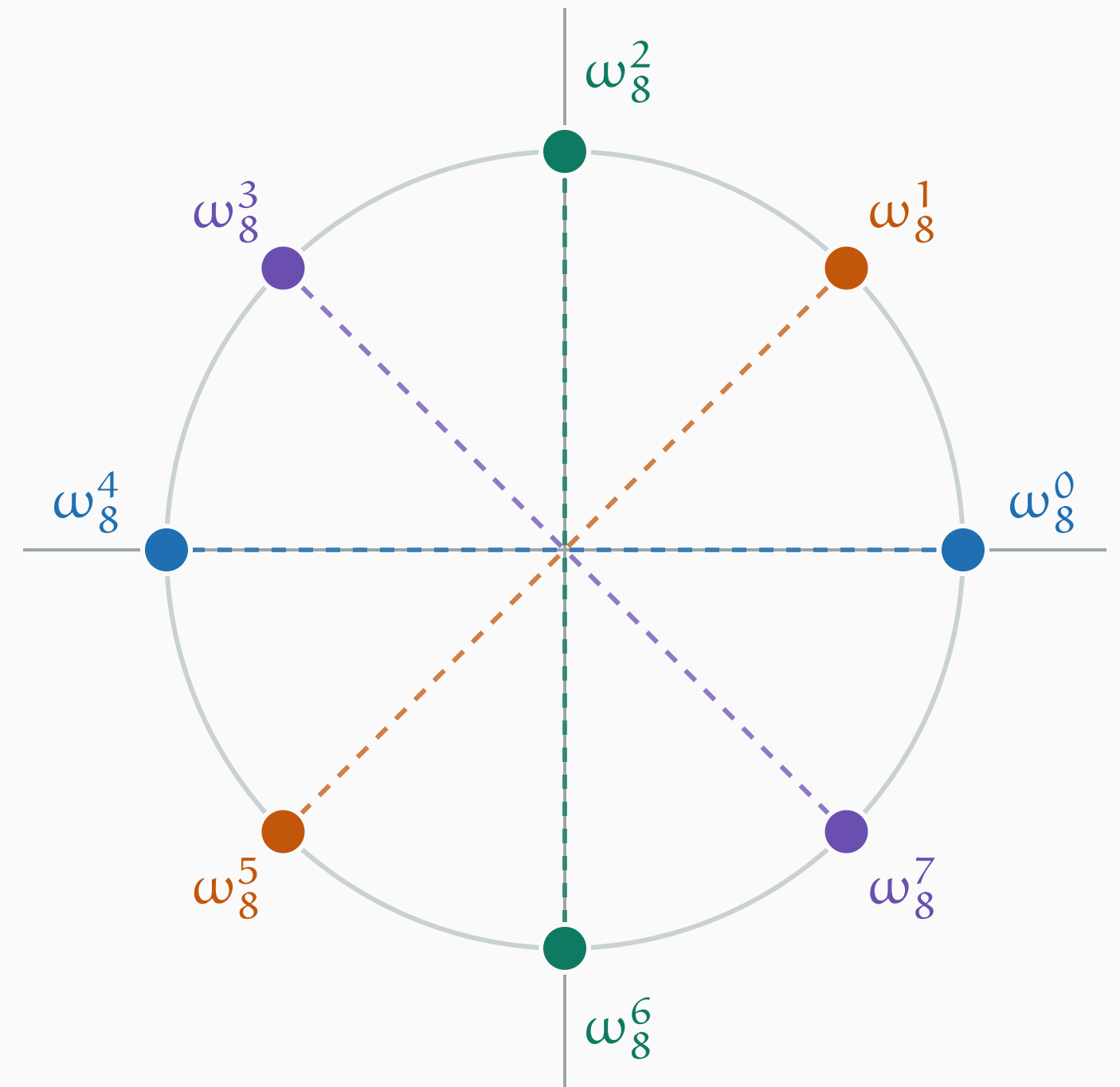
Since $\omega_n^{n/2} = e^{\pi i} = -1$, we get $\omega_n^{k+n/2} = \omega_n^k \cdot \omega_n^{n/2} = -\omega_n^k$.

SQUARES. $(\omega_n^k)^2 = \omega_{n/2}^k$.

$(\omega_n^k)^2 = e^{2\pi i \cdot 2k/n} = e^{2\pi i k/(n/2)} = \omega_{n/2}^k$.

Squaring the n roots of unity of order n gives the $n/2$ roots of order $n/2$, each twice (the *halving lemma*),

and these are again made of $\pm$ pairs, all the way down to 1.



the 8 roots of unity of order 8

Recall that $A(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$ is the given polynomial.

DEFINITION. The **discrete Fourier transform** of $a = (a_0, \dots, a_{n-1})$ is $\text{DFT}_n(a) = (y_0, \dots, y_{n-1})$, where

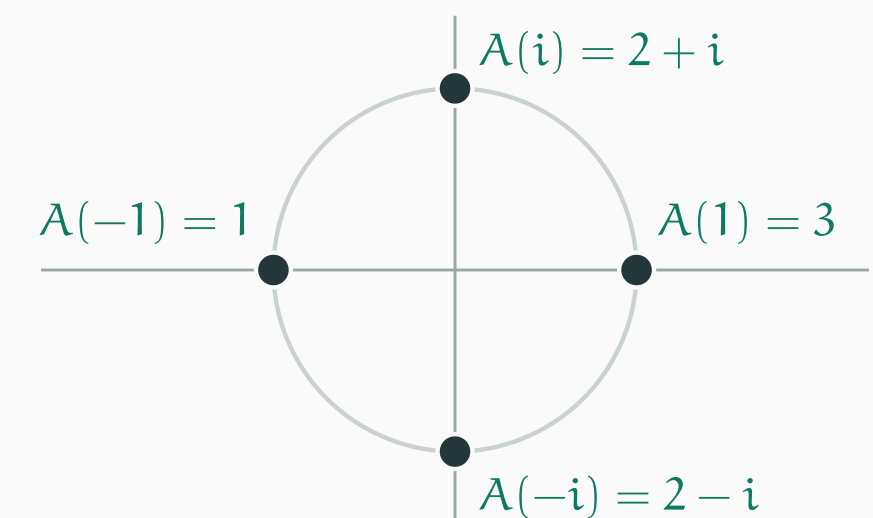
$$y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{kj} \quad (0 \leq k \leq n-1).$$

That is: the values of A at the n -th roots of unity.

The **Fast Fourier Transform** (FFT) computes them in $O(n \log n)$ time, for n a power of 2.

Example: $n = 4$, $\omega_4 = i$, and $A(x) = 2 + x$, so $a = (2, 1, 0, 0)$:

k	0	1	2	3
ω_4^k	1	i	-1	$-i$
$y_k = A(\omega_4^k)$	3	$2 + i$	1	$2 - i$



Split a into $a^{\text{even}} = (a_0, a_2, \dots, a_{n-2})$
and $a^{\text{odd}} = (a_1, a_3, \dots, a_{n-1})$, and let

$$y^{\text{even}} = \text{DFT}_{n/2}(a^{\text{even}}), \quad y^{\text{odd}} = \text{DFT}_{n/2}(a^{\text{odd}}),$$

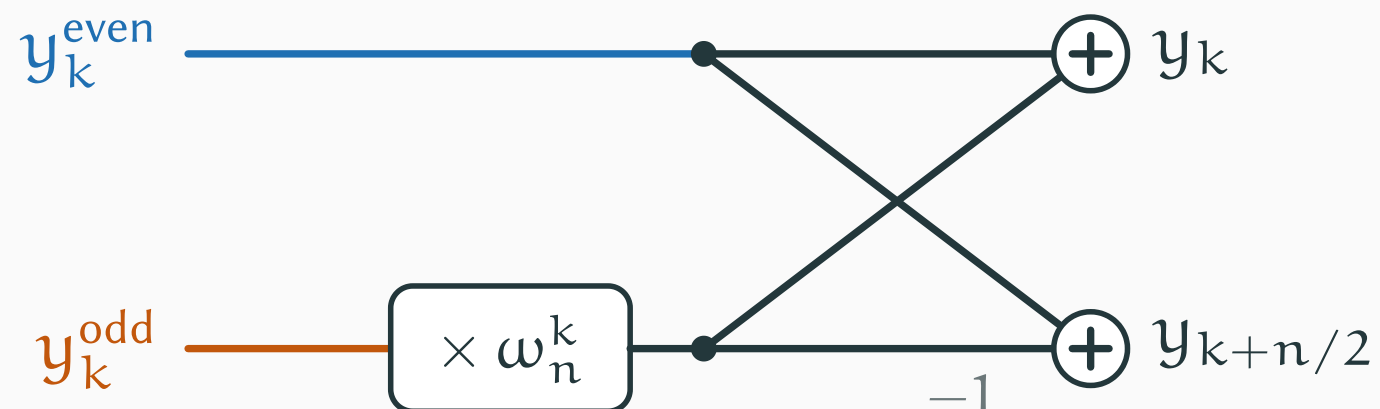
that is, $y_k^{\text{even}} = A_{\text{even}}(\omega_{n/2}^k)$ and $y_k^{\text{odd}} = A_{\text{odd}}(\omega_{n/2}^k)$ for $0 \leq k < n/2$.

$$y_k = A(\omega_n^k) = A_{\text{even}}((\omega_n^k)^2) + \omega_n^k A_{\text{odd}}((\omega_n^k)^2) = y_k^{\text{even}} + \omega_n^k y_k^{\text{odd}}$$

by $A(x) = A_{\text{even}}(x^2) + x A_{\text{odd}}(x^2)$ by $(\omega_n^k)^2 = \omega_{n/2}^k$

$$y_{k+n/2} = A(-\omega_n^k) = A_{\text{even}}((\omega_n^k)^2) - \omega_n^k A_{\text{odd}}((\omega_n^k)^2) = y_k^{\text{even}} - \omega_n^k y_k^{\text{odd}}$$

by $\omega_n^{k+n/2} = -\omega_n^k$



A butterfly: one multiplication and two additions give both y_k and $y_{k+n/2}$. All $n/2$ of them: $O(n)$ time.

```

FFT(a)  where  $a = (a_0, \dots, a_{n-1})$ ,  $n$  a power of 2
1  if  $n = 1$  then return  $a$ 
2   $y^{\text{even}} \leftarrow \text{FFT}((a_0, a_2, \dots, a_{n-2}))$ 
3   $y^{\text{odd}} \leftarrow \text{FFT}((a_1, a_3, \dots, a_{n-1}))$ 
4   $\omega \leftarrow 1$  (invariant:  $\omega = \omega_n^k$ )
5  for  $k = 0$  to  $n/2 - 1$  do
6     $y_k \leftarrow y_k^{\text{even}} + \omega \cdot y_k^{\text{odd}}$ 
7     $y_{k+n/2} \leftarrow y_k^{\text{even}} - \omega \cdot y_k^{\text{odd}}$ 
8     $\omega \leftarrow \omega \cdot \omega_n$ 
9  return  $y = (y_0, \dots, y_{n-1})$ 
    
```

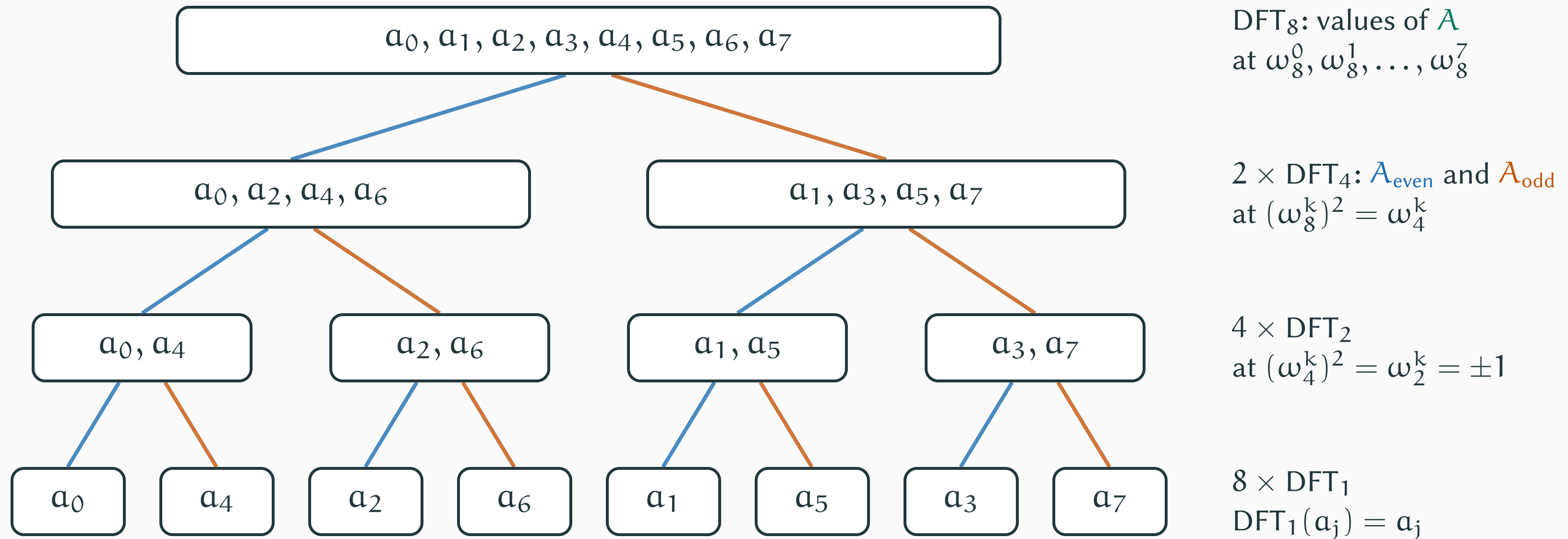
Base case: $\text{DFT}_1(a_0) = a_0 \omega_1^0 = a_0$.

Divide: two calls on $n/2$ coefficients.

Combine: $n/2$ butterflies, $\Theta(n)$ time.

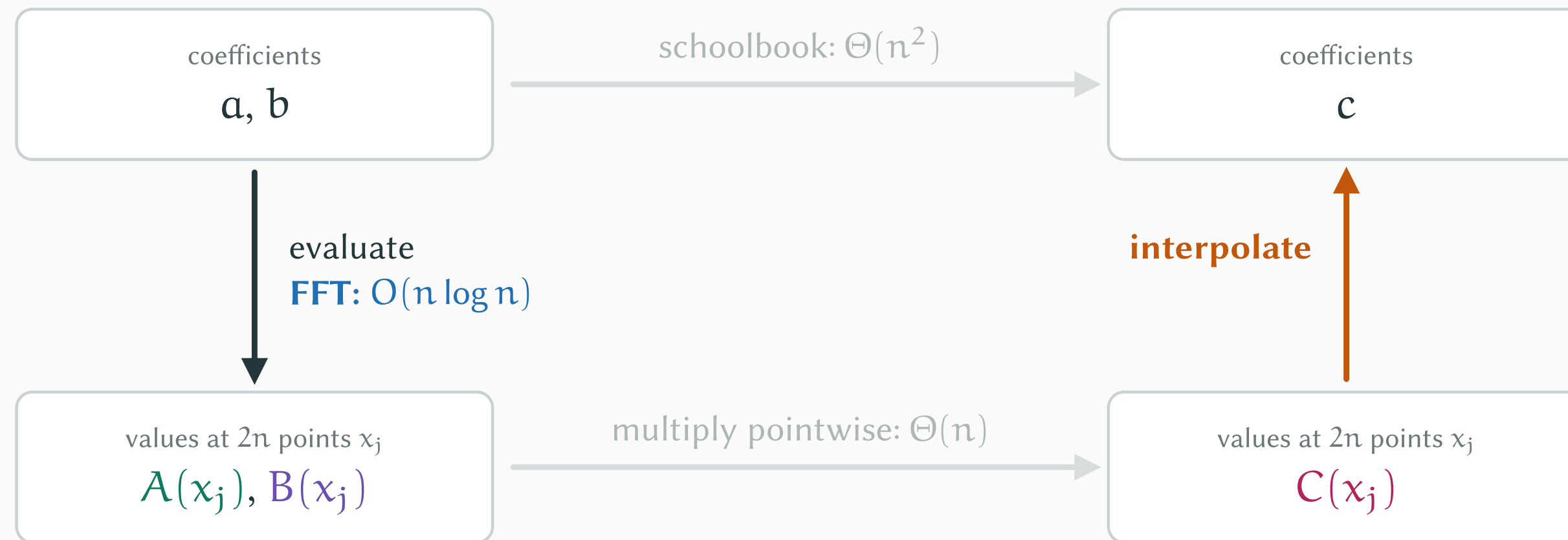
$$T(n) = 2T(n/2) + \Theta(n)$$

so $T(n) = \Theta(n \log n)$.



Then back up: each node combines the values of its two children with butterflies.

At the top, for $0 \leq k \leq 3$: $y_k = y_k^{\text{even}} + \omega_8^k y_k^{\text{odd}}$, $y_{k+4} = y_k^{\text{even}} - \omega_8^k y_k^{\text{odd}}$.



We can evaluate fast at the roots of unity: DFT_n in $O(n \log n)$.

Now the way back: from $y = \text{DFT}_n(a)$ to a .

? *Is interpolation harder than evaluation?*

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

In short $y = V_n a$, where V_n has entries $(V_n)_{kj} = \omega_n^{kj}$.

Interpolation is solving $y = V_n a$ for a .

? *Is V_n invertible?*

Yes: V_n is the **Vandermonde matrix** of the distinct points $\omega_n^0, \dots, \omega_n^{n-1}$:

$\det V_n = \prod_{0 \leq j < k \leq n-1} (\omega_n^k - \omega_n^j) \neq 0$, so $a = V_n^{-1} y$.

? *What is V_n^{-1} ?*

For $n = 4$, with $\omega_4 = i$:

$$V_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

THEOREM. V_n^{-1} has entries $(V_n^{-1})_{jk} = \frac{1}{n} \omega_n^{-jk}$. So if $y = \text{DFT}_n(a)$, then

$$a_j = \frac{1}{n} \sum_{k=0}^{n-1} y_k \omega_n^{-jk} \quad (0 \leq j \leq n-1).$$

This is the DFT of y with ω_n^{-1} in place of ω_n , divided by n .

Inverse FFT: run the FFT with $\omega_n^{-1} = e^{-2\pi i/n}$ instead of ω_n , then divide by n .

It works because ω_n^{-1} has the same two properties:

- $\omega_n^{-(k+n/2)} = -\omega_n^{-k}$,
- $(\omega_n^{-k})^2 = \omega_{n/2}^{-k}$.

Interpolation at the roots of unity also takes $O(n \log n)$ time.

Let $W_{jk} = \frac{1}{n} \omega_n^{-jk}$. We show $WV_n = I$.

Entry (j, j') of WV_n is

$$\frac{1}{n} \sum_{k=0}^{n-1} \omega_n^{-jk} \omega_n^{kj'} = \frac{1}{n} \sum_{k=0}^{n-1} \omega_n^{sk}, \quad s = j' - j.$$

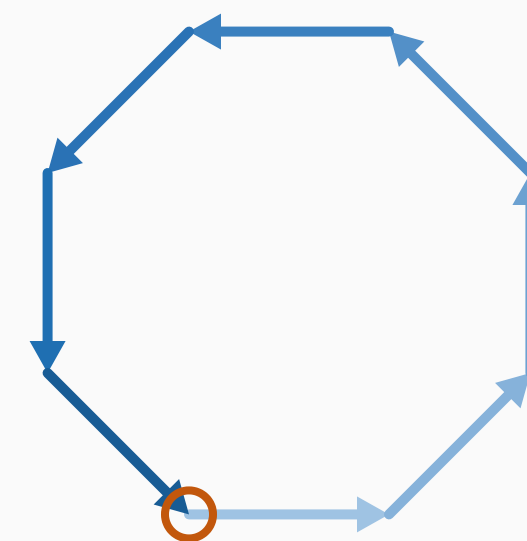
- **If $j = j'$:** $s = 0$, so all n terms are 1, and the entry is 1.
- **If $j \neq j'$:** $0 < |s| < n$, so the entry is 0, by the lemma on the right.

So $W = V_n^{-1}$: interpolating is a DFT with ω_n^{-1} in place of ω_n , divided by n .

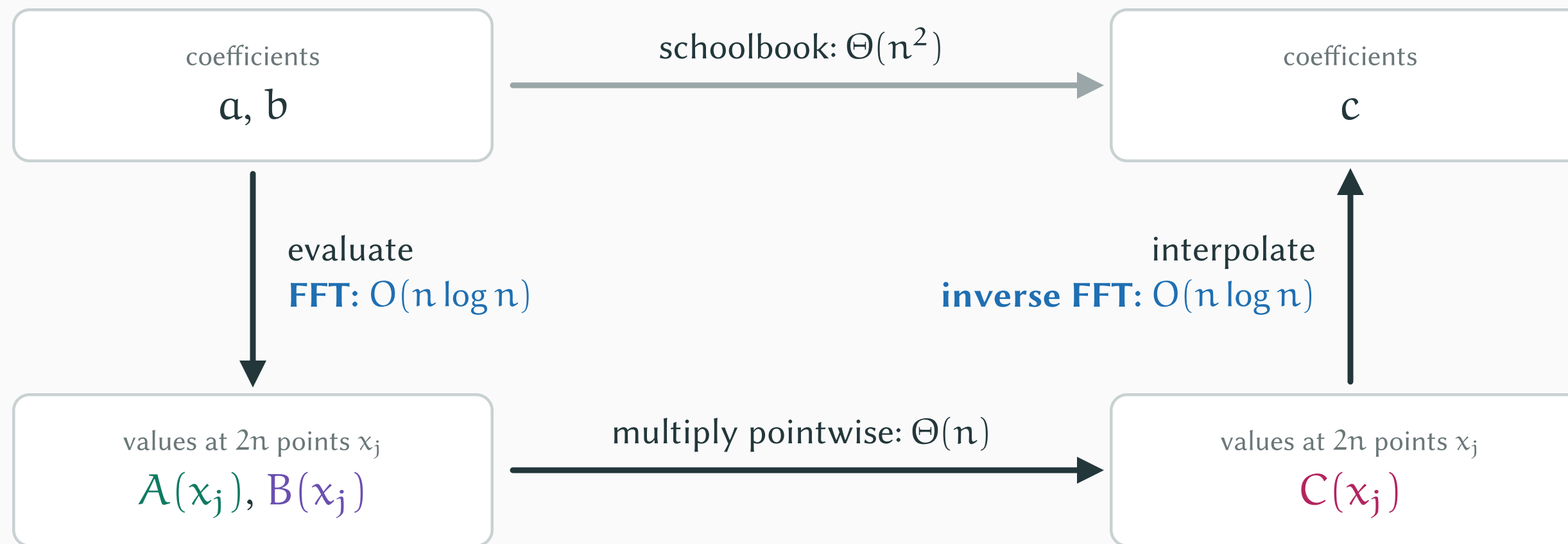
SUMMATION LEMMA. If n does not divide s , then $\sum_{k=0}^{n-1} \omega_n^{sk} = 0$.

Proof. $\omega_n^s \neq 1$, and by the geometric series

$$\sum_{k=0}^{n-1} (\omega_n^s)^k = \frac{(\omega_n^s)^n - 1}{\omega_n^s - 1} = \frac{1 - 1}{\omega_n^s - 1} = 0.$$



$$s = 1 : \quad \sum_{k=0}^7 \omega_8^k = 0$$



- 1 Pad a and b with zeros to length N ,
the smallest power of 2 with $N \geq 2n$. $O(n)$
- 2 $y^A \leftarrow \text{FFT}(a)$ and $y^B \leftarrow \text{FFT}(b)$. $O(n \log n)$
- 3 $y_k^C \leftarrow y_k^A \cdot y_k^B$ for $0 \leq k \leq N - 1$. $O(n)$
- 4 $c \leftarrow \text{inverse FFT}(y^C)$. $O(n \log n)$

THEOREM. Two polynomials with n coefficients each can be multiplied in $O(n \log n)$ time.

WORKED EXAMPLE: $(2 + x)(1 + 2x)$

Multiply $A(x) = 2 + x$ and $B(x) = 1 + 2x$.

Padded to 4 coefficients; here $\omega_4 = i$.

k	0	1	2	3	
coefficients a	2	1	0	0	
coefficients b	1	2	0	0	
$A(\omega_4^k)$	3	$2 + i$	1	$2 - i$	FFT(a)
$B(\omega_4^k)$	3	$1 + 2i$	-1	$1 - 2i$	FFT(b)
$C(\omega_4^k)$	9	$5i$	-1	$-5i$	pointwise product
coefficients c	2	5	2	0	inverse FFT: $\frac{1}{4} (8, 20, 8, 0)$

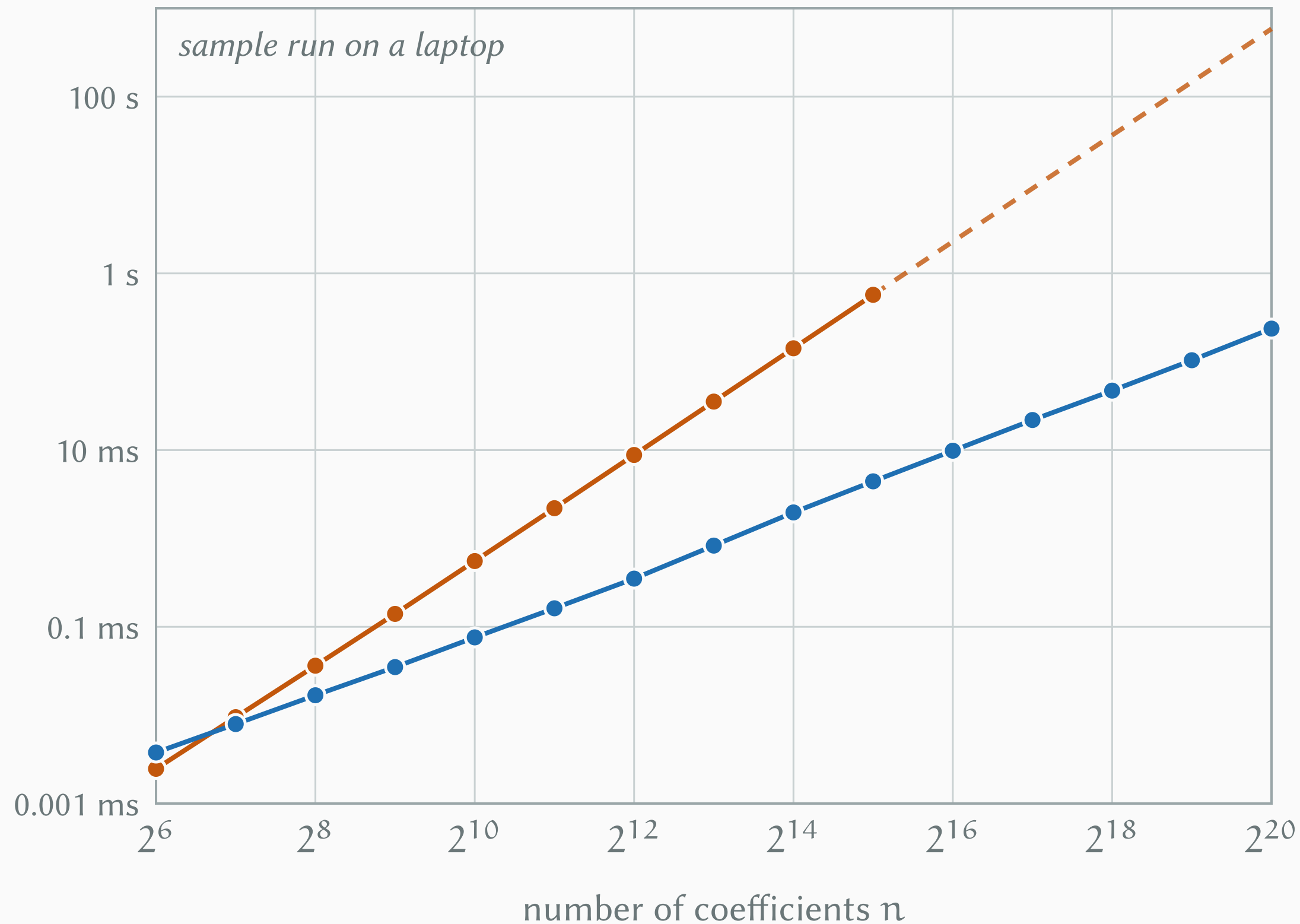
e.g. $(2 + i)(1 + 2i) = 2 + 4i + i + 2i^2 = 5i$

e.g. $c_1 = \frac{1}{4} (9 + 5i(-i) + (-1)(-1) + (-5i)i) = 5$

$C(x) = A(x) B(x) = 2 + 5x + 2x^2.$

How MUCH FASTER?

PUTTING IT TOGETHER



Multiply two random polynomials with n coefficients, in this browser.

— schoolbook, $\Theta(n^2)$

— FFT, $\Theta(n \log n)$

At $n = 2^{20}$:

FFT: 237 ms

Schoolbook: about 10 min
(extrapolated)

- ➊ **Change the representation.** Values at $2n$ points determine the product $A(x) B(x)$, and multiplying values takes $O(n)$ time.
- ➋ **Choose the points cleverly.** At the roots of unity, evaluation splits into two problems of half the size: the points form $\pm$ pairs, and their squares are again roots of unity.
So $T(n) = 2 T(n/2) + \Theta(n) = \Theta(n \log n)$.
- ➌ **Interpolation is evaluation again**, with ω_n^{-1} in place of ω_n and a factor $1/n$.

Two polynomials with n coefficients can be multiplied in $O(n \log n)$ time.

The FFT is everywhere: audio, images and video, telecommunications, numerical analysis, and the fastest algorithms for multiplying huge integers (Schönhage and Strassen 1971; Harvey and van der Hoeven 2019, $O(n \log n)$).

History: Gauss used it around 1805; Cooley and Tukey rediscovered it in 1965.

Reference: CLRS, Chapter 30.