Isomorphismus

Definition Gruppen-Isomorphismus

Seien $(G, +)$ und $(G', \cdot)$ Gruppen. Die Abbildung $f : G \to G'$ heißt *Gruppen-Isomorphismus*, falls gilt

1. f ist bijektiv
2. $f(u + v) = f(u) \cdot f(v)$ für alle $u, v \in G$, die sogenannte *Homomorphismus*-Eigenschaft von f.

Wir nennen G und G' *isomorph*, falls ein solcher Gruppen-Isomorphismus existiert. Notation: $G \cong G'$.

Zyklische Gruppen und $(\mathbb{Z}_m, +)$

Satz Isomorphie zu $\mathbb{Z}_m$

Sei G eine zyklische Gruppe mit Ordnung m. Dann gilt $G \cong (\mathbb{Z}_m, +)$.

Beweis:

- Da G zyklisch ist, gibt es ein $a \in G$ mit $G = \{a^i \mid i \in \mathbb{Z}_m\}$.
- Wir betrachten die Abbildung $f : \mathbb{Z}_m \to G, i \mapsto a^i$.
- Wegen $G = \{a^i \mid i \in \mathbb{Z}_m\}$ ist f surjektiv.
- Da $|G| = |\mathbb{Z}_m|$, ist f ebenfalls bijektiv.
- f ist ein Homomorphismus, da für alle $i, j \in \mathbb{Z}_m$ gilt
$$f(i + j) = a^{i+j} = a^i \cdot a^j = f(i) \cdot f(j).$$
- Damit ist f ein Gruppen-Isomorphismus und $G \cong (\mathbb{Z}_m, +)$.

Diskreter Logarithmus Problem

DLP: Diskreter Logarithmus Problem

Gegeben: G mit Generator $a \in G$, Element $b \in G$
Gesucht: $i \in \mathbb{Z}_{|G|}$ mit $a^i = b$

Lösung des DLP:

- Betrachten den Gruppen-Isomorphismus $f : i \mapsto a^i$.
- Für die Umkehrfunktion $f^{-1} : a^i \mapsto i$ gilt $f(b) = i$.
- D.h. f^{-1} löst das Diskrete Logarithmus Problem.
- Da DLP in einigen Gruppen als schweres Problem gilt, kann f nicht immer in beiden Richtungen effizient berechenbar sein.

Komplexität modularer Arithmetik

Satz Modulare Addition

Sei $m \in \mathbb{N}$ mit Bitlänge n und $a, b \in \mathbb{Z}_m$. Dann kann $a + b \bmod m$ in Zeit $\mathcal{O}(n)$ berechnet werden.

Beweis: Schulmethode

- Schreibe alle Operanden in Binärform $a = a_{n-1} \dots a_0 = \sum_{i=0}^{n} a_i 2^i$.
- Addiere a, b bitweise mit Übertrag, beginnend bei a_0, b_0.
- Falls $a + b > m$, subtrahiere bitweise m vom Ergebnis.
- **Laufzeitkomplexität** dieser Methode: $\mathcal{O}(n) = \mathcal{O}(\log m)$.
- D.h. Addition/Subtraktion besitzen Laufzeit linear in der Größe der Operanden.

Komplexität modularer Multiplikation

Satz Modulare Multiplikation

Sei $m \in \mathbb{N}$ mit Bitlänge n und $a, b \in \mathbb{Z}_m$. Dann kann $a \cdot b \bmod m$ in Zeit $\mathcal{O}(n^2)$ berechnet werden.

Beweis:

Algorithmus MULTIPLIKATION SCHULMETHODE

EINGABE: $a = a_{n-1} \ldots a_{n-1}, b, m$

1. $c \leftarrow 0$; $h \leftarrow b$;
2. For $i \leftarrow 0$ to $n-1$
   1. If $(a_i = 1)$ then $c \leftarrow c + h \bmod m$
   2. $h \leftarrow 2h \bmod m$

AUSGABE: $a + b \bmod m$

- **Korrektheit:** $a \cdot b = \left(\sum_{i=0}^{n} a_i 2^i\right) \cdot b = \sum_{i=0}^{n} a_i \cdot \left(b \cdot 2^i\right) \bmod m$.
- In Schritt 2.1 wird $h = b \cdot 2^i \bmod m$ addiert.
- **Laufzeit:** $n \cdot \mathcal{O}(n) = \mathcal{O}(n^2) = \mathcal{O}(\log^2 m)$

Komplexität modularer Division

Korollar Modulare Division

Sei $m \in \mathbb{N}$ mit Bitlänge n und $a \in \mathbb{Z}_m$, $b \in \mathbb{Z}_m^*$. Dann kann $\frac{a}{b} \bmod m$ in Zeit $\mathcal{O}(n^2)$ berechnet werden.

Beweis:

- Berechne $b^{-1} \bmod m$ mit EEA in Zeit $\mathcal{O}(\log^2 m)$.
- Berechne $a \cdot b^{-1} \bmod m$ in Zeit $\mathcal{O}(\log^2 m)$.

Die Karatsuba Methode

Satz von Karatsuba

Sei $m \in \mathbb{N}$ mit Bitlänge n und $a, b \in \mathbb{Z}_m$. Dann kann $a \cdot b \bmod m$ in Zeit $\mathcal{O}(n^{\log_2 3})$ berechnet werden.

Beweis:

- Vereinfachende Annahme: Bitlänge ist Zweierpotenz $n = 2^k$.
- Teilen Operanden in der Mitte, d.h. $a = A_1 2^{\frac{n}{2}} + A_0$ mit $A_1 = \sum_{i=\frac{n}{2}}^{n-1} a_i 2^i$ und $A_0 = \sum_{i=0}^{\frac{n}{2}-1} a_i 2^i$. Dann gilt

$$\begin{aligned} a \cdot b &= \left(A_1 2^{\frac{n}{2}} + A_0\right)\left(B_1 2^{\frac{n}{2}} + B_0\right) \\ &= A_1 B_1 2^n + (A_1 B_0 + A_0 B_1) 2^{\frac{n}{2}} + A_0 B_0 \\ &= A_1 B_1 2^n + ((A_0 + A_1)(B_0 + B_1) - A_0 B_0 - A_1 B_1) 2^{\frac{n}{2}} + A_0 B_0 \end{aligned}$$

- D.h. die Multiplikation von n-Bit Zahlen a, b kann zurückgeführt werden auf 3 Multiplikationen von $\frac{n}{2}$-Bit Zahlen.
- Rekursiv: $\frac{n}{2}$-Bit mittels 3 Multiplikationen mit $\frac{n}{4}$-Bit Zahlen, usw.

Laufzeit Karatsuba

Beweis: Laufzeit der Karatsuba-Methode

- Rekursiver Aufruf erfordert Aufwand von 6 Additionen und 2 Shifts.
- Sei $T(n)$ die Laufzeit zum Multiplizieren zweier n-Bit Zahlen.
- Dann gilt $T(n) = 3T(\frac{n}{2}) + c \cdot n$ für festes $c > 0$.

$$
\begin{aligned}
T(n) &= 3\left(3T\left(\frac{n}{4}\right) + c\frac{n}{2}\right) + cn = 3^2 T\left(\frac{n}{4}\right) + cn\left(1 + \frac{3}{2}\right) \\
&= 3^2\left(3T\left(\frac{n}{8}\right) + c\frac{n}{4}\right) + cn\left(1 + \frac{3}{2}\right) \\
&= 3^3 T\left(\frac{n}{8}\right) + cn\left(1 + \frac{3}{2} + \left(\frac{3}{2}\right)^2\right) \\
&= \vdots \\
&= 3^i T\left(\frac{n}{2^i}\right) + cn\sum_{j=0}^{i-1}\left(\frac{3}{2}\right)^j
\end{aligned}
$$

- Abbruch der Rekursion für $T(1)$, d.h. $n = 2^i$ bzw. $i = \log_2 n$.

Laufzeit Karatsuba

Beweis: Fortsetzung Laufzeit Karatsuba-Methode

- Wir erhalten für $i = \log_2 n$ und $T(1) = \mathcal{O}(1)$

$$\begin{aligned} T(n) &= 3^{\log_2 n} \cdot T(1) + cn \sum_{j=0}^{\log_2 n - 1} \left(\frac{3}{2}\right)^j \\ &= n^{\log_2 3} \cdot \mathcal{O}(1) + cn \cdot \frac{\left(\frac{3}{2}\right)^{\log_2 n} - 1}{\frac{3}{2} - 1} \\ &= \mathcal{O}(n^{\log_2 3}) + \mathcal{O}\left(n \cdot \frac{3^{\log_2 n}}{2^{\log_2 n}}\right) \\ &= \mathcal{O}(n^{\log_2 3}) \approx \mathcal{O}(n^{1.59}) \end{aligned}$$

Anmerkung:

- Wir lernen bald ein Verfahren kennen mit Komplexität $\mathcal{O}(n \log n \log\log n) = \mathcal{O}(n^{1+\epsilon})$ für jedes $\epsilon > 0$.

Modulare Exponentiation

Satz Modulare Exponentiation

Sei $m \in \mathbb{N}$ mit Bitlänge n und $a, b \in \mathbb{Z}_m$. Dann kann $a^b \bmod m$ in Zeit $\mathcal{O}(n^3)$ berechnet werden.

Beweis:

Algorithmus SQUARE & MULTIPLY

EINGABE: a, $b = b_{n-1} \dots b_0$, m

1. $c \leftarrow 1$
2. For $i \leftarrow 0$ to $n-1$
   1. If $(b_i = 1)$ then $c \leftarrow c \cdot a \bmod m$
   2. $a \leftarrow a^2 \bmod m$

AUSGABE: $c = a^b \bmod m$

- **Korrektheit:** $a^b = a^{\sum_{i=0}^{n-1} b_i 2^i} = \prod_{i=0}^{n-1} a_i^{b_i 2^i} = \prod_{i=0}^{n-1} \left(a_i^{2^i}\right)^{b_i}$
- **Laufzeit:** $n \cdot \mathcal{O}(n^2) = \mathcal{O}(n^3) = \mathcal{O}(\log^3 m)$

Kleiner Satz von Fermat

Satz von Fermat

Sei p prim. Dann gilt

1. $a^{p-1} = 1 \bmod p$ für alle $a \in \mathbb{Z}_p^*$.
2. $a^p = a \bmod p$ für alle $a \in \mathbb{Z}_p$.

Beweis:

- ad 1: Nach Satz von Euler gilt $a^{|G|} = a^{|\mathbb{Z}_p^*|} = a^{p-1} = 1$.
- ad 2: Für alle $a \in \mathbb{Z}_p^*$ folgt damit $a^p = a \bmod p$.
- Weiterhin gilt für $a = 0$ die Identität $0^p = 0 \bmod p$.