3.5 Schnelle Fouriertransformation (FFT, DFT)

3.5.1 Grundlagen

Ein Polynom $P = \sum_i a_i x^i \in \mathbb{C}[x]$ vom Grad $\leq n$ ist eindeutig durch seine Koeffizienten a_i bestimmt, d.h. man hat eine Bijektion

$$\begin{aligned} \{\text{Polynome} \in \mathbb{C}[x] \text{ vom Grad} \leq n\} &\rightarrow \mathbb{C}^{n+1} \\ P_{\vec{a}} = \sum_{i=0}^{n} a_i x^i &\mapsto \vec{a} = (a_0, \ldots, a_n). \end{aligned}$$

Problem: $P_{\vec{a}} \cdot P_{\vec{b}} = P_{\vec{c}}$ mit $\vec{c} = (c_0, \ldots, c_{2n})$, $c_k = \sum_i a_{k-i} b_i$, und die naive Berechnung von $\vec{c}$ benötigt $\Theta(n^2)$ Operationen.

Bemerkung: $\vec{c} = \vec{a} * \vec{b}$ mit $c_k = \sum_i a_{k-i} b_i$ ist die Faltung von $\vec{a}$ und $\vec{b}$.

Es gibt noch eine weitere eindeutige Darstellung eines Polynoms.

Lemma 144
Seien $P = \sum_{i=0}^{n} a_i x^i$ und $Q = \sum_{j=0}^{n} b_j x^j$ Polynome ($\in \mathbb{C}[x]$) vom Grad $\leq n$ und seien $\omega_0, \ldots, \omega_n \in \mathbb{C}$ paarweise verschiedene Elemente. Dann gilt:

$$P = Q \iff P(\omega_i) = Q(\omega_i) \quad \text{für alle } i = 0, \ldots, n.$$

Beweis:
„$\Rightarrow$“: Klar.
„$\Leftarrow$“: Es gelte $P(\omega_i) = Q(\omega_i)$ für $i = 0, \ldots, n$. Dann ist jedes ω_i eine Nullstelle des Polynoms $P - Q$. Da $\mathrm{grad}(P - Q) \leq n$ gilt, folgt $P - Q = 0$ aus Satz 139. □

Man kann leicht zeigen, dass es zu jedem Tupel $(b_0, \ldots, b_n) \in \mathbb{C}^{n+1}$ (genau) ein Polynom $f \in \mathbb{C}[x]$ vom Grad $\leq n$ gibt, mit $f(\omega_i) = b_i$ für $i = 0, \ldots, n$ (z.B. das Newtonsche Interpolationspolynom, benannt nach Sir Isaac Newton (1643–1727)).

Somit erhalten wir eine weitere Bijektion:

$$\begin{aligned} \{\text{Polynome} \in \mathbb{C}[x] \text{ vom Grad} \leq n\} &\to \mathbb{C}^{n+1} \\ P &\mapsto (P(\omega_0), \ldots, P(\omega_n)) \end{aligned}$$

Vorteil:

$$P \times Q \mapsto (P(\omega_0)Q(\omega_0), \ldots, P(\omega_n)Q(\omega_n)) = \\ (P(\omega_0), \ldots, P(\omega_n)) \cdot (Q(\omega_0), \ldots, Q(\omega_n)).$$

Multiplikation benötigt nur $O(n)$ Operationen. „$\cdot$“ auf der rechten Seite bezeichnet hier das komponentenweise (Hadamard) Vektorprodukt (Jacques S. Hadamard (1865–1963)).

Problem: Bijektion i.a. zu komplex.

Definition 145

Ein $\omega \in \mathbb{C}$ heißt primitive n-te Einheitswurzel, wenn $\omega^k \neq 1$ für alle $k = 1, \ldots, n-1$ und $\omega^n = 1$ gilt, d.h. $\mathrm{ord}(\omega) = n$ in $\mathbb{C}^* = \mathbb{C} \setminus 0$.

Bemerkung: Es ist $\omega = e^{2i\pi/n}$ eine primitive n-te Einheitswurzel.

Definition 146

Sei $\omega \in \mathbb{C}$ eine primitive n-te Einheitswurzel, $n \in \mathbb{N}$. Die Abbildung

$$\begin{aligned} \mathcal{F}_{n,\omega}: \quad & \mathbb{C}^n \to \mathbb{C}^n, \\ & \vec{a} = (a_0, \ldots, a_{n-1}) \mapsto (P_{\vec{a}}(1), P_{\vec{a}}(\omega), \ldots, P_{\vec{a}}(\omega^{n-1})) \end{aligned}$$

heißt diskrete Fouriertransformation; wir schreiben auch kurz $\mathcal{F}$ für $\mathcal{F}_{n,\omega}$.

Die Fouriertransformation ist nach Jean Baptiste Joseph Fourier (1768–1830) benannt.

Bemerkung: $\mathcal{F}$ is nach Lemma 144 und anschließender Bemerkung eine Bijektion.

Lemma 147
Seien $\vec{a}, \vec{b} \in \mathbb{C}^n$ so, dass auch $\vec{a} * \vec{b} \in \mathbb{C}^n$. Dann gilt

$$\mathcal{F}(\vec{a} * \vec{b}) = \mathcal{F}(\vec{a}) \cdot \mathcal{F}(\vec{b}).$$

Beweis:
Es gilt

$$\begin{aligned}
\mathcal{F}(\vec{a}) \cdot \mathcal{F}(\vec{b}) &= (P_{\vec{a}}(1)P_{\vec{b}}(1), P_{\vec{a}}(\omega)P_{\vec{b}}(\omega), \ldots, P_{\vec{a}}(\omega^{n-1})P_{\vec{b}}(\omega^{n-1})) \\
&= (P_{\vec{c}}(1), P_{\vec{c}}(\omega), \ldots, P_{\vec{c}}(\omega^{n-1})) \\
&= \mathcal{F}(\vec{c}), \quad \text{mit } \vec{c} = \vec{a} * \vec{b}.
\end{aligned}$$

□

Idee: Berechne $\vec{a} * \vec{b}$ vermöge $\mathcal{F}^{-1}(\mathcal{F}(\vec{a}) \cdot \mathcal{F}(\vec{b}))$. Die komponentenweise Multiplikation $F(\vec{a}) \cdot \mathcal{F}(\vec{b})$ benötigt nur $O(n)$ Operationen.

Jedoch: $\mathcal{F}$ ist eine lineare Abbildung $\mathcal{F}(\vec{a}) = \Omega \cdot \vec{a}$, mit $\Omega = (\omega^{kl})_{0 \leq l,k \leq n-1}$. Die Matrixmultiplikation benötigt aber $\Omega(n^2)$ Operationen (also keine offensichtliche Verbesserung im Vergleich zur klassischen Polynom-Multiplikation)!

Ausweg: "Divide and Conquer"!!!

3.5.2 Berechnung der diskreten Fouriertransformation (FFT)

Sei $n = 2^k$ eine 2er-Potenz. Zerlege $\vec{a} = (a_0, \ldots, a_{n-1})$ in einen

geraden Anteil	$\vec{a}_g = (a_0, a_2, \ldots, a_{n-2})$	und einen
ungeraden Anteil	$\vec{a}_u = (a_1, a_3, \ldots, a_{n-1})$	

Dann gilt:

$$P_{\vec{a}}(x) = P_{\vec{a}_g}(x^2) + xP_{\vec{a}_u}(x^2)\,.$$

Beispiel 148

Sei $\vec{a} = (1, 2, 4, 8)$, also $P_{\vec{a}}(x) = 1 + 2x + 4x^2 + 8x^3$. Damit ist $\vec{a}_g = (1, 4)$ und $\vec{a}_u = (2, 8)$, also

$$\begin{aligned} &P_{\vec{a}_g}(x^2) + xP_{\vec{a}_u}(x^2) \\ &\qquad = 1 \cdot (x^2)^0 + 4 \cdot (x^2)^1 + x \cdot (2 \cdot (x^2)^0 + 8 \cdot (x^2)^1) \\ &\qquad = 1 + 2 \cdot x + 4 \cdot x^2 + 8 \cdot x^3 \end{aligned}$$

Lemma 149

Ist $\mathcal{F}_{\frac{n}{2},\omega^2}(\vec{a}_g) = (c_0, \ldots, c_{\frac{n}{2}-1})$ *und* $\mathcal{F}_{\frac{n}{2},\omega^2}(\vec{a}_u) = (d_0, \ldots, d_{\frac{n}{2}-1})$*, so gilt* $\mathcal{F}_{n,\omega}(\vec{a}) = (e_0, \ldots, e_{n-1})$ *mit*

$$\begin{aligned} e_i &= P_{\vec{a}}(\omega^i) \\ &= P_{\vec{a}_g}(\omega^{2i}) + \omega^i P_{\vec{a}_u}(\omega^{2i}) \\ &= c_i + \omega^i d_i \\ e_{\frac{n}{2}+i} &= P_{\vec{a}}(\omega^{\frac{n}{2}+i}) \\ &= P_{\vec{a}_g}(\omega^{2(\frac{n}{2}+i)}) + \omega^{\frac{n}{2}+i} P_{\vec{a}_u}(\omega^{2(\frac{n}{2}+i)}) \\ &= c_i + \omega^{\frac{n}{2}+i} d_i \end{aligned}$$

für $i = 0, \ldots, \frac{n}{2} - 1$.

Bem.: ω^2 ist primitive $\frac{n}{2}$-te Einheitswurzel. Natürlich ist $\omega^{2\frac{n}{2}} = 1$.

Dies liefert folgenden Divide-and-Conquer-Algorithmus:

DFT($\vec{a}$,ω)
  Eingabe: $\vec{a} = (a_0, \ldots, a_{n-1})$, $n = 2^k$, ω
  Ausgabe: $\mathcal{F}_{n,\omega}(\vec{a}) = (e_0, \ldots, e_{n-1})$

  if $n = 1$ then $e_0 := a_0$
  else
    $\vec{a}_g := (a_0, a_2, \ldots, a_{n-2})$
    $\vec{a}_u := (a_1, a_3, \ldots, a_{n-1})$
    $(c_0, \ldots, c_{\frac{n}{2}-1})$:=DFT($\vec{a}_g, \omega^2$)
    $(d_0, \ldots, d_{\frac{n}{2}-1})$:=DFT($\vec{a}_u, \omega^2$)
    for $i = 0$ to $\frac{n}{2} - 1$ do
      $e_i := c_i + \omega^i d_i$
      $e_{\frac{n}{2}+i} := c_i + \omega^{\frac{n}{2}+i} d_i$
    endfor
  endif
  return$(e_0, \ldots, e_{n-1})$

Satz 150

Der Algorithmus DFT *berechnet* $\mathcal{F}_{n,\omega}(\vec{a})$ *auf Eingabe* $n = 2^k$, $\vec{a}$, ω *in* $T(n) = O(n \log n)$ *Operationen.*

Beweis:

Aus dem Algorithmus erhält man folgende Rekursion

$$T(n) = 2T(n/2) + cn$$

mit einer Konstante $c > 0$ und $T(1) = 1$. Mit $n = 2^k$ folgt

$$\begin{aligned} T(2^k) &= 2T(2^{k-1}) + cn = 2(2T(2^{k-2}) + cn/2) + cn \\ &= \ldots = 2^\ell T(2^{k-\ell}) + \ell cn \end{aligned}$$

Speziell für $\ell = k$ gilt $T(2^k) = kc2^k + 2^k T(1)$, und wir erhalten $T(2^k) = O(2^k k) = O(n \log n)$. □