Primzahl-Testverfahren

Hans-Gert Gräbe, Leipzig

http://www.informatik.uni-leipzig.de/~graebe

2. Februar 2005

1 Einfache Primzahl-Testverfahren

Ist eine große ganze Zahl gegeben, so ist es in vielen Fällen einfach zu erkennen, dass es sich um eine zusammengesetzte Zahl handelt. So sind z.B. 50 % aller Zahlen gerade. Eine Probedivision durch die 4 Primzahlen < 10 lässt uns bereits 77 % aller Zahlen als zusammengesetzt erkennen. Übrig bleibt eine Grauzone von möglichen Kandidaten von Primzahlen, für die ein aufwändigeres Verfahren herangezogen werden muss, um die Primzahleigenschaft auch wirklich zu beweisen.

Ein erstes solches Verfahren ist die **Methode der Probedivision**. Die folgenden Prozeduren sind in **MuPAD** [3] geschrieben, sehen aber in jedem anderen Computeralgebra-System (CAS) ähnlich aus.

```
primeTestByTrialDivision:=proc(m:Dom::Integer) local z;
begin
  if (m<3) then return(bool(m=2)) end_if;
  z:=2;
  while z*z<=m do
      if m mod z = 0 then return(FALSE) end_if; z:=z+1;
  end_while;
  TRUE;
end;
```

oder in etwas effizienterer Form (Aufwand auf $\frac{1}{3}$ reduziert, indem nur auf Faktoren der Form $6k \pm 1$ getestet wird)

```
moreEfficientPrimeTestByTrialDivision:=proc(m:Dom::Integer) local z;
begin
  if (m<2) then return(FALSE)
    elif (m mod 2 = 0) then return(bool(m=2));
    elif (m mod 3 = 0) then return(bool(m=3))
  end_if;
  z:=5;
  while z*z<=m do
      if m mod z = 0 then return(FALSE) end_if; z:=z+2;
      if m mod z = 0 then return(FALSE) end_if; z:=z+4;
  end_while;
  TRUE;
end;
```

`bool` erzwingt dabei die boolesche Auswertung des zurückzugebenden Ausdrucks, da sonst standardmäßig angenommen wird, dass es sich um eine Gleichung handelt, der man in einem symbolischen Kontext nur dann einen Wahrheitswert zuordnen kann, wenn in ihr keine (ungebundenen) Symbole auftreten.

Der Aufwand für dieses Verfahren ist am größten, wenn m eine Primzahl ist, d.h. wirklich alle Tests bis zum Ende durchgeführt werden müssen. Die Laufzeit ist dabei von der Größenordnung $O(\sqrt{m})$, also exponentiell in der Bitlänge $l(m) = \log_2(m)$ der zu untersuchenden Zahl ($\sqrt{m} = 2^{l(m)/2}$).

Dieses Verfahren liefert uns allerdings für eine zusammengesetzte Zahl neben der Antwort auch einen Faktor, so dass eine rekursive Anwendung nicht nur die Primzahleigenschaft prüfen kann, sondern für Faktorisierungen geeignet ist. Experimente mit CAS zeigen dagegen, dass Faktorisieren offensichtlich um Größenordnungen schwieriger ist als der reine Primzahltest.

Gleichwohl wird in allen CAS der Test mit Probedivision für eine Liste von kleinen Primzahlen als Vortest angewendet und die aufwändigeren Verfahren nur für solche Zahlen eingesetzt, die „nicht offensichtlich“ keine Primzahlen sind. In der folgenden Prozedur ist `smallPrimes` eine Liste aller „kleinen“ Primzahlen:

```
smallPrimesTest:=proc(m:Dom::Integer) local i,smallPrimes;
begin
  smallPrimes:=[2, 3, 5, 7, 11, 13, 17, 19, 23, 29];
  if (m<2) then return(FALSE) end_if;
  for i in smallPrimes do
       if (m mod i = 0) then return(bool(m=i)) end_if;
  end_for;
  FAIL;
end;
```

In dieser Prozedurdefinition wird eine dreiwertige Logik ausgenutzt, die neben den (sicheren) Wahrheitswerten `TRUE` und `FALSE` auch noch die Möglichkeit `FAIL` („nicht entschieden“) erlaubt.

2 Der Restklassenring $\mathbb{Z}_m$

Die weiteren Primtestverfahren beruhen alle auf zahlentheoretischen Zusammenhängen, die sich beim Rechnen mit Resten ergeben.

Bekanntlich nennt man zwei Zahlen $a, b \in \mathbb{Z}$ *kongruent modulo* m (und schreibt $a \equiv b \pmod{m}$), wenn ihre Differenz durch m teilbar ist, also bei Division durch m der Rest 0 bleibt. So gilt $127 \equiv 1 \pmod{7}$, aber ebenso $127 \equiv 8 \pmod{7}$, denn in beiden Fällen ist die Differenz durch 7 teilbar.

Die eingeführte Relation ist eine Äquivalenzrelation, so dass wir die zugehörigen Äquivalenzklassen betrachten können, die als *Restklassen* bezeichnet werden. Die Restklasse $\pmod{7}$, in der sich die Zahl 1 befindet, besteht etwa aus den Zahlen

$$[1]_7 = \{\ldots, -20, -13, -6, 1, 8, 15, \ldots, 127, \ldots\} = \{7k + 1 \mid k \in \mathbb{Z}\}.$$

Die Darstellungen $z \equiv 1 \pmod{7}$, $7 \mid (z - 1)$, $z = 7k + 1$, $z \in [1]_7$ und $[z]_7 = [1]_7$ sind also äquivalent zueinander. Wir werden diese unterschiedlichen Schreibweisen im Weiteren frei wechselnd verwenden. Die Menge der Restklassen modulo m bezeichnen wir mit $\mathbb{Z}_m$.

Addition und Multiplikation sind mit der Restklassenbildung verträglich, so dass die Menge $\mathbb{Z}_m$ sogar einen Ring bildet. Im Gegensatz zu den ganzen Zahlen kann dieser Ring aber Nullteiler besitzen. So ist etwa $2, 3 \not\equiv 0 \pmod{6}$, dagegen $2 \cdot 3 = 6 \equiv 0 \pmod{6}$.

2.1 Ein wichtiger Satz über endliche Mengen

In den weiteren Betrachtungen wird mehrfach der folgende wichtige Satz über endliche Mengen angewendet.

Satz 1 *Sei* $\phi: M_1 \longrightarrow M_2$ *eine Abbildung zwischen zwei gleichmächtigen endlichen Mengen in sich selbst. Dann gilt*

$$\phi \text{ ist injektiv, d.h. } \phi(x_1) = \phi(x_2) \Rightarrow x_1 = x_2 \tag{1}$$

genau dann, wenn

$$\phi \text{ ist surjektiv, d.h. } \forall\, y \in M_2\ \exists\, x \in M_1 : y = \phi(x) \tag{2}$$

Beweis: Offensichtlich, denn

(1) heißt: jedes $y \in M_2$ hat *höchstens* ein Urbild,

(2) heißt: jedes $y \in M_2$ hat *mindestens* ein Urbild

In Wirklichkeit hat wegen der Gleichmächtigkeit in beiden Fällen jedes $y \in M_2$ *genau* ein Urbild $x \in M_1$. $\square$

Dieser Satz ist für unendliche Mengen falsch. So ist z.B. die Abbildung $\phi_1 : \mathbb{N} \to \mathbb{N}$ via $\phi_1(n) = 2\,n$ zwar injektiv, aber nicht surjektiv, die Abbildung $\phi_2 : \mathbb{N} \to \mathbb{N}$ via $\phi_2(n) = n \; div \; 10$ surjektiv, aber nicht injektiv.

2.2 Die Gruppe der prime Restklassen

Eine Restklasse $[a]_m$ heißt *prim*, wenn ein (und damit jeder) Vertreter dieser Restklasse zu m teilerfremd ist, wenn also $\gcd(a, m) = 1$ gilt. So sind etwa $\pmod 7$ alle Restklassen verschieden von $[0]_7$ prim, $\pmod 8$ dagegen nur die Restklassen $[1]_8, [3]_8, [5]_8$ und $[7]_8$ und $\pmod 6$ gar nur die beiden Restklassen $[1]_6$ und $[5]_6$. Wir bezeichnen die Menge der primen Restklassen mit $\mathbb{Z}_m^*$.

Prime Restklassen haben bzgl. der Multiplikation eine besondere Eigenschaft. Es gilt für eine prime Restklasse $a \pmod m$ die *Kürzungsregel*

$$a \cdot x \equiv a \cdot y \pmod m \;\Rightarrow\; x \equiv y \pmod m.$$

Dies lässt sich sofort aus $m \,|\, (a\,x - a\,y) = a\,(x - y)$ und $\gcd(a, m) = 1$ herleiten.

Anders formuliert: Die Multiplikationsabbildung

$$\phi_a : \mathbb{Z}_m \longrightarrow \mathbb{Z}_m \quad \text{via} \quad x \pmod m \mapsto a\,x \pmod m$$

ist injektiv und somit nach obigem Satz als Abbildung zwischen gleichmächtigen endlichen Mengen auch surjektiv. Zu einer primen Restklasse $a \in \mathbb{Z}_m$ gibt es also stets ein (eindeutig bestimmtes) $a' \in \mathbb{Z}_m$, so dass $\phi_a(a') = a \cdot a' \equiv 1 \pmod m$ gilt. a' (auch a^{-1}) bezeichnet man als die zu a *inverse Restklasse*.

Da die Menge aller Restklassen $\mathbb{Z}_m$ endlich ist, ist es auch die Menge der primen Restklassen $\mathbb{Z}_m^*$. Ihre Anzahl bezeichnet man mit dem Symbol $\phi(m)$. Die zugehörige Funktion in Abhängigkeit von m bezeichnet man als die *Eulersche ϕ-Funktion*.

Produkt- und Inversenbildung führen nicht aus der Menge der primen Restklassen $\mathbb{Z}_m^*$ heraus. Eine solche unter Multiplikation und Inversenbildung abgeschlossene Struktur bezeichnet man auch als **Gruppe**. Die Gruppe $\mathbb{Z}_m^*$ enthält genau $\phi(m)$ Elemente. Allgemeine Gruppeneigenschaften spezifizieren zu interessanten zahlentheoretischen Sätzen.

So bezeichnet man etwa für ein Element a einer Gruppe G die Mächtigkeit der von a erzeugten Untergruppe $\langle a \rangle = \{1, a, a^2, \ldots\} \subset G$ als die *Ordnung* $ord(a)$ von a. Im Falle endlicher Gruppen ist diese Ordnung immer endlich und es gilt

$$ord(a) = n_0 = \min\{n > 0 \,:\, a^n = 1\}.$$

Weiter gilt

$$a^n = 1 \;\Rightarrow\; n_0 \,|\, n,$$

denn aus $n = q\,n_0 + r$ mit $0 \le r < n_0$ (Division mit Rest) und $1 = a^n = (a^{n_0})^q \cdot a^r = 1 \cdot a^r$ folgt $r = 0$.

Satz 2 (Satz von Lagrange) *Ist H eine Untergruppe von G, so ist $|H|$ ein Teiler von $|G|$.*

Beweis: Für $a \in G$ betrachten wir die Mengen $aH = \{ah \mid h \in H\}$, die man auch als *Nebenklassen* von a bzgl. H bezeichnet. Zwei solche Nebenklassen a_1H und a_2H sind

(1) gleichmächtig (die Multiplikationsabbildung ϕ_a bildet H umkehrbar eindeutig auf aH ab, Umkehrabbildung ist $\phi_{a^{-1}}$)

(2) disjunkt, wenn $a_1^{-1}a_2 \notin H$ (wäre $a \in a_1H \cap a_2H$, also $a = a_1h_1 = a_2h_2$, so wäre $a_1^{-1}a_2 = h_1h_2^{-1} \in H$)

(3) und fallen für $a_1^{-1}a_2 \in H$ zusammen ($a_2 \cdot h = a_1 \cdot (a_1^{-1}a_2h)$.

G zerfällt also in paarweise disjunkte gleichgroße Teile, von denen ein Teil $H = 1 \cdot H$ ist. □

Insbesondere ist die Gruppenordnung $|G|$ durch die Ordnung $ord(a)$ jedes Elements teilbar und es gilt stets $a^{|G|} = 1$. Für die Gruppe der primen Restklassen bedeutet das:

Folgerung 1 (Satz von Euler) *Ist $(a, m) = 1$, so gilt $a^{\phi(m)} \equiv 1 \pmod m$.*

Ein Spezialfall dieses Satzes:

Folgerung 2 (Kleiner Satz von Fermat) *Ist m eine Primzahl und $1 < a < m$, so gilt $a^{m-1} \equiv 1 \pmod m$.*

3 Der Fermat-Test

Den kleinen Satz von Fermat kann man zu einem Verfahren ausbauen, mit dem zusammengesetzte Zahlen sicher erkannt werden können, ohne deren Faktorzerlegung zu berechnen. Dieser Test beruht auf der folgenden Umkehrung des Satzes:

> *Gilt für* **eine** *Zahl a mit $1 < a < m$* **nicht** *$a^{m-1} \equiv 1 \pmod m$, so kann m keine Primzahl sein.*

Eine Realisierung in MuPAD hätte etwa folgende Gestalt:

```
singleFermatTest:=proc(m:Dom::Integer, a:Dom::Integer) local D;
begin
D:=Dom::IntegerMod(m);
iszero(D(a)^(m-1)-1);
end;
```

Hierbei wird das Potenzieren in $\mathbb{Z}_m$ ausgeführt, d.h. in jedem Zwischenschritt sofort $\pmod m$ reduziert und nicht zuerst a^{m-1} als ganze Zahl berechnet (was zu einer Riesenzahl führen würde) und dann erst $\pmod m$ reduziert. Wir haben dabei die Möglichkeiten des Domain-Konzepts von MuPAD genutzt, das so in anderen CAS nicht zur Verfügung steht.

Mit binärem Potenzieren (wiederholtes Quadrieren) sind die Kosten dieses Verfahrens von der Größenordnung $O(l(m)^3)$ ($\phi(m) \sim m$, $\log(m)$ Multiplikationen von Zahlen der Länge $l(m)$ mit anschließender Restreduktion), also polynomial in der Bitlänge der zu untersuchenden Zahl.

Der Fermat-Test ist allerdings nur ein notwendiges Kriterium. Genauer gesagt können wir aus $a^{m-1} \not\equiv 1 \pmod m$ mit Sicherheit schließen, dass m eine zusammengesetzte Zahl ist. Ansonsten können wir den Test mit einer anderen Basis a wiederholen, weil vielleicht zufällig $ord(a) \mid m - 1$ galt.

Auf dieser Basis kann man folgenden **Las-Vegas-Test** ansetzen:

Mache den Fermat-Test für c verschiedene (zufällig gewählte) Basen $a_1, \ldots, a_c$.

Ist einmal $a_i^{m-1} \not\equiv 1 \pmod{m}$, so ist m garantiert eine zusammengesetzte Zahl.

Ist stets $a_i^{m-1} \equiv 1 \pmod{m}$, so ist m wahrscheinlich (hoffentlich mit großer Wahrscheinlichkeit) eine Primzahl.

Eine MuPAD-Implementierung hätte etwa folgende Gestalt:

```
LasVegasFermatTest:=proc(m:Dom::Integer, c:Dom::Integer)
  local D,i,a,r;
begin
  D:=Dom::IntegerMod(m);
  r:=random(m); /* r ist Zufallszahlengenerator */
  for i from 1 to c do
      a:=r();
      if igcd(a,m) <> 1 then return(FALSE) end_if;
         /* a ist keine prime Restklasse, m also zusammengesetzt */
      if not iszero(D(a)^(m-1)-1) then return(FALSE) end_if;
         /* der FermatTest schlägt zu */
  end_for;
  TRUE;
end;
```

`r:=random(m)` gibt dabei gemäß der Dokumentation keine Zufallszahl, sondern eine (nullstellige) Funktionsdefinition zurück, `r()` ruft diese auf. Ist a zufällig keine prime Restklasse, dann ist m zusammengesetzt und der berechnete gcd sogar ein echter Teiler von m. Dieser (sehr selten auftretende) Fall wird gesondert abgefangen.

Was können wir über die Wahrscheinlichkeit im unsicheren Zweig dieses Tests aussagen?

Satz 3 *Die Menge*

$$P_m := \{a \in \mathbb{Z}_m^* \ : \ a^{m-1} \equiv 1 \pmod{m}\}$$

der Restklassen $\pmod{m}$*, für welche der Fermat-Test kein Ergebnis liefert, ist eine Untergruppe der Gruppe der primen Restklassen $\mathbb{Z}_m^*$.*

Nach dem Satz von Lagrange ist somit $|P_m|$ ein Teiler von $\phi(m) = |\mathbb{Z}_m^*|$. Ist m also zusammengesetzt und $P_m \neq \mathbb{Z}_m^*$, dann erkennt der Fermat-Test für eine zufällig gewählte Basis in wenigstens der Hälfte der Fälle, dass m zusammengesetzt ist.
In diesem Fall ist die Wahrscheinlichkeit, dass im unsicheren Zweig des LasVegas-FermatTests m keine Primzahl ist, höchstens 2^{-c}, also bei genügend vielen Tests verschwindend klein.

Ist andererseits m eine zusammengesetzte Zahl und $P_m = \mathbb{Z}_m^*$, so kann der Fermat-Test m prinzipiell nicht von Primzahlen unterscheiden. Gibt es solche Zahlen ?

Antwort: Ja, z.B. die Zahl $561 = 3 \cdot 11 \cdot 17$, denn für $a \in \mathbb{Z}_{561}^*$ gilt

$$a^{560} \equiv 1 \pmod{3},\ a^{560} \equiv 1 \pmod{11} \text{ und } a^{560} \equiv 1 \pmod{17},\ \text{also } a^{560} \equiv 1 \pmod{561}.$$

4 Der Satz von Carmichael

Dieses Beispiel, in welchem der Fermat-Test fehlschlägt, ist nicht zufällig gefunden worden, sondern ergibt sich aus folgenden Überlegungen:
Ist $m = p_1^{a_1} \ldots p_k^{a_k}$ die Primfaktorzerlegung von m und $m_i = p_i^{a_i}$, so gilt mit $x^{\phi(m_i)} \equiv 1 \pmod{m_i}$ auch für jedes Vielfache c von m_i die Beziehung $x^c \equiv 1 \pmod{m_i}$. Nehmen wir insbesondere

ein **gemeinsames** Vielfaches der m_i, etwa $c = \text{lcm}(\phi(m_1), \ldots, \phi(m_n))$, so gilt $x^c \equiv 1 \pmod{m_i}$ für **alle** $i = 1, \ldots, n$, also $x^c - 1 \,|\, \text{lcm}(m_1, \ldots, m_n) = m$, denn die Faktoren m_i sind paarweise teilerfremd.

Satz 4 (Satz von Carmichael) *Ist $m = p_1^{a_1} \ldots p_k^{a_k}$ die Primfaktorzerlegung von m, so gilt*

$$a^{\psi(m)} \equiv 1 \pmod{m}$$

mit

$$\psi(m) = \text{lcm}(p_1^{a_1-1}(p_1 - 1), \ldots, p_k^{a_k-1}(p_k - 1))$$

Die Zahl $\psi(m)$ bezeichnet man auch als die Carmichael-Zahl *von m.*

Beweis: Für eine Primzahlpotenz p^a gilt offensichtlich $\phi(p^a) = p^a - p^{a-1}$. $\square$

Beispiel 1 $\psi(12) = 2,\ \psi(18) = 6,\ \psi(24) = 4,\ \psi(36) = 6.$

$c = \psi(m)$ ist für zusammengesetzte Zahlen m also ein deutlich kleinerer Exponent als $\phi(m)$, für welchen immer noch

$$a \in \mathbb{Z}_m^* \ \Rightarrow\ a^c \equiv 1 \pmod{m}$$

gilt und welcher damit ein gemeinsames Vielfaches der Gruppenordnungen aller Elemente $a \in \mathbb{Z}_m^*$ ist. Es stellt sich heraus, dass c in vielen Fällen der kleinste solche Exponent ist und in anderen Fällen der kleinste solche Exponent gerade gleich $\frac{c}{2}$ ist. Damit können wir auch genauer sagen, für welche Zahlen der Fermat-Test fehlschlägt.

Satz 5 *Für die ungerade zusammengesetzte Zahl m schlägt der Fermat-Test genau dann systematisch fehl, wenn $\psi(m)$ ein Teiler von $m - 1$ ist.*
Solche Zahlen kann der Fermat-Test also prinzipiell nicht von Primzahlen unterscheiden, wenn der Test mit einem $a \in \mathbb{Z}_m^*$ ausgeführt wird.

Zusammengesetzte Zahlen dieser Art nennt man *Carmichael-Zahlen.* Weitere Carmichael-Zahlen sind $1105 = 5 \cdot 13 \cdot 17$ und $1729 = 7 \cdot 13 \cdot 19$.

Aufgabe 1

a) Zeigen Sie, dass Carmichael-Zahlen m stets quadratfrei sind und immer wenigstens 3 Primfaktoren haben. Führen Sie dazu die Annahmen $m = p^a \cdot q$ und $m = p \cdot q$ jeweils zum Widerspruch.

b) Zeigen Sie, dass $N = (6t + 1)(12t + 1)(18t + 1)$ eine Carmichaelzahl ist, wenn $6t + 1, 12t + 1$ und $18t + 1$ Primzahlen sind.

Carmichaelzahlen sind unter den Primzahlen also recht selten. So gibt es unter den Zahlen $< 10^{15}$ nur etwa 10^5 solcher Zahlen. Andererseits gibt es unendlich viele solche Zahlen und Alford, Granville und Pomerance (1994) haben sogar gezeigt, dass sich die Anzahl $C(x)$ der Carmichaelzahlen kleiner x durch $C(x) \geq x^{2/7}$ nach unten hin abschätzen lässt, d.h. ihre Zahl exponentiell mit der Bitlänge von x wächst.

5 Der Rabin-Miller-Test

Ein Primzahltest ohne solche systematischen Ausnahmen beruht auf der folgenden Verfeinerung des Fermat-Tests: Für eine Primzahl m und eine zufällig gewählte Zahl $1 < a < m$ muss $a^{m-1} \equiv 1 \pmod{m}$ gelten. Ist $m - 1 = 2^t \cdot q$ die Zerlegung des Exponenten in eine Zweierpotenz und einen

ungeraden Anteil q, so ergibt die Restklasse $b = a^q \pmod m$ nach t-maligem Quadrieren den Rest 1. Bezeichnet u das Element in der Folge $\{b, b^2, b^4, b^8, \ldots, b^{2^t}\}$, wo erstmals $u^2 \equiv 1 \pmod m$ gilt, so muss, wenn m eine Primzahl ist, $u \equiv -1 \pmod m$ gelten. Ist dagegen m keine Primzahl, so hat die Kongruenz $u^2 \equiv 1 \pmod m$ noch andere Lösungen.

In der Tat, ist $m = p \cdot q$ für zwei teilerfremde Faktoren p, q, so gibt es nach dem chinesischen Restklassensatz eine Restklasse $a_1 \pmod m$ mit $a_1 \equiv 1 \pmod p$ und $a_1 \equiv -1 \pmod q$ und eine weitere Restklasse $a_2 \pmod m$ mit $a_2 \equiv -1 \pmod p$ und $a_2 \equiv 1 \pmod q$. Für beide gilt $a_i^2 \equiv 1 \pmod p$ und $a_i^2 \equiv 1 \pmod q$, also auch $a_i^2 \equiv 1 \pmod m$, aber $a_i \not\equiv \pm 1 \pmod m$.

Wählt man a zufällig aus, so ist die Wahrscheinlichkeit, dass u bei zusammengesetztem m auf eine solche Restklasse trifft, d.h. $u \not\equiv -1 \pmod m$, aber $u^2 \equiv 1 \pmod m$ gilt, groß. Eine genauere Analyse zeigt, dass diese Wahrscheinlichkeit sogar wenigstens $\frac{3}{4}$ beträgt, d.h. ein Las-Vegas-Test auf dieser Basis mit c Durchläufen nur mit der Wahrscheinlichkeit 4^{-c} fehlerhaft antwortet.

Auf der beschriebenen Verfeinerung beruht der folgende **Rabin-Miller-Test**

```
primeTestRabinMiller:=proc(m:Dom::Integer,c:Dom::Integer)
  local a,q,b,t,i,j,r,D;
begin
  D:=Dom::IntegerMod(m);
  q:=m-1; t:=0; r:=random(m);
  while q mod 2 = 0 do q:=q/2; t:=t+1 end_while;
        /* nun ist  m-1 = 2^t * q */
  for i from 1 to c do
    a:=r(); b:=D(a)^q;
    if iszero(b-1) or iszero(b+1) then next end_if;
    /* keine Information, wenn b = 1 oder b = -1 (mod m) */
    for j from 1 to (t-1) do
      /* nun ist  b <> 1 oder -1 (m) */
      b:=b*b;
        if iszero(b-1) then return(FALSE) end_if;
        if iszero(b+1) then break end_if;   /* keine Information */
    end_for;
    if iszero(b+1) then next;         /* keine Information */
    else return(FALSE) end_if;
  end_for;
  TRUE;
end_proc;
```

Die Hauptschleife (Iterationsvariable i) führt den eigentlichen Test für c verschiedene zufällig gewählte Basen a aus. In jedem dieser Tests wird zunächst $b \equiv a^q \pmod m$ berechnet. Ist bereits $b \equiv \pm 1 \pmod m$, so kann für diese Basis keine Aussage getroffen werden. Ansonsten quadrieren wir b:

> Erhalten wir den Rest 1, so war $b \not\equiv \pm 1 \pmod m$, aber $b^2 \equiv 1 \pmod m$, also ist m nicht prim.
>
> Erhalten wir den Rest -1, so wird im nächsten Schritt $b^2 \equiv 1 \pmod m$, woraus wir jedoch kein Kapital schlagen können. Aus der gewählten Basis a kann keine Aussage getroffen werden.
>
> Anderenfalls quadrieren wir noch einmal.

Das Quadrieren ist nach $(t-1)$ Schritten abzubrechen, denn $b^{2^t} \equiv a^{m-1} \pmod m$. Da nach Verlassen der inneren Schleife stets $b \not\equiv 1 \pmod m$ gilt, kommen folgende Fälle in Frage:

> $b^2 \not\equiv 1 \pmod m$, also m nicht prim nach dem Fermat-Test.
>
> $b \not\equiv -1 \pmod m$, aber $b^2 \equiv 1 \pmod m$, also ist m nicht prim.

$b \equiv -1 \pmod{m}$. In diesem Fall kann aus der gewählten Basis a keine Aussage getroffen werden.

6 Primzahl-Zertifikate

Neben dem Rabin-Miller-Test gibt es noch eine Reihe weiterer Primzahltests, die andere zahlentheoretische Zusammenhänge (etwa über quadratische Reste – der Solovay-Strassen-Test) verwenden. All diesen Tests haftet der Makel an, dass Primzahlen zwar mit hoher Wahrscheinlichkeit korrekt erkannt werden, aber nicht mit Sicherheit bekannt ist, ob es sich wirklich um Primzahlen handelt. Die genannten Algorithmen sind für praktische Belange, d.h. in Bereichen, in denen sie noch mit vertretbarem Zeitaufwand angewendet werden können, ausreichend und wurden auch in der Form in den verschiedenen CAS implementiert.

Aus dem Axiom-Handbuch:

> `prime?(n)` returns true if n is prime and false if not. The algorithm used is Rabin's probabilistic primality test. If `prime? n` returns false, n is proven composite. If `prime? n` returns true, `prime?` may be in error however, the probability of error is very low and is zero below $25 * 10^9$ (due to a result of Pomerance et al), below 10^{12} and 10^{13} due to results of Pinch, and below 341550071728321 due to a result of Jaeschke. Specifically, this implementation does at least 10 pseudo prime tests and so the probability of error is $< 4^{-10}$...

Aus dem Maple-Handbuch:

- The function `isprime` is a probabilistic primality testing routine.
- It returns false if n is shown to be composite within one strong pseudo-primality test and one Lucas test and returns true otherwise. If `isprime` returns true, n is „very probably prime - see [1], Band 2, Section 4.5.4, Algorithm P for a reference and [4]. No counter example is known and it has been conjectured that such a counter example must be hundreds of digits long.

Aus dem Mathematica-Handbuch:

- `PrimeQ` first tests for divisibility using small primes, then uses the Miller-Rabin strong pseudoprime test base 2 and base 3, and then uses a Lucas test.
- As of 1997, this procedure is known to be correct only for $n < 10^{16}$, and it is conceivable that for larger n it could claim a composite number to be prime.

Möchte man für gewisse Anwendungen sicher gehen, dass es sich bei der untersuchten Zahl garantiert um eine Primzahl handelt, benötigen wir ein *Zertifikat* für die Primzahleigenschaft.

Ein solches Zertifikat kann erstellt werden, wenn man zu der vermuteten Primzahl m ein Erzeugendes a der zyklischen Gruppe $\mathbb{Z}_m^*$ angibt (zusammen mit einem Beweis, dass dieses Element die behauptete Eigenschaft besitzt).

Satz 6 *Die ungerade Zahl m ist genau dann eine Primzahl, wenn $\mathbb{Z}_m^*$ eine zyklische Gruppe der Ordnung $m-1$ ist, d.h. wenn es ein Element $a \in \mathbb{Z}_m^*$ gibt, so dass $ord(a) = m-1$ gilt.*
Ein Element $a \in \mathbb{Z}_m^*$ hat genau dann die Ordnung $ord(a) = m-1$, wenn $a^{m-1} \equiv 1 \pmod{m}$, aber für alle Primteiler p der Zahl $m-1$ die Beziehung $a^{\frac{m-1}{p}} \not\equiv 1 \pmod{m}$ gilt.

Die letzte Bedingung sichert, dass $ord(a)$ durch $m-1$, aber keinen Teiler von $m-1$ teilbar ist.

Betrachten wir als Beispiel die Primzahl $m = 55499821019$. Für einen *Beweis* der Primzahleigenschaft prüfen wir die Voraussetzungen des Satzes für $a = 2$. Mit MuPAD erhalten wir

```
m:=55499821019;
Z:=Dom::IntegerMod(m);
u:=Factored::factors(factor(m-1));
```

$$[2, 17, 1447, 1128091]$$

```
map(u,p->Z(2)^((m-1)/p));
```

$$[55499821018, 44871471456, 31660914932, 44713683701]$$

```
Z(2)^(m-1);
```

$$1$$

Die Voraussetzungen des Satzes sind also erfüllt, so dass 2 die Gruppe $\mathbb{Z}_m^*$ erzeugt.

Oftmals ist allerdings für eine konkrete Restklasse a die Beziehung $a^{\frac{m-1}{p}} \not\equiv 1 \pmod{m}$ nur für einige der Primfaktoren von $m-1$ erfüllt und es ist schwierig, eine Restklasse zu finden, die für *alle* Primteiler passt.

```
m:=nextprime(2*10^10); // = 20000000089
Z:=Dom::IntegerMod(m);
u:=Factored::factors(factor(m-1));
```

$$[2, 3, 67, 1381979]$$

```
for c in [2,3,5,7,11,13,17,23,29] do
print(map(u,p->mods(expr(Z(c)^((m-1)/p)),m)))
end_for;
```

$$[1, 1, 3108157330, -2457953819]$$
$$[1, 1, -4865156001, -7521929036]$$
$$[1, 5697767833, 6300564838, -3927279952]$$
$$[-1, 1, 1, -855778974]$$
$$[1, 1, 8042780317, -7269258111]$$
$$[1, -5697767834, 8735956541, -5244188757]$$
$$[1, 1, -2293750453, 9516320400]$$
$$[1, 5697767833, -8450150690, -4573270973]$$
$$[-1, -5697767834, 8111724708, -1296001895]$$

Erst $a = 29$ hat die geforderte Eigenschaft, dass $a^{\frac{m-1}{p}} \neq 1$ für *alle* Primteiler p von m gilt. Es stellt sich, mit Blick auf den Chinesischen Restklassensatz nicht verwunderlich, heraus, dass es genügt, für jeden Primteiler *seine* Restklasse a zu finden, womit die Rechnungen in diesem Beispiel bereits für $a = 7$ beendet werden können.

Satz 7 *Seien $\{p_1, \ldots, p_k\}$ die (verschiedenen) Primfaktoren von $m-1$. Dann gilt*

$$\exists\, a \in \mathbb{Z}_m^*\ \forall i\ a^{\frac{m-1}{p_i}} \not\equiv 1 \pmod{m}$$

genau dann, wenn

$$\forall i\ \exists\, a_i \in \mathbb{Z}_m^*\ a_i^{\frac{m-1}{p_i}} \not\equiv 1 \pmod{m},$$

d.h. es gibt eine gemeinsame Basis für alle Primteiler von $m-1$, wenn es für jeden Primteiler einzeln eine passende Basis gibt.

Zur Bestimmung eines Primzahlzertifikats für die primzahlverdächtige Zahl m reicht es also aus, für jeden Teiler p der Zahl $m-1$ in einer Liste von kleinen Zahlen eine solche Zahl a_p zu finden, dass $a_p^{\frac{m-1}{p}} \not\equiv 1 \pmod{m}$ gilt:

```
certifyPrime:=proc(m) local u,v,p,a,D;
begin
  if not isprime(m) then error(expr2text(m)." ist nicht prim") end_if;
  D:=Dom::IntegerMod(m);
  u:=null(); v:=Factored::factors(factor(m-1));
  for p in v do
    for a in [2,3,5,7,11,13,17,19,23] do
       if not iszero(D(a)^((m-1)/p)-1) then u:=u,[p,a]; break end_if;
    end_for;
  end_for;
  if nops([u])<>nops(v) then
    error("Kein Zertifikat für ".expr2text(m)." gefunden");
  end_if;
  [u];
end_proc;
```

Auf obiges Beispiel angewendet erhalten wir damit folgendes Zertifikat:

```
certifyPrime(m);
```

$$[[2,7],[3,5],[67,2],[1381979,2]]$$

Der erste Eintrag gibt dabei jeweils den Primfaktor von $m-1$, der zweite die für diesen Primfaktor p geeignete Basis a_p an.

Dieses Kriterium geht also davon aus, dass eine Faktorzerlegung der Zahl $m-1$ berechnet werden kann. Für Zahlen in der Nachbarschaft einer Primzahl m ist das erstaunlicherweise oft möglich.

Besonders einfach ist ein solcher Nachweis für Primzahlen der Form $m = 2^a + 1$, da hier die Zahl $m-1$ nur durch 2 teilbar ist. Allerdings stellt es sich heraus, dass dies höchstens dann der Fall ist, wenn a selbst Zweierpotenz ist. Zahlen der Form $F_n := 2^{2^n} + 1$ bezeichnet man als *Fermatzahlen*, da sie erstmals von Fermat untersucht wurden, der in einem Brief an Mersenne die Behauptung aufstellte, dass alle Zahlen F_n prim sind. Er konnte dies für die ersten 5 Fermatzahlen $3, 5, 17, 257$ und 65537 nachweisen, vermerkte allerdings, dass er die Frage für die nächste Fermatzahl $F_5 = 4294967297$ nicht entscheiden könne. Dies wäre allerdings mit dem Fermat-Test für die Basis $a = 3$ (vom Rechenaufwand abgesehen) gar nicht so schwierig gewesen:

```
m:=2^(2^5)+1;
Z:=Dom::IntegerMod(m);
Z(3)^(m-1);
```

$$3029026160$$

Die Basis 3 kann man generell für den Fermattest von F_n verwenden und zeigen, dass auch die nächsten Fermatzahlen (deren Stellenzahl sich allerdings jeweils verdoppelt) zusammengesetzt sind.

Primzahlen dieser Gestalt spielen eine große Rolle in der Frage der Konstruierbarkeit regelmäßiger n-Ecke mit Zirkel und Lineal. So konnte Gauss die Konstruierbarkeit des regelmäßigen 17-Ecks nachweisen, weil die Primzahl 17 in dieser Reihe auftritt, vgl. etwa [2].

Die Faktorisierung $F_5 = 641 \cdot 6700417$ fand erstmals Euler, allerdings scheiterte er bereits an der nächsten Zahl $F_6 = 18446744073709551617$, deren Faktorisierung

$$F_6 = 67280421310721 \cdot 274177$$

erst im Jahre 1880 entdeckt wurde. Ein modernes CAS berechnet diese Zerlegung heute im Bruchteil einer Sekunde, kommt aber bei der nächsten Fermatzahl

$$F_7 = 340282366920938463463374607431768211457$$

bereits in Schwierigkeiten. Man geht heute davon aus, dass es außer den bereits gefundenen keine weiteren primen Fermatzahlen gibt. Für einen Beweis dieser Vermutung gibt es jedoch nicht einmal ansatzweise Ideen, vgl. [4].

Neben den Fermatzahlen spielen auch Primzahlen der Form $m = 2^a - 1$ eine wichtige Rolle. Die größten heute bekannten Primzahlen haben diese Gestalt. Da $m - 1$ keine so einfache Faktorisierung wie für Fermatzahlen besitzt, ist hier die Erstellung eines Primzahlzertifikats schwieriger, jedoch hat $m-1 = 2\,(2^{a-1}-1)$ wieder einen Faktor derselben Gestalt, was für eine teilweise Faktorisierung oft sehr hilfreich ist. Primzahlen der Form $M_p = 2^p - 1$ bezeichnet man als *Mersennesche Primzahlen*.

Moderne Primtestverfahren verwenden auch andere Gruppen als $\mathbb{Z}_m^*$ zum Test, insbesondere solche, die mit elliptischen Kurven verbunden sind. Allerdings kennt man bis heute noch keinen sicheren Primzahltest mit garantiert polynomialem (in $l(m)$) Laufzeitverhalten.

Literatur

[1] D.E. Knuth. *The art of computer programming.* Addison Wesley, 1991.

[2] E. Kunz. *Algebra.* Vieweg, 1991.

[3] Das Computeralgebra-System MuPAD, `http://www.mupad.de`.

[4] H. Riesel. *Prime numbers and computer methods for factorization.* Birkhäuser, 1994.

Attribution Section

graebe (2004-09-03): Contributed to KoSemNet
graebe (2005-02-02): Revision and translation to MuPAD