Algorithmische Methoden 1

Wolfgang Windsteiger

RISC Institut

Das Thema der heutigen Vorlesung

Arithmetische Grundoperationen

Arithmetische Grundoperationen

- Algorithmen basieren auf Zifferndarstellung der Operanden.
- Algorithmen sind aus der Schule bekannt.
- Addition:

$$\begin{array}{cccccc} & 9 & 7 & 3 & 1 & \\ + & 0 & 8 & 2 & 9 & \\ \hline & 1 & 1 & 0 & 1 & 0 \quad \text{Übertrag} \\ \hline 1 & 0 & 5 & 6 & 0 & \\ & \leftarrow & \leftarrow & \leftarrow & \leftarrow & \end{array}$$

- Multiplikation:

$$\begin{array}{cccccccc} & & a_1 & a_2 & a_3 & a_4 & & b_1 \; b_2 \; b_3 \\ & & \downarrow & \downarrow & \downarrow & \downarrow & & \downarrow \; \downarrow \; \downarrow \\ & & 9 & 7 & 3 & 1 & \cdot & 8 \; 2 \; 9 \\ \hline & & 8 & 7 & 5 & 7 & 9 & \quad a \cdot b_3 \\ + & 1 & 9 & 4 & 6 & 2 & 0 & \quad a \cdot b_2 \cdot 10^1 \\ + & 7 & 7 & 8 & 4 & 8 & 0 & 0 \quad a \cdot b_1 \cdot 10^2 \\ \hline & 8 & 0 & 6 & 6 & 9 & 9 & 9 \end{array}$$

Analyse der klassischen Algorithmen

- Elementaroperationen bei der Addition: Addition zweier Ziffern mit dem Übertrag aus der vorhergehenden Stelle resultierend in einer Resultatsziffer und einem Übertrag.
- Ablauf Addition: obige Elementaroperation wird wiederholt ausgeführt, abhängig von Stellenzahl der Operanden.
- Elementaroperationen bei der Multiplikation: Multiplikation zweier Ziffern resultiernd in einer Resultatsziffer und einem Übertrag, Addition wie oben.
- Ablauf Multiplikation: obige Elementaroperation wird wiederholt ausgeführt, abhängig von Stellenzahl der Operanden.

Natürliche und ganze Zahlen fixer Länge

- $B = 2 \rightsquigarrow z_i \in \{0, 1\}$.
- $L \rightsquigarrow \omega$, ω fix (von physikalischen Parametern des Prozessors abhängig).
- $$a = \sum_{i=1}^{\omega} z_i 2^{\omega - i}.$$
- $z_1 = 0$ erlaubt.
- Negative Zahlen $\rightsquigarrow$ 1 Bit für Vorzeichen und $\omega - 1$ Bits für Betrag der Zahl (Vorzeichen-Betrag-Darstellung).
- Da ω fix ist, können die Algorithmen zur Addition und Multiplikation in elektronischen Bauteilen realisiert werden (Hintereinanderschaltung von Elementaroperationen).
- Multiplikation im Fall von $B = 2$ besonders einfach.

Kenngrößen

	Vorzeichenlose Darstellung	
ω	$N_{\min}$	$N_{\max}$
16	0	$2^{16} - 1 = 65535$
32	0	$2^{32} - 1 \approx 4.3 \cdot 10^9$
64	0	$2^{64} - 1 \approx 1.8 \cdot 10^{19}$

	Vorzeichen-Betrag-Darstellung	
ω	$Z_{\min}$	$Z_{\max}$
16	$-2^{15} + 1 = -32767$	$2^{15} - 1 = 32767$
32	$-2^{31} + 1 \approx -2.1 \cdot 10^9$	$2^{31} - 1 \approx 2.1 \cdot 10^9$
64	$-2^{63} + 1 \approx -9 \cdot 10^{18}$	$2^{63} - 1 \approx 9 \cdot 10^{18}$

Kenngrößen

Vorzeichenlose Darstellung		Vorzeichen-Betrag-Darstellung	
Ziffernfolge $z_1 \dots z_{16}$	$a \in \mathbb{N}_0$	Ziffernfolge $z_1 \dots z_{16}$	$a \in \mathbb{Z}$
0000000000000000	0	0000000000000000	0
0000000000000001	1	0000000000000001	1
0000000000000010	2	0000000000000010	2
0000000000000011	3	0000000000000011	3
$\vdots$	$\vdots$	$\vdots$	$\vdots$
0111111111111111	32767	0111111111111111	32767
1000000000000000	32768	1000000000000000	-0
1000000000000001	32769	1000000000000001	-1
$\vdots$	$\vdots$	$\vdots$	$\vdots$
1111111111111111	65535	1111111111111111	-32767

Natürliche und ganze Zahlen beliebiger Länge

- Idee: Bei zugrundeliegender Wortlänge ω wähle $B = 2^{\omega-1}$.
- Ziffern $\rightsquigarrow$ $(\omega - 1)$-Bit-Zahlen, die in den oben erwähnten Standard-Datenstrukturen repräsentiert werden können.
- Beliebig lange Zahlen $\rightsquigarrow$ beliebig viele Ziffern.
- Datenstruktur Tupel zum Speichern der Ziffern $\rightsquigarrow \mathcal{Z}_B$.

Natürliche und ganze Zahlen beliebiger Länge

- $() \rightsquigarrow 0$, andernfalls ist die von $z = \left(z_1, \ldots, z_{L(z)}\right) \in \mathcal{Z}_B$ beschriebene ganze Zahl $a \in \mathbb{Z} \setminus \{0\}$:

$$a = \begin{cases} \sum_{i=1}^{L(z)} z_i B^{L(z)-i} & \text{falls } z_1 > 0 \\ -(|z_1| + \sum_{i=2}^{L(z)} z_i B^{L(z)-i}) & \text{falls } z_1 < 0 \end{cases}$$

- Für ein beliebiges Tupel z ist die kanonische Form in $\mathcal{Z}_B$:

$$\text{kanonisch}_{\mathcal{Z}_B}(z) := \begin{cases} \text{kanonisch}_{\mathcal{Z}_B}(z_{2:n}) & \text{falls } z \neq () \text{ und } z_1 = 0 \\ z & \text{sonst} \end{cases}$$

Beispiel

Beispiel

$(54321, 12345, 67890) \in \mathcal{Z}_{2^{31}}$ *stellt die Zahl*

$$a = 54321 \cdot 2^{62} + 12345 \cdot 2^{31} + 67890 = 250511396233504824035634 \in \mathbb{Z}$$

dar.
$(-54321, 12345, 67890)$ *steht dann für* $-a$.

Arithmetische Operationen

- Algorithmen wie aus der Schule bekannt.
- Elementaroperationen auf Zahlen fixer Länge ω ausführbar *leadsto* Reserve-Bit für Übertrag bzw. Vorzeichen.
- Durch variable Länge: Schleifen.

Exemplarisch: Multiplikation

Algorithmus *MultZ*: Klassischer Multiplikationsalgorithmus in $\mathbb{Z}$

$z \leftarrow (0 \mid k = 1, \ldots, L(a) + L(b))$
for i **from** $L(b)$ **to** 1 **by** -1
  $c \leftarrow 0$
  for j **from** $L(a)$ **to** 1 **by** -1
    $p \leftarrow a_j \cdot b_i$
    $q \leftarrow p + z_{i+j} + c$
    $(c, z_{i+j}) \leftarrow (q \operatorname{div} B, q \bmod B)$
  $z_i \leftarrow c$
$z \leftarrow \text{kanonisch}_{\mathcal{Z}_B}(z)$
return z

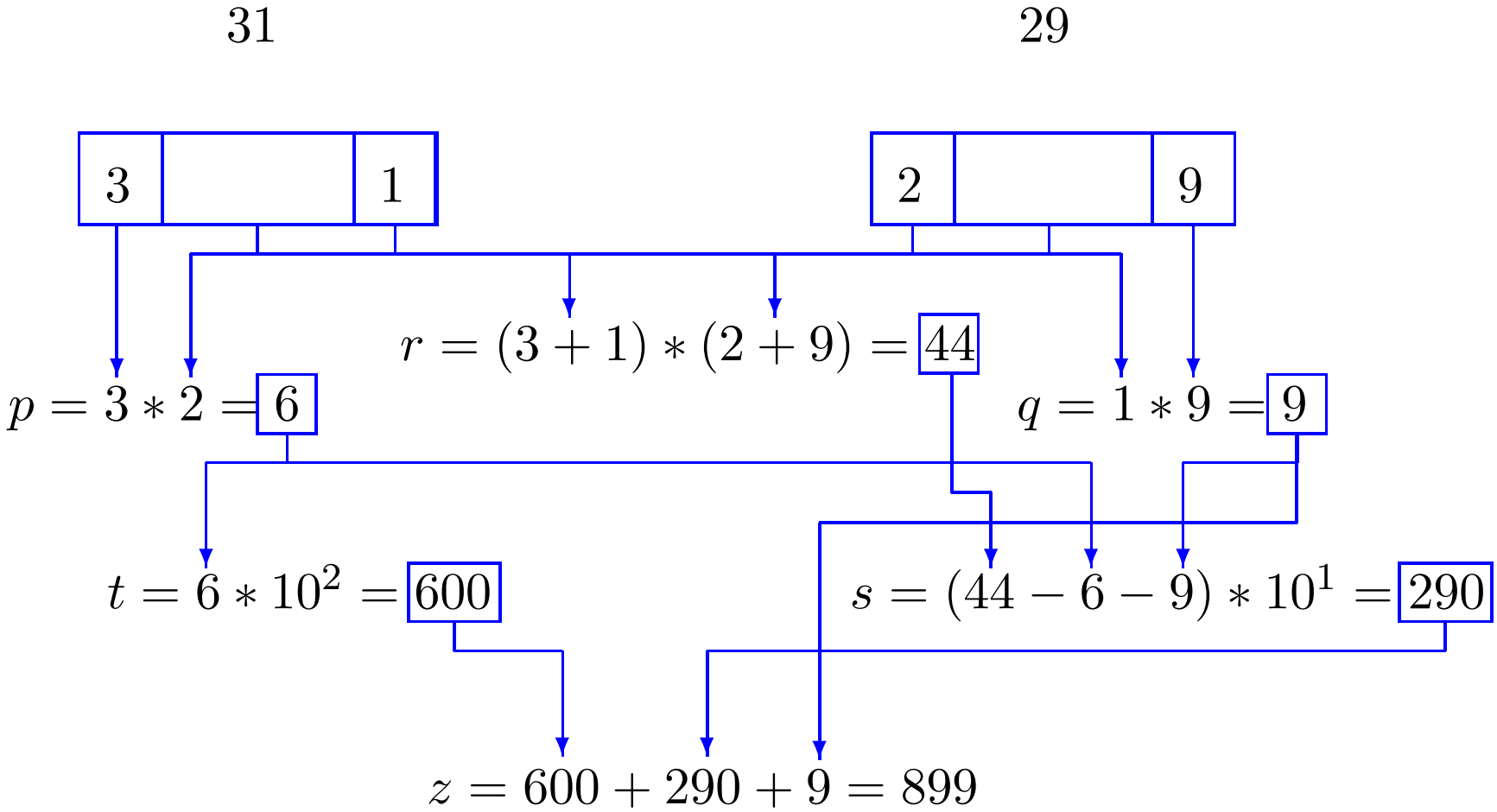
Aufruf: $MultZ(a, b)$
Eingabe: $a, b \in \mathcal{Z}_B$
mit: $a_1, b_1 > 0$.
Ausgabe: $z \in \mathcal{Z}_B$
mit: $\sum_{k=1}^{L(z)} z_k B^{L(z)-k} = \sum_{j=1}^{L(a)} a_j B^{L(a)-j} . \sum_{i=1}^{L(b)} b_i B^{L(b)-i}$.

Polymorphismus!

Schnelles Multiplizieren

Beispiel (Kleine Zahlen)

Berechne $31 \cdot 29$. *Resultat ist* 899.



Schnelles Multiplizieren: Allgemeines Prinzip

Betrachte a mit $L(a) = 2n$. Die durch a beschriebene Zahl α kann dann gemäß

$$\alpha = \sum_{i=1}^{2n} a_i B^{2n-i} = B^n \cdot \sum_{i=1}^{n} a_i B^{n-i} + \sum_{i=n+1}^{2n} a_i B^{2n-i} = B^n \bar{\alpha} + \tilde{\alpha}$$

geschrieben werden, wobei die Zahlen $\bar{\alpha}$ und $\tilde{\alpha}$ durch die Tupel

$$\bar{a} := a_{1:n} \quad \text{bzw.} \quad \tilde{a} := a_{n+1:2n}$$

dargestellt sind.
Analog dazu ist β durch $\bar{b} := b_{1:n}$ bzw. $\tilde{b} := b_{n+1:2n}$ dargestellt.

Schnelles Multiplizieren: Allgemeines Prinzip

Die Berechnung des Produkts $\alpha \cdot \beta$ kann wegen

$$\begin{aligned} \alpha \cdot \beta &= (B^n \bar{\alpha} + \tilde{\alpha}) \cdot (B^n \bar{\beta} + \tilde{\beta}) = \\ &= B^{2n} \bar{\alpha}\bar{\beta} + B^n((\bar{\alpha} + \tilde{\alpha})(\bar{\beta} + \tilde{\beta}) - \bar{\alpha}\bar{\beta} - \tilde{\alpha}\tilde{\beta}) + \tilde{\alpha}\tilde{\beta} \end{aligned}$$

auf die Berechnung dreier Produkte kürzerer Zahlen zurückgeführt werden, nämlich

$$\begin{aligned} p &= \bar{\alpha} \cdot \bar{\beta} \\ q &= \tilde{\alpha} \cdot \tilde{\beta} \\ r &= (\bar{\alpha} + \tilde{\alpha})(\bar{\beta} + \tilde{\beta}). \end{aligned}$$

Karatsuba Algorithmus

Spezifikation wie *MultZ*

$n \leftarrow (\max(L(a), L(b)) + 1) \operatorname{div} 2$
if $n = 1$
  $z \leftarrow a \cdot b$
else
  $a \leftarrow \mathit{Fülle}_{\mathcal{Z}_B}(a, 2n), b \leftarrow \mathit{Fülle}_{\mathcal{Z}_B}(b, 2n)$
  $\bar{a} \leftarrow \text{kanonisch}_{\mathcal{Z}_B}(a_{1:n})$, $\tilde{a} \leftarrow \text{kanonisch}_{\mathcal{Z}_B}(a_{n+1:2n})$
  $\bar{b} \leftarrow \text{kanonisch}_{\mathcal{Z}_B}(b_{1:n})$, $\tilde{b} \leftarrow \text{kanonisch}_{\mathcal{Z}_B}(b_{n+1:2n})$
  $p \leftarrow \mathit{MultZKaratsuba}(\bar{a}, \bar{b})$
  $q \leftarrow \mathit{MultZKaratsuba}(\tilde{a}, \tilde{b})$
  $r \leftarrow \mathit{MultZKaratsuba}(\bar{a} + \tilde{a}, \bar{b} + \tilde{b})$
  $s \leftarrow \mathit{Verschiebe}_{\mathcal{Z}_B}(r - p - q, n)$, $t \leftarrow \mathit{Verschiebe}_{\mathcal{Z}_B}(p, 2n)$
  $z \leftarrow t + s + q$
return z

Was bringt das?

Einschub: Komplexität von Algorithmen

- „Aufwand“ von Algorithmen:
  Zeitkomplexität: Beurteilung des Rechenzeitaufwands.
  Raumkomplexität: Einschätzung des Speicherplatzbedarfs.
- Betrachtung des Aufwands in Abhängigkeit von der „Größe“ des Inputs $\rightsquigarrow$ Größenkategorien für Eingaben.
- In jeder Kategorie kann der Aufwand schwanken $\rightsquigarrow$ best case, average case, worst case.
- Aussagen über Größenordnung (ähnlich wie bei Kondition).

Landau-Notation

Bezeichnung (Landau-Notation für reell-wertige Funktionen auf $\mathbb{N}$)

Sei $f : \mathbb{N} \to \mathbb{R}$. Der Ausdruck $O(f(n))$ beschreibt eine Funktion $g : \mathbb{N} \to \mathbb{R}$, für die Konstanten $C \in \mathbb{R}^+$ und $N \in \mathbb{N}$ existieren, sodass

$$|g(n)| \leq C \cdot |f(n)| \quad \text{für } n \geq N. \tag{27}$$

Für (27) ist auch die Schreibweise $g(n) = O(f(n))$ üblich, und man sagt, $g(n)$ hat die Ordnung $f(n)$.

Intuitiv:

- $O(f(n))$ ist eine Funktion, die für große Inputwerte nicht wesentlich größere Funktionswerte als $f(n)$ annimmt.
- $O(f(n))$ ist eine Größe, die nicht schneller wächst als $f(n)$.

Komplexität Multiplikation

- $M(n)$ ... Aufwand für Multiplikation $\alpha \cdot \beta$ mit $L(\alpha) = L(\beta) = n$.
- Klassischer Algorithmus *MultZ*: Zeitkomplexität $M(n) = O(n^2)$.

$$M(2n) \approx (2n)^2 = 4n^2 \approx 4M(n)$$

- Karatsuba Algorithmus *MultZKaratsuba*:

$$\alpha \cdot \beta \rightsquigarrow \underbrace{\bar{\alpha} \cdot \bar{\beta}}_{(1)}, \underbrace{\tilde{\alpha} \cdot \tilde{\beta}}_{(2)} \text{ und } \underbrace{(\bar{\alpha} + \tilde{\alpha})(\bar{\beta} + \tilde{\beta})}_{(3)}$$

(1) und (2) sind mit Aufwand $M(n)$ ausführbar
(3) mit $M(n) + O(n)$ und die restlichen Operationen: $O(n)$

$$M(2n) \leq 3M(n) + C \cdot n \tag{28}$$

Zeitkomplexität Karatsuba Algorithmus $O(n^{\log_2(3)}) \approx O(n^{1.585})$