Chapter 1
Divide and Conquer

Part 2: Polynomial Multiplication

Algorithm Theory
WS 2013/14

Fabian Kuhn

Representation of Polynomials

Coefficient representation:

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

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

- Example:

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

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

Representation of Polynomials

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

Addition:

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

- Time: $O(n)$

$(a_0 + a_1x + a_2x^2 + a_3x^3)(b_0 + b_1x + b_2x^2 + b_3x^3)$

$= a_0b_0 + (a_0b_1 + a_1b_0)x + (a_0b_2 + a_1b_1 + a_2b_0)x^2 + \underline{(a_0b_3 + a_1b_2 + a_2b_1 + a_3b_0)}x^3 + \dots$

Multiplication:

x^i

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

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

Operations Point-Value Representation

Degree-n polynomials

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

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

Addition:

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

- Time: $O(n)$

Multiplication: degr. = n → need 2n+1 points

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

- Time: $O(n)$

$\underline{O(n^{\log_3 2}) \doteq O(n^{1.59})}$

Faster Polynomial Multiplication?

Multiplication is fast when using the point-value representation

$a_0, \ldots, a_{n-1}$ $b_0, \ldots, b_{n-1}$

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

p, q of degree $n-1$, n coefficients
↓ **Evaluation** at points $x_0, x_1, \ldots, x_{2n-1}$ naive: $O(n^2)$
$2 \times 2n$ point-value pairs $(x_i, p(x_i))$ and $(x_i, q(x_i))$
↓ **Point-wise multiplication** $O(n)$ time!
$2n$ point-value pairs $(x_i, p(x_i)q(x_i))$
↓ (**Interpolation**) ??
$p(x)q(x)$ of degree $2n-2$, $2n-1$ coefficients

Point-Value Representation of p, q

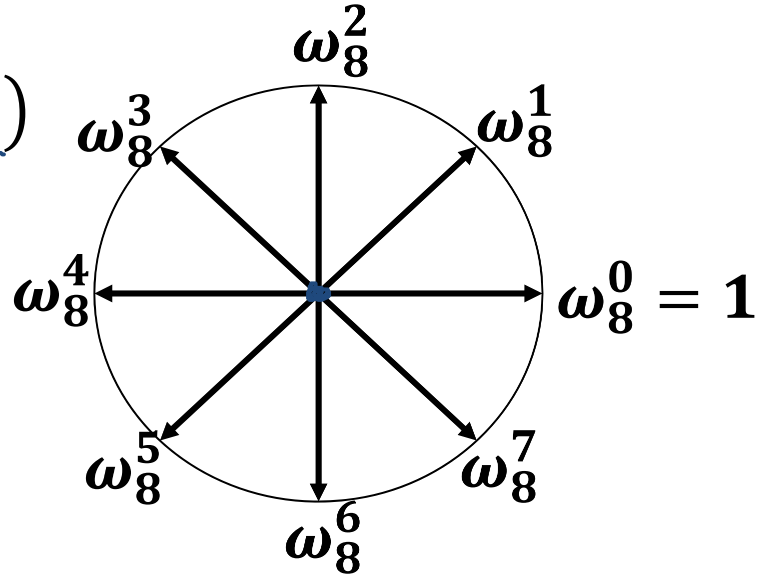
$N = 2n$

- 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 Nth root of unity:

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

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



Powers of ω_n (roots of unity):

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

$x_0, x_1, \dots, x_{N-1}$

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

Im

Re

Discrete Fourier Transform

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

$a_0 + a_1 x + \dots + a_{N-1} x^{N-1}$

Discrete Fourier Transform (DFT):

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

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

ω_N^i

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

Example

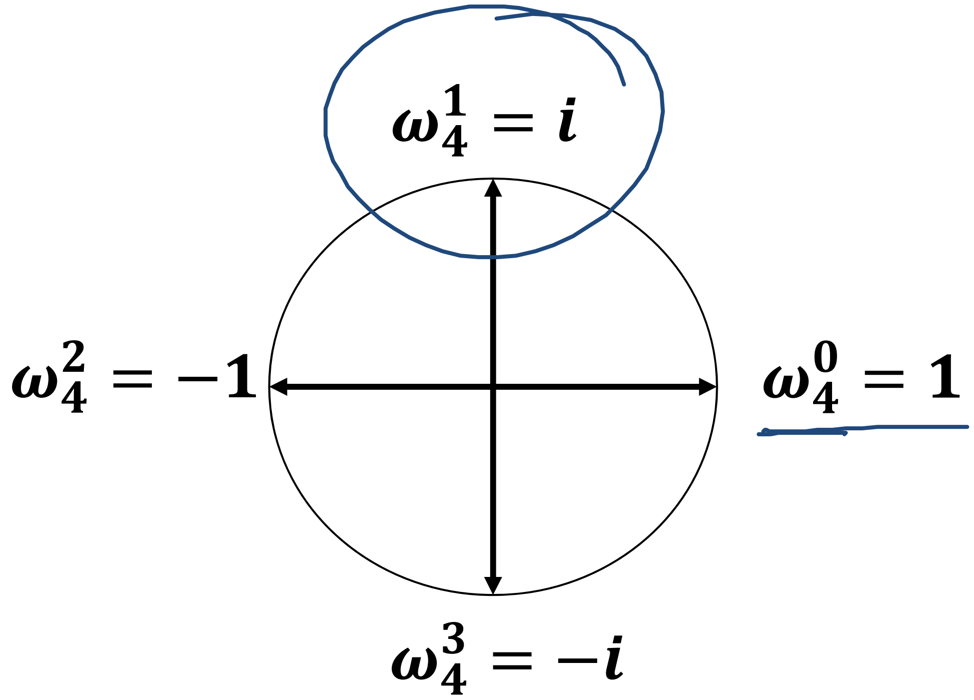
N=4

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

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

- Choose $N = 4$

- Roots of unity:



Example

$i^2 = -1$

- 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 ω_4^k:

$$\left(\omega_4^0, p(\omega_4^0)\right) = \left(1, p(1)\right) = (1, 6)$$
$$\left(\omega_4^1, p(\omega_4^1)\right) = \left(i, p(i)\right) = (i, 15 + 15i)$$
$$\left(\omega_4^2, p(\omega_4^2)\right) = \left(-1, p(-1)\right) = (-1, -36)$$
$$\left(\omega_4^3, p(\omega_4^3)\right) = \left(-i, p(-i)\right) = (-i, 15 - 15i)$$

$(0, 18, -15, 3)$

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

$$\mathbf{DFT_4}(\boldsymbol{a}) = (\mathbf{6}, \mathbf{15} + \mathbf{15}\boldsymbol{i}, -\mathbf{36}, \mathbf{15} - \mathbf{15}\boldsymbol{i})$$

Faster Polynomial Multiplication? $N = 2n$

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

$p = a_0 + \dots + a_{n-1}x^{n-1}$, $q(x) = b_0 + \dots$

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

Evaluation at points $\omega_{2n}^0, \omega_{2n}^1, \dots, \omega_{2n}^{2n-1}$ $DFT_{2n}(a)$ $DFT_{2n}(b)$

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

Point-wise multiplication

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

(**Interpolation**)

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

Properties of the Roots of Unity

- **Cancellation Lemma:**

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

$$(*)\quad \boldsymbol{\omega_{dn}^{dk} = \omega_n^k}, \qquad (**)\quad \boldsymbol{\omega_n^{k+n} = \omega_n^k}$$

- **Proof:**

$$\omega_N = e^{\frac{2\pi i}{N}}$$

$$(*)\quad \omega_{dn}^{dk} = \left(e^{\frac{2\pi i}{dn}}\right)^{dk} = e^{\frac{2\pi i \cancel{d} k}{\cancel{d} n}} = \Big(\underbrace{e^{\frac{2\pi i}{n}}}_{\omega_n}\Big)^k = \omega_n^k$$

$$(**)\quad \omega_n^{k+n} = e^{\frac{2\pi i k}{n} + \frac{2\pi i n}{n}} = e^{\frac{2\pi i k}{n}} \cdot \underbrace{e^{2\pi i}}_{1} = \left(e^{\frac{2\pi i}{n}}\right)^k = \omega_n^k$$

Divide-and-Conquer Approach

ω_N^i

- Divide $p(x)$ of degree $N-1$ (N is even) into 2 polynomials of degree $N/_2 - 1$ differently than in Karatsuba's algorithm

$p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{N-1}x^{N-1}$

- $p_0(x) = a_0 + a_2x + a_4x^2 + \cdots + a_{N-2}x^{N/_2-1}$ (even coeff.)

$p_1(x) = a_1 + a_3x + a_5x^2 + \cdots + a_{N-1}x^{N/_2-1}$ (odd coeff.)

$$p(x) = a_0 + a_2x^2 + a_4x^4 + \cdots + a_{N-2}x^{N-2}$$
$$+ a_1x + a_3x^3 + a_5x^5 + \cdots + a_{N-1}x^{N-1}$$

$$p(x) = p_0(x^2) + x \cdot p_1(x^2)$$

Discrete Fourier Transform

$p(\omega_N^k)$

$\omega_{dn}^{dk} = \omega_n^k$

$\omega_n^{k+n} = \omega_n^k$

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

$p(x) = p_0(x^2) + x p_1(x^2)$

$$\boxed{p(\omega_N^k)} = p_0\big((\omega_N^k)^2\big) + \omega_N^k \cdot p_1\big((\omega_N^k)^2\big)$$

$0, \dots, N/2 - 1$

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

$(\omega_N^k)^2 = \omega_N^{2k} = \omega_{N/2}^k = \omega_{N/2}^{k-N/2}$

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

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

Example

$p(\omega_N^k) = p_0(\omega_{N/2}^k) + \omega_N^k p_1(\omega_{N/2}^k)$

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

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

$N = 4$:

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

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

  Let $\mathrm{DFT}_N(p) \coloneqq \mathrm{DFT}_N(a)$

Recursive structure:

- For $N = 4$:

$$\big(\mathrm{DFT}_4(p)\big)_k = p(\omega_4^{\mathrm{k}})$$
$$= \big(\mathrm{DFT}_2(p_0)\big)_{k \bmod 2} + \omega_4^k \cdot \big(\mathrm{DFT}_2(p_1)\big)_{k \bmod 2}$$

- General N (assume N is even):

$$\big(\mathrm{DFT}_{\mathrm{N}}(p)\big)_k = p(\omega_{\mathrm{N}}^{\mathrm{k}})$$
$$= \big(\mathrm{DFT}_{\mathrm{N}/2}(p_0)\big)_{k \bmod N/2} + \omega_N^k \cdot \big(\mathrm{DFT}_{\mathrm{N}/2}(p_1)\big)_{k \bmod N/2}$$

Computation of DFT_N

- Divide-and-conquer algorithm for $\mathrm{DFT}_N(p)$:

1. Divide

$N \leq 1$: $\mathrm{DFT}_1(p) = a_0$

$O(N)$

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

2. Conquer

Solve $\mathrm{DFT}_{N/2}(p_0)$ and $\mathrm{DFT}_{N/2}(p_1)$ recursively

3. Combine

Compute $\mathrm{DFT}_N(p)$ based on $\mathrm{DFT}_{N/2}(p_0)$ and $\mathrm{DFT}_{N/2}(p_1)$ $O(N)$

Analysis

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

$$\underline{T(N)} = \underline{2T\left(N/_2\right)} + \underline{O(N)}, \qquad T(1) = O(1)$$

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

$$\underline{T(N) = O(N \cdot \log N)}$$

Small Improvement

Claim: $\omega_N^k = -\omega_N^{k-N/2}$

Polynomial p of degree $N-1$:

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

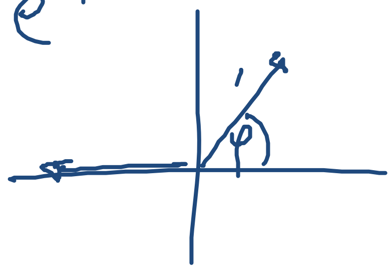
$$= \begin{cases} p_0\left(\omega_{N/2}^k\right) + \omega_N^k \cdot p_1\left(\omega_{N/2}^k\right) & \text{if } k < N/2 \\ p_0\left(\omega_{N/2}^{k-N/2}\right) \underbrace{- \omega_N^{k-N/2}}_{+\omega_N^k} \cdot p_1\left(\omega_{N/2}^{k-N/2}\right) & \text{if } k \geq N/2 \end{cases}$$

$e^{i\varphi}$

$k = \frac{N}{2} + 1$

$$\omega_N^{k-N/2} = e^{\frac{2\pi i k}{N} - \frac{2\pi i N}{2N}} = \omega_N^k \cdot e^{-\pi i} = -\omega_N^k$$

($e^{\frac{2\pi i k}{N}} = \omega_N^k$)



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

Example

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

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

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

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

Fast Fourier Transform (FFT) Algorithm

Algorithm FFT(a)

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

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

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

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

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

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

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

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

  $\omega \coloneqq \omega \cdot \omega_N$

end;

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

Example

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

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

$\omega_4^0 = 1$

$\omega_4^1 = i$

$p(x)$

$p_0(x) = -15x + 0$

$p_1(x) = 3x + 18$

$\downarrow \mathrm{DFT}_2(p_0)$

$\omega_2^0 = 1$

$\omega_2^1 = -1$

$\mathrm{DFT}_2(p_0) = (-15, 15)$

$\mathrm{DFT}_2(p_1) = (21, 15)$

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

$= -15 + 21 = +6$

$p(\omega_4^1) = p_0(\omega_2^1) + \omega_4^1 p_1(\omega_2^1) = 15 + 15i$

$p(\omega_4^2) = p_0(\omega_2^0) - \omega_4^0 p_1(\omega_2^0) = -36$

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

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** $O(n \log n)$ time

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

↓ **Point-wise multiplication** $O(n)$ time

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

↓ **Interpolation**

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

Interpolation

$p(x_i) = y_i$

Convert point-value representation into coefficient representation

Input: $(x_0, y_0), \dots, (x_{n-1}, y_{n-1})$ with $x_i \neq x_j$ for $i \neq j$

Output:

Degree-$(n-1)$ polynomial with coefficients $\boxed{a_0, \dots, a_{n-1}}$ such that

$$\begin{aligned}
p(x_0) &= a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_{n-1} x_0^{n-1} \overset{!}{=} y_0 \\
p(x_1) &= a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_{n-1} x_1^{n-1} \overset{!}{=} y_1 \\
&\ \ \vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
p(x_{n-1}) &= a_0 + a_1 x_{n-1} + a_2 x_{n-1}^2 + \cdots + a_{n-1} x_{n-1}^{n-1} \overset{!}{=} y_{n-1}
\end{aligned}$$

→ linear system of equations for $a_0, \dots, a_{n-1}$

Interpolation

$x_i = \omega_n^i$

Matrix Notation:

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

- System of equations solvable iff $x_i \neq x_j$ for all $i \neq j$

$W_{ij} = \omega_n^{ij}$

Special Case $x_i = \omega_n^i$:

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

Interpolation

- Linear system:

$$W \cdot \boldsymbol{a} = \boldsymbol{y} \quad \Longrightarrow \quad \boldsymbol{a} = W^{-1} \cdot \boldsymbol{y}$$

$$W_{i,j} = \omega_n^{ij}, \qquad \boldsymbol{a} = \begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \end{pmatrix}, \qquad \boldsymbol{y} = \begin{pmatrix} y_0 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

Claim:

$$\boldsymbol{W_{ij}^{-1} = \frac{\omega_n^{-ij}}{n}}$$

Proof: Need to show that $W^{-1}W = I_n$

DFT Matrix Inverse $(W^{-1})_{ij} = \frac{\omega_n^{-ij}}{n}$ $(W)_{ij} = \omega_n^{ij}$

$$\underline{W^{-1}W} = \underbrace{\begin{pmatrix} & & \cdots & \\ \frac{1}{n} & \frac{\omega_n^{-i}}{n} & \cdots & \frac{\omega_n^{-(n-1)i}}{n} \\ & & \vdots & \\ & & \cdots & \end{pmatrix}}_{W^{-1}} \cdot \underbrace{\begin{pmatrix} \cdots & 1 & \cdots \\ \cdots & \omega_n^{j} & \cdots \\ \cdots & \omega_n^{2j} & \cdots \\ & \vdots & \\ \cdots & \omega_n^{(n-1)j} & \cdots \end{pmatrix}}_{W}$$

$$\underline{(W^{-1}W)_{ij}} = \frac{1}{n} \cdot \sum_{k=0}^{n-1} \omega_n^{-ik} \omega_n^{jk} = \frac{1}{n} \sum_{k=0}^{n-1} \omega_n^{k(i-j)} \quad (*)$$

$i = j \rightarrow (*) = 1$

$i \neq j \rightarrow (*) = 0$

DFT Matrix Inverse

$$(W^{-1}W)_{i,j} = \sum_{\ell=0}^{n-1} \frac{\omega_n^{\ell(j-i)}}{n}$$

Need to show $(W^{-1}W)_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$

Case $\boldsymbol{i = j}$:

$$(W^{-1}W)_{ii} = \sum_{\ell=0}^{n-1} \frac{1}{n} = \underline{1} \quad \checkmark$$

DFT Matrix Inverse

$$(W^{-1}W)_{i,j} = \sum_{\ell=0}^{n-1} \frac{\omega_n^{\ell(j-i)}}{n}$$

Case $i \neq j$:

$$\omega_n^{n(j-i)} = \omega_1^{j-i} = 1$$

$$(W^{-1}W)_{i,j} = \frac{1}{n} \underbrace{\sum_{\ell=0}^{n-1} \left(\omega_n^{j-i}\right)^{\ell}}_{\text{geom. series}} = \frac{1}{n} \frac{1-\left(\omega_n^{j-i}\right)^n}{1-\omega_n^{j-i}}$$

$$= \frac{1}{n} \frac{1-1}{1-\omega_n^{j-i}} = 0$$

$$\sum_{i=0}^{n-1} q^i = \frac{1-q^n}{1-q}$$

$$W^{-1}W = I_n$$

Inverse DFT

- $$W^{-1} = \begin{pmatrix} & & \cdots & \\ \frac{1}{n} & \frac{\omega_n^{-k}}{n} & \cdots & \frac{\omega_n^{-(n-1)k}}{n} \\ & & \vdots & \\ & & \cdots & \end{pmatrix}$$

- We get $\boldsymbol{a} = W^{-1} \cdot \boldsymbol{y}$ and therefore

$$a_k = \begin{pmatrix} \frac{1}{n} & \frac{\omega_n^{-k}}{n} & \cdots & \frac{\omega_n^{-(n-1)k}}{n} \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

$$= \frac{1}{n} \cdot \sum_{j=0}^{n-1} \omega_n^{-kj} \cdot y_j$$