# Algorithmen

Michael Kaufmann

25/10/2021 – 3. Vorlesung
Rekursionen

# Organisatorisches

**Vorlesungstermine:** Mo 10.30-12 und Di 8.15 -9.30
**Tutorien:** meist in Präsenz. Vor allem Präsenzblatt!
**Help Desk:** Online Mo ab 12.00
**Besprechung der Übungsaufgaben:** Mo 16-18 in zoom

**Organisation:** https://moodle.zdv.uni-tuebingen.de/

# Gliederung

**I. Einführung**
- Motivation und Notationen
- Laufzeitanalyse, Rekursionen

**II. Grundlegende Datenstrukturen**
**III. Graphenalgorithmen**
**IV. Sortieren**
**V. Suchen**
**VI. Generische algorithmische Methoden**
**VII. Algorithmen auf Zeichenketten**

# Teaser aus Dasgupta, Mehlhorn et al.

**Multiplikation zweier Integer**

Für Integer der Länge jeweils $n$ braucht Schulmethode $O(n^2)$ Elementaroperationen

Formell: Wir nehmen Binärzahlen der Länge $n$ an, und sei $n$ eine Zweierpotenz (kann man immer schön halbieren).

Machen einen anderen Ansatz, nämlich Divide & Conquer:

- Spalte $x$ und $y$ in linke und rechte Hälften, so dass
$$x = 2^{n/2} x_l + x_r \text{ und } \qquad y = 2^{n/2} y_l + y_r$$
- Multipliziere kleinere Teilprobleme und Addiere:
$$x \cdot y = 2^n x_l y_l + 2^{n/2}(x_l y_r + x_r y_l) + x_r y_r$$

$\rightarrow$ haben jetzt 4 Teilprobleme,
i.e. 4 Multiplikationen zweier Zahlen der Länge $n/2$

# Multiplikation zweier Zahlen

Nun verwenden wir Rekursion auf die Multiplikation zweier Zahlen der Länge $n/2$.

Das ergibt

$$T(n) = 4 \cdot T(n/2) + O(n) \text{ wobei}$$
$$T(1) = 1$$

Dabei ist $T(n)$ die Laufzeit für die Multiplikation zweier Zahlen der Länge $n$, $n > 1$.

Kurzes Nachdenken führt zu $O(n^2)$.

# Multiplikation - Verbesserung

Versuche eine der 4 Multiplikationen zu sparen !
Das Ziel ist dann : $T(n) = 3T(n/2) + O(n)$ für $n > 1$.
Und zwar so:

$$\begin{aligned} x \cdot y &= 2^n x_l y_l + 2^{n/2}(x_l y_r + x_r y_l) + x_r y_r \\ &= 2^n x_l y_l + 2^{n/2}((x_l + x_r)(y_l + y_r) - x_l y_l - x_r y_r)) + x_r y_r \\ &= (2^n - 2^{n/2}) x_l y_l + (1 - 2^{n/2}) x_r y_r + 2^{n/2}((x_l + x_r)(y_l + y_r) \end{aligned}$$

Das ergibt 3 Multiplikationen rekursiv und ein paar Summationen mehr ($O(n)$).

Lösung obiger Rekursionsgleichung ergibt $O(n^{log 3}) = O(n^{1.59})$

Wie kommt man da drauf ? Wie löst man Rekursionen ?

# Rekursionen

**Merge-Sort:** $T(n) = 2T(n/2) + n$

**Binäre Suche:** $T(n) = T(n/2) + 1$

**Beachte**: $n$ wird als Zweierpotenz angenommen !

**Beachte**: $T(1)$ muss spezifiert werden, meist $T(1) = 1$.

1. **Raten und Beweisen**:

a. **Raten** $T(n) = O(\log n)$ für binäre Suche

b. **Beweisen** $T(n) \leq c \log n$ für geeignete Konstante $c$

per Induktion: $T(2)$ ist ok, $T(n)$ auch durch Einsetzen mit $c = 2$.

# Rekursion: 2. Ausrechnen

$T(n) = T(n/2) + 1$, $T(1) = 1$ und $n$ ist Zweierpotenz.

$$\begin{aligned}
T(n) &= T(n/2) + 1 \\
&= T(n/4) + 1 + 1 \\
&= T(n/8) + 1 + 1 + 1 \\
&\vdots \\
&= T(n/2^i) + i \\
&= T(n/2^{i+1}) + 1 + i
\end{aligned}$$

mache hier Induktion über $i$ !

für $i = \log n$:

$$\begin{aligned}
&= T(n/2^{\log n}) + \log n \\
&= T(1) + \log n \\
&= 1 + \log n = O(\log n)
\end{aligned}$$

# Rekursion: 3. Mastertheorem

**Satz**: Sei $a, b \geq 1$ konstant. $f(n)$, $T(n) \geq 0$ mit

$$\mathbf{T(n) = aT(n/b) + f(n)}$$

1. Für $f(n) = O(n^{\log_b a - \epsilon})$ mit $\epsilon > 0$ gilt $\quad T(n) = \Theta(n^{\log_b a})$

2. Für $f(n) = \Theta(n^{\log_b a})$ gilt $\quad T(n) = \Theta(n^{\log_b a} \cdot \log_b n)$

3. Für $f(n) = \Omega(n^{\log_b a + \epsilon})$ mit $\epsilon > 0$ und
   und $a \cdot f(n/b) \leq c \cdot f(n)$ für $c < 1$ und $n$ genügend groß, gilt

$$T(n) = \Theta(f(n))$$

# Rekursion: 3. Mastertheorem

**Annahme**: $n = b^i$, also Teilproblemgrößen $1, b, b^2, \ldots$

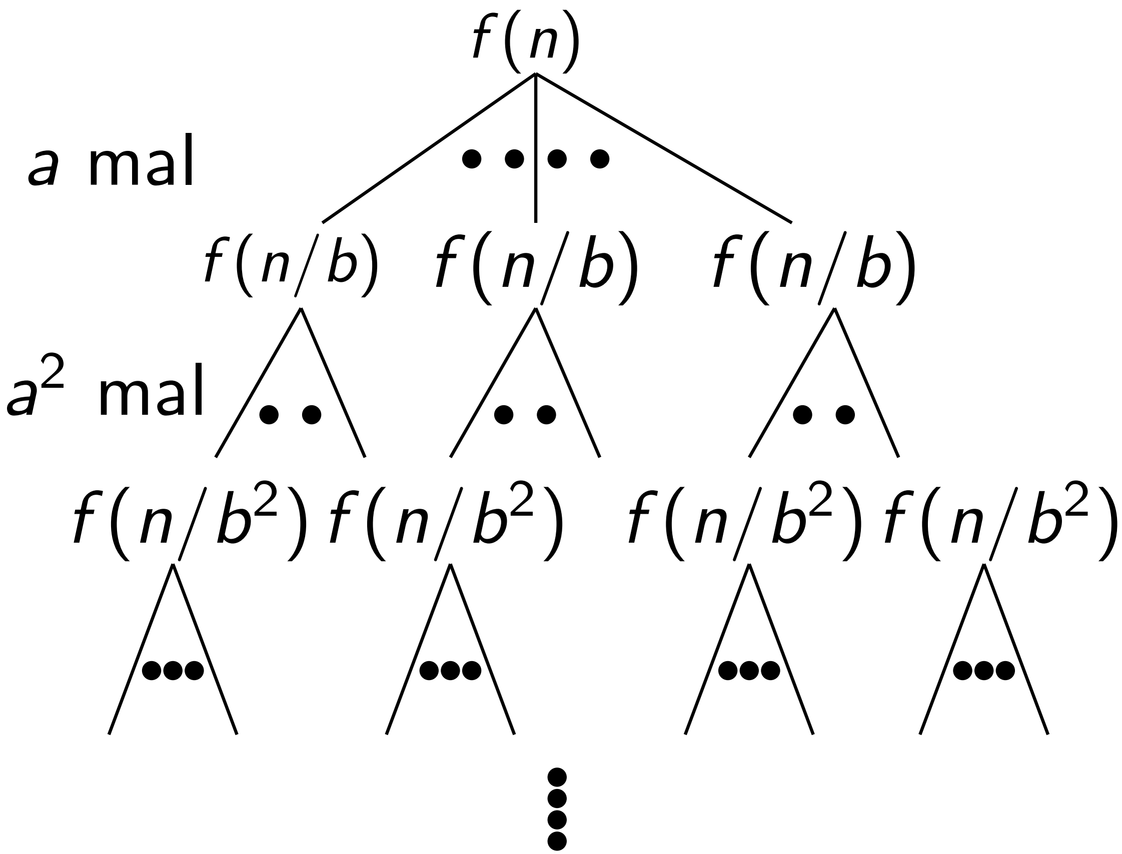
**Lemma**: Sei $a, b \geq 1$, $f(n) \geq 0$, $n = b^i$ für $i \in N$. Mit $T(1) = \Theta(1)$ gilt

$$T(n) = \Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n - 1} a^j \cdot f(n/b^j)$$

**Beweis**:
$$\begin{aligned}
T(n) &= f(n) + aT(n/b) \\
&= f(n) + af(n/b) + a^2 T(n/b^2) \\
&= f(n) + af(n/b) + a^2 f(n/b^2) + a^3 T(n/b^3) \\
&\vdots \\
&= f(n) + \ldots + a^{\log_b n - 1} \cdot f(n/b^{\log_b n - 1}) + a^{\log_b n} T(1) \\
&= \Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n - 1} a^j \cdot f(n/b^j)
\end{aligned}$$

$$T(n) = \Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n - 1} a^j \cdot f(n/b^j)$$

## Rekursionsbaum:




$f(n)$

$af(n/b)$

$a^2 f(n/b^2)$

$a^3 f(n/b^3)$

$\vdots$

$\Theta(n^{\log_b a})$

---

obige Summe

$\Rightarrow$ 3 Fälle

$$T(n) = \Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n - 1} a^j \cdot f(n/b^j)$$

**Lemma**: Sei $a, b \geq 1$, f(n) definiert auf Potenzen von $b$.
Sei $g(n) = \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j)$.

1. Ist $f(n) = O(n^{\log_b a - \epsilon})$ mit $\epsilon > 0$, so ist $g(n) = O(n^{\log_b a})$
2. Ist $f(n) = \Theta(n^{\log_b a})$, so ist $g(n) = \Theta(n^{\log_b a} \cdot \log n)$.
3. Ist $af(n/b) \leq cf(n)$, mit $c < 1, n > b$, so ist $g(n) = \Theta(f(n))$.

$\Rightarrow$ **Mastertheorem**:

1. $T(n) = \Theta(n^{log_b a}) + O(n^{log_b a}) = \Theta(n^{log_b a})$.
2. $T(n) = \Theta(n^{log_b a}) + \Theta(n^{log_b a} \cdot \log n) = \Theta(n^{log_b a} \cdot \log n)$.
3. Ist $f(n) = \Omega(n^{\log_b a + \epsilon})$ mit $\epsilon > 0$, und $af(n/b) \leq cf(n)$, so ist $T(n) = \Theta(n^{\log_b a}) + \Theta(f(n)) = \Theta(f(n))$.

# Mastertheorem - Anwendung

**Beispiele:**

Einfacher Ansatz für Multiplikation:

$$T(n) = 4 \cdot T(n/2) + O(n) \text{ mit } T(1) = 1$$

Setze ein: $a = 4, b = 2, f(n) = cn$.
Damit gilt: $\log_b a = \log_2 4 = 2$. Und $n^{2-\epsilon} > cn$ für $\epsilon = 1/2$

Wende Fall 1 des Mastertheorems an und somit

$$T(n) = O(n^{\log_b a}) = O(n^2)$$

Besserer Ansatz für Multiplikation:

$$T(n) = 3 \cdot T(n/2) + O(n) \text{ mit } T(1) = 1$$

Setze ein: $a = 3, b = 2, f(n) = cn$.
Also: $\log_b a = \log_2 3$ Und $n^{\log 3-\epsilon} > cn$ für $\epsilon = 1/2$

Wende Fall 1 des Mastertheorems an und somit

$$T(n) = O(n^{\log_b a}) = O(n^{\log_2 3}) = O(n^{1.59})$$

**Weitere Beispiele:**

Binäre Suche:

$$T(n) = T(n/2) + O(1) \text{ mit } T(1) = 1$$

Setze ein: $a = 1, b = 2, f(n) = 1 = n^0$

Damit gilt: $\log_b a = \log_2 1 = 0$. Und $n^0 = f(n)$.

Wende also Fall 2 des Mastertheorems an und somit

$$T(n) = O(n^0 \log_2 n) = O(\log_2 n)$$

GIBT ES FÄLLE, WO MASTERTHEOREM NICHT ANWENDBAR ? ÜBERLEGE ! UND DANN ??