4. Algorithmen auf Zahlen

Themen:

- Multiplikation von binären Zahlen
- Matrixmultiplikation

4.1 Multiplikation ganzer Zahlen

Schulmethode zur Multiplikation von n-stelligen Binärzahlen a und b:

n=8:

```
aaaaaaaa * bbbbbbbb
    xxxxxxxx
     xxxxxxxx
      xxxxxxxx
       xxxxxxxx
        xxxxxxxx
         xxxxxxxx
          xxxxxxxx
           xxxxxxxx
- - - - - - - - - - - - - - - -
  xxxxxxxxxxxxxxxx
```

(evtl. fallen Zeilen weg, wenn das zugehörige Bit des Multiplikators 0 ist).

Lemma:
Die Schulmethode benötigt zur Multiplikation zweier binärer n-Bit Zahlen maximal $2n(n-1)=O(n^2)$ einstellige Bitadditionen.

Beweisskizze:
Zur Addition einer Zeile auf das Gesamtergebnis benötigt man max. n Bitadditonen zuzüglich max. n weiter Bitadditionen für die ggf. anfallenden Überträge, also 2n. Insgesamt sind n-1 Bitstrings auf das Gesamtergebnis zu addieren (nur die erste Zeile kann so übernommen werden).

Entwicklung eines **wesentlich schnelleren** Algorithmus:

Zentrale Methode: **Divide and conquer** (teile und herrsche)

- Zerlege das Problem in zwei Teile
- löse das Problem für die beiden Teile
- füge die Teilergebnisse zum Gesamtergebnis zusammen.

Wie teilt man zwei n-stellige Binärzahlen $a=(a_{n-1},...a_0)$ und $b=(b_{n-1},...,b_0)$?

Setze (für n gerade)

$a^{(1)}=(a_{n-1},...a_{n/2})$, $a^{(0)}=(a_{n/2-1},...a_0)$,
$b^{(1)}=(b_{n-1},...b_{n/2})$, $b^{(0)}=(b_{n/2-1},...b_0)$,

Dann gilt:

$$a=a^{(1)}\cdot 2^{n/2} + a^{(0)}$$
$$b=b^{(1)}\cdot 2^{n/2} + b^{(0)}$$

Dann gilt ferner:

$$a\cdot b= (a^{(1)}\cdot 2^{n/2} + a^{(0)}) \cdot (b^{(1)}\cdot 2^{n/2} + b^{(0)})$$
(1) $$= a^{(1)} \cdot b^{(1)}\cdot 2^{2\ n/2} + a^{(1)} \cdot b^{(0)}\ 2^{n/2} + a^{(0)} \cdot b^{(1)}\ 2^{n/2} + a^{(0)} \cdot b^{(0)}\ .$$

Laufzeit: 4*“Multiplikation halber Zahlen“+Addition

$T(n)=4*T(n/2)+O(n)$

Beachte: Multiplikationen mit 2 = Linksschieben = 1 Operation

Leider gilt: $T(n)=O(n^2)$, also nicht besser als die Schulmethode.

Geniale Idee:

Beobachte:

$$a^{(1)} \cdot b^{(0)} + a^{(0)} \cdot b^{(1)} = (a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}).$$

mittlerer Term in (1) ↑ ↑

hier mit 2 Multiplikationen und 1 Addition — hier mit 1 Multiplikation und 4 Additionen

Insgesamt:

- Gespart: 1 Multiplikation
- Geopfert: 3 Additionen
- Da Multiplikationen aufwendiger sind als Additionen, hat man insgesamt Operationen und Zeit gespart.

Laufzeit nun:

$$T(n) = 2\,T(\,n/2\,) + T(\,n/2\,+1) + O(n) \leq 3\,T(\,n/2\,+1) + O(n).$$

↑

wg. Addition zweier n/2-stelliger Zahlen.

Konstante in O(n):

$$a \cdot b = a^{(1)} \cdot b^{(1)} \cdot 2^{2\,n/2} + a^{(1)} \cdot b^{(0)}\, 2^{n/2} + a^{(0)} \cdot b^{(1)}\, 2^{n/2} + a^{(0)} \cdot b^{(0)}$$

$$= a^{(1)} \cdot b^{(1)} \cdot 2^{2\,n/2} + ((a^{(1)} + a^{(0)}) \cdot (b^{(1)} + b^{(0)}) - (a^{(1)} \cdot b^{(1)} + a^{(0)} \cdot b^{(0)}))\, 2^{n/2} + a^{(0)} \cdot b^{(0)}.$$

↑ ↑ ↑ ↑ ↑ ↑

$2n + 2(n/2) + 2(n/2) + 2n + 2n + 2n$

$=10n$ Bit-Operationen

Merke: Addition von 2 binären Zahlen zu je n Bits erfordert

n Bit-Additionen für die Zahlen selbst
\+ n Bit-Additionen für evtl. Überträge
= 2n Bit-Operationen.

Lösen der Rekursionsgleichung (Annahme n **Zweierpotenz**):

$$T(n) = 3\,T(n/2+1) + 10n$$
$$\leq 3^2\,T(n/2^2+2) + 3\cdot 10(n/2)+2+10n$$
$$= 3^3\,T(n/2^3+2) + 3^2\cdot 10(n/2^2)+2+3\cdot 10(n/2)+2+10n$$
... nach k Iterationen
$$\leq 3^k\,T(n/2^k+2) + \sum_{i=0}^{k-1}(3^i\cdot 10(n/2^i)+2)$$

für k=log n gilt dann

$$\leq 3^{\log n}\,T(n/2^{\log n}+2) + \sum_{i=0}^{\log n-1}(3^i\cdot 10(n/2^i)+2)$$

$\uparrow = T(3)$

$$\leq 2^{\log 3 \log n}\,T(3) + 10\sum_{i=0}^{\log n-1}(3^i(n/2^i)+2)$$
$$\leq 9n^{\log 3} + 10n\sum_{i=0}^{\log n-1}(3^i/2^i)+2\log n, \text{ da } T(3)\leq 9$$
$$\leq 9n^{\log 3} + 10n\left((3/2)^{\log n}-1\right)/(3/2-1)+2\log n$$
$$\leq 9n^{\log 3} + 20n\cdot 2^{\log 1.5 \log n}-1+2\log n$$
$$\leq 9n^{\log 3} + 20n^{\log 3}+2\log n$$
$$\leq 29n^{\log 3} +2\log n = O(n^{\log 3}). \qquad (\log 3 \approx 1.585)$$

Satz

Zwei n-stellige Binärzahlen können mit $29n^{\log 3} +2\log n = O(n^{\log 3})$ vielen Bit-Operationen multipliziert werden.

ABER:

Groß-O (d.h. asymptotischer Laufzeitvergleich) beschönigt oft die wahren Verhältnisse: Für welche n ist denn der neue Algorithmus besser als der alte:

$$29n^{\log 3} +2\log n < n^2 \;??$$

Ab etwa n=500-stelligen Binärzahlen.

Für n<500 ist die Schulmethode effizienter.

Verbesserung:
Breche den Algorithmus bei Binärzahlen der Länge 2^m+2 ab und gehe von dort zur Schulmethode über. Wie findet man den optimalen Umschaltzeitpunkt:

...

$T(n) \leq 3^k T(n/2^k +2) + \sum_{i=0}^{k-1} (3^i \cdot 10(n /2^i)+2)$

Rechne von hier für k=log n - m weiter:

...

Ergebnis nach längerer Rechnerei:
optimaler Umschaltzeitpunkt: Länge der Binärzahlen zwischen 10 und 18.
Konstante sinkt dann auf etwa 10.

Satz
Zwei n-stellige Binärzahlen können mit $10n^{\log 3} +2 \log n=O(n^{\log 3})$ vielen Bit-Operationen multipliziert werden.

Algorithmus schlägt Schulmethode ab etwa n=50.

Aktuell bester Algorithmus:

Satz (A. Schönhage, V. Strassen, 1971)
Zwei n-stellige Binärzahlen können mit O(n log n log logn) vielen Bit-Operationen multipliziert werden.

4.2 Matrizenmultiplikation

Anwendungen in allen Gebieten:

- Wegeprobleme in Graphen
- Physik
- lineare Algebra

Gegeben zwei nxn-Matrizen A und B.
Gesucht: Matrix $C=A \cdot B$

Schulmethode

- je Matrixelement c_{ij} sind n Multiplikationen und n-1 Additionen erforderlich
- n^2 Matrixelemente sind zu berechnen: $n^2(n+n-1)=2n^3-n^2=O(n^3)$

Geniale Ideen:

- Divide-and-conquer
- intelligente Berechnungsmethode, die Multiplikationen auf Kosten von Additionen spart

Generalannahme: n Zweierpotenz, sonst Matrix auf nächste Zweierpotenz erweitern

Vorgehen (Divide and conquer):

Teile A und B in je 4 gleichgroße n/2xn/2Teilmatrizen auf:

$$A = \begin{matrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{matrix} \qquad B = \begin{matrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{matrix}$$

und berechne

$$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21} \qquad C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$$
$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21} \qquad C_{22} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}$$

Zu berechnen sind also rekursiv 8 Matrixmultiplikationen von n/2xn/2-Matrizen und 4 Matrixadditionen ($=n^2$). Laufzeit:

$$T(n)=8T(n/2)+n^2.$$

Leider: Lösung wieder $O(n^3)$ und damit kein Gewinn gegenüber Schulmethode.

2. Idee: Spare durch geschickte Zwischenrechnungen eine Multiplikation ein.

Berechne:

$$M_1 = (A_{12}-A_{22})(B_{21}+B_{22}) \qquad M_2 = (A_{11}+A_{22})(B_{11}+B_{22})$$
$$M_3 = (A_{11}-A_{21})(B_{11}+B_{12}) \qquad M_4 = (A_{11}+A_{12})B_{22}$$
$$M_5 = A_{11}(B_{12}-B_{22}) \qquad M_6 = A_{22}(B_{21}-B_{11})$$
$$M_7 = (A_{21}+A_{22})B_{11}$$

Und dann:

$$C_{11} = M_1+M_2-M_4+M_6$$
$$C_{12} = M_4+M_5$$
$$C_{21} = M_6+M_7$$
$$C_{22} = M_2-M_3+M_5-M_7.$$

Insgesamt: 7 Multiplikationen, 18 Additionen.

(**Algorithmus von V. Strassen, 1969**)

Laufzeitanalyse:

$$T(n)=7T(n/2)+18(n/2)^2=7T(n/2)+9n^2/2.$$

Lösung durch Abwicklung/Iteration:

$$T(n)=7\cdot(7T(n/4)+(9/2)(n/2)^2)+9n^2/2$$
$$=7^2\ T(n/4)+7\cdot(9/2)(n/2)^2)+9n^2/2$$

Nach k Iterationen erhält man:

$$=7^k\ T(n/2^k)+\sum_{i=0}^{k-1}\ 7^{i}\cdot(9/2)(n/2^i)^2)$$

mit k=log n erhält man

$$=7^{\log n}\ T(1)+\sum_{i=0}^{\log n-1}\ (7/4)^i$$

$$=2^{\log 7 \log n}+(9/2)n^2\ \sum i_{=0}^{\log n-1}\ \ (7/4)^i$$

$$=n^{\log 7}+(9/2)n^2\ (((7/4)^{\log n}-1)/(7/4-1)$$
$$\leq n^{\log 7}+(9/2)n^2\ (4/3)\ n^{\log 7-\log 4}$$
$$\leq n^{\log 7}+6\ n^{\log 7}$$
$$=7\ n^{\log 7}\ a\ 7n^{2.81}.$$

Satz:

Der Algorithmus von Strassen beötigt zur Multiplikation zweier nxn-Matrizen $7n^{\log 7}$ viele arithmetische Operationen.

ABER:

Nachrechnen ergibt: Strassen-Algorithmus schlägt Schulmethode ab n≥700.

Verbesserung:

7 Multiplikationen und **15** Additionen ==> Strassen-Algorithmus schlägt Schulmethode ab n≥310.

Weitere Verbesserung:

Gehe zu Schulmethode über, wenn diese effizienter ist als der Strassen-Algorithmus. Ab welcher Matrizenmgröße $2^m x 2^m$ empfiehlt sich das?

Wie oben rechnen:

...

$T(n)=7^k\ T(n/2^k)+\sum_{i=0}^{k-1}\ 7^i \cdot (9/2)(n/2^i)^2)$

mit k=log n-m weiterrechnen und bestmögliches m ermitteln.

...

Ergebnis: Umschaltzeitpunkt liegt zwischen 8 und 16.
Konstante sinkt dann auf etwa 4.

Satz

Der Algorithmus von Strassen beötigt zur Multiplikation zweier nxn-Matrizen $4n^{\log 7}$ viele arithmetische Operationen, wenn man bei kleinen Matrizen auf die Schulmethode umsteigt.

Algorithmus schlägt Schulmethode ab etwa n=40.

Matrixmultiplikationsrennen:

Jahr	Autoren	Laufzeit
1969	Strassen	$O(n^{2,808})$
1979	Pan	$O(n^{2,781})$
1979	Bin, Capovani, Lotti, Romani	$O(n^{2,7799})$
1979	Schönhage	$O(n^{2,548})$
1979	Pan	$O(n^{2,522})$
1982	Coppersmith, Winograd	$O(n^{2,496})$
1982	Strassen	$O(n^{2,47})$
1982	Coppersmith, Winograd	$O(n^{2,38})$

Algorithmen wegen der **Konstanten** nur von theoretischem Interesse.