Division mit Schulmethode

Satz Division mit Rest von Polynomen

Seien $a(x), b(x) \in \mathbb{Q}[x]$ mit $b(x) \neq 0$. Dann gibt es eindeutige $q(x), r(x) \in \mathbb{Q}[x]$ mit $a(x) = q(x) \cdot b(x) + r(x)$ und $\text{grad}(r) < \text{grad}(b)$.

Beweis:

- Sei $\text{grad}(a) = n$ und $\text{grad}(b) = m$.
- Für $n < m$ setze $q(x) = 0$ und $r(x) = a(x)$. Sei also $n \geq m$.
- **Existenz:** Beweis per Induktion über n.
- **IA:** Für $n = 0$ gilt $m = 0$. Setze $q(x) = \frac{a_0}{b_0}$ und $r(x) = 0$.
- **IS** $n-1 \to n$: Sei $\text{grad}(a) = n$.
- Dann besitzt $a'(x) = a(x) - \frac{a_n}{b_m} x^{n-m} b(x)$ Grad höchstens $n-1$.
- IV: $\exists q'(x), r'(x)$ mit $a'(x) = q'(x)b(x) + r'(x)$, $\text{grad}(r') < \text{grad}(b)$.
- D.h. $a(x) = a'(x) + \frac{a_n}{b_m} \cdot b(x) = (q'(x) + \frac{a_n}{b_m} x^{n-m})b(x) + r'(x)$ mit $\text{grad}(r') < \text{grad}(b)$.

Eindeutigkeit von Quotient und Rest

Beweis: Eindeutigkeit

- Sei $a = qb + r = q'b + r'$ mit $(q, r) \neq (q', r')$ und $\text{grad}(r), \text{grad}(r') < b$.
- D.h. $b(q - q') = r' - r$.
- Wegen $\text{grad}(r' - r) \leq \max\{\text{grad}(r), \text{grad}(r')\} < b$ folgt $q = q'$.
- Damit ist $0 = r' - r$ und daher $r = r'$.

Anmerkung:

- Beweis liefert Komplexität $\mathcal{O}(n^2)$ der Division mit Rest.

Definition Point-value Form

Sei $a \in R[x]$ mit $\text{grad}(a) \leq n$. Seien $x_0, \ldots, x_n \in R$ paarweise verschieden und $y_i = a(x_i)$ für $i = 0, \ldots, n$. Dann bezeichnen wir die Menge $\{(x_0, y_0), \ldots, (x_n, y_n)\}$ als *Point-value Form* von a.

Koeffizienten aus Point-value Form

Satz Berechnung der Koeffizienten

Sei $\{(x_0, y_0), \ldots, (x_n, y_n)\}$ eine Point-value Form. Dann existiert ein eindeutiges $a \in R[x]$ mit $\mathrm{grad}(a) \leq n$ und $a(x_i) = y_i$ für $i = 0, \ldots, n$.

Beweis:

- Wir definieren die Koeffizienten $a_0, \ldots, a_n$ von a als Lösung von

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

- Die Matrix heißt Vandermonde-Matrix V.
- V ist invertierbar für paarweise verschiedene x_i.
- Damit existiert genau eine Lösung

$$(a_0, \ldots, a_n)^t = V^{-1}(y_0, \ldots, y_n)^t.$$

Lagrange Interpolation

Beweis: alternativ mittels Lagrange Interpolation

- Wir setzen $p_i(x) = \frac{\prod_{j \neq i} x - x_j}{\prod_{j \neq i} x_i - x_j}$ und $a(x) = \sum_{i=0}^{n} y_i p_i(x)$.
- Jedes Polynom p_i besitzt Grad n und damit $\mathrm{grad}(a) = n$.
- Es gilt $p_i(x_k) = \begin{cases} 1 & \text{für } i = k \\ 0 & \text{sonst} \end{cases}$.
- Daraus folgt $a(x_i) = y_i$ für $i = 0, \ldots, n$.

Arithmetik in Point-value Form

Addition/Subtraktion:

- Sei $a(x) = \{(x_0, y_0), \ldots, (x_n, y_n)\}$, $b(x) = \{(x_0, y'_0), \ldots, (x_n, y'_n)\}$.
- Dann gilt $(a \pm b)(x) = \{(x_0, y_0 \pm y'_0), \ldots, (x_n, y_n \pm y'_n)\}$.
- D.h. Addition/Subtraktion besitzt lineare Komplexität $\mathcal{O}(n)$.

Multiplikation:

- Seien $\mathrm{grad}(a), \mathrm{grad}(b) < n$ und
$$a(x) = \{(x_0, y_0), \ldots, (x_{2n-1}, y_{2n-1})\},$$
$$b(x) = \{(x_0, y'_0), \ldots, (x_{2n-1}, y'_{2n-1})\}.$$
- Dann gilt $(ab)(x) = \{(x_0, y_0 y'_0), \ldots, (x_{2n-1}, y_{2n-1} y'_{2n-1})\}$.
- D.h. die Multiplikation besitzt ebenfalls lineare Komplexität $\mathcal{O}(n)$.

n-te Einheitswurzeln

Problem:

- Konvertierung von Koeffizientenform in Point-value Form erfordert das Evaluieren an n Stellen, d.h. $\mathcal{O}(n^2)$ mit Horner-Schema.
- Konvertierung von Point-value Form in Koeffizientenform erfordert Lagrange Interpolation, d.h. wiederum Komplexität $\mathcal{O}(n^2)$.

Lösung:

- Werte Polynome an geeigneten Stellen aus, den Einheitswurzeln.

Definition n-te Einheitswurzel

Sei $\omega \in \mathbb{C}$. Wir nennen ω eine *n-te Einheitswurzel* falls $\omega^n = 1$

Anmerkungen:

- Die Werte $e^{2\pi i \frac{k}{n}}$, $k = 0, \ldots, n-1$ sind genau die n-ten Einheitswurzeln. Es gilt $e^{2\pi ik} = (e^{2\pi i})^k = 1$.
- Erinnerung: $e^{i\alpha} = \cos(\alpha) + i \sin(\alpha)$. Die n-ten Einheitswurzeln teilen den Einheitskreis in n Segmente.
- Wir bezeichnen $\omega_n = e^{\frac{2\pi i}{n}}$ als primitive n-te Einheitswurzel.

Einheitswurzeln bilden Gruppe

Satz Gruppe der Einheitswurzeln

Sei $\omega_n = e^{\frac{2\pi i}{n}}$. Dann ist $(\langle w_n \rangle, \cdot)$ eine zyklische Gruppe der Ordnung n.

Beweis:

- Die Ordnung von ω_n ist n. Es gilt $\omega_n^n = e^{2\pi i} = 1$.
- Die Gruppe ist abgeschlossen, denn $\omega_n^i \cdot \omega_n^j = \omega_n^{i+j \bmod n}$.
- Das neutrale Element der Gruppe ist 1.
- Das zu ω_n^i inverse Element ist ω_n^{n-i}.

Lemma Eliminationslemma

Für alle $n, k \in \mathbb{N}_0$ und $d \in \mathbb{N}$ gilt $\omega_{dn}^{dk} = \omega_n^k$.

Beweis:

- Es gilt $\omega_{dn}^{dk} = e^{2\pi i \frac{dk}{dn}} = e^{2\pi i \frac{k}{n}} = \omega_n^k$.

Halbierungslemma

Korollar zum Eliminationslemma

Sei $n \in \mathbb{N}$ gerade. Dann gilt $\omega_n^{\frac{n}{2}} = (-1)$.

Beweis:

- Es gilt $\omega_n^{\frac{n}{2}} = \omega_2 = e^{\pi i} = \cos(\pi) + i \sin(\pi) = (-1)$.

Lemma Halbierungslemma

Sei $n \in \mathbb{N}$ gerade. Dann sind die Quadrate der n-ten Einheitswurzeln genau die $\frac{n}{2}$-ten Einheitswurzeln.

Beweis:

- Eliminationslemma liefert $(\omega_n^k)^2 = \omega_n^{2k} = \omega_{\frac{n}{2}}^k$ für $k \in [n]$.
- Je zwei n-te Einheitswurzeln besitzen dasselbe Quadrat, denn

$$(\omega_n^k)^2 = (-\omega_n^k)^2 = (\omega_n^{\frac{n}{2}} \cdot \omega_n^k)^2 = (\omega_n^{k+\frac{n}{2}})^2 \text{ für alle } k \in [n].$$

Die Diskrete Fourier-Transformation (DFT)

Definition Diskrete Fourier-Transformation

Sei $a(x) = \sum_{i=1}^{n} a_i x^i \in R[x]$ vom Grad $n-1$. Sei $y_k = a(\omega_n^k)$ für $k \in \mathbb{Z}_k$.

1. Der Vektor $a = (a_0, \ldots, a_{n-1})$ heißt *Koeffizientenvektor* von $a(x)$.
2. $(y_0, \ldots, y_{n-1}) \in \mathbb{C}^n$ heißt *diskrete Fouriertransfomierte* von a.

Wir schreiben $y = \mathrm{DFT}_n(a, \omega)$.

Idee der Schnellen Fourier-Transformation (FFT)

Idee: FFT

- Sei $a(x) = \sum_{i=0}^{n-1} a_i x^i$. Wir nehmen vereinfachend $n = 2^k$ an.
- Wir definieren $a_g(x) = a_0 + a_2 x + a_4 x^2 + \ldots + a_{n-2} x^{\frac{n}{2}-1}$ und
$$a_u(x) = a_1 + a_3 x + a_5 x^2 + \ldots + a_{n-1} x^{\frac{n}{2}-1}.$$
- Dann gilt $a(x) = a_g(x^2) + x \cdot a_u(x^2)$.
- D.h. anstatt $a(x)$ an den n-ten Einheitswurzeln auszuwerten, werten wir $a_g(x)$ und $a_u(x)$ an deren Quadraten aus.
- Quadrate der n-ten Einheitswurzeln sind $\frac{n}{2}$-te Einheitswurzeln.
- D.h. wir evaluieren 2 Polynome vom Grad kleiner $\frac{n}{2}$ an $\frac{n}{2}$ Stellen.
- Danach kombinieren wir die Auswertungen von a_g und a_u.
- Sei $y = (y_0, \ldots, y_{n-1}) = \mathrm{DFT}(a, \omega)$. Dann gilt:
$$y_k = a(\omega_n^k) = a_g(\omega_n^{2k}) + \omega_n^k \cdot a_u(\omega_n^{2k}) = a_g(\omega_{\frac{n}{2}}^k) + \omega_n^k \cdot a_u(\omega_{\frac{n}{2}}^k).$$

FFT-Algorithmus

Algorithmus FFT

EINGABE: $(a_0, \ldots, a_{n-1}, n)$ mit $n = 2^k$

1. If $(n = 1)$ return $a = (a_0)$
2. $\omega_n \leftarrow e^{\frac{2\pi i}{n}}; \omega \leftarrow 1;$
3. $(y_0^g, y_1^g, \ldots, y_{\frac{n}{2}-1}^g) \leftarrow \mathrm{FFT}(a_0, a_2, \ldots, a_{n-2}, \frac{n}{2})$
4. $(y_0^u, y_1^u, \ldots, y_{\frac{n}{2}-1}^u) \leftarrow \mathrm{FFT}(a_1, a_3, \ldots, a_{n-1}, \frac{n}{2})$
5. For $k \leftarrow 0$ to $\frac{n}{2} - 1$
   1. $y_k \leftarrow y_k^g + \omega y_k^u$
   2. $y_{k+\frac{n}{2}} \leftarrow y_k^g - \omega y_k^u$
   3. $\omega \leftarrow \omega \cdot \omega_n$

AUSGABE: $(y_0, \ldots, y_n) = \mathrm{DFT}(a, \omega)$

Laufzeit FFT

Satz FFT

Sei $a = (a_0, \ldots, a_{n-1})$ ein Koeffizientenvektor mit $n = 2^k$. Algorithmus FFT berechnet bei Eingabe von (a, n) in Zeit $\mathcal{O}(n \log n)$ $\mathrm{DFT}(a, n)$.

Laufzeit: Sei $T(n)$ die Gesamtlaufzeit.

- Rekursion in Schritt 3,4: $2T(\frac{n}{2})$.
- Kosten pro Rekursionsschritt in Schritt 2,5: $\mathcal{O}(n)$.
- Kosten für $T(1)$ in Schritt 1: $\mathcal{O}(1)$.
- Liefert Rekursionsgleichung $T(n) = 2T(\frac{n}{2}) + \mathcal{O}(n)$, $T(1) = \mathcal{O}(1)$.
- Daraus folgt eine Laufzeit von $T(n) = \mathcal{O}(n \log n)$. (Übungsaufgabe)

Korrektheit FFT

Korrektheit:

- Schritt 1: $y = \mathrm{DFT}(a_0) = a(\omega_1) = a_0\omega_1^0 = a_0$.
- Schritt 2+5.2: ω enthält stets ω_n^k. Die Inititialisierung mit $k = 0$ erfolgt in Schritt 2, das Update erfolgt in Schritt 5.2.
- Schritt 3+4:
  - $(y_0^g, y_1^g, \ldots, y_{\frac{n}{2}-1}^g) = \mathrm{DFT}_{\frac{n}{2}}(a_g, \omega)$, d.h. $y_k^g = a_g(\omega_{\frac{n}{2}}^k) = a_g(\omega_n^{2k})$.
  - $(y_0^u, y_1^u, \ldots, y_{\frac{n}{2}-1}^u) = \mathrm{DFT}_{\frac{n}{2}}(a_u, \omega)$, d.h. $y_k^u = a_u(\omega_{\frac{n}{2}}^k) = a_u(\omega_n^{2k})$.
- Schritt 5.1: Es gilt für $k = 0, \ldots, \frac{n}{2} - 1$

$$y_k = y_k^g + \omega_n^k \cdot y_k^u = a_g(\omega_n^{2k}) + \omega_n^k \cdot a_u(\omega_n^{2k}) = a(\omega_n^k).$$

- Schritt 5.2: Es gilt für $k + \frac{n}{2} = \frac{n}{2}, \ldots, n - 1$

$$\begin{aligned} y_{k+\frac{n}{2}} &= y_k^g - \omega_n^k y_k^u = y_k^g + \omega_n^{k+\frac{n}{2}} y_k^u \\ &= a_g(\omega_n^{2k}) + \omega_n^{k+\frac{n}{2}} a_u(\omega_n^{2k}) \\ &= a_g(\omega_n^{2k+n}) + \omega_n^{k+\frac{n}{2}} a_u(\omega_n^{2k+n}) = a(\omega_n^{k+\frac{n}{2}}). \end{aligned}$$