Diskrete Strukturen
Großübung

Amelie Heindl

Lehrstuhl für Logik und Semantik

Technische Universität Berlin

Sommersemester 2024

Themenüberblick

Themenüberblick: Vorlesungswoche 10

- RSA
- Chinesischer Restsatz
- Kleiner Satz von Fermat
- Eulersche φ-Funktion

Modulare Exponentiation

Modulare Exponentiation: Grundlagen

Unter Ausnutzung der Rechenregel zur Produktberechnung, kann auch ein einfaches Vorgehen zum Potenzieren definiert werden.

Wenn $a^e \mod m$ berechnet werden soll, stellt man a^e als Produkt von Faktoren $a^{e_1}, \ldots, a^{e_k}$ dar, wobei alle Exponenten Zweierpotenzen sind und berechnet dann die Zwischenergebnisse modulo m.

Dafür muss nur die Exponentiation mit Zweierpotenzen vorberechnet werden, was mit Rechenregeln und wiederverwenden von Ergebnissen einfach geht.

Modulare Exponentiation: Beispiel

Es soll $5^{98} \mod 11$ berechnet werden.

1. Exponent in Zweierpotenzen zerlegen.

$$\begin{aligned} 5^{98} \bmod 11 &= (5^{64+32+2}) \bmod 11 \\ &= (5^{64} \cdot 5^{32} \cdot 5^{2}) \bmod 11 \\ &= ((5^{64} \bmod 11) \cdot (5^{32} \bmod 11) \cdot (5^{2} \bmod 11)) \bmod 11 \end{aligned}$$

2. Potenzieren mit Zweierpotenzen.

$$\begin{aligned} 5^{2} \bmod 11 &= 25 \bmod 11 = 3 \\ 5^{4} \bmod 11 &= ((5^{2} \bmod 11) \cdot (5^{2} \bmod 11)) \bmod 11 = 3 \cdot 3 \bmod 11 = 9 \\ 5^{8} \bmod 11 &= (9 \cdot 9) \bmod 11 = 81 \bmod 11 = 4 \\ 5^{16} \bmod 11 &= (4 \cdot 4) \bmod 11 = 16 \bmod 11 = 5 \\ 5^{32} \bmod 11 &= (5 \cdot 5) \bmod 11 = 25 \bmod 11 = 3 \\ 5^{64} \bmod 11 &= (3 \cdot 3) \bmod 11 = 9 \bmod 11 = 9 \end{aligned}$$

3. Zwischenergebnisse zusammensetzen.

$$\begin{aligned} ((5^{64} \bmod 11) \cdot (5^{32} \bmod 11) \cdot (5^{4} \bmod 11)) \bmod 11 &= (9 \cdot 3 \cdot 3) \bmod 11 \\ &= 81 \bmod 11 = 4 \end{aligned}$$

Modulare Exponentiation: Algorithmus

Dieses Vorgehen könnte man beispielsweise mit folgendem Algorithmus formalisieren:

Algorithm: Schnelle modulare Exponentiation

input : (x, e, n)
output: $x^e \bmod n$

```
1 begin
2     y := 1;
3     while e > 0 do
4         if e ist ungerade then
5             y := x · y mod n;
6         x := x · x mod n;
7         e := ⌊e/2⌋;
8     return y
```

Beispiel:

Berechnen von $5^{98} \mod 11$

x	e	y
5	98	1
3	49	1
9	24	3
4	12	3
5	6	3
3	3	3
9	1	9
	0	4

Algorithmus von Euklid

Algorithmus von Euklid: Grundlagen

Die Eigenschaften des modulo-Rechnens ermöglichen auch ein einfaches Verfahren zur Berechnung des größten gemeinsamen Teilers zweier ganzer Zahlen.

Algorithm: euklid(a, b)

input : $a, b \in N$ mit $a \leq b$
output: ggT(a, b)

```
1 begin
2     if a | b then
3         return a
4     else
5         return euklid(b mod a, a)
```

Die Eingaben werden in jeden rekursiven Aufruf kleiner, bleiben jedoch natürliche Zahlen. Somit terminiert der Algorithmus.

In der Vorlesung wurde auch die Korrektheit der Ausgabe bewiesen.

Algorithmus von Euklid: Beispiel

Wir wollen nun mit dem Algorithmus von Euklid den größten gemeinsamen Teiler von 80 und 101 berechnen.

Algorithm: euklid(a, b)

input : $a, b \in N$ mit $a \leq b$
output: ggT(a, b)

```
1 begin
2     if a | b then
3         return a
4     else
5         return
              euklid(b mod a, a)
```

- euklid$(80, 101)$
  - $80 \nmid 101$
  - euklid$(21, 80)$
    - $21 \nmid 80$
    - euklid$(17, 21)$
      - $17 \nmid 21$
      - euklid$(4, 17)$
        - $4 \nmid 17$
        - euklid$(1, 4)$
          - $1 \mid 4$
          - **return** 1

Algorithmus von Euklid: Lemma von Bézout

Besonders nützlich ist der Euklidische Algorithmus in einer erweiterten Version, die zusätzliche Ausgaben berechnet. Diese Version findet im RSA-Kryptosystem Anwendung.

Dafür benötigen wir noch folgendes Resultat:

Lemma von Bézout

Seinen $a, b \in \mathbb{Z}$. Dann existieren $x, y \in \mathbb{Z}$ mit $\mathrm{ggT}(a, b) = x \cdot a + y \cdot b$.

Der größte gemeinsame Teiler zweier Zahlen lässt sich also als Linearkombination von diesen mit ganzzahligen Koeffizienten darstellen.
Mit dem erweiterten Euklidischen Algorithmus kann man diese Koeffizienten berechnen.

Algorithmus von Euklid: Erweiterter Algorithmus

Algorithm: erw-euklid(a, b)

input : $a, b \in N$ mit $a \leq b$

output: $x, y \in \mathbb{Z}$ mit $\text{ggT}(a, b) = xa + xb$

```
1 begin
2     if a | b then
3         return (1, 0);
4     else
5         (x', y') := erw-euklid(b mod a, a);
6         x := y' − x' · ⌊b/a⌋;
7         y := x';
8         return (x, y)
```

Algorithmus von Euklid: Erweitertes Beispiel

Wir wollen nun den erweiterten Algorithmus von Euklid für die Zahlen 80 und 101 ausführen.

Algorithm: erw-euklid(a, b)

input : $a, b \in N$ mit $a \leq b$
output: $x, y \in \mathbb{Z}$ mit $\text{ggT}(a, b) = xa + xb$

```
1 begin
2     if a | b then
3         return (1,0);
4     else
5         (x', y') :=
            erw-euklid(b mod a, a);
6         x := y' − x' · ⌊b/a⌋;
7         y := x';
8         return (x, y)
```

- erw-euklid$(80, 101)$
  - $80 \nmid 101$
  - $(x', y') :=$ erw-euklid$(21, 80)$
    - $21 \nmid 80$
    - $(x', y') :=$ erw-euklid$(17, 21)$
      - $17 \nmid 21$
      - $(x', y') :=$ erw-euklid$(4, 17)$
        - $4 \nmid 17$
        - $(x', y') :=$ erw-euklid$(1, 4)$
          - $1 \mid 4$
          - **return** $(1,0)$

Algorithmus von Euklid: Erweitertes Beispiel

Wir wollen nun den erweiterten Algorithmus von Euklid für die Zahlen 80 und 101 ausführen.

Algorithm: erw-euklid(a,b)

input : $a,b \in N$ mit $a \leq b$
output: $x,y \in \mathbb{Z}$ mit $\text{ggT}(a,b) = xa+xb$

```
1 begin
2     if a | b then
3         return (1,0);
4     else
5         (x',y') :=
            erw-euklid(b mod a, a);
6         x := y' − x' · ⌊b/a⌋;
7         y := x';
8         return (x,y)
```

- erw-euklid$(80,101)$
  - $80 \nmid 101$
  - $(x',y') :=$ erw-euklid$(21,80)$
    - $21 \nmid 80$
    - $(x',y') :=$ erw-euklid$(17,21)$
      - $17 \nmid 21$
      - $(x',y') :=$ erw-euklid$(4,17)$
        - $4 \nmid 17$
        - $(x',y') := (1,0)$
        - $x := 0 - 1 \cdot \lfloor \frac{17}{4} \rfloor = -4$
        - $y := 1$
        - **return** (-4,1)

Algorithmus von Euklid: Erweitertes Beispiel

Wir wollen nun den erweiterten Algorithmus von Euklid für die Zahlen 80 und 101 ausführen.

Algorithm: erw-euklid(a,b)

input : $a,b \in N$ mit $a \leq b$
output: $x,y \in \mathbb{Z}$ mit $\text{ggT}(a,b) = xa + xb$

```
1 begin
2     if a | b then
3         return (1,0);
4     else
5         (x',y') :=
            erw-euklid(b mod a, a);
6         x := y' − x' · ⌊b/a⌋;
7         y := x';
8         return (x,y)
```

- erw-euklid$(80,101)$
  - $80 \nmid 101$
  - $(x',y') :=$ erw-euklid$(21,80)$
    - $21 \nmid 80$
    - $(x',y') :=$ erw-euklid$(17,21)$
      - $17 \nmid 21$
      - $(x',y') := (-4,1)$
      - $x := 1 + 4 \cdot \lfloor \frac{21}{17} \rfloor = 5$
      - $y := -4$
      - **return** (5,-4)

Algorithmus von Euklid: Erweitertes Beispiel

Wir wollen nun den erweiterten Algorithmus von Euklid für die Zahlen 80 und 101 ausführen.

Algorithm: erw-euklid(a, b)

input : $a, b \in N$ mit $a \leq b$
output: $x, y \in \mathbb{Z}$ mit $\text{ggT}(a, b) = xa + xb$

```
1 begin
2     if a | b then
3         return (1,0);
4     else
5         (x', y') :=
            erw-euklid(b mod a, a);
6         x := y' − x' · ⌊b/a⌋;
7         y := x';
8         return (x, y)
```

- erw-euklid$(80, 101)$
  - $80 \nmid 101$
  - $(x', y') :=$ erw-euklid$(21, 80)$
    - $21 \nmid 80$
    - $(x', y') := (5, -4)$
    - $x := -4 - 5 \cdot \lfloor \frac{80}{21} \rfloor = -19$
    - $y := 5$
    - **return** (-19,5)

Algorithmus von Euklid: Erweitertes Beispiel

Wir wollen nun den erweiterten Algorithmus von Euklid für die Zahlen 80 und 101 ausführen.

Algorithm: erw-euklid(a, b)

input : $a, b \in N$ mit $a \leq b$
output: $x, y \in \mathbb{Z}$ mit $\text{ggT}(a, b) = xa + xb$

```
1 begin
2     if a | b then
3         return (1,0);
4     else
5         (x', y') :=
            erw-euklid(b mod a, a);
6         x := y' − x' · ⌊b/a⌋;
7         y := x';
8         return (x, y)
```

- erw-euklid$(80, 101)$
  - $80 \nmid 101$
  - $(x', y') := (-19, 5)$
  - $x := 5 + 19 \cdot \lfloor \frac{101}{80} \rfloor = 24$
  - $y := -19$
  - **return** (24,-19)

Chinesischer Restsatz

Chinesischer Restsatz: Simultane Kongruenzen

Bei der Betrachtung von modulo m haben wir gesehen, dass zwei Zahlen kongruent sind, wenn sie beim Teilen durch m den gleichen Rest ergeben. Kongruent modulo m zu sein bedeutet also so etwas wie gleich zu sein, nachdem man alle Faktoren m entfernt hat.
Ähnlich zu linearen Gleichungssystemen, kann man Kongruenzsysteme (genannt simultane Kongruenzen) definieren.

Simultane Kongruenz

Seien $m_1, \ldots, m_n \geq 2$ und $a_1, \ldots, a_n \in \mathbb{Z}$. Dann ist

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x &\equiv a_n \pmod{m_n} \end{aligned}$$

eine simultane Kongruenz.

中国剩余定理：同余方程组

在考虑模数 m 时，我们看到，如果两个数字在除以 m 时余数相同，那么它们是同余的。因此，模数 m 同余意味着在去掉所有模数 m 的因数后是相同的。类似于线性方程组，我们可以定义同余方程组（称为同余方程组）。

Chinesischer Restsatz: Simultane Kongruenzen

Beispiel: Wir betrachten die simultane Kongruenz

$$x \equiv 2 \pmod 3$$
$$x \equiv 4 \pmod 7$$

Die erste Kongruenz wird beispielsweise erfüllt von

$$\ldots, 2, 5, 11, 14, 17, 20, \ldots$$

Die zweite Kongruenz wird beispielsweise erfüllt von

$$\ldots, 4, 11, 18, 25, 32, 39, \ldots$$

Eine Lösung ist $x = 11$.

Dies sind genau die Äquivalenzklassen von 2 in der modulo-3-Relation und von 4 in der modulo-7-Relation.

Chinesischer Restsatz: Aussage

Nun ist die Frage naheliegend, ob und wieviele Lösungen jede simultane Kongruenz hat. Für gewisse Werte liefert der chinesische Restsatz eine Antwort.

$a, b \in \mathbb{N}$ sind genau dann teilerfremd, wenn $\text{ggT}(a, b) = 1$ gilt.

Chinesischer Restsatz

Seien $a_1, \ldots, a_n, m_1, \ldots, m_n \in \mathbb{N}$, wobei $m_1, \ldots, m_n$ paarweise teilerfremd sind. Sei weiter $m = \prod_{i=1}^{n} m_i$. Dann existiert genau ein $x \in \mathbb{Z}_m = \{0, \ldots, m-1\}$ mit

$$x \equiv a_i \pmod{m_i} \quad \text{für alle } i \in \{1, \ldots, n\}$$

Wenn wir also eine Zahl suchen, die jeweils modulo m_i in der gleichen Äquivalenzklasse liegt wie a_i, finden wir genau eine passende Äquivalenzklasse modulo dem Produkt aller m_i, sofern diese teilerfremd sind.

Chinesischer Restsatz: Beispiel

Der Beweis des Chinesischen Restsatzes zeigt nicht nur die Existenz der eindeutigen Lösung, sondern liefert auch ein Verfahren, um diese zu berechnen. Dieses Vorgehen betrachten wir nun an einem Beispiel.

Wir betrachten die folgende simultane Kongruenz:

$$x \equiv 1 \pmod 3$$
$$x \equiv 3 \pmod 5$$

Also Lösung findet sich also $x = 13$.

Wir definieren die Bijektion:

$$\mu : \mathbb{Z}_{15} \to \mathbb{Z}_3 \times \mathbb{Z}_5$$
$$x \mapsto (x \bmod 3, x \bmod 5)$$

Diese lässt sich darstellen als:

$\mathbb{Z}_3$ \ $\mathbb{Z}_5$	0	1	2	3	4
0	0	6	12	3	9
1	10	1	7	13	4
2	5	11	2	8	14

$\mathbb{Z}_{15}$

Kleiner Satz von Fermat

Kleiner Satz von Fermat: Grundlagen

Kleiner Satz von Fermat

Für alle $n \in \mathbb{N}$ mit $n \geq 2$ gilt:
n ist eine Primzahl gdw. $a^{n-1} \equiv 1 \pmod{n}$ für alle $a \in \mathbb{Z}_n \setminus \{0\}$.

Beispiel: Wir können den Satz nutzen, um Rechnungen modulo einer Primzahl zu vereinfachen. Es gilt

$$\begin{aligned} 3^{26} &\equiv 3^{12+12+2} \bmod 13 \\ &\equiv 3^{12} \cdot 3^{12} \cdot 3^{2} \bmod 13 \\ &\equiv 1 \cdot 1 \cdot 9 \bmod 13 \\ &\equiv 9 \bmod 13 \end{aligned}$$

Und damit folgt $3^{26} \bmod 13 = 9$.

Kleiner Satz von Fermat: Primzahltest

Da der kleine Satz von Fermat eine Bedingung für die Primalität von Zahlen liefert, kann man ihn für einen Primzahltest verwenden.

Ablauf:

- Die Eingabe ist eine Zahl $n \geq 3$.
- Eine Zahl $a \in \{2, \ldots, n-1\}$ wird zufällig gewählt.
- Falls $a^{n-1} \equiv 1 \pmod{n}$ gilt, ist die Ausgabe "n ist möglicherweise prim", andernfalls "n ist zusammengesetzt".

Falls "n ist zusammengesetzt" ausgegeben wird, ist n mit Sicherheit nicht prim. Falls "n ist möglicherweise prim" ausgegeben wird, ist nicht sicher, dass n prim ist. Wiederholt man in diesem Fall den Test mit einem neuen a und bekommt wieder die Ausgabe "n ist möglicherweise prim", steigt die Wahrscheinlichkeit dafür, dass n tatsächlich prim ist.

Eulersche φ-Funktion

Eulersche φ-Funktion: Grundlagen

Die Eulersche φ-Funktion gibt an, wie viele kleinere teilerfremde natürliche Zahlen es zu einer Zahl gibt.

Eulersche φ-Funktion

Sei $n \geq 2$. Dann ist $\varphi(n) = |\{a \in \mathbb{Z}_n \setminus \{0\} \mid \text{ggT}(a, n) = 1\}|$.

Eulersche φ-Funktion: Grundlagen

Eine einfache Berechnungsmethode für φ ist bekannt.

Primzahlen p haben nur sich selbst und 1 als positive Teiler, somit sind sie teilerfremd zu allen Zahlen in $\{1, \ldots, p-1\}$ und es gilt

$$\varphi(p) = p - 1$$

.

Primzahlpotenzen p^e haben nur die Vielfachen von p als Teiler und es gilt

$$\varphi(p^e) = p^e - p^{e-1} = (p-1)p^{e-1}$$

.

Allgemein gilt für eine Zahl n mit der Primfaktorzerlegung $n = \prod_{i=1}^{k} p_i^{e_i}$:

$$\varphi(n) = \prod_{i=1}^{k} (p_i - 1)p^{e_i - 1}$$

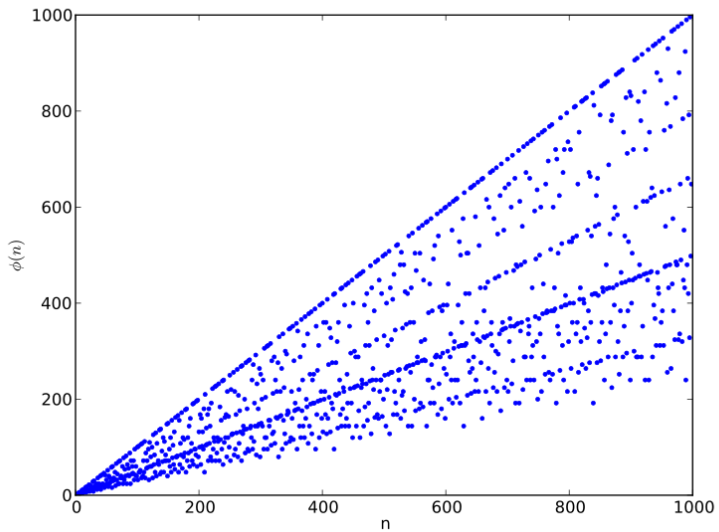
.

Eulersche φ-Funktion: Beispiele

Wir berechnen nun $\varphi(n)$ für einige Zahlen n:

- $\varphi(30) = \varphi(2) \cdot \varphi(3) \cdot \varphi(5)$
  $= 1 \cdot 2 \cdot 4 = 8$
- $\varphi(125) = \varphi(5^3) = 4 \cdot 5^2 = 100$
- $\varphi(484) = \varphi(2^2) \cdot \varphi(11^2)$
  $= 1 \cdot 2 \cdot 10 \cdot 11 = 220$

$\varphi(n)$ für $n \in \{2, \ldots, 1000\}$:



Source: https://de.wikipedia.org/wiki/Eulersche_Phi-Funktion

Eulersche φ-Funktion: Satz von Euler

Mithilfe der φ-Funktion können wir nun ein Ähnliches Resultat wie den kleinen Satz von Fermat für eine größere Menge von Zahlen formulieren.

Satz von Euler

Für alle $n \in \mathbb{N}$ mit $n \geq 2$ gilt:

$$a^{\varphi(n)} \equiv 1 \pmod{n} \qquad \text{für alle } a \in \mathbb{Z}_n^* := \{a \in \mathbb{Z}_n \setminus \{0\} \mid \text{ggT}(a, n) = 1\}.$$

Wir haben bereits gesehen, dass für Primzahlen p folgendes gilt:

$$\varphi(p) = p - 1$$

Das zeigt, dass eine Richtung des kleinen Satzes von Fermat nur ein Spezialfall des Satzes von Euler ist mit $n \in \mathbb{P}$.

RSA

RSA: Grundlagen

Das RSA-Verfahren ist ein asymmetrisches Public-key Kryptosystem.
Es ermöglicht einen Austausch von verschlüsselten Nachrichten, wobei ein öffentlicher und ein privater Schlüssel verwendet werden.
Der Ablauf des Protokolls besteht aus drei Teilen.

1. Schlüsselerzeugung:
   - Der Empfänger wählt Primzahlen p, q mit $p \neq q$ und berechnet $n = p \cdot q$ und $\varphi(n) = (p-1)(q-1)$
   - Er wählt $k \in \mathbb{N}$, für das $\mathrm{ggT}(\varphi(n), k) = 1$ gilt.
   - Er findet $\ell \in \mathbb{N}$, für das $k \cdot \ell \equiv 1 (\bmod \varphi(n))$.
   - (n, k) ist nun der öffentliche Schlüssel und ℓ der private Schlüssel.
2. Verschlüsseln:
   - Der Sender will Nachricht $M \in \{0, \ldots, n-1\}$ schicken.
   - Er berechnet $S := M^k \bmod n$ und sendet S.
3. Entschlüsseln:
   - Der Empfänger empfängt S und berechnet $M' := S^\ell \bmod n$.

Die Sicherheit des RSA-Verfahrens beruht auf der Annahme, dass n nicht effizient faktorisiert werden kann.

RSA: Beispiel

Wir betrachten einen beispielhaften Ablauf des RSA-Verfahrens:

1.
   - Wir wählen $p = 7, q = 11$ und berechnen $n = 77$, $\varphi(77) = 60$.
   - Wir wählen $k = 13$ (offensichtlich gilt $\text{ggT}(60, 13) = 1$).
   - Mit dem erweiterten Euklidischen Algorithmus finden wir $\ell = 37$.

Schlüsselerzeugung:

- Der Empfänger wählt Primzahlen p, q mit $p \neq q$ und berechnet $n = p \cdot q$ und $\varphi(n) = (p-1)(q-1)$
- Er wählt $k \in \mathbb{N}$, für das $\text{ggT}(\varphi(n), k) = 1$ gilt.
- Er findet $\ell \in \mathbb{N}$, für das $k \cdot \ell \equiv 1 (\bmod \varphi(n))$.
- (n, k) ist nun der öffentliche Schlüssel und ℓ der private Schlüssel.

RSA: Beispiel

Wir betrachten einen beispielhaften Ablauf des RSA-Verfahrens:

1.
   - Wir wählen $p = 7, q = 11$ und berechnen $n = 77$, $\varphi(77) = 60$.
   - Wir wählen $k = 13$ (offensichtlich gilt $\mathrm{ggT}(60, 13) = 1$).
   - Mit dem erweiterten Euklidischen Algorithmus finden wir $\ell = 37$.
2.
   - Wir wählen $M = 4$.
   - Wir berechnen und senden $S = 4^{13} \bmod 77 = 53$.

Verschlüsseln:

- Der Sender will Nachricht $M \in \{0, \ldots, n-1\}$ schicken.
- Er berechnet $S := M^k \bmod n$ und sendet S.

RSA: Beispiel

Wir betrachten einen beispielhaften Ablauf des RSA-Verfahrens:

1.
   - Wir wählen $p = 7, q = 11$ und berechnen $n = 77$, $\varphi(77) = 60$.
   - Wir wählen $k = 13$ (offensichtlich gilt $\text{ggT}(60, 13) = 1$).
   - Mit dem erweiterten Euklidischen Algorithmus finden wir $\ell = 37$.
2.
   - Wir wählen $M = 4$.
   - Wir berechnen und senden $S = 4^{13} \bmod 77 = 53$.
3.
   - Wir empfangen $S = 53$.
   - Wir berechnen $M' = 53^{37} \bmod 77 = 4$.

Entschlüsseln:

- Der Empfänger empfängt S und berechnet $M' := S^{\ell} \bmod n$.

Feedback, Fragen und Vorschläge zur Großübung gerne an:

a.heindl@tu-berlin.de