

Seminar
Computeralgebra

WS 2007/2008

Prof. Dr. Sebastian Iwanowski

Langzahlarithmetik
Addition und Multiplikation

Jörg Fitzner (minf2913)

28.11.2007

1. Motivation

Bedarf an Zahlen in der Computeralgebra...?

- Formeln enthalten Zahlen
 - Faktoren, Potenzen, etc.
 - zusätzlich zu den Symbolen
- Umformungen müssen mit diesen arbeiten
 - exaktes Rechnen („verlustfrei“)
 - Erkennung quantitativer Gleichheit

→ *Notwendigkeit einer Zahlenarithmetik*

Anforderungen an die Zahlearithmetik

- exakte Darstellung (keine Näherungen)
- beliebig große/kleine Zahlen

→ „Langzahlen“

- beliebig lange ganze Zahlen (Zahlenraum \mathbb{Z})
- Grundlage der Darstellung weiterer Zahlenräume
 - Rationale Zahlen (\mathbb{Q})
 - bestimmte irrationale Reelle Zahlen ($\subset \mathbb{R}$)
 - Komplexe Zahlen (\mathbb{C}), Quaternionen, etc.

2. Zahlssysteme und Darstellungen

Zahlssysteme

- Arabisches Positionssystem

Beispiele:

„1999“

„440“

- Ziffernfolge aus Ziffern 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Wertigkeit jeder Stelle abhängig von Position in Ziffernfolge (Multiplikationsfaktoren sind Potenzen zur Basis 10, aufsteigend von Rechts nach Links)
- Zahl ist Summe aller Stellenprodukte

- Römisches Konstellationssystem

Beispiele:

„MCMXCIX“

„CDXL“

- Symbolfolge aus Symbolen I, V, X, L, C, D, M
- Symbole links von höherwertigen Symbolen werden subtrahiert, alle anderen Symbole werden addiert
- normierte Reihenfolge der Teilsequenzen, sowie Einschränkungen
 - *komplexere Rechenalgorithmen*

- Strichliste, ...

Darstellungsformen

- kanonische direkte Darstellung

- Linearkombination von Potenzen einer bestimmten Basis $B \in \mathbb{N}_{\geq 2}$
- Ziffern der Ziffernfolge sind **Koeffizienten der Linearkombination**
- Ziffern sind Symbole für Zahlen von *Null* bis $B-1$
- Länge einer Zahl z : $\#z$ = Anzahl der Ziffern in z

- $$z = \sum_{k=0}^{\#z-1} z_k B^k \quad \text{mit } z_k \in [0, 1, \dots, B-1]$$

- **Beispiele**

- $B=10 \rightarrow$ arabisches Zahlssystem (Dezimalzahlen) „440₁₀“
- $B=2 \rightarrow$ Binärsystem (Dualzahlen) „110111000₂“

→ *Darstellungsform ist eindeutig (ohne führende Nullen) und existent*

Darstellungsformen

- indirekte Darstellungen
 - **ganzzahliger Quotient mit modalem Rest** (DIV und MOD) zu einem bestimmten ganzzahligen Divisor $D \in \mathbb{Z}_{\neq 0}$
 - **Beispiel:** $1999 = 4 \cdot 440 + 239 = (4, 239)_{440}$
 - **Primfaktorpotenzen**
 - **Beispiel:** $440 = 2^3 \cdot 3^0 \cdot 5^1 \cdot 7^0 \cdot 11^1 = \text{PRIME}(3, 0, 1, 0, 1)$
- *diese Darstellungsformen sind eindeutig und existent*

Eigenschaften von Darstellungsformen

- Existenz
 - für jede Zahl existiert mindestens eine Darstellung
 - *wichtig, um jede Zahl darstellen zu können*
- Eindeutigkeit
 - für jede Zahl existiert maximal eine Darstellung
 - *wichtig, um Gleichheit von Zahlen (leicht) zu erkennen*

Kurzzahldarstellung im Computer

- Linearkombinationskoeffizienten (Ziffernfolge)
- zur Basis $B = 2$
- Länge beschränkt durch Wortlänge des Systems (z.B. 32)
- implizites Vorzeichen
 - Basiskomplementdarstellung negativer Zahlen
 - eingeschränkter Zahlenbereich $z \in \mathbb{Z}$, mit $-2^{\text{Wortlänge}-1} \leq z < 2^{\text{Wortlänge}-1}$

Kurzzahlarithmetik im Computer

- Grundoperationen i.d.R. in Hardware realisiert
 - Addition \rightarrow Operanden $-2^{\text{Wortlänge}-2} \leq z < 2^{\text{Wortlänge}-2}$
 - Subtraktion \rightarrow Operanden $-2^{\text{Wortlänge}-1} < z < 2^{\text{Wortlänge}-1}$
 - Multiplikation \rightarrow Operanden $-2^{\lfloor (\text{Wortlänge}-1)/2 \rfloor} \leq z < 2^{\lfloor (\text{Wortlänge}-1)/2 \rfloor}$
 - Division \rightarrow Operanden $-2^{\text{Wortlänge}-1} < z < 2^{\text{Wortlänge}-1}$

Langzahldarstellung im Computer

1 Wort
=
1 Ziffer

- Linearkombinationskoeffizienten (Ziffernfolge)
 - **pro Ziffernstelle ein Datenwort**
 - zur Basis $B = 2^{\lfloor (\text{Wortlänge}-1)/2 \rfloor}$
 - zusätzlich explizites Vorzeichen (\rightarrow Zahlenraum \mathbb{Z})
 - Implementierung mit dynamischer Datenstruktur
 - Lineare Liste der Ziffern
 - Variables Array der Ziffern
 - *Datenstruktur wird dynamisch der Länge der Zahl angepasst*
- \rightarrow algorithmisch beliebig lange ganze Zahlen verarbeitbar**
(Größenbeschränkung nur durch Speicher des System)

3. Algorithmen der Langzahlarithmetik

Grundoperationen

1. Relationale Operationen
2. Addition/Subtraktion
3. Multiplikation

➔ *Grundoperationen der Langzahlarithmetik werden auf **Anwendung der Kurzzahlarithmetik** zurückgeführt*

$$a \langle ? \rangle b$$

3.1. Relationale Operationen

1. Allgemeine Vergleichsoperation

- Ziffern jeweils gleicher Stellen werden, beginnend bei der letzten, miteinander verglichen
 - Gesamtergebnis steht fest, sobald Ziffern einer Stelle unterschiedlich, oder die vorderste Stelle der kürzeren Zahl verglichen wurde
 - falls Gesamtergebnis noch nicht fest steht, nächste Stelle vergleichen
 - Gesamtvergleichsergebnis ist Ergebnis des letzten Ziffernvergleichs
- jeder Ziffernvergleich kann durch 1 Kurzzahlvergleich durchgeführt werden
- zusätzlich Vergleich des Vorzeichens notwendig
- maximal $\min(\#a, \#b) + 1$ Kurzzahlvergleiche

$$a \geq b \in O(\min(\#a, \#b))$$

2. Prädikative Vergleiche

$a \geq b ?$

- auf Basis der allgemeinen Vergleichsoperation können leicht spezifische Vergleichsoperationen implementiert werden

$= \neq < \leq > \geq$

3.2. Addition/Subtraktion

1. Addition natürlicher Zahlen

$$|a|+|b|$$

Inkrementelle Methode

- zur größeren Zahl wird 1 addiert
 - dies wird solange wiederholt, bis die Anzahl der Inkrementierungen den Wert der kleineren Zahl erreicht hat
- jede Addition von 1 zu einer Zahl z kann durch $2 \cdot \#z$ Kurzzahladditionen durchgeführt werden
- maximal $2 \cdot \max(\#a, \#b) \cdot (B^{\min(\#a, \#b)} - 1)$ Kurzzahladditionen
- für gleichlange Operanden: $|a|+|b| \in O(\#z \cdot 2^{\#z \cdot \lfloor (Wortlänge-1)/2 \rfloor})$
- *exponentielles Wachstumsverhalten* → **ungeeigneter Algorithmus**

Addition natürlicher Zahlen

$$|a|+|b|$$

Schulmethode („schriftliche Addition“)

- Ziffern an den gleichen Stellen werden jeweils addiert (beginnend bei der letzten Stelle)
 - eventuell entstehender Übertrag (≤ 1) wird jeweils bei der nächst höherwertigen Stelle mit hinzuaddiert
- jede Addition von zwei Ziffern (bzw. Ziffer und Übertrag) kann durch eine Kurzzahladdition durchgeführt werden
- maximal $2 \cdot \max(\#a, \#b)$ Kurzzahladditionen
- Summe maximal um eine Stelle länger

$$|a|+|b| \in O(\max(\#a, \#b))$$

$$\#(|a|+|b|) \leq \max(\#a, \#b)+1$$

$$|a| - |b|$$

für $|a| \geq |b|$

2. Subtraktion natürlicher Zahlen

Schulmethode („schriftliche Subtraktion“)

- Voraussetzung: Minuend \geq Subtrahend
 - Ziffern an den gleichen Stellen werden jeweils subtrahiert (beginnend bei der letzten Stelle)
 - ergibt sich dabei eine negative Zahl, entsteht ein Übertrag und für diese Stelle wird das Basiskomplement des Betrags dieser Zahl gebildet
 - eventuell entstehender Übertrag (≤ 1) wird jeweils bei der nächst höherwertigen Stelle zusätzlich abgezogen
- ➔ jede Subtraktion von zwei Ziffern (bzw. Ziffer und Übertrag) kann durch eine Kurzzahlsubtraktion durchgeführt werden
- ➔ jedes Basiskomplement kann durch zwei Kurzzahloperationen gebildet werden
- ➔ maximal $4 \cdot \max(\#a, \#b) - 3$ Kurzzahloperationen
- ➔ Differenz maximal genauso lang

$$|a| - |b| \in O(\#a)$$

$$\#(|a| - |b|) \leq \#a$$

3. Addition ganzer Zahlen

$a+b$

Rückführung auf Operation natürlicher Zahlen

- falls $sign(a) = sign(b) \Rightarrow |a+b| = |a|+|b|$, $sign(a+b) = sign(a)$
- falls $sign(a) \neq sign(b)$
 - falls $|a| \geq |b| \Rightarrow |a+b| = |a|-|b|$, $sign(a+b) = sign(a)$
 - falls $|a| < |b| \Rightarrow |a+b| = |b|-|a|$, $sign(a+b) = sign(b)$
- Beträge und Vorzeichen leicht zugreifbar
(aufgrund getrennter Darstellung von Betrag und Vorzeichen)

➔ ein Langzahlvergleich notwendig

➔ maximal $4 \cdot \max(\#a, \#b) + \min(\#a, \#b) - 2$ Kurzzahlop.

➔ Summe maximal um eine Stelle länger

$$a+b \in O(\max(\#a, \#b))$$

$$\#(a+b) \leq \max(\#a, \#b) + 1$$

4. Subtraktion ganzer Zahlen

$a-b$

Rückführung auf Addition ganzer Zahlen

- es gilt $a-b = a + (-b)$
- Vorzeichen leicht invertierbar
(aufgrund getrennter Darstellung von Betrag und Vorzeichen)

→ maximal $4 \cdot \max(\#a, \#b) + \min(\#a, \#b) - 1$ Kurzzahlop.

→ Summe maximal um eine Stelle länger

$a-b \in O(\max(\#a, \#b))$

$\#(a-b) \leq \max(\#a, \#b) + 1$

Langzahladdition und -Subtraktion

- **Gesamtkomplexität** (Kostenmaß: Kurzzahloperationen)
 - $O(\max(\#a, \#b))$
 - $O(\#z)$ bei gleich langen Operanden

3.3. Multiplikation

$$a \cdot b$$

1. Grundsätzliches

- Langzahldarstellung: Betrag und Vorzeichen getrennt
- für Beträge genügt Algorithmus für natürliche Zahlen
- für das ganzzahlige Resultat gilt

$$|a \cdot b| = |a| \cdot |b|, \quad \text{sign}(a \cdot b) = \text{sign}(a) \cdot \text{sign}(b)$$

→ *jeder Algorithmus zur Multiplikation natürlicher Zahlen kann leicht auf Multiplikation ganzer Zahlen erweitert werden*

→

$$|a| \cdot |b|$$

- Produkt maximal so lang wie beide Operanden zusammen $\#(a \cdot b) \leq \#a + \#b$
- Multiplikationen mit Potenzen der Basis können durch einfaches Anhängen von Stellen (mit Ziffer 0) erfolgen

$$a \cdot b = \sum_{i=1}^b a$$

2. Rückführung auf Langzahladdition

Inkrementelle Methode

- die größere Zahl wird zu sich selbst addiert (Langzahladdition)
 - dies wird solange wiederholt, bis die Anzahl der Additionen den Wert der kleineren Zahl erreicht hat
- jeder Additionsschritt kann durch $2 \cdot \max(\#a, \#b)$ Kurzzahladditionen durchgeführt werden
- maximal $2 \cdot \max(\#a, \#b) \cdot \min(a, b) + \min(\#a, \#b)$ Kurzzahladditionen
- bei gleich langen Operanden $a \cdot b \in O(\#z \cdot B^{\#z}) \rightarrow$ **ungeeigneter Algorithmus**

Da zur Abschätzung der Komplexität immer der Worst-Case betrachtet wird, werden die Algorithmen im Folgenden der Einfachheit halber immer mit gleich langen Operanden betrachtet.

$$a \cdot b = \sum_{k=0}^{\#a-1} a_k B^k \cdot \sum_{l=0}^{\#b-1} b_l B^l = \sum_{l=0}^{\#b-1} \left(\sum_{k=0}^{\#a-1} a_k \cdot b_l \cdot B^{k+l} \right)$$

3. Rückführung auf Kurzzahlarithmetik

Fortgeschrittene Schulmethode („schriftliche M.“)

- alle Kombinationen von Ziffern beider Zahlen werden paarweise miteinander multipliziert (Kurzzahlmultiplikation)
- Kombinationsresultate werden kumulativ zu den entsprechenden Stellen des Gesamtergebnisses addiert
 - dabei entstehende Überträge werden unmittelbar in die nachfolgende Akkumulation einbezogen
- ➔ jede Ziffernmultiplikation kann durch 1 Kurzzahlmultiplikation durchgeführt werden
- ➔ jeder Akkumulationsschritt von Teilprodukt und Übertrag zu einer Resultatstelle kann durch 2 Kurzzahladditionen durchgeführt werden
- ➔ maximal $3 \cdot \#z^2$ Kurzzahloperationen

$$a \cdot b \in O(\#z^2)$$

$$a = \vec{a} \cdot B^{\#z/2} + \vec{a} \quad , \quad b = \vec{b} \cdot B^{\#z/2} + \vec{b} \quad \Rightarrow \quad a \cdot b = \vec{a} \cdot \vec{b} \cdot B^{\#z} + (\vec{a} \cdot \vec{b} + \vec{a} \cdot \vec{b}) \cdot B^{\#z/2} + \vec{a} \cdot \vec{b}$$

4. Rekursive Bisektionierung

für $\#z = 2^n$

Symmetrische Komposition

- jeder Operand wird der Länge nach in zwei Hälften aufgeteilt
- alle paarweisen Kombinationen der entstandenen vier Teile werden miteinander multipliziert (ergibt vier Teilprodukte)
- Teilprodukte werden entsprechend ihrer resultierenden Wertigkeit summiert
- **Teilprodukte werden rekursiv durch erneute Halbierung berechnet**
 - Produkte zweier Ziffern werden mittels Kurzzahlmultiplikation berechnet
- ➔ ergibt 4 rekursive Multiplikationen pro Schritt
 - dies führt letztlich zu den selben Ziffernmultiplikationen, wie bei der Schulmethode — *nämlich schlicht allen Ziffernkombinationen!*
- ➔ im r . Rekursionsschritt ergeben sich 3 Langzahladditionen mit Längen $\leq \#z/2^{r-1}$
- ➔ maximal $13 \cdot \#z^2 - 12 \cdot \#z$ Kurzzahloperationen $a \cdot b \in O(\#z^2)$

Rekursive Bisektionierung

für $\#z = 2^n$

Algorithmus von Karatsuba

- es gilt $\vec{a} \cdot \vec{b} + \vec{a} \cdot \vec{b} = \vec{a} \cdot \vec{b} + \vec{a} \cdot \vec{b} + (\vec{a} - \vec{a}) \cdot (\vec{b} - \vec{b})$
- durch Einsetzen in die Bisektionsgleichung ergibt sich
$$\Rightarrow a \cdot b = \vec{a} \cdot \vec{b} \cdot B^{\#z} + (\vec{a} \cdot \vec{b} + \vec{a} \cdot \vec{b} + (\vec{a} - \vec{a}) \cdot (\vec{b} - \vec{b})) \cdot B^{\#z/2} + \vec{a} \cdot \vec{b}$$
- rekursive Berechnung (analog zur symmetrischen Komposition)
- ➔ durch Mehrfache Verwendung von Teilprodukten ergeben sich nur 3 rekursive Multiplikationen pro Schritt, bei gleicher Rekursionstiefe
 - *weniger Multiplikationen als bei der symmetrischen Komposition!*
- ➔ im r . Rekursionsschritt ergeben sich 6 Langzahlladditionen mit Längen $\leq \#z/2^{r-1}$
- ➔ langsames Wachstum der Kurzzahloperationen $a \cdot b \in O(\#z^{\log_2(3)}) \subseteq O(\#z^{1,6})$

4. Fazit

- **exakte Arithmetik für beliebig lange ganze Zahlen in Computersystemen**
- **Langzahladdition/-Subtraktion** $a \pm b \in O(\#z)$
- **Langzahlmultiplikation**
 - Schulalgorithmus $a \cdot b \in O(\#z^2)$
 - Karatsuba-Algorithmus $a \cdot b \in O(\#z^{\log_2(3)})$
 - Toom-Cook-Algorithmus $a \cdot b \in O(\#z^{1+\varepsilon})$, mit $\varepsilon > 0$
 - Schönhage-Strassen-Algorithmus $a \cdot b \in O(\#z \cdot \log(\#z) \cdot \log(\log(\#z)))$

Ausblick

- **weitere Operationen sind auf Grundlage der Multiplikation definiert, bzw. davon abgeleitet**
 - Division
 - Potenzierung, Wurzelziehen, Logarithmus
 - Fakultät
 - ...
- ***effiziente Algorithmen zu ihrer Berechnung...?***

Zeit für Fragen...