Übung 1: Algorithmen und Datenstrukturen

Theodor Bajusz (7159556), Valerij Dobler (7068135),
Matz Radloff (6946325), Robin Wannags (6948409)

14. November 2020

Aufgabe 1

Betrachten Sie zwei Algorithmen A und B für das gleiche Problem. Algorithmus A benötigt bei einer Eingabe der Größe n genau $2n^2$ Basisoperationen, während Algorithmus B genau $100n\lceil log_2 n\rceil$ Basisoperationen benötigt. Betrachten Sie zwei Computer C_1 und C_2. Computer C_1 (Supercomputer) kann pro Sekunde $4,16 \cdot 10^{17}$ Basisoperationen durchführen. Computer C_2 (Handy) kann hingegen nur $3 \cdot 10^{11}$ Basisoperationen pro Sekunde durchführen.

a)

Wie lange braucht Algorithmus A auf beiden Computern, um ein Problem der Größe $n_1 = 200, n2 = 2.7 \cdot 10^9$ und $n^3 = 10^{16}$ zu lösen? (Lösungen in Sekunden)

n_i	200	$2,7 \cdot 10^9$	10^{16}
C_1	ca. $1,92 \cdot 10^{-13}$	ca. 35	ca. $4,81 \cdot 10^{14}$
C_2	ca. $2,67 \cdot 10^{-7}$	$4,86 \cdot 10^7$	ca. $6,67 \cdot 10^{20}$

b)

Wie lange braucht Algorithmus B auf beiden Computern, um ein Problem der Größe $n_1 = 200, n2 = 2.7 \cdot 10^9$ und $n^3 = 10^{16}$ zu lösen?

n_i	200	$2,7 \cdot 10^9$	10^{16}
C_1	ca. $3,85 \cdot 10^{-13}$	ca. $2,08 \cdot 10^-5$	ca. 130
C_2	ca. $5,33 \cdot 10^{-7}$	$28,8$	$1,8 \cdot 10^8$

c)

Für welche Problemgrößen ist Algorithmus A schneller und für welche ist Algorithmus B schneller, wenn beide Algorithmen auf dem gleichen Computer laufen?

Algorithmus A ist für die Eingabe $n = 200$ schneller. Bei den größeren Eingaben ist jeweils Algorithmus B schneller.

Aufgabe 2

Betrachten Sie das Problem *Zweit-Kleinstes-Element*

- *Eingabe:* Ein Array $A[1, \dots, n]$ von $n \geq 2$ Zahlen
- *Ausgabe:* Ein Index i, sodass es einen Index $j \neq i$ gibt mit $A[j] \leq A[i]$ und für alle Indizes $k \in \{1, 2, \dots, n\} \setminus \{j\}$ gilt $A[k] \geq A[i]$.

a)

Beschreiben Sie in Pseudocode einen Algorithmus der das Problem *Zweit-Kleinstes-Element* löst. (3 Punkte)

Algorithm 1: ZweitKleinstesElement(A)

Result: Das zweit kleinste Element

```
1  if A[1] ≤ A[2] then
2  |   min ← 1;
3  |   i ← 2;
4  else
5  |   min ↔ i;
6  c ← 3;
7  while c <= length(A) do
8  |   if A[c] ≤ A[min] then
9  |   |   i ← min;
10 |   |   min ← c;
11 |   else if A[c] < A[i] then
12 |   |   i ← c;
13 |   else
14 |   |   NOP;
15 |   c++;
16 end
17 return i;
```

NOP: No-Operation

b)

Beweisen Sie die Korrektheit Ihres Algorithmus mit Hilfe einer geeigneten Schleifeninvariante.

Behauptung. Kandidat für $I(c, i, min)$
$A[i] = min\{\ A[j] \mid j \in \{1, 2, \dots, c-1\} \setminus min\}$

Beweis. Durch Schleifeninvariante

1. Initialisierung: O.b.d.A.: $A[1] \leq A[2] \implies i = 2, min = 1$, sonst analog
   Somit gilt nach Zeile 6:

- $I(c, i, min) = I(3, 2, 1)$
- $I(3, 2, 1):\quad ''A[2] = min\{\ A[j] \mid j \in \{1, 2, \ldots, 3-1\} \setminus 1\} = min\{\ A[2]\ \}''$

2. Erhaltung: In Iteration c soll gelten:

- Annahme: $I(c, i, min)$ nach Zeile 7
- Ziel: $I(c+1, i, min)$ nach Zeile 15
- nach Zeile 8:
  $A[c] \leq A[min] = min\{A[1], \ldots, A[c-1]\}$
  $\implies A[c] = min\{A[1], \ldots, A[c]\}$
  $\implies A[i] = min\{\ A[j] \mid j \in \{1, 2, \ldots, c\} \setminus c\}$
  $\iff I(c+1, i, min)$ schon nach Zeile 10, weil $i \leftarrow min$ und $min \leftarrow c$
- nach Zeile 11:
  $A[i] > A[c] > A[min] = min\{A[1], \ldots, A[c]\}$
  $\implies A[c] = min\{\ A[j] \mid j \in \{1, 2, \ldots, c\} \setminus min\}$
  $\implies I(c+1, i, min)$ nach Zeile 12, weil $i \leftarrow c$
- nach Zeile 13 impliziert
  $A[min] = min\{A[1], \ldots, A[c]\} \wedge A[i] = min\{\ A[j] \mid j \in \{1, 2, \ldots, c\} \setminus min\}$
  $\implies I(c+1, i, min)$
- also: $I(c+1, i, min)$ nach Zeile 14

3. Terminierung:

- Ende letzter Schleifendurchlauf:
  $c = length(A)$
- nach Zeile 15 gilt also
  $I(length(A) + 1, i, min)$
  $\iff A[min] = min\{A[1], \ldots, A[c]\} \wedge A[i] = min\{\ A[j] \mid j \in \{1, 2, \ldots, c\} \setminus min\}$
- gilt folglich auch vor Zeile 16

$\square$

c)

Analysieren Sie die worst-case Laufzeit des formulierten Algorithmus.

Zeile	Laufzeit
1	$\mathcal{O}(1)$
2	$\mathcal{O}(1)$
3	$\mathcal{O}(1)$
4	$\mathcal{O}(1)$
5	$\mathcal{O}(1)$
6	$\mathcal{O}(1)$
7	$\sum_{i=2}^{n} \mathcal{O}(i)$
8	$\mathcal{O}(1)$
9	$\mathcal{O}(1)$
10	$\mathcal{O}(1)$
11	$\mathcal{O}(1)$
12	$\mathcal{O}(1)$
13	$\mathcal{O}(1)$
14	$\mathcal{O}(1)$
15	$\mathcal{O}(1)$
17	$\mathcal{O}(1)$

$$\overset{Theorem 2.7}{\Longrightarrow} T(n) = \mathcal{O}(1) + \mathcal{O}(n-2) \cdot (8 \cdot \mathcal{O}(1)) + \mathcal{O}(1) = \mathcal{O}(n)$$

Da wir immer über jedes Element des Arrays iterieren müssen, ist die asymptotische Laufzeit gleichzeitig die best- und worst-case Laufzeit unseres Algorithmus. Die Laufzeit ist nicht ausschlaggebend variabel.

Aufgabe 3

Zeigen Sie, dass $(ln(x))^k = O(x^\epsilon)$. Hierbei sind k, ϵ Konstanten größer Null.
Ein möglicher Hinweis: Zeigen Sie die Aussage zunächst für k = 1.

Beweis.

$$lim_{x\to\infty} \frac{x^\epsilon}{(ln(x))^k} = lim_{x\to\infty} \frac{e^{x^\epsilon}}{e^{(ln(x))^k}} = lim_{x\to\infty} \frac{e^{x^\epsilon}}{x \cdot k} \overset{(*)}{\to} \infty$$

(*) Da $f(x) = e^{x^\epsilon}$ exponentiell wächst und $g(x) = x \cdot k$ hingegen nur linear. □

Aufgabe 4

Ordnen Sie die folgenden Funktionen gemäß Ihres asymptotischen Wachstums und begründen Sie Ihre Antwort:

$$f_1(n) = 3^n,\ f_2(n) = n \cdot ln(n),\ f_3(n) = 2^n,\ f_4(n) = e^{log_2(n)},\ f_5(n) = n^n, f_6(n) = n^{3/2}, f_7(n) = n!$$

Wir wollen nun beweisen, dass

$$f_4(n) \in o(f_2(n)) \wedge f_2(n) \in o(f_6(n)) \wedge f_6(n) \in o(f_3(n))$$

$$\wedge f_3(n) \in o(f_1(n)) \wedge f_1(n) \in o(f_7(n)) \wedge f_7(n) \in o(f_5(n))$$

gilt.

Behauptung. $f_4 \in o(f_2) \iff e^{log_2(n)} \in o(n \cdot ln(n))$

Beweis.

$$f_4(n) = e^{log_2(n)} = e^{\frac{ln(n)}{ln(2)}} = n^{\frac{1}{ln(2)}}, \quad f_2(n) = n \cdot ln(n)$$

$$lim_{n\to\infty}\frac{f_2(n)}{f_4(n)} = lim_{n\to\infty}\frac{n \cdot ln(n)}{n^{\frac{1}{ln(2)}}} = lim_{n\to\infty} n^{1-\frac{1}{ln(2)}} \cdot ln(n) \to \infty$$

□

Behauptung. $f_2 \in o(f_6) \iff n \cdot ln(n) \in o(n^{\frac{3}{2}})$

Beweis.

$$f_2(n) = n \cdot ln(n), \quad f_6(n) = n^{\frac{3}{2}} = n^1 \cdot n^{\frac{1}{2}} = n \cdot \sqrt{n}$$

$$lim_{n\to\infty}\frac{f_6(n)}{f_2(n)} = lim_{n\to\infty}\frac{n \cdot \sqrt{n}}{n \cdot ln(n)} = lim_{n\to\infty}\frac{\sqrt{n}}{ln(n)}$$

Da beide Terme divergieren, kann man nach L'Hôspital beide Terme ableiten und die Grenzwerte stimmen überein.

$$\overset{\frac{d}{dn}}{=} lim_{n\to\infty}\frac{n}{2\sqrt{n}} \overset{\frac{d}{dn}}{=} lim_{n\to\infty}\frac{2\sqrt{n}}{2} = lim_{n\to\infty}\sqrt{n} \to \infty$$

□

Behauptung. $f_6 \in o(f_3) \iff n^{\frac{3}{2}} \in o(2^n)$

Beweis.

$$lim_{n\to\infty}\frac{f_3(n)}{f_6(n)} = lim_{n\to\infty}\frac{2^n}{n^{\frac{3}{2}}}$$

Da beide Terme divergieren, kann man nach L'Hôspital beide Terme ableiten und die Grenzwerte stimmen überein.

$$\overset{\frac{d}{dn}}{=} lim_{n\to\infty}\frac{2^n ln(2)}{\frac{3\sqrt{n}}{2}} = lim_{n\to\infty}\frac{2^{n+1} ln(2)}{3\sqrt{n}}$$

Wir wenden wieder den H'Hôspital an.

$$\overset{\frac{d}{dn}}{=} lim_{n\to\infty}\frac{2^{n+1} ln^2(2)\sqrt{n}}{3} \to \infty$$

□

Behauptung. $f_3 \in o(f_1) \iff 2^n \in o(3^n)$

Beweis.

$$lim_{n\to\infty}\frac{f_1(n)}{f_3(n)} = lim_{n\to\infty}\frac{3^n}{2^n} = lim_{n\to\infty}\left(\frac{3}{2}\right)^n \to \infty$$

□

Behauptung. $f_1 \in o(f_7) \iff 3^n \in o(n!)$

Beweis.
Induktionsbehauptung: $\forall n \in \mathbb{N}_{>6} : 3^n < n!$
Induktionsanfang: $n = 7 :\quad 3^7 = 2187$
$7! = 5040$ ✓

Induktionsvoraussetzung: Für ein beliebiges, aber festes $n > 6$ gilt: $3^n < n!$
Induktionsschritt: $3^{n+1} < (n+1)! \iff 3 \cdot 3^n < (n+1) \cdot n!$ durch Division beider Seiten der Ungleichung mit 3 bekommen wir raus $\underbrace{3^n < n!}_{IB} < \frac{n+1}{3} \cdot n!$. Da der Bruch $\frac{n+1}{3}$ nach Induktionsvoraussetzung größer 1 ist, folgt daraus die Induktionsbehauptung. □

Behauptung. $f_7 \in o(f_5) \iff n! \in o(n^n)$

Beweis.

$$lim_{n\to\infty}\frac{f_5(n)}{f_7(n)} = lim_{n\to\infty}\frac{n^n}{n!} = lim_{n\to\infty}\underbrace{\frac{n}{1}\frac{n}{2}\frac{n}{3}\cdots}_{>1}\underbrace{\frac{n}{n}}_{=1} \to \infty$$

□