Computer Algebra

Sebastian Iwanowski
FH Wedel

2. Integer Arithmetics
2.1 Integer representation, comparisons, addition, multiplication

Referenzen zum Nacharbeiten (in German):

Köpf 3.1,3.2, Kaplan 4.1 (bis 4.1.3)
Seminararbeit 2 (Jörg Fitzner)

Computer Algebra 2

Representation of integers in a computer

Representation of „short numbers“

- One bit per digit → works for number base 2 only
- One word (e.g.: 32 bit) per number → limits the absolute value of representable numbers
- Arithmetic operations implemented in hardware
  → limits the absolute value of input numbers even further
  → makes run time independent of number size
- Size of short numbers is limited by O(word length)

Details: Fitzner 7

Computer Algebra 2

Representation of integers in a computer

Representation of „long numbers“

- One **word** per digit → works for number base O(word length)
- List of words per number → no limit for size of representable numbers
  → makes run time dependent on number size
- Extra bit for sign → for representation of arbitrary integers

Details: Fitzner 8

Computer Algebra 2

Algorithms for long numbers of size O(n)

Comparison operators

- Compare each digit seperately starting from highest digit

- At latest when last digit of shorter word is reached, the result can be decided

→ run time O(min{#a,#b}) = O(n)

→ This works for the operators =, ≠, <, ≤, >, ≥

Details: Fitzner 10/11

Computer Algebra 2

Algorithms for long numbers of size O(n)

Addition and subtraction („school method“)

- Perform the operations digit by digit starting with the least digit.
- This results in at most 2 short number operations (considering the carriage number).

  → run time O(max{#a,#b}) = O(n)
- Integer operations may be performed with natural number operations plus sign manipulations.

  → run time O(max{#a,#b}) = O(n)
- Subtraction need not be considered separate from addition considering integers.

  → run time O(max{#a,#b}) = O(n)

Details: Fitzner 13 – 17 + Assignment 1

Computer Algebra 2

Algorithms for long numbers of size O(n)

Multiplication („school method“)

- Sign computation may be performed separately.

  → constant run time (independent of number size)

- Compute digit times number digit by digit starting with the least digit.

- This results in at most 2 short number operations (considering the carriage number).

  → run time $O(\max\{\#a,\#b\}) = O(n)$

- Shift the result according to the position of the multiplicator digit.

  → run time $O(n)$

- Compute the sum of the $O(n)$ resulting integers using long number addition.

  → run time $O(n)$ for 2 long numbers

  → run time $O(n^2)$ for n long numbers

Details: Fitzner 20

Computer Algebra 2

Algorithms for long numbers of size O(n)

Multiplication more sophisticated

Recursive bisection for numbers $a = a_1 \cdot \text{base}^{n/2} + a_2$ and $b = b_1 \cdot \text{base}^{n/2} + b_2$

- Split numbers into two halves of equal size.

long number addition for size at most 2n

- Compute the result $a \cdot b = a_1 \cdot b_1 \cdot \text{base}^n + (a_1 \cdot b_2 + a_2 \cdot b_1) \cdot \text{base}^{n/2} + a_2 \cdot b_2$

long number multiplication for size n/2 shift operation

- Needing 4 long number multiplications and 3 long number additions, the run time satisfies the following recursive formula:

$$T(n) = 4\,T(n/2) + O(n) \quad \Rightarrow \quad T(n) \in O(n^2)$$

- This is no improvement yet to the school method !

Details: Fitzner 21

Computer Algebra 2

Algorithms for long numbers of size O(n)

Multiplication more sophisticated (with idea of Karatsuba)

Recursive bisection for numbers $a = a_1 \cdot base^{n/2} + a_2$ and $b = b_1 \cdot base^{n/2} + b_2$

- Use the equality $a_1 \cdot b_2 + a_2 \cdot b_1 = a_1 \cdot b_1 + a_2 \cdot b_2 + (a_1 - a_2) \cdot (b_2 - b_1)$
- Compute the result $a \cdot b = a_1 \cdot b_1 \cdot base^n + (a_1 \cdot b_1 + a_2 \cdot b_2 + (a_1 - a_2) \cdot (b_2 - b_1)) \cdot base^{n/2} + a_2 \cdot b_2$
- Reusing the result for $a_1 \cdot b_1$ and $a_2 \cdot b_2$,
  this needs 3 long number multiplications and 6 long number additions
- Run time satisfies the following recursive formula:

  $$T(n) = 3\,T(n/2) + O(n) \quad \Rightarrow \quad T(n) \in O(n^{\log_2(3)})$$

- Since $\log_2(3) \approx 1.6$, this is indeed an improvement to the school method !

Details: Fitzner 22