

Vorlesung

Effiziente Algorithmen und Komplexitätstheorie

Sommersemester 2008

Ingo Wegener

Über Primzahltests

ein neues Problem zu $q \in \mathbb{N}$ **entscheide**
 Ist q eine Primzahl?

Erinnerung p prim $\Leftrightarrow |\{r \in \mathbb{N} \mid (r \mid q)\}| = 2$

Ein einfacher Algorithmus

1. For $i \in \{2, 3, \dots, q-1\}$
2. If $i \mid q$ Then Ausgabe „ q ist keine Primzahl“; STOP.
3. Ausgabe „ q ist Primzahl“

Beobachtung $(\exists 1 < r < \lfloor \sqrt{q} \rfloor : r \mid q)$
 $\Leftrightarrow (\exists \lfloor \sqrt{q} \rfloor < r' < q : r' \mid q)$

Ein besserer einfacher Algorithmus

1. For $i \in \{2, 3, \dots, \lfloor \sqrt{q} \rfloor\}$
2. If $i \mid q$ Then Ausgabe „ q ist keine Primzahl“; STOP.
3. Ausgabe „ q ist Primzahl“

Noch ein Algorithmus

Beispiel für $q = 167$ Sieb des Eratosthenes

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52
53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86
87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103
104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137
138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154
155	156	157	158	159	160	161	162	163	164	165	166	167				

Ein Blick auf den besseren einfachen Primzahltest

Beobachtungen

- offensichtlich korrekt
- liefert sogar zusätzlich Teiler von q
- Laufzeit $\Theta(\sqrt{q})$

Also Problem gelöst?

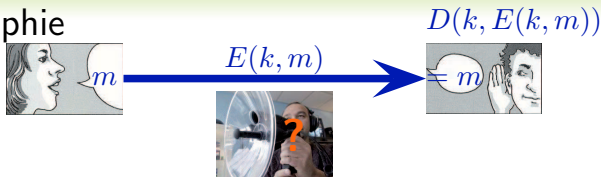
Beobachtung bei Eingabe q ist Eingabelänge = $\Theta(\log q)$

also Laufzeit $\Theta(\sqrt{q}) = \Theta\left((\log q)^{\frac{1}{2} \cdot (\log q) / \log \log q}\right)$
exponentiell in der Eingabelänge

Hinweis vielleicht intuitiv klarer bei Vorstellung
 Eingabe $q \in \{1, 2, \dots, 9\} \cdot \{0, 1, \dots, 9\}^*$
 mit gelegentlicher Interpretation als $q \in \mathbb{N}$

Aber wen interessieren so riesig große Zahlen?

Kryptographie



Definition 12.1

Ein kryptographisches System besteht aus den folgenden Komponenten.

- endliche Schlüsselmenge K
- Nachrichtenmenge M
- Menge der Kryptogramme Y
- Chiffrierfunktion $E: K \times M \rightarrow Y$
- Dechiffrierfunktion $D: K \times Y \rightarrow M$
mit $\forall k \in K: \forall m \in M: D(k, E(k, m)) = m$

Moderne Kryptographie

Wünsche

- E effizient berechenbar
- D effizient berechenbar
- bei Kenntnis von $E(k, m)$, E und D
trotzdem m nicht effizient berechenbar

Problem sicherer Austausch des **geheimen** Schlüssels k

Anmerkung bis 1976 in der Tat ein Problem

Diffie, Hellmann (1976): New Directions in Cryptography.

zentrale Idee Mache Schlüssel k **öffentlich**

↪ **Kryptosysteme mit öffentlichem Schlüssel**
public key cryptography

Kryptosysteme mit öffentlichem Schlüssel

Ablauf verschlüsselte Kommunikation $A \rightarrow B$

- ① A und B einigen sich **öffentlich** auf ein Kryptosystem.
- ② B erzeugt Schlüssel k und **geheime Zusatzinformation** $l(k)$.
- ③ B **veröffentlicht** seinen Schlüssel.
- ④ A hat **geheime Nachricht** m und berechnet Kryptogramm $y = E(k, m)$.
- ⑤ A schickt B **öffentlich** $E(k, m)$.
- ⑥ B berechnet $D(k, y)$ **mit Hilfe von** $l(k)$.

Wünsche

- E effizient berechenbar
- D effizient berechenbar bei Kenntnis von y , k und $l(k)$
- D **nicht** effizient berechenbar bei Kenntnis von y und k **ohne Kenntnis von** $l(k)$

Das RSA-Kryptosystem – Schlüsselerzeugung

Rivest, Shamir, Adleman (1978): A Method of Obtaining Digital Signatures and Public-Key Cryptosystems.

- 1 Erzeuge zwei Primzahlen $p, q > 2^l$ mit $l + 1$ Bits **geheim**.
- 2 Berechne $n := p \cdot q$.
- 3 Berechne Eulerfunktion
 $\phi(n) := |\{a \in \{1, 2, \dots, n - 1 \mid \text{ggT}(a, n) = 1\}|$ **geheim**.
Anmerkung $\phi(n) = (p - 1) \cdot (q - 1)$ weil $n = p \cdot q$ und p, q prim
- 4 Berechne Zufallszahl $e \in \{2, 3, \dots, n - 1\}$ mit
 $\text{ggT}(e, \phi(n)) = 1$.
- 5 Berechne $d \equiv e^{-1} \pmod{\phi(n)}$ **geheim**.
- 6 **Veröffentliche** Schlüssel $k = (n, e)$.
- 7 **Geheime Information** ist $l(k) = (\phi(n), d)$.

Das RSA-Kryptosystem – Anwendung

Wir haben öffentlich Schlüssel $k = (n, e)$
 geheim Zusatzinformation $l(k) = (\phi(n), d)$

Erinnerung $n = p \cdot q$ (p, q jeweils Länge $l + 1$)
 $\phi(n) = |\{a \in \{1, 2, \dots, n - 1 \mid \text{ggT}(a, n) = 1\}|$
 $d \equiv e^{-1} \pmod{\phi(n)}$

Sender A hat geheime Nachricht $m \in \{0, 1\}^*$

Darstellung $m = \underbrace{m_1}_{\in \{0,1\}^{2l}} \underbrace{m_2}_{\in \{0,1\}^{2l}} \cdots \underbrace{m_j}_{\in \{0,1\}^{2l}}$

Interpretation $\forall i \in \{1, 2, \dots, j\}: m_i \in \mathbb{N}_0$ mit $0 \leq m_i < 2^{2l}$

Sender A schickt öffentlich $\forall i: y_i = E((n, e), m_i) \equiv m_i^e \pmod{n}$

Empfänger B dechiffriert geheim $\forall i: D((n, d), y_i) \equiv y_i^d \pmod{n}$

Beobachtung Kenntnis von $m_i \pmod{n}$ ausreichend
 weil $m_i < 2^{2l}$ und $n \geq 2^{2l}$

Das RSA-Kryptosystem – Sicherheit

Beobachtung Wenn aus n (mit $n = p \cdot q$) p und q **effizient** berechenbar
dann $l(k) = (\phi(n), d)$ effizient berechenbar
also RSA-Kryptosystem **nicht sicher**

also Sicherheit des RSA-Kryptosystems
beruht auf Schwierigkeit der Primfaktorzerlegung

Anmerkung bei Kenntnis von n (mit $n = p \cdot q$) und $\phi(n)$
sind p, q **effizient** berechenbar

$$\text{denn } \phi(n) = (p - 1) \cdot (q - 1) = (p - 1) \cdot \left(\frac{n}{p} - 1\right)$$

$$= n - p - \frac{n}{p} + 1$$

$$\text{also } p \cdot \phi(n) = np - p^2 - n + p$$

$$\text{also } p^2 + p \cdot (\phi(n) - n - 1) + n = 0$$

$$\text{also Auflösen von } x^2 + x \cdot (\phi(n) - n - 1) + n = 0$$

nach x liefert $p \checkmark$

Das RSA-Kryptosystem – Implementierung

Was brauchen wir für eine effiziente Implementierung?

Voraussetzung Wir rechnen alles $\text{mod } m$ ($m = n = 2^{2l}$).

Kostenmodell $a \text{ mod } m$ sowie Addition, Subtraktion, ..
mit solchen Zahlen **in Zeit $O(1)$ berechenbar**
Darstellung von m braucht $\lfloor \log m \rfloor + 1$ Bits

- Addition, Subtraktion, Multiplikation, Division
für Zahlen $< m = 2^{2l}$ ✓
- Potenzieren $\text{mod } m$
- $\text{ggT}(a, b)$ für $a, b \leq m$
- multiplikative Inverse $\text{mod } m$
- zufällige Primzahlen p mit $m/2 \leq p \leq m$

also Beschäftigung mit Zahlentheorie **sinnvoll**
Algorithmen für Primzahlen erforderlich

Schnelles Potenzieren

Aufgabe Berechne $x^n \bmod m$.

Algorithmus 12.2

Eingabe $x, n \in \mathbb{N}$

Ausgabe x^n

1. $r := 1$
2. While $n > 1$
3. If n ungerade Then
4. $r := r \cdot x$
5. $n := n - 1$
6. $x := x \cdot x$
7. $n := n/2$
8. Ausgabe $x \cdot r$

Theorem 12.3

Algorithmus 12.2 berechnet x^n mit $\leq 1 + 2 \log n$ Multiplikationen.

Beweis der Korrektheit von Algorithmus 12.2

mit vollständiger Induktion

zu zeigen $\forall t: r_t \cdot x_t^{n_t} = x^n$

mit $r_0 := 1, x_0 := x, n_0 := n$

und Index wächst mit jedem While-Durchlauf

Induktionsanfang $t = 0$

Beobachtung $r_0 \cdot x_0^{n_0} = 1 \cdot x^n = x^n \checkmark$

Induktionsschritt

1. Fall $n_t > 1$ gerade

Beobachtung $r_{t+1} = r_t, x_{t+1} = x_t \cdot x_t, n_{t+1} = n_t/2$

also $r_{t+1} \cdot x_{t+1}^{n_{t+1}} = r_t \cdot (x_t \cdot x_t)^{n_t/2} = r_t \cdot x_t^{n_t} = x^n \checkmark$

2. Fall $n_t > 1$ ungerade

Beobachtung $r_{t+1} = r_t \cdot x_t, x_{t+1} = x_t \cdot x_t, n_{t+1} = (n_t - 1)/2$

also $r_{t+1} \cdot x_{t+1}^{n_{t+1}} = r_t \cdot x_t \cdot (x_t \cdot x_t)^{(n_t-1)/2} = r_t \cdot x_t \cdot x_t^{n_t-1}$
 $= r_t \cdot x_t^{n_t} = x^n \checkmark$

Beweis der Laufzeitschranke für Algorithmus 12.2

mit vollständiger Induktion über n

zu zeigen #Multiplikationen in While-Schleife = $M(n) \leq 2 \log n$

Induktionsanfang $n = 1$

Beobachtung $M(n) = 0 = 2 \log 1 = 2 \log n$ ✓

Induktionsschritt

Beobachtung $M(n) = \max \{1 + M(n/2), 2 + M((n-1)/2)\}$
 $< 2 + 2 \log(n/2) = 2 + 2(\log(n) - 1) = 2 \log n$ ✓

□

klar alle Berechnungen auch mod m ausführbar

dann alle Zahlen nicht zu groß ✓

Schnelle ggT-Berechnung

Algorithmus 12.4 (Euklidischer Algorithmus)

Eingabe $a \geq b \in \mathbb{N}$

Ausgabe $\text{ggT}(a, b)$

1. $c := a \bmod b$
2. If $c = 0$
3. Then Ausgabe b
4. Else Ausgabe $\text{ggT}(b, c)$

Theorem 12.5

Der euklidische Algorithmus berechnet $\text{ggT}(a, b)$ in Zeit $O(\log(a + b))$. Die Anzahl der modulo-Operationen ist $\leq \left\lceil \log_{3/2}(a + b) \right\rceil$.

Korrektheit des euklidischen Algorithmus

Beobachtung $\exists k \in \mathbb{N}: a = k \cdot b + c$
mit $c \in \{0, 1, \dots, b-1\}$

1. Fall $c = 0$

dann $b \mid a \rightsquigarrow \text{ggT}(a, b) = b$ ✓

2. Fall $c > 0$

dann **zu zeigen** $\text{ggT}(a, b) = \text{ggT}(b, c)$

Sei $r = \text{ggT}(b, c)$

Beobachtung $r \mid a = k \cdot b + c = r \cdot (k \cdot \frac{b}{r} + \frac{c}{r})$

Annahme $\exists r' > r: r' = \text{ggT}(a, b)$

Beobachtung $r' \mid a = k \cdot b + c$ und $r' \mid b$
 $\Rightarrow r' \mid c$ und $r' \mid b$

also $r' > r$ ist gemeinsamer Teile von b und c

also $r \neq \text{ggT}(b, c)$ **Widerspruch**

also $\text{ggT}(a, b) = \text{ggT}(b, c)$ ✓

Beweis Laufzeitschranke für den euklidischen Algorithmus

zu zeigen $\# \text{modulo-Operationen} \leq \lfloor \log_{3/2}(a+b) \rfloor$

Beweis mit vollständiger Induktion über $a+b$

Induktionsanfang $a+b \leq 2$

also $a=b=1$

Beobachtung

$\# \text{mod.-Op.} = 1 = \lfloor \log_{3/2} 2 \rfloor = \lfloor \log_{3/2}(a+b) \rfloor$ ✓

Induktionsschluss

Beobachtung $\# \text{mod.-Op.} = 1 + \lfloor \log_{3/2}(b+c) \rfloor$

Behauptung $b+c \leq \frac{2}{3}(a+b)$

dann $\# \text{mod.-Op.} = 1 + \lfloor \log_{3/2}(b+c) \rfloor$
 $\leq 1 + \lfloor \log_{3/2}(a+b) - 1 \rfloor = \lfloor \log_{3/2}(a+b) \rfloor$ ✓

Beweis von Theorem 12.5

Wir haben $c = a \bmod b$

zu zeigen $b + c \leq \frac{2}{3}(a + b)$

Beobachtung $c \leq a - b$

also $a \geq c + b$

Beobachtung $b > c$

also $b = \frac{b}{2} + \frac{b}{2} > \frac{b}{2} + \frac{c}{2} = \frac{b+c}{2}$

zusammen $a + b > c + b + \frac{b+c}{2} = \frac{3}{2} \cdot (b + c)$

also $b + c \leq \frac{2}{3}(a + b)$



Multiplikative Inverse

oft nützlich zu $a \geq b \in \mathbb{N}$ mit $\text{ggT}(a, b) = 1$
 multiplikatives Inverse $b^{-1} \bmod a$ berechnen können

Algorithmus 12.6

Eingabe $a \geq b \in \mathbb{N}$ mit $\text{ggT}(a, b) = 1$

Ausgabe $b^{-1} \bmod a$

1. If $b = 1$ Then Ausgabe 1. STOP.
2. $c := a \bmod b$
3. $c^* := c^{-1} \bmod b$
4. Ausgabe $\left(\frac{1-c^* \cdot a}{b}\right) \bmod a$

Theorem 12.7

Algorithmus 12.6 berechnet $b^{-1} \bmod a$ in Zeit $O(\log(a + b))$.

Korrektheit von Algorithmus 12.6

mit vollständiger Induktion über b

Induktionsanfang $b = 1$

Beobachtung Ausgabe 1

Beobachtung $b \cdot 1 = 1 \cdot 1 \equiv 1 \pmod{a}$ ✓

Induktionsschritt

Beobachtung Ausgabe $\left(\frac{1-c^* \cdot a}{b}\right) \pmod{a}$

Beobachtung $c = a \pmod{b}$

also $\exists k: a = k \cdot b + c$ mit $c \in \{0, 1, \dots, b-1\}$

also $b > c$

Beobachtung $c > 0$ sonst $\text{ggT}(a, b) = b > 1$ Widerspruch

Erinnerung $\text{ggT}(b, c) = \text{ggT}(a, b) = 1$

also rekursiver Aufruf in Zeile 3 zulässig ✓

Beweis von Theorem 12.7

zu zeigen $\left(\frac{1-c^* \cdot a}{b}\right) \bmod a = b^{-1} \bmod a$
mit $c^* = c^{-1} \bmod b$

Beobachtung $c^* = c^{-1} \bmod b$
 $\Rightarrow \exists k' \in \mathbb{Z}: c \cdot c^* + k' \cdot b = 1$

Beobachtung $1 - c^* \cdot a = 1 - c^* \cdot (k \cdot b + c) = 1 - c^* \cdot k \cdot b - c^* \cdot c$
 $= c \cdot c^* + k' \cdot b - c^* \cdot k \cdot b - c^* \cdot c = b \cdot (k' - c^* \cdot k)$

also Ausgabe $\left(\frac{1-c^* \cdot a}{b}\right) \bmod a$ ganzzahlig ✓

Beobachtung

$\left(\frac{1-c^* \cdot a}{b}\right) \cdot b \bmod a \equiv 1 - c^* \cdot a \bmod a \equiv 1 \bmod a$ ✓

also Ausgabe korrekt ✓

Laufzeit analog zur ggT-Berechnung $O(\log(a+b))$



Zur Entspannung

ein konkretes Beispiel

Berechne $5^{-1} \bmod 11$ mit Algorithmus 12.6

Beobachtung $\text{ggT}(11, 5) = 1$, $11 \geq 5$

also Algorithmus 12.6 **anwendbar** ✓

1. Schritt $c := a \bmod b = 11 \bmod 5 = 1$

2. Schritt rekursiver Aufruf mit $a = 5, b = 1$
klar $1^{-1} = 1$, also $c^* = 1$

3. Schritt Ausgabe $\left(\frac{1-c^* \cdot a}{b}\right) \bmod a = \left(\frac{1-1 \cdot 11}{5}\right) \bmod 11$
 $= \frac{-10}{5} \bmod 11 = -2 \bmod 11 \equiv 9 \bmod 11$

Probe $9 \cdot 5 \bmod 11 = 45 \bmod 11 \equiv 1 \bmod 11$ ✓

Auf dem Weg zur Berechnung von Primzahlen

Erinnerung Wir **glauben**, dass Faktorisieren schwierig,
sonst RSA **unsicher**

Hoffnung Problem PRIMES **Test „ q prim?“**
einfacher als Faktorisierung

Fakt PRIMES \in P
Agrawal, Kayal, Saxena (2002): PRIMES is in P

Idee Wähle uniform zufällig Zahl q .
Teste, ob q prim ist.

Ist das nicht zu primitiv? – Dauert das nicht zu lange?

Über die Häufigkeit von Primzahlen

Wie oft muss man im Durchschnitt uniform zufällig eine Zahl wählen, bis man auf eine Primzahl stößt?

Betrachte $\pi(n) := |\{p \leq n \mid p \text{ prim}\}|$

Man kann zeigen

- $\lim_{n \rightarrow \infty} \pi(n) \cdot \frac{n}{\ln n} = 1$
- $\forall n \geq 2: \frac{n}{\log n} - 2 \leq \pi(n) \leq \frac{3n}{\log n}$

also
$$E(\#\text{Versuche}) \leq \left(\frac{(n/\log n) - 2}{n}\right)^{-1}$$

$$= \left(\frac{1}{\log n} - \frac{1}{n}\right)^{-1} = \frac{n \log n}{n - \log n} = \Theta(\log n)$$

also bei effizientem Primzahltest **sehr effizient**

Fakt in der Praxis AKS-Algorithmus **zu langsam**
 darum schneller randomisierter Test mit einseitigem Fehler

Mitteilung Kenntnisse in Zahlentheorie **hilfreich** und **erforderlich**

Zum Aufwärmen und Beruhigen

Erinnerung untere Schranke für $\pi(n)$ für uns **wichtig**

Ziel Selbst untere Schranke für $\pi(n)$ beweisen.

vorab

Definition 12.8

Für $m \in \mathbb{N}$ und Primzahl p ist $\nu_p(m) := \max \{k \mid (p^k \mid m)\}$.

Beobachtung $m = \prod_{\text{Primzahl } p \mid m} p^{\nu_p(m)}$

Lemma 12.9

$\forall n \in \mathbb{N}, p \text{ prim: } \nu_p(n!) = \sum_{k \geq 1} \left\lfloor \frac{n}{p^k} \right\rfloor$

Beweis von Lemma 12.9

zu zeigen $\nu_p(n!) \geq \sum_{k \geq 1} \left\lfloor \frac{n}{p^k} \right\rfloor$
 mit $\nu_p(n!) = \max \{k \mid (p^k \mid n!)\}$

Betrachte $R_{p,n} := \{(i, k) \mid i \in \{1, 2, \dots, n\} \text{ und } p^k \mid i\}$

Beispiel für $p = 2, n = 14$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$k = 1$	0	1	0	1	0	1	0	1	0	1	0	1	0	1
2	0	0	0	1	0	0	0	1	0	0	0	1	0	0
3	0	0	0	0	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Berechne $|R_{p,n}|$ durch spaltenweise Addition

$$\begin{aligned}
 |R_{p,n}| &= \sum_{i=1}^n |\{k \geq 1 \mid (p^k \mid i)\}| \\
 &= \sum_{i=1}^n \max \{k \geq 1 \mid (p^k \mid i)\} = \sum_{i=1}^n \nu_p(i) = \nu_p(n!)
 \end{aligned}$$

Noch einmal zählen...

Wir betrachten $R_{p,n} := \{(i, k) \mid i \in \{1, 2, \dots, n\} \text{ und } p^k \mid i\}$

Beispiel für $p = 2, n = 14$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$k = 1$	0	1	0	1	0	1	0	1	0	1	0	1	0	1
2	0	0	0	1	0	0	0	1	0	0	0	1	0	0
3	0	0	0	0	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Wir haben $|R_{p,n}| = \nu_p(n!)$
durch **spaltenweise Addition**

Berechne $|R_{p,n}|$ durch **zeilenweise Addition**

$$|R_{p,n}| = \sum_{k \geq 1} |\{i \mid i \in \{1, 2, \dots, n\} \text{ und } p^k \mid i\}|$$

$$= \sum_{k \geq 1} \left\lfloor \frac{n}{p^k} \right\rfloor$$

also $\nu_p(n!) = \sum_{k \geq 1} \left\lfloor \frac{n}{p^k} \right\rfloor$



Noch eine Hilfsaussage

Lemma 12.10

$\forall p \text{ prim}, l = \nu_p \left(\binom{2n}{n} \right) : p^l \leq 2n$

Beweis.

$$\begin{aligned} l &= \nu_p \left(\binom{2n}{n} \right) = \nu_p \left(\frac{(2n)!}{(n!) \cdot (n!)} \right) = \nu_p(2n!) - 2 \cdot \nu_p(n!) \\ &= \sum_{k \geq 1} \left\lfloor \frac{2n}{p^k} \right\rfloor - 2 \cdot \sum_{k \geq 1} \left\lfloor \frac{n}{p^k} \right\rfloor = \sum_{k \geq 1} \left\lfloor \frac{2n}{p^k} \right\rfloor - 2 \cdot \left\lfloor \frac{n}{p^k} \right\rfloor \end{aligned}$$

Beobachtung

- Summanden mit $p^k > 2n$ alle 0
- $\forall y: \lfloor 2y \rfloor - 2 \lfloor y \rfloor \in \{0, 1\}$
- also alle Summanden $\in \{0, 1\}$
- also $l \leq \max \{k \geq 1 \mid p^k \leq 2n\}$
- also $p^l \leq 2n$



Anzahl von Primzahlen und Binomialkoeffizienten

Lemma 12.11

$$\forall n \in \mathbb{N}: \binom{2n}{n} \leq (2n)^{\pi(2n)}$$

Beweis.

Betrachte
$$\binom{2n}{n} = \underbrace{p_1^{k_1} \cdot p_2^{k_2} \cdots p_r^{k_r}}_{\text{eindeutige Primfaktorzerlegung}}$$

klar $\forall i \in \{1, 2, \dots, r\}: p_i \mid (2n)!$

also $\max\{p_i \mid 1 \leq i \leq r\} < 2n$ und $r \leq \pi(2n)$

Wir wissen $\forall i \in \{1, 2, \dots, r\}: p_i^{k_i} \leq 2n$
aus Lemma 12.10

also
$$\binom{2n}{n} = \prod_{i=1}^r p_i^{k_i} \leq (2n)^r \leq (2n)^{\pi(2n)}$$

□

Zur Anzahl von Primzahlen

Theorem 12.12

$$\forall n \geq 2: \pi(n) \geq \frac{n}{\log n} - 2$$

Beweis.

Erinnerung $\frac{2^{2n}}{2n} \leq \binom{2n}{n} \leq 2^{2n}$

also $(2n)^{\pi(2n)} \geq \frac{2^{2n}}{2n}$
mit Lemma 12.11

also $\pi(2n) \geq \log_{2n} \left(\frac{2^{2n}}{2n} \right) = \frac{2n}{\log(2n)} - 1$

also für gerade n Behauptung gezeigt
für ungerade n sehr ähnlich □

(Nur ganz wenig) Gruppentheorie

Definition 12.13

Menge G mit binärer Operation \circ heißt **Gruppe**, wenn gilt:

- $\forall a, b, c \in G: (a \circ b) \circ c = a \circ (b \circ c)$ (**Assoziativität**)
- $\exists e \in G: \forall a \in G: a \circ e = e \circ a = a$ (**neutrales Element**)
- $\forall a \in G: \exists b \in G: a \circ b = b \circ a = e$ (**inverse Elemente**)

Gilt zusätzlich $\forall a, b \in G: a \circ b = b \circ a$ (Kommutativität), so heißt (G, \circ, e) **abelsche Gruppe**.

Beobachtungen

- e ist eindeutig
- $\forall a \in G: \text{ inverses Element } a^{-1} \text{ eindeutig}$
- $\forall a, b, c \in G: a \circ c = b \circ c \Leftrightarrow a = b \Leftrightarrow c \circ a = c \circ b$

Beispiele

- $(\mathbb{Z}, +, 0)$ unendliche Gruppe
- $(\{1\}, \cdot, 1)$ endliche Gruppe

Untergruppen

Definition 12.14

Sei (G, \circ, e) Gruppe. $H \subseteq G$ heißt **Untergruppe**, wenn (H, \circ, e) Gruppe ist.

Zu einer Untergruppe H von G definieren wir Relation \sim_H durch
 $\forall a, b \in G: a \sim_H b \Leftrightarrow b^{-1} \circ a \in H$

Beobachtung \sim_H ist **Äquivalenzrelation**

- **reflexiv** $a \sim_H a$, weil $a^{-1} \circ a = e \in H$ ✓
- **symmetrisch** $a \sim_H b$ (also $b^{-1} \circ a \in H$) $\Rightarrow b \sim_H a$, weil
 $(b^{-1} \circ a) \circ (a^{-1} \circ b) = e$ und wegen Abschlusseigenschaft
 dann auch in H ✓
- **transitiv** $a \sim_H b$ und $b \sim_H c \Rightarrow a \sim_H c$, weil
 $(c^{-1} \circ b) \circ (b^{-1} \circ a) = c^{-1} \circ a \in H$ wegen
 Abschlusseigenschaft ✓

Über Untergruppen \sim_H

Wir haben für Gruppe G und Untergruppe H
 Äquivalenzrelation \sim_H mit $a \sim_H b \Leftrightarrow b^{-1} \circ a \in H$

Betrachte für $b \in G$ Abbildung $f_b: [b]_H \rightarrow G$
 mit $f(a) = b^{-1} \circ a$

klar $\forall a \in [b]_H: f(a) \in H$

Beobachtung $\forall a, a' \in [b]_H: a \neq a' \Leftrightarrow f(a) \neq f(a')$

also f_b Bijektion zwischen $[b]_H$ und H
 mit $|[b]_H| = |H|$

Anmerkung für endliche Gruppen G
 H abgeschlossen gegen \circ mit $e \in H$ ist Untergruppe

Gilt das auch für unendliche Gruppen?

Nein! Beispiel $(\mathbb{Z}, +, 0)$ und \mathbb{N}

Über Untergruppen

Theorem 12.15

G endliche Gruppe, H Untergruppe von G .

$$|H| \mid |G|$$

Beweis.

Betrachte C_1, C_2, \dots, C_r Äquivalenzklassen von \sim_H

klar G endlich, also H endlich, also r endlich

gerade gesehen $\forall i: |C_i| = |H|$

Erinnerung Äquivalenzklassen paarweise disjunkt

also $G = \bigcup_{i=1}^r C_i$ und $|G| = r \cdot |H|$

also $|H| \mid |G|$



Zyklische Gruppen

Definiere für Gruppe (G, \circ, e) , $a \in G$ und $i \in \mathbb{Z}$

$$a^i := \begin{cases} e & \text{falls } i = 0 \\ a \circ a^{i-1} & \text{falls } i > 0 \\ (a^{-1})^{(-i)} & \text{falls } i < 0 \end{cases}$$

Beobachtungen

- $\forall a \in G, i \in \mathbb{Z}: (a^i)^{-1} = a^{-i}$
- $\forall a \in G, i, j \in \mathbb{Z}: a^{i+j} = a^i \circ a^j$
- $\forall a, b \in G$ mit $a \circ b = b \circ a: (a \circ b)^i = a^i \circ b^i$

Definition 12.16

Sei (G, \circ, e) Gruppe, $a \in G$. Es ist $\langle a \rangle := \{a^i \mid i \in \mathbb{Z}\}$, $\text{ord}_G(a) := |\langle a \rangle|$ heißt **Ordnung** von a in G . a heißt **erzeugendes Element**, wenn $\langle a \rangle = G$ gilt. G heißt **zyklisch**, wenn G erzeugendes Element hat.

Wichtige Gruppen

Beobachtungen

- $(\mathbb{Z}, +, 0)$ ist unendliche Gruppe
- $(\mathbb{Z}_n, + \bmod n, 0)$ mit $\mathbb{Z}_n = \{0, 1, 2, \dots, n-1\}$ ist endliche Gruppe

Lemma 12.17

Sei (G, \circ, e) Gruppe, $a \in G$.

- 1 Wenn a^i verschieden sind für alle $i \in \mathbb{Z}$, dann ist $\text{ord}_G(a) = \infty$ und (G, \circ, e) ist isomorph zu $(\mathbb{Z}, +, 0)$.
- 2 Wenn $a^i = a^j$ für ein $i < j$, dann ist $\text{ord}_G(a) \leq j - i$.

Beweis.

- 1 **Beobachtung** $i \rightarrow a^i$ definiert Isomorphismus
weil $0 \rightarrow a^0 = e$, $i + j \rightarrow a^{i+j} = a^i \circ a^j$, $-i \rightarrow a^{-i} = (a^i)^{-1}$

Beweis von Theorem 12.17

Behauptung $a^i = a^j$ für ein $i < j \Rightarrow \text{ord}_G(a) \leq j - i$

klar $a^{j-i} = a^j \circ (a^i)^{-1} = a^j \circ (a^j)^{-1} = a^j \circ a^{-j} = e$

zu zeigen $|\langle a \rangle| \leq j - i$

Betrachte a^l für beliebiges $l \in \mathbb{Z}$

Definiere $r = l \bmod (j - i)$, $q = \lfloor l / (j - i) \rfloor$

also $l = q \cdot (j - i) + r$ mit $0 \leq r < j - i$.

also $a^l = a^{q \cdot (j-i) + r} = a^{q \cdot (j-i)} \circ a^r = a^r$

also $\langle a \rangle \subseteq \{a^0, a^1, \dots, a^{j-i-1}\}$

□

Über endliche Gruppen

Theorem 12.18

Sei (G, \circ, e) Gruppe, $a \in G$ mit $\text{ord}_G(a) = n$ für ein $m \in \mathbb{N}$.

- 1 $\forall i \leq j: a^i = a^j \Leftrightarrow n \mid (j - i)$
- 2 $\langle a \rangle$ ist isomorph zu $(\mathbb{Z}_n, + \text{ mod } n, 0)$.

Beweis.

- 1 Wir haben $\langle a \rangle = \{a^0, a^1, \dots, a^{n-1}\}$, weil $\text{ord}_G(a) = n$
also $\exists i \in \{0, 1, \dots, n-1\}: a^i = a^n$

Behauptung $a^n = a^0 = e$

Widerspruchsbeweis Annahme $a^n = a^i$ mit

$i \in \{1, 2, \dots, n-1\}$

dann $\text{ord}_G(a) \leq n - i < n$ (Lemma 12.17) Widerspruch

also $a^n = a^0 = e$ ✓

Beweis von Theorem 12.18

zu zeigen $\forall i \leq j: a^i = a^j \Leftrightarrow n \mid (j - i)$

Beweis „ \Leftarrow “ Gelte $n \mid (j - i)$

also $\exists q \in \mathbb{Z}: q \cdot n = j - i$

Erinnerung $a^n = e$

also $a^i = a^i \circ (a^n)^q = a^{q \cdot n + i} = a^j$ ✓

Beweis „ \Rightarrow “ Gelte $a^i = a^j$

also $a^{j-i} = a^j \circ a^{-i} = a^j \circ a^{-j} = e = a^0$

Definiere $r = (j - i) \bmod n, q = \lfloor (j - i)/n \rfloor$

also $j - i = q \cdot n + r$ mit $0 \leq r < n$

also $a^{j-i} = a^{q \cdot n + r} = a^r$

Weil $a^{j-i} = e$, also $a^r = e$, also $r = 0$

also $n \mid (j - i)$ ✓

② Beobachtung $i \rightarrow a^i$ definiert Isomorphismus

