# Algoritmer, datastrukturer och komplexitet

- Envariabelpolynom
  - olika lagringssätt
  - addition, multiplikation
  - evaluering, interpolation
- DFT – diskret fouriertransform
- FFT – snabb fouriertransform

## Operationer på envariabelpolynom

Polynomet $A(x)=\sum_{j=0}^{n-1} a_j x^j = a_0+a_1x+\ldots+a_{n-1}x^{n-1}$

kan lagras i <u>koefficientform</u> som $\langle a_0, a_1, \ldots, a_{n-1}\rangle$

Evaluering:

Med Horners regel:

$A(x)=a_0+x(a_1+x(a_2+\ldots+x(a_{n-2}+x\cdot a_{n-1})\ldots))$

Tid: $\Theta(n)$

Addition:

$$\sum_{j=0}^{n-1} a_j x^j = \sum_{j=0}^{n-1} b_j x^j + \sum_{j=0}^{n-1} c_j x^j$$

beräknas genom $a_j = b_j + c_j$ för $0 \le j \le n-1$

Tid: $\Theta(n)$

Multiplikation:

$$\sum_{j=0}^{2n-2} a_j x^j = \left(\sum_{j=0}^{n-1} b_j x^j\right)\cdot\left(\sum_{j=0}^{n-1} c_j x^j\right)$$

beräknas genom $a_j = \sum_{k=0}^{j} b_k \cdot c_{j-k}$

Tid: $\Theta(n^2)$

## Alternativ lagringsmetod

Polynomet $A(x)=\sum_{j=0}^{n-1} a_j x^j$ kan också lagras i <u>punkt-värde-form</u> som

$\{(x_0,y_0),(x_1,y_1),\ldots,(x_{n-1},y_{n-1})\}$ där $y_j = A(x_j)$

Entydigt, ty bara ett $(n-1)$-gradspolynom går genom $n$ givna punkter [Nummekursen]

Interpolation (övergång till koefficientform):

Med Lagranges interpolationsformel

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

Tid: $\Theta(n^2)$

Addition:

$\{(x_0, y_0+y_0'),(x_1,y_1+y_1'),\ldots,(x_{n-1},y_{n-1}+y_{n-1}')\}$

Tid: $\Theta(n)$

Multiplikation:

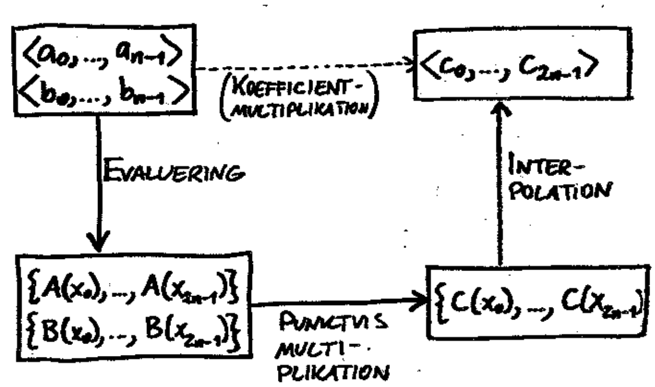
$\{(x_0, y_0\cdot y_0'),(x_1,y_1\cdot y_1'),\ldots,(x_{2n-2},y_{2n-2}\cdot y_{2n-2}')\}$

Tid: $\Theta(n)$

Problem: $2n-1$ punkter krävs.

## Snabb polynommultiplikation i koefficientform

Om man kunde <u>evaluera</u> och <u>interpolera</u> snabbt skulle man kunna <u>multiplicera</u> snabbt med hjälp av punktvis multiplikation




Om evaluering och interpolation gick att göra i tid $o(n^2)$ skulle detta vara snabbare än vanlig polynommultiplikation.

Välj evalueringspunkterna $x_0,\ldots,x_{2n-1}$ listigt!

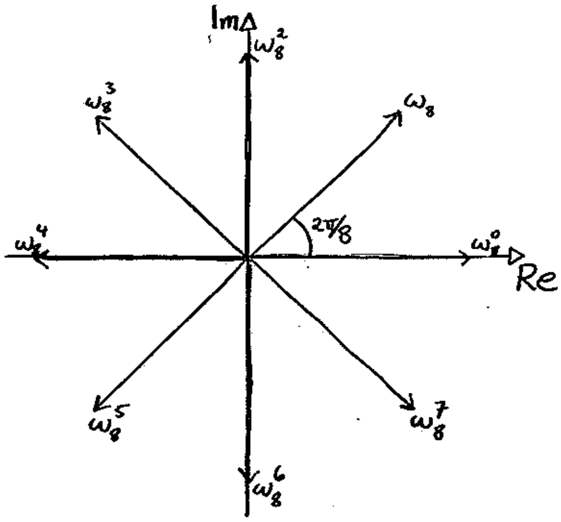
## Komplexa enhetsrötter

Välj att evaluera polynomen i komplexa enhetsrötter, dvs

$\omega$ så att $\omega^n = 1$

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

Använd $\omega_n^0, \omega_n^1, \omega_n^2, \ldots, \omega_n^{n-1}$




## DFT

Diskret fouriertransform

$$A(x) = \sum_{j=0}^{n-1} a_j x^j$$

polynom vars koefficienter är $\langle a_0, a_1, \ldots, a_{n-1} \rangle$

$$\boxed{\begin{array}{l} DFT_n(\langle a_0, a_1, \ldots, a_{n-1}\rangle) = \langle y_0, y_1, \ldots, y_{n-1}\rangle \\ \text{där } y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{jk} = \sum_{j=0}^{n-1} a_j e^{2\pi i jk/n} \end{array}}$$

$DFT_n$ transformerar en $n$-vektor till en annan $n$-vektor.

Jämför den vanliga fouriertransformen som transformerar en funktion $f(x)$ till en annan funktion $\hat{f}(t)$:

$$\hat{f}(t) = \int_{-\infty}^{\infty} f(x) e^{-itx} dx$$

## FFT – snabb beräkning av DFT

Dela upp $A(x) = \sum_{j=0}^{n-1} a_j x^j =$

$$= a_0 + a_2 x^2 + a_4 x^4 + \ldots + a_{n-2} x^{n-2} +$$

$$+ x(a_1 + a_3 x^2 + a_5 x^4 + \ldots + a_{n-1} x^{n-2}) =$$

$$= A^{[0]}(x^2) + x \cdot A^{[1]}(x^2)$$

där $A^{[0]}(x) = a_0 + a_2 x + a_4 x^2 + \ldots + a_{n-2} x^{\frac{n}{2}-1}$

$A^{[1]}(x) = a_1 + a_3 x + a_5 x^2 + \ldots + a_{n-1} x^{\frac{n}{2}-1}$

$DFT_n(\langle a_0, a_1, \ldots, a_{n-1}\rangle) = \langle y_0, y_1, \ldots, y_{n-1}\rangle$ där

$$y_k = DFT_{\frac{n}{2}}(\langle a_0, a_2, \ldots, a_{n-2}\rangle)_k + \omega_n^k DFT_{\frac{n}{2}}(\langle a_1, a_3, \ldots, a_{n-1}\rangle)_k$$

$DFT_1(\langle a \rangle) = a \cdot \omega_1^0 = a$

Dekomposition!

# FFT-algoritm

Anta att $n$ är en tvåpotens.

$\mathrm{DFT}_n(\langle a_0, a_1, \ldots, a_{n-1}\rangle) =$

IF $n=1$ THEN RETURN $\langle a_0\rangle$

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

$\omega \leftarrow 1$

$y^{[0]} \leftarrow \mathrm{DFT}_{\frac{n}{2}}(\langle a_0, a_2, \ldots, a_{n-2}\rangle)$

$y^{[1]} \leftarrow \mathrm{DFT}_{\frac{n}{2}}(\langle a_1, a_3, \ldots, a_{n-1}\rangle)$

FOR $k \leftarrow 0$ TO $\frac{n}{2}-1$ DO

$\quad y_k \leftarrow y^{[0]}_k + \omega \cdot y^{[1]}_k$

$\quad y_{k+\frac{n}{2}} \leftarrow y^{[0]}_k - \omega \cdot y^{[1]}_k$

$\quad \omega \leftarrow \omega \cdot \omega_n$

RETURN $\langle y_0, y_1, \ldots, y_{n-1}\rangle$

Analys:

$$T(n) = \begin{cases} \Theta(1) & \text{om } n=1 \\ 2T(\frac{n}{2}) + \Theta(n) & \text{om } n>1 \end{cases}$$

$$\Rightarrow T(n) = \Theta(n \log n)$$

# Invers till DFT

$\bar{y} = \mathrm{DFT}_n(\bar{a}) \iff \bar{y} = V_n \bar{a}$ dvs

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} = \underbrace{\begin{pmatrix} \omega_n^0 & \omega_n^0 & \omega_n^0 & \cdots & \omega_n^0 \\ \omega_n^0 & \omega_n^1 & \omega_n^2 & \cdots & \omega_n^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega_n^0 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)(n-1)} \end{pmatrix}}_{V_n = \text{Vandermondematrisen}} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

$V_n[k,j] = \omega_n^{kj}$

Vi söker $\bar{a} = \mathrm{DFT}_n^{-1}(\bar{y}) \iff \bar{a} = V_n^{-1}\bar{y}$

Sats: $V_n^{-1}[j,k] = \frac{1}{n}\omega_n^{-kj}$

$$\boxed{\begin{array}{l} \mathrm{DFT}_n^{-1}(\langle y_0, y_1, \ldots, y_{n-1}\rangle) = \langle a_0, a_1, \ldots, a_{n-1}\rangle \\ \text{där } a_j = \frac{1}{n}\sum_{k=0}^{n-1} y_k \omega_n^{-kj} = \frac{1}{n}\sum_{k=0}^{n-1} y_k e^{-2\pi i jk/n} \end{array}}$$

Jämför inversen till den vanliga Fouriertransformen:

$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{f}(t) e^{itx}\,dt$$

# Polynommultiplikation – FFT

$$\sum_{j=0}^{2n-1} c_j x^j = \left(\sum_{j=0}^{n-1} a_j x^j\right) \cdot \left(\sum_{j=0}^{n-1} b_j x^j\right)$$

Evaluering

$\langle y_0, \ldots, y_{2n-1}\rangle = \mathrm{DFT}_{2n}(\langle a_0, \ldots, a_{n-1}, 0, \ldots, 0\rangle)$

$\langle y'_0, \ldots, y'_{2n-1}\rangle = \mathrm{DFT}_{2n}(\langle b_0, \ldots, b_{n-1}, 0, \ldots, 0\rangle)$

$\langle c_0, \ldots, c_{2n-1}\rangle = \underset{\text{Interpolation}}{\mathrm{DFT}_{2n}^{-1}}(\langle \underbrace{y_0 \cdot y'_0, \ldots, y_{2n-1} \cdot y'_{2n-1}}_{\text{punktvis multiplikation}}\rangle)$

Tid: $\Theta(n \log n)$