Wiederholung

- Gruppen
  - Ordnung: Gruppe, Element
  - Satz von Euler: $a^{ord(G)} = 1$
  - Elementordung teilt Gruppenordnung
- Untergruppen
  - Satz von Lagrange
    - Untergruppenordnung teilt Gruppenordnung
  - Nebenklassen von Untergruppen
  - Faktorgruppe: G/H
- Gruppenisomorphismen
  - Jede zykl. Gruppe mit m Elementen ist isomorph zu $(\mathbb{Z}_m, +)$.

Beispiel: Faktorgruppe $(\mathbb{Z}/m\mathbb{Z},+)$

- $m\mathbb{Z}=\{0, \pm m, \pm 2m, \ldots\}$ ist additive Untergruppe von $\mathbb{Z}$.

- Verschiedene Nebenklassen von $m\mathbb{Z}$:
  - $0 + m\mathbb{Z} = m\mathbb{Z}$
  - $1 + m\mathbb{Z} = \{1, 1\pm m, 1 \pm 2m, \ldots\}$
  - $2 + m\mathbb{Z} = \{2, 2 \pm m, 2 \pm 2m, \ldots\}$

    …
  - $m-1 + m\mathbb{Z} = \{m-1, m-1\pm m, m-1\pm 2m, \ldots\}$

- Faktorgruppe $(\mathbb{Z}/m\mathbb{Z}, +)$:
  - $\mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z} \to \mathbb{Z}/m\mathbb{Z}$, $(a+m\mathbb{Z}, b+m\mathbb{Z}) \mapsto (a+b) + m\mathbb{Z}$
  - $(a+b) + m\mathbb{Z} = (a+b \bmod m) + m\mathbb{Z}$
  - Isomorphismus $(\mathbb{Z}/m\mathbb{Z},+) \cong (\mathbb{Z}_m,+)$:

    f: $\mathbb{Z}/m\mathbb{Z} \to \mathbb{Z}_m$, $a+m\mathbb{Z} \mapsto a \bmod m$

Komplexität modularer Arithmetik

Ziel: Berechne $c=a+b \bmod m$ für $a,b \in \mathbb{Z}_m$ mit $2^{n-1} \leq m < 2^n$.

Bitdarstellung:

- Schreibe $a = \sum_{i=0}^{n-1} a_i 2^i$ als Bitstring $a_{n-1}a_{n-2}\ldots a_0$ der Länge n

Algorithmus Addition:

Eingabe: $a=a_{n-1}\ldots a_0$, $b=b_{n-1}\ldots b_0$, $m=m_{n-1}\ldots,m_0$

1. c $\leftarrow$ Bitweise Addition von a,b mit Übertrag beginnend mit a_0, b_0.
2. If (c > m) then c $\leftarrow$ Bitweise Subtraktion c-m

Ausgabe: $c=c_n\ldots c_0$

Korrektheit: klar
Laufzeit: $\mathcal{O}(n)=\mathcal{O}(\log m)$

Komplexität der Multiplikation

Ziel: Berechne c=a*b mod m

Algorithmus Multiplikation-Schulmethode

Eingabe: $a=a_{n-1}\ldots a_0$, $b_{n-1}\ldots b_0$, $m=m_{n-1}\ldots m_0$

1. c $\leftarrow$ 0; hilf $\leftarrow$ b;
2. for i $\leftarrow$ 0 to n-1
   1. if (a_i = 1) then c $\leftarrow$ c + hilf mod m;
   2. hilf $\leftarrow$ 2*hilf mod m;

Ausgabe: $c=c_{n-1}\ldots c_0$

Korrektheit: klar

Laufzeit: $n*\mathcal{O}(n) = \mathcal{O}(n^2) = \mathcal{O}(\log^2 m)$

Rekursive Multiplikation

Vereinfachende Annahme: $n=2^k$

- Schreiben $a= A_1*2^{n/2}+A_0$ mit $A_1=\sum_{i=n/2}^{n-1}a_i2^i$, $A_0=\sum_{i=0}^{n/2-1} a_i2^i$

- $a * b = (A_1*2^{n/2}+A_0)(B_1*2^{n/2}+B_0)$
  $= A_1B_1*2^n + (A_1+B_0)(A_0+B_1)*2^{n/2} + A_0B_0$
  $= A_1B_1*2^n + ((A_0+A_1)(B_0+B_1)-(A_0B_0+A_1B_1))*2^{n/2} + A_0B_0$

- 1 Multiplikation von n-Bit Zahlen zurückgeführt auf:
  - 3 Multiplikationen von n/2-Bit Zahlen
  - 6 Additionen + 2 Shifts
  - Rekursionsgleichung:
    $3*T(n/2) + c*n$ für konstantes c und $T(1)=\mathcal{O}(1)$.

Laufzeitanalyse

$T(n) = 3*T(n/2) + c*n$

$= 3*(3*T(n/4) + c*n/2) + c*n = 3^2*T(n/4) + cn(1+3/2)$

$= 3^2*(3*T(n/8) + c*n/4) + cn(1+3/2)$

$= 3^3*T(n/8) + cn(1+3/2+3^2/4) = \ldots$

$= 3^i*T(n/2^i) + cn\sum_{j=0}^{i-1} (3/2)^j$

- Abbruch für $n=2^i$, d.h. $i=\log_2 n$:

$\Rightarrow T(n) = 3^{\log_2 n}*T(1) + cn*2*(3/2^{\log_2 n} - 1)$

$= n^{\log_2 3}*\mathcal{O}(1) + \mathcal{O}(n*n^{\log_2 3}/n) = \mathcal{O}(n^{\log_2 3}) \approx \mathcal{O}(n^{1.59})$

Wir lernen bald ein Verfahren mit asymptotischer Laufzeit:

$\mathcal{O}(n \log n \log\log n) = \mathcal{O}(n^{1+\epsilon})$ für jedes $\epsilon>0$

Schnelle Exponentiation

- Ziel: Berechne $c = a^b \bmod m$ mit $b<m<2^n$
- Beachte: (b-1)-malige Multiplikation hat Laufzeit
  - $\mathcal{O}(bn^2)=\mathcal{O}(2^n n^2)$, d.h. exponentiell in der Bitlänge n

Algorithmus Square and Multiply

Eingabe: $a=a_{n-1}\ldots a_0$, $b=b_{n-1}\ldots b_0$

1. $c \leftarrow 1$
2. for i=0 to n-1
   1. if (b_i=1) then $c \leftarrow c*a \bmod m$
   2. $a \leftarrow a^2 \bmod m$

Ausgabe: c

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 Iterationen mit Laufzeit $\mathcal{O}(n^2)$: Gesamtlaufzeit $\mathcal{O}(n^3)=\mathcal{O}(\log^3 m)$

Kleiner Satz von Fermat

Satz von Euler: Sei G eine multiplikative Gruppe mit neutralem Element 1. Dann gilt für alle $a \in G$:

$$a^{|G|} = 1.$$

Kleiner Satz von Fermat: Sei p prim. Dann gilt für alle $a \in \mathbb{Z}_p^*$:

$$a^{p-1} = 1 \bmod p.$$

Beweis: $|\mathbb{Z}_p^*| = p-1$.

Korollar: Sei p prim. Dann gilt für alle $a \in \mathbb{Z}_p$:

$$a^p = a \bmod p.$$

- Gleichung gilt für alle $a \in \mathbb{Z}_p^*$ nach Kleinem Satz von Fermat:
- Für a=0 gilt $0^p = 0$.

Rückrichtung

- Die umgekehrte Aussage

  $\forall a \in \mathbb{Z}_p^*: a^{p-1}=1 \bmod p \Rightarrow p$ prim

  gilt **nicht!**

- Carmichael Zahlen
  - C = {n | n zusammengesetzt, $a^{n-1}=1$ für alle $a \in \mathbb{Z}_n^*$. }
  - C = {561, 1105, 1729, …}
  - |C| = ∞ (Beweis 1994)

Fermatsche Pseudoprimzahlen

Sei n zusammengesetzt.

- n heisst Fermatsche Pseudoprimzahl zur Basis $a \in \mathbb{Z}_n^*$ falls

$$a^{n-1}=1 \bmod n.$$

Satz: Sei $n \in \mathbb{N}$ zusammengesetzt, keine Carmichael-Zahl. Dann ist n Pseudoprimzahl zu höchstens der Hälfte aller Basen $a \in \mathbb{Z}_n^*$.

- $U=\{a \in \mathbb{Z}_n^* \mid a^{n-1}=1 \bmod n\}$ Untergruppe von $\mathbb{Z}_n^*$
  - Abgeschlossenheit: $a,b \in U \Rightarrow (ab)^{n-1} = a^{n-1} * b^{n-1} = 1 \bmod n$.
  - Neutrales Element: $1 \in U$.
  - Inverses Element zu $a \in U$ ist a^{n-2}:
    - $a * a^{n-2} = a^{n-1} = 1 \bmod n$.
    - $a^{n-2} \in U$, denn $(a^{n-2})^{n-1} = (a^{n-1})^{n-2} = 1 \bmod n$.
- Da n keine Carmichael-Zahl ist, gilt $U \neq \mathbb{Z}_n^*$.
- Satz von Lagrange:

$$|U|*|\text{ind}_U(\mathbb{Z}_n^*)| = |\mathbb{Z}_n^*|, \quad \text{d.h. } |\text{ind}_U(\mathbb{Z}_n^*)| \geq 2, \text{ bzw. } |U| \leq \tfrac{1}{2}|\mathbb{Z}_n^*|$$

Primzahltest für Nicht-Carmichaelzahlen

Algorithmus Fast-Primtest

Eingabe: n, k $\in \mathbb{N}$, n keine Carmichaelzahl

1. for i=1 to k
   1. Wähle a $\in \mathbb{Z}_n \setminus \{0\}$ zufällig
   2. Falls ggT(a, n) > 1 Ausgabe „n zusammengesetzt“.
   3. Falls $a^{n-1} \neq 1 \bmod n$ Ausgabe „n zusammengesetzt“.
2. Ausgabe „n prim“.

- Laufzeit: $\mathcal{O}(k \log^3 n) = \mathcal{O}(\log^3 n)$ für konstantes k
- In der Praxis: k=80 genügt.

Korrektheit

- Falls n prim:
  - Ausgabe ist stets „n prim“.
- Falls n zusammengesetzt:
  - Manchmal fehlerhafte Ausgabe „n prim“.
  - Für alle $a \in \mathbb{Z}_n \setminus \mathbb{Z}_n^*$: Ausgabe korrekt.
  - Falls n keine Pseudoprimzahl zur Basis a: Ausgabe korrekt.
    - a ist Zeuge für Zusammengesetztheit von n.
  - Pro Iteration:
    Ws(Ausgabe „n zusammengesetzt“ | n zusammengesetzt) $\geq$ ½
  - Nach k Iterationen:
    ϵ' = Ws(Ausgabe „n prim“ | n zusammengesetzt)
    = (1 – Ws(Ausgabe „n zusammengesetzt“ | n zusammengesetzt))$^k \leq 2^{-k}$
- Fehlerwahrscheinlichkeit:
  - ϵ = Ws(n zusammengesetzt | Ausgabe „n prim“) $\approx \epsilon'$

Chinesischer Restsatz

Spezieller CRT-Satz: Seien m,n $\in \mathbb{N}$ teilerfremd.

Dann existiert genau eine Lösung x mod mn des Gleichungssystems

$$\left| \begin{array}{rcl} x & = & a \bmod m \\ x & = & b \bmod n \end{array} \right| .$$

Existenz:

- EEA liefert r,s mit mr + ns = 1
  $\Rightarrow$ mr = 1 mod n und ns = 1 mod m
- Sei x = ans + bmr mod mn.
  $\Rightarrow$ x = a mod m und x = b mod n

Eindeutigkeit mod mn: Sei x‘ zweite Lösung.

- x = a = x‘ mod m und x = b = x‘ mod n
  $\Rightarrow$ m | x-x‘ und n | x-x‘
  $\Rightarrow$ mn | x-x‘ (wegen m,n teilerfremd)
  $\Rightarrow$ x = x‘ mod mn