Informatik II
Divide & Conquer

G. Zachmann
Clausthal University, Germany
zach@in.tu-clausthal.de

Algorithmen-Design-Techniken

- Entwurfsverfahren für Algorithmen:
  1. Divide and Conquer
  2. Dynamische Programmierung
  3. Greedy Verfahren
  4. Backtracking
- Unser erstes allg. Verfahren: Teile und Herrsche; Divide et impera
- Allgemeine Problem-unabhängige Formulierung des Prinzips:
  - D&C-Verfahren = Methode V zur Lösung des Problems P der Größe n:
  - (Basisfall) Falls n < d, löse das Problem direkt, sonst
  - (Divide) teile P in zwei oder mehr kleine Teile $P_1, \ldots, P_k$, $k \geq 2$
  - (Conquer) Löse jedes Teilproblem P_i rekursiv mit der Methode V (auf gleiche Art)
  - (Merge) Setze die Teillösungen zusammen

Schnelle Multiplikation

- Schulverfahren (nach Adam Riese):

$$
\begin{array}{r}
x_{n-1}x_{n-2} \cdots\cdots x_1x_0 \;\times\; y_{n-1}y_{n-2} \cdots\cdots y_1y_0 \\
\hline
c^0{}_n c^0{}_{n-1} \cdots\cdots\cdots c^0{}_0 \\
c^1{}_n c^1{}_{n-1} \cdots\cdots\cdots c^1{}_0 \quad \\
\vdots \qquad\qquad \vdots \qquad\qquad \\
c^{n-1}{}_n c^{n-1}{}_{n-1} \cdots\cdots\cdots c^{n-1}{}_0 \qquad\qquad\qquad \\
\hline
z_{2n}\, z_{2n-1} \cdots\cdots\cdots\cdots\cdots\cdots z_1\, z_0
\end{array}
$$

- Ziffern-Multiplikationen: n^2
- Ziffern-Additionen: $\sim n^2$

Algorithmus von Karatsuba und Ofman

- Sei $n = 2m$ und b die Basis einer b-adischen Darstellung (z.B. dezimal, $b = 10$):

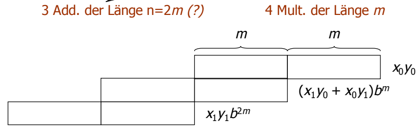
$$x = x_0 + x_1 b^m \qquad y = y_0 + y_1 b^m$$

shift gratis

$$xy = x_0y_0 + (x_1y_0 + x_0y_1)\, b^m + x_1y_1\, b^{2m}$$

3 Add. der Länge n=2*m (?)*

4 Mult. der Länge *m*



2 Additionen der Länge $\frac{n}{2} = m$

Aufwand

- Annahme: $n = 2^k$

$$
\begin{aligned}
T(n) &= 4T\left(\tfrac{n}{2}\right) + \left(2\tfrac{n}{2} + n\right) \\
&= 4T\left(\tfrac{n}{2}\right) + 2n \\
&= 4\left(4T\left(\tfrac{n}{4}\right) + 2\tfrac{n}{2}\right) + 2n \\
&= 16T\left(\tfrac{n}{4}\right) + 4n + 2n \\
&\;\vdots \\
&= \underbrace{4^k}_{2^{2k}} T(1) + 2n \underbrace{\sum_{i=0}^{k-1} 2^i}_{\approx 2^k = n} \\
&\approx c(n^2 + 2n^2) \in \Theta\left(n^2\right)
\end{aligned}
$$

Additionen

Ziffern-Additionen

Ziffern-Multiplikationen

→ Bislang noch keine Verbesserung

Der Trick von Karatsuba und Ofman

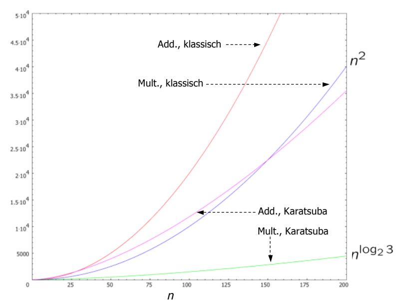
- Zu berechnen: $xy = x_0y_0 + (x_1y_0 + x_0y_1)b^m + x_1y_1b^{2m}$
- Umformung: $(x_1y_0 + x_0y_1) = (x_1 + x_0)(y_0 + y_1) - x_0y_0 - x_1y_1$
- Berechne also nur noch 3 Produkte:

$$
\begin{aligned}
P_1 &= x_0y_0 \\
P_2 &= x_1y_1 \\
P_3 &= (x_1 + x_0)(y_0 + y_1)
\end{aligned}
$$

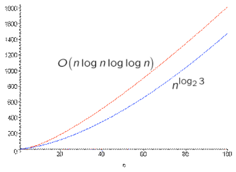
Aufwand

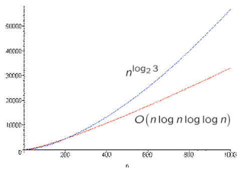
$$
\begin{aligned}
T(n) &= 3T\left(\tfrac{n}{2}\right) + cn \\
&= 3\left(3T\left(\tfrac{n}{4}\right) + c\tfrac{n}{2}\right) + cn \\
&\;\vdots \\
&= 3^k T(1) + cn \sum_{i=0}^{k-1} \left(\tfrac{3}{2}\right)^i \\
&\in O\left(3^{\log_2 n}\right) = O\left(n^{\log_2 3}\right)
\end{aligned}
$$

- Übungsaufgabe: Anzahl Additionen besser abschätzen
- Der Algorithmus von Karatsuba und Ofman ist schneller, als der bekannte Algorithmus zur Multiplikation, die rekursiven Aufrufe benötigen allerdings mehr Speicherplatz (*space-time trade-off*)



- Asymptotische Verbesserung von Schönhage und Strassen
  - Schnelle Multiplikation mit einer Laufzeit von $O(n \log n \log\log n)$





Matrix-Multiplikation (MM)

- Aufgabe: berechne $C = A \cdot B, \quad A, B \in \mathbb{R}^{n\times n}$
- Normale MM benötigt $O(n^3)$ Multiplikationen (und etwa genauso viele Additionen)
- Geht es schneller?
- Idee: zerlege A, B, C in Blöcke:

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

mit $a_{ij}, b_{ij}, c_{ij} \in \mathbb{R}^{\frac{n}{2}\times\frac{n}{2}}$

- Ausmultiplizieren ergibt:

$$c_{11} = a_{11}b_{11} + a_{12}b_{21}$$
$$\vdots \qquad \rightarrow \text{8 MM der Größe } \tfrac{n}{2} \times \tfrac{n}{2}$$
$$c_{22} = a_{21}b_{11} + a_{22}b_{22}$$

- Idee: berechne diese rekursiv durch Zerlegung in Blöcke
  → Divide-and-conquer-Algorithmus für o.g. Aufgabe
- Aufwand:

für Additionen

$$\begin{aligned} T(n) &= 8T\left(\tfrac{n}{2}\right) + cn^2 \\ &= 8\left(8T\left(\tfrac{n}{4}\right) + c\left(\tfrac{n}{2}\right)^2\right) + cn^2 \\ &= 8^2 T\left(\tfrac{n}{2^2}\right) + \tfrac{8}{2^2}cn^2 + cn^2 \\ &= \vdots \\ &= 8^k T(1) + cn^2 \sum_{i=0}^{k-1} 2^i \\ &\approx c'n^3 + 2^k cn^2 \\ &\approx c'n^3 + cn^3 \in O\left(n^3\right) \end{aligned}$$

für Additionen

für Multiplikationen

Idee von Strassen [1969]

- Berechne zunächst etwas umständliche Zwischenprodukte:

$$\begin{aligned} Q_1 &\equiv (a_{11} + a_{22})(b_{11} + b_{22}) \\ Q_2 &\equiv (a_{21} + a_{22})b_{11} \\ Q_3 &\equiv a_{11}(b_{12} - b_{22}) \\ Q_4 &\equiv a_{22}(-b_{11} + b_{21}) \\ Q_5 &\equiv (a_{11} + a_{12})b_{22} \\ Q_6 &\equiv (-a_{11} + a_{21})(b_{11} + b_{12}) \\ Q_7 &\equiv (a_{12} - a_{22})(b_{21} + b_{22}) \end{aligned}$$

- Damit kann man die c_{ij} so ausrechnen:

$$\begin{aligned} c_{11} &= Q_1 + Q_4 - Q_5 + Q_7 \\ c_{12} &= Q_2 + Q_4 \\ c_{21} &= Q_3 + Q_5 \\ c_{22} &= Q_1 + Q_3 - Q_2 + Q_6 \end{aligned}$$

Aufwand

$$
\begin{aligned}
T(n) &= 7T\left(\tfrac{n}{2}\right) + cn^2 \\
&= 7\left(7T\left(\tfrac{n}{4}\right) + c\left(\tfrac{n}{2}\right)^2\right) + cn^2 \\
&= \cdots \\
&= 7^k T(1) + cn^2 \underbrace{\sum_{i=0}^{k-1} \left(\tfrac{7}{4}\right)^i}_{\approx \left(\frac{7}{4}\right)^{\log_2 n} = n^{\log_2\left(\frac{7}{4}\right)}} \\
&\approx c' n^{\log_2 7} + cn^{2+\log_2\left(\frac{7}{4}\right)} \\
&\approx c' n^{2.8\ldots} + cn^{2.8\ldots} \in O(n^{2.8\ldots})
\end{aligned}
$$

für Additionen

für Multiplikationen

Bemerkungen

- Aktuell kleinster Exponent = 2.376
  [Coppersmith & Winograd, 1990]
- Untere Schranke für Exponent = 2 (klar, da n^2 viele Elemente)
- Eine ähnliche Formel gibt es für Matrix-Inversion