Algorithmen und Programmierung III, WS 03/04

Lena Schlipf
Benjamin Jankovic
Tutor: Till Zoppke

1. Übungsblatt *(Musterlösung, 17.11.2003)*

Abgabe: 30.10.2003

Aufgabe 1

(i)

(b) $\log_2 n$ < (f) $\sqrt{n}$ < (d) n < (g) $n(\log_2 n)^2$
< (h) n^2 < (a) n^3 < (c) $1{,}8^n$ < (e) 3^n

(‚<' steht hier für „wächst schwächer als")

Begründung:

$\log_2 n = O\left(\sqrt{n}\right)$, denn $\lim_{n\to\infty} \frac{\log_2 n}{\sqrt{n}} = 0$

$\sqrt{n} = O(n)$, denn $\lim_{n\to\infty} \frac{\sqrt{n}}{n} = \lim_{n\to\infty} \frac{1}{\sqrt{n}} = 0$

$n = O(n(\log_2 n)^2)$, denn $\lim_{n\to\infty} \frac{n}{n(\log_2 n)^2} = \lim_{n\to\infty} \frac{1}{(\log_2 n)^2} = 0$

$n(\log_2 n)^2 = O(n^2)$, denn $\lim_{n\to\infty} \frac{n(\log_2 n)^2}{n^2} = \lim_{n\to\infty} \frac{(\log_2 n)^2}{n} = 0$

$n^2 = O(n^3)$, denn $\lim_{n\to\infty} \frac{n^2}{n^3} = \lim_{n\to\infty} \frac{1}{n} = 0$

$n^3 = O(1.8^n)$, denn $\lim_{n\to\infty} \frac{n^3}{1.8^n} = 0$

$1.8^n = O(3^n)$, denn $\lim_{n\to\infty} \frac{1.8^n}{3^n} = \lim_{n\to\infty} \left(\frac{3}{5}\right)^n = 0$

Bemerkung:

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ bedeutet, daß $\frac{f(n)}{g(n)}$ beschränkt ist, also wächst $g(n)$ stärker als $f(n)$.

(ii)

Berechnet mit: $f(n) \geq \frac{f(x)}{1000} \Rightarrow x \Rightarrow f^{-1}(1000 \cdot f(n))$,

Für $n = 20$:

(a) 200 (b) 20^{1000} (c) 31 (d) 20000
(e) 26 (f) $2 \cdot 10^7$ (g) 2839 (h) 632

Für $n = 1000$:

(a) 10000 (b) 1000^{1000} (c) 1011 (d) 10^6
(e) 1006 (f) 10^5 (g) 300017 (h) 31622

Bemerkung:

Für die Aufgabenteile (g) ließ sich nicht so einfach die Umkehrfunktion f^{-1} bilden, deshalb wurden die Werte mittels Mathematica graphisch ermittelt. (Schnittpunkt von $1000 \cdot f(n)$ und $f(x)$)

Aufgabe 2

Die technische Verbesserung der Hardware kann allenfalls die Laufzeit um einen Faktor verbessern. Durch den Entwurf effizienter Algorithmen können Verbesserungen im Bereich der Potenzordnung erzielt werden (evtl. sogar von exponentieller auf polynomielle Laufzeit). Das heißt für Probleme, die große Datensätze als Eingabe besitzen, daß man auf effiziente Algorithmen auch in Zukunft nicht verzichten kann. Für Probleme mit kleinen Datensätzen mit evtl. konstanter Größe kann allerdings die Hardware viel mehr bezwecken als ein guter Algorithmus.

Aufgabe 8

(a)

$3n^2 + 4n + 32 + \frac{27}{2}n \cdot \log_2 n = \Theta(n^2)$, denn

$\lim_{n \to \infty} \frac{n^2}{3n^2 + 4n + 32 + \frac{27}{2}n \cdot \log_2 n} = \frac{1}{3}$ und

$\lim_{n \to \infty} \frac{3n^2 + 4n + 32 + \frac{27}{2}n \cdot \log_2 n}{n^2} = 3$, also gilt

$3n^2 + 4n + 32 + \frac{27}{2}n \cdot \log_2 n = O(n^2)$ und

$n^2 = O(3n^2 + 4n + 32 + \frac{27}{2}n \cdot \log_2 n)$

(b)

$\max\left\{n \log_2 n, (\log_2 n)^4\right\} = \Theta(n \log_2 n)$, denn es gilt

$n \log_2 n = \Theta(n \log_2 n)$ und

$\lim_{n \to \infty} \frac{(\log_2 n)^4}{n \log_2 n} = \lim_{n \to \infty} \frac{(\log_2 n)^3}{n} = \lim_{n \to \infty} \frac{\log_2 n}{\sqrt[3]{n}} = \lim_{n \to \infty} \frac{n}{2^{\sqrt[3]{n}}} = \lim_{n \to \infty} \frac{1}{2^{\sqrt[3]{n} - \log_2 n}} = 0$

(c)

$2^{2n + \log_2 n} = \Theta\left(2^{2n + \log_2 n}\right) = \Theta\left(2^{2n} \cdot 2^{\log_2 n}\right) = \Theta\left(4^n \cdot n\right)$

Aufgabe 9

```
static void selectionSort (int [] a){

      for (int i=0; i < a.length-1; i++){

            // min soll die Position beinhalten, in der das Minimum des noch
            // unsortierten Listenteils steht. Initialisiert wird mit Pos. i.

            int min = i;

            // Sequentiell wird die Liste nach der neuen Minimums-Position
            // durchsucht, indem mit dem bisherigen Minimum verglichen wird.

            for (int j=i+1; j<a.length; j++)    if (a[j]<a[min]) min=j;

            // Wurde die Restliste einmal durchlaufen, d.h. kennt man die
            // Position des Minimums, werden die Inhalte von min und i vertauscht

            int temp = a[i];
            a[i] =  a[min];
            a[min] = temp;
      }
}
```