Zahlentheorie

Diskrete Strukturen

Uta Priss
ZeLL, Ostfalia

Sommersemester 2016

Agenda

Hausaufgaben

Primzahlen

Teiler und Modulo

Hashfunktion

SetlX

Was ist hier falsch:

```
folge := [[(-1)**m*2**n : n in [1..m]], m in [1..10]];
```

Warum implementiert man Folgen als Listen und nicht als Mengen?

SetIX

Was ist hier falsch:

```
folge := [[(-1)**m*2**n : n in [1..m]], m in [1..10]];
```

Warum implementiert man Folgen als Listen und nicht als Mengen?

$\{1, 1, 2, 2, 3, 3\}$???
$[1, 1, 2, 2, 3, 3]$

„Eine ganze Zahl a heißt durch eine natürliche Zahl b teilbar ...“
Kann ein Teiler negativ sein?

-17 % 5 = 3 oder 17 % 5 = -2.

Satz 2.49 $a = qm + r$ ... Diese sind eindeutig bestimmt, indem man festlegt, dass $0 \leq r < m$ sein soll .

Wie wird modulo von negativen Zahlen von Programmiersprachen behandelt?

Ihre Fragen

Die Zahl der Primzahlen ist aus dem Grund unendlich, da man nicht beweisen kann, welches die letzte Primzahl ist ?

Warum kann man Hashwerte nicht zurückrechnen?

Können wir mehr über Primzahlen, Kryptographie und Quantencomputer erfahren? (nicht klausurrelevant)

Primzahlen, Kryptographie und Quantencomputer (RSA-Algorithmus: Seite 99 im Buch)

Es gibt keinen schnellen Algorithmus, um für eine große Zahl herauszufinden, ob die Zahl als Produkt zweier Primzahlen dargestellt werden kann ($n = pq$).

Man kodiert eine geheime Botschaft als Zahl und rechnet diese mit einer Formel modulo n in eine andere Zahl um.

Modulorechnung kann man nicht einfach umkehren.

Nur jemand, der ($n = pq$) kennt, kann die ursprüngliche Botschaft entschlüsseln.

Mit Quantencomputern könnte man schneller rechnen und $n = pq$ durch ausprobieren lösen.

Schreiben Sie diese beiden Sätze als Formeln

„Primzahlen sind nur durch sich selbst und eins teilbar.“

„Jede natürliche Zahl größer als 1 ist entweder selbst eine Primzahl, oder sie lässt sich als Produkt von Primzahlen schreiben.“

„Primzahlen sind nur durch sich selbst und eins teilbar.“

1. Versuch: $p \in \mathbb{P} \Longrightarrow p \bmod p = 0 \wedge p \bmod 1 = 0$
Problem: das gilt für alle Zahlen. Das Wort „nur“ fehlt.

Lösungen:
$p \in \mathbb{P} \Longleftrightarrow \forall_{n \in \mathbb{N}, 1<n<p} : p \bmod n \neq 0$

$p \in \mathbb{P} \Longleftrightarrow \forall_{\{n \in \mathbb{N} | 1<n<p\}} : p \bmod n \neq 0$

$p \in \mathbb{P} \Longleftrightarrow \forall_{n \in \mathbb{N}} : (1 < n < p \Longrightarrow p \bmod n \neq 0)$

$p \in \mathbb{P} \Longleftrightarrow \forall_{n \in \mathbb{N}} : (p \bmod n = 0 \Longrightarrow n = 1 \vee n = p)$

„Jede natürliche Zahl größer als 1 ist entweder selbst eine Primzahl, oder sie lässt sich als Produkt von Primzahlen schreiben.“

1. Versuch: $n \in N \wedge n > 1 \Longrightarrow$
$(n \in \mathbb{P} \vee \exists_{k>1} : p_1, p_2, ..., p_k \in \mathbb{P} \wedge n = p_1 \cdot p_2 \cdot ... \cdot p_k)$
Problem: das „entweder ... oder“ fehlt.

Lösung:
$n \in N \wedge n > 1 \Longrightarrow$
$(n \notin \mathbb{P} \Longleftrightarrow \exists_{k>1} : p_1, p_2, ..., p_k \in \mathbb{P} \wedge n = p_1 \cdot p_2 \cdot ... \cdot p_k)$

entweder A oder B
$\Longleftrightarrow (A \vee B) \wedge (\neg(A \wedge B)) \Longleftrightarrow (A \vee B) \wedge (\neg A \vee \neg B)$
$\Longleftrightarrow ((\neg A \Longrightarrow B) \wedge (B \Longrightarrow \neg A)) \Longleftrightarrow (\neg A \Longleftrightarrow B)$

Ein Algorithmus um Primzahlen zu finden

	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

Ein Algorithmus um Primzahlen zu finden

	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~~

Ein Algorithmus um Primzahlen zu finden

	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~~

Ein Algorithmus um Primzahlen zu finden

	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~~

Ein Algorithmus um Primzahlen zu finden

	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~~

Dieser Algorithmus heißt „Sieb des Eratosthenes“.

Schreiben Sie den Algorithmus als Menge mit Set Comprehension Notation.

Wie viele Primzahlen gibt es?

Wie viele Primzahlen gibt es?

$(1 * 2 * 3) + 1 = 7$

$(1 * 2 * 3 * 5) + 1 = 31$

$(1 * 2 * 3 * 5 * 7) + 1 = 211$

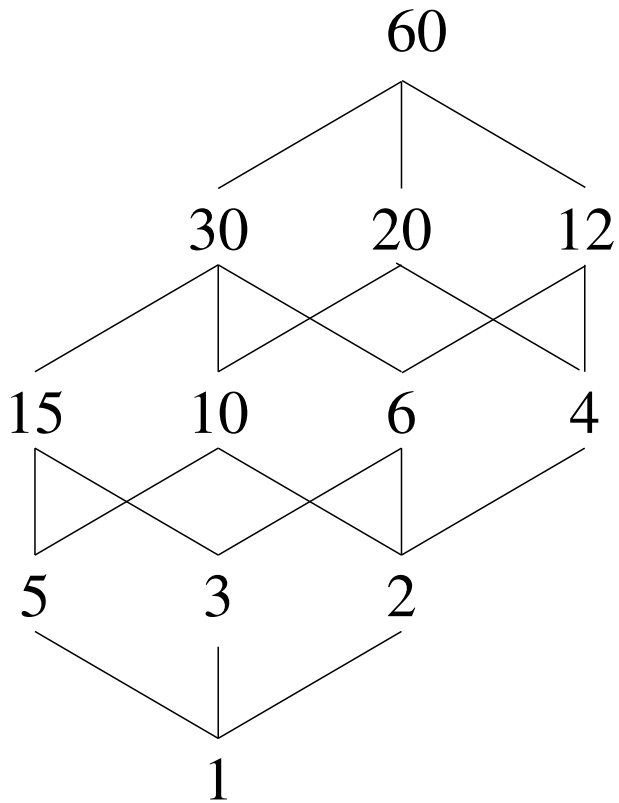
$(1 * 2 * 3 * 5 * 7 * 11) + 1 = 2311$

Nicht alle Zahlen in dieser Liste sind Primzahlen:
$(1 * 2 * 3 * 5 * 7 * 11 * 13) + 1 = 30031 = 59 * 509$

Warum kann es keine größte Primzahl geben?

Was ist dies für ein Beweisverfahren?

Primfaktorzerlegung



Lesen Sie die Primfaktorzerlegung von 60 aus dem Bild ab.

Zeichnen Sie ein Bild der Primfaktorzerlegung von 81 und von 36.

Was tut diese Funktion? Was ist die Definitionsmenge und die Wertemenge:

```
fragezeichen := procedure(n) {
      s := {2..n };
      return s - {p*q : [p, q] in s >< s };
};
```

Wie kann man den ggT berechnen?

Euklidischer Algorithmus für den ggT von $a, b \in \mathbb{N}$

Setzt man $r_0 = a, r_1 = b$ und r_k rekursiv als Rest der Definition von r_{k-2} durch r_{k-1}, so bricht diese Rekursion ab und es gilt $r_k = ggT(a, b)$.

Beispiel: ggT(26,14)
26 % 14 = 12
14 % 12 = 2
12 % 2 = 0
ggT(14,12) = 2

Warum liefert dieser Algorithmus den ggT als Ergebnis?
Probieren Sie es aus mit ein paar Beispielen.

Euklidischer Algorithmus für den ggT von $a, b \in \mathbb{N}$

Setzt man $r_0 = a, r_1 = b$ und r_k rekursiv als Rest der Definition von r_{k-2} durch r_{k-1}, so bricht diese Rekursion ab und es gilt $r_k = ggT(a, b)$.

Beispiel: ggT(26,14)
26 % 14 = 12
14 % 12 = 2
12 % 2 = 0
ggT(14,12) = 2

Warum liefert dieser Algorithmus den ggT als Ergebnis?
Probieren Sie es aus mit ein paar Beispielen.

q teilt 137 $\Longrightarrow (137 + 2) \bmod q = 0 + 2 \bmod q = 2$
Warum folgt daraus, dass 137 und 139 teilerfremd sind?

Zwei Zahlen a und b sind kongruent modulo m genau dann, wenn $a - b = km$ mit $k \in \mathbb{Z}$.

Schreiben Sie „Kongruenz modulo m“ als Relation.
Zeigen Sie, dass es sich um eine Äquivalenzrelation handelt.
Wie sehen die Äquivalenzklassen aus?

Schreiben Sie „Kongruenz modulo m“ als Relation.

Für fest gewähltes $m \in \mathbb{N}$

$$krg_m \subseteq \mathbb{Z} \times \mathbb{Z}$$
$$(a, b) \in krg_m :\Longleftrightarrow \exists_{k \in \mathbb{Z}} : a - b = km.$$

Beispiel Transitivität:
$(a, b) \in krg_m \wedge (b, c) \in krg_m \Longrightarrow a - b = k_1 m, b - c = k_2 m \Longrightarrow$
$a - (k_2 m + c) = k_1 m \Longrightarrow a - c = k_1 m + k_2 m = (k_1 + k_2)m \Longrightarrow$
$(a, c) \in krg_m$

Zeigen Sie, dass zwei Äquivalenzklassen genau dann disjunkt sind, wenn sie nicht gleich sind.

Zur Erinnerung: $[a] = \{x \in A \mid (a, x) \in R\}$

Tipp: Starten Sie mit der Negation: Zwei Äquivalenzklassen sind genau dann gleich, wenn sie ...

Zwei Äquivalenzklassen genau dann gleich, wenn sie nicht disjunkt sind. (Für leere Mengen gilt dies nicht, da die leere Menge zu sich selbst disjunkt ist.)

Zu zeigen: A $\Longrightarrow$ B und B $\Longrightarrow$ A
„... gleich ... “ $\Longrightarrow$ „... nicht disjunkt...“
Wenn 2 nicht leere Mengen gleich sind, dann haben sie alle Elemente gemeinsam, also mindestens 1 gemeinsames Element, also sind sie nicht disjunkt.

„... nicht disjunkt...“ $\Longrightarrow$ „... gleich ... “
$\exists_y : y \in \{x \in A \mid (a,x) \in R\}$ und $y \in \{x \in A \mid (b,x) \in R\}$
$\Longrightarrow (a,y) \in R \wedge (b,y) \in R \Longrightarrow (a,y) \in R \wedge (y,b) \in R \Longrightarrow$
$(a,b) \in R$
Da R eine Äquivalenzrelation ist, muss $[a] = [b]$ sein.

Hashtabelle

Speicherplatz	Schlüssel	Wert
	Braunschweig	0531
	Hannover	0511
	Wolfenbüttel	05331

Hashfunktion H: Wortlänge Modulo 5

$H(\text{Braunschweig}) = H(\text{Wolfenbüttel}) = 12 \bmod 5 = 2$
$H(\text{Hannover}) = 8 \bmod 5 = 3$

Hashtabelle

Speicherplatz	Schlüssel	Wert
2	Braunschweig	0531
3	Hannover	0511
	Wolfenbüttel	05331

Hashfunktion H: Wortlänge Modulo 5

$H(\text{Braunschweig}) = H(\text{Wolfenbüttel}) = 12 \bmod 5 = 2$
$H(\text{Hannover}) = 8 \bmod 5 = 3$

Hashtabelle

Speicherplatz	Schlüssel	Wert
2	Braunschweig	0531
3	Hannover	0511
4	Wolfenbüttel	05331

Wie sucht man nach Braunschweig und Wolfenbüttel?
Warum kann man Hashwerte nicht zurückrechnen?