

Aufwand

- ▷ begrenzender Schlüsselfaktor
- ▷ es macht kaum einen Unterschied, ob eine Berechnung zu lange dauert oder gar nicht endet
- ▷ gesucht sind Methoden zur Aufwandsermittlung und -analyse
- ▷ Versuch in **CE-S**:
Gleichungsanwendungen brauchen Zeit

CE-S-Ansatz zur Aufwandsermittlung

Zeitaufwand als maximale Zahl von Gleichungsanwendungen, die nötig sind, um eine Operation für bestimmte Argumente auszuwerten in Abhängigkeit von der Länge der eingegebenen Zeichenketten

in Zeichen: $T^{OP}(n)$

bzw. $T^{OP}(n_1, \dots, n_k)$

↓
Zahl der
Zeichenkettensargumente

Vorgehensweise (1)

- ▷ Auswirkung der spezifizierten Gleichungen auf die Aufwandfunktion

z.B.

$$\text{insert}(x, L) = x \Rightarrow T_{\text{insert}}(0) = 1$$

$$\text{insert}(x, yv) = \begin{cases} xyv & \text{if } x \leq y \\ y \text{ insert}(x, v) & \text{else} \end{cases}$$

$$\Rightarrow T_{\text{insert}}(n+1) \stackrel{?}{=} 1 + \begin{cases} 0 & \text{falls } x \leq y \\ T_{\text{insert}}(n) & \text{sonst} \end{cases}$$
$$\stackrel{\uparrow}{=} 1 + T_{\text{insert}}(n)$$

schlechteste Fall

Vorgehensweise (2)

- ▷ Aufstellen einer nichtrekursiven Aufwandshypothese (Beweis meist durch vollständige Induktion)

z.B.

$$T_{\text{insert}}(n) = n + 1 \quad \text{für alle } n \in \mathbb{N}$$

- ▷ statt einer exakten Aufwandsaussage kann auch nach oben abgeschätzt werden (was oft einfacher ist)

Vorgehensweise (3)

- ▷ beim Beweis von Aufwandsabschätzungen werden oft Eigenschaften der Operationen benötigt (Beweis mit Hilfe von Gleichwertigkeit oft mit vollständiger Induktion über Wörter)

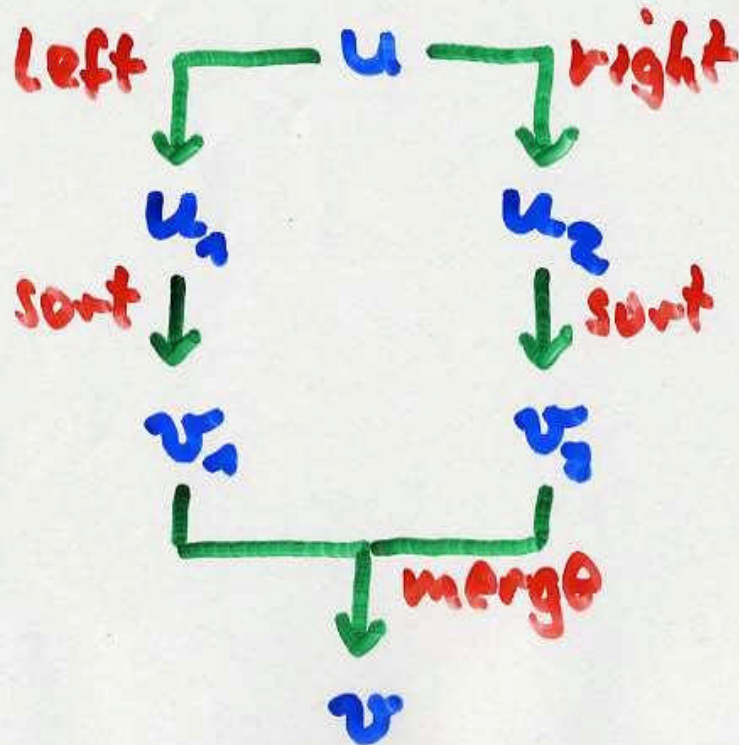
z.B. $T_{\text{sort}}(n) = \frac{(n+2)(n+1)}{2}$

falls $T_{\text{insort}}(n) = n+1$

und $\text{length}(\text{sort}(u)) = \text{length}(u)$

Beispiel: Sortieren durch Mischen

Idee: sortiere linke und rechte Hälfte eines Wortes und mische die sortierten Hälften zusammen



Sortieren durch Mischen

mergesort

ops: $\text{sort} : A^* \rightarrow A^*$; $\text{left}, \text{right} : A^* \rightarrow A^*$; $\text{merge} : A^* \times A^* \rightarrow A^*$

vars:

eqns: $\text{sort}(\perp) = \perp$

$\text{sort}(x) = x$

$\text{sort}(u) = \text{merge}(\text{sort}(\text{left}(u)), \text{sort}(\text{right}(u)))$

falls $\text{length}(u) \geq 2$

Aufwandsermittlung (für Sortieren durch Mischen)

(1) was die spezifizierten Gleichungen sagen:

$$T_{\text{sort}}(0) = 1$$

$$T_{\text{sort}}(1) = 1$$

$$T_{\text{sort}}(n) = 1 + T_{\text{merge}}(\bar{n}_1, \bar{n}_2) + T_{\text{left}}(n) + T_{\text{right}}(n) \\ + T_{\text{sort}}(n_1) + T_{\text{sort}}(n_2)$$

$\text{length}(\text{sort}(\text{left}(u)))$

$\text{length}(\text{sort}(\text{right}(u)))$

$\text{length}(\text{left}(u))$

$\text{length}(\text{right}(u))$

Sortieren durch Mischen

mergesort

ops: $\text{sort}: A^* \rightarrow A^*$; $\text{left}, \text{right}: A^* \rightarrow A^*$; $\text{merge}: A^* \times A^* \rightarrow A^*$

vars: $x, y \in A$; $u, v \in A^*$

eqns: $\text{sort}(\perp) = \perp$

$\text{sort}(x) = x$

$\text{sort}(u) = \text{merge}(\text{sort}(\text{left}(u)), \text{sort}(\text{right}(u)))$

falls $\text{length}(u) \geq 2$

$\text{left}(\perp) = \perp$

$\text{right}(\perp) = \perp$

$\text{left}(x) = x$

$\text{right}(x) = \perp$

$\text{left}(xuy) = x \text{ left}(u)$

$\text{right}(xuy) = \text{right}(u)y$

$\text{merge}(\perp, v) = v$

$\text{merge}(u, \perp) = u$

$\text{merge}(xu, yv) = \text{if } x \leq y \text{ then } x \text{ merge}(u, yv)$
else $y \text{ merge}(xu, v)$

Aufwandsermittlung (für Sortieren durch Mischen)

(1) was die spezifizierten Gleichungen sagen:

$$T^{\text{sort}}(0) = 1$$

$$T^{\text{sort}}(1) = 1$$

$$T^{\text{sort}}(n) = 1 + T^{\text{merge}}(\bar{n}_1, \bar{n}_2) + T^{\text{left}}(n) + T^{\text{right}}(n) \\ + T^{\text{sort}}(n_1) + T^{\text{sort}}(n_2)$$

$$\begin{array}{c} \text{length}(\text{sort}(\text{left}(u))) = \text{length}(\text{left}(u)) \\ \text{length}(\text{sort}(\text{right}(u))) = \text{length}(\text{right}(u)) \end{array}$$

(3) Eigenschaften der Operationen ausnutzen (Beob. 3 in Kap. 3)

$$T^{\text{sort}}(2m) = 1 + T^{\text{merge}}(m, m) + T^{\text{left}}(n) + T^{\text{right}}(n) + 2 \cdot T^{\text{sort}}(m)$$

$$\begin{aligned} \text{1. Fall: } n &= 1 + 2m + m + 1 + m + 1 + 2 \cdot T^{\text{sort}}(m) \\ &= 3 + 4m + 2T^{\text{sort}}(m) \quad (\text{vgl. Bsp. 4.4 im Skript}) \end{aligned}$$

$$T^{\text{sort}}(2m+1) = 1 + T^{\text{merge}}(m+1, m) + T^{\text{left}}(n) + T^{\text{right}}(n) + T^{\text{sort}}(m+1) + T^{\text{sort}}(m)$$

$$\begin{aligned} \text{2. Fall: } n &= 1 + 2m+1 + m+1 + m+1 + T^{\text{sort}}(m+1) + T^{\text{sort}}(m) \\ &= 4 + 4m + T^{\text{sort}}(m+1) + T^{\text{sort}}(m) \quad (\text{vgl. 4.4}) \end{aligned}$$

Aufwandsermittlung (fortgesetzt)

(2) nichtrekursive Abschätzung

Vermutung: $T^{\text{sort}}(n) \leq 5 \cdot n \cdot k$ mit $2^{k-1} < n \leq 2^k$ ($k \geq 1$)

Beweis durch vollständige Induktion über k :

IA: $k=1$, d.h. $n=2$ und somit nach Fall 1:

$$T^{\text{sort}}(2) = 3 + 4 \cdot 1 + 2 \cdot 1 = 9 \leq 5 \cdot 2 \cdot 1$$

IS: $k+1$, d.h. $2^k < n \leq 2^{k+1}$

2. Fall: $n = 2m+1$, d.h. $2^{k-1} \leq m < m+1 \leq 2^k$ und somit

$$T^{\text{sort}}(2m+1) = 4 + 4m + T^{\text{sort}}(m+1) + T^{\text{sort}}(m)$$

$$\stackrel{\text{IV}}{\leq} 4 + 4m + 5 \cdot (m+1) \cdot k + 5 \cdot m \cdot k$$

$$= 4 + 4m + 5(2m+1)k$$

$$\leq 5(2m+1) + 5(2m+1)k = 5(2m+1)(k+1)$$

1. Fall analog (vgl. 4.4. im Skript)

unendlicher Aufwand

sort₀

ops: $\text{sort}: A^* \rightarrow A^*$

vars: $x, y \in A; u, v \in A^*; w \in A^*$

eqs: $\text{sort}(uxyv) = \text{sort}(uyxv)$

$\text{sort}(w) = w$ falls is-sorted(w)

Aufwand gemäß 1. Gleichung:

$$T^{\text{sort}}(n) = 1 + T^{\text{sort}}(n) \text{ für } n \geq 2$$

aber: diese Gleichung besitzt in $\mathbb{N}$
keine Lösung

Das große "O"

Aus $f(n) \leq c \cdot g(n) + c_0$ für alle $n \geq n_0$

wird kurz: $f \in O(g)$

z.B. $T_{\text{insertsort}}, T_{\text{qsort}} \in O(n^2)$
 $T_{\text{mergesort}} \in O(n \lceil \log n \rceil)$

¹
kleinste nat. Zahl oberhalb
des Logarithmus dualis