Primfaktorzerlegung und Primzahltests

Maximilian Scholz
Technische Universität Hamburg-Harburg

11. Juli 2014

Zusammenfassung

Mit dieser Ausarbeitung im Rahmen des Proseminars Mathematik will ich eine Einleitung in die Welt der Primzahlen im Allgemeinen und der Primfaktorzerlegung im besonderen schaffen.
Ich beginne mit einer Einleitung zu den Primzahlen und zeige anhand von Beispielen, wie sich die Problematik der Primfaktorzerlegung auf verschiedenen Wegen angehen lässt. Hierzu beleuchte ich nicht nur die Algorithmen im einzelnen sondern zeige auch, welche Werkzeuge benutzt werden und erkläre auch diese detaillierter, da mir deren Herleitungen in der Fachliteratur oft zu kurz kommen.
Letztendlich ist mein Ziel Verständnis in einem sehr begrenzten Bereich zu schaffen, anstatt in die Breite zu gehen. Denn nur durch wirkliches Verständnis können neue Entdeckungen wie zum Beispiel der AKS-Algorithmus gemacht werden.

Inhaltsverzeichnis

1 Einführung in Primzahlen

Die wichtigsten Dinge, die es zu Primzahlen im Bezug auf diesen Text gibt lassen sich wie folgt zusammenfassen:

- Definition: Jede natürliche Zahl > 1, die nur von sich selber und 1 geteilt wird, ist eine Primzahl. Dies lässt sich auch so beschreiben, dass eine Primzahl nur beim Teilen durch 1 und sich selbst keinen Rest hat.
- Der griechische Mathematiker Euklid hat bewiesen, dass es unendlich viele Primzahlen gibt. Würde dies nicht gelten, könnte es zu Problemen bei modernen kryptographischen Verfahren kommen, da diese auf immer größer werdenden Primzahlen beruhen.
- Ebenfalls von Euklid kommt der Beweis dafür, dass sich jede natürliche Zahl als Produkt von Primzahlen darstellen lässt. Später entdeckte Gauss, dass diese sogenannte Zerlegung bis auf die Reihenfolge der einzelnen Elemente eindeutig ist.
  Dies ist besonders interessant, weil es uns ermöglicht, zusammengesetzte Zahlen (also Zahlen, die nicht prim sind) daran zu erkennen, dass wir einen Faktor ungleich der Zahl oder 1 gefunden haben. Außerdem birgt dies die Grundlage zu manchen Angriffen auf kryptografische Verfahren wie RSA, auf die in diesem Text allerdings nicht weiter eingegangen wird.

Dieser Text wird sich im weiteren Verlauf vor allem mit der Zerlegung von Zahlen in deren Primfaktoren befassen.

2 Sieb des Eratosthenes

Das Sieb des Eratosthenes ist ein etwa 2000 Jahre altes Verfahren, um alle Primzahlen in einem gegebenen Zahlenbereich zu finden.

2.1 Idee

Das Sieb des Eratosthenes bestimmt alle Primzahlen N, indem es alle zusammengesetzten Zahlen streicht. Daher kommt der Name: Sieb.
Das Abbruchkriterium ist ein schönes Beispiel, wie unnötiger Aufwand vermieden werden kann.

2.2 Beispiel

Bevor die genaue Funktionsweise behandelt wird, hier einmal ein Beispiel für die Funktionsweise der Methode.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25				

Tabelle 1: Eratosthenes 1

Zu Beginn werden alle Zahlen in dem Bereich, in dem nach Primzahlen gesucht werden soll aufgeschrieben. Hier $n = 25$.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25				

Tabelle 2: Eratosthenes 2

Die Zahlen 0 und 1 sind per Definition keine Primzahlen, wir können sie somit streichen.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25				

Tabelle 3: Eratosthenes 3

Die erste nicht gestrichene Zahl ist die 2. Gleichzeitig ist die 2 die erste Primzahl. Da alles Zahlen die durch 2 teilbar sind nicht prim sind, können wir alle geraden Zahlen streichen.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25				

Tabelle 4: Eratosthenes 4

Die nächste nicht gestrichene Zahl ist die 3. Somit ist auch die 3 eine Primzahl. Wieder werden alle Vielfachen von 3 gestrichen.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25				

Tabelle 5: Eratosthenes 5

Die 4 ist schon gestrichen, da sie durch 2 teilbar ist. Die nächste nicht gestrichene Zahl ist die 5. Damit ist die 5 die nächste Primzahl und alle Vielfachen von 5 werden gestrichen.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25				

Tabelle 6: Eratosthenes 6

Da nur die Zahlen bis 25 betrachtet werden, ist die Methode an dieser Stelle fertig. Mit $5 = \sqrt{25}$ haben wir das Abbruchkriterium erreicht und damit alle möglichen Teiler der Zahlen < 25 abgearbeitet. Die übrigen Zahlen, die grün unterlegt sind, sind Primzahlen. Wieso wir fertig sind, wenn wir alle Vielfachen von 5 gestrichen haben, wird im nächsten Abschnitt deutlich.

2.3 Mathematik

2.3.1 Funktionsweise

Die Methode des quadratischen Siebes lässt sich wie folgt beschreiben:

1. Wähle eine natüriche Zahl $n > 1$. Dies ist die obere Grenze der Zahlen, in denen nach Primzahlen gesucht wird.

2. Die kleinste noch nicht gestrichene Zahl m mit $2 \leq m$ wird die neue Ausgangszahl. Im Beispiel sind das die 2 bei in Tabelle 3, die 3 bei Tabelle 4 und die 5 bei Tabelle 5.

3. Solange $m^2 \leq n$ gilt, streiche alle Vielfachen $c \cdot m$ ($c \in \mathbb{N}$) mit $m^2 \leq cm \leq n$. Im Beispiel durch die rote Färbung zu erkennen.

4. Übrig bleiben alle Primzahlen zwischen 0 und n. Im Beispiel durch die grüne Färbung zu erkennen.

Interessant ist zum einen das Abbruchkriterium. Wieso reicht es $m \leq \sqrt{n}$ zu betrachten um alle Primzahlen bis n zu finden. Außerdem sollte die intuitive

Funktionsweise des Streichens von Vielfachen nicht als Beweis dafür gesehen werden, dass die Methode des Siebens wirklich in jedem Fall funktioniert. Für diese beiden Teile wird im folgenden ein Beweise gezeigt.

2.3.2 Beweis

Beide Teile lassen sich zusammen beweisen:
Der kleinste Teiler einer zusammengesetzten Zahl n ist eine Primzahl p. Es gilt $p > 1$.
Da $p|n$ gilt, gilt auch $\frac{n}{p}|n$. Das Ergebnis von $\frac{n}{p}$ ist natürlich auch ein Teiler von n, da es mit p multipliziert n ergibt.
Da p der kleinste Teiler ist, gilt $p \leq \frac{n}{p}$, da jeder andere Teiler mindestens so großwie p sein muss. Daraus folgt:

$$p^2 \leq n$$

Wenn wir mit dem Sieb des Eratosthenes nach Primzahlen von 1 bis N suchen, werden also alle zusammengesetzten Zahlen $n < N$ beim Sieben mit einer Zahl p, für die gilt $p \leq \sqrt{n}$, gestrichen.
Die übrigen Zahlen müssen also Primzahlen sein.

3 Pollard Rho Methode

3.1 Geburtstagsproblem

Die Idee des Geburtstagsproblems ist die Frage, wie hoch die Wahrscheinlichkeit auf einem Geburtstag mit N Personen ist, dass zwei Personen am gleichen Tag Geburtstag haben.
Eine Möglichkeit das Problem einfacher zu machen ist, stattdessen das inverse Problem zu betrachten, also wie hoch die Wahrscheinlichkeit (P) ist, dass kein Geburtstag doppelt vorkommt.
Bei einem Besucher ist die Wahrscheinlichkeit

$$P(1) = \frac{356}{356}$$

da alle 356 Tage zur Wahl stehen. Bei zwei Besuchern ist die Wahrscheinlichkeit

$$P(2) = \frac{356}{356} \cdot \frac{364}{365}$$

bei drei Besuchern gilt

$$P(3) = \frac{356}{356} \cdot \frac{364}{365} \cdot \frac{363}{365}$$

Im Allgemeinen gilt:

$$P(N) = \frac{365 \cdot 364...(365 - N + 1)}{365^N}$$

Da wir große Primzahlen betrachten werden interessiert uns auch das Verhalten des Geburtstagsproblems für große N. Beim Betrachten von vielen Geburtstagen mit N Personen liegt, für große N, die Zahl der Wiederholungen die man durchschnittlich braucht um einen doppelten Geburtstag zu erhalten bei:

$$\sqrt{\frac{\pi \cdot N}{2}}$$

3.2 Hase Igel Algorithmus

Der Hase Igel Algorithmus wird benutzt, um Zyklen in Folgen zu finden. Die Idee ist, die Folge mit zwei Zeigern zu durchlaufen, dem Hase und dem Igel, die auf dem gleichen Feld starten. Falls es einen Zyklus in der Folge gibt, werden sich Hase und Igel irgendwann wieder treffen, da der schnellere Hase den langsameren Igel einholt und überrundet.

3.2.1 Beispiel

In diesem Beispiel ist eine Folge zu sehen, die einen Zyklus beinhaltet.

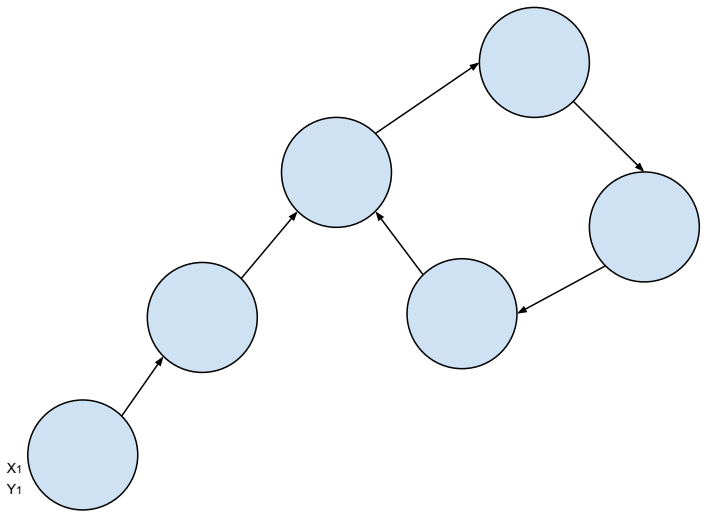

Abbildung 1: Schritt 1

Der Hase, hier dargestellt durch Y und der Igel, hier dargestellt durch X starten an der gleichen Position. Der Igel bewegt sich um ein Element, der Hase um zwei Elemente vorwärts.

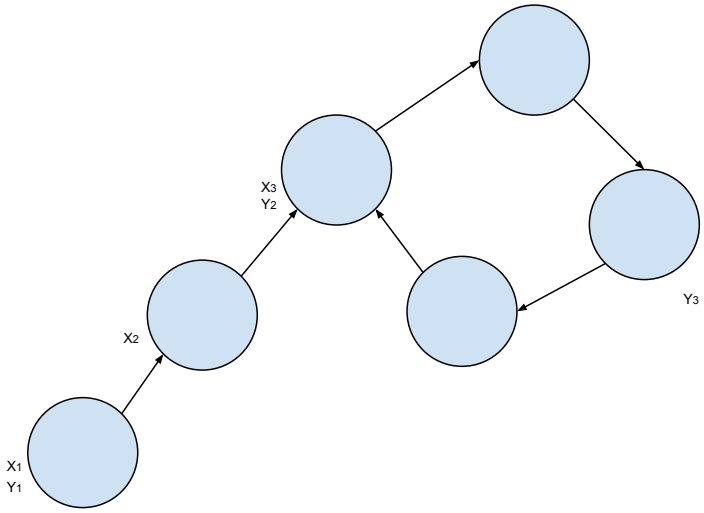

Abbildung 2: Schritt 2

Das ist der Zustand, bevor der Hase das erste mal im Kreis läuft. Es ist zu erkennen, dass der Hase doppelt so weit ist, wie der Igel.

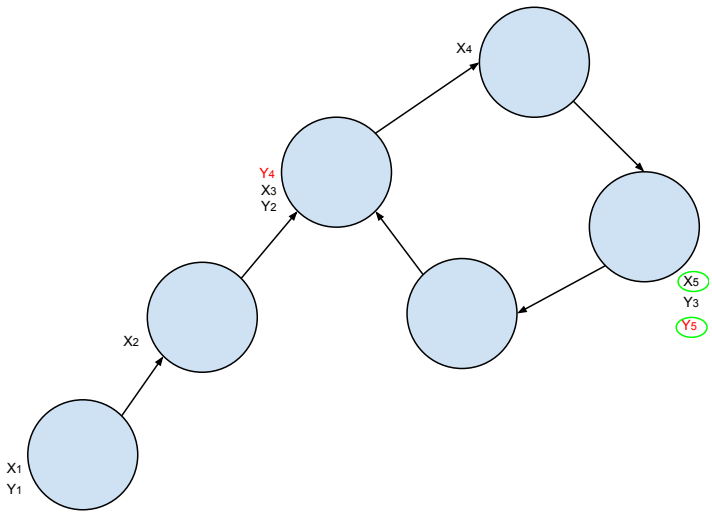

Abbildung 3: Schritt 3

Dies ist der Zustand, wenn Hase und Igel sich treffen. Die roten Zahlen zeigen, dass der Hase im Kreis läuft und die grün eingekreisten Zahlen zeigen, dass Hase und Igel sich in ihrem fünften Schritt treffen.

3.2.2 Mathematik

- Sei M eine endliche Menge mit der Abbildung $f : M \to M$. Die Menge M wird im Beispiel durch die Kreise dargestellt, die Abbildung f durch die Pfeile, die die Kreise verbinden.

- Man wähle $x_0 \in M$ und erzeuge die Folge $x_0, x_1, x_2, ...$ mit $x_{i+1} = f(x_i)$. Dies wird im Beispiel durch den Igel dargestellt, der sich entlang der Pfeile mit einer Schrittweite von 1 durch die Menge bewegt.

- $\exists\ i, j \in \mathbb{N}$, sodass $i \neq j$ und $x_i = x_j$ gilt. Hiermit wird gesagt, dass es einen Zyklus gibt. Wenn der Igel zwei mal auf das gleiche Feld kommt, muss er im Kreis gegangen sein.

- Die Folge $y_0, y_1, y_2, ...$ gegeben durch $y_0 = x_0$ und $y_{i+1} = f(f(y_i))$ ist gleich der Folge $x_0, x_2, x_4, ...$
  Hiermit wird der Hase beschrieben, der sich doppelt so schnell bewegt wie der Igel.

- Es gibt ein $c > 0$, sodass $x_c = x_{2c}$.

Dies ist die Bedingung dafür, dass es möglich ist, dass sich Hase und Igel wieder treffen und zwar wenn beide gleich viele Schritte gemacht haben. c für den Igel, $2c$ für den Hasen, da der Hase eine doppelt so große Schrittweite hat wie der Igel.

Ein Beweis für diesen Algorithmus wird in Abschnitt 3.3.5 geführt.

3.3 Pollard Rho Algorithmus

3.3.1 Kongruenz modulo p

Bevor wir mit dem nächsten Algorithmus anfangen können, brauchen wir noch ein weiteres Werkzeug.
Modulo Rechnen sollte bekannt sein. Weniger bekannt ist aber die Kongruenz (mod p).

$$a \equiv b \pmod{p} \Leftrightarrow p|(a - b)$$

Gesprochen wird es: a ist kongruent b modulo p. wenn zwei Zahlen kongruent modulo p sind, bedeutet es, dass sie beim modulo nehmen mit p den gleichen Rest haben. Dies führt wie oben beschrieben dazu, dass die Differenz von a und b durch p teilbar ist. Diese Eigenschaft wird durch folgende Umformung noch einmal verdeutlicht:

$$a \equiv b \pmod{p} \Leftrightarrow a = p \cdot x + r, \quad b = p \cdot y + r$$

$$a - b = p(x - y) + (r - r) = p(x - y)$$

$$p|p(x - y)$$

3.3.2 Idee

Die Idee für den Algorithmus basiert auf folgender Schlussfolgerung:
Sei n eine zusammengesetzte Zahl und p ein Primfaktor von n.
Gesucht sind a, b, sodass:
$a \equiv b \pmod{p} \Leftrightarrow p|a - b$
Daraus folgt $1 < ggT(a - b, n) \leq n$.
Wenn $a \neq b$ gilt, ist $ggT(a - b, n)$ ein nichttrivialer Primfaktor von n.

Der interessante Teil hierbei ist, die Folgerung, dass wenn $a \equiv b \pmod{p}$ gilt, auch $1 < ggT(a - b, n) \leq n$ gilt. Dies ist sehr einfach zu erklären, wenn man die Folgerung $a \equiv b \pmod{p} \Leftrightarrow p|a - b$ betrachtet. p ist Teiler von n und von $a - b$. Da p eine Primzahl ist, ist sie größer als 1 und maximal n da sie n teilt.

3.3.3 Algorithmus

Der Algorithmus funktioniert wie folgt:

- Sei $f(x)$ eine ganzzahlige Polynomfunktion und $s \in \mathbb{Z}$.
- Man erzeuge eine Folge von Pseudozufallszahlen mit:

  $$x_0 = s, x_{i+1} = f(x_i) \pmod{n}$$

  Die Folge wird irgendwann periodisch werden, da sie beschränkt ist und sich ihre Elemente dadurch irgendwann wiederholen müssen.

- Anstatt wie beim Hase Igel Algorithmus in 3.2 danach zu suchen, dass sich Hase und Igel treffen($x_k = y_k$) suchen wir nach einem Ergebnis, das folgende Bedingung erfüllt: $1 < ggT(x_k - y_k, n) < n$. Denn damit haben wir einen Teiler von n gefunden, was unser Ziel war. Da in dem Fall $x \equiv y \pmod{p}$ gilt, können wir den Hase Igel Algorithmus benutzen, denn auch hier suchen wir nach einer Wiederholung und zwar nach zwei verschiedene Zahlen, die kongruent modulo p sind.

3.3.4 Beispiel

Gesucht: Primfaktorzerlegung von N=143

Parameter: $x_0 = y_0 = 0,\ f(x) = (x^2 + 1) \pmod{N}$

k	$x_k = f(x_{k-1})$	$y_k = f(f(y_{k-1}))$	$ggT(x_k - y_k, N)$
0	0	0	0
1	1	2	1
2	2	26	1
3	5	15	1
4	26	26	143
5	105	15	1
6	15	26	11

Zu sehen ist hier der Ablauf Des Algorithmus wenn $N = 143$ gewählt wird. Die Funktion $f(x)$ ist eine oft genutzte Standardfunktion, es können aber auch andere benutzt werden. In jedem Schritt bewegen sich Igel und Hase weiter, indem f entsprechend oft angewandt wird, und der grte gemeinsame Teiler wird berechnet. Wenn wir einen ggt gefunden haben, der die Bedingung $1 < ggT(x_k - y_k, n) < n$ erfüllt, hat der Algorithmus einen Teiler von N gefunden. In diesem Fall die 11. Mit $\frac{143}{11} = 13$ erhält man den zweiten Primfaktor.

3.3.5 Mathematik

Es gibt drei Dinge, die noch zu beweisen sind. Zuerst betrachten wir die Behauptung, dass es mit dem Hase Igel Algorithmus möglich ist im Kreis zu laufen. Der folgende Ausdruck sagt, dass es möglich ist mit verschieden vielen Schritten auf das gleiche Element zu kommen:

$$\exists\ i, j \in \mathbb{N}, \text{sodass } i \neq j \text{ und } x_i = x_j$$

Als nächstes brauchen wir eine Funktion g, die die natürlichen Zahlen auf eine Menge M abbildet:

$$\text{Sei } g : \mathbb{N} \to M \text{ gegeben durch } g(t) = f^t(x_0)$$

Da M beschränkt ist, kann g nicht injektiv sein. Daraus folgt:

$$\exists\ i, j \in \mathbb{N}, i \neq j \text{ , sodass } g(i) = g(j) \text{ und damit } x_i = x_j \text{ bei } i \neq j$$

Als nächstes wollen wir beweisen, das der Hase sich so bewegt, wie wir es wollen. Wir behaupten also:

$$\text{Die Folge } y_0, y_1, y_2, ... \text{gegeben durch } y_0 = x_0 \text{und } y_{i+1} = f(f(y_i))$$

$$\text{ist gleich der Folge } x_0, x_2, x_4, ...$$

Der Beweis ist sehr einfach. Aus $x_{m+2} = f(f(x_m))$ folgt $y_m = x_{2m}$.

Als letztes müssen wir beweisen, dass sich Hase und Igel treffen können. Wir fordern also:

$$\text{Es gibt ein } c > 0, \text{ sodass } x_c = x_{2c}$$

Angenommen es gibt j und i, sodass $x_i = x_j$ für $j > i$ gilt.

Falls $c \geq i$ gilt und $2c = c + k(j - i) \geq i$ ist, mit $k \geq 0$, soll $x_c = x_{2c}$ gelten. Da sich der Hase und Igel treffen, wenn wir zwei Zahlen gefunden haben, die kongruent modulo p sind, darf die Differenz von c und $2c$ nur ein Vielfaches von p sein. da $j - i$ ein Vielfaches von p ist, müssen wir die obige Forderung erfüllen. Dafür wählt man einfach:

$$k \geq 0, \text{ sodass } c = k(j - i) \geq i \text{ gilt.}$$

Und erhält so das gesuchte c.

Hiermit haben wir gezeigt, dass der Hase Igel Algorithmus funktioniert und der darauf aufbauende Pollard Rho Algorithmus auch.

3.4 Komplexität

Neben der Tatsache, dass der Algorithmus funktioniert, ist auch noch interessant, wie schnell er ist. Dafür betrachten wir einmal die einzelnen Teile, die wir benutzen.

Wir suchen keine Geburtstag aber Wiederholungen modulo p, in der Form von: $x_k \equiv x_{2k} \pmod{p}$. Da $p \leq \sqrt{n}$, also $\sqrt{p} \leq \sqrt[4]{n}$ gilt, erhalten wir mit Hilfe des Geburtstagsproblems einen durchschnittlichen Aufwand von:

$$\mathcal{O}(\sqrt{\frac{\pi p}{2}}) \Rightarrow \mathcal{O}(\sqrt[4]{n})$$

Für große Zahlen finden wir also nach durchschnittlich $\sqrt[4]{n}$ Versuchen einen Primfaktor. Um eine genauere Aussage treffen zu können, müssen wir außerdem noch den Aufwand betrachten, der pro Schritt erforderlich ist.
Zum einen ist das der Euklidischen Algorithmus zum Finden des größten gemeinsamen Teilers und zum anderen der Aufwand, den die Funktion f erfordert. Zusammengefasst sieht die Komplexität wie folgt aus:

$$(\mathcal{O}(\text{Euklid}) + \mathcal{O}(f)) \cdot \mathcal{O}(\sqrt[4]{n})$$

4 Fazit

Wichtig zu erwähnen ist, dass der Pollard Rho Algorithmus ein probabilistischer Algorithmus ist. Er findet also nur durchschnittlich in einer bestimmten Zeit eine Primfaktorzerlegung. Je nach Wahl der Funktion f und der Anfangswerte, kann man deutlich schneller oder aber gar nicht auf eine Zerlegung stoÃ§en. Die Wahrscheinlichkeit, mit der sich ein Primfaktor findet, lässt sich durch die Zahl der Schritte festlegen. Die Wahrscheinlichkeit nach $\sqrt{t} \cdot \mathcal{O}(\sqrt[4]{n})$ Schritten einen Primfaktor zu finden ist $> 1 - e^{-t}$.
Wenn einfach nur geprüft werde soll, ob eine gegebene Zahl eine Primzahl ist, kann mit dieser Methode nicht eindeutig gesagt werden, dass es eine Primzahl ist. Nur das Gegenteil, dass die Zahl zusammengesetzt ist, lässt sich durch das Finden einer Zerlegung zeigen.
Als Alternative gibt es sogenannte deterministische Algorithmen. Diese finden immer eine Lösung, sind im Durchschnitt aber oft deutlich langsamer als probabilistische, was den probabilistischen Algorithmen in der Praxis eine Existenzberechtigung gibt. Beispiele für schnelle deterministische Primfaktorzerlegungen sind das sogenannte Quadratisches Sieb oder das Zahlkörpersieb.

Zuletzt sei noch erwähnt, dass es 2002 drei indischen Studenten gelungen einen deterministischen Primzahltest zu entwickeln, der in polynomialer Zeit läuft.

Literatur

[1] Niels Lauritzen (2006) Concrete Abstract Algebra
ISBN 0-521-53410-0

[2] www.bk2boint.dnsalias.org/int_neu/tl_files/Material\
%20Informatik/erathostenes/sieb.pdf
Juni 2014.