WS 2003/04

Diskrete Strukturen I

Ernst W. Mayr

mayr@in.tum.de
Institut für Informatik
Technische Universität München

11-21-2003

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

Lemma

Für $\vec{a}, \vec{b} \in \mathbb{C}^n$ gilt

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

Beweis.

Es gilt

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

□

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

Lemma

Für $\vec{a}, \vec{b} \in \mathbb{C}^n$ *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 $\Omega = (\omega^{kl})_{0 \le l,k \le n-1}$. Die Matrixmultiplikation benötigt aber $O(n^2)$ Operationen ($\rightarrow$ offensichtlich keine Verbesserung zur klassischen Polynom-Multiplikation)
Ausweg: "Divide and Conquer"!!!

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:

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

und somit folgt:

Berechnung der diskreten Fouriertransformation (FFT)

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

$$\begin{array}{rl} \text{geraden Anteil} & \vec{a}_g = (a_0, a_2, \ldots, a_{n-2}) \quad \text{und einen} \\ \text{ungeraden Anteil} & \vec{a}_u = (a_1, a_3, \ldots, a_{n-1}) \end{array}$$

Dann gilt:

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

und somit folgt:

Lemma

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

$$\begin{aligned} e_i &= c_i + \omega^i b_i \\ e_{\frac{n}{2}+i} &= c_i + \omega^{\frac{n}{2}+i} b_i \end{aligned}$$

für $i = 0, \ldots, \frac{n}{2} - 1$.

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

Dies liefert folgenden Divide-and-Conquer Algorithmus:

Eingabe: $\vec{a} = (a_0, \ldots, a_{n-1})$, $n = 2^k$, ω

Ausgabe: $\mathcal{F}_{n,\omega}(\vec{a}) = (e_0, \ldots, e_{n-1})$

```
DFT(a⃗,ω)
if n = 1 then e_0 := a_0
else
    a⃗_g := (a_0, a_2, ..., a_{n-2})
    a⃗_u := (a_1, a_3, ..., a_{n-1})
    (c_0, ..., c_{n/2-1}) :=DFT(a⃗_g, ω^2)
    (d_0, ..., d_{n/2-1}) :=DFT(a⃗_u, ω^2)
    for i = 0 to n/2 - 1 do
        e_i := c_i + ω^i b_i
        e_{n/2+i} := c_i + ω^{n/2+i} b_i
   endfor
endif
return(e_0, ..., e_{n-1})
```

Satz

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 $n = 2^{k+1}$ folgt

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

Speziell für $\ell = k+1$ gilt $T(2^{k+1}) = 2^{k+1}T(1) + (2k+1)c2^{k+1}$, und wir erhalten $T(2^k) = O(2^k k) = O(n \log n)$. $\square$

Satz

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 $n = 2^{k+1}$ folgt

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

Speziell für $\ell = k + 1$ gilt $T(2^{k+1}) = 2^{k+1}T(1) + (2k + 1)c2^{k+1}$, und wir erhalten $T(2^k) = O(2^k k) = O(n \log n)$. $\square$

Berechnung der inversen diskreten Fouriertransformation

Satz

Es gilt

$$\mathcal{F}_{n,\omega}^{-1} = \frac{1}{n}\mathcal{F}_{n,\omega^{-1}}.$$

Bem.: ω^{-1} ist ebenso eine primitive n-te Einheitswurzel. Zum Beweis von Satz 4 benötigen wir folgendes Lemma:

Lemma

Ist ω eine primitive n-te Einheitswurzel, so gilt

$$\sum_{j=0}^{n-1} \omega^{kj} = 0$$

für alle $k = 1, \ldots, n-1$.

Berechnung der inversen diskreten Fouriertransformation

Satz

Es gilt

$$\mathcal{F}_{n,\omega}^{-1} = \frac{1}{n}\mathcal{F}_{n,\omega^{-1}}.$$

Bem.: ω^{-1} ist ebenso eine primitive n-te Einheitswurzel. Zum Beweis von Satz 4 benötigen wir folgendes Lemma:

Lemma

Ist ω eine primitive n-te Einheitswurzel, so gilt

$$\sum_{j=0}^{n-1} \omega^{kj} = 0$$

für alle $k = 1, \ldots, n-1$.

Beweis. Für jedes $a \in \mathbb{C}$ gilt $\sum_{j=0}^{n-1} a^j = \frac{a^n - 1}{a-1}$. Speziell für $a = \omega^k$ ist $a^n = \omega^{kn} = 1$, $(k = 1, \ldots, n-1)$. □

Nun zum Beweis von Satz 4.

Beweis.
Sei $\vec{e} = \mathcal{F}_{n,\omega}(\vec{a}) = (e_0, \ldots, e_{n-1})$. Wir zeigen, dass

$$\frac{1}{n}\mathcal{F}_{n,\omega^{-1}}(\vec{e}) = \vec{a}$$

gilt.

$$\begin{aligned} P_{\vec{e}}(\omega^{-k}) &= \sum_{j=0}^{n-1} e_j \omega^{-kj} = \sum_{j=0}^{n-1} P_{\vec{a}}(\omega^j)\omega^{-kj} \\ &= \sum_{j=0}^{n-1}\sum_{i=0}^{n-1} a_i \omega^{ij}\omega^{-kj} = \sum_{i=0}^{n-1} a_i \sum_{j=0}^{n-1} \omega^{(i-k)j} = n a_k, \end{aligned}$$

denn nach Lemma 5 ist $\sum_{j=0}^{n-1} \omega^{(i-k)j} = 0$, falls $i \neq k$.
Im Fall $i = k$ gilt $\sum_{j=0}^{n-1} \omega^{(i-k)j} = n$. □

Beweis. Für jedes $a \in \mathbb{C}$ gilt $\sum_{j=0}^{n-1} a^j = \frac{a^n - 1}{a-1}$. Speziell für $a = \omega^k$ ist $a^n = \omega^{kn} = 1$, $(k = 1, \ldots, n-1)$. $\square$

Nun zum Beweis von Satz 4.

Beweis.
Sei $\vec{e} = \mathcal{F}_{n,\omega}(\vec{a}) = (e_0, \ldots, e_{n-1})$. Wir zeigen, dass

$$\frac{1}{n}\mathcal{F}_{n,\omega^{-1}}(\vec{e}) = \vec{a}$$

gilt.

$$\begin{aligned} P_{\vec{e}}(\omega^{-k}) &= \sum_{j=0}^{n-1} e_j \omega^{-kj} = \sum_{j=0}^{n-1} P_{\vec{a}}(\omega^j)\omega^{-kj} \\ &= \sum_{j=0}^{n-1}\sum_{i=0}^{n-1} a_i \omega^{ij}\omega^{-kj} = \sum_{i=0}^{n-1} a_i \sum_{j=0}^{n-1} \omega^{(i-k)j} = n a_k, \end{aligned}$$

denn nach Lemma 5 ist $\sum_{j=0}^{n-1} \omega^{(i-k)j} = 0$, falls $i \neq k$.
Im Fall $i = k$ gilt $\sum_{j=0}^{n-1} \omega^{(i-k)j} = n$. $\square$