

Stefan Fischer
Benjamin
Merle Kammer
04.05.2017

1	2	3	Σ

Aufgabe 1

a)

Zu Beginn wird die Glühbirne aus Stockwerk 1 geworfen. Geht sie nicht kaputt, so gehen wir zu Stockwerk 2 und werfen sie von dort aus dem Fenster. Wir gehen solange die Stockwerke nach oben (lineare Suche) bis die Glühbirne zerbricht. Auf Stockwerk 168 enden wir aus folgenden Gründen:

a) geht die Glühbirne kaputt, so wurde das Stockwerk k gefunden

b) geht sie nicht kaputt, dann muss das Stockwerk 169 das gesuchte Stockwerk sein (denn $1 \leq k \leq 169$)

b)

"Lineare Suche", nur dass die Glühbirne aus jedem zweiten Stockwerk geworfen wird. Geht sie ab einem Stockwerk kaputt, so wird nur noch das Stockwerk darunter getestet und dann ist k bestimmt.

c)

D. h. max. 25 Versuche

Idee: Glühbirne wird zu Beginn aus Stockwerk 25 geworfen. Geht sie kaputt, so wird anschließend die lineare Suche wie in a) durchgeführt (von Stockwerk 1-24). Geht die Glühbirne nicht kaputt, so gehen wir 24 Stockwerke weiter und werfen die Glühbirne raus. Geht sie kaputt, dann lineare Suche von Stockwerk 26-48. Geht sie nicht kaputt, 23 Stockwerke weitergehen usw.

$$25^{+24} \rightarrow 49^{+23} \rightarrow 72^{+22} \rightarrow 94^{+21} \rightarrow 115^{+20} \rightarrow 135^{+19} \rightarrow 154 \dots$$

d)

mit Glück: minimal 1 Versuch (Glühbirne wird aus erstem Stock geworfen und geht kaputt)

ohne Glück: um die minimale Anzahl rauszufinden, die man ohne Glück braucht, kann man die gaußsche Summenformel verwenden. Sie lautet $\frac{n \cdot (n+1)}{2}$ wobei n für die Anzahl der Versuche steht. Um in unserem Beispiel das Minimum herauszufinden, stellen wir folgende Ungleichung auf $\frac{n \cdot (n+1)}{2} > 169$. Probiert man jetzt einige Werte aus, erkennt man, dass für 18 Versuche die Ungleichung mit $171 > 169$ noch erfüllt wäre. Bei 17 Versuchen ergibt sich die Gleichung $153 > 169$, welche schon nicht mehr erfüllt ist. Somit ist bei 18 Versuchen das Minimum erreicht. Dies könnte man nun mit der Methode aus Aufgabe c) noch einmal Schritt für Schritt nachvollziehen.

e)

- a) für allgemeines n höchstens $n-1$ Versuche
 b) für allgemeines n $\lceil \frac{n}{2} \rceil$ Versuche
 c)
 d) immer 1 oder $\frac{n \cdot (n+1)}{2} > 169$

Aufgabe 2

Aufgabe 3

Laufzeit von Algorithmus A: $T_A(n) = (\log_8 n + 1) \cdot n$ mit $T_A(1) = 1$

Laufzeit von Algorithmus B: $T_B(n) = 8T_B(\frac{n}{8}) + n^\alpha$ mit $\alpha \in \mathbb{R}_+$

a)

Annahmen: n ist eine Achterpotenz und $T_B(1) = 1$

$$\begin{aligned}
 T_B(n) &= 8T_B\left(\frac{n}{8}\right) + n^\alpha \\
 &= 8\left(8T_B\left(\frac{n}{64}\right) + \left(\frac{n}{8}\right)^\alpha\right) + n^\alpha \\
 &= 8\left(8\left(8T_B\left(\frac{n}{512}\right) + \left(\frac{n}{64}\right)^\alpha\right) + \left(\frac{n}{8}\right)^\alpha\right) + n^\alpha \\
 &\vdots \\
 &= 8^i \cdot T_B\left(\frac{n}{8^i}\right) + \sum_{k=0}^{i-1} \left(8^k \cdot \left(\frac{n}{8^k}\right)^\alpha\right)
 \end{aligned}$$

für $i = \log_8 n$ gilt:

$$\begin{aligned}
 &= 8^{\log_8 n} \cdot T_B\left(\frac{n}{8^{\log_8 n}}\right) + \sum_{k=0}^{(\log_8 n)-1} \left(8^k \cdot \left(\frac{n}{8^k}\right)^\alpha\right) \\
 &= n \cdot T_B(1) + \sum_{k=0}^{(\log_8 n)-1} \left(8^k \cdot \left(\frac{n}{8^k}\right)^\alpha\right)
 \end{aligned}$$

Da: $n = 8^x$ angenommen werden soll, gilt:

$$\begin{aligned}
 &= 8^x + \sum_{k=0}^{(\log_8 8^x)-1} \left(8^k \cdot \left(\frac{8^x}{8^k}\right)^\alpha\right) \\
 &= 8^x + \sum_{k=0}^{x-1} \left(8^k \cdot \left(\frac{8^x}{8^k}\right)^\alpha\right)
 \end{aligned}$$

Behauptung: $T_B(n) = 8^x + \sum_{k=0}^{x-1} \left(8^k \cdot \left(\frac{8^x}{8^k}\right)^\alpha\right)$

Beweis durch vollständige Induktion:

Induktionsanfang: $T(1) = 1$, denn $8^0 + \sum_{k=0}^{0-1} \left(8^k \cdot \left(\frac{8^0}{8^k}\right)^\alpha\right) = 1 + \sum_{k=0}^{-1} \left(8^k \cdot \left(\frac{1}{8^k}\right)^\alpha\right) = 1 \quad \checkmark$

Induktionsvoraussetzung: Sei $n = 8^x$ mit $x \in \mathbb{N}_0$ und es gelte $T(8^x) = 8^x + \sum_{k=0}^{x-1} \left(8^k \cdot \left(\frac{8^x}{8^k}\right)^\alpha\right)$

Induktionsbehauptung: Es gilt $T(8^{x+1}) = 8^{x+1} + \sum_{k=0}^{(x+1)-1} \left(8^k \cdot \left(\frac{8^{x+1}}{8^k}\right)^\alpha\right)$

Beweis:

$$\begin{aligned}
 T(8^{x+1}) &= 8T_B\left(\frac{8^{x+1}}{8}\right) + (8^{x+1})^\alpha \\
 &= 8T_B(8^x) + (8^{x+1})^\alpha \\
 &= 8T_B(8^x) + (8^{x+1})^\alpha \\
 &= 8\left(8^x + \sum_{k=0}^{x-1} \left(8^k \cdot \left(\frac{8^x}{8^k}\right)^\alpha\right)\right) + (8^{x+1})^\alpha \\
 &= 8 \cdot 8^x + 8 \cdot \sum_{k=0}^{x-1} \left(8^k \cdot \left(\frac{8^x}{8^k}\right)^\alpha\right) + (8^{x+1})^\alpha \\
 &= 8^{x+1} + \sum_{k=0}^{x-1} \left(8^{k+1} \cdot \left(\frac{8^x}{8^k}\right)^\alpha\right) + (8^{x+1})^\alpha \\
 &= 8^{x+1} + \sum_{k=1}^x \left(8^{k+1-1} \cdot \left(\frac{8^x}{8^{k-1}}\right)^\alpha\right) + (8^{x+1})^\alpha \\
 &= 8^{x+1} + \sum_{k=1}^x \left(8^k \cdot \left(\frac{8^x \cdot 8^1}{8^k}\right)^\alpha\right) + (8^{x+1})^\alpha \\
 &= 8^{x+1} + \sum_{k=1}^x \left(8^k \cdot \left(\frac{8^{x+1}}{8^k}\right)^\alpha\right) + (8^{x+1})^\alpha
 \end{aligned}$$

Es gilt: $(8^{x+1})^\alpha = (8^0 \cdot \left(\frac{8^{x+1}}{8^0}\right)^\alpha)$

Daher ziehen wir $(8^{x+1})^\alpha$ in die Summe und verringern die Startvariabel um Eins:

$$= 8^{x+1} + \sum_{k=0}^x \left(8^k \cdot \left(\frac{8^{x+1}}{8^k}\right)^\alpha\right) \quad \square$$

Somit ist bewiesen, dass $T_B(n) = 8T_B\left(\frac{n}{8}\right) + n^\alpha = n + \sum_{k=0}^{(\log_8 n)-1} \left(8^k \cdot \left(\frac{n}{8^k}\right)^\alpha\right)$ mit $\alpha \in \mathbb{R}_+$ gilt.

b)

$$T_A(n) = (\log_8 n + 1) \cdot n \text{ mit } T_A(1) = 1 \text{ und } T_B(n) = n + \sum_{k=0}^{(\log_8 n)-1} \left(8^k \cdot \left(\frac{n}{8^k}\right)^\alpha\right)$$

Algorithmus A ist schneller als B $\forall \alpha > 1$.

Beweis:

Fall 1: $\alpha = 1$

$$\begin{aligned}
T_B(n) &= n + \sum_{k=0}^{(\log_8 n)-1} (8^k \cdot (\frac{n}{8^k})^1) \\
&= n + \sum_{k=0}^{(\log_8 n)-1} (8^k \cdot \frac{n^1}{8^k \cdot 1}) \\
&= n + \sum_{k=0}^{(\log_8 n)-1} n \\
&= n + ((\log_8 n - 1) - 0 + 1)n \\
&= n + n \cdot \log_8 n = T_A(n)
\end{aligned}$$

Für $\alpha = 1$ sind die Algorithmen A und B somit gleich schnell. Wir überprüfen daher im Folgenden die Fälle $\alpha < 1$ und $\alpha > 1$ um zu bestimmen, wann der Algorithmus A schneller als B ist.

Wir verwenden dabei das Mastertheorem um die Laufzeit von $T_B(n)$ in Abhängigkeit von α zu bestimmen: $T_B(n) = 8T_B(\frac{n}{8}) + n^\alpha \Rightarrow a = 8, b = 8, f(n) = n^\alpha$ und $\log_b a = \log_8 8 = 1$

Fall 2: $\alpha < 1$

MT Fall 1

$f(n^\alpha) = O(n^{1-\epsilon}) \xRightarrow{\text{MT Fall 1}} T(n) = O(n^1)$ d.h., dass $T_B(n) \in O(n)$

$T_A(n) = n \cdot \log_8 n + n \notin O(n) \Rightarrow T_A(n) > T_B(n)$ d.h., dass Algorithmus B für $\alpha < 1$ schneller läuft.

Fall 3: $\alpha > 1$

$f(n^\alpha) = O(n^{1+\epsilon})$ also Fall 3 des Mastertheorems. Wir überprüfen daher die folgende Bedingung:

$a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$ für $c < 1$ und n genügend groß:

$$8 \cdot f(\frac{n}{8}) \leq c \cdot f(n) \Leftrightarrow 8 \frac{n^\alpha}{8^\alpha} \leq c \cdot n^\alpha \Leftrightarrow \frac{8^1 \cdot n^\alpha}{8^\alpha} \leq c \cdot n^\alpha \Leftrightarrow \frac{n^\alpha}{8^{\alpha-1}} \leq c \cdot n^\alpha \text{ für } \frac{1}{8^{\alpha-1}} < c < 1$$

MT Fall 3

$\xRightarrow{\text{MT Fall 3}} T(n) = O(n^\alpha)$ d.h. $T_B(n) \in O(n^\alpha)$ $T_A(n) = n \cdot \log_8 n + n \in O(n^\alpha)$

aber $n^\alpha \notin O(T_A(n)) \Rightarrow T_A(n) < T_B(n)$ d.h., dass Algorithmus A für $\alpha > 1$ schneller ist als Algorithmus B. \square