Wiederholung

- Chinesischer Restsatz
  - $\mathbb{Z}_N \cong \mathbb{Z}_{m_1} \times \ldots \times \mathbb{Z}_{m_n}$
  - $\mathbb{Z}_N^* \cong \mathbb{Z}_{m_1}^* \times \ldots \times \mathbb{Z}_{m_n}^*$
  - Quadratwurzeln der 1 sind ($\pm$ 1, $\pm$ 1, …, $\pm$ 1)
  - Nichttriviale Quadratwurzeln erlauben Faktorisierung

- RSA
  - Schlüsselgenerierung
  - Ver-/ und Entschlüsselung
  - Faktorisieren $\Leftrightarrow$ Berechnen von ϕ(N) $\Leftrightarrow$ Berechnen von d

Sicherheit von RSA

RSA-Problem
Gegeben: N, e, $c=m^e \bmod N$
Gesucht: $m \in \mathbb{Z}_N$

- D.h. $m=c^{1/e} \bmod N$ ist die e-te Wurzel von c in $\mathbb{Z}_N$.
- Beachte: Spezialfälle für e-te Wurzeln sind leicht.
  - Sei $m^e < N$. Dann kann $c^{1/e}$ in $\mathbb{N}$ leicht berechnet werden.

- Faktorisieren ist nicht NP-schwer (unter geeigneten Komplexitätsannahmen).
- Offenes Problem:
  - Algorithmus für e-te Wurzeln $\Rightarrow$ Algorithmus zum Faktorisieren

Polynome

Form $p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots a_1 x + a_0$

- a_i: Koeffizienten des Polynoms
  - $p(x) \in R[x]$, falls $a_i \in R$ (z.B. $R=\mathbb{Z}, \mathbb{R}, \mathbb{C}$)
- $grad(p) := max\{i \in \mathbb{N}_0 | a_i \neq 0\}$
  - $grad(0) := (-1)$

Annahme: Operationen (+,-,*,/) auf a_i und x in Zeit $\mathcal{O}(1)$.

Frage: Komplexität von (+,-,*,/,Auswerten) für Polynome?

Auswerten eines Polynoms

Naive Auswertung von $p(x) = a_n x^n + \ldots + a_0$:

- $\sum_{i=0}^{n+1} i = \mathcal{O}(n^2)$ Multiplikationen und n Additionen
- Gesamt: $\mathcal{O}(n^2)$

Hornerschema:

$p(x) = ((\ldots((a_n * x + a_{n-1}) * x + a_{n-2}) + \ldots) * x + a_1) * x + a_0$

- n Multiplikationen und n Additionen
- Gesamt: $\mathcal{O}(n)$

Addition/Subtraktion von p(x)

Seien $a(x) = a_n x^n + \ldots + a_0$, $b(x) = b_m x^m + \ldots + b_0$

- ObdA n>m. Setze $b_{m+1}=\ldots=b_n=0$.

$(a+b)(x) = c_n x^n + \ldots + c_0$ mit $c_i = a_i + b_i$ für $0 \leq i \leq n$

- Laufzeit Addition/Subtraktion: $\mathcal{O}(n)$

Multiplikation Schulmethode

Seien $a(x) = a_n x^n + \ldots + a_0$, $b(x) = b_m x^m + \ldots + b_0$

$(a*b)(x) = (a_n * b_m) * x^{n+m} + (a_{n-1} b_m + a_n b_{m-1}) x^{n+m-1} + \ldots + (a_0 b_1 + a_1 b_0) x + a_0 b_0$
$= \sum_{i=0}^{n+m} c_i x^i$ mit $c_i = \sum_{j=0}^{i} a_j b_{i-j}$

- Beachte: $a_{n+1} = \ldots = a_{n+m} = 0$ und $b_{m+1} = \ldots = b_{n+m} = 0$
- Laufzeit: ObdA $n \geq m$
  - Höchstens 2n+1 Koeffizienten
  - $\mathcal{O}(n)$ Additionen und Multiplikationen pro Koeffizient
  - Gesamt: $\mathcal{O}(n^2)$ Operationen
- Ziel: Laufzeit $\mathcal{O}(n \log n)$

Division Schulmethode

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

Sei grad(a)=n, grad(b)=m.
Existenz:
Für n<m: Setze q(x)=0, r(x)=a(x).
Sei n $\geq$ m. Beweis per Induktion über n

- IA: Für n=0 gilt m=0. Setze q(x)=1 und r(x)=0.
- IS n-1 $\rightarrow$ n: a‘(x) = a(x) – $a_n/b_m x^{n-m}$*b(x) hat Grad $\leq$ n-1
  - IV: $\exists$ q‘(x),r‘(x) mit a‘(x) = q‘(x)*b(x) + r‘(x) mit grad(r‘)<grad(b)
    $\Rightarrow$ $a(x) = a‘(x) + a_n/b_m$*b(x) = $(q‘(x) + a_n/b_m *x^{n-m})$*b(x) + r‘(x) mit grad(r‘)<grad(b).

Eindeutigkeit:
Sei a=q*b+r=q‘b+r‘ mit (q,r)$\neq$(q‘,r‘) und grad(r), grad(r‘) < b

- Es gilt b(q-q‘) = r‘-r
- Wegen grad(r‘-r) $\leq$ grad(r‘) < b muss q=q‘ gelten.
- Damit ist 0 = r‘-r und daher r=r‘.

Liefert Methode mit Laufzeit $\mathcal{O}(n^2)$.

Point-value Repräsentation

- Sei a(x) Polynom vom Grad < n.
- Seien $x_0,x_1,\ldots,x_{n-1}$ paarweise verschieden.
- Sei $y_i = a(x_i)$ für $0 \leq i \leq n-1$.
- $\{(x_0,y_0), (x_1,y_1),\ldots,(x_{n-1},y_{n-1})\}$ heisst Point-value Form von a(x).

Satz: Sei $\{(x_0,y_0), (x_1,y_1),\ldots,(x_n,y_n)\}$ eine Point-value Form. Dann gibt es ein eindeutiges a(x) mit grad(a)<n mit $a(x_i)=y_i$ für i=0,…,n.

Finde Koeffizienten $a_0, \ldots, a_{n-1}$ mit

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \ldots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \ldots & x_{n-1}^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

Vandermonde-Matrix ist invertierbar für paarweise verschiedene x_i.

$\Rightarrow$ Existenz und Eindeutigkeit des Lösungsvektors $(a_0,\ldots,a_{n-1})$.

Alternativ: Lagrange Interpolation

Lagrange Interpolation

- Setze

$$p_k(x) = \frac{\prod_{j \neq k} x - x_j}{\prod_{j \neq k} x_k - x_j} \quad \text{und} \quad a(x) = \sum_{k=0}^{n} y_k p_k(x)$$

- Jedes der $p_k(x)$ hat Grad n, daher auch a(x).
- $p_k(x_k)=1$ und $p_k(x_i) = 0$ für $i \neq k$
- Daher gilt $a(x_i) = y_i$ für i=0,…,k

Arithmetik mit Point-value Form

Addition: Sei c(x) = a(x)+b(x).

- a(x) = $\{(x_0,y_0), \ldots, (x_n,y_n)\}$
- b(x) = $\{(x_0,y_0'), \ldots, (x_n, y_n')\}$
- c(x) = $\{(x_0,y_0+y_0'),\ldots, (x_n,y_n+y_n')\}$

Addition in $\mathcal{O}(n)$

Multiplikation: Sei c(x) = a(x)*b(x), grad(a), grad(b) < n

- a(x) = $\{(x_0,y_0),\ldots,(x_{2n-1}, y_{2n-1})\}$
- b(x) = $\{(x_0, y_0'),\ldots,(x_{2n-1}, y_{2n-1}')\}$
- c(x) = $\{(x_0, y_0*y_0'), \ldots, (x_{2n-1}, y_{2n-1}*y_{2n-1}')\}$

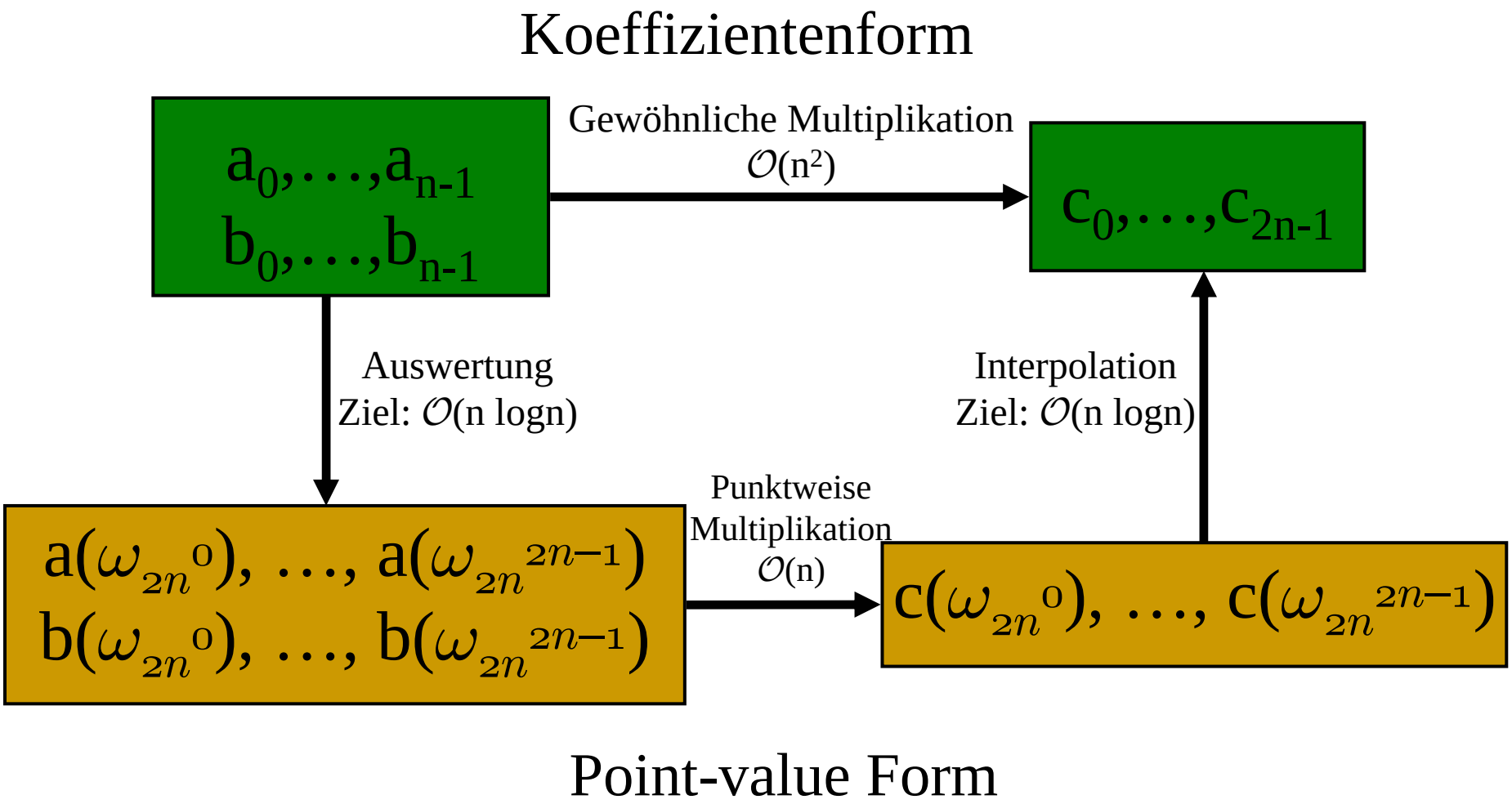
Multiplikation in $\mathcal{O}(n)$

Problem:

- Konvertierung Koeffizientenform $\rightarrow$ Point-value Form:
  - Auswerten von Polynom an n Stellen: $\mathcal{O}(n^2)$ mit Horner-Schema
- Konvertierung Point-value Form $\rightarrow$ Koeffizientenform:
  - Lagrange-Interpolation in Zeit $\mathcal{O}(n^2)$.

Lösung: Werten an geeigneten Stellen aus: 2n-te Einheitswurzeln ω_{2n}.

Multiplizieren von Polynomen



n-te Einheitswurzeln

Def: Eine n-te Einheitswurzel ist eine komplexe Zahl $\omega \in \mathbb{C}$ mit der Eigenschaft $\omega^n = 1$

- Es gibt exakt n viele n-te Einheitswurzeln

  $e^{2\pi ik/n}$ für k=0,1,…,n-1
- Erinnerung: $e^{i\alpha} = \cos(\alpha) + i*\sin(\alpha)$
  - n-te Einheitswurzeln besitzen gleichen Abstand auf dem Einheitskreis in der komplexen Ebene
- $\omega_n = e^{2\pi i/n}$ heißt primitive n-te Einheitswurzel
  - Alle n-ten Einheitswurzeln sind Potenzen von ω_n