SRT-Division:

Ergänzungen zu meiner Mitmachvorlesung am 1. Tag der Mathematik in Oldenburg am 8. November 2000

Wiland Schmale[1]

Die Abkürzung "SRT-Division" setzt sich zusammen aus den Anfangsbuchstaben von Sweeney (IBM), Robinson (Univ. Illinois) und Tocher (Imperial College) (laut Pratt [Pra] p. 2.), die unabhängig voneinander und bereits vor 1960 Divisionen dieses Typs betrachtet haben.

Die SRT-Division gehört zur Klasse der so genannten "digit-recurence"- Verfahren. Sie wird unter anderem in Kryptochips benutzt. Am Bekanntesten ist wohl ihre Anwendung in Pentium-Prozessoren geworden und zwar wegen des inzwischen schon legendären "Pentium Bug".

1994 wurden in großer Stückzahl Rechner mit fehlerhafter Division ausgeliefert. Da der Fehler in seltenen Konstellationen und bei hohen Genauigkeitsanforderungen auftrat wurde er von normalen Nutzern nicht bemerkt. Laut [Pra] wurde der Fehler innerhalb Intel bemerkt aber unabhängig davon von Nicely (Primzalzwilling-Forschung), der ihn im Oktober 1994 publizierte. Eine Witze-Sammlung darüber erwähnt [Ed] als [3]. Die Kosten des Fehlers werden von Intel auf 464 Millionen $ geschätzt.

Dieser Text versucht, die SRT-Division verständlich aber genau einzuführen. Dies ist m.E. besser möglich, wenn man sie gleich in etwas verallgemeinerter Form behandelt. Der Nachweis, dass die SRT-Division "funktioniert" wird vollständig geführt. Eine sehr schöne Visualisierungsmöglichkeit bietet das Pratt'sche Robotermodell, mit dem diese kurze Einführung schliesst. Um den "Pentium Bug" vollständig nachzuvollziehen, muss noch mehr ins Detail gegangen werden. Für den ganzen Text sind verallgemeinerte g-adische Zifferndarstellungen so, wie sie in der Mitmachvorlesung eingeführt wurden grundlegend und zwar insbesondere für $g = 4$. Die Mitmachvorlesung und dieser Text beruhen auf den beiden am Ende angegebenen Arbeiten der Autoren Alan Edelman und Vaughan Pratt.

Die SRT-Division ist ein Divisionverfahren, das in erster Linie für symmetrische und redundante Ziffernmengen konzipiert wurde.

Im Folgenden sind stets gegeben:

- **Divident p, Divisor d,** o. E. $p, d \in \mathbb{N}_+$
- **Basis g**, bezüglich der die Division durchgeführt wird
- $\mathbb{S}_{2a} := \{-a, \ldots, 0, \ldots, a\}$ mit $a \in \mathbb{N}_+$ ist die bei der Division zugelassene Ziffernmenge. **Es gelte $2a + 1 \geq g$.**

[1]Carl von Ossietzky Universität Oldenburg, Fachbereich Mathematik, Ammerländer Heerstraße 114-118, 26111 Oldenburg, Tel.: 0441/798-3238, E-Mail: wiland.schmale@uni-oldenburg.de

Wir lassen offen, bezüglich welcher Basis p, d dargestellt sind.
Aufgabe soll sein: $q := \frac{p}{d}$ darzustellen als

$$q = \sum_{k=0}^{N} q_k g^{-k} + \text{ Fehler} \qquad q_k \in \mathbb{S}_{2a}$$

wobei N die Anzahl der Iterationen und den Fehler bestimmt.

Am häufigsten wird die SRT-Division mit folgenden Parametern angewendet:

$$g = 4,\ \mathbb{S}_4 = \{-2, 1, 0, 1, 2\}$$

$\mathbb{S}_4$ besteht nur aus 2-er Potenzen**!** mit Vorzeichen. Dies und die Redundanz sind die Gründe für die höhere Geschwindigkeit der SRT-Division ([Ed] p. 55.)

Typischerweise besteht die SRT-Division aus zwei Komponenten: dem Divisionsalgorithmus und der Ziffernauswahlfunktion, die dem Algorithmus in jedem Iterationsschritt sagt, welche Ziffer aus $\{-a, \ldots, a\}$ gewählt werden kann. Im folgenden Algorithmus wird die Ziffernauswahl offen gelassen. Eine einfache Strategie wäre, jeweils die kleinste mögliche Ziffer zu wählen.

Ein SRT-Algorithmus, Verallgemeinerung des in [Ed] p. 55 angegebenen Algorithmus für beliebige g, a, p, d:

Eingabe: g, a, p, d und die Schrittanzahl N.
Die folgende Bedingung sei erfüllt:

$$p \leq \frac{agd}{g-1} \tag{1}$$

Wenn (1) nicht gilt, gibt es eine Potenz g^r, so dass (1) mit dg^r statt d gilt ("Shift").

$p_0 := p$
für $k = 0, 1, \ldots, N$:
Bestimme in Abhängigkeit von p_k und d eine Ziffer $q_k \in \mathbb{S}_{2a}$ derart, dass mit

$$p_{k+1} := g(p_k - q_k d)$$

gilt:

$$|p_{k+1}| \leq \frac{adg}{g-1} \tag{2}$$

Ausgabe: Näherungswert für $\frac{p}{d}$: $\sum_{k=0}^{N} q_k\, g^{-k}$

Zur Motivation der Beziehungen (1) und (2) im Algorithmus:
O.E. (sonst "Shift" mit g-Potenz) will man haben:

$$\frac{p}{d} \leq a + ag^{-1} + ag^{-2} + \ \cdots \ = (a, aaa\cdots)$$

Daraus folgt:

$$\frac{p}{d} \leq a \sum_{k\geq 0} g^{-k} = \frac{a}{1-g^{-1}} = \frac{ag}{(g-1)}$$

bzw.

$$p \leq \frac{adg}{(g-1)}$$

Ist nun etwa

$$\frac{p}{d} = q_0 + q_1 g^{-1} + \cdots = q_0 + g^{-1} \overbrace{(q_1 + g^{-1}(q_2 + \cdots))}^{=: \frac{p_1}{d}}$$

dann muss gelten:

$$\frac{p}{d} - q_0 = g^{-1} \cdot \frac{p_1}{d} \quad \text{oder} \quad p_1 = g(p - q_0 d)$$

Letzteres motiviert (2).

Dass dieses Verfahren funktioniert, besagen die beiden folgenden Behauptungen:

Sei $a \geq \frac{g-1}{2}$.

(a) Wenn $|p_k| \leq \frac{adg}{g-1}$ (vgl.(1)), dann gibt es $q_k \in \mathbb{S}_{2a}$ derart, dass (2) gilt.

(b)

$$\text{Es gilt: } \frac{p}{d} - \sum_{k=0}^{N} q_k g^{-k} = \frac{p_{N+1}}{d} g^{-(N+1)} \tag{3}$$

$$\text{und } \frac{p_{N+1}}{d} = \left(\sum_{k=N+1}^{\infty} q_k g^{-k} \right) g^{N+1} \tag{4}$$

$$\text{und } \left| \frac{p}{d} - \sum_{k=0}^{N} q_k g^{-k} \right| \leq \frac{a}{g-1} g^{-N} \tag{5}$$

$$\text{und } \lim_{N\to\infty} \sum_{k=0}^{N} q_k g^{-k} = \frac{p}{d}$$

Bemerkung

Wenn die Behauptungen (a) und (b) erst einmal bewiesen sind, so besagen sie nicht nur, dass die SRT-Division sinnvoll ist, sondern auch noch, dass $\frac{p}{d}$ eine Darstellung $\frac{p}{d} = \sum_{i\geq 0}^{\infty} q_i g^{-i}$ tatsächlich auch besitzt, dass $\frac{p_{N+1}}{d} g^{-(N+1)}$ der Fehler nach $N+1$ Schritten ist und dass wegen (4) zwangsläufig gelten muss

$$|p_{N+1}| \leq \frac{adg}{g-1}$$

was diese Bedingung im Algorithmus nachträglich noch einmal erklärt.

Beweis der Behauptungen (a) und (b):

(a) Wir versuchen p_k mit Hilfe von $q_k d$ unter die Schranke $\frac{ad}{g-1}$ zu bekommen, denn nur dann ist

$$|p_{k+1}| = |g(p_k - q_k d)| \leq \frac{adg}{g-1}.$$

Die Länge $\frac{2ad}{g-1}$ des "Ziel"-Intervalls

$$I = \left[-\frac{ad}{g-1}, \frac{ad}{g-1}\right]$$

ist $\geq d$, denn es wurde $a \geq \frac{g-1}{2}$ vorausgesetzt. Da außerdem

$$-\frac{ad}{g-1} = -\frac{adg}{g-1} + ad \leq p_k + ad$$

und entsprechend

$$p_k - ad \leq \frac{adg}{g-1} - ad = \frac{ad}{g-1}$$

folgt insgesamt:

Es gibt ein $q_k \in \mathbb{S}_{2a}$ mit $|p_k - q_k d| \leq \frac{ad}{g-1}$ und mit $|p_{k+1}| \leq \frac{adg}{g-1}$.

(b) Es ist $p - (q_o + q_1 g^{-1} + \cdots + q_N g^{-N}) \cdot d =$
$(\cdots(($$\underbrace{\underbrace{\underbrace{(p - q_0 d)g}_{=p_1} - q_1 d)g}_{=p_2} - q_2 d)g}_{=p_3} - \cdots - q_N d)g \cdot g^{-(N+1)} = p_{N+1} g^{-(N+1)}$
$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{=p_{N+1}}$

Damit folgt (3) und (4), aber auch (5):

$$\left|\frac{p}{d} - \sum_{k=0}^{N} q_k g^{-k}\right| = \left|\frac{p_{N+1}}{d} g^{-(N+1)}\right| \leq \frac{ag}{g-1} g^{-(N+1)} = \frac{a}{g-1}\, g^{-N}$$

Die behauptete Konvergenz ergibt sich unmittelbar aus (5). □

Natürlich gibt es im Algorithmus immer nur endlich viele Möglichkeiten für die Wahl von q_k. Da die Schrittzahl N vorgegeben wird, erhält man unabhängig von der Wahl von q_k einen Näherungswert

$$q_0 + \cdots + q_N g^{-1} \quad \text{für} \quad \frac{p}{d} \quad \text{dessen Fehler} \quad < \frac{adg^{-(N-1)}}{g-1} \quad \text{ist .}$$

Es bleibt das Problem der jeweiligen Auswahl von q_k, der so genannten **Ziffernauswahlfunktion**.

Ein einfaches Beispiel:

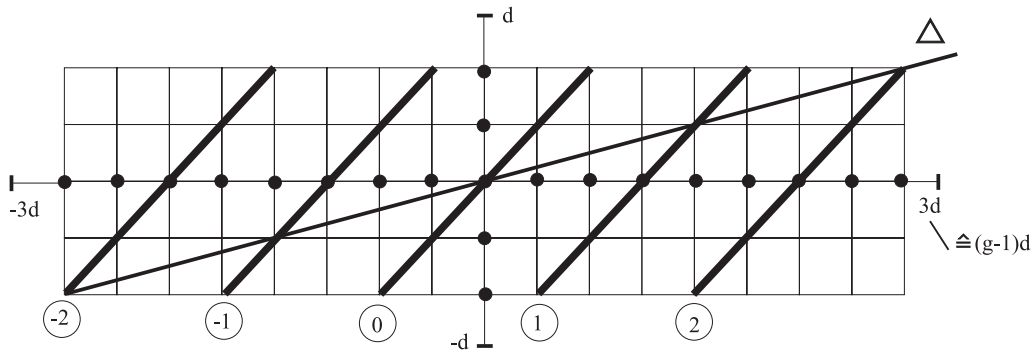
Sei $Z_k = \{q \in \mathbb{S}_{2a} : |p_k - qd| \leq \frac{ad}{g-1}\}$ die beim k-ten Schritt zur Auswahl stehende und stets endliche(!) Ziffernmenge. Eine mögliche Ziffernauswahl ist dann z.B.: $\min Z_k$

In der Praxis werden aber völlig andere und viel kompliziertere Auswahlfunktionen benutzt. Sie sind motiviert durch den Wunsch nach einfacher und schneller Hardware-Realisierung und werden z. B. beim Pentium-Chip in Form einer Hardware-Tabelle, einer sogenannten "look up table" realisiert. Dabei ist dann meistens $g = 4$. Es werden neuerdings aber auch höhere 2-er Potenzen benutzt. Die "look-up-table" ist normalerweise Firmengeheimnis. Der Eingangs erwähnte "Pentium-bug" entstand durch eine fehlerhafte "look-up table". Nur mit Hilfe dieses Pentium-"bug" konnte von Mathematikern ein Beispiel einer solchen geheimgehaltenen Tabelle rekonstruiert und publiziert werden (viele mathematische Details bei [Ed]). Selbst wenn man die fehlerhafte "look-up table" kennt, ist es nicht leicht herauszufinden ob und mit welchen p, d ein Fehler tatsächlich entsteht. Siehe auch hierzu: [Ed].

Vorteile der SRT-Division kurz gefasst für $g = 4$, $a = 2$:

- symmetrische Ziffernmenge
- alle Ziffern sind 2-er Potenzen (ab $a = 3$ nicht mehr)
- Redundanz ist sogar ein Vorteil auf Grund einer besonderen Addition, die in Computern benutzt wird. (Carry-save Addition, siehe [Ed])

Eine reizvolle Veranschaulichung der SRT-Divsision gibt [Pra] für $g = 4$, $a = 2$:



Ein kleiner Roboter startet im Punkt $(p,\ \frac{1}{4}p) =: P$ auf der Diagonalen $\triangle$. Damit P im Rechteck liegt, muss (1) gelten. Er wandert zuerst in Nord/Süd-Richtung, damit er auf eine der 5 Strecken $\textcircled{-2}, \cdots, \textcircled{2}$ trifft. Je nachdem, wo P_0 liegt, gibt es Wahlmöglichkeiten für den Roboter.

Einmal auf der Strecke $\textcircled{q}$, $q \in \mathbb{S}_4$, angekommen, merkt er sich q und wandert horizontal und zwar so, dass er im Punkt $P_1 = (p_1,\ \frac{1}{4}p_1)$ auf $\triangle$ trifft (eindeutig!). Für $p_o := p,\ p_1,\ q_0 := q$ ist die Bedingung (2) mit $k = 0$ erfüllt !! P_1 ist nun der neue Startpunkt für den Roboter und er verhält sich in P_1 genauso wie vorher im Punkt P. Ausgehend von P_1 erreicht er einen neuen Punkt P_2 auf $\triangle$, usf. Die nacheinander angesteuerten Graden liefern die Ziffern $q_0,\ q_1,\ q_2,\ \cdots, q_N$ für den Näherungswert $\sum_{k=0}^{N} q_k\ g^{-k}$ für $\frac{p}{d}$.

Literatur

[Pra] Vaughan Pratt: The anatomy of the Pentium bug LNCS 915 (1995) p 97-107

[Ed] Alan Edelmann: The mathematics of the Pentium divison bug, SIAM Review **39** 54-67 (1997)