COMPUTERPRAKTIKUM POLYVALENZ

Primzahltests

Einführung. Das Testen natürlicher Zahlen auf Primaität ist spätestens, seitdem es kryptographische Verfahren gibt, welche große Primzahlenverwenden, keine rein theoretische Spielerei. Nachfolgend sind einige der grundlegenden Primzahlentests aufgeführt:

Satz 1 (Lucas)**.** Sei $n \in \mathbb{N}, n \geq 2$. Falls eine natürliche Zahl $1 < a < n$ existiert, so dass $a^{n-1} \equiv 1 \mod n$ und $a^{\frac{n-1}{p}} \not\equiv 1 \mod n$ für jeden Primteiler p von $n-1$ dann ist n eine Primzahl.

Die Zahlen der Form $F_k = 2^{2^k} + 1, k \in \mathbb{N}$ nennt man *Fermat-Zahlen*. Es ist eine offene Frage, ob es unendlich viele Fermat-Primzahlen gibt, d.h. Fermat-Zahlen, die prim sind. Für $0 \geq k \geq 4$ ist dies der Fall.

Satz 2 (Pépin)**.** F_k für $k \geq 1$ ist genau dann eine Primzahl, falls

$$3^{\frac{F_k-1}{2}} \equiv -1 \mod F_k.$$

Zahlen der Form $M_p = 2^p - 1$, wobei p prim ist, nennt man *Mersenne-Zahlen*.

Satz 3 (Lucas-Lehmer Test)**.** Seien $p > 2$ eine Primzahl und die Folge $(s_n)_{n\in\mathbb{N}}$ rekursiv definiert durch

$$s_1 := 4, \qquad s_{n+1} := s_n^2 - 2.$$

Dann ist M_p genau dann eine Primzahl, wenn M_p ein Teiler von s_{p-1} ist.

Satz 4 (1. Teil des Miller-Rabin Tests)**.** Sei n eine Primzahl, $a \in \mathbb{Z}$ und $\mathrm{ggT}(a, n) = 1$. Setze $n - 1 = 2^s d$ mit d ungerade. Dann gilt entweder

$$a^d \equiv 1 \mod n$$

oder

$$\exists r \in \{0, \ldots, s-1\} : a^{d2^r} \equiv -1 \mod n$$

Ist $n \in \mathbb{N}$ beliebig und gilt für eine Basis a, dass keine der beiden Fälle eintreten, so ist n keine Primzahl und man nennt a einen *Zeugen gegen die Primalität* von n.

Satz 5 (2. Teil des Miller-Rabin Tests)**.** Es sei $n \in \mathbb{N}$ mit $n \geq 3$ nicht prim. Dann gilt

$$\left| \left\{ a \in \{1, \ldots, n-1\} : \begin{array}{l} a \text{ ist kein Zeuge gegen die} \\ \text{Primalität von } n \text{ und } (a, n) = 1 \end{array} \right\} \right| \leq \frac{n-1}{4}.$$

Aufgabe 1. Implementieren Sie den Lucas-Primzahltest und den ersten Teil des Miller-Rabbin-Primzahltest als Funktionen in `GAP`. Implementieren Sie ebenso den Pépin-Test für Fermat'sche und den Lucas-Lehmer-Test für Mersenneprimzahlen.

Aufgabe 2. Wie viele potenzielle Fermat'sche und Mersenneprimzahlen lassen sich auf einer studentischen Rechner an unserer Uni in einer vernünftigen Zeit auf Primalität überprüfen? Vergleichen Sie die Laufzeiten ihrer Programme mit denen der `GAP`-Funktion `IsPrimeInt`.

Aufgabe 3. Wählen Sie einige sehr große Zahlen, diese dürfen ruhig mehrere hundert Stellen haben, und wenden Sie die von ihnen implementierten Lucas- und Miller-Rabbin-Tests auf diese Zahlen an. Der Miller-Rabbin-Test soll dabei eine Sicherheit von 90% haben. Vergleichen Sie die Laufzeiten ihrer Programme mit denen der `GAP`-Funktion `IsPrimeInt`.

Beschaffe Sie sich, auf eine Ihnen freigestellte Weise, einige sehr große Primzahlen und wiederholen Sie die Aufgabe mit diesen.

Aufgabe 4. Bestimmen Sie mit Hilfe des Miller-Rabbin-Tests mit einer Wahrscheinlichkeit von 90% alle Primzahlen zwischen 10^{50} und $10^{50} + 500$.

Reichen Sie eine Dokumentation Ihrer Ergebnisse ein. Falls Sie in den Aufgaben 2-4 ebenfalls Programme geschrieben haben, so reichen Sie dieses auch ein, ansonsten genügen log-files.