Primzahltests

Hintergrund

- Bei vielen Verschlüsselungs-Systemen (z.B. RSA):

  grosse Primzahlen werden benötigt

  $\uparrow$

  $\approx 10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000$

- Vorgehensweise, um grosse Primzahlen zu erzeugen:

  while (noch keine Primzahl gefunden) **do**
      Wähle (zufällig) eine grosse Zahl z
      Teste, ob z eine Primzahl ist
      Falls ja: **return** z
      Falls nein: Mache weiter
  end

  betrachtete Ansätze

  1. Probedivision, kleine Teiler (heute)
  2. probabilistische Primzahl-Tests (nächste Woche)
     - Fermat Test
     - Miller Rabin Test

Ansatz I: Probedivision

- Basis-Vorgehen:

  TESTPRIMZAHL(n)
      Für jede Primzahl $p < \sqrt{n}$
          Teste, ob p ein Teiler von n ist
          Falls ja: **return** "nicht prim"
          Falls nein: mache weiter
      return "prim"
  end

- Themen:
  1. Häufigkeit von Primzahlen
  2. Analyse des Algorithmus
  3. Eigenschaften von Zahlen bezüglich **kleinen** Primteilern

Häufigkeit von Primzahlen

Definition

Für $x \in \mathbb{N} : \pi(x) :=$ # Primzahlen, die $\leq$ x sind.

- **Aufgabe:** $\pi(10) = ?$, $\pi(17) = ?$

Satz (ohne Begründung)

$$\lim_{x \to \infty} \frac{\pi(x)}{\left(\frac{x}{\ln(x)}\right)} = 1$$

Folgerung

$$\pi(x) \approx \frac{x}{\ln(x)}$$

- **Aufgabe:** Geben Sie eine Abschätzung für die Wahrscheinlichkeit, dass eine zufällig gewählte, 100-stellige Zahl prim ist.

Analyse des Basis-Algorithmus

- # benötigte Probedivisionen = $\pi(\sqrt{n}) \underset{x:=\sqrt{n}}{\overset{\approx}{\uparrow}} \dfrac{\sqrt{n}}{\ln(\sqrt{n})}$
- **Bem:** Die zugehörige Laufzeit ist exponentiell (bez. Eingabelänge) $\Rightarrow$ für grosse n ist keine effiziente Ausführung möglich.
- Ansatz für ausgefeiltere Algorithmen:
  1. Test auf kleine Primteiler (bis zu einer vorgegebenen Schranke)
  2. komplexere Analysen

Kleine Primteiler

Ziel der Analyse: Abschätzen, was Tests auf kleine Primteiler bringen (Bzw. Wie viele Primzahl-Kandidaten können so ”ausgesiebt” werden?)

vorbereitende Überlegung:

- # (Zahlen zwischen 1 und 30, die nicht durch 2,3 oder 5 teilbar sind) $= \varphi(30)$
- Der Anteil aller Zahlen zwischen 1 und 30, die nicht durch 2,3 oder 5 teilbar sind, beträgt $\frac{\varphi(30)}{30}$.
- Die Wahrscheinlichkeit, dass eine zufällig gewählte Zahl nicht durch 2,3 oder 5 teilbar ist, beträgt $\frac{\varphi(30)}{30}$.

Verallgemeinerung auf die ersten r Primzahlen $p_1, p_2, \ldots, p_r$ ergibt:

Satz

Mit $P := p_1 \cdot p_2 \cdots p_r$ gilt:
Die Wahrscheinlichkeit, dass eine zufällig gewählte Zahl nicht durch eine der Zahlen $p_1, p_2, \ldots, p_r$ teilbar ist, beträgt $\frac{\varphi(P)}{P}$.

Kleine Primteiler

Folgerung: Die Wahrscheinlichkeit, dass eine zufällig gewählte Zahl durch eine der Zahlen $p_1, p_2, \ldots, p_r$ teilbar ist, beträgt

$$\begin{aligned} 1-\frac{\varphi(P)}{P} &= 1-\frac{(p_1-1)(p_2-1)\cdots(p_r-1)}{p_1\cdot p_2\cdots p_r} \\ &= 1-\left(1-\frac{1}{p_1}\right)\cdot\left(1-\frac{1}{p_2}\right)\cdots\left(1-\frac{1}{p_r}\right) \end{aligned}$$

Zahlenbeispiele

K	10	100	1000	10'000
Anteil mit Primteiler $\leq K$	77.14%	87.97%	91.90%	93.91%

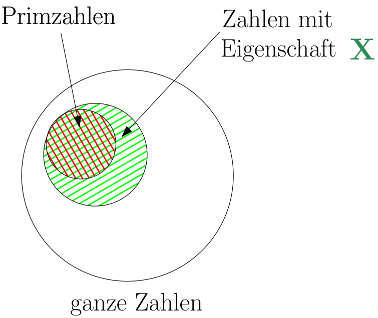
Zusatz-Bemerkung

- Wichtiges Resultat in der Algorithmik: deterministische Primzahltests in Polynomialzeit möglich!
  (Resultat von Lenstra und Pomerance, 2003)
- gefeiertes Ergebnis
- noch nicht gross Einzug in Praxis gefunden (gem. aktuellem Wissensstand)

Ausblick: probabilistische Primzahltests

Strategie, um zu testen, ob eine gegebene Zahl prim ist

- Tests mit einer Zufallskomponente:
  Grundidee: Teste auf eine Eigenschaft **X** (die **effizient** prüfbar ist)



- Einige Zahlen werden fälschlicherweise als Primzahlen klassifiziert!
- Aber: Wird eine Zahl als nicht-prim klassifiziert, dann stimmt es!