Arithmetik mit sehr großen Zahlen

Zahlendarstellung durch Folgen von Ziffern:

37201148 bedeutet

$$3\cdot 10^7+7\cdot 10^6+2\cdot 10^5+0\cdot 10^4+1\cdot 10^3+1\cdot 10^2+4\cdot 10^1+8\cdot 10^0$$

Allgemein:

n-stellige Zahl bzgl Basis B dargestellt durch Folge von n „Ziffern“

$a_{n-1}a_{n-2}\cdots a_2a_1a_0$ bedeutet

$$a_{n-1}B^{n-1}+a_{n-2}B^{n-2}+\cdots a_2B^2+a_1B^1+a_0B^0 \quad = \quad \sum_{0\leq i<n} a_iB^i$$

Ziffern $0\leq a_i < B$

$B=10$ „dezimal“ $B=2$ „binär”

Arithmetik mit sehr großen Zahlen

Zahlendarstellungen manipulieren, so dass Rechenoperationen dargestellt werden

z.B. Addition:

aus zwei Ziffernfolgen A und B

produziere Ziffernfolge S,

die die Summe der von A und B dargestellten Zahlen darstellt.

Grundoperationen auf Ziffern

Addition von zwei Ziffern in einer Zeiteinheit

x,y : Ziffern $\rightarrow$ s,c : Ziffern

$s = (x+y) \bmod B$

$c = (x+y) \operatorname{div} B$ (“Carry”, “Übertrag”)

Grundoperationen auf Ziffern

Addition von zwei Ziffern in einer Zeiteinheit

DIGITADD(x,y : Ziffern) $\to$ s,c : Ziffern

$s = (x+y) \bmod B$

$c = (x+y) \operatorname{div} B$ (“Carry”, “Übertrag”)

Noch besser:

Addition von zwei Ziffern und einem Übertrag in einer Zeiteinheit

DIGITADDWITHCARRY(x,y,ü : Ziffern) $\to$ s,c : Ziffern

$s = (x+y+ü) \bmod B$

$c = (x+y+ü) \operatorname{div} B$ (“Carry”, “Übertrag”)

Grundoperationen auf Ziffern

Multiplikation von zwei Ziffern in einer Zeiteinheit

DigitMult(x,y : Ziffern)$\rightarrow$ p,c : Ziffern

$p = (x \cdot y) \bmod B$

$c = (x \cdot y) \operatorname{div} B$ ("Carry", "Übertrag")

Noch besser:

Addition von zwei Ziffern und einem Übertrag in einer Zeiteinheit

DigitMultWithCarry(x,y,ü : Ziffern)$\rightarrow$ p,c : Ziffern

$p = (x \cdot y + ü) \bmod B$

$c = (x \cdot y + ü) \operatorname{div} B$ ("Carry", "Übertrag")

Einfache Operationen auf n-stelligen Zahlen

X und Y .... n-stellige Zahlen

Addition: $X + Y$ braucht n Schritte (erweiterte Ziffernadditionen)

Ergebnis Z …. $(n+1)$-stellig

```
ü = 0
for i from 0 to n-1 do
    s,c = DigitAddWithCarry( X[i],Y[i],ü )
    Z[i] = s
    ü = c
Z[n] = ü
```

Einfache Operationen auf n-stelligen Zahlen

X und Y .... n-stellige Zahlen

Addition: X + Y braucht n Schritte (erweiterte Ziffernadditionen)

Ergebnis Z …. (n+1)-stellig

Asymptotische Laufzeit

```
ü = 0
for i from 0 to n-1 do
    s,c = DigitAddWithCarry( X[i],Y[i],ü )
    Z[i] = s
    ü = c
Z[n] = ü
```

O(1) O(n) O(n)

Einfache Operationen auf n-stelligen Zahlen

X .... n-stellige Zahl

Multiplikation von X mit Ziffer b

braucht n Schritte (erweiterte Ziffernmultiplikationen).

Ergebnis Z …. (n+1)-stellig

```
ü = 0
for i from 0 to n-1 do
    s,c = DigitMultWithCarry( X[i],b,ü )
    Z[i] = s
    ü = c
Z[n] = ü
```

Einfache Operationen auf n-stelligen Zahlen

X .... n-stellige Zahl

Multiplikation von X mit Ziffer b
braucht n Schritte (erweiterte Ziffernmultiplikationen).

Ergebnis Z …. (n+1)-stellig

Asymptotische Laufzeit

```
ü = 0
for i from 0 to n-1 do
    s,c = DigitMultWithCarry( X[i],b,ü )
    Z[i] = s
    ü = c
Z[n] = ü
```

O(1)

O(n)

O(n)

Einfache Operationen auf n-stelligen Zahlen

X .... n-stellige Zahl

Multiplikation von X mit B^k (“k Nullen anhängen”)

braucht $n+k$ Schritte

Ergebnis Z …. $(n+k)$-stellig

for i **from** 0 **to** k-1 **do** Z[i] = 0
for i **from** 0 **to** n-1 **do** Z[k+i] = X[i]

Einfache Operationen auf n-stelligen Zahlen

X .... n-stellige Zahl

Multiplikation von X mit B^k (“k Nullen anhängen”)

braucht n+k Schritte

Ergebnis Z …. (n+k)-stellig

Asymptotische Laufzeit

for i **from** 0 **to** k-1 **do** Z[i] = 0 O(k)
for i **from** 0 **to** n-1 **do** Z[k+i] = X[i] O(n)

$\left.\begin{matrix} O(k) \\ O(n) \end{matrix}\right\} O(n+k)$

Einfache Operationen auf n-stelligen Zahlen

X und Y .... n-stellige Zahlen

Addition: $X + Y$ braucht n Schritte (erweiterte Ziffernadditionen)

Multiplikation von X mit Ziffer b
braucht n Schritte (erweiterte Ziffernmultiplikationen).

Multiplikation von X mit B^i (“i Nullen anhängen”)

braucht $n+i$ Schritte

Subtraktion $X - Y$ braucht ebenfalls nur n Schritte

Quadrieren einer n-stelligen Zahl

$$A = \sum_{0 \le i < n} a_i B^i$$

Schulmethode:

$$A^2 = A \cdot A = \left(\sum_{0 \le i < n} a_i B^i\right) \cdot A = \sum_{0 \le i < n} \underbrace{\underbrace{(a_i \cdot A)}_{\text{Mult. Zahl mit Ziffer}} \cdot B^i}_{\text{Mult. Zahl mit } B^i}$$

Quadrieren einer n-stelligen Zahl

$A = \sum_{0 \leq i < n} a_i B^i$

Schulmethode:

$$A^2 = A \cdot A = \left(\sum_{0 \leq i < n} a_i B^i\right) \cdot A = \sum_{0 \leq i < n} \underbrace{\underbrace{(a_i \cdot A)}_{\text{Mult. Zahl mit Ziffer}} \cdot B^i}_{\text{Mult. Zahl mit } B^i}$$

```
E = 0
for i=0 to n-1 do
     U = MultMitZiffer(A,A[i])
     V = MultMitBasisPotenz(U,B^i)
     E = E + V
return E
```

Quadrieren einer n-stelligen Zahl

$A = \sum_{0 \leq i < n} a_i B^i$

Asymptotische Laufzeit $O(n^2)$

Schulmethode:

$$A^2 = A \cdot A = \left(\sum_{0 \leq i < n} a_i B^i\right) \cdot A = \underbrace{\sum_{0 \leq i < n} \underbrace{(a_i \cdot A)}_{\text{Mult. Zahl mit Ziffer}} \cdot B^i}_{\text{Mult. Zahl mit } B^i}$$

```
E = 0
for i=0 to n-1 do
      U = MultMitZiffer(A,A[i])
      V = MultMitBasisPotenz(U,B^i)
      E = E + V
return E
```

n Schritte

$n+i \leq 2n$ Schritte

$\leq 2n$ Schritte

$\leq 5n$ Schritte

in n Iterationen $\leq 5n^2$ Schritte

Quadrieren einer n-stelligen Zahl

Karatsuba-Ofmann-Methode: („Teile und Herrsche“)

$$A = \sum_{0 \leq i < n} a_i B^i = A_1 B^{n/2} + A_0$$

$$A_0 = a_{n/2-1} \cdots a_1 a_0$$
$$A_1 = a_{n-1} \cdots a_{n/2+1} a_{n/2}$$

$$A^2 = (A_1 B^{n/2} + A_0)^2 = (A_1)^2 B^n + (2 A_1 A_0) B^{n/2} + (A_0)^2$$

Quadrieren einer n-stelligen Zahl A kann zurückgeführt werden auf
- das Quadrieren zweier $(n/2)$-stelligen Zahlen,
- eine Produktberechnung zweier $(n/2)$-stelliger Zahlen
- Overhead proportional zu n (Additionen, Multiplikationen mit Basispotenz)

Quadrieren einer n-stelligen Zahl

Karatsuba-Ofmann-Methode: („Teile und Herrsche“)

$$A^2 = (A_1 B^{n/2} + A_0)^2 = (A_1)^2 B^n + \mathbf{(2A_1A_0)} B^{n/2} + (A_0)^2$$

$$\mathbf{(2A_1A_0)} = (A_1)^2 + (A_0)^2 - (A_1 - A_0)^2$$

Quadrieren einer n-stelligen Zahl A kann zurückgeführt werden auf
das Quadrieren **dreier** $(n/2)$-stelligen Zahlen,
~~eine Produktberechnung zweier (n/2)-stelliger Zahlen~~
Overhead proportional zu n (Additionen, Multiplikationen mit Basispotenz)

Quadrieren einer n-stelligen Zahl

Karatsuba-Ofmann-Methode: („Teile und Herrsche“)

$$A^2 = (A_1 B^{n/2} + A_0)^2 = (A_1)^2 B^n + \mathbf{(2A_1A_0)}B^{n/2} + (A_0)^2$$

$$\mathbf{(2A_1A_0)} = (A_1)^2 + (A_0)^2 - (A_1-A_0)^2$$

```
K-O-Quadriere(A)
   if n=length(A)≤ 1 then use brute force
   schneide A in A_1 und A_0
   U = K-O-Quadriere(A_1)
   V = K-O-Quadriere(A_0)
   W = K-O-Quadriere(A_1-A_0)
   return U·B^n + (U+V-W)·B^(n/2) + V
```

Laufzeitabschätzung für K-O-Quadriere()

$f(n)$ .. Anzahl der Ziffernoperationen von K-O-Quadriere(A) bei n-stelligem A

$f(n) \leq 1$ falls $n \leq 1$

$f(n) \leq 3 \cdot f(n/2) + c \cdot n$ für eine von n unabhängige Konstante c

Laufzeitabschätzung für K-O-Quadriere()

$f(n)$ .. Anzahl der Ziffernoperationen von K-O-Quadriere(A) bei n-stelligem A

$f(n) \leq 1$ falls $n \leq 1$

$f(n) \leq 3 \cdot f(n/2) + c \cdot n$ für eine von n unabhängige Konstante c

$$f(n) \leq cn + 3 \cdot f(n/2) \leq cn + 3(c \cdot n/2 + 3f(n/4)) = (1+3/2)cn + 3^2 f(n/2^2) \leq$$

$$(1+3/2)cn + 3^2(c \cdot n/2^2 + 3f(n/2^3)) \qquad = (1+3/2+(3/2)^2)cn + 3^3 f(n/2^3) \leq$$

$$(1+3/2+(3/2)^2)cn + 3^3(c \cdot n/2^3 + 3f(n/2^4)) = (1+3/2+(3/2)^2+(3/2)^3)cn + 3^3 f(n/2^3) \leq$$

$$(1+3/2+\cdots+(3/2)^{k-1})cn + 3^k f(n/2^k)$$

Laufzeitabschätzung für K-O-Quadriere()

$f(n)$ .. Anzahl der Ziffernoperationen von K-O-Quadriere(A) bei n-stelligem A

$f(n) \leq 1$ falls $n \leq 1$

$f(n) \leq 3 \cdot f(n/2) + c \cdot n$ für eine von n unabhängige Konstante c

$$f(n) \leq cn + 3 \cdot f(n/2) \leq cn + 3(c \cdot n/2 + 3f(n/4)) = (1+3/2)cn + 3^2 f(n/2^2) \leq$$

$$(1+3/2)cn + 3^2(c \cdot n/2^2 + 3f(n/2^3)) \qquad = (1+3/2+(3/2)^2)cn + 3^3 f(n/2^3) \leq$$

$$(1+3/2+(3/2)^2)cn + 3^3(c \cdot n/2^3 + 3f(n/2^4)) = (1+3/2+(3/2)^2+(3/2)^3)cn + 3^3 f(n/2^3) \leq$$

$$(1+3/2+\cdots+(3/2)^{k-1})cn + 3^k f(n/2^k)$$

Für $k=\log_2 n$, also $2^k = n$ ergibt das

$$f(n) \leq 2(3/2)^k cn + 3^k f(1) = (2c+1)3^k = (2c+1)n^{\log_2 3} = O(n^{1.59})$$

Besser als $5n^2 = O(n^2)$ der Schulmethode !?