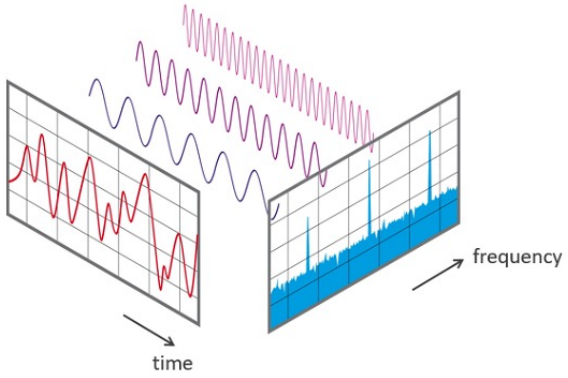
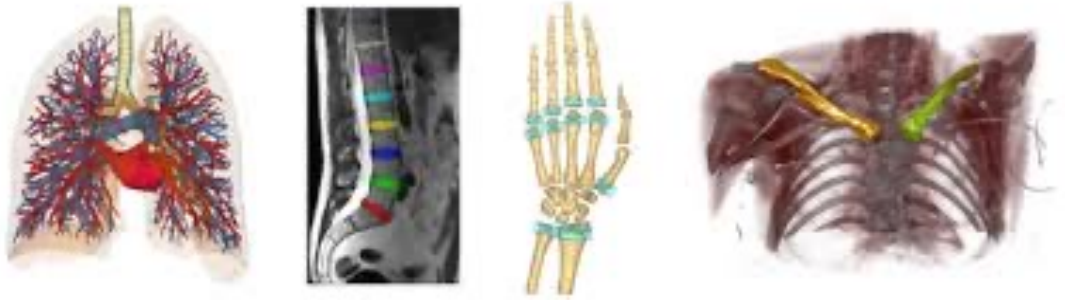
Fast Fourier Transform

Fang-Lue Zhang

Fast Fourier Transform (FFT)

- FFT is a useful algorithm. Invented by signal processing researchers, widely used in many areas.





The FFT is one of the truly great computational developments of this [20th] century. It has changed the face of science and engineering so much that it is not an exaggeration to say that life as we know it would be very different without the FFT. - *Charles van Loan (Cornell University)*

FFT for a simple problem: Polynomial Multiplication

$A(x) = x^2 + 3x + 2, \quad B(x) = 2x^2 + 1$

$C(x) = A(x) \cdot B(x)$

We can just use the coefficients to represent polynomials

$A(x) = 2 + 3x + x^2$. → A=[2, 3, 1]

$B(x) = 1 + 2x^2$ → B = [1, 0, 2]

$C(x) = 2 + 3x + 5x^2 + 6x^3 + 2x^4$ → C = [2, 3, 5, 6, 2]

C[k] is coefficient of k^{th} degree term of polynomial C(x)

The complexity of the multiplication is $O(n^2)$, n is the degree of the polynomials A(x) and B(x).

Polynomial Multiplication

- General Polynomial Multiplication:

$$A(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

$$B(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_n x^n$$

$$C(x) = A(x)B(x) = \sum_{i=0}^{2n} c_i x^i = c_0 + c_1 x + c_2 x^2 + \cdots + c_{2n} x^{2n}$$

We only need to care about the coefficients

Can we do it faster?

Two different ways of polynomial representation:

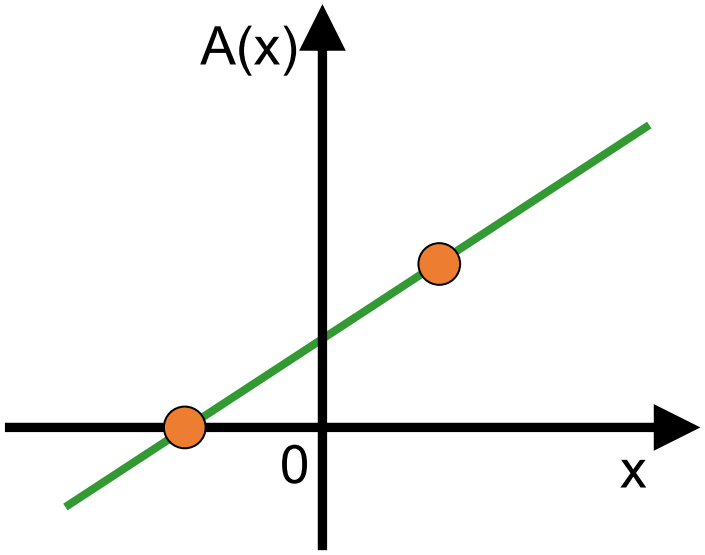
- Coefficient Representation

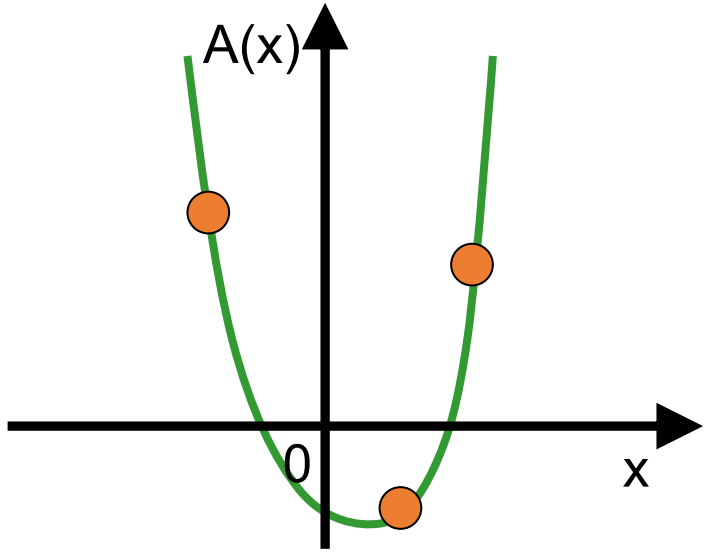
$$A(x) = \sum_{i=0}^{n} a_i x^i$$

- **Point Representation**

  A polynomial of degree n can be uniquely represented by n + 1 points

  E.g. 2 points determine a line; 3 points determine a parabola





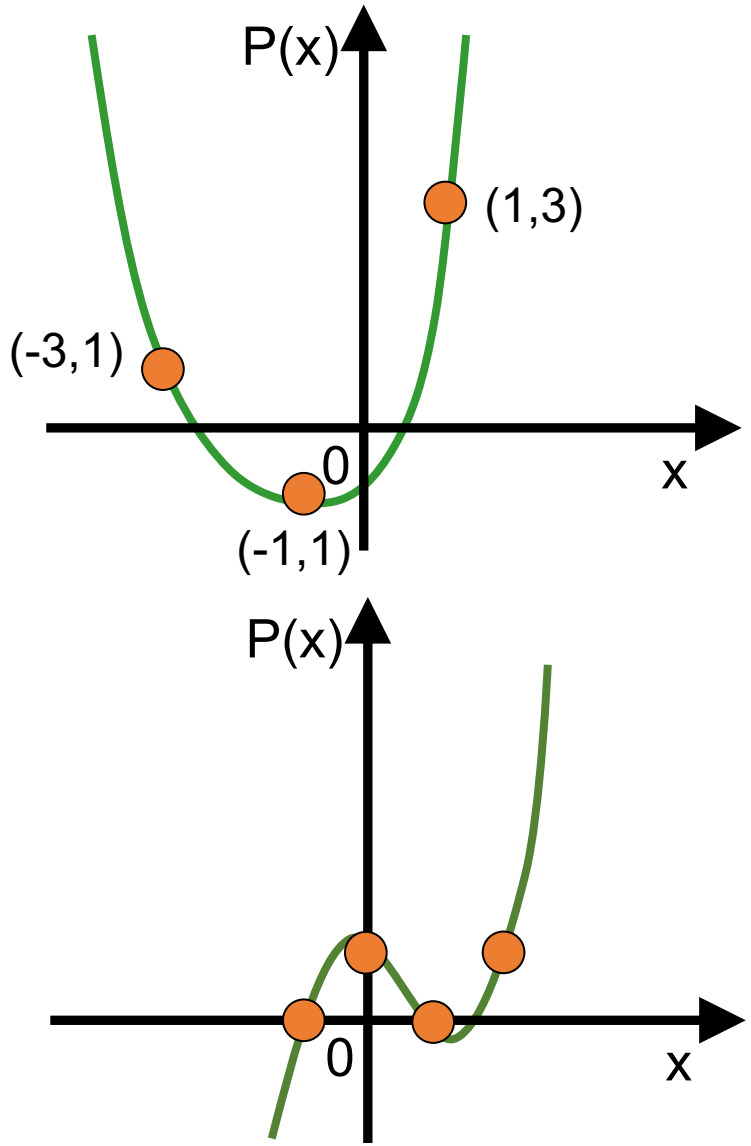
Polynomial Representation by Points

- (d+1) points uniquely define a degree d polynomial
- The UNIQUE quadratic function that goes through (-3,1), (-1,-1) and (1,3) is:

$$P(x) = 0.75x^2 + 2x + 0.25$$

- The UNIQUE cubic function that goes through (-1,0), (0,1), (1,0), (2,1) is:

$$P(x) = -\frac{2}{3}x^3 - x^2 - \frac{2}{3}x + 1$$



Polynomial Representation by Points

If you have n+1 points on the polynomial:

$\{x_0, x_1, x_2, \ldots\ldots, x_n\}$

You can reconstruct coefficients of P(x) from n + 1 values $\{A(x_0), A(x_1), \ldots\ldots, A(x_n)\}$, which requires **$O(n^2)$** time using Lagrange Interpolation

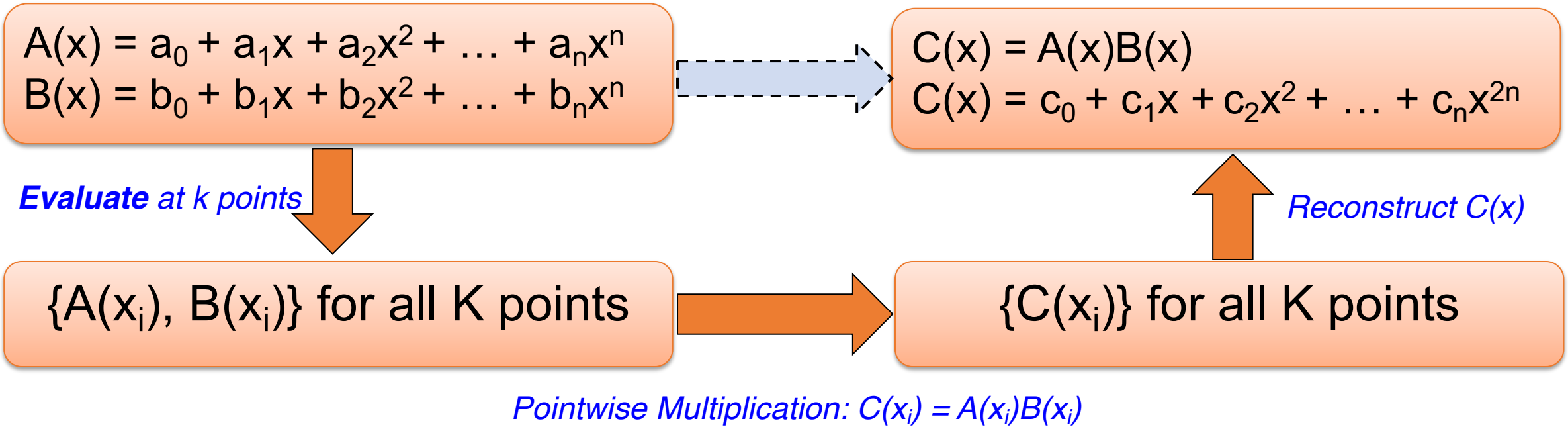
-It is called a **Value Representation** of a polynomial.

A New Approach for Polynomial Multiplication

Evaluate A(x) and B(x) at $x_1, x_2, \ldots, x_k$, then obtain the values of the resultant polynomial, C(x) at the k points by:

- **$A(x_i) * B(x_i) = C(x_i)$**

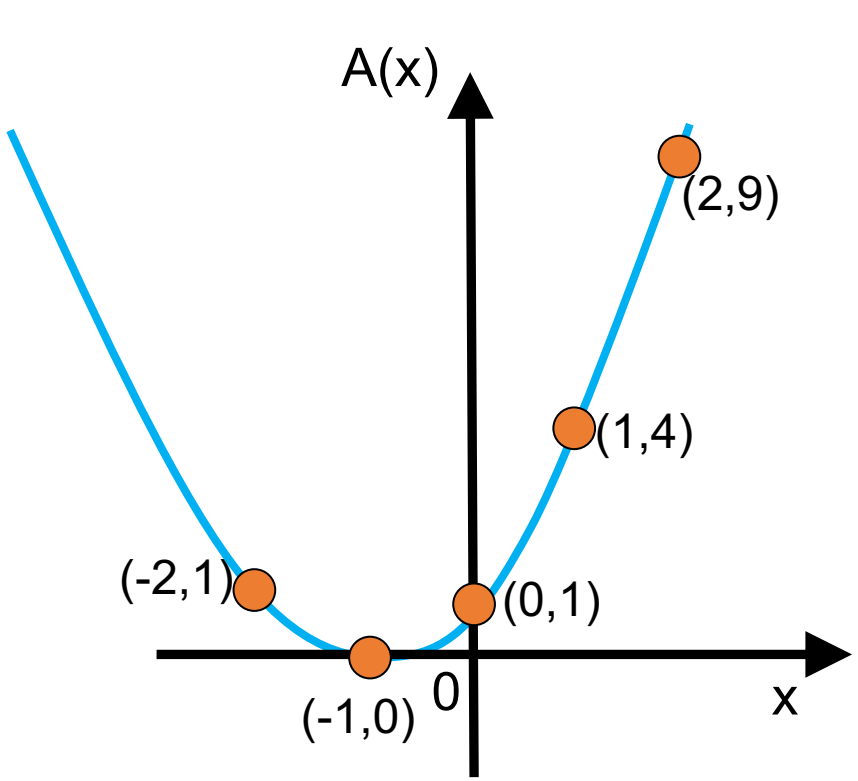
Using the K points on the polynomial C to **interpolate** the polynomial.

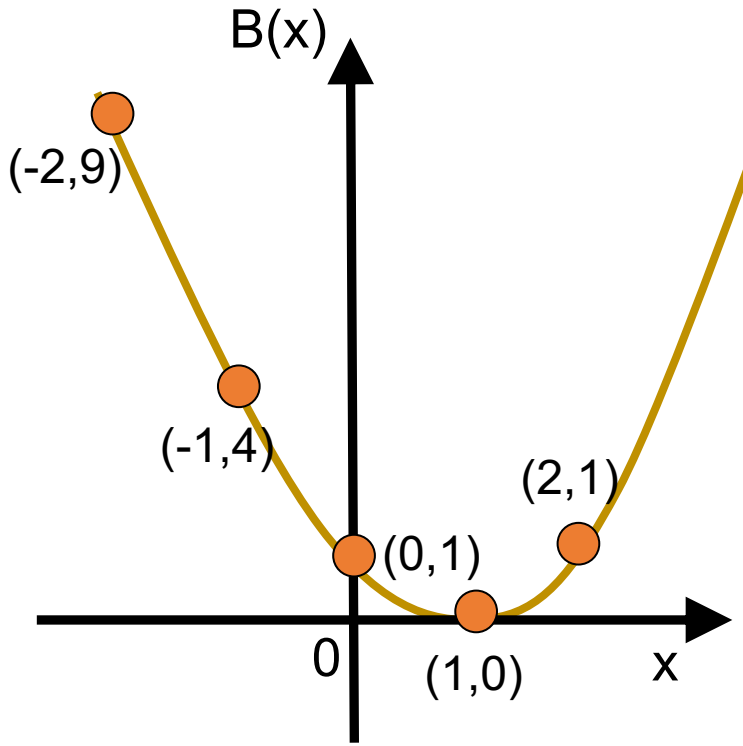


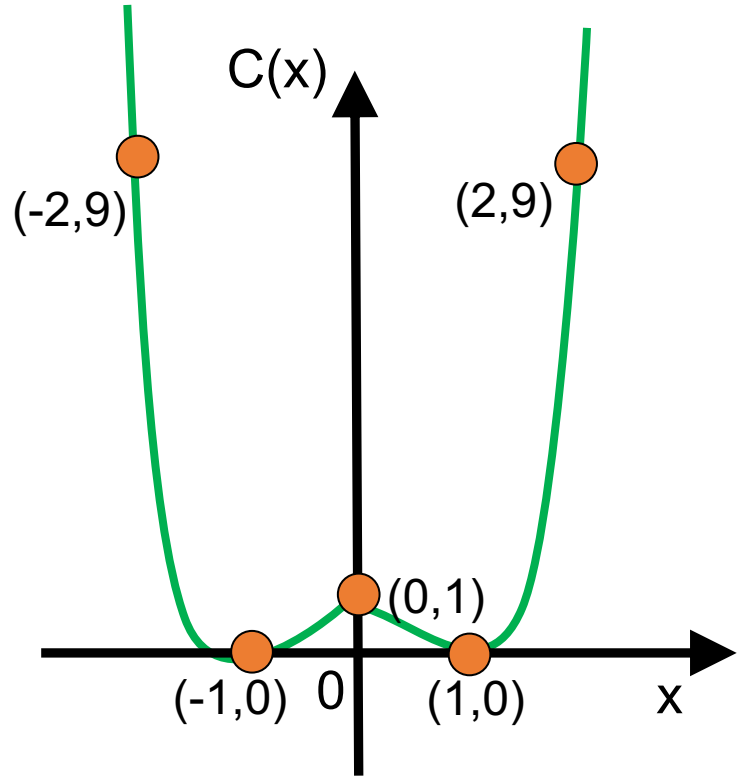
Value-based polynomial Multiplication

$$A(x) = x^2 + 2x + 1, \qquad B(x) = x^2 - 2x + 1, \qquad C(x) = A(x)B(x)$$

Five points for x = {-2, -1, 0, 1, 2}, multiplication in value representation is only O(n)



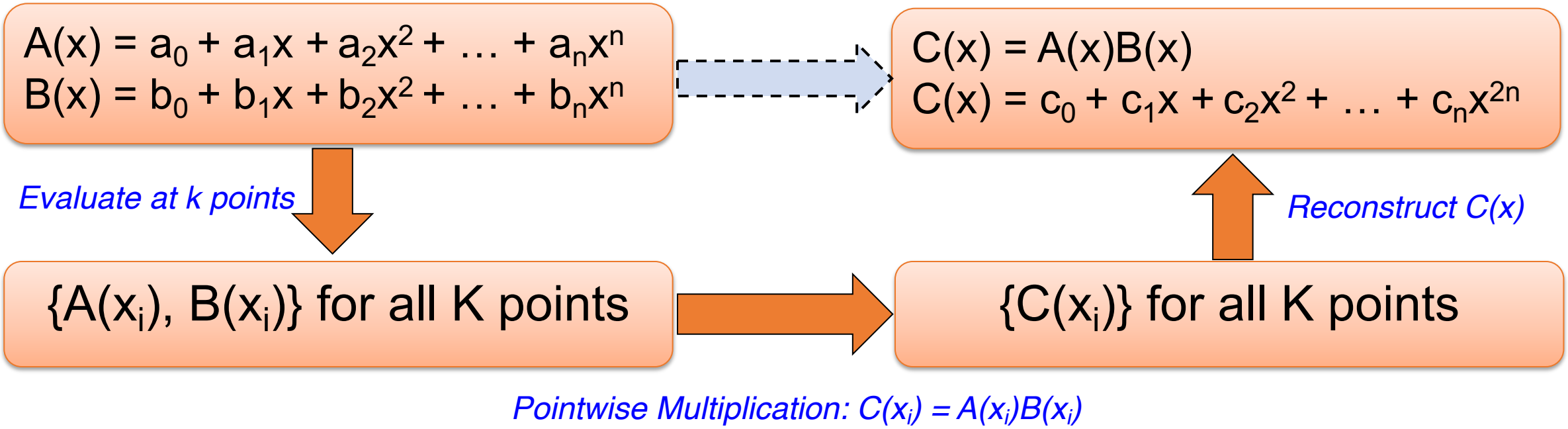




A New Approach for Polynomial Multiplication

Question 1: How many points do we need?

Question 2: What is the complexity?

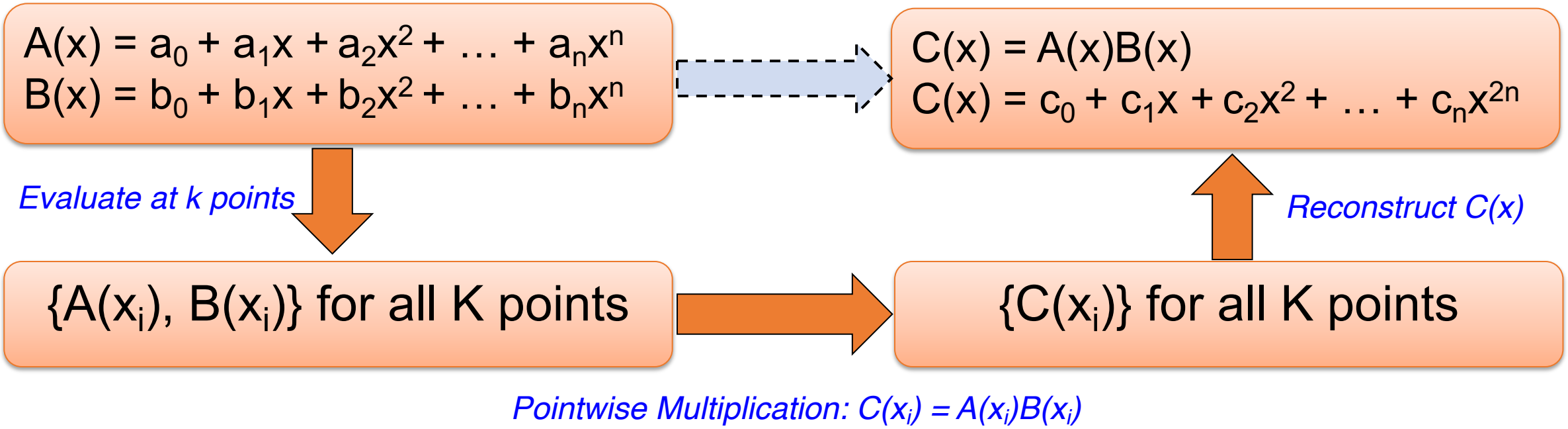


A New Approach for Polynomial Multiplication

Question 1: How many points do we need?

We need **K = 2n + 1** points to reconstruct a polynomial of degree 2n.

Question 2: What is the complexity?



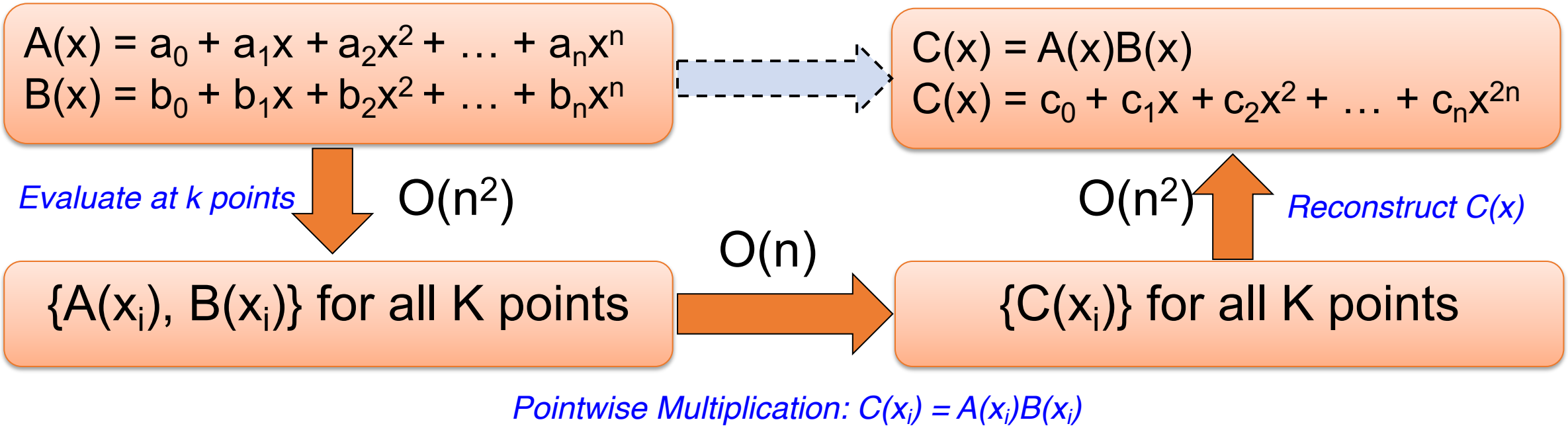
A New Approach for Polynomial Multiplication

Question 1: How many points do we need?

We need **K = 2n + 1** points to reconstruct a polynomial of degree 2n.

Question 2: What is the complexity?

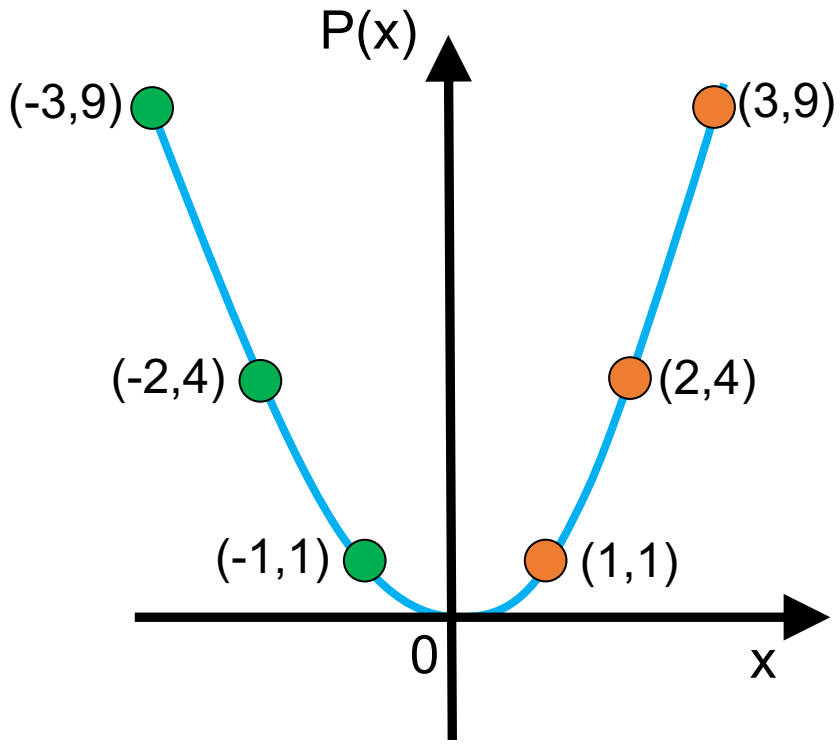
Still **$O(n^2)$** if no other smart ideas are introduced.



Evaluation can be faster!

- Instead of considering a general polynomial, let's look at some special ones:

$$P(x) = x^2$$



The value of one point immediately implies the value of another: P(x) = P(-x)

P(-1) = P(1) = 1 (-1,1), (1,1)

P(-2) = P(2) = 4 (-2,4), (2,4)

P(-3) = P(3) = 9 (-3,9), (3,9)

Evaluate 3 times, but you get 6 points on the polynomial !

Evaluation can be faster!

- Instead of considering a general polynomial, let's look at some special ones:

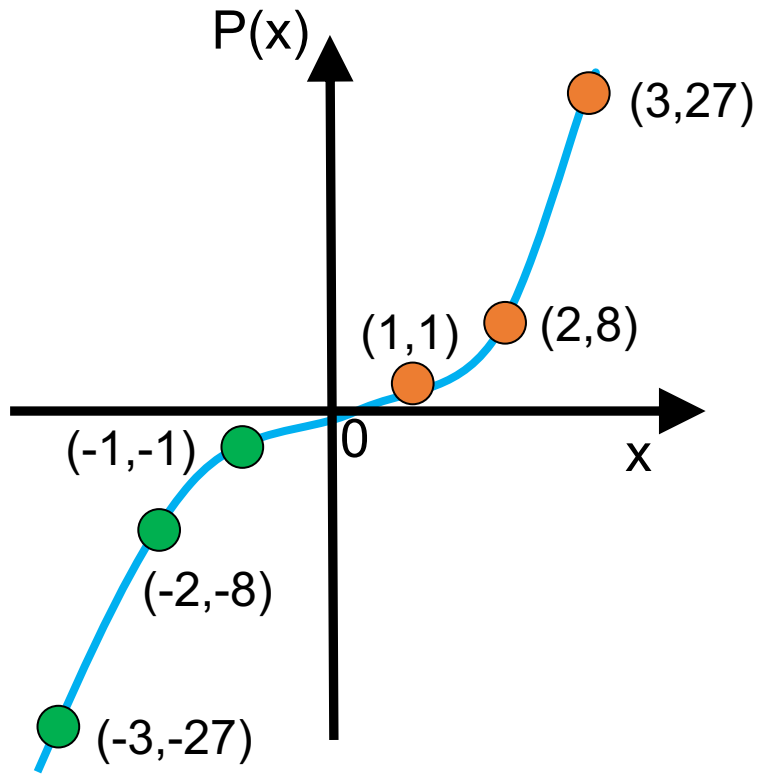
$$P(x) = x^3$$

The value of one point immediately implies the value of another: P(x) = -P(-x)

P(-1) = -P(1) = -1 (-1,-1), (1,1)

P(-2) = -P(2) = -8 (-2,-8), (2,8)

P(-3) = -P(3) = -9 (-3,-27), (3,27)



In the above odd and even single term polynomials, We only need to evaluate half of the points from which we immediately know **the value of the respective negative points**

Evaluation can be faster!

- Extend this idea to a more general polynomial:

$$P(x) = 3x^6 + 4x^5 + 2x^4 - 7x^3 - 8x^2 + 6x + 1$$

- Evaluating at **n** points: $\pm x_1, \pm x_2, \ldots\ldots, \pm x_{n/2}$ **(they are positive/negative paired)**
- **Split P(x):** $P(x) = (3x^6 + 2x^4 - 8x^2 + 1) + (4x^5 - 7x^3 + 6x)$

$$= \underbrace{(3x^6 + 2x^4 - 8x^2 + 1)}_{P_{even}(x^2)} + x\underbrace{(4x^4 - 7x^2 + 6)}_{P_{odd}(x^2)}$$

$$P_{even}(x^2) = 3(x^2)^3 + 2(x^2)^2 - 8(x^2) + 1 \qquad P_{odd}(x^2) = 4(x^2)^2 - 7(x^2) + 6$$

$$\mathbf{P(x_i) = P_{even}(x_i^2) + x_i P_{odd}(x_i^2)}$$

$$\mathbf{P(-x_i) = P_{even}(x_i^2) - x_i P_{odd}(x_i^2)}$$

Evaluation can be faster!

$$P(x_i) = P_{even}(x_i^2) + x_i P_{odd}(x_i^2)$$
$$P(-x_i) = P_{even}(x_i^2) - x_i P_{odd}(x_i^2)$$

$$P_{even}(x^2) = 3(x^2)^3 + 2(x^2)^2 - 8(x^2) + 1 \qquad P_{odd}(x^2) = 4(x^2)^2 - 7(x^2) + 6$$

- What are the degrees of P_{even} and P_{odd}? $n/2$!
- Now we only need to evaluate P_{even} and P_{odd} at $n/2$ points: : $x_1^2, x_2^2, \ldots\ldots, x_{n/2}^2$

  to obtain the values for the n points: $P(\pm x_1), P(\pm x_2), \ldots\ldots, P(\pm x_{n/2})$

Divide and Conquer??

How to make a **O(NlogN)** recursive algorithm ?

Evaluation of an n degree polynomial $P(x)$ at points $\{\pm x_1, \pm x_2, \dots, \pm x_{n/2}\}$

Evaluate $P_{even}(x^2) = p_0 + p_2 x^2 + \cdots + p_n x^n$ at points $\{x_1^2, x_2^2, \dots, x_{n/2}^2\}$

Evaluate $P_{odd}(x^2) = p_1 x + p_3 x^3 + \cdots + p_{n-1} x^{n-1}$ at points $\{x_1^2, x_2^2, \dots, x_{n/2}^2\}$

??? Can we further split it to two smaller polynomials. **???**

$\{\pm x_1, \pm x_2, \dots, \pm x_{n/2}\}$ are $\pm$ paired

$\{x_1^2, x_2^2, \dots, x_{n/2}^2\}$ **are NOT $\pm$ paired !!**

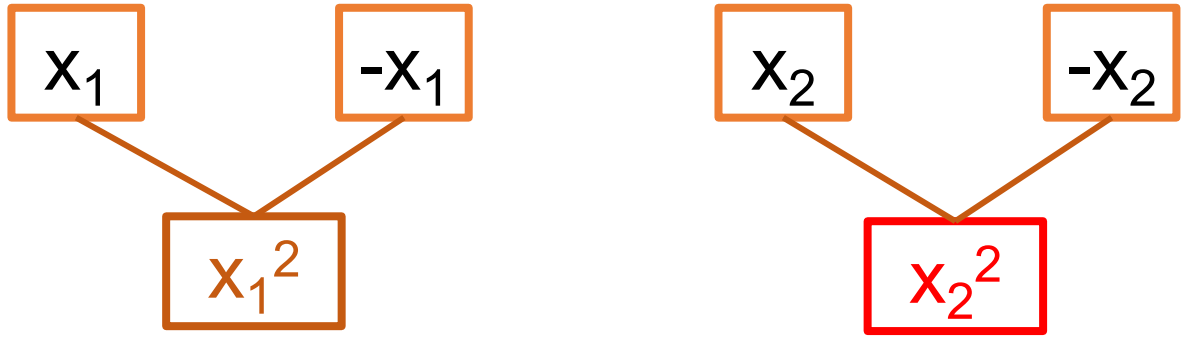
What kind of points are needed?

- A simple example: $P(x) = -7x^3 - 8x^2 + 6x + 1$

- We need degree + 1 = **4** points for this cubic polynomial, which are two pairs of positive/negative points for P(x): x_1 , $-x_1$, x_2 , $-x_2$.

$$\mathbf{P(x_i) = P_{even}(x_i^2) + x_i P_{odd}(x_i^2)}$$

For the even and odd parts, we use:



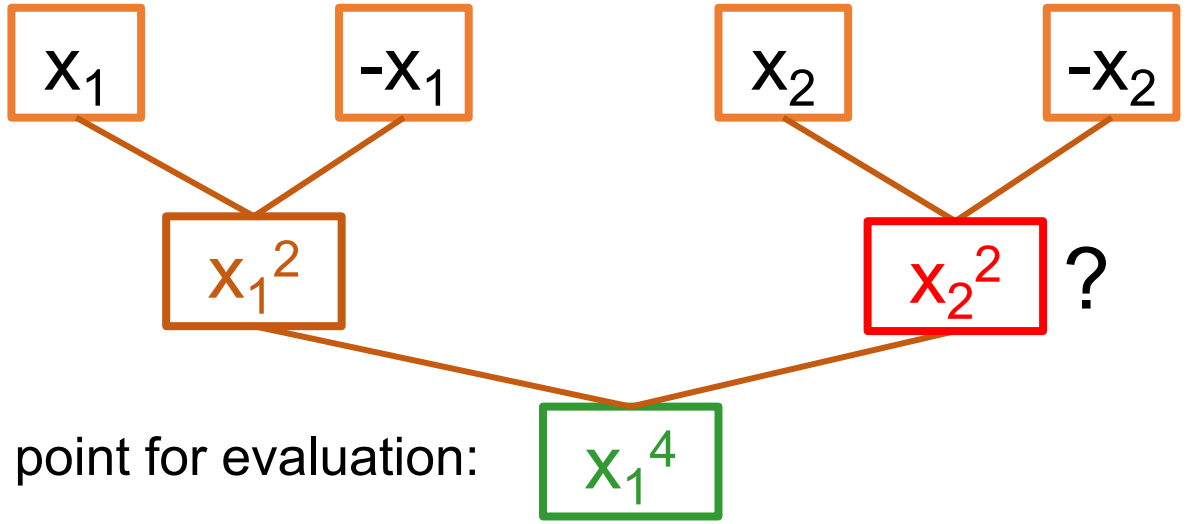
What kind of points are needed?

- A simple example: $P(x) = -7x^3 - 8x^2 + 6x + 1$

- We need degree + 1 = **4** points for this cubic polynomial, which are two pairs of positive/negative points for P(x): x_1 , $-x_1$, x_2 , $-x_2$.

$$\mathbf{P(x_i) = P_{even}(x_i^2) + x_i P_{odd}(x_i^2)}$$

For the even and odd parts, we use:



Further split $\mathbf{P_{even}(x_i^2)}$ and $\mathbf{P_{odd}(x_i^2)}$, the point for evaluation:

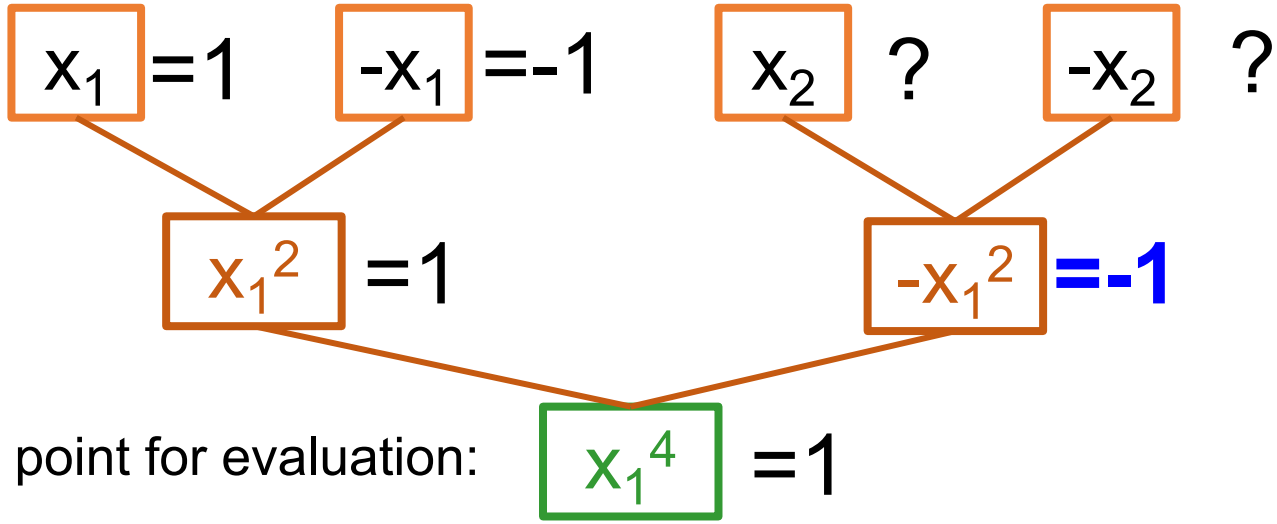
What kind of points are needed?

- A simple example: $P(x) = -7x^3 - 8x^2 + 6x + 1$

- We need degree + 1 = **4** points for this cubic polynomial, which are two pairs of positive/negative points for P(x): x_1 , $-x_1$, x_2 , $-x_2$.

$$\mathbf{P(x_i) = P_{even}(x_i^2) + x_i P_{odd}(x_i^2)}$$

For the even and odd parts, we use:



Further split $\mathbf{P_{even}(x_i^2)}$ and $\mathbf{P_{odd}(x_i^2)}$, the point for evaluation:

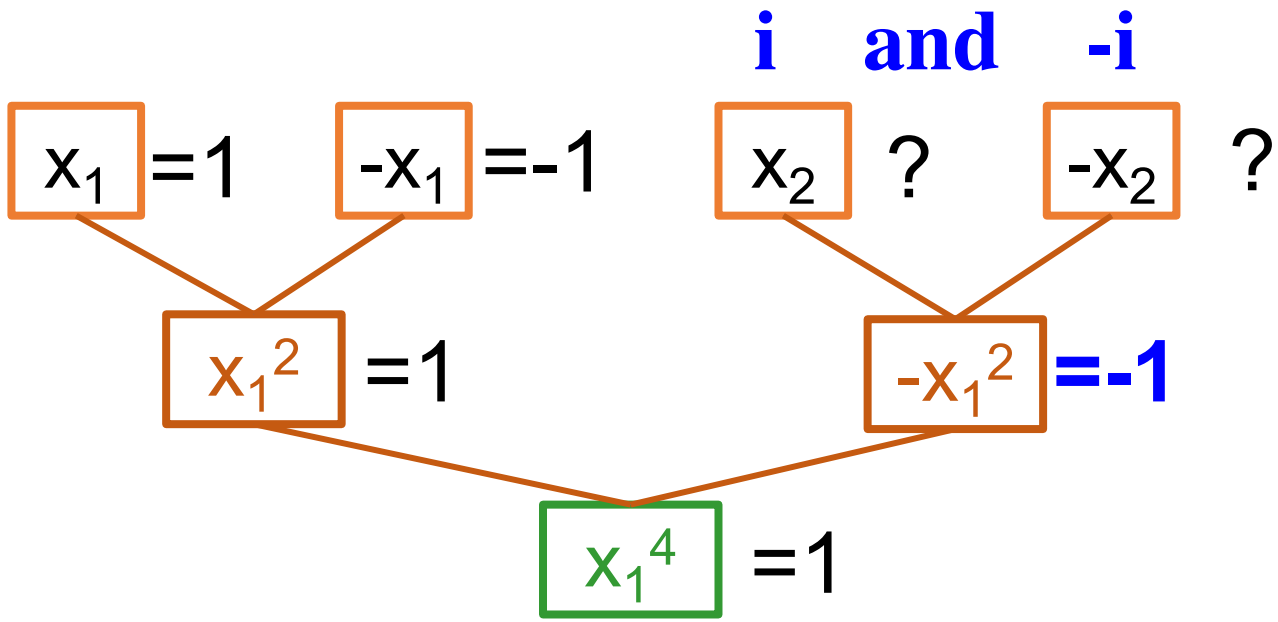
We need something which is not a "real" number

- **What x_2 should we choose so that when we square it, we get -1?**
- **Complex number!**

What we actually did here:
We solve the equation

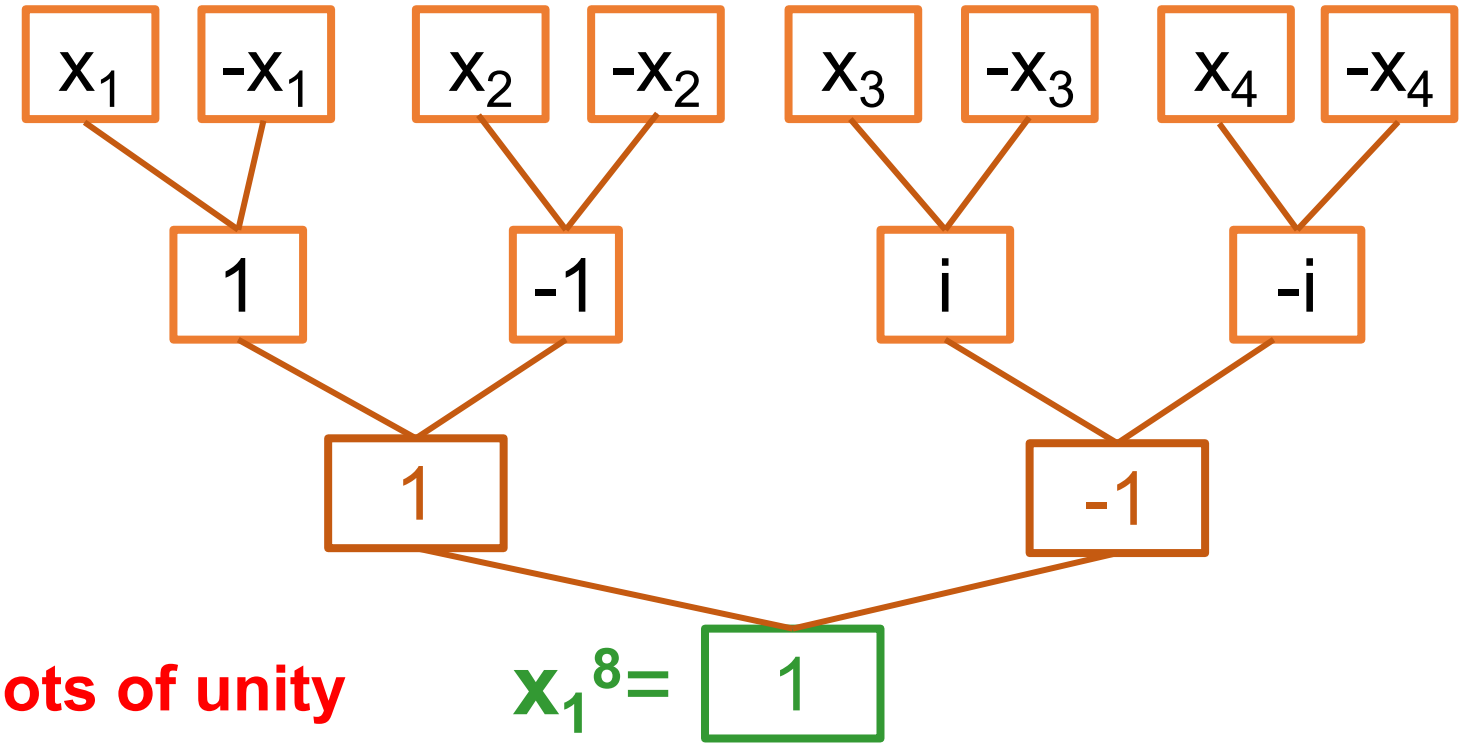
$$x^4 = 1$$

It has four solutions.
They are the 4^{th} roots of unity



General Solution

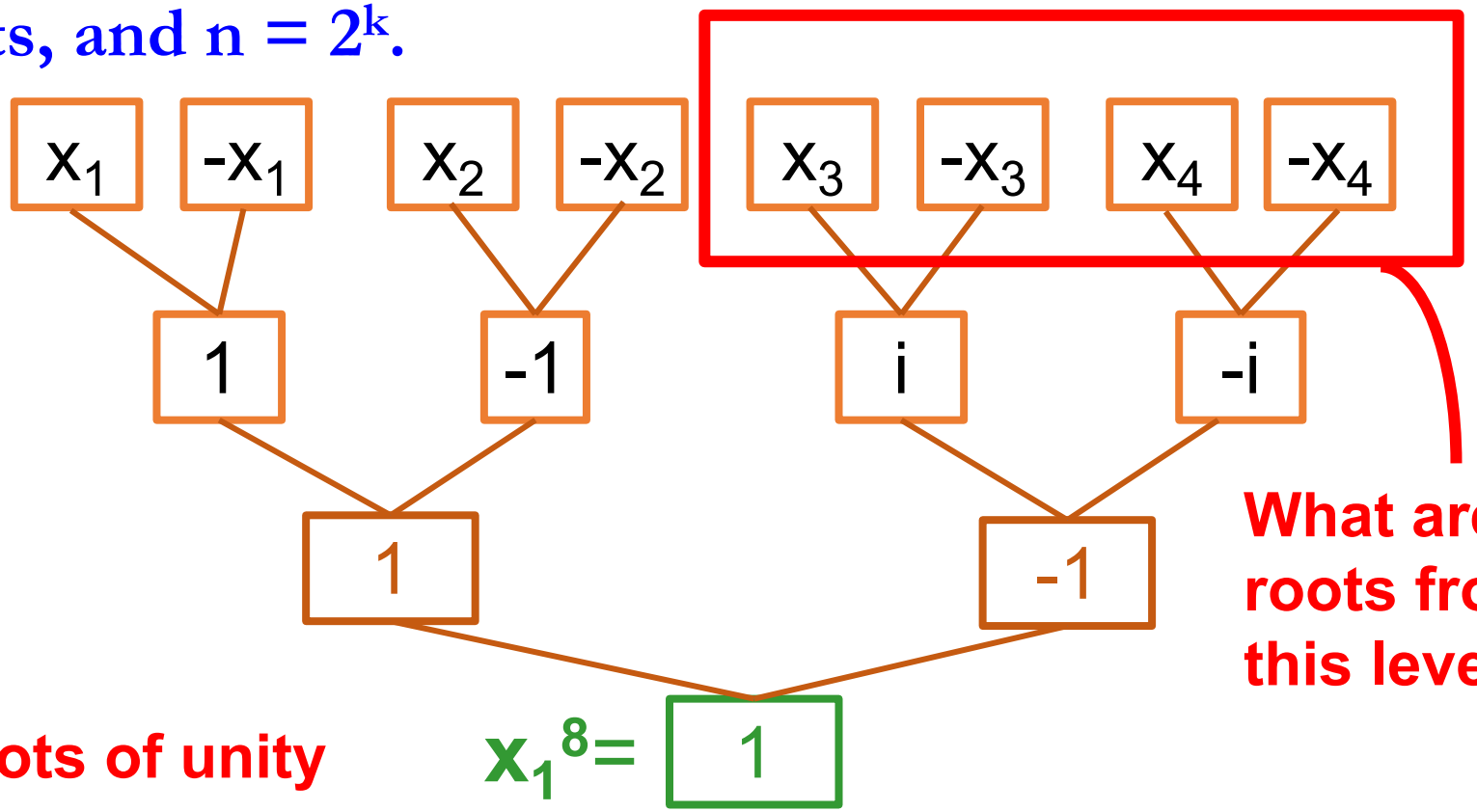
- If P(x) has a degree 5, we need n > 6 points. Let n = 8 (powers of 2).



The points are the 8th roots of unity

General Solution

- **If P(x) has a degree d,** $P(x) = p_0 + p_1x + p_2x^2 + \cdots + p_nx^n$
  we use n > d+1 points, and n = 2^k.



The points are the n^{th} roots of unity

Recall Complex Numbers

- A complex number is:

  $x = x.\mathrm{Re} + i * x.\mathrm{Im}$ (Re = real part, Im = imaginary part), $i = \sqrt{-1}$ ($i^2 = -1$)

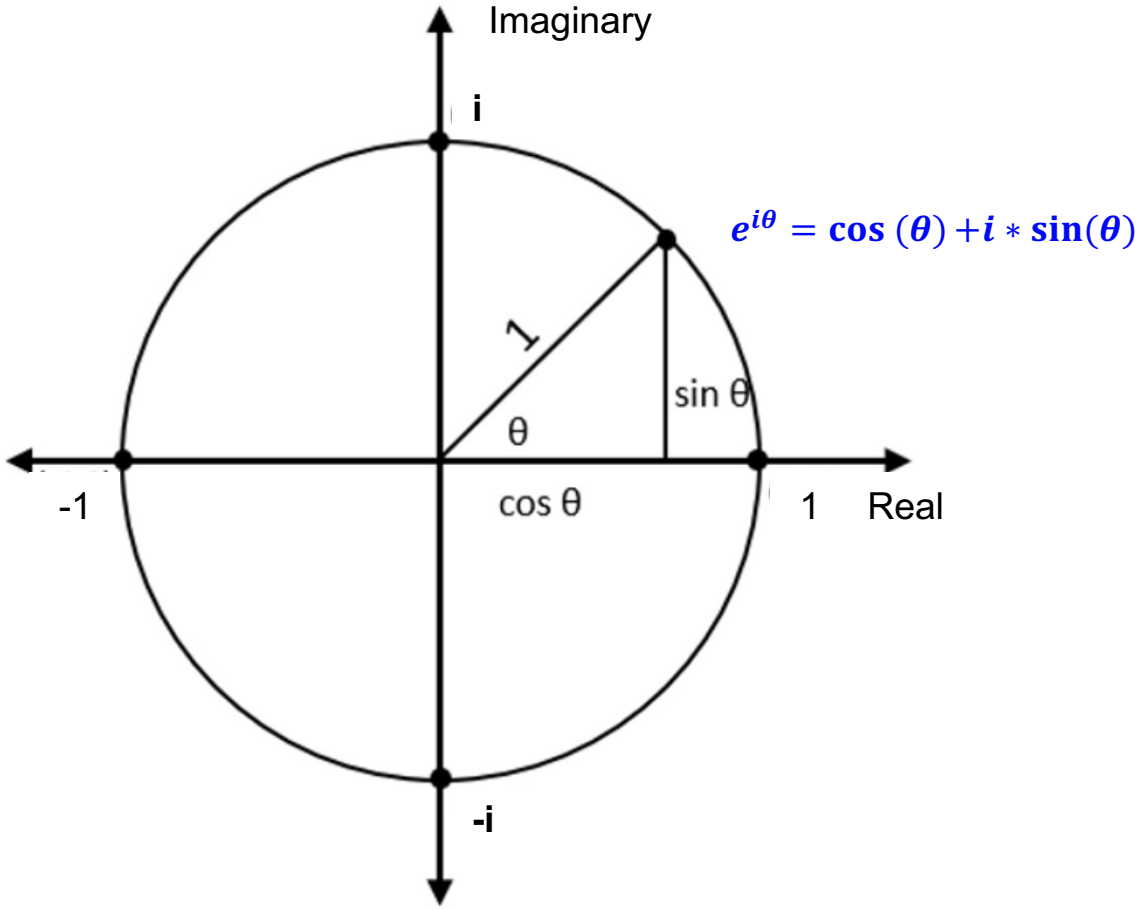
- Basic operations:
  - Addition:
    - $(a + b * i) + (c + d * i) = (a + c) + (b + d) * i$
  - Subtraction:
    - $(a + b * i) - (c + d * i) = (a - c) + (b - d) * i$
  - Multiplication
    - $(a + b * i) * (c + d * i) = (ac - bd) + (bc + ad) * i$
  - Exponential
    - $e^{a+b*i} = e^a * e^{b*i} = e^a * (\cos b + i * \sin b) = e^a \cos b + e^a \sin b * i$

Nth roots of unity

- **Euler's formula states that for any real number θ :**

  $e^{i\theta} = \cos(\theta) + i * \sin(\theta)$ **(e is the base of the natural logarithm)**

- The angle is θ
- The real part (x axis) is $\cos(\theta)$
- The imaginary part (y axis) is $\sin(\theta)$
- Increase θ -> increase angle

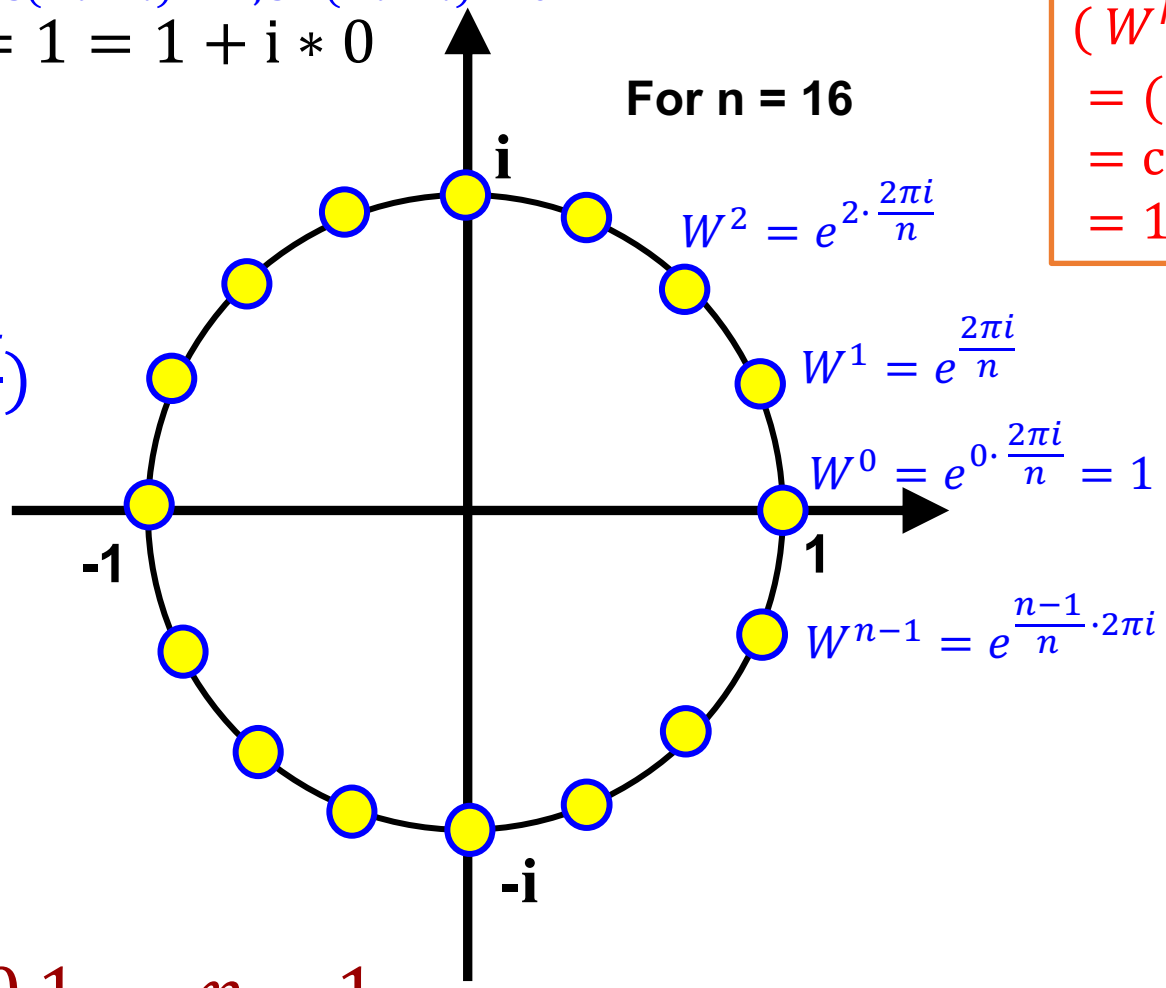


Nth roots of unity

$\cos(2\pi * k) = 1, \sin(2\pi * k) = 0$

Nth roots of unity: $z^n = 1 = 1 + \mathrm{i} * 0$

$$Let \quad W = e^{2\pi i/n}$$
$$= \cos\left(\frac{2\pi}{n}\right) + i * \sin\left(\frac{2\pi}{n}\right)$$



$$(W^k)^n = (e^{k*2\pi i/n})^n = e^{k*2\pi i} = \cos(k*2\pi) + i\sin(k*2\pi) = 1$$

Each W^k is an Nth root of unity

$$W^k = e^{k\cdot 2\pi i/n} \quad k = 0,1,\ldots,n-1$$

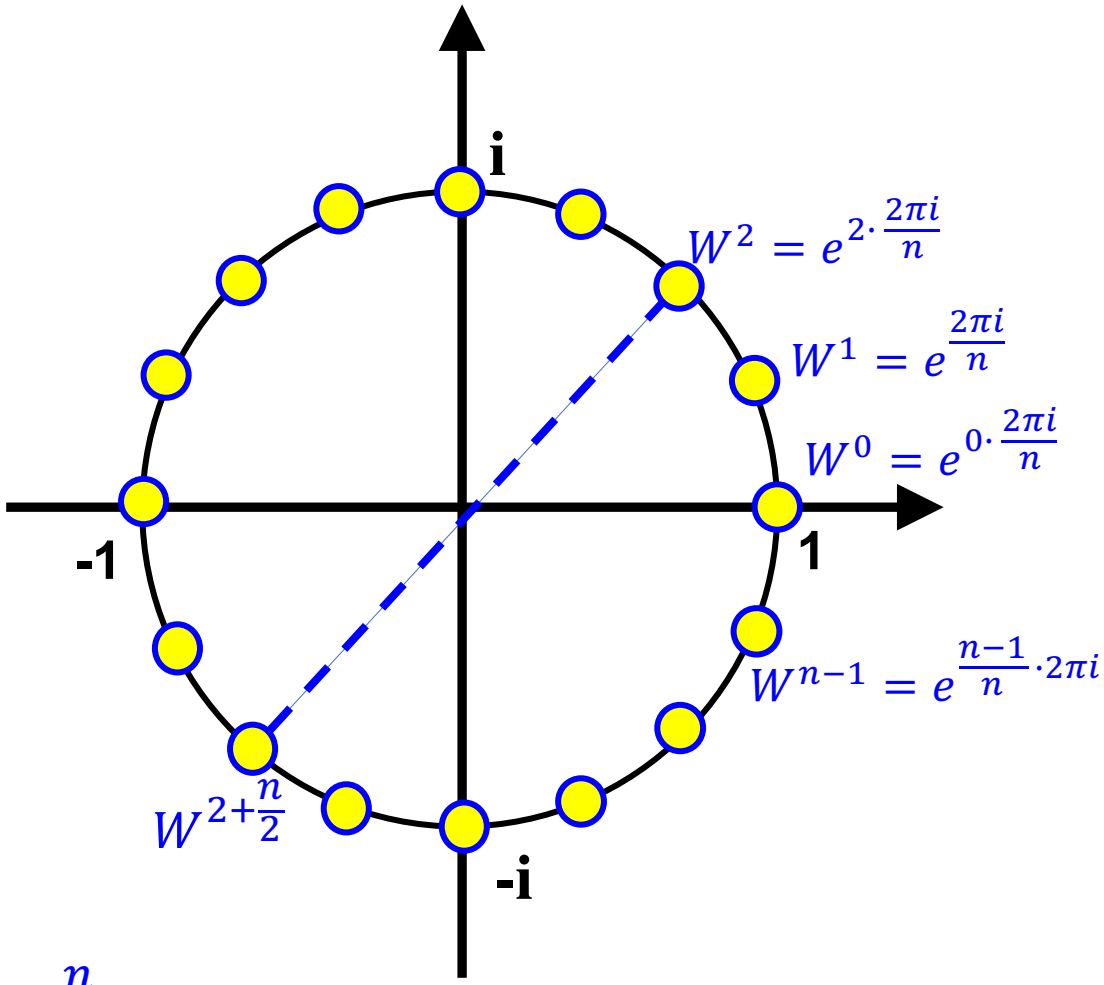
We will evaluate P(x) at $W^0,\ W^1,\ W^2, \ldots\ldots,\ W^{n-1}$

Why does this group of points work?

Evaluate P(x) at $W^0,\ W^1,\ W^2, \ldots\ldots,\ W^{n-1}$

They are $\pm$ paired points!

$W^{j+\frac{n}{2}} = -W^j \rightarrow (W^{j+\frac{n}{2}}, -W^j)\ are\ \pm paired$

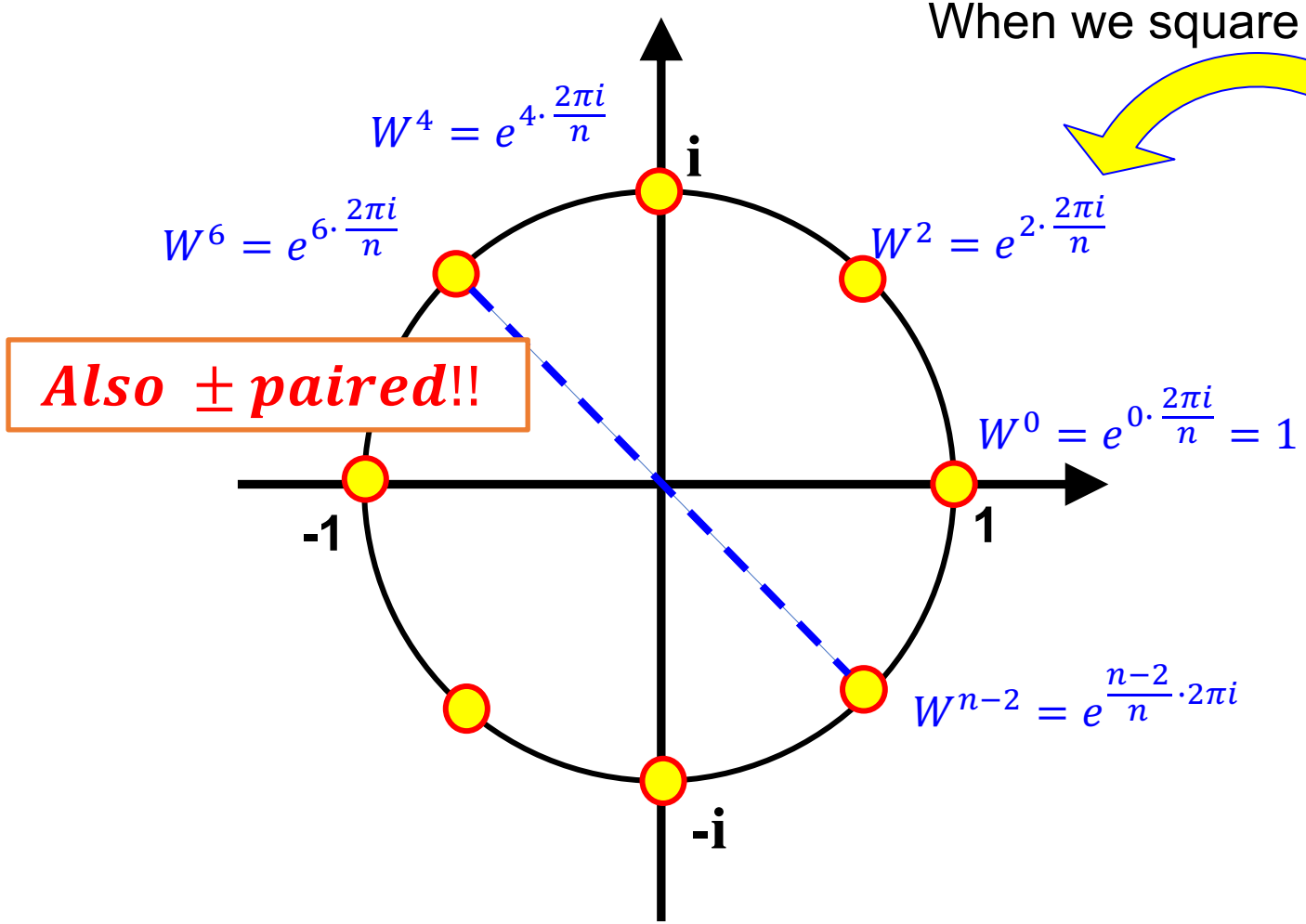


$W^0,\ W^1,\ W^2, \ldots\ldots,\ W^{n-1} \Rightarrow \pm W^0, \pm W^1, \ldots, \pm W^{\frac{n}{2}-1}$

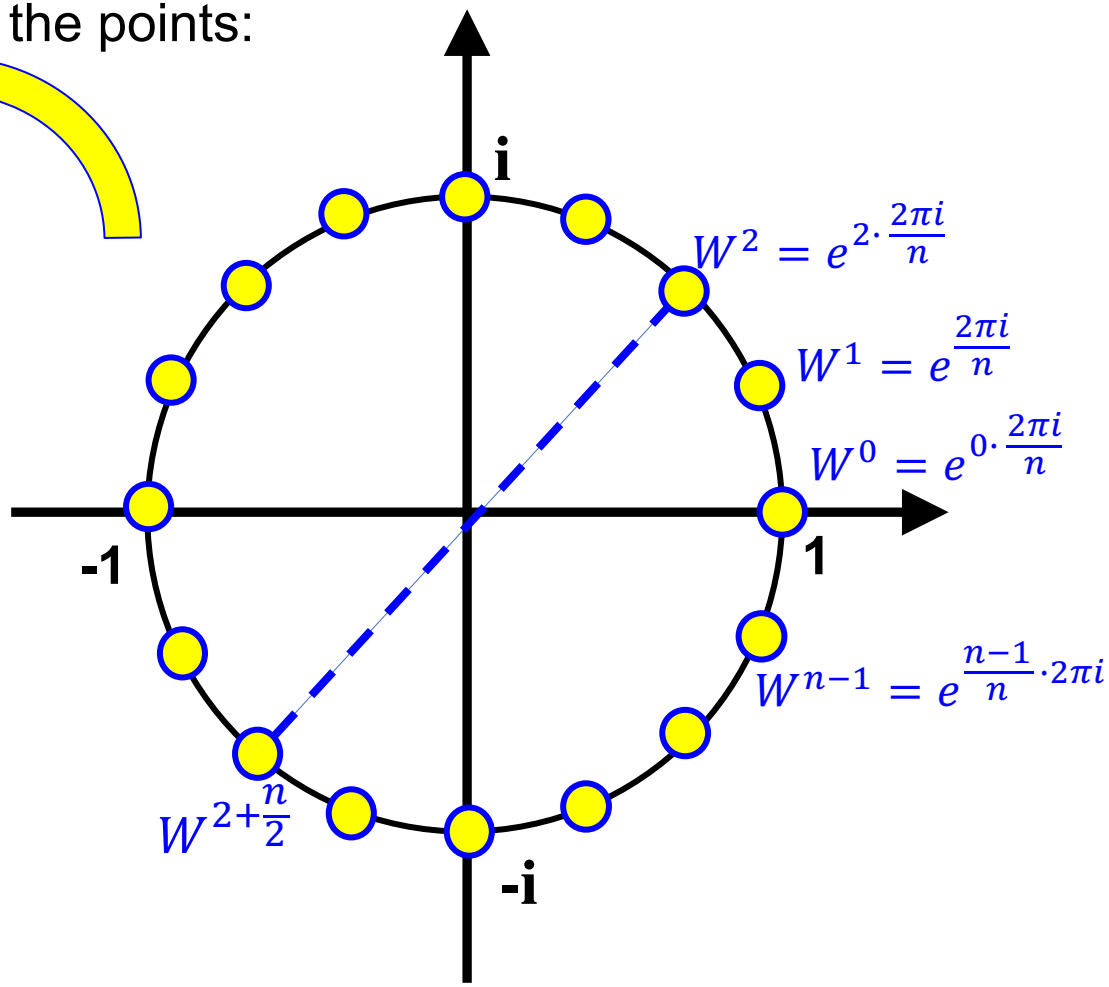
Why does this group of points work?

$$P(x_i) = P_{even}(x_i^2) + x_i P_{odd}(x_i^2)$$
$$P(-x_i) = P_{even}(x_i^2) - x_i P_{odd}(x_i^2)$$

When we square the points:



Also ± paired!!



We end up with these points

n/2 roots of unity: $W^0,\ W^2,\ W^4, \ldots\ldots,\ W^{2\left(\frac{n}{2}-1\right)} = (\pm W^0)^2, (\pm W^1)^2, \ldots, \left(\pm W^{\frac{n}{2}-1}\right)^2$

Fast Fourier Transform for Polynomial Evaluation

FFT

$$P(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_{n-1} x^{n-1}$$

$$Evaluate\ at\ \{W^0, W^1, \ldots, W^{n-1}\},\ where\ W = e^{\frac{2\pi i}{n}}$$

A recursive case is two calls of FFTs:

The base case: when n = 1, P(1)

FFT

$$P_{even}(x^2) = p_0 + p_2 x^2 + \cdots + p_n x^{n-2}$$

$$Evaluate\ at\ points\ \{W^0, W^2, \ldots, W^{n-2}\}$$

FFT

$$P_{odd}(x^2) = p_1 + p_3 x^2 + \cdots + p_{n-1} x^{n-2}$$

$$Evaluate\ \ at\ points \{W^0, W^2, \ldots, W^{n-2}\}$$

$$\boldsymbol{P(x_k) = P_{even}(x_k^2) + x_k P_{odd}(x_k^2)}$$

$$\boldsymbol{P(-x_k) = P_{even}(x_k^2) - x_k P_{odd}(x_k^2)}$$

$$\boldsymbol{k \in \{0, 1, \ldots, (n/2 - 1)\}}$$

Fast Fourier Transform for Polynomial Evaluation

FFT

$$P(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_{n-1} x^{n-1}$$

$$Evaluate\ at\ \{W^0, W^1, \ldots, W^{n-1}\},\ where\ W = e^{\frac{2\pi i}{n}}$$

A recursive case is two calls of FFTs:

The base case: when n = 1, P(1)

FFT

$$P_{even}(x^2) = p_0 + p_2 x^2 + \cdots + p_n x^{n-2}$$

$$Evaluate\ at\ points\ \{W^0, W^2, \ldots, W^{n-2}\}$$

FFT

$$P_{odd}(x^2) = p_1 + p_3 x^2 + \cdots + p_{n-1} x^{n-2}$$

$$Evaluate\ \ at\ points \{W^0, W^2, \ldots, W^{n-2}\}$$

$$x_k = W^k$$

$$\boldsymbol{P(W^k) = P_{even}(W^{2k}) + W^k P_{odd}(W^{2k})}$$

$$\boldsymbol{P(-W^k) = P_{even}(W^{2k}) - W^k P_{odd}(W^{2k})}$$

$$\boldsymbol{k \in \{0, 1, \ldots, (n/2 - 1)\}}$$

Fast Fourier Transform for Polynomial Evaluation

FFT

$$P(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_{n-1} x^{n-1}$$

$$Evaluate\ at\ \{W^0, W^1, \ldots, W^{n-1}\},\ where\ W = e^{\frac{2\pi i}{n}}$$

A recursive case is two calls of FFTs:

The base case: when n = 1, P(1)

FFT

$$P_{even}(x^2) = p_0 + p_2 x^2 + \cdots + p_n x^{n-2}$$

$$Evaluate\ at\ points\ \{W^0, W^2, \ldots, W^{n-2}\}$$

$$y_e = [P_{even}(W^0), P_{even}(W^2), \ldots]$$

FFT

$$P_{odd}(x^2) = p_1 + p_3 x^2 + \cdots + p_{n-1} x^{n-2}$$

$$Evaluate\ \ at\ points\{W^0, W^2, \ldots, W^{n-2}\}$$

$$y_o = [P_{odd}(W^0), P_{odd}(W^2), \ldots]$$

$$x_k = W^k$$

$$W^{k+\frac{n}{2}} = -W^k$$

$$\boldsymbol{P}(W^{\boldsymbol{k}}) = \quad y_e[k] + W^{\boldsymbol{k}} y_o[k]$$

$$\boldsymbol{P}\left(W^{\boldsymbol{k}+\frac{\boldsymbol{n}}{\boldsymbol{2}}}\right) = \quad y_e[k] - W^{\boldsymbol{k}} y_o[k]$$

$$\boldsymbol{k} \in \{\boldsymbol{0}, \boldsymbol{1}, \ldots, (\boldsymbol{n/2 - 1})\}$$

Fast Fourier Transform for Polynomial Evaluation

FFT

$$P(x) = p_0 + p_1x + p_2x^2 + \cdots + p_{n-1}x^{n-1}$$

$$Evaluate\ at\ \{W^0, W^1, \dots, W^{n-1}\},\ where\ W = e^{\frac{2\pi i}{n}}$$

A recursive case is two calls of FFTs:

The base case: when n = 1, P(1)

FFT

$$P_{even}(x^2) = p_0 + p_2x^2 + \cdots + p_nx^{n-2}$$

$$Evaluate\ at\ points\ \{W^0, W^2, \dots, W^{n-2}\}$$

$$y_e = [P_{even}(W^0), P_{even}(W^2), \dots]$$

FFT

$$P_{odd}(x^2) = p_1 + p_3x^2 + \cdots + p_{n-1}x^{n-2}$$

$$Evaluate\ \ at\ points\{W^0, W^2, \dots, W^{n-2}\}$$

$$y_o = [P_{odd}(W^0), P_{odd}(W^2), \dots]$$

$$x_k = W^k$$

$$W^{k+\frac{n}{2}} = -W^k$$

$$\boldsymbol{P}(W^{\boldsymbol{k}}) = \quad y_e[k] + W^{\boldsymbol{k}}y_o[k]$$

$$\boldsymbol{P}\left(W^{\boldsymbol{k}+\frac{\boldsymbol{n}}{\boldsymbol{2}}}\right) = \quad y_e[k] - W^{\boldsymbol{k}}y_o[k]$$

$$\boldsymbol{k} \in \{\boldsymbol{0}, \boldsymbol{1}, \dots, (\boldsymbol{n/2 - 1})\}$$

$$y_e[k] = P_{even}(W^{2k})$$

$$y_o[k] = P_{odd}(W^{2k})$$

Change the denotation to make the coding easier!

Output: $y = [P(W^0), P(W^1), \dots, P(W^{n-1})]$, *where* $W = e^{\frac{2\pi i}{n}}$

Fast Fourier Transform for Polynomial Evaluation

function FFT (P) //P is a sequence of coefficients

n = length(P); //n needs to be a power of 2

if (n==1)

return P; // It means the polynomial only has p_0

$W = e^{\frac{2\pi i}{n}}$;

$P_e, P_o = [p_0, p_2, \dots, p_{n-2}], [p_1, p_3, \dots, p_{n-1}]$

y_e, y_o = FFT(P_e), FFT(P_o);

Array[n] y;

for j from 0 to n/2:

$y[j] = y_e[j] + W^j y_o[j]$;

$y[j + n/2] = y_e[j] - W^j y_o[j]$;

return y;

$$P(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_{n-1} x^{n-1}$$

$$Evaluate\ at\ \{W^0, W^1, \dots, W^{n-1}\},\ where\ W = e^{\frac{2\pi i}{n}}$$

FFT

The base case: when n = 1, P(1)

FFT

$$P_{even}(x^2) = p_0 + p_2 x^2 + \cdots + p_n x^n$$

$$Evaluate\ at\ points\ \{W^0, W^2, \dots, W^{n-2}\}$$

$$y_e = [P_{even}(W^0), P_{even}(W^2), \dots]$$

FFT

$$P_{odd}(x^2) = p_1 + p_3 x^2 + \cdots + p_{n-1} x^{n-2}$$

$$Evaluate\ \ at\ points\{W^0, W^2, \dots, W^{n-2}\}$$

$$y_o = [P_{odd}(W^0), P_{odd}(W^2), \dots]$$

$$\boldsymbol{P}(W^{\boldsymbol{k}}) = y_e[k] + W^{\boldsymbol{k}} y_o[k]$$

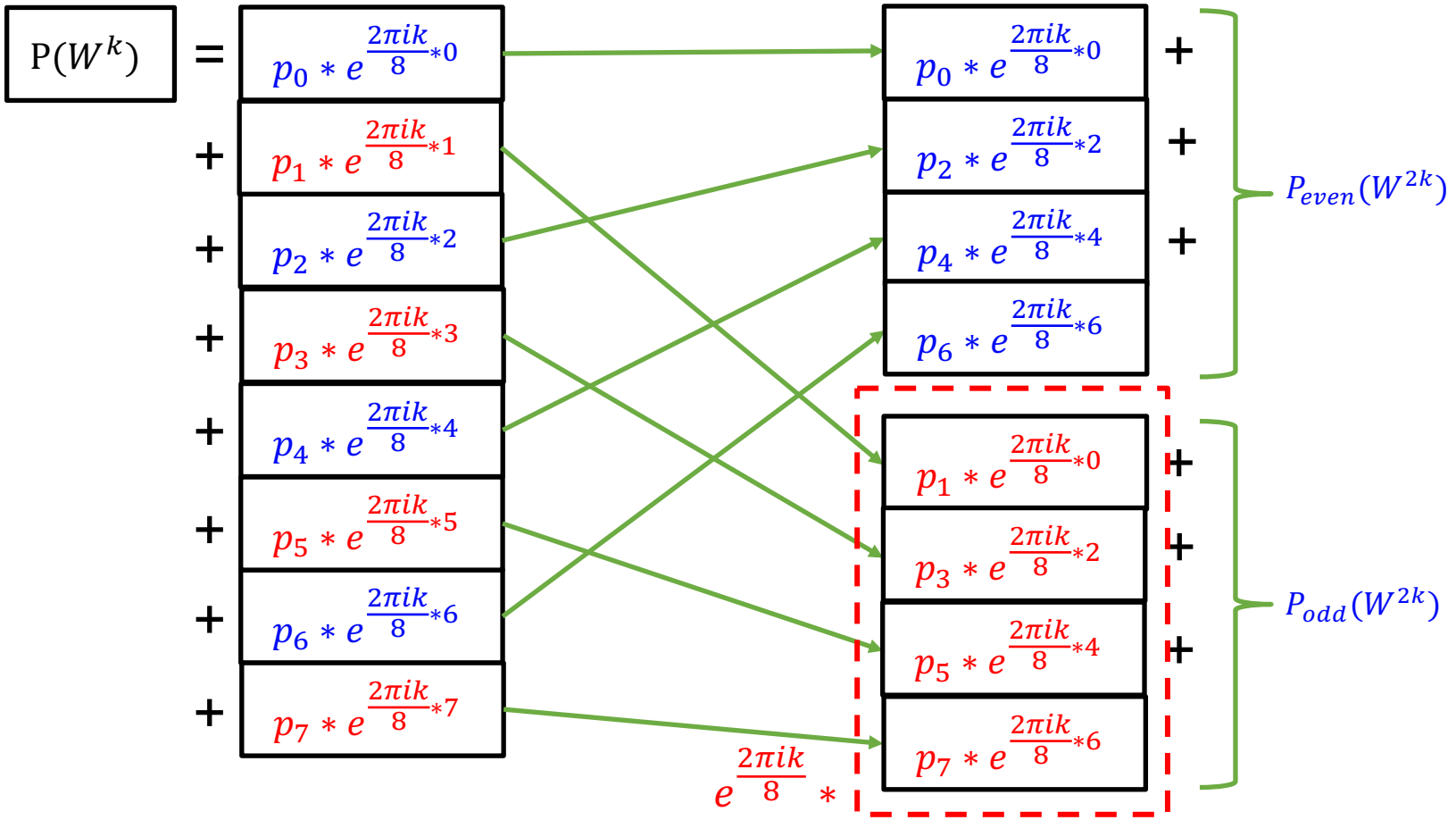
$$\boldsymbol{P}\left(W^{\boldsymbol{k}+\frac{\boldsymbol{n}}{\boldsymbol{2}}}\right) = y_e[k] - W^{\boldsymbol{k}} y_o[k]$$

$$\boldsymbol{k} \in \{\boldsymbol{0}, \boldsymbol{1}, \dots, (\boldsymbol{n/2 - 1})\}$$

Output: $y = [P(W^0), P(W^1), \dots, P(W^{n-1})]$

Fast Fourier Transform – 8 points example

- **Example**: 7 degree polynomial, 8 coefficients sequence, $\mathrm{P(x)} = \mathrm{p}_0 + p_1 x + p_2 x^2 + \cdots + p_7 x^7$



Fast Fourier Transform – 8 points example

- Consider even index, for $k = 0, \ldots, 7$

$P_{even}(W^{2k})=$

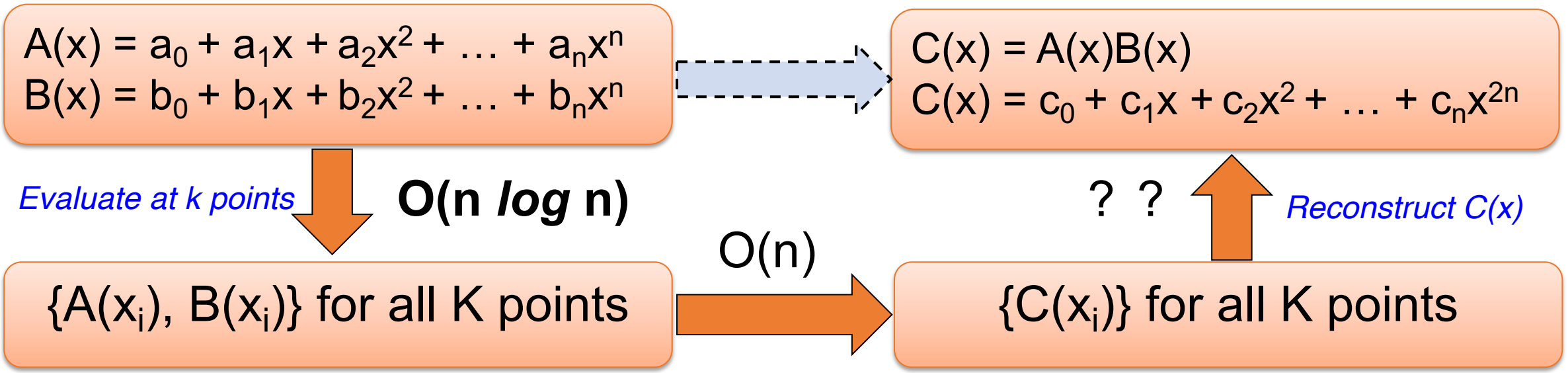
$p_0 * e^{\frac{2\pi ik}{8}*0}$	+	=	$p_0 * e^{\frac{2\pi ik}{4}*0}$	+	
$p_2 * e^{\frac{2\pi ik}{8}*2}$	+		$p_2 * e^{\frac{2\pi ik}{4}*1}$	+	Can further divide into two 2-point FFT problem
$p_4 * e^{\frac{2\pi ik}{8}*4}$	+		$p_4 * e^{\frac{2\pi ik}{4}*2}$	+	
$p_6 * e^{\frac{2\pi ik}{8}*6}$			$p_6 * e^{\frac{2\pi ik}{4}*3}$		

- To evaluate $P_{even}(W^{2k})$, do FFT for polynomial $[p_0, p_2, p_4, p_6]$
  - 8-point FFT evaluate at $\{e^{\frac{2\pi ik}{8}}\}, k = 0, 1, 2, \ldots, 7$

→For even term, it is a 4-point FFT, evaluate at $\{e^{\frac{2\pi ik}{4}}\}, k = 0, 1, 2, 3$

How about odd term polynomial?

FFT for Polynomial Multiplication



Pointwise Multiplication: $C(x_i) = A(x_i)B(x_i)$

How to generate the polynomial based on the values?

Interpolation (Value representation to Coefficients of P)

- Evaluation and interpolation are closely connected
- An alternative perspective of the evaluation of a polynomial P at n points:

$$P(x_0) = p_0 + p_1 x_0 + p_2 {x_0}^2 + \cdots + p_{n-1} {x_0}^{n-1}$$

$$P(x_1) = p_0 + p_1 x_1 + p_2 {x_1}^2 + \cdots + p_{n-1} {x_1}^{n-1}$$

$$\cdots\cdots \qquad \cdots\cdots \qquad \cdots\cdots$$

$$P(x_n) = p_0 + p_1 x_{n-1} + p_2 {x_{n-1}}^2 + \cdots + p_{n-1} {x_{n-1}}^{n-1}$$

$$\begin{bmatrix} P(x_0) \\ P(x_1) \\ \cdots \\ P(x_{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \cdots \\ p_{n-1} \end{bmatrix}$$

Here $x_k = W^k, W = e^{\frac{2\pi i}{n}}$

Interpolation (Value representation to Coefficients of P)

- Evaluation and interpolation are closely connected
- An alternative perspective of the evaluation of a polynomial P at n points:

$$P(x_0) = p_0 + p_1 x_0 + p_2 {x_0}^2 + \cdots + p_{n-1} {x_0}^{n-1}$$

$$P(x_1) = p_0 + p_1 x_1 + p_2 {x_1}^2 + \cdots + p_{n-1} {x_1}^{n-1}$$

$$\cdots\cdots \qquad \cdots\cdots \qquad \cdots\cdots$$

$$P(x_n) = p_0 + p_1 x_{n-1} + p_2 {x_{n-1}}^2 + \cdots + p_{n-1} {x_{n-1}}^{n-1}$$

$$\begin{bmatrix} P(W^0) \\ P(W^1) \\ \cdots \\ P(W^{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W & W^2 & \cdots & W^{n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & W^{n-1} & W^{2(n-1)} & \cdots & W^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \cdots \\ p_{n-1} \end{bmatrix}$$

(This is Discrete Fourier Transform matrix (DFT))

After substituting all the $x_k = W^k, W = e^{\frac{2\pi i}{n}}$

Interpolation (Value representation to Coefficients of P)

- Interpolation involves inversing the DFT matrix:

$$\begin{bmatrix} P(W^0) \\ P(W^1) \\ \dots \\ P(W^{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W & W^2 & \dots & W^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & W^{n-1} & W^{n-2} & \dots & W^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \dots \\ p_{n-1} \end{bmatrix}$$

Interpolation

$$\begin{bmatrix} p_0 \\ p_1 \\ \dots \\ p_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W & W^2 & \dots & W^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & W^{n-1} & W^{2(n-1)} & \dots & W^{(n-1)(n-1)} \end{bmatrix}^{-1} \begin{bmatrix} P(W^0) \\ P(W^1) \\ \dots \\ P(W^{n-1}) \end{bmatrix}$$

Inverse Matrix of DFT

Interpolation (Value representation to Coefficients of P)

- Interpolation involves inversing the DFT matrix:

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W & W^2 & \dots & W^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & W^{n-1} & W^{n-2} & \dots & W^{(n-1)(n-1)} \end{bmatrix}^{-1} = \frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W^{-1} & W^{-2} & \dots & W^{-(n-1)} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & W^{-(n-1)} & W^{-2(n-1)} & \dots & W^{-(n-1)(n-1)} \end{bmatrix}$$

$$\text{Interpolation} \begin{bmatrix} p_0 \\ p_1 \\ \dots \\ p_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W & W^2 & \dots & W^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & W^{n-1} & W^{2(n-1)} & \dots & W^{(n-1)(n-1)} \end{bmatrix}^{-1} \begin{bmatrix} P(W^0) \\ P(W^1) \\ \dots \\ P(W^{n-1}) \end{bmatrix}$$

Inverse Matrix of DFT

Interpolation (Value representation to Coefficients of P)

- Evaluation (FFT)

$$W = e^{\frac{2\pi i}{n}}$$

$$\begin{bmatrix} P(W^0) \\ P(W^1) \\ \dots \\ P(W^{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W & W^2 & \dots & W^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & W^{n-1} & W^{2(n-1)} & \dots & W^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \dots \\ p_{n-1} \end{bmatrix}$$

$$\mathbf{FFT}(p_0, p_1, \dots, p_{n-1})$$

- Interpolation (Inverse FFT, from values to coeffs)

$$\begin{bmatrix} p_0 \\ p_1 \\ \dots \\ p_{n-1} \end{bmatrix} = \frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W^{-1} & W^{-2} & \dots & W^{-(n-1)} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & W^{-(n-1)} & W^{-2(n-1)} & \dots & W^{-(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} P(W^0) \\ P(W^1) \\ \dots \\ P(W^{n-1}) \end{bmatrix}$$

$$\mathbf{IFFT}(P(W'^0), P(W'^1), \dots, P(W'^{n-1}))$$

If we let $W' = \frac{1}{n} e^{\frac{-2\pi i}{n}}$, it will have the exactly same format as FFT!

Recap

Evaluation of an n degree polynomial $P(x)$ at points $\{\pm x_1, \pm x_2, \dots, \pm x_{n/2}\}$

Evaluate $P_{even}(x^2) = p_0 + p_2 x^2 + \cdots + p_n x^n$ at points $\{x_1^2, x_2^2, \dots, x_{n/2}^2\}$

Evaluate $P_{even}(x^2) = p_1 + p_3 x^2 + \cdots + p_{n-1} x^{n-2}$ at points $\{x_1^2, x_2^2, \dots, x_{n/2}^2\}$

$$P(W^k) = P_{even}(W^{2k}) + W^k P_{odd}(W^{2k})$$
$$P(-W^k) = P_{even}(W^{2k}) - W^k P_{odd}(W^{2k})$$
$$k \in \{0, 1, \dots, (n/2 - 1)\}$$