Föreläsning 18 i ADK

Algoritmkonstruktion: polynomberäkningar och FFT

Stefan Nilsson

KTH

Operationer på envariabelpolynom

- Polynomet $A(x) = \sum_{j=0}^{n-1} a_j x^j = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$
- Kan lagras i **koefficientform** som $\langle a_0, a_1, \ldots, a_{n-1} \rangle$

Evaluering:

- Med Horners regel:
- $A(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-2} + x \cdot a_{n-1}) \cdots))$
- Tid: $\Theta(n)$

Operationer på envariabelpolynom

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 $0 \leqslant n-1$
- 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 **punkt-värde-form**:
- $\{(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}$ 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
- $A(x) = \sum_{k=0}^{n-1} y_k \frac{\prod_{j \neq k} (x - x_j)}{\prod_{j \neq k} (x_k - x_j)}$
- Tid: $\Theta(n^2)$

Alternativ lagringsmetod

Addition:

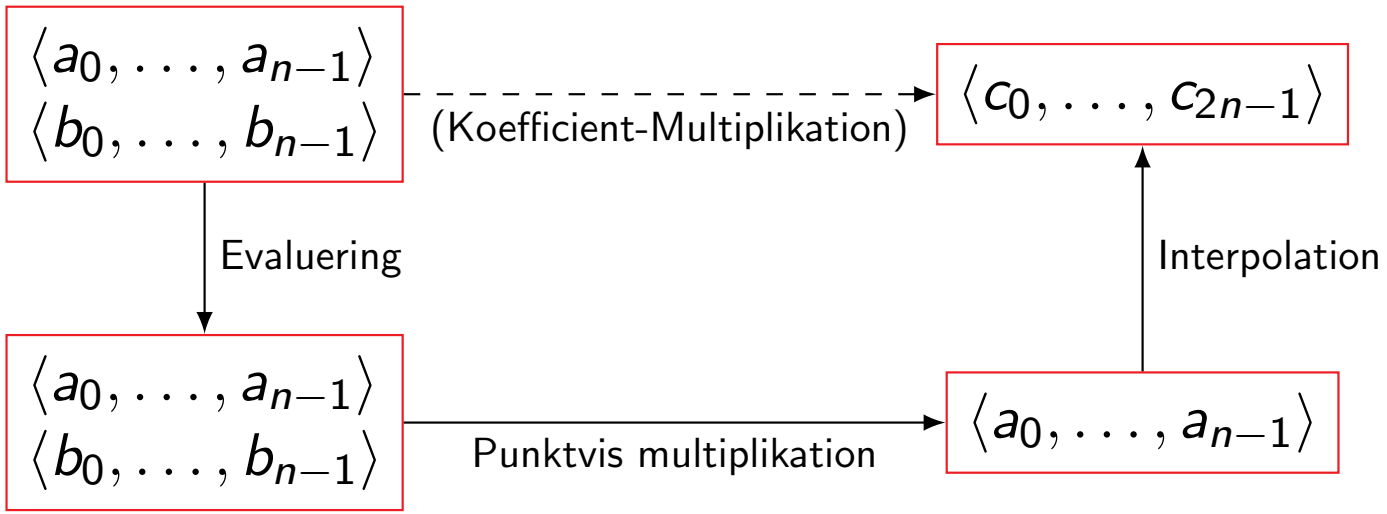
- $\{(x_0, y_0 + y'_0), (x_1, y_1 + y'_1), \cdots, (x_{n-1}, y_{n-1} + y'_{n-1})\}$
- Tid: $\Theta(n)$

Multiplikation:

- $\{(x_0, y_0 \cdot y'_0), (x_1, y_1 \cdot y'_1), \cdots, (x_{n-1}, y_{2n-2} \cdot y'_{2n-2})\}$
- Tid: $\Theta(n)$
- Problem: $2n - 1$ punkter krävs

Snabb polynommultiplikation i koefficientform

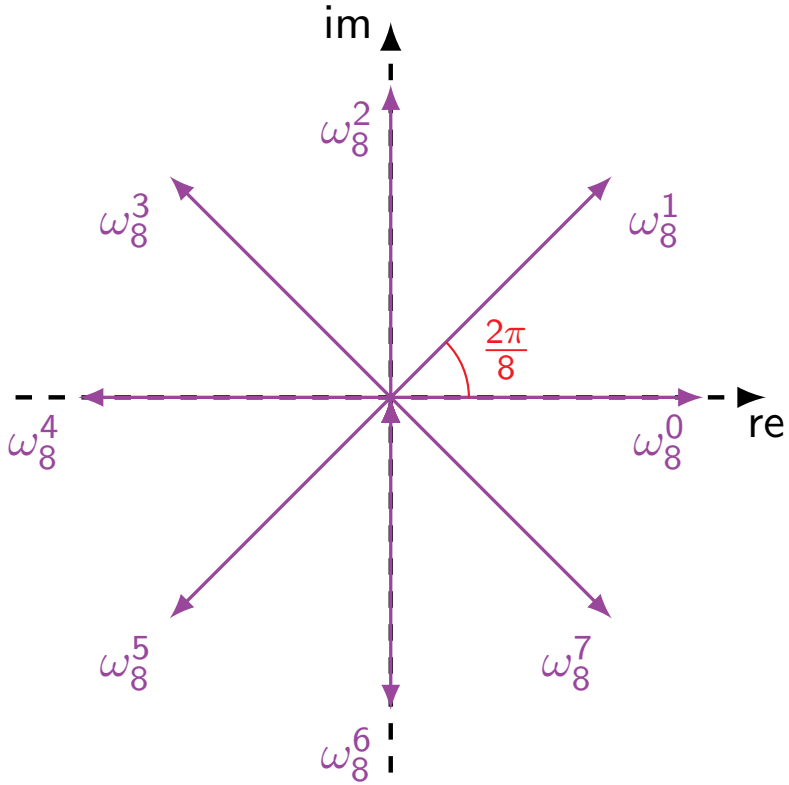
Om man kunder evaluera och interpolera snabbt så skulle man kunna multiplicera snabbt med hjälp av punktvis multiplikation



- Om evaluering och interpolation gick att göra i tid $o(n^2)$ så 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 ω 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 $\langle a_0, a_1, \ldots, a_{n-1} \rangle$

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

Där $y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{jk} = \sum_{j=0}^{n-1} a_j e^{2\pi ijk/n}$

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

Jämför den vanliga fouriertransformationen 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 $A(x) = \sum_{j=0}^{n-1} a_j x^j =$
  $= a_0 + a_2 x^2 + a_4 x^4 + \cdots + a_{n-2} x^{n-2} +$
  $+ x(a_1 + a_3 x^2 + a_5 x^4 + \cdots + a_{n-1} x^{n-2}) =$
  $= A^{[0]}(x^2) + x \cdot A^{[1]}(x^2)$
- $A^{[0]}(x) = a_0 + a_2 x + a_4 x^2 + \cdots + a_{n-2} x^{n/2-1}$
  $A^{[1]}(x) = a_1 + a_3 x + a_5 x^2 + \cdots + a_{n-1} x^{n/2-1}$
- $\mathrm{DFT}_n(\langle a_0, a_1, \ldots, a_{n-1} \rangle) = \langle y_0, y_1, \ldots, y_{n-1} \rangle$ där
  $y_k = \mathrm{DFT}_{\frac{n}{2}}(\langle a_0, a_2, \ldots, a_{n-2} \rangle)_k + \omega_n^k \mathrm{DFT}_{\frac{n}{2}}(\langle a_1, a_3, \ldots, a_{n-1} \rangle)_k$
- $\mathrm{DFT}_1(\langle a \rangle) = a \cdot \omega_1^0 = a$
- Dekomposition!

FFT-algoritm

Anta att n är en tvåpotens

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

- **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**
  - $y_k \leftarrow y_k^{[0]} + \omega y_k^{[1]}$
  - $y_{k+\frac{n}{2}} \leftarrow y_k^{[0]} - \omega y_k^{[1]}$
  - $\omega \leftarrow \omega \cdot \omega_n$
- **return** $\langle y_0, y_1, \ldots, y_{n-1}\rangle$

Analys: $T(n) = \begin{cases} \Theta(1) \text{ om } n = 1 \\ 2T(\frac{n}{2}) + \Theta(n) \text{ om } n > 1 \end{cases} \quad \Rightarrow T(n) = \Theta(n \log n)$

Invers till DFT

$\bar{y} = \text{DFT}_n(\bar{a}) \Leftrightarrow \bar{y} = V_n\bar{a}$, dvs:

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} = \underbrace{\begin{pmatrix} \omega_n^0 & \omega_n^0 & \omega_n^0 & \cdots & \omega_n^0 \\ \omega_n^0 & \omega_n^1 & \omega_n^2 & \cdots & \omega_n^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega_n^0 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)(n-1)} \end{pmatrix}}_{\substack{V_n = \text{ vandermondematrisen} \\ V_n[k,j] = \omega_n^{kj}}} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

Vi söker $\bar{a} = \text{DFT}^{-1}(\bar{y}) \Leftrightarrow \bar{a} = V^{-1}\bar{y}$

Invers till DFT

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

$$\text{DFT}_n^{-1}(\langle y_0, y_1, \ldots, y_{n-1}\rangle) = \langle a_0, a_1, \ldots, a_{n-1}\rangle$$

där $a_j = \frac{1}{n}\sum_{k=0}^{n-1} y_k \omega_n^{-kj} = \frac{1}{n}\sum_{k=0}^{n-1} y_k \omega_n^{-2\pi ijk/n}$

Jämför med inversen till den vanliga fouriertransformationen:

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

Polynommultiplikation - FFT

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

$$\left.\begin{aligned} \langle y_0, \ldots, y_{n-1}\rangle &= \mathrm{DFT}_{2n}(\langle a_0, \ldots, a_{n-1}, 0, \ldots, 0\rangle) \\ \langle y'_0, \ldots, y'_{n-1}\rangle &= \mathrm{DFT}_{2n}(\langle b_0, \ldots, b_{n-1}, 0, \ldots, 0\rangle) \end{aligned}\right\} \text{Evaluering}$$

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

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