29. Rechnen mit großen Zahlen

(bzgl. fehlender Aufzeichnungen orginal aus Bodirksy-Skript übernommen)

Betrachte ein Computerprogramm der folgenden eingeschränkten Form: jeder Befehl ist von der Form $x := y \cdot z, \quad x := y - z, \quad x := y \cdot z$ oder $x := 1$. Wir nehmen an, dass jede Variable genau einmal auf der linken Seite einer solchen Zuweisung auftaucht, und dass sich alle weiteren Auftreten der Variablen später im Programm befinden. Daher sind zu jedem Zeitpunkt der Auswertung des Programmes die Variablen auf der rechten Seite ei- ner Zuweisung bereits ausgewertet. Sei x0 die Variable, die auf der linken Seite der letzten Zuweisung auftaucht. Wenn wir das Programm ausführen, wird dieser Variablen eine eindeutige ganze Zahl zugewiesen. Wir wollen gerne wissen, ob diese Zahl die Null ist. Das Problem, zu gegebenem Programm festzustellen, ob die letzte Zuweisung im Programm Null liefert, nennen wir Test-auf-Null.
Das Problem mit diesem Problem ist, dass die Werte der Variablen sehr großwerden können, so dass wir exponentiell viel Zeit benötigen, um diese Werte zu berechnen. Hierzu ein Beispiel.: $x_1 := 1$
$x_2 := x_1 + x_1$
$x_3 := x_2 \cdot x_2$
$x_4 := x_3 \cdot x_3$
$x_5 := x_4 \cdot x_4$
$... := ...$
$x_n := x_{n-1} \cdot x_{n-1}$
Hier wird x_2 der Wert 2 zugewiesen,x_3 der Wert $2^2 = 4$,x_4 der Wert $2^2 \cdot 2^2 = 2^4 = 16$,x_5 der Wert $2^4 \cdot 2^4 = 2^8$, und x_n der Wert $2^{2^{n-1}}$, wie man leicht mit vollständiger Induktion zeigt. Die Binärdarstellung dieser Zahl hat die Länge 2n 1: zu groß, um von einem polynomiellen Algorithmus ausgerechnet zu werden. Durch Anwendung von Subtraktion im Programm ist es jedoch nicht ausgeschlossen, dass die letzte Variable Null ist, auch wenn die Zwischenergebnisse sehr großsind.

Gibt es dennoch einen Algorithmus polynomieller Laufzeit, der das Problem Test-auf-Null löst? Dies ist ein wichtiges offenes Problem der theoretischen Informatik.
Proposition 20. Es gibt einen randomisierten Algorithmus mit garantiert polynomieller Laufzeit für das Problem Test-auf-Null. Falls die letzte Variable im Programm zu Null auswertet, so sagt das Programm garantiert ?Ja?. Ansonsten sagt das Programm mit Wahrscheinlichkeit von mindestens 2/3 ?Nein?.

Beweisidee. Um die Größenexplosion der Werte in den Variablen zu vermeiden, werden wir modulo einer großen zufälligen Zahl m rechnen.

Bitte mehr Details. Wir verwenden den in Abbildung 9 angegebenen Algorithmus. Falls die letzte Variable zu Null auswertet, dann auch modulo m. Wenn der Algorithmus also ?Nein? ausgibt, so ist die Antwort sicherlich korrekt. Falls die letzte Variable nicht zu Null auswertet, so kann man zeigen, dass der Algorithmus mit Wahrscheinlichkeit von

mindestens 2/3 akzeptiert. Die genaue Rechnung findet man in [6], Lemma 6-U; und verwendet tiefer liegende Resultate über die Verteilung der Primzahlen. □
Der genau Wert 2/3 in Proposition 20 ist hier von keiner besonderen Bedeutung: durch wiederholtes Anwenden des Algorithmus kann man diese Fehlerwahrscheinlichkeit sehr schnell verbessern.