CS 170: Algorithms

CS 170: Algorithms

S

CS 170: Algorithms

H

S

CS 170: Algorithms

H

H

S

CS 170: Algorithms

H

H H

S

CS 170: Algorithms

H

H H

S H

CS 170: Algorithms

H

H H

S H

.

CS 170: Algorithms

H

H H

S H

CS 170: Algorithms

H

H H

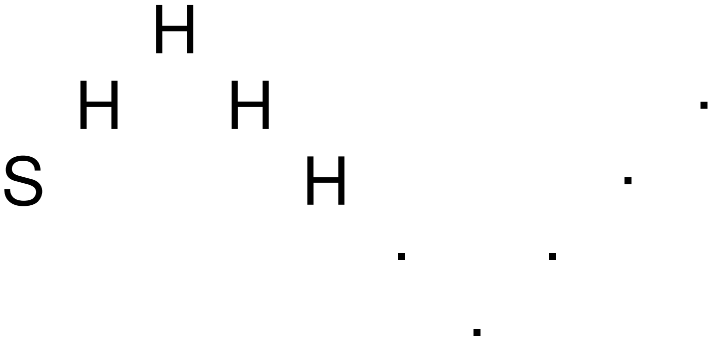
S H

CS 170: Algorithms

H

H H

S H


H
H H
S H

CS 170: Algorithms

H

H H

S H

No laptops please.

CS 170: Algorithms

H

H H

S H

No laptops please.

Thank you

CS 170: Algorithms

H

H H

S H

No laptops please.

Thank you !

CS 170: Algorithms

H

H H

S H

No laptops please.

Thank you ! !

CS 170: Algorithms

H

H H

S H

No laptops please.

Thank you ! ! !

CS 170: Algorithms

H

H H

S H

No laptops please.

Thank you ! ! ! !

CS 170: Algorithms

H

H H

S H

No laptops please.

Thank you ! ! ! ! !

CS 170: Algorithms

H

H H

S H

No laptops please.

Thank you ! ! ! ! ! !

CS 170: Algorithms

H

H H

S H

No laptops please.

Thank you ! ! ! ! ! ! !

CS 170: Algorithms

H

H H

S H

No laptops please.

Thank you ! ! ! ! ! ! ! !

Convolution = polynomial multiplication.

System: $a_0, \ldots, a_{n-1}$

Convolution $=$ polynomial multiplication.

System: $a_0, \ldots, a_{n-1}$

Signal: $b_0, \ldots, b_{n-1}$

Convolution $=$ polynomial multiplication.

System: $a_0, \ldots, a_{n-1}$

Signal: $b_0, \ldots, b_{n-1}$

At time 0:

Convolution $=$ polynomial multiplication.

System: $a_0, \ldots, a_{n-1}$

Signal: $b_0, \ldots, b_{n-1}$

At time 0:

$b_0 a_0$

Convolution = polynomial multiplication.

System: $a_0, \ldots, a_{n-1}$

Signal: $b_0, \ldots, b_{n-1}$

At time 0:

$b_0 a_0$

At time 1:

Convolution = polynomial multiplication.

System: $a_0, \ldots, a_{n-1}$

Signal: $b_0, \ldots, b_{n-1}$

At time 0:

$b_0 a_0$

At time 1:

$b_0 a_1 + b_1 a_0$

Convolution = polynomial multiplication.

System: $a_0, \ldots, a_{n-1}$

Signal: $b_0, \ldots, b_{n-1}$

At time 0:

$b_0 a_0$

At time 1:

$b_0 a_1 + b_1 a_0$

At time i:

Convolution = polynomial multiplication.

System: $a_0, \ldots, a_{n-1}$

Signal: $b_0, \ldots, b_{n-1}$

At time 0:

$b_0 a_0$

At time 1:

$b_0 a_1 + b_1 a_0$

At time i:

$b_0 a_i + b_1 a_{i-1} + \cdots b_i a_0$

Convolution = polynomial multiplication.

System: $a_0, \ldots, a_{n-1}$

Signal: $b_0, \ldots, b_{n-1}$

At time 0:

$b_0 a_0$

At time 1:

$b_0 a_1 + b_1 a_0$

At time i:

$b_0 a_i + b_1 a_{i-1} + \cdots b_i a_0$

Response: $c_0, \ldots, c_{2n-1}$

Convolution = polynomial multiplication.

System: $a_0, \ldots, a_{n-1}$

Signal: $b_0, \ldots, b_{n-1}$

At time 0:

$b_0 a_0$

At time 1:

$b_0 a_1 + b_1 a_0$

At time i:

$b_0 a_i + b_1 a_{i-1} + \cdots b_i a_0$

Response: $c_0, \ldots, c_{2n-1}$

$c_i = \sum_j b_j a_{i-j}$

Convolution = polynomial multiplication.

System: $a_0, \ldots, a_{n-1}$

Signal: $b_0, \ldots, b_{n-1}$

At time 0:

$b_0 a_0$

At time 1:

$b_0 a_1 + b_1 a_0$

At time i:

$b_0 a_i + b_1 a_{i-1} + \cdots b_i a_0$

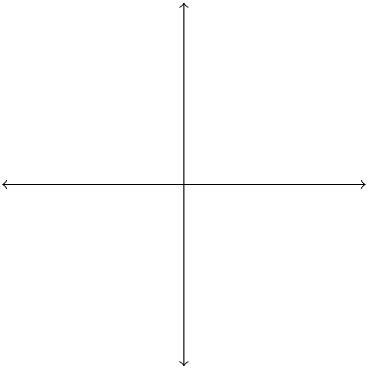
Response: $c_0, \ldots, c_{2n-1}$

$c_i = \sum_j b_j a_{i-j}$

Same as multiplying polynomials $A(\cdot)$ and $B(\cdot)$!

The nth complex roots of unity.

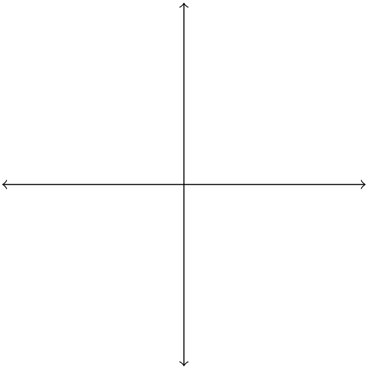
Solutions to $z^n = 1$



The nth complex roots of unity.

Solutions to $z^n = 1$

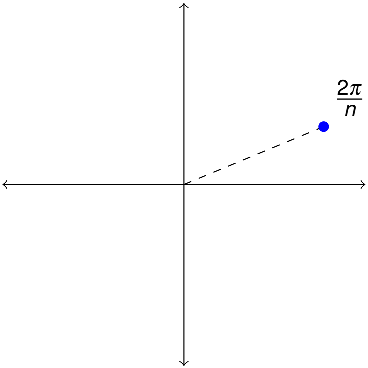
$$(1, \tfrac{2\pi}{n})^n = (1, \tfrac{2\pi}{n} \times n) = (1, 2\pi) = 1!$$



The nth complex roots of unity.

Solutions to $z^n = 1$

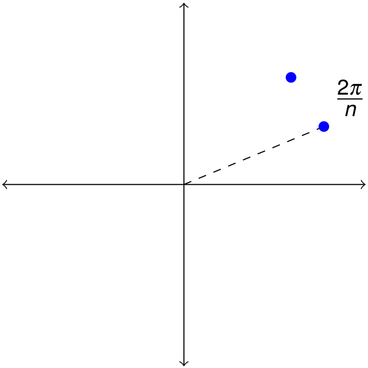
$$(1, \tfrac{2\pi}{n})^n = (1, \tfrac{2\pi}{n} \times n) = (1, 2\pi) = 1!$$



The nth complex roots of unity.

Solutions to $z^n = 1$

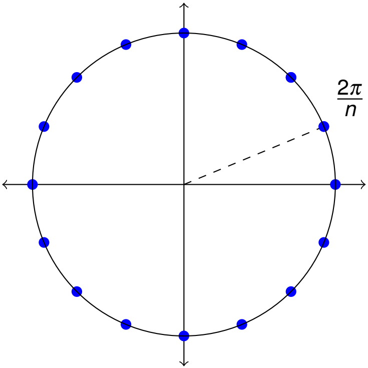
$$(1, \tfrac{4\pi}{n})^n = (1, \tfrac{4\pi}{n} \times n) = (1, 4\pi) = 1!$$



The nth complex roots of unity.

Solutions to $z^n = 1$

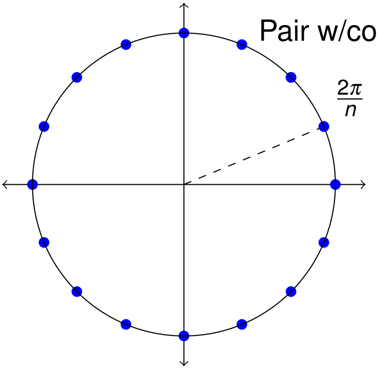
$$(1, \tfrac{2k\pi}{n})^n = (1, \tfrac{2k\pi}{n} \times n) = (1, 2k\pi) = 1!$$



The nth complex roots of unity.

Solutions to $z^n = 1$

$$(1, \theta + \pi)^2 = (1, 2\theta + 2\pi) = \boxed{(1, 2\theta)} = (1, \theta)^2.$$

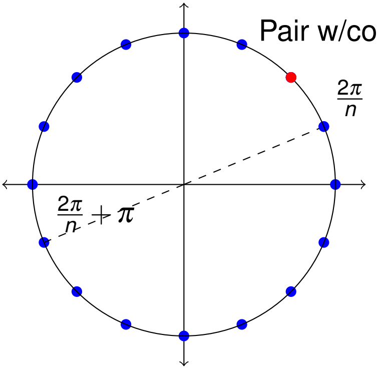


Pair w/common square!

The nth complex roots of unity.

Solutions to $z^n = 1$

$$(1, \theta + \pi)^2 = (1, 2\theta + 2\pi) = \boxed{(1, 2\theta)} = (1, \theta)^2.$$

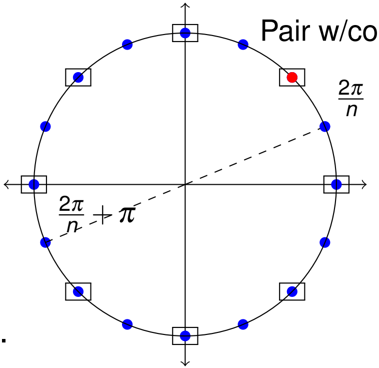


Pair w/common square!

The *n*th complex roots of unity.

Solutions to $z^n = 1$

$$(1, \theta+\pi)^2 = (1, 2\theta+2\pi) = \boxed{(1, 2\theta)} = (1, \theta)^2.$$



Squares: $\frac{n}{2}$th roots.

The Fast Fourier Transform!

Defn: $\omega = (1, \frac{2\pi}{n}) = e^{\frac{2\pi}{n} i}$, nth root of unity.

The Fast Fourier Transform!

Defn: $\omega = (1, \frac{2\pi}{n}) = e^{\frac{2\pi}{n} i}$, nth root of unity.
Pairs: ω^i and $\omega^{i+n/2}$

The Fast Fourier Transform!

Defn: $\omega = (1, \frac{2\pi}{n}) = e^{\frac{2\pi}{n} i}$, nth root of unity.
Pairs: ω^i and $\omega^{i+n/2} = \omega^i \omega^{n/2}$

The Fast Fourier Transform!

Defn: $\omega = (1, \frac{2\pi}{n}) = e^{\frac{2\pi}{n} i}$, nth root of unity.
Pairs: ω^i and $\omega^{i+n/2} = \omega^i \omega^{n/2} = -\omega^i$.

The Fast Fourier Transform!

Defn: $\omega = (1, \frac{2\pi}{n}) = e^{\frac{2\pi}{n} i}$, nth root of unity.
Pairs: ω^i and $\omega^{i+n/2} = \omega^i \omega^{n/2} = -\omega^i$. Common square.

The Fast Fourier Transform!

Defn: $\omega = (1, \frac{2\pi}{n}) = e^{\frac{2\pi}{n} i}$, nth root of unity.
Pairs: ω^i and $\omega^{i+n/2} = \omega^i \omega^{n/2} = -\omega^i$. Common square.
Common Squares: are $\frac{n}{2}$ root of unity.

The Fast Fourier Transform!

Defn: $\omega = (1, \frac{2\pi}{n}) = e^{\frac{2\pi}{n} i}$, nth root of unity.
Pairs: ω^i and $\omega^{i+n/2} = \omega^i \omega^{n/2} = -\omega^i$. Common square.
Common Squares: are $\frac{n}{2}$ root of unity.

Fast Fourier Transform:

The Fast Fourier Transform!

Defn: $\omega = (1, \frac{2\pi}{n}) = e^{\frac{2\pi}{n} i}$, nth root of unity.
Pairs: ω^i and $\omega^{i+n/2} = \omega^i \omega^{n/2} = -\omega^i$. Common square.
Common Squares: are $\frac{n}{2}$ root of unity.

Fast Fourier Transform:
Evaluate $A(x) = a_0 + a_1 x + a_2 x^2 + \cdots a_{n-1} x^{n-1}$

The Fast Fourier Transform!

Defn: $\omega = (1, \frac{2\pi}{n}) = e^{\frac{2\pi}{n} i}$, nth root of unity.
Pairs: ω^i and $\omega^{i+n/2} = \omega^i \omega^{n/2} = -\omega^i$. Common square.
Common Squares: are $\frac{n}{2}$ root of unity.

Fast Fourier Transform:
Evaluate $A(x) = a_0 + a_1 x + a_2 x^2 + \cdots a_{n-1} x^{n-1}$
on $\omega^0, \omega, \omega^2, \ldots, \omega^{n-1}$.

The Fast Fourier Transform!

Defn: $\omega = (1, \frac{2\pi}{n}) = e^{\frac{2\pi}{n} i}$, nth root of unity.
Pairs: ω^i and $\omega^{i+n/2} = \omega^i \omega^{n/2} = -\omega^i$. Common square.
Common Squares: are $\frac{n}{2}$ root of unity.

Fast Fourier Transform:
Evaluate $A(x) = a_0 + a_1 x + a_2 x^2 + \cdots a_{n-1} x^{n-1}$
on $\omega^0, \omega, \omega^2, \ldots, \omega^{n-1}$.

Procedure:

The Fast Fourier Transform!

Defn: $\omega = (1, \frac{2\pi}{n}) = e^{\frac{2\pi}{n} i}$, nth root of unity.
Pairs: ω^i and $\omega^{i+n/2} = \omega^i \omega^{n/2} = -\omega^i$. Common square.
Common Squares: are $\frac{n}{2}$ root of unity.

Fast Fourier Transform:
Evaluate $A(x) = a_0 + a_1 x + a_2 x^2 + \cdots a_{n-1} x^{n-1}$
on $\omega^0, \omega, \omega^2, \ldots, \omega^{n-1}$.

Procedure:
Recursively compute A_e and A_o on $\frac{n}{2}$ roots of unity:

The Fast Fourier Transform!

Defn: $\omega = (1, \frac{2\pi}{n}) = e^{\frac{2\pi}{n} i}$, nth root of unity.
Pairs: ω^i and $\omega^{i+n/2} = \omega^i \omega^{n/2} = -\omega^i$. Common square.
Common Squares: are $\frac{n}{2}$ root of unity.

Fast Fourier Transform:
Evaluate $A(x) = a_0 + a_1 x + a_2 x^2 + \cdots a_{n-1} x^{n-1}$
on $\omega^0, \omega, \omega^2, \ldots, \omega^{n-1}$.

Procedure:
Recursively compute A_e and A_o on $\frac{n}{2}$ roots of unity:
$\omega^2, \omega^4, \omega^6, \ldots, \omega^n$.

The Fast Fourier Transform!

Defn: $\omega = (1, \frac{2\pi}{n}) = e^{\frac{2\pi}{n} i}$, nth root of unity.
Pairs: ω^i and $\omega^{i+n/2} = \omega^i \omega^{n/2} = -\omega^i$. Common square.
Common Squares: are $\frac{n}{2}$ root of unity.

Fast Fourier Transform:
Evaluate $A(x) = a_0 + a_1 x + a_2 x^2 + \cdots a_{n-1} x^{n-1}$
on $\omega^0, \omega, \omega^2, \ldots, \omega^{n-1}$.

Procedure:
Recursively compute A_e and A_o on $\frac{n}{2}$ roots of unity:
$\omega^2, \omega^4, \omega^6, \ldots, \omega^n$.

For each $i \leq n/2$.

The Fast Fourier Transform!

Defn: $\omega = (1, \frac{2\pi}{n}) = e^{\frac{2\pi}{n} i}$, nth root of unity.
Pairs: ω^i and $\omega^{i+n/2} = \omega^i \omega^{n/2} = -\omega^i$. Common square.
Common Squares: are $\frac{n}{2}$ root of unity.

Fast Fourier Transform:
Evaluate $A(x) = a_0 + a_1 x + a_2 x^2 + \cdots a_{n-1} x^{n-1}$
on $\omega^0, \omega, \omega^2, \ldots, \omega^{n-1}$.

Procedure:
Recursively compute A_e and A_o on $\frac{n}{2}$ roots of unity:
$\omega^2, \omega^4, \omega^6, \ldots, \omega^n$.

For each $i \leq n/2$.

$$A(\omega^i) = A_e(\omega^{2i}) + \omega^i A_o(\omega^{2i})$$

The Fast Fourier Transform!

Defn: $\omega = (1, \frac{2\pi}{n}) = e^{\frac{2\pi}{n}i}$, nth root of unity.
Pairs: ω^i and $\omega^{i+n/2} = \omega^i\omega^{n/2} = -\omega^i$. Common square.
Common Squares: are $\frac{n}{2}$ root of unity.

Fast Fourier Transform:
Evaluate $A(x) = a_0 + a_1x + a_2x^2 + \cdots a_{n-1}x^{n-1}$
on $\omega^0, \omega, \omega^2, \ldots, \omega^{n-1}$.

Procedure:
Recursively compute A_e and A_o on $\frac{n}{2}$ roots of unity:
$\omega^2, \omega^4, \omega^6, \ldots, \omega^n$.

For each $i \leq n/2$.

$$A(\omega^i) = A_e(\omega^{2i}) + \omega^i A_o(\omega^{2i})$$

$$A(\omega^{i+n/2}) = A_e(\omega^{2i}) - \omega^i A_o(\omega^{2i})$$

The Fast Fourier Transform!

Defn: $\omega = (1, \frac{2\pi}{n}) = e^{\frac{2\pi}{n} i}$, nth root of unity.
Pairs: ω^i and $\omega^{i+n/2} = \omega^i \omega^{n/2} = -\omega^i$. Common square.
Common Squares: are $\frac{n}{2}$ root of unity.

Fast Fourier Transform:
Evaluate $A(x) = a_0 + a_1 x + a_2 x^2 + \cdots a_{n-1} x^{n-1}$
on $\omega^0, \omega, \omega^2, \ldots, \omega^{n-1}$.

Procedure:
Recursively compute A_e and A_o on $\frac{n}{2}$ roots of unity:
$\omega^2, \omega^4, \omega^6, \ldots, \omega^n$.

For each $i \leq n/2$.

$$A(\omega^i) = A_e(\omega^{2i}) + \omega^i A_o(\omega^{2i})$$

$$A(\omega^{i+n/2}) = A_e(\omega^{2i}) - \omega^i A_o(\omega^{2i})$$

Runtime Recurrence:

The Fast Fourier Transform!

Defn: $\omega = (1, \frac{2\pi}{n}) = e^{\frac{2\pi}{n} i}$, nth root of unity.
Pairs: ω^i and $\omega^{i+n/2} = \omega^i \omega^{n/2} = -\omega^i$. Common square.
Common Squares: are $\frac{n}{2}$ root of unity.

Fast Fourier Transform:
Evaluate $A(x) = a_0 + a_1 x + a_2 x^2 + \cdots a_{n-1} x^{n-1}$
on $\omega^0, \omega, \omega^2, \ldots, \omega^{n-1}$.

Procedure:
Recursively compute A_e and A_o on $\frac{n}{2}$ roots of unity:
$\omega^2, \omega^4, \omega^6, \ldots, \omega^n$.

For each $i \leq n/2$.

$$A(\omega^i) = A_e(\omega^{2i}) + \omega^i A_o(\omega^{2i})$$

$$A(\omega^{i+n/2}) = A_e(\omega^{2i}) - \omega^i A_o(\omega^{2i})$$

Runtime Recurrence:
$T(n) = 2T(n/2) + O(n)$

The Fast Fourier Transform!

Defn: $\omega = (1, \frac{2\pi}{n}) = e^{\frac{2\pi}{n} i}$, nth root of unity.
Pairs: ω^i and $\omega^{i+n/2} = \omega^i \omega^{n/2} = -\omega^i$. Common square.
Common Squares: are $\frac{n}{2}$ root of unity.

Fast Fourier Transform:
Evaluate $A(x) = a_0 + a_1 x + a_2 x^2 + \cdots a_{n-1} x^{n-1}$
on $\omega^0, \omega, \omega^2, \ldots, \omega^{n-1}$.

Procedure:
Recursively compute A_e and A_o on $\frac{n}{2}$ roots of unity:
$\omega^2, \omega^4, \omega^6, \ldots, \omega^n$.

For each $i \leq n/2$.

$$A(\omega^i) = A_e(\omega^{2i}) + \omega^i A_o(\omega^{2i})$$

$$A(\omega^{i+n/2}) = A_e(\omega^{2i}) - \omega^i A_o(\omega^{2i})$$

Runtime Recurrence:
$T(n) = 2T(n/2) + O(n) = O(n \log n)$!

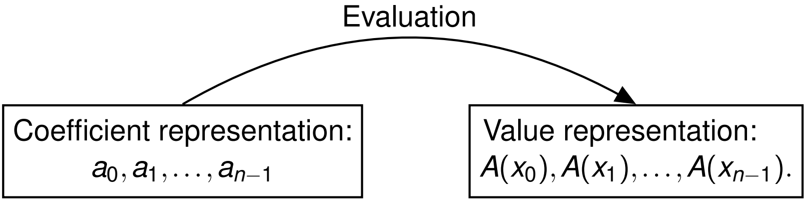
Multiplying polynomials?

Coefficient representation:
$a_0, a_1, \ldots, a_{n-1}$

Value representation:
$A(x_0), A(x_1), \ldots, A(x_{n-1})$.

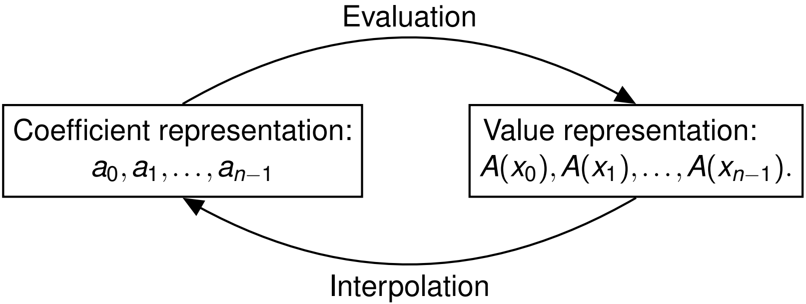
Multiplying polynomials?

Evaluation: $O(n \log n)$ if choose $1, \omega, \omega^2, \ldots, \omega^{n-1}$.



Multiplying polynomials?

Evaluation: $O(n \log n)$ if choose $1, \omega, \omega^2, \ldots, \omega^{n-1}$.



Interpolation: From points $A(x_0), \ldots, A(x_{n-1})$ to "function".

Multiplying polynomials?

Evaluation: $O(n \log n)$ if choose $1, \omega, \omega^2, \ldots, \omega^{n-1}$.

$$
\begin{tikzcd}
\boxed{\begin{array}{c}\text{Coefficient representation:}\\ a_0, a_1, \ldots, a_{n-1}\end{array}} \arrow[r, bend left, "\text{Evaluation}"] & \boxed{\begin{array}{c}\text{Value representation:}\\ A(x_0), A(x_1), \ldots, A(x_{n-1}).\end{array}} \arrow[l, bend left, "\text{Interpolation}"]
\end{tikzcd}
$$

Interpolation: From points $A(x_0), \ldots, A(x_{n-1})$ to “function”.
How?

Polynomial Evaluation and Matrices

Compute $A(\cdot)$ from a_i's:

$$\begin{bmatrix} A(x_0) \\ A(x_1) \\ \vdots \\ A(x_{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots x_1^{n-1} \\ & & \vdots & \cdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots x_{n-1}^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix}$$

Compute inverse of matrix above.

Polynomial Evaluation and Matrices

Compute $A(\cdot)$ from a_i's:

$$\begin{bmatrix} A(x_0) \\ A(x_1) \\ \vdots \\ A(x_{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots x_1^{n-1} \\ & & \vdots & \cdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots x_{n-1}^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix}$$

Compute inverse of matrix above.
Multiply.

Polynomial Evaluation and Matrices

Compute $A(\cdot)$ from a_i's:

$$\begin{bmatrix} A(x_0) \\ A(x_1) \\ \vdots \\ A(x_{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots x_1^{n-1} \\ & & \vdots & \cdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots x_{n-1}^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix}$$

Compute inverse of matrix above.
Multiply. $O(n^2)$!

Polynomial Evaluation and Matrices

Compute $A(\cdot)$ from a_i's:

$$\begin{bmatrix} A(x_0) \\ A(x_1) \\ \vdots \\ A(x_{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots x_1^{n-1} \\ & & \vdots & \cdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots x_{n-1}^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix}$$

Compute inverse of matrix above.
Multiply. $O(n^2)$!

Doh!!

Polynomial Evaluation and Matrices

Compute $A(\cdot)$ from a_i's:

$$\begin{bmatrix} A(x_0) \\ A(x_1) \\ \vdots \\ A(x_{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots x_1^{n-1} \\ & & \vdots & \cdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots x_{n-1}^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix}$$

Compute inverse of matrix above.
Multiply. $O(n^2)$!

Doh!!

Also, computing inverse not even easy.

Getting done?

FFT: ω is complex nth root of unity

Getting done?

FFT: ω is complex nth root of unity
and matrix is ...

$$M_n(\omega) = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{(n-1)} \\ & & & & \\ & & & & \\ & & & & \end{bmatrix}$$

Getting done?

FFT: ω is complex nth root of unity
and matrix is ...

$$M_n(\omega) = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{(n-1)} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix}$$

Getting done?

FFT: ω is complex nth root of unity
and matrix is ...

$$M_n(\omega) = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{(n-1)} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & & \vdots & & \\ 1 & \omega^j & \omega^{2j} & \cdots & \omega^{j(n-1)} \\ & & & & \\ & & & & \end{bmatrix}$$

Getting done?

FFT: ω is complex nth root of unity
and matrix is ...

$$M_n(\omega) = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{(n-1)} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & & \vdots & & \\ 1 & \omega^j & \omega^{2j} & \cdots & \omega^{j(n-1)} \\ \vdots & & \vdots & & \\ 1 & \omega^{(n-1)} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{bmatrix}$$

Getting done?

FFT: ω is complex nth root of unity
and matrix is ...

$$M_n(\omega) = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{(n-1)} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & & \vdots & & \\ 1 & \omega^j & \omega^{2j} & \cdots & \omega^{j(n-1)} \\ \vdots & & \vdots & & \\ 1 & \omega^{(n-1)} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{bmatrix}$$

Compute inverse of $M_n(\omega)$?

Geometry and FFT.

Each row is orthogonal.

Geometry and FFT.

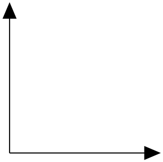
Each row is orthogonal.
Multiply by $M_n(\omega)$: project point onto each row (and scaled.)

Geometry and FFT.

Each row is orthogonal.
Multiply by $M_n(\omega)$: project point onto each row (and scaled.)
Rigid Rotation (and scaling.)!

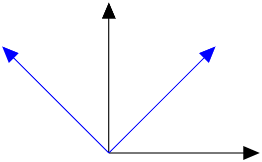
Geometry and FFT.

Each row is orthogonal.
Multiply by $M_n(\omega)$: project point onto each row (and scaled.)
Rigid Rotation (and scaling.)!



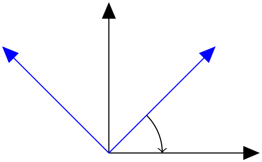
Geometry and FFT.

Each row is orthogonal.
Multiply by $M_n(\omega)$: project point onto each row (and scaled.)
Rigid Rotation (and scaling.)!



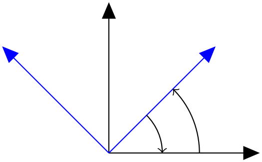
Geometry and FFT.

Each row is orthogonal.
Multiply by $M_n(\omega)$: project point onto each row (and scaled.)
Rigid Rotation (and scaling.)!



Geometry and FFT.

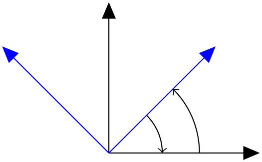
Each row is orthogonal.
Multiply by $M_n(\omega)$: project point onto each row (and scaled.)
Rigid Rotation (and scaling.)!



Reverse Rotation is inverse operation.

Geometry and FFT.

Each row is orthogonal.
Multiply by $M_n(\omega)$: project point onto each row (and scaled.)
Rigid Rotation (and scaling.)!



Reverse Rotation is inverse operation.
Scaling: for rotation, axis should have length 1, FFT length n.

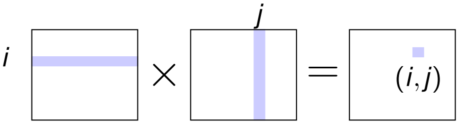
Algebraically.

Inversion formula: $(M_n(\omega))^{-1} = \frac{1}{n} M_n(\omega^{-1})$.

Algebraically.

Inversion formula: $(M_n(\omega))^{-1} = \frac{1}{n} M_n(\omega^{-1})$.

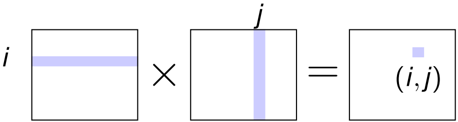
$C = M_n(\omega) \times M_n(\omega^{-1})$?



Algebraically.

Inversion formula: $(M_n(\omega))^{-1} = \frac{1}{n} M_n(\omega^{-1})$.

$C = M_n(\omega) \times M_n(\omega^{-1})$?



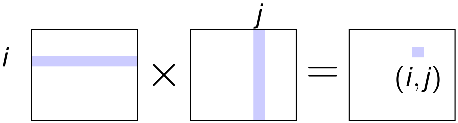
Recall: $\omega = e^{2\pi/n}$.

$$c_{ij} \;=\; \sum_k \omega^{ik} \omega^{-kj}$$

Algebraically.

Inversion formula: $(M_n(\omega))^{-1} = \frac{1}{n} M_n(\omega^{-1})$.

$C = M_n(\omega) \times M_n(\omega^{-1})$?



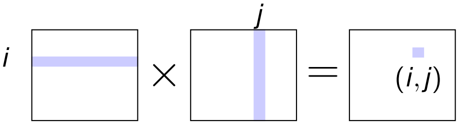
Recall: $\omega = e^{2\pi/n}$.

$$c_{ij} = \sum_k \omega^{ik} \omega^{-kj} = \sum_k \omega^{(ik-kj)}$$

Algebraically.

Inversion formula: $(M_n(\omega))^{-1} = \frac{1}{n} M_n(\omega^{-1})$.

$C = M_n(\omega) \times M_n(\omega^{-1})$?



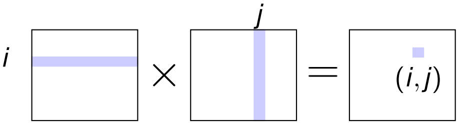
Recall: $\omega = e^{2\pi/n}$.

$$c_{ij} = \sum_k \omega^{ik} \omega^{-kj} = \sum_k \omega^{(ik-kj)} = \sum_k \omega^{k(i-j)}$$

Algebraically.

Inversion formula: $(M_n(\omega))^{-1} = \frac{1}{n} M_n(\omega^{-1})$.

$C = M_n(\omega) \times M_n(\omega^{-1})$?



Recall: $\omega = e^{2\pi/n}$.

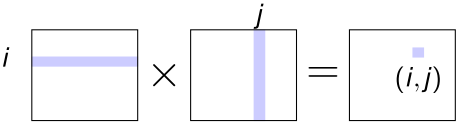
$$c_{ij} \;=\; \sum_k \omega^{ik}\omega^{-kj} = \sum_k \omega^{(ik-kj)} = \sum_k \omega^{k(i-j)}$$

If $i \neq j$, let $r = \omega^{(i-j)}$ (note: $r^n = 1$)

Algebraically.

Inversion formula: $(M_n(\omega))^{-1} = \frac{1}{n} M_n(\omega^{-1})$.

$C = M_n(\omega) \times M_n(\omega^{-1})$?



Recall: $\omega = e^{2\pi/n}$.

$$c_{ij} \quad = \quad \sum_k \omega^{ik}\omega^{-kj} = \sum_k \omega^{(ik-kj)} = \sum_k \omega^{k(i-j)}$$

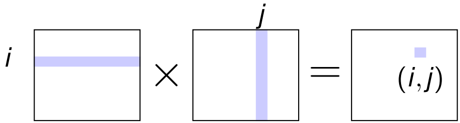
If $i \neq j$, let $r = \omega^{(i-j)}$ (note: $r^n = 1$)

$c_{ij} = 1 + r + r^2 + \cdots + r^{n-1}$

Algebraically.

Inversion formula: $(M_n(\omega))^{-1} = \frac{1}{n} M_n(\omega^{-1})$.

$C = M_n(\omega) \times M_n(\omega^{-1})$?



Recall: $\omega = e^{2\pi/n}$.

$$c_{ij} = \sum_k \omega^{ik}\omega^{-kj} = \sum_k \omega^{(ik-kj)} = \sum_k \omega^{k(i-j)}$$

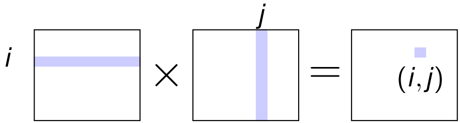
If $i \neq j$, let $r = \omega^{(i-j)}$ (note: $r^n = 1$)

$c_{ij} = 1 + r + r^2 + \cdots + r^{n-1} = \frac{1-r^n}{1-r}$

Algebraically.

Inversion formula: $(M_n(\omega))^{-1} = \frac{1}{n} M_n(\omega^{-1})$.

$C = M_n(\omega) \times M_n(\omega^{-1})$?



Recall: $\omega = e^{2\pi/n}$.

$$c_{ij} = \sum_k \omega^{ik}\omega^{-kj} = \sum_k \omega^{(ik-kj)} = \sum_k \omega^{k(i-j)}$$

If $i \neq j$, let $r = \omega^{(i-j)}$ (note: $r^n = 1$)

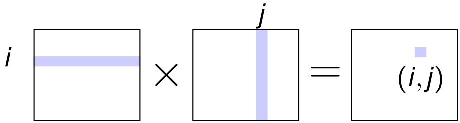
$c_{ij} = 1 + r + r^2 + \cdots + r^{n-1} = \frac{1-r^n}{1-r}$

Case $i = j$:

Algebraically.

Inversion formula: $(M_n(\omega))^{-1} = \frac{1}{n} M_n(\omega^{-1})$.

$C = M_n(\omega) \times M_n(\omega^{-1})$?



Recall: $\omega = e^{2\pi/n}$.

$$c_{ij} = \sum_k \omega^{ik}\omega^{-kj} = \sum_k \omega^{(ik-kj)} = \sum_k \omega^{k(i-j)}$$

If $i \neq j$, let $r = \omega^{(i-j)}$ (note: $r^n = 1$)

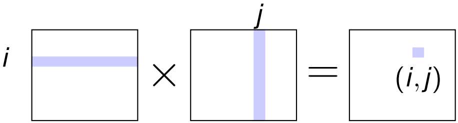
$c_{ij} = 1 + r + r^2 + \cdots + r^{n-1} = \frac{1-r^n}{1-r}$

Case $i = j$**:** $r = \omega^0 = 1$

Algebraically.

Inversion formula: $(M_n(\omega))^{-1} = \frac{1}{n} M_n(\omega^{-1})$.

$C = M_n(\omega) \times M_n(\omega^{-1})$?



Recall: $\omega = e^{2\pi/n}$.

$$c_{ij} = \sum_k \omega^{ik} \omega^{-kj} = \sum_k \omega^{(ik-kj)} = \sum_k \omega^{k(i-j)}$$

If $i \neq j$, let $r = \omega^{(i-j)}$ (note: $r^n = 1$)

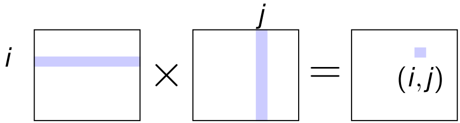
$c_{ij} = 1 + r + r^2 + \cdots + r^{n-1} = \frac{1-r^n}{1-r}$

Case $i = j$: $r = \omega^0 = 1$ and $c_{ii} = n$.

Algebraically.

Inversion formula: $(M_n(\omega))^{-1} = \frac{1}{n}M_n(\omega^{-1})$.

$C = M_n(\omega) \times M_n(\omega^{-1})$?



Recall: $\omega = e^{2\pi/n}$.

$$c_{ij} = \sum_k \omega^{ik}\omega^{-kj} = \sum_k \omega^{(ik-kj)} = \sum_k \omega^{k(i-j)}$$

If $i \neq j$, let $r = \omega^{(i-j)}$ (note: $r^n = 1$)

$c_{ij} = 1 + r + r^2 + \cdots + r^{n-1} = \frac{1-r^n}{1-r}$

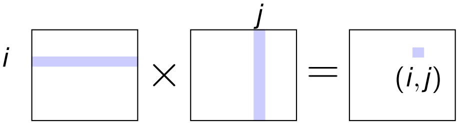
Case $i = j$**:** $r = \omega^0 = 1$ and $c_{ii} = n$.

Case $i \neq j$**:**

Algebraically.

Inversion formula: $(M_n(\omega))^{-1} = \frac{1}{n} M_n(\omega^{-1})$.

$C = M_n(\omega) \times M_n(\omega^{-1})$?



Recall: $\omega = e^{2\pi/n}$.

$$c_{ij} = \sum_k \omega^{ik}\omega^{-kj} = \sum_k \omega^{(ik-kj)} = \sum_k \omega^{k(i-j)}$$

If $i \neq j$, let $r = \omega^{(i-j)}$ (note: $r^n = 1$)

$c_{ij} = 1 + r + r^2 + \cdots + r^{n-1} = \frac{1-r^n}{1-r}$

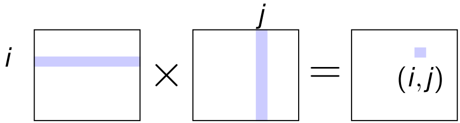
Case $i = j$**:** $r = \omega^0 = 1$ and $c_{ii} = n$.

Case $i \neq j$**:** $r^n = (\omega^{(i-j)})^n$

Algebraically.

Inversion formula: $(M_n(\omega))^{-1} = \frac{1}{n} M_n(\omega^{-1})$.

$C = M_n(\omega) \times M_n(\omega^{-1})$?



Recall: $\omega = e^{2\pi/n}$.

$$c_{ij} \quad = \quad \sum_k \omega^{ik}\omega^{-kj} = \sum_k \omega^{(ik-kj)} = \sum_k \omega^{k(i-j)}$$

If $i \neq j$, let $r = \omega^{(i-j)}$ (note: $r^n = 1$)

$c_{ij} = 1 + r + r^2 + \cdots + r^{n-1} = \frac{1-r^n}{1-r}$

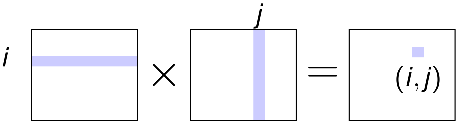
Case $i = j$**:** $r = \omega^0 = 1$ and $c_{ii} = n$.

Case $i \neq j$**:** $r^n = (\omega^{(i-j)})^n = (\omega^n)^{(i-j)} = 1^n$

Algebraically.

Inversion formula: $(M_n(\omega))^{-1} = \frac{1}{n} M_n(\omega^{-1})$.

$C = M_n(\omega) \times M_n(\omega^{-1})$?



Recall: $\omega = e^{2\pi/n}$.

$$c_{ij} = \sum_k \omega^{ik}\omega^{-kj} = \sum_k \omega^{(ik-kj)} = \sum_k \omega^{k(i-j)}$$

If $i \neq j$, let $r = \omega^{(i-j)}$ (note: $r^n = 1$)

$c_{ij} = 1 + r + r^2 + \cdots + r^{n-1} = \frac{1-r^n}{1-r}$

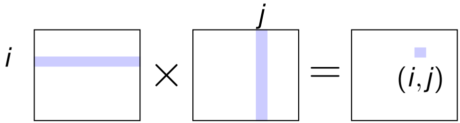
Case $i = j$**:** $r = \omega^0 = 1$ and $c_{ii} = n$.

Case $i \neq j$**:** $r^n = (\omega^{(i-j)})^n = (\omega^n)^{(i-j)} = 1^n \implies c_{ij} = 0$.

Algebraically.

Inversion formula: $(M_n(\omega))^{-1} = \frac{1}{n} M_n(\omega^{-1})$.

$C = M_n(\omega) \times M_n(\omega^{-1})$?



Recall: $\omega = e^{2\pi/n}$.

$$c_{ij} \; = \; \sum_k \omega^{ik}\omega^{-kj} = \sum_k \omega^{(ik-kj)} = \sum_k \omega^{k(i-j)}$$

If $i \neq j$, let $r = \omega^{(i-j)}$ (note: $r^n = 1$)

$c_{ij} = 1 + r + r^2 + \cdots + r^{n-1} = \frac{1-r^n}{1-r}$

Case $i = j$**:** $r = \omega^0 = 1$ and $c_{ii} = n$.

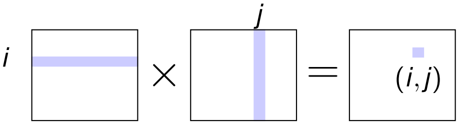
Case $i \neq j$**:** $r^n = (\omega^{(i-j)})^n = (\omega^n)^{(i-j)} = 1^n \implies c_{ij} = 0$.

For C – diagonals are n and the off-diagonals are 0.

Algebraically.

Inversion formula: $(M_n(\omega))^{-1} = \frac{1}{n} M_n(\omega^{-1})$.

$C = M_n(\omega) \times M_n(\omega^{-1})$?



Recall: $\omega = e^{2\pi/n}$.

$$c_{ij} = \sum_k \omega^{ik}\omega^{-kj} = \sum_k \omega^{(ik-kj)} = \sum_k \omega^{k(i-j)}$$

If $i \neq j$, let $r = \omega^{(i-j)}$ (note: $r^n = 1$)

$c_{ij} = 1 + r + r^2 + \cdots + r^{n-1} = \frac{1-r^n}{1-r}$

Case $i = j$**:** $r = \omega^0 = 1$ and $c_{ii} = n$.

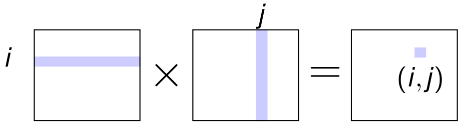
Case $i \neq j$**:** $r^n = (\omega^{(i-j)})^n = (\omega^n)^{(i-j)} = 1^n \implies c_{ij} = 0$.

For C – diagonals are n and the off-diagonals are 0.
Divide by n get identity!

Algebraically.

Inversion formula: $(M_n(\omega))^{-1} = \frac{1}{n} M_n(\omega^{-1})$.

$C = M_n(\omega) \times M_n(\omega^{-1})$?



Recall: $\omega = e^{2\pi/n}$.

$$c_{ij} \;=\; \sum_k \omega^{ik}\omega^{-kj} = \sum_k \omega^{(ik-kj)} = \sum_k \omega^{k(i-j)}$$

If $i \neq j$, let $r = \omega^{(i-j)}$ (note: $r^n = 1$)

$c_{ij} = 1 + r + r^2 + \cdots + r^{n-1} = \frac{1-r^n}{1-r}$

Case $i = j$**:** $r = \omega^0 = 1$ and $c_{ii} = n$.

Case $i \neq j$**:** $r^n = (\omega^{(i-j)})^n = (\omega^n)^{(i-j)} = 1^n \implies c_{ij} = 0$.

For C – diagonals are n and the off-diagonals are 0.
Divide by n get identity!

Inversion formula: $M_n(\omega)^{-1} = \frac{1}{n} M_n(\omega^{-1})$.

Computing inverse.

FFT works with points with basic root of unity: ω or ω^{-1}

Computing inverse.

FFT works with points with basic root of unity: ω or ω^{-1}

$1, \omega^{-1}, \omega^{-2}, \ldots, \omega^{-(n-1)}$.

Computing inverse.

FFT works with points with basic root of unity: ω or ω^{-1}

$1, \omega^{-1}, \omega^{-2}, \ldots, \omega^{-(n-1)}$.

ω^{-1} is a primitive *n*th root of unity!

Computing inverse.

FFT works with points with basic root of unity: ω or ω^{-1}

$1, \omega^{-1}, \omega^{-2}, \ldots, \omega^{-(n-1)}$.

ω^{-1} is a primitive *n*th root of unity!

Evaluation: FFT(A,ω).

Computing inverse.

FFT works with points with basic root of unity: ω or ω^{-1}

$1, \omega^{-1}, \omega^{-2}, \ldots, \omega^{-(n-1)}$.

ω^{-1} is a primitive nth root of unity!

Evaluation: FFT(A,ω).
Interpolation: $\frac{1}{n}$ FFT(A,ω^{-1}).

Computing inverse.

FFT works with points with basic root of unity: ω or ω^{-1}
$1, \omega^{-1}, \omega^{-2}, \ldots, \omega^{-(n-1)}$.
ω^{-1} is a primitive nth root of unity!

Evaluation: FFT(A,ω).
Interpolation: $\frac{1}{n}$ FFT(A,ω^{-1}).

$\implies O(n \log n)$ time for multiplying degree n polynomials.

Computing inverse.

FFT works with points with basic root of unity: ω or ω^{-1}

$1, \omega^{-1}, \omega^{-2}, \ldots, \omega^{-(n-1)}$.

ω^{-1} is a primitive nth root of unity!

Evaluation: FFT(A,ω).
Interpolation: $\frac{1}{n}$ FFT(A,ω^{-1}).

$\implies O(n \log n)$ time for multiplying degree n polynomials.

Fast convolution!

Computing inverse.

FFT works with points with basic root of unity: ω or ω^{-1}

$1, \omega^{-1}, \omega^{-2}, \ldots, \omega^{-(n-1)}$.

ω^{-1} is a primitive nth root of unity!

Evaluation: FFT(A,ω).
Interpolation: $\frac{1}{n}$ FFT(A,ω^{-1}).

$\implies O(n \log n)$ time for multiplying degree n polynomials.

Fast convolution! Digital signal processing.

Computing inverse.

FFT works with points with basic root of unity: ω or ω^{-1}

$1, \omega^{-1}, \omega^{-2}, \ldots, \omega^{-(n-1)}$.

ω^{-1} is a primitive nth root of unity!

Evaluation: FFT(A,ω).

Interpolation: $\frac{1}{n}$ FFT(A,ω^{-1}).

$\implies O(n \log n)$ time for multiplying degree n polynomials.

Fast convolution! Digital signal processing. Many other things.

Computing inverse.

FFT works with points with basic root of unity: ω or ω^{-1}
$1, \omega^{-1}, \omega^{-2}, \ldots, \omega^{-(n-1)}$.
ω^{-1} is a primitive nth root of unity!

Evaluation: FFT(A,ω).
Interpolation: $\frac{1}{n}$ FFT(A,ω^{-1}).

$\implies O(n \log n)$ time for multiplying degree n polynomials.

Fast convolution! Digital signal processing. Many other things.

Cool!!