02 – Polynomial Multiplication and Fast Fourier Transform

Fast Fourier Transform

- FFT algorithms compute the discrete Fourier transform (DFT).

- Many applications in engineering, science and mathematics.

  Digital signal processing (e.g. UMTS and LTE)

  Image processing

  Data compression

  Partial differential equations

- Popular algorithm by J. Cooley and J.W. Tukey, 1965, based on earlier ideas of C.F. Gauß in 1805.

- Included in Top 10 Algorithms of the 20th Century by IEEE journal *Computing in Science & Engineering* (2000).

1. Polynomials

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

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

$a_0, \ldots, a_n \in R$, $a_n \neq 0$: coefficients of *p*

degree of *p*: highest power of *x* in *p* (= *n*)

Example:

$p(x) = 3x^3 – 15x^2 + 18x$

Set of all real polynomials: *R[x]*

2. Operations on polynomials

$p, q \in R[x]$

$$p(x) = a_n x^n + \ldots + a_1 x^1 + a_0$$
$$q(x) = b_n x^n + \ldots + b_1 x^1 + b_0$$

1. Addition

$$\begin{aligned} p(x) + q(x) &= (a_n x^n + \ldots + a_0) + (b_n x^n + \ldots + b_0) \\ &= (a_n + b_n) x^n + \ldots + (a_1 + b_1) x^1 + (a_0 + b_0) \end{aligned}$$

Operations on polynomials

2. Multiplication:

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

c_i : What products of monomials have degree *i* ?

$$\Rightarrow c_i = \sum_{j=0}^{i} a_j b_{i-j} \quad i = 0, \ldots, 2n.$$

$$a_{n+1} = \ldots = a_{2n} = 0, b_{n+1} = \ldots = b_{2n} = 0$$

Operations on polynomials

3. Evaluation at a specific point x_0: Horner‘s method

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

Running time: O(*n*)

3. Representation of polynomials

p(x) $\in$ *R[x]*

Possible representations of *p(x)*:

1. Coefficient representation

$$p(x) = a_n x^n + \ldots + a_1 x^1 + a_0$$

Example:

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

Representation of polynomials

2. Product of linear factors

$p(x) \in R[x]$

$$p(x) = a_n(x - x_1)\ldots(x - x_n)$$

Example:

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

Representation of polynomials

3. Point-value representation

Interpolation lemma:

Any polynomial $p(x) \in R[x]$ of degree n is uniquely defined by $n+1$ pairs $(x_i, p(x_i))$, where $i = 0,\ldots,n$ and $x_i \neq x_j$ for $i \neq j$.

Example:

The polynomial

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

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

Operations on polynomials

p, q $\in$ *R[x],* degree(*p*) = degree(*q*) = *n*

- **Coefficient representation**

  Addition: O(*n*)

  Multiplication: O(n^2)

  Evaluation at x_0 : O(*n*)

- **Point-value representation**

$$p = (x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$$

$$q = (x_0, z_0), (x_1, z_1), \ldots, (x_n, z_n)$$

Operations on polynomials

Addition:

$$p+q=(x_0,y_0+z_0),(x_1,y_1+z_1),\ldots,(x_n,y_n+z_n)$$

Running time: O(*n*)

Multiplication:

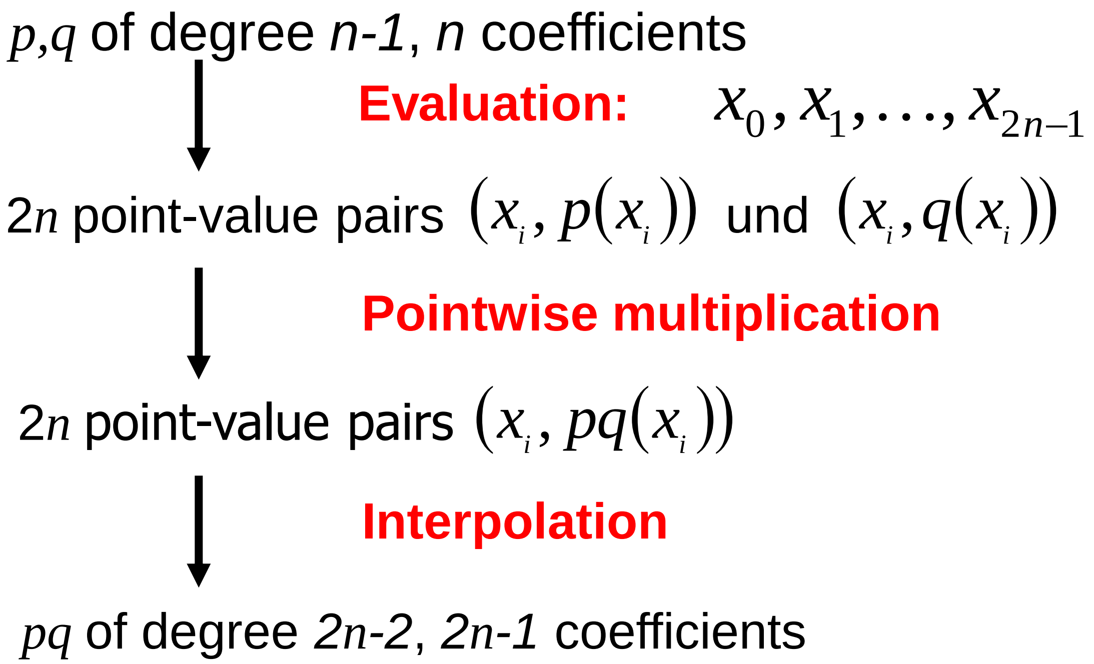
$$p\cdot q=(x_0,y_0\cdot z_0),(x_1,y_1\cdot z_1),\ldots,(x_n,y_n\cdot z_n)$$

(Condition: *n* ≥ degree(*pq*))
Running time: O(*n*)

Evaluation at point *x´:* ??
Convert polynomial to coefficient representation (interpolation)

Polynomial multiplication

Compute the product of two polynomials *p, q* of degree < *n*:



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

↓ **Evaluation:** $x_0, x_1, \ldots, x_{2n-1}$

2n point-value pairs $(x_i, p(x_i))$ und $(x_i, q(x_i))$

↓ **Pointwise multiplication**

2n point-value pairs $(x_i, pq(x_i))$

↓ **Interpolation**

pq of degree *2*n*-2*, *2*n*-1* coefficients

Divide-and-conquer approach

Idea: (assume *n* is even)

$$
\begin{aligned}
p(x) &= a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} \\
&= a_0 + a_2 x^2 + \ldots + a_{n-2} x^{n-2} + \\
&\quad a_1 x + a_3 x^3 + \ldots + a_{n-1} x^{n-1} \\
&= a_0 + a_2 x^2 + \ldots + a_{n-2} \left(x^2\right)^{(n-2)/2} + \\
&\quad x\left(a_1 + a_3 x^2 + \ldots + a_{n-1} \left(x^2\right)^{(n-2)/2}\right) \\
&= p_0\left(x^2\right) + x p_1\left(x^2\right)
\end{aligned}
$$

$$p_0(x) = a_0 + a_2 x + \ldots + a_{n-2} x^{(n-2)/2}$$
$$p_1(x) = a_1 + a_3 x + \ldots + a_{n-1} x^{(n-2)/2}$$

Select $x_0, \ldots, x_{2n-1}$ such that the computations of $p(x_k)$ and $p(x_{k+n})$ are almost identical.

Representation of *p(x)*

Assume: degree(*p*) < *n*

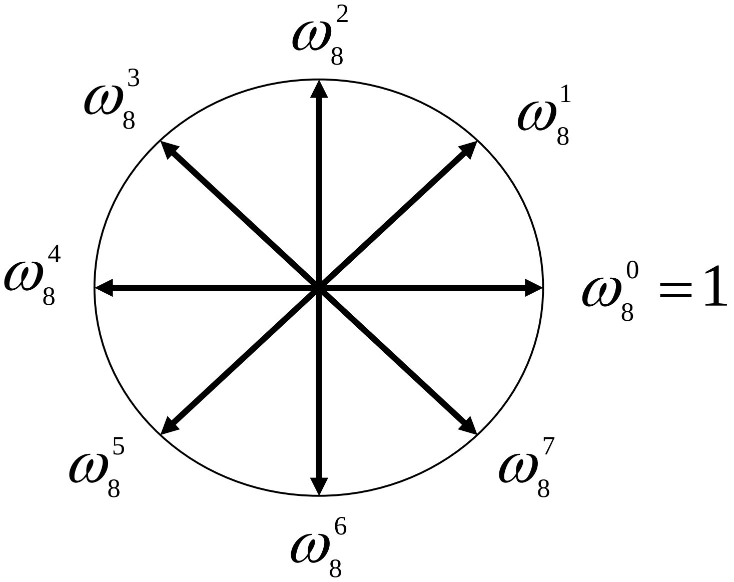
Principal *n*th root of unity

$$\omega_n = e^{2\pi i/n}$$

$i = \sqrt{-1}$ $e^{2\pi i} = 1$

Powers of ω_n (roots of unity):

1 = $\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$



Discrete Fourier Transform

The values $p(\omega_n^i)$ of the *n* powers of ω_n uniquely define *p* if degree(*p*) < *n*.

Discrete Fourier Transform (DFT)

$$DFT_n(p) = \left(p(\omega_n^0), p(\omega_n^1), \ldots, p(\omega_n^{n-1})\right)$$

Example: *n* = 4

$$e^{ix} = \cos x + i \sin x$$

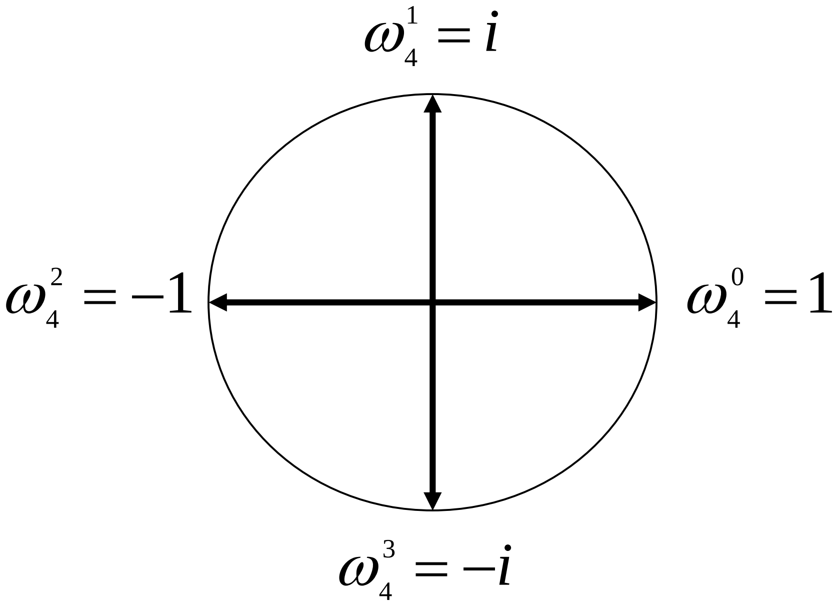
$$\omega_4^0 = e^{0i} = \cos(0) + i \sin(0) = 1$$

$$\omega_4^1 = e^{2\pi i/4} = \cos(\pi/2) + i \sin(\pi/2) = i$$

$$\omega_4^2 = \left(e^{2\pi i/4}\right)^2 = \cos \pi + i \sin \pi = -1$$

$$\omega_4^3 = \left(e^{2\pi i/4}\right)^3 = \cos(3\pi/2) + i \sin(3\pi/2) = -i$$

Evaluation at the roots of unity



$\omega_4^1 = i$

$\omega_4^2 = -1$

$\omega_4^0 = 1$

$\omega_4^3 = -i$

Evaluation at the roots of unity

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

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

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

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

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

$$DFT_4(p) = (6, 15 + 15i, -36, 15 - 15i)$$

4. Properties of the roots of unity

$\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$ form a multiplicative group

Cancellation lemma:
For any integers *n* > 0, *k* ≥ 0 and *d* > 0 we have:

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

Proof:

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

Therefore:

$$\omega_n^{n/2} = \omega_2^1 = -1$$

Polynomial multiplication

Compute the product of two polynomials *p, q* of degree < *n*:

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

↓ **Evaluation:** $\omega_{2n}^{0}, \omega_{2n}^{1}, \ldots, \omega_{2n}^{2n-1}$

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

↓ **Pointwise multiplication**

2n point-value pairs $\left(\omega_{2n}^{i}, pq\left(\omega_{2n}^{i}\right)\right)$

↓ **Interpolation**

pq of degree *2n-2*, *2n-1* coefficients

5. Discrete Fourier Transform

$$DFT_n(p) = \left(p(\omega_n^0), p(\omega_n^1), \ldots, p(\omega_n^{n-1})\right)$$

Fast Fourier Transform:

Computation of *$DFT_n(p)$* by means of a divide-and-conquer approach.

Discrete Fourier Transform

Idea: (assume *n* is even)

$$
\begin{aligned}
p(x) &= a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} \\
&= a_0 + a_2 x^2 + \ldots + a_{n-2} x^{n-2} + \\
&\quad a_1 x + a_3 x^3 + \ldots + a_{n-1} x^{n-1} \\
&= a_0 + a_2 x^2 + \ldots + a_{n-2} \left(x^2\right)^{(n-2)/2} + \\
&\quad x\left(a_1 + a_3 x^2 + \ldots + a_{n-1} \left(x^2\right)^{(n-2)/2}\right) \\
&= p_0\left(x^2\right) + x p_1\left(x^2\right)
\end{aligned}
$$

$$p_0(x) = a_0 + a_2 x + \ldots + a_{n-2} x^{(n-2)/2}$$

$$p_1(x) = a_1 + a_3 x + \ldots + a_{n-1} x^{(n-2)/2}$$

Discrete Fourier Transform

Evaluation for *k* = 0, ... , *n* – 1:

$$p(\omega_n^k) = p_0\left((\omega_n^k)^2\right) + \omega_n^k p_1\left((\omega_n^k)^2\right) = \begin{cases} p_0\left(\omega_{n/2}^k\right) + \omega_n^k p_1\left(\omega_{n/2}^k\right), \\ \text{if } k < n/2 \\ p_0\left(\omega_{n/2}^{k-n/2}\right) + \omega_n^k p_1\left(\omega_{n/2}^{k-n/2}\right), \\ \text{if } k \geq n/2 \end{cases}$$

$$\begin{aligned} DFT_n(p) &= \left(p_0(\omega_{n/2}^0), \ldots, p_0(\omega_{n/2}^{n/2-1}),\ \ p_0(\omega_{n/2}^0), \ldots, p_0(\omega_{n/2}^{n/2-1})\right) \\ &+ \left(\omega_n^0 p_1(\omega_{n/2}^0), \ldots, \omega_n^{n/2-1} p_1(\omega_{n/2}^{n/2-1}),\ \ \omega_n^{n/2} p_1(\omega_{n/2}^0), \ldots, \omega_n^{n-1} p_1(\omega_{n/2}^{n/2-1})\right) \end{aligned}$$

Discrete Fourier Transform

Example:

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

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

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

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

Computation of DFT_n

$$DFT_n(p) = \left(p(\omega_n^0), p(\omega_n^1), \ldots, p(\omega_n^{n-1})\right)$$

Base case: n = 1 (degree(p) = n -1 = 0)

$DFT_1(p) = a_0$

General case :

Divide:

Divide p into p_0 and p_1

Conquer:

Recursively compute $DFT_{n/2}(p_0)$ and $DFT_{n/2}(p_1)$.

Merge:

For k = 0, ... , n –1 compute:

$$DFT_n(p)_k = (DFT_{n/2}(p_0), DFT_{n/2}(p_0))_k + \omega_n^k \cdot (DFT_{n/2}(p_1), DFT_{n/2}(p_1))_k$$

A further improvement

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

$$= \begin{cases} p_0(\omega_{n/2}^k) + \omega_n^k p_1(\omega_{n/2}^k) & \text{if } k < n/2 \\ p_0(\omega_{n/2}^{k-n/2}) - \omega_n^{k-n/2} p_1(\omega_{n/2}^{k-n/2}) & \text{if } k \geq n/2 \end{cases}$$

Thus, for $k < n/2$:

$$p_0(\omega_{n/2}^k) + \omega_n^k p_1(\omega_{n/2}^k) = p(\omega_n^k)$$
$$p_0(\omega_{n/2}^k) - \omega_n^k p_1(\omega_{n/2}^k) = p(\omega_n^{k+n/2})$$

A further improvement

Example:

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

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

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

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

6. Fast Fourier Transform

Algorithm: *FFT*

Input: Array *a* containing the *n* coefficients of a polynomial *p* and $n = 2^k$

Output: $DFT_n(p)$

```
if $n = 1$ then  /* p is constant */
    return $a$;
$d^{[0]} := FFT([a_0, a_2, \dots, a_{n-2}], n/2)$;
$d^{[1]} := FFT([a_1, a_3, \dots, a_{n-1}], n/2)$;
$\omega_n := e^{2\pi i/n}$;
$\omega := 1$;
for $k = 0$ to $n/2 - 1$ do      /* $\omega = \omega_n^k$ */
    $d_k := d_k^{[0]} + \omega \cdot d_k^{[1]}$;
    $d_{k+n/2} := d_k^{[0]} - \omega \cdot d_k^{[1]}$;
    $\omega := \omega_n \cdot \omega$;
return $d$;
```

FFT: Example

$$p(x) = 3x^3 - 15x^2 + 18x + 0$$

a = [0, 18, -15, 3]

$a^{[0]}$ = [0, -15] $a^{[1]}$ = [18, 3]

FFT([0, -15], 2) = (FFT([0],1) + FFT([-15],1), FFT([0],1) - FFT([-15],1))
= (-15,15)

FFT([18, 3],2) = (FFT([18],1) + FFT([3],1), FFT([18],1) - FFT([3],1))
= (21,15)

k = 0 ; ω = 1

d_0 = -15 + 1 * 21 = 6 d_2 = -15 – 1 * 21 = -36

k = 1 ; ω = i

d_1 = 15 + i*15 d_3 = 15 – i*15

FFT(*a*, 4) = (6, 15+15i, -36, 15 –15i)

7. Analysis

$T(n)$ = time required for computing $DFT_n(p)$

$$T(1) = O(1)$$

$$T(n) = 2\,T(n/2) + O(n)$$

$$= O(n \log n)$$

Polynomial multiplication

Compute the product of two polynomials *p, q* of degree < *n*:

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

↓ **Evaluation via FFT:** $\omega_{2n}^{0}, \omega_{2n}^{1}, \ldots, \omega_{2n}^{2n-1}$

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

↓ **Pointwise multiplication**

2n point-value pairs $(\omega_{2n}^{i}, pq(\omega_{2n}^{i}))$

↓ **Interpolation**

pq of degree *2*n*-2*, *2*n*-1* coefficients

Interpolation

Convert the point-value representation into coefficient representation.

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

Output: Polynomial p with coefficients $a_0,\ldots, a_{n-1}$, such that

$$\begin{aligned}
p(x_0) &= a_0 + a_1 x_0 + \ldots + a_{n-1} x_0^{n-1} &= y_0 \\
p(x_1) &= a_0 + a_1 x_1 + \ldots + a_{n-1} x_1^{n-1} &= y_1 \\
p(x_2) &= a_0 + a_1 x_2 + \ldots + a_{n-1} x_2^{n-1} &= y_2 \\
&\vdots & \vdots \\
p(x_{n-1}) &= a_0 + a_1 x_{n-1} + \ldots + a_{n-1} x_{n-1}^{n-1} &= y_{n-1}
\end{aligned}$$

Interpolation

Matrix notation:

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

Interpolation

System of equations

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

solvable if $x_i \neq x_j$ for all $i \neq j$.

Special case (here) : $x_i = \omega_n^i$

Definition: $V_n = \left(\omega_n^{ij}\right)_{i,j}, \; a = (a_i), \quad y = (y_i)$

$$V_n a = y \quad \Rightarrow \quad a = V_n^{-1} y$$

Interpolation

Theorem:
For any $0 \le i, j \le n - 1$ we have:

$$\left(V_n^{-1}\right)_{ij} = \frac{\omega_n^{-ij}}{n}$$

Proof:

$$V_n^{-1} = \left(\frac{\omega_n^{-ij}}{n}\right)_{i,j}$$

We have to show:

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

Interpolation

Consider t he entry of $V_n^{-1}V_n$ in line i and column j:

$$\left(V_n^{-1}V_n\right)_{ij} =$$

$$\begin{pmatrix} & & \cdots & \\ \frac{1}{n} & \frac{\omega_n^{-i}}{n} & \cdots & \frac{\omega_n^{-i(n-1)}}{n} \\ & & \vdots & \\ & & \cdots & \end{pmatrix} \begin{pmatrix} \cdots & 1 & \cdots \\ \cdots & \omega_n^{j} & \cdots \\ \cdots & \omega_n^{2j} & \cdots \\ & \vdots & \\ \cdots & \omega_n^{(n-1)j} & \cdots \end{pmatrix}_{ij}$$

Interpolation

$$\left(V_n^{-1}V_n\right)_{ij} = \sum_{k=0}^{n-1} \frac{\omega_n^{-ik}}{n} \omega_n^{jk} = \frac{1}{n} \sum_{k=0}^{n-1} \omega_n^{(-i+j)k}$$

Case 1: *i* = *j*

$$\frac{1}{n} \sum_{k=0}^{n-1} \omega_n^{(-i+j)k} = \frac{1}{n} \sum_{k=0}^{n-1} \omega_n^{0 \cdot k} = 1$$

Case 2: *i* ≠ *j*, i.e. $-(n-1) \leq -i + j \leq n-1$

thus $n \nmid -i + j$:

$$\frac{1}{n} \sum_{k=0}^{n-1} \omega_n^{(-i+j)k} = 0$$

Interpolation

Summation lemma:

For any integer $n > 0$, $l \geq 0$ with $n \nmid l$:

$$\sum_{k=0}^{n-1} \omega_n^{lk} = 0$$

Proof:

$$\sum_{k=0}^{n-1} \left(\omega_n^l\right)^k = \frac{\left(\omega_n^l\right)^n - 1}{\omega_n^l - 1} = \frac{\left(\omega_n^n\right)^l - 1}{\omega_n^l - 1} = 0$$

Interpolation

$$a_i = \left(V_n^{-1} y\right)_i$$

$$= \left(\frac{1}{n}, \frac{\omega_n^{-i}}{n}, \ldots, \frac{\omega_n^{-i(n-1)}}{n}\right) \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

$$= \sum_{k=0}^{n-1} y_k \frac{\omega_n^{-ik}}{n}$$

$$= \frac{1}{n} \sum_{k=0}^{n-1} y_k \left(\omega_n^{-i}\right)^k$$

Interpolation

$$a = \frac{1}{n}\left(\sum_{k=0}^{n-1} y_k \left(\omega_n^{-0}\right)^k, \sum_{k=0}^{n-1} y_k \left(\omega_n^{-1}\right)^k, \cdots, \sum_{k=0}^{n-1} y_k \left(\omega_n^{-(n-1)}\right)^k\right)$$

$$r(x) = y_0 + y_1 x + y_2 x^2 + \cdots + y_{n-1} x^{n-1}$$

$$a = \frac{1}{n}\left(r\left(\omega_n^{-0}\right), r\left(\omega_n^{-1}\right), \ldots, r\left(\omega_n^{-(n-1)}\right)\right)$$

Interpolation and DFT

$$a = \frac{1}{n}\left(r\left(\omega_n^{-0}\right), r\left(\omega_n^{-1}\right), \ldots, r\left(\omega_n^{-(n-1)}\right)\right)$$

$$a = \frac{1}{n}\left(r\left(\omega_n^{n}\right), r\left(\omega_n^{n-1}\right), \ldots, r\left(\omega_n^{1}\right)\right) \quad \text{since } \omega_n^n = 1$$

$$a_i = \frac{1}{n}\left(DFT_n(r)\right)_{n-i} \qquad (i \neq 0)$$

$$a_0 = \frac{1}{n}\left(DFT_n(r)\right)_0$$

Polynomial multiplication by FFT

Compute the product of two polynomials *p, q* of degree < *n*:

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

↓ **Evaluation by FFT:** $\omega_{2n}^{0}, \omega_{2n}^{1}, \ldots, \omega_{2n}^{2n-1}$

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

↓ **Pointwise multiplication**

2n point-value pairs $\left(\omega_{2n}^{i}, pq\left(\omega_{2n}^{i}\right)\right)$

↓ **Interpolation via FFT**

pq of degree *2n-2*, *2n-1* coefficients