
Grundlagen: Algorithmen und Datenstrukturen

Abgabetermin: In der jeweiligen Tutorübung

Hausaufgabe 1

Betrachten Sie den Algorithmus 1 zur Bestimmung eines minimalen Elements in einem Integer-Array.

Algorithmus 1: Min

```
Input : int  $A[]$ , int  $n$ 
1 int  $i = 1$ 
2 int  $min = A[0]$ 
3 while  $i < n$  do
4   | if  $A[i] < min$  then
5   |   |  $min = A[i]$ 
6   |   |  $i = i + 1$ 
7 return  $min$ 
```

Bestimmen Sie die exakte Worst-Case-Laufzeit des Algorithmus, indem Sie die exakte Anzahl der Rechenoperationen, wie sie in der Vorlesung definiert wurden (Vergleiche, Zuweisungen, Additionen, ...), in Abhängigkeit der Eingabegröße berechnen (keine asymptotische Abschätzungen durch Landau-Symbole).

Berechnen Sie nun auch die Worst-Case Laufzeit mit Hilfe der Landau-Symbole.

Lösungsvorschlag

Um die Worst-Case-Laufzeit zu bestimmen, betrachten wir für die Eingabe ein Array, dessen Werte monoton fallen. Zum Beispiel könnte das i -te Feld $A[i]$ des Array genau den Wert $n - i$ beinhalten. Dann muss in jedem Schleifendurchlauf die Zeile 5 im Algorithmus ausgeführt werden.

Wir berechnen die Laufzeit in Zeitschritten: Zeile 1 und 2: jeweils ein Zeitschritt. Die While-Schleife wird $n - 1$ -mal ausgeführt. Dafür ist jeweils ein Vergleich = eine Zeiteinheit in Zeile 3 nötig. Genauso wird in jedem Schleifendurchlauf die Bedingung in Zeile 4 überprüft. Daraufhin wird min der Wert $A[i]$ zugewiesen und der Zähler i um Eins erhöht. Dies ergibt für einen Schleifendurchlauf also mindestens 3 Zeiteinheiten (man kann auch einen Vierten für einen Sprung zur While-Bedingung berechnen).

Nun wird ein letztes Mal die Bedingung in Zeile 3 ausgewertet. Da diese negativ ausfällt, springt das Programm in die letzte Zeile und gibt den Wert min zurück. Dies ergibt drei weitere Zeiteinheiten (Vergleich, Sprung, return).

Zusammen ergeben sich also: $2 + (1 + 3)(n - 1) + 3 = 5 + 4(n - 1)$ Zeiteinheiten.

Mit der asymptotischen Analyse geht dies deutlich einfacher. Die Zeilen 1,2 und 7 werden mit jeweils $\mathcal{O}(1)$ abgeschätzt. In jedem Schleifendurchlauf ergeben sich 4 Operationen in $\mathcal{O}(1)$ (Zwei mal Vergleich, eine Zuweisung und einmal Addition). Das ganze $\mathcal{O}(n)$ mal ergibt also $\mathcal{O}(n)$ Zeiteinheiten für die While-Schleife. Nun noch $\mathcal{O}(1) + \mathcal{O}(n)$ mit $\mathcal{O}(n)$ abschätzen und wir erhalten die Laufzeit des Algorithmus.

Hausaufgabe 2

In welcher Relation bezüglich der Landau-Symbole stehen die Funktionen $\text{ld } n = \log_2 n$ und $\ln n = \log_e n$? Beweisen Sie ihre Aussage.

Lösungsvorschlag

Behauptung: $\ln n \in \Theta(\text{ld } n)$. Zu zeigen ist also:

$$1. \ln n \in \mathcal{O}(\text{ld } n) \Leftrightarrow \exists c > 0 : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : \ln n \leq c \text{ld } n$$

Wir wählen $c = 1$ und $n_0 = 1$: Somit gilt:

$$\ln n \leq \text{ld } n \Leftrightarrow e^{\ln n} = n \leq 2^{\text{ld } n} \leq e^{\text{ld } n}$$

Somit gilt $\ln n \in \mathcal{O}(\text{ld } n)$.

$$2. \ln n \in \Omega(\text{ld } n) \Leftrightarrow \text{ld } n \in \mathcal{O}(\ln n)$$

Wir zeigen mit $c = \text{ld } e$ und $n_0 = 1$: $\text{ld } n \leq c \ln n = (\text{ld } e) \frac{\text{ld } n}{\text{ld } e} = \text{ld } n$. Dies ist sicherlich für alle $n \geq 1$ erfüllt. Dies wiederum impliziert $\ln n \in \Omega(\text{ld } n)$ und somit gilt $\text{ld } n \in \Theta(\ln n)$.

Aufgabe 1

Beweisen Sie folgende Aussagen durch Induktion für alle $n \in \mathbb{N} := \{1, 2, 3, 4, 5, \dots\}$:

$$1. \sum_{i=0}^n i^3 = \frac{n^2(n+1)^2}{4},$$

$$2. F_{n+1}F_{n-1} - F_n^2 = (-1)^n,$$

$$3. 19 | (5 \cdot 2^{3n+1} + 3^{3n+2}) \quad (\text{In Worten: } 19 \text{ teilt } (5 \cdot 2^{3n+1} + 3^{3n+2})),$$

wobei F_n die n-te Fibonaccizahl nach der rekursiven Definition $F_n = F_{n-1} + F_{n-2}$ mit den Anfangswerten $F_0 = 0$ und $F_1 = 1$ ist.

Lösungsvorschlag

$$1. \text{ Induktionsanfang: } n = 1: \sum_{i=0}^1 i^3 = 1^3 = 1 = \frac{1 \cdot 4}{4} = \frac{1^2(1+1)^2}{4} \quad \checkmark$$

Induktionsannahme (IA): $\sum_{i=0}^n i^3 = \frac{n^2(n+1)^2}{4}$ gelte für ein beliebiges, aber festes n .

Induktionsschluss: $n \rightarrow n + 1$

$$\begin{aligned} \sum_{i=0}^{n+1} i^3 &= \sum_{i=0}^n i^3 + (n+1)^3 \stackrel{IA}{=} \frac{n^2(n+1)^2}{4} + \frac{4(n+1)^3}{4} = \frac{(n+1)^2(4(n+1) + n^2)}{4} \\ &= \frac{(n+1)^2(((n+1) + 1)^2)}{4} \end{aligned}$$

□

2. Induktionsanfang: $n = 1: F_2 F_0 - F_1^2 = 1 \cdot 0 - (1)^2 = -1 = (-1)^1$

Induktionsannahme (IA): $F_{n+1} F_{n-1} - F_n^2 = (-1)^n$ gelte für ein beliebiges, aber festes n .

Induktionsschluss: $n \rightarrow n + 1$

$$\begin{aligned} F_{n+2} F_n - F_{n+1}^2 &\stackrel{\text{Fib.def.}}{=} (F_{n+1} + F_n) F_n - (F_n + F_{n-1}) F_{n+1} = F_{n+1} F_n + F_n^2 - F_n F_{n+1} - F_{n-1} F_{n+1} \\ &= F_n^2 - F_{n-1} F_{n+1} = -1(-F_n^2 + F_{n-1} F_{n+1}) \stackrel{IA}{=} -1(-1)^n = (-1)^{n+1} \end{aligned}$$

□

3. Induktionsanfang: $n = 0: (5 \cdot 2^{3 \cdot 0 + 1} + 3^{3 \cdot 0 + 2}) = 5 \cdot 2^1 + 3^2 = 10 + 9 = 19$ und 19 teilt offensichtlich 19 ✓

Bemerkung: Wir zeigen hier der aufgrund der deutlich vereinfachten Rechnung, dass $19 | (5 \cdot 2^{3n+1} + 3^{3n+2})$ für alle $n \in \mathbb{N}_0$ gilt. Dies beinhaltet natürlich die geforderte Aussage der Aufgabenstellung.

Induktionsannahme (IA): $19 | (5 \cdot 2^{3n+1} + 3^{3n+2})$ gelte für ein beliebiges, aber festes n .

Induktionsschritt: $n \rightarrow n + 1$

$$\begin{aligned} (5 \cdot 2^{3(n+1)+1} + 3^{3(n+1)+2}) &= 3^{3n+3+2} + 5 \cdot 2^{3n+3+1} \\ &= 27(3^{3n+2}) + 27(5 \cdot 2^{3n+1}) - 27(5 \cdot 2^{3n+1}) + 8(5 \cdot 2^{3n+1}) \\ &= 27(5 \cdot 2^{3n+1} + 3^{3n+2}) - 19(5 \cdot 2^{3n+1}) \stackrel{IA}{=} 27 \cdot 19x - 19(5 \cdot 2^{3n+1}) \\ &= 19(27x - (5 \cdot 2^{3n+1})) \quad (1) \end{aligned}$$

Somit ist gezeigt das Formel (1) ein Vielfaches von 19 ist. Daher teilt also 19 auch $(5 \cdot 2^{3(n+1)+1} + 3^{3(n+1)+2})$.

□

Aufgabe 2

Ordnen Sie die Funktionen $\sqrt{\text{ld}(n+1)}$, n^2 , $(\text{ld } n)^{\text{ld } n}$, 2^n , n , 10^{100} , \sqrt{n} , $n^{\text{ld } n}$, $\text{ld } \sqrt{n+1}$, $n \log n$ nach Ihrem Wachstum.

Lösungsvorschlag

Wir zeigen zuerst, dass die Funktionen $n^{\text{ld } n}$ und $(\text{ld } n)^{\text{ld } n}$ identisch sind.

$$\text{ld}(n^{\text{ld } n}) = (\text{ld } \text{ld } n) \text{ld } n = \text{ld}((\text{ld } n)^{\text{ld } n})$$

Daher folgt direkt $n^{\text{ld } n} \in \Theta((\text{ld } n)^{\text{ld } n})$.

Wir zeigen $10^{100} \in o(\sqrt{\text{ld}(n+1)})$. Wir dürfen uns in Abhängigkeit von c ein n_0 wählen. Wählen wir $n_0 \geq 2^{\frac{10^{200}}{c^2}} - 1$ (Wie kommt man darauf? Einfach die Ungleichung

$10^{100} \leq c\sqrt{\text{ld}(n+1)}$ nach n auflösen), so ergibt sich für ein beliebiges $c > 0$ die zu überprüfende Aussage: $10^{100} \leq c\sqrt{\text{ld} 2^{\frac{10^{200}}{c^2}}} = c\sqrt{\frac{10^{200}}{c^2}} = c\frac{10^{100}}{c} = 10^{100}$.

Behauptung: $\sqrt{\text{ld}(n+1)} \in o(\text{ld} \sqrt{n+1})$ da:

$$\lim_{n \rightarrow \infty} \frac{\sqrt{\text{ld}(n+1)}}{\text{ld} \sqrt{n+1}} = \lim_{n \rightarrow \infty} \frac{(\text{ld}(n+1))^{\frac{1}{2}}}{\text{ld}(n+1)^{\frac{1}{2}}} = \lim_{n \rightarrow \infty} \frac{(\text{ld}(n+1))^{\frac{1}{2}}}{\frac{1}{2}((\text{ld} n + 1))^2} = \lim_{n \rightarrow \infty} \frac{2}{(\text{ld}(n+1))^{\frac{1}{2}}}$$

Da $\text{ld} n$ eine streng monoton steigende Funktion ist, gilt also $\lim_{n \rightarrow \infty} \frac{\sqrt{\text{ld} n + 1}}{\text{ld} \sqrt{n+1}} \rightarrow 0$. Damit folgt die Behauptung.

Als nächstes zeigen wir $\text{ld} \sqrt{n+1} \in o(\sqrt{n})$:

$$\lim_{n \rightarrow \infty} \frac{\text{ld}(n+1)}{2\sqrt{n}} \stackrel{\nu H.}{=} \lim_{n \rightarrow \infty} \frac{\sqrt{n}}{(\ln 2)(n+1)} < \lim_{n \rightarrow \infty} \frac{\sqrt{n}}{(\ln 2)(\sqrt{n}\sqrt{n+1})} \rightarrow 0$$

Dabei kann die Anwendung von l'Hopital auch als Anwendung der in der Vorlesung vorgestellten Regel $f'(n) \in o(g'(n)) \Rightarrow f(n) \in o(g(n))$ zu verstanden werden.

Die Aussage $\sqrt{n} \in o(n)$ ist schnell gezeigt: $\lim_{n \rightarrow \infty} \frac{\sqrt{n}}{\sqrt{n}\sqrt{n}} = \lim_{n \rightarrow \infty} \frac{1}{\sqrt{n}} = 0$ gilt, da \sqrt{n} eine streng monoton steigende Funktion ist.

Ebenfalls ergibt sich $n \in o(n \log n)$ und $n \log n \in o(n^2)$ durch Kürzen, l'Hôpital und Ausnutzen der Monotonie.

Wir zeigen, dass $\lim_{n \rightarrow \infty} \frac{1}{n^{(\text{ld} \text{ld} n) - 2}} \rightarrow 0$, was $n^2 \in o(n^{\text{ld} \text{ld} n})$ impliziert. Dies ist erfüllt, da $\text{ld} \text{ld} n$ streng monoton steigend ist und für alle $n > 16$ größer als 2 ist.

Bleibt abschliessend noch zu zeigen: $n^{\text{ld} \text{ld} n} \in o(2^n)$. Zur Vereinfachung des Beweises schätzen wir zunächst $n^{\text{ld} \text{ld} n}$ mit $n^{\text{ld} n}$ nach oben ab. Des weiteren verwenden wir, dass $2^n = n^{\frac{n}{\text{ld} n}}$ gilt. Somit ist zu zeigen: $\lim_{n \rightarrow \infty} \frac{n^{\text{ld} n}}{n^{\frac{n}{\text{ld} n}}} = \lim_{n \rightarrow \infty} \frac{1}{n^{\frac{n}{\text{ld} n} - \text{ld} n}} \rightarrow 0$.

Bleibt also zu zeigen, dass $\frac{n}{\text{ld} n} - \text{ld} n > 0$ wenn $n \rightarrow \infty$ gilt, denn dann gilt $\lim_{n \rightarrow \infty} \frac{n^{\text{ld} n}}{n^{\frac{n}{\text{ld} n}}} \rightarrow 0$. Erstere Gleichung ist aber für jedes $n \geq 2^5$ erfüllt.

Aufgabe 3

Beweisen Sie die in der Vorlesung angegebene Regel:

Für alle Funktionen $f(n), g(n)$ mit $\exists n_0 \in \mathbb{N} : \forall n > n_0 : f(n), g(n) > 0$ gilt:

$$\mathcal{O}(f(n) + g(n)) = \mathcal{O}(f(n)) + \mathcal{O}(g(n))$$

wobei $\mathcal{O}(f(n)) + \mathcal{O}(g(n)) := \{h_f(n) + h_g(n) \mid h_f(n) \in \mathcal{O}(f(n)) \wedge h_g(n) \in \mathcal{O}(g(n))\}$ (Hier ist also die Gleichheit der beiden Mengen zu zeigen).

Lösungsvorschlag

$\mathcal{O}(f(n) + g(n)) \subseteq \mathcal{O}(f(n)) + \mathcal{O}(g(n))$:

Es muss gezeigt werden, dass jede beliebige Funktion $h(n) \in \mathcal{O}(f(n) + g(n))$ auch in $\mathcal{O}(f(n)) + \mathcal{O}(g(n))$ liegt, d.h., dass $h(n)$ als Summe von zwei Funktionen $h_f(n) \in \mathcal{O}(f(n))$ und $h_g(n) \in \mathcal{O}(g(n))$ dargestellt werden kann:

$$\begin{aligned} h(n) &\in \mathcal{O}(f(n) + g(n)) \\ \Leftrightarrow \exists c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0 : h(n) &\leq c \cdot (f(n) + g(n)) \\ \Rightarrow \exists c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0 : h(n) - c \cdot g(n) &\leq c \cdot f(n) \\ \Leftrightarrow \exists c > 0 (h(n) - c \cdot g(n)) &\in \mathcal{O}(f(n)) \end{aligned}$$

Für die Funktionen $h_f(n) = h(n) - c \cdot g(n)$ und $h_g(n) = c \cdot g(n)$ gilt:

- $h(n) = h_f(n) + h_g(n)$
- $h_f(n) \in \mathcal{O}(f(n))$
- $h_g(n) \in \mathcal{O}(g(n))$

D.h. $h(n) \in \mathcal{O}(f(n)) + \mathcal{O}(g(n))$

$\mathcal{O}(f(n)) + \mathcal{O}(g(n)) \subseteq \mathcal{O}(f(n) + g(n))$:

Es muss gezeigt werden, dass jede beliebige Funktion $h(n) \in \mathcal{O}(f(n)) + \mathcal{O}(g(n))$ auch in $\mathcal{O}(f(n) + g(n))$ liegt.

Dies wird gezeigt, indem aus der Darstellung $h_f(n) + h_g(n)$ für $h(n)$ (und den Eigenschaften $h_f(n) \in \mathcal{O}(f(n))$ und $h_g(n) \in \mathcal{O}(g(n))$) geeignete $c > 0$ und $n_0 \in \mathbb{N}$ hergeleitet werden:

$$\begin{aligned} & h(n) \in \mathcal{O}(f(n)) + \mathcal{O}(g(n)) \\ \Leftrightarrow & \exists h_f(n) \in \mathcal{O}(f(n)) \exists h_g(n) \in \mathcal{O}(g(n)) : h(n) = h_f(n) + h_g(n) \\ \Rightarrow & \exists c_f > 0 \exists n_f \in \mathbb{N} \forall n \geq n_f : h_f(n) \leq c_f \cdot f(n) \\ & \wedge \exists c_g > 0 \exists n_g \in \mathbb{N} \forall n \geq n_g : h_g(n) \leq c_g \cdot g(n) \\ \Rightarrow & \exists c_f, c_g > 0 \exists n_f, n_g \in \mathbb{N} \forall n \geq \max\{n_f, n_g\} : h_f(n) + h_g(n) \leq c_f \cdot f(n) + c_g \cdot g(n) \\ & \text{Wähle: } c := \max\{c_f, c_g\} > 0 \text{ und } n_0 := \max\{n_f, n_g\} \\ \Rightarrow & \forall n \geq n_0 : h_f(n) + h_g(n) \leq c \cdot (f(n) + g(n)) \\ \Rightarrow & h(n) = h_f(n) + h_g(n) \in \mathcal{O}(f(n) + g(n)). \end{aligned}$$