Kapitel 2: Algebraische und zahlen-theoretische Algorithmen (Teil II)

(Effiziente Algorithmen, WS 2018)

Gerhard Woeginger

WS 2018, RWTH

Algebraische und zahlen-theoretische Algorithmen

- Multiplikation von ganzen Zahlen
- Multiplikation von Matrizen
- Multiplikation von Polynomen

- Der grösste gemeinsame Teiler
- Modulare Potenzfunktion
- Primzahltest

Zum Aufwärmen:
Der grösste gemeinsame Teiler

Der Euklidische Algorithmus

Euklidischer Algorithmus:

```
1  function Euklid(a,b): integer
2      if b==0 then return a
3              else return Euklid(b, a mod b)
```

$$\begin{aligned}
\text{Euklid}(33, 39) &= \text{Euklid}(39,\ 33) \\
&= \text{Euklid}(33,\ 6) \\
&= \text{Euklid}(6,\ 3) \\
&= \text{Euklid}(3, 0) \\
&= 3
\end{aligned}$$

Analyse des Euklidischen Algorithmus (1)

Definition: Fibonacci Zahlen

Die Fibonacci Zahlen sind definiert durch $F_0 = 0$ und $F_1 = 1$,
und durch die Gleichung $F_n = F_{n-1} + F_{n-2}$ für $n \geq 2$.

n	0	1	2	3	4	5	6	7	8	9	10	11	12	...
F_n	0	1	1	2	3	5	8	13	21	34	55	89	144	...

$$F_n \approx \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^n \qquad \text{mit Basiszahl } \frac{1+\sqrt{5}}{2} \approx 1.618$$

Analyse des Euklidischen Algorithmus (2)

Hilfssatz

Es seien $a > b \geq 1$ zwei ganze Zahlen,
für die der Aufruf von Euklid(a, b) zu $k \geq 1$ rekursiven Aufrufen führt.
Dann gilt $a \geq F_{k+2}$ und $b \geq F_{k+1}$.

- Beweis mit vollständiger Induktion über $k \geq 1$
- Der Fall $k = 1$: Aus $b \geq 1$ folgt $b \geq F_2 = 1$.
  Aus $a > b \geq 1$ folgt $a \geq F_3 = 2$.
- Induktionsschritt: Wenn Euklid(a, b) zu k rekursiven Aufrufen führt,
  dann führt Euklid$(b, a \bmod b)$ zu mindestens $k - 1$ Aufrufen.
  Die Induktionsannahme liefert dann $b \geq F_{k+1}$ (!) und $a \bmod b \geq F_k$.
- Aus $a > b$ folgt $\lfloor a/b \rfloor \geq 1$.
  Daraus erhalten wir $b + (a \bmod b) \;=\; b + (a - b\lfloor a/b \rfloor) \;\leq\; a$.
- Ergo $a \;\geq\; b + (a \bmod b) \;\geq\; F_{k+1} + F_k = F_{k+2}$

Analyse des Euklidischen Algorithmus (3)

Satz von Lamé

Es seien $a > b \geq 1$ zwei ganze Zahlen mit $b < F_k$.
Dann macht Euklid(a, b) weniger als $k \geq 1$ rekursive Aufrufe.

Die Schranke "$b < F_k$" im Satz von Lamé ist bestmöglich:

- Euklid(F_{k+1}, F_k) macht nämlich genau $k - 1$ rekursive Aufrufe.
- Der erste rekursive Aufruf ist Euklid$(F_k, F_{k+1} \bmod F_k)$
- Der erste rekursive Aufruf ist Euklid(F_k, F_{k-1})
- Der Rest folgt mit Induktion

Zusammenfassung

Der Satz von Lamé und $F_n \approx 1.618^n$ implizieren:

Satz

Für zwei Eingabezahlen $a > b \geq 1$ mit n Bits
terminiert der Euklidische Algorithmus nach $O(n)$ rekursiven Aufrufen.

- Bei jedem Aufruf wird eine Modulo-Operation durchgeführt.
- Modulo-Operationen sind gleich teuer wie Divisionen und Multiplikationen von n-stelligen Zahlen

Modulare Potenzfunktion

Modulare Potenzfunktion

Problem: Modulares Potenzieren

Eingabe: Drei positive ganze Zahlen a, n, m

Gesucht: Der Wert $a^n \bmod m$

Achtung:

- Wenn man a^n berechnet, indem man einfach n-mal mit der Zahl a multipliziert, dann kostet das n Multiplikationen
- Wenn man a^n berechnet, indem man einfach n-mal mit der Zahl a multipliziert, dann arbeitet man mit Zahlen mit $\Theta(n \log a)$ Stellen

Ein Rechenbeispiel

Wir betrachten den Exponenten $n = 789$

- $n = 789 = 512 + 256 + 16 + 4 + 1 = 2^9 + 2^8 + 2^4 + 2^2 + 2^0$
- Wir berechnen der Reihe nach die Hilfswerte
  $H_0 = a \mod m$,
  $H_1 = a^2 \mod m$,
  $H_2 = a^4 \mod m$,
  $H_3 = a^8 \mod m$,
  $H_4 = a^{16} \mod m$,
  und allgemein $H_k = a^{2^k} \mod m$
- Dabei rechnen wir: $H_{k+1} := H_k^2 \mod m$
- Zum Schluss multiplizieren wir $H_9 \cdot H_8 \cdot H_4 \cdot H_2 \cdot H_0$
  (und dabei wird nach jeder Multiplikation das Ergebnis wieder modulo m genommen)

Modulare Potenzfunktion: Algorithmus

Modular-Exponentiation(a, n, m)

```
1  result= 1;
2  let n[k], n[k-1], ..., n[0] be
3      the binary representation of n;
4  for i= k downto 0 do
5      result= (result*result) mod m;
6      if n[i]==1 then result= (result*a) mod m;
7  endfor
8
9  return result
```

Für Eingabezahlen a, n, m mit k Bits
kann die Modulare Potenzfunktion in $O(k^3)$ Zeit berechnet werden.

Noch ein Rechenbeispiel (1)

Das folgende Schema für den Exponenten $n = 15$ verwendet sechs Multiplikationen:

$$\begin{aligned} a^2 &= a \cdot a & 2 &= 1 + 1 \\ a^4 &= a^2 \cdot a^2 & 4 &= 2 + 2 \\ a^8 &= a^4 \cdot a^4 & 8 &= 4 + 4 \\ \\ a^3 &= a^2 \cdot a & 3 &= 2 + 1 \\ a^7 &= a^4 \cdot a^3 & 7 &= 4 + 3 \\ a^{15} &= a^8 \cdot a^7 & 15 &= 8 + 7 \end{aligned}$$

Noch ein Rechenbeispiel (2)

Das folgende Schema für den Exponenten $n = 15$ verwendet nur fünf Multiplikationen:

$$\begin{aligned} a^2 &= a \cdot a & 2 &= 1 + 1 \\ a^3 &= a^2 \cdot a & 3 &= 2 + 1 \\ a^6 &= a^3 \cdot a^3 & 6 &= 3 + 3 \\ a^{12} &= a^6 \cdot a^6 & 12 &= 6 + 6 \\ a^{15} &= a^{12} \cdot a^3 & 15 &= 12 + 3 \end{aligned}$$

Additionsketten (1)

Definition: Additionskette

Eine Additionskette für eine ganze Zahl $n \geq 2$ ist
eine Zahlenfolge $\langle x_0, x_1, \ldots, x_k \rangle$ mit folgenden Eigenschaften:

- Es gilt $x_0 = 1$
- Für jeden Index $i \geq 2$ kann die Zahl x_i als Summe $x_i = x_j + x_k$ mit $0 \leq j, k < i$ geschrieben werden
- Es gilt $x_k = n$

Die Zahl k ist die Länge der Additionskette.

Mit $\ell(n)$ bezeichnen wir die Länge der kürzesten Additionskette für n.

Beispiel:
Wir haben gesehen, dass $\ell(15) \leq 5$ gilt

Additionsketten (2)

- Trivial: Für alle $n \geq 2$ gilt $\ell(2n) \leq \ell(n) + 1$.
- Fakt: Es existieren Zahlen n mit $\ell(2n) \leq \ell(n)$.
  Zum Beispiel gilt $\ell(382) = \ell(191) = 11$
- Fakt: Es existieren Zahlen n mit $\ell(2n) \leq \ell(n) - 1$.
  (Neill Michael Clift hat 2011 ausgerechnet,
  dass $n = 375494703$ die kleinste derartige Zahl ist.)
- Niemand weiss, ob es eine Zahl n mit $\ell(2n) \leq \ell(n) - 2$ gibt

- Scholz-Brauer Vermutung (1937):
  Für alle $n \geq 2$ gilt $\ell(2^n - 1) = \ell(n) + n - 1$
- Komplexität der Berechnung von $\ell(n)$: unbekannt

Primzahl-Test

Primzahlen

Wolfram-Alpha behauptet:

```
10^200 + 357 is a prime number
```

Wolfram-Alpha behauptet auch:

```
10^200 + 349 is not a prime number

Partial factorization:
2399 × 7079 × 91033 × 360628131971 × 17936569401172469686
07844523635531601680271722217839963531653895049263302135
21518177267973390218599635401580277396813257728949954049
24965089532082651688920261909909036099958383
(4 prime factors, 1 composite factor)
```

Primzahlen

Carl Friedrich Gauss, *"Disquisitiones Arithmeticae"*, (1801):

"Problema, numeros primos a compositis dignoscendi, hosque in factores suos primos resolvendi, ad gravissima ac utilissima totius arithmeticae pertinere, et geometrarum tum veterum tum recentiorum industriam ac sagacitatem occupavisse, tam notum est, ut de hac re copiose loqui superfluum foret."

"Dass das Problem, die Primzahlen von den zusammengesetzten zu unterscheiden und letztere in ihre Primfaktoren zu zerlegen, zu den wichtigsten und nützlichsten der ganzen Arithmetik gehört und den Fleiss und die Weisheit der Geometer der Antike und der Neuzeit beschäftigt hat, ist so bekannt, dass es überflüssig ist, viel darüber zu sagen."

Primzahlen

Problem: Primzahl

Eingabe: Eine positive ganze Zahl n in Dezimaldarstellung

Frage: Ist n eine Primzahl?

- Cryptosysteme (wie zum Beispiel RSA) benötigen sehr grosse Primzahlen
- Die Sicherheit von solchen Systemen beruht auf der (bis heute unbewiesenen) Annahme, dass das Faktorisieren von natürlichen Zahlen ein algorithmisch schwieriges Problem ist
- Faktorisieren $\neq$ Primzahl-Test

Primzahl-Test (1)

Ein korrekter, deterministischer Algorithmus:

```
1   for t= 1 to sqrt(n) do
2       if (n mod t == 0) then return "COMPOSITE"
3
4   return "PRIME"
```

Aber: Laufzeit nicht polynomiell beschränkt in Länge der Eingabe

Primzahl-Test (2)

Ein randomisierter Algorithmus:

```
1   Pick random number t from {2,...,sqrt(n)}
2   if (n mod t == 0) then return "COMPOSITE"
3                     else return "PRIME"
```

- Wenn Test COMPOSITE sagt, dann ist n wirklich nicht-prim
- Wenn Test PRIME sagt, dann ist n nicht notwendigerweise prim

Übung

Betrachte den Spezialfall $n = q^2$, wobei q prim ist.

- Frage: Wie gross ist die Fehlerwahrscheinlichkeit?
- Frage: Wie oft muss man den Test wiederholen, damit die Fehlerwahrscheinlichkeit unter $1/2$ sinkt?

Zahlen-theoretisches Werkzeug #1

Definition

Mit $\mathbb{Z}_n^*$ bezeichnen wir die multiplikative Gruppe modulo n. Elemente von $\mathbb{Z}_n^*$ sind die Restklassen modulo n, die relativ prim zu n sind:

$$\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid \mathrm{ggT}(a, n) = 1\}$$

	1	2	4	7	8	11	13	14
1	1	2	4	7	8	11	13	14
2	2	4	8	14	1	7	11	13
4	4	8	1	13	2	14	7	11
7	7	14	13	4	11	2	1	8
8	8	1	2	11	4	13	14	7
11	11	7	14	2	13	1	8	4
13	13	11	7	1	14	8	4	2
14	14	13	11	8	7	4	2	1

Zahlen-theoretisches Werkzeug #2

Satz (Kleiner Satz von Fermat)

Für eine Primzahl p und $a \in \mathbb{Z}_p^*$ gilt:

$$a^{p-1} \equiv 1 \pmod{p}$$

Beispiel (mit Primzahl $p = 5$):

$$1^4 \equiv 1 \equiv 1 \pmod{5}$$

$$2^4 \equiv 16 \equiv 1 \pmod{5}$$

$$3^4 \equiv 81 \equiv 1 \pmod{5}$$

$$4^4 \equiv 256 \equiv 1 \pmod{5}$$

Beispiel (mit Nicht-Primzahl $p = 8$):

$$3^7 \equiv 3 \not\equiv 1 \pmod{8}$$

Fermat Test

Fermat Test (1): zur Basis 2

Kleiner Fermat $\Longrightarrow$ Jedes prime $n \neq 2$ erfüllt $2^{n-1} \equiv 1 \pmod{n}$

Fermat Test zur Basis 2:

```
1   If modular-exp(2,n-1,n) <> 1
2           then return "COMPOSITE"
3           else return "PRIME"
```

Dieser Test ist halb-seitig fehlerhaft:

- Die Zahl $n = 341 = 11 \cdot 31$ ist nicht-prim
- Dennoch gilt $2^{340} \equiv 1 \pmod{341}$

Fermat Test (2): zur Basis 3

Kleiner Fermat $\Longrightarrow$ Jedes prime $n \neq 3$ erfüllt $3^{n-1} \equiv 1 \pmod{n}$

Fermat Test zur Basis 3:

```
1   If modular-exp(3,n-1,n) <> 1
2           then return "COMPOSITE"
3           else return "PRIME"
```

Dieser Test ist halb-seitig fehlerhaft:

- Die Zahl $n = 91 = 7 \cdot 13$ ist nicht-prim
- Dennoch gilt $3^{90} \equiv 1 \pmod{91}$

Fermat Test (3): randomisiert

Fermat Test:

```
1   Randomly pick integer a in {2,...,n-2}
2   If ggT(a,n)>1
3          then return "COMPOSITE"
4      Else if modular-exp(a,n-1,n) <> 1
5          then return "COMPOSITE"
6      Else return "PROBABLY PRIME"
```

- Wenn Test COMPOSITE sagt, dann ist n wirklich nicht-prim
- Wenn Test PROBABLY PRIME sagt, dann ist n nicht notwendigerweise prim

Pseudo-Primzahlen (1)

Definition (Pseudo-Primzahl)

Es sei $n \geq 3$ eine ungerade, zusammengesetzte Zahl, und es sei $a \in \mathbb{Z}_n^*$. Falls $a^{n-1} \equiv 1 \pmod{n}$ gilt,

- dann ist n eine Pseudo-Primzahl zur Basis a;
- dann ist a eine betrügerische Basis für n.

Beispiel:

$n = 341$ ist Pseudo-Primzahl zur Basis $a = 2$

$a = 2$ ist eine betrügerische Basis für $n = 341$

Beispiel:

$n = 91$ ist Pseudo-Primzahl zur Basis $a = 3$

$a = 3$ ist eine betrügerische Basis für $n = 91$

Pseudo-Primzahlen (2)

Satz

Jede ungerade, zusammengesetzte Zahl $n \geq 3$ erfüllt genau eine der folgenden beiden Aussagen:

- Alle Basen $a \in \mathbb{Z}_n^*$ sind Betrüger für n.
- Höchstens die Hälfte aller Basen $a \in \mathbb{Z}_n^*$ sind Betrüger für n.

Beweis:

- Aus $a^{n-1} \equiv 1 \pmod{n}$ und $b^{n-1} \equiv 1 \pmod{n}$ folgt immer auch $(ab)^{n-1} \equiv 1 \pmod{n}$. Ergo: $ab \in \mathbb{Z}_n^*$ ist ebenfalls betrügerisch.
- Das inverse Element a^{-1} von a in $\mathbb{Z}_n^*$ erfüllt $(a^{-1})^{n-1} \equiv 1 \pmod{n}$. Ergo: Mit a ist immer auch a^{-1} betrügerisch.
- Das neutrale Element 1 ist betrügerisch.
- Ergo: Die betrügerischen Basen bilden Untergruppe von $\mathbb{Z}_n^*$. Kardinalität einer Untergruppe teilt Kardinalität der Gruppe $\mathbb{Z}_n^*$.

Carmichael Zahlen (1)

Definition (Carmichael Zahl)

Eine ungerade, zusammengesetzte Zahl $n \geq 3$ ist eine Carmichael Zahl, falls alle Basen $a \in \mathbb{Z}_n^*$ Betrüger für n sind.

- Carmichael Zahlen sind tödlich für den Fermat Test:
  Für Carmichael Zahlen liefert der Test immer die falsche Antwort.
- Die ersten Carmichael Zahlen wurden 1912 vom amerikanischen Mathematiker Robert Daniel Carmichael entdeckt (*"On composite numbers P which satisfy the Fermat congruence"*)
- Kleinste Carmichael Zahl: $n = 561 = 3 \cdot 11 \cdot 17$ (Nachrechnen!)
- William Robert Alford, Andrew Granville und Carl Pomerance haben 1994 bewiesen, dass es unendlich viele Carmichael Zahlen gibt

Carmichael Zahlen (2)

Satz (Alwin Reinhold Korselt, 1899)

Eine ungerade, zusammengesetzte Zahl $n \geq 3$ ist eine Carmichael Zahl, genau dann wenn die Primfaktorisierung $n = p_1\, p_2 \cdots p_k$ die folgenden Eigenschaften besitzt:

- $p_i \neq p_j$ für $i \neq j$ (keine wiederholten Primteiler)
- $p_i - 1 \mid n - 1$ für $1 \leq i \leq k$

Fermat Test: Zusammenfassung

Der Fermat Test zeigt folgendes Verhalten für ungerade Eingabe-Zahlen $n \geq 3$:

- Falls n Primzahl, ist die Antwort immer PROBABLY PRIME
- Falls n Carmichael Zahl, ist die Antwort immer PROBABLY PRIME
- Falls n nicht-prim und Nicht-Carmichael, ist die Antwort COMPOSITE mit Wahrscheinlichkeit mindestens $1/2$

Miller-Rabin Test

Bessere Behandlung von betrügerischen Basen

Zentrale Idee:

- Wir betrachten den Exponenten $n-1$ im Fermat Test genauer.
- Setze $n-1 = 2^t u$, mit ungeradem $u \geq 1$ und $t \geq 1$

Neuer Test für die Basis a:

```
 1   y[0]:= modular-exp(a,u,n)
 2
 3   for i:=1 to t do
 4       y[i]:= y[i-1]^2 (mod n);
 5       if y[i]==1 and y[i-1]<>1 and y[i-1]<>-1
 6               then return "COMPOSITE"
 7   endfor
 8
 9   if y[t]<>1 then return "COMPOSITE"
10
11   return "PROBABLY PRIME"
```

Rechenbeispiel

Wir betrachten die Carmichael Zahl $n = 561 = 3 \cdot 11 \cdot 17$
mit $n - 1 = 560 = 2^4 \cdot 35$, und $t = 4$, und $u = 35$.

Dann gilt für die Basis $a = 7$:

- $y_0 := \text{modular-exp}(7, 35, 561) = 241$
- $y_1 := 241^2 \equiv 298 \pmod{561}$
- $y_2 := 298^2 \equiv 166 \pmod{561}$
- $y_3 := 166^2 \equiv 67 \pmod{561}$
- $y_4 := 67^2 \equiv 1 \pmod{561}$

Wegen $y_4 = 1$ und $y_3 \neq 1$ und $y_3 \neq -1$
wird Pseudo-Primzahl $n = 561$ nun korrekt als nicht-prim erkannt

Zahlen-theoretisches Werkzeug #3

Satz (Quadratwurzelsatz)

Für eine Primzahl n besitzt die Gleichung $x^2 \equiv 1$ nur zwei Lösungen über $\mathbb{Z}_n^*$: Die Lösung $x = 1$ und die Lösung $x = -1$.

Beweis:

- Schreibe die Gleichung $x^2 \equiv 1$ als $(x-1)(x+1) \equiv 0 \pmod{n}$
- Wenn n prim ist, muss einer der beiden Faktoren durch n teilbar sein

Anmerkung:
Für die Nicht-Primzahl $n = 8$ gilt $1^2 \equiv 3^2 \equiv 5^2 \equiv 7^2 \equiv 1 \pmod{8}$

Was antwortet der neue Test?

Die Zahlenfolge $\langle y_0, y_1, \ldots, y_t \rangle$ für eine ungerade Zahl $n \geq 3$ fällt in eine der folgenden vier Kategorien:

(1) Die Folge $\langle *, *, *, *, \ldots, \gamma \neq 1 \rangle$ endet nicht mit 1: Dann ist n keine Primzahl (laut Fermat Test), und Miller-Rabin erkennt das korrekt.

(2) Die Folge $\langle 1, 1, 1, 1, \ldots, 1 \rangle$ besteht nur aus 1en: Keine Indizien. Miller-Rabin antwortet PROBABLY PRIME.

(3) Die Folge $\langle *, *, \ldots, *, -1, 1, 1, \ldots, 1 \rangle$ endet mit 1, und das letzte Folgenelement $\neq 1$ ist -1: Keine Indizien. Miller-Rabin antwortet PROBABLY PRIME.

(4) Die Folge $\langle *, *, \ldots, *, \gamma \neq \pm 1, 1, 1, \ldots, 1 \rangle$ endet mit 1, und das letzte Elemente $\neq 1$ ist ungleich ± 1: Dann ist n keine Primzahl (laut Quadratwurzelsatz), und Miller-Rabin erkennt das korrekt.

Analyse von Miller-Rabin

Das Hauptresultat zum Miller-Rabin Test lautet:

Satz

Für eine ungerade Nicht-Primzahl n liefert der Miller-Rabin Test mit Wahrscheinlichkeit mindestens $1/2$ die korrekte Antwort COMPOSITE.

Vorbereitende Schritte im Beweis:

- O.B.d.A. ist $n = p_1\, p_2 \cdots p_k$ eine Carmichael Zahl: Zusammengesetzte Nicht-Carmichael Zahlen werden ja bereits von dem in den Miller-Rabin Test integrierten Fermat Test mit Wahrscheinlichkeit mindestens $1/2$ identifiziert.
- Wir zerlegen $n = n_1\, n_2$ in zwei relativ prime Faktoren $n_1, n_2 \geq 2$

Beweis des Hauptresultats (1)

Erster Beweisschritt: Definition von k, b, und B

- Ein Paar (b, k) mit $b \in \mathbb{Z}_n^*$ und $k \in \{0, \ldots, t\}$ heisst gut, falls $b^{2^k u} \equiv -1 \pmod{n}$ gilt.
- Das Paar $(n-1, 0)$ ist gut.

- Unter allen guten Paaren fixieren wir jetzt ein Paar (b, k) mit dem grösstmöglichen Wert k.
- Also: Es gilt $b^{2^k u} \equiv -1 \pmod{n}$.
  Und: Für alle $a \in \mathbb{Z}_n^*$ und alle $j \geq k+1$ gilt $a^{2^j u} \not\equiv -1 \pmod{n}$.

- Wir definieren $B = \{x \in \mathbb{Z}_n^* \mid x^{2^k u} \equiv \pm 1 \pmod{n}\}$.
- Diese Menge B enthält alle (betrügerischen) Basen, für die Miller-Rabin mit PROBABLY PRIME antwortet.
- Unser Ziel ist es, $|B| \leq |\mathbb{Z}_n^*|/2$ zu zeigen.

Beweis des Hauptresultats (2)

Zweiter Beweisschritt: $B \neq \mathbb{Z}_n^*$

- Nach dem Chinesischen Restsatz existiert ein $c \in \mathbb{Z}_n^*$ mit $c \equiv b \pmod{n_1}$ und mit $c \equiv 1 \pmod{n_2}$
- Aus $c \equiv b \pmod{n_1}$ folgt $c^{2^k u} \equiv b^{2^k u} \equiv -1 \pmod{n_1}$
  Aus $c \equiv 1 \pmod{n_2}$ folgt $c^{2^k u} \equiv 1 \pmod{n_2}$
- Daraus folgt: $c^{2^k u} \not\equiv 1 \pmod{n}$ und $c^{2^k u} \not\equiv -1 \pmod{n}$
- Ergo: $c^{2^k u} \notin B$, und daher wie gewünscht $B \neq \mathbb{Z}_n^*$

Beweis des Hauptresultats (3)

Letzter Beweisschritt: B ist Untergruppe von $\mathbb{Z}_n^*$

- Betrachte $x_1, x_2 \in B$.
  Also: $x_1^{2^k u} \equiv \pm 1 \pmod{n}$ und $x_2^{2^k u} \equiv \pm 1 \pmod{n}$
- Dann gilt $(x_1 x_2)^{2^k u} \equiv \pm 1 \pmod{n}$ und daher $x_1 x_2 \in B$.

- Damit ist der Beweis des Hauptresultats abgeschlossen.

Anmerkungen

Deterministischer Primzahltest in polynomieller Zeit (1)

- Gary L. Miller hat 1975 einen deterministischen Primzahltest entwickelt. Die Laufzeit dieses Tests beträgt $O(\log^4 n)$, falls die Erweiterte Riemannsche Vermutung stimmt.
  (Aus diesem Test hat sich dann der Miller-Rabin Test entwickelt.)

- Die indischen Mathematiker Manindra Agrawal, Neeraj Kayal und Nitin Saxena (AKS) haben 2002 einen deterministischen Primzahltest entwickelt. Die Laufzeit dieses Tests beträgt $O(\log^{12} n)$.

- Carl Pomerance und Hendrik Lenstra haben die Laufzeit von AKS auf $O(\log^6 n)$ verbessert.

Deterministischer Primzahltest in polynomieller Zeit (2)

Der AKS Primzahltest:

1. Check whether n is a perfect power:
   if $n = a^b$ for integers $a > 1$ and $b > 1$, then return COMPOSITE
2. Find the smallest r such that $\mathrm{ord}_r(n) > (\log_2 n)^2$
3. If $1 < a < \gcd(a, n)$ for some $a \leq r$, then return COMPOSITE
4. If $n \leq r$, then return PRIME
5. For $a = 1$ to $\sqrt{\phi(r)} \log_2 n$ do:
   if $(x + a)^n \neq x^n + a$ in $\mathbb{Z}_n[x]/(x^r - 1)$, then return COMPOSITE
6. Return PRIME