Lösungen der Hausaufgaben von Übungsblatt 1

Algorithmen und Datenstrukturen (WS 2013, Ulrike von Luxburg)

Lösungen zu Aufgabe 1

(a)

$$1/n \prec 1 \prec \log\log n \prec \log n \asymp \log(n^3) \prec \log(n^{\log n}) \prec n^{0.01} \prec \sqrt{n} \prec n\log n \prec n^8 \prec 2^n \prec 8^n \prec n! \prec n^n$$

Es gilt $1/n \prec 1$, weil $(1/n)/1 \to 0$. Ebenso $1 \prec \log\log n$ wegen $1/(\log\log n) \to 0$ und $\log\log n \prec \log n$ wegen $(\log\log n)/\log n \to 0$. Weiterhin gilt $\log(n^3) = 3\log(n) \in \Theta(\log n)$. Außerdem $\log(n^{\log n}) = (\log n)^2$ und somit $\log n/\log(n^{\log n}) = 1/\log n \to 0$. Weiterhin ist $(\log n)^2/n^{0.01} \to 0$ äquivalent zu $\log n/n^{0.005} \to 0$, was bekannterweise wahr ist. Zudem gilt $n^{0.01} \prec \sqrt{n} = n^{0.5}$, da $n^{0.01}/n^{0.5} = 1/n^{0.49} \to 0$. Ebenso folgt $\sqrt{n} \prec n\log n$ wegen $n^{0.5}/(n\log n) = 1/(n^{0.5}\log n) \to 0$. Analog folgt $n\log n \prec n^8$ wegen $n\log n/n^8 = \log n/n^7 \to 0$. Es ist bekannt, dass $n^c \prec 2^n$ für alle $c \in \mathbb{R}$, somit $n^8 \prec 2^n$. Weiterhin gilt wegen $2^n/8^n = (1/4)^n \to 0$, dass $2^n \prec 8^n$. Zudem folgt aus $8^n/n! = (8\cdot 8\cdots 8\cdot 8)/(n\cdot(n-1)\cdots 2\cdot 1) \leq (8\cdot 8^8)/(n\cdot 8\cdot 7\cdots 2\cdot 1) \in \mathcal{O}(8/n)$, dass wegen $8/n \to 0$ auch $8^n \prec n!$. Zu guter Letzt liefert $n!/n^n = (n\cdot(n-1)\cdots 2\cdot 1)/(n\cdot n\cdots n\cdot n) \leq 1/n$ wegen $1/n \to 0$, dass auch $n! \prec n^n$.

(b) (i) Richtig, weil $\log_b(n) = \log_2(b)^{-1}\log_2(n) \in \Theta(\log_2 n)$

(ii) Falsch, da z.B. $n \in \mathcal{O}(n)$ gilt, aber nicht $n \in \omega(n)$.

(iii) Richtig. "$\Leftarrow$" ist klar, da $f_1(n) = n \in \Theta(n)$. Zeigen "$\Rightarrow$" per Kontraposition: Für $c < 1$ folgt mit geometrischer Reihe, dass $f_c(n) = \sum_{i=0}^{n} c^i = 1/(1-c) \in \mathcal{O}(1) \neq \Theta(n)$ gilt, und für $c > 1$ folgt, dass $f_c(n) \geq c^n \in \omega(n) \neq \Theta(n)$.

Lösungen zu Aufgabe 2

(a) Induktionsanfang (IA) für Rekursion 2. Ordnung: Es ist $F_6 = 8 \geq 2^3$ und $F_7 = 13 \geq 2^{3.5} \approx 11.31$. Erhalten somit die Induktionsvoraussetzung (IV), dass $F_n \geq 2^{0.5n}$ für ein $n \geq 7$ gilt, sowie für alle $n' \in \{6,\ldots,n\}$. Zeigen nun mit dem Induktionsschritt (IS), dass daraus die Induktionsbehauptung (IB) $F_{n+1} \geq 2^{0.5(n+1)}$ folgt:

$$F_{n+1} = F_n + F_{n-1} \overset{IV}{\geq} 2^{0.5n} + 2^{0.5(n-1)} = 2^{0.5n} + \frac{2^{0.5n}}{\sqrt{2}} \geq 2\cdot\frac{2^{0.5n}}{\sqrt{2}} = \sqrt{2}\cdot 2^{0.5n} = 2^{0.5(n+1)}$$

Dabei verwenden wir die IV für n und für $n-1$, beide ≥ 6.

(b) Ein geeignetes c muss letztlich im Induktionsschritt erfüllen, dass

$$F_{n+1} = F_n + F_{n-1} \leq 2^{cn} + 2^{c(n-1)} \overset{!}{\leq} 2^{c(n+1)}$$

gilt. Die letzte Ungleichung ist mittels Division durch 2^{cn} äquivalent zu $1 + \frac{1}{2^c} \overset{!}{\leq} 2^c$ und beispielsweise für $c = 0.7$ erfüllt, welches wir nun festhalten. Dies liefert auch einen gültigen IA mit $F_0 = 0 \leq 2^{0.7\cdot 0} = 1$ und $F_1 = 1 \leq 2^{0.7\cdot 1} \approx 1.62$, und somit die IV $F_n \leq 2^{0.7n}$ für ein $n \geq 1$ und alle $n' \in \{0,\ldots,n\}$. Müssen nun die IB $F_{n+1} \leq 2^{0.7(n+1)}$ zeigen. Betrachte dazu

$$F_{n+1} = F_n + F_{n-1} \overset{IV}{\leq} 2^{0.7n} + 2^{0.7(n-1)} \overset{(\star)}{\leq} 2^{0.7(n+1)},$$

wobei $(\star)$ bereits bei der Wahl von c gezeigt wurde.

Lösungen zu Aufgabe 3

(a) Für $n = 0$ gilt $\begin{pmatrix} F_0 \\ F_1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} F_0 \\ F_1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^0 \cdot \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$, somit ist der IA gezeigt. Erhalten somit $(\star)$ aus der Aufgabenstellung als IV. Nun der IS:

$$\begin{pmatrix} F_{n+1} \\ F_{n+2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} F_n \\ F_{n+1} \end{pmatrix} \overset{IV}{=} \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \cdot \begin{pmatrix} F_0 \\ F_1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{n+1} \cdot \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

(b) Das Beispiel X^{64} lässt sich wie folgt durch lediglich 6 Multiplikationen berechnen: $X \cdot X = X^2, X^2 \cdot X^2 = X^4, \ldots, X^{32} \cdot X^{32} = X^{64}$. Für Nicht-Zweierpotenzen geschieht dies ähnlich, mittels binärer Exponentiation ("Square-And-Multiply") oder einer seiner Varianten, zum Beispiel der Folgenden: Betrachte den Exponenten n in Binärdarstellung als $n = (b_\ell \ldots b_0)_1$, d.h., mit $\ell \in \mathcal{O}(\log n)$ Bits. Erhalten somit die Darstellung:

$$X^n = X^{\sum_{b_i=1} 2^i} = \prod_{b_i=1} X^{2^i}$$

Berechne nun mittels ℓ Multiplikationen alle Potenzen X^{2^i} für $i = 1 \ldots \ell$. Setze das Ergebnis zusammen als Produkt derjenigen Potenzen, deren Bit gesetzt ist, also als $X^n = \prod_{b_i=1} X^{2^i}$, was maximal weitere $(\ell + 1)$ Multiplikationen benötigt. Insgesamt fallen auf diese Weise höchstens $2\ell + 1 \in \mathcal{O}(\log n)$ Multiplikationen an.

(c) Haben in Aufgabenteil (b) gesehen, dass $\mathcal{O}(\log n)$ viele Matrixmultiplikationen genügen um $\left(\begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix}\right)^n$ zu berechnen. Jede Matrixmultiplikation von 2×2-Matrizen lässt sich berechnen per

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{pmatrix},$$

also mittels höchstens 8 skalaren Multiplikationen und 4 skalaren Additionen. Wir wissen aus der vorigen Aufgabe, dass die skalaren Zahlenwerte nicht schneller als exponentiell wachsen und somit $\mathcal{O}(n)$ Bits zu ihrer Darstellung genügen. Somit ist der Aufwand jeder einzelnen Matrixmultiplikation durch $\mathcal{O}(n^{1.59})$ beschränkt. Insgesamt erhalten wir also die Laufzeit $\mathcal{O}(n^{1.59} \log n)$, was asymptotisch echt schneller als $\mathcal{O}(n^2)$ ist.

[Anm: Durch geschickteres Betrachten der Bitlängen lässt sich diese Schranke sogar auf $\mathcal{O}(n^{1.59})$ senken, bzw. allgemeiner auf $\mathcal{O}(m(n))$, wobei $m(n)$ die minimal notwendige Anzahl Zeitschritte zur Multiplikation zweier n-Bit-Zahlen bezeichnet.]