Merge Sort Laufzeitanalyse

haben gezeigt: Für $n = 2^q$ ist

$$C(2^q) = (q-1) \cdot 2^q + 1$$

Umschreiben für n:

$$C(n) = (\log n - 1) \cdot n + 1 \in O(n \log n)$$

für $n = 2^q$

Anzahl der Zuweisungen: (auch für $n = 2^q$)

$$A(2n) = 2 \cdot A(n) + 4n$$
$$A(2) = 4$$

hat Lösung: $A(n) = 2q \cdot 2^q = 2n \log n \in O(n \log n)$

jetzt für allgemeines n:

Es sei n' die erste 2er-Potenz $\geq n$, $n' = 2^q$

$$\Rightarrow 2^{q-1} < n \leq 2^q = n' \Rightarrow \underline{2^q < 2n}$$

$$\Rightarrow \underline{q - 1 < \log n \leq q = \log n'}$$

$$C(n) \leq C(n') = C(2^q) = \underbrace{(q-1)}_{<\log n} \cdot \underbrace{2^q}_{<2n} + 1$$

$\uparrow$ # Vergleiche wächst monoton

$$\leq 2n \cdot \log n + 1$$

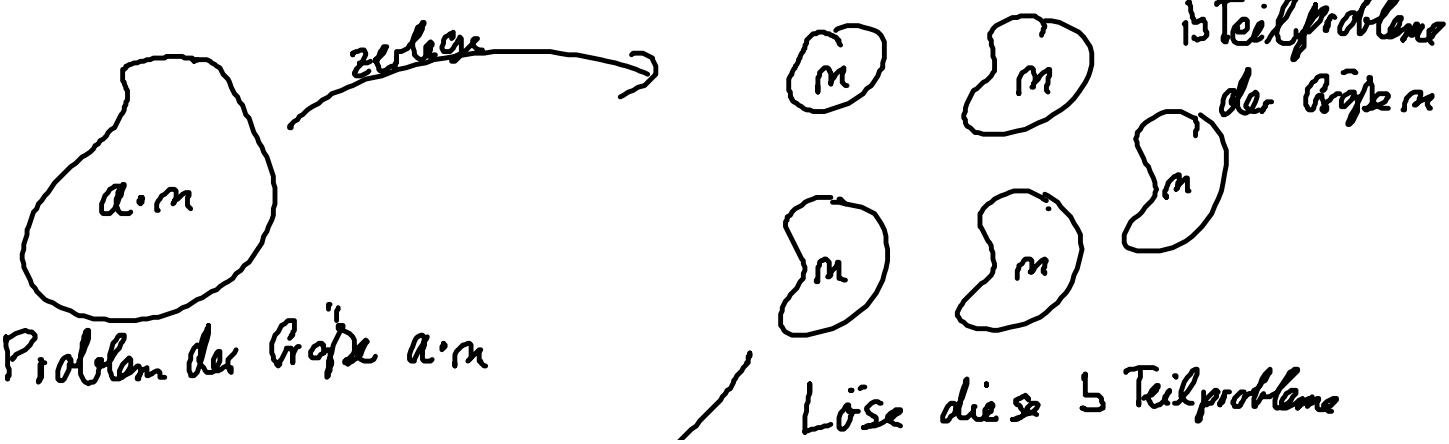
$$\in O(n \log n)$$

analog für $A(n)$

Theorem: MergeSort sortiert ein Array der Länge n in $O(n \log n)$ Vergleichen und Zuweisungen.

Diese Analyse beruht auf dem allgemeinen Prinzip des „Divide & Conquer": (oder „Aufteilung & Beschleunigung")

Sei $f(n)$ eine Laufzeitfunktion für ein Problem der Größe n:



zusammenfügen

Lösung des Ausgangsproblems

Satz (Aufteilungs- und Beschleunigungssatz):

Es seien $a > 0$, b, c natürliche Zahlen und $f: \mathbb{N} \to \mathbb{N}$ mit

$$f(1) \leq \frac{c}{a}$$

$$f(a \cdot n) \leq b \cdot f(n) + c \cdot n \quad \text{für } n > 1$$

Dann gilt

$$f(n) \in \begin{cases} O(n) & \text{falls } a > b \\ O(n \log n) & \text{falls } a = b \\ O(n^{\log_a b}) & \text{falls } a < b \end{cases}.$$

Anwendung auf MergeSort: $a = b = 2$, $c = 2$

AB-Satz $\Rightarrow C(n) \in O(n \log n)$

Beweis des AB-Satzes:

zunächst Annahme: $n = a^q$, $q \in \mathbb{N} \cup \{0\}$

Beh: $f(n) \leq \frac{c}{a} \cdot n \cdot \sum_{i=0}^{q} \left(\frac{b}{a}\right)^i$

Beweis der Beh. über vollst. Ind. über q:

$q = 0$: $f(1) \leq \frac{c}{a} = \frac{c}{a} \cdot \underbrace{n \cdot \sum_{i=0}^{0} \left(\frac{b}{a}\right)^i}_{=1}$ ✓

Ind.-Schluss $q \to q+1$:

$$f(a^{q+1}) = f(a \cdot a^q) \leq b \cdot f(a^q) + c \cdot n$$

$$\overset{\text{IV}}{\leq} b \cdot \left(\frac{c}{a} \cdot a^q \cdot \sum_{i=0}^{q} \left(\frac{b}{a}\right)^i \right) + c \cdot a^q$$

$$= \frac{c}{a} \cdot a^{q+1} \cdot \frac{b}{a} \cdot \sum_{i=0}^{q} \left(\frac{b}{a}\right)^i + \frac{c}{a} \cdot a^{q+1}$$

$$= \frac{c}{a} \cdot a^{q+1} \cdot \sum_{i=1}^{q+1} \left(\frac{b}{a}\right)^i + \frac{c}{a} \cdot a^{q+1} \cdot 1$$

$$= \frac{c}{a} \cdot a^{q+1} \cdot \sum_{i=0}^{q+1} \left(\frac{b}{a}\right)^i$$

□

Also gilt: $f(n) \leq \frac{c}{a} \cdot n \cdot \sum_{i=0}^{q} \left(\frac{b}{a}\right)^i$

Betrachte 3 Fälle:

(i) $a > b \Rightarrow \sum_{i=0}^{q} \left(\frac{b}{a}\right)^i \leq \sum_{i=0}^{\infty} \left(\frac{b}{a}\right)^i = \frac{1}{1 - \frac{b}{a}} = \frac{a}{a-b}$ (↑ geom. Reihe)

$$\Rightarrow f(n) \leq \underbrace{\frac{c}{a} \cdot \frac{a}{a-b}}_{\text{konstant}} \cdot n \in O(n)$$

(ii) $a = b \Rightarrow f(n) \leq \frac{c}{a} \cdot n \cdot (q+1)$

$$= \frac{c}{a} \cdot n \left(\log_a n + 1\right)$$

$$= \frac{c}{a} \cdot n \left(\frac{1}{\log a} \cdot \log n + 1 \right)$$

$$\in O(n \log n)$$

□

(iii) $a < b$: $f(n) \leq \frac{c}{a} \cdot a^q \cdot \sum_{i=0}^{q} \left(\frac{b}{a}\right)^i$

$$= \frac{c}{a} \cdot \sum_{i=0}^{q} b^i \cdot a^{q-i} = \frac{c}{a} \cdot \sum_{i=0}^{q} b^{q-i} \cdot a^i$$

$$= \frac{c}{a} \cdot b^q \cdot \underbrace{\sum_{i=0}^{q} \left(\frac{a}{b}\right)^i}_{\leq \frac{b}{b-a}}$$

$$\leq \frac{c}{a} \cdot \frac{b}{b-a} \cdot b^q$$

Einschub:

$$b^q = b^{\log_a n} = \left(a^{\log_a b}\right)^{\log_a n} = \left(a^{\log_a n}\right)^{\log_a b} = n^{\log_a b}$$

$$= \underbrace{\frac{c}{a} \cdot \frac{b}{b-a}}_{\text{konstant}} \cdot n^{\log_a b} \in O(n^{\log_a b})$$

Verallgemeinerung:

Satz (Allgemeiner AB-Satz): Seien $a > 0$, $b > 0$ und

$$f(a \cdot n) = b \cdot f(n) + g(n)$$

Dann hat f das folgende asymptotische Wachstumsverhalten:

(i) Ist $g(n) \in O(n^{\log_a b - \varepsilon})$ für ein $\varepsilon > 0$, so ist $f(n) \in \Theta(n^{\log_a b})$.

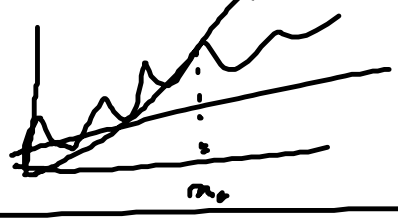
(ii) Ist $g(n) \in \Theta(n^{\log_a b})$, so ist

$f(n) \in \Theta(n^{\log_a b} \cdot \log n)$

(iii) Ist $g(n) \in \Omega(n^{\log_a b + \varepsilon})$ für ein $\varepsilon > 0$ und
ist $b \cdot g(n) \leq c \cdot g(n)$ für ein $c < 1$ und alle $n \geq n_0$ (für mod.)
so ist $f(n) \in \Theta(g(n))$

ohne Beweis. ☒

Hinweis: Hatten $f, g: \mathbb{N} \to \mathbb{N}$ definiert
Regel von l'Hôpital gilt für $f, g: \mathbb{R} \to \mathbb{R}$



Beispiele zum AB-Satz:

1) Max. im einem Array mittels Aufteilung- und
Beschleunigung finden:

$C(2) = 1$
$C(2n) = 2 \cdot C(n) + 1$

da $a = b = 2$

spezieller AB-Satz: $C(n) \in \Theta(n \log n)$

allgemeiner AB-Satz:

$n^{\log_a b} \overset{a=b=2}{=\!=} n$

$g(n) = 1 \in O\left(n^{\log_a b - \frac{1}{2}}\right)$

$\overset{\text{1. Fall}}{\Longrightarrow} f(n) \in \Theta\left(n^{\log_a b}\right) = \Theta(n)$

2) $f(4 \cdot n) = 3 \cdot f(n) + \underbrace{n \log n}_{g(n)}$

a ↑ b ↑

$g(n)$ nicht linear
$\Rightarrow$ spezieller AB-Satz macht keine Aussage

$n^{\log_a b} = n^{\log_4 3} = n^{0.793\ldots}$

$g(n) = n \log n \in \Omega(n) \subseteq \Omega\left(n^{0.793\ldots + \varepsilon}\right)$

für $\varepsilon = 0.2$

Regularitätsbedingung an g:

$b \cdot g\left(\frac{n}{a}\right) = 3 \cdot \frac{n}{4} \cdot \log \frac{n}{4} < \frac{3}{4} n \log n$

$\leq c \cdot n \log n$ für $c = 0.9$

↓

erfüllt $\Rightarrow f(n) \in \Theta(g(n)) = \Theta(n \log n)$.

3) konkrete Anwendung: Multiplikation großer Dualzahlen

x, y seien n-stellige Dualzahlen (n sei 2er Potenz)

<u>Problem</u>: berechne $x \cdot y$

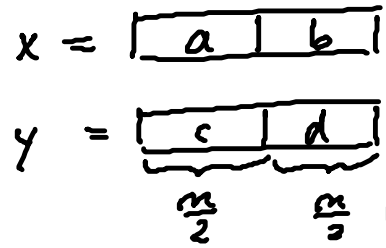
naiver Algorithmus „schriftliche Multiplikation"

$\Theta(n^2)$

$89 \cdot 23$

$= 1780$
$\quad 267$
$\overline{2047}$

besserer Algorithmus:



$x = a \cdot 2^{\frac{n}{2}} + b$

$y = c \cdot 2^{n/2} + d$

Dann ist

$$x \cdot y = (a \cdot 2^{n/2} + b) \cdot (c \cdot 2^{n/2} + d)$$
$$= a \cdot c \cdot 2^n + (a \cdot d + bc) \cdot 2^{n/2} + b \cdot d$$

Damit haben wir das Problem, zwei n-stellige Zahlen zu multiplizieren auf das Problem zurückgeführt 4 $\frac{n}{2}$-stellige Zahlen zu multiplizieren

+ Shifts und Additionen
$\rightarrow \mathcal{O}(n)$

$\Rightarrow$ Laufzeitfunktion T:

$$T(n) = 4 \cdot T\left(\frac{n}{2}\right) + c \cdot n$$

sp. AB-Satz (3. Fall) $\Rightarrow$ $T(n) \in \mathcal{O}(n^{\log_2 4}) = \mathcal{O}(n^2)$

(kein Vorteil gegenüber der naiven Multiplikation)

Idee: Versuche $b=4$ zu reduzieren auf $b=2$