Höhere Algorithmik - 1. Übungsblatt

Martin Lenders (Mi. 14-16), Ralf Müller-Zimmermann (Di. 14-16)

16. August 2014

Aufgabe 1 O-Notation

- $2^{(2^n)}$
  Steigt exponentiell exponentiell.
- 2^n
  Steigt linear exponentiell.
- $n^{\log\log n}$
  Steigt logarithmisch exponentiell.
- $(\lceil\log(n)\rceil)! \in O(n!)$
- n^2
  Die Fakultätsfunktion steigt schneller als die Quadratfunktion. Unabhängig von der Wachstumsgeschwindigkeit des Wertes vor der Fakultät, wird die Quadratfunktion irgendwann eigeholt.
  $\Rightarrow (\lceil\log(n)\rceil)! \in \Omega(n^2)$
  $n^n > 2^n \Rightarrow n^{\log\log n} \in \Omega(n^2)$
- $4^{\log n} = 2^{\log n} * 2^{\log n} = n \cdot n = n^2 \Rightarrow n^2 \in \Theta(4^{\log n})$
- $(\sqrt{2})^{\log n} = \sqrt{2^{\log n}} = \sqrt{n}$
  e^n steigt schneller als $n^2 \Rightarrow$ Die Umkehrfunktionen haben umgekehrte Eigenschaft. $\Rightarrow n^2 \in \Omega((\sqrt{2})^{\log n})$
- $\log^2 n = \frac{\ln^2 n}{\ln^2 2} = c \cdot \ln n$
  Der Nenner ist konstant, daher wird der Zähler für grosse n nicht mehr so stark beeinflusst, sodass er $\ln n$ überholt.
- $\ln n$

$$\ln n = \frac{\log_2 n}{\log_2 e} = c \cdot \log^2 n$$
$$\Rightarrow \log^2 \in \Theta(\ln n)$$

- $n^{\frac{1}{\log n}}$

$$\log_2 n = x \Leftrightarrow 2^x = n$$
$$\Rightarrow (2^x)^{\frac{1}{x}} = 2^{x \cdot \frac{1}{x}} = 2 \quad \text{(Konstant)}$$
$$2 < c \cdot \ln n \Rightarrow \ln n \in \Omega(n^{\frac{1}{\log n}})$$

Aufgabe 2 Sammelbilder

(a). **Zu zeigen:** $E[X] = \sum_{i=1}^{n} E[X_i]$

Herleitung:

$$\begin{aligned}
\sum_{i=1}^{n} E[X_i] &= \sum_{i=1}^{n} \left(\sum_{j=1}^{\infty} j * P(X_i = j) \right) \\
&= \sum_{j=1}^{\infty} \left(\sum_{i=1}^{n} j * P(X_i = j) \right) \\
&= \sum_{j=1}^{\infty} j * \left(\sum_{i=1}^{n} P(X_i = j) \right) \\
&= E\left[\sum_{i=1}^{n} X_i \right] \\
&= E[X]
\end{aligned}$$

(b). **Gesucht:** $E[X_i]$

Herleitung:

$$\begin{aligned}
E[X_i] &= \frac{1}{p_i} \\
p_i &= \frac{n-i}{n} \\
\Rightarrow E[X_i] &= \frac{n}{n-i}
\end{aligned}$$

(c). **Zu zeigen:** $E[X] = O(n \log n)$

Herleitung:

$$\begin{aligned}
E[X] &= \sum_{i=1}^{n} E[X_i] \\
&= \sum_{i=1}^{n} \frac{n}{n-i} \\
&= n * \left(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \right) \\
&= n * \sum_{i=1}^{n} \frac{1}{i} \\
&= n \log n
\end{aligned}$$

Aufgabe 3 Varianten Mergesort

(a). **Selection Sort** sortiert die Elemente einer Liste, indem er eine Auswahl (*Selection*), die zu Beginn des Algorithmus leer ist, auffüllt, indem er die Verbleibende Liste nach ihrem kleinsten Element durchsucht und durch eine Vertauschun in die Auswahl aufnimmt.

Worst-Case-Szenario: Im schlimmsten Fall ist die Liste falsch herum sortiert, da der Algorithmus so bei jedem Durchlauf den aktuellen Suchraum von vorne bis hinten durchsuchen muss. Im ersten Durchlauf sind das also $n-1$ Elemente, im zweiten $n-2$ usw. Die Laufzeit für den Worst Case ist also

$$\sum_{i=1}^{n-1} i = \frac{(n-1) \cdot n}{2} = \frac{1}{2}\left(n^2 - n\right) \in O(n^2).$$

Mergesort sortiert die Elemente einer Liste, indem er eine Liste in seine Elemente zerlegt und rekursiv diese wieder zusammensetzt, wobei er jedes mal beim zusammensetzen die neu entstandenen Teillisten sortiert.

Worst-Case-Szenario: Im schlimmsten Fall müssen die Elemente bei jedem Verschmelzen wechselweise eingefügt werden. Für das Verschmelzen von zwei k-elementigen Listen sind also $2k-2$

Vergleiche nötig, womit für einen Rekursionsschritt mit einer n-elementigen $n-1$ Vergleiche durchgeführt werden. Das Verschmelzen bei einer n-elementigen Folge wird $\log_2 n$ mal aufgerufen. Die Laufzeit für den Worst Case ist also

$$(n-1) \cdot \log_2 n \in O(n \log n)$$

(b). (i) Wir nehmen an, dass n und M Zweierpotenzen sind. Für den normalen Mergesort würden wir $\log_2 n$-mal die Listen verschmelzen (s. 3a). Davon entfallen aber nun die Rekursionsschritte auf die Teillisten, die der Selectionsort nun übernimmt, wir haben also nur $\log_2 n - \log_2(M-1) = \log_2\left(\frac{n}{M-1}\right)$ Verschmelzungen. Der Selectionsort sortiert alle Teillisten der Länge $m = \frac{M}{2}$ und benötigt dafür $\frac{1}{2}(m^2 - m)$ Vergleiche (s. 3a). Die Gesamtanzahl an Vergleichen ist daher:

$$\log_2\left(\frac{n}{M-1}\right) + \frac{1}{2}\left[\frac{M^2}{4} - \frac{M}{2}\right] \in O(n \log n)$$

(ii) Wir ziehen für den Laufzeitvergleich zusätzlich zu den Vergleichen des M-Mergesorts auch noch die $\frac{n}{M-1} - 1$ Verschmelzungen mit ein:

$$\log_2\left(\frac{n}{M-1}\right) \cdot \left(\frac{n}{M-1} - 1\right) + \frac{1}{2}\left[\frac{M^2}{4} - \frac{M}{2}\right] < (n-1) \cdot \log_2 n$$

(c). Es wurden mit zufälligen Listen der Längen 10, 1000, 50000, 1000000 und mit $M \in \{2, 3, 5, 10, 20, 40, 50, 60\}$ die Algorithmen ausgewertet. M-Mergesort benötigte dabei mindestens genauso viele Vergleiche wie Mergesort, allerdings war es in der Laufzeit deutlich besser. Es lief, mit Abweichungen, etwa um $\frac{4}{3}$-mal schneller als Mergesort. Das optimale M wurde durch Änderung des Parameters M, bei mehreren Durchläufen auf ein Liste konstanter Länge, ermittelt. Aufgrund der Varianz konnte das optimale M zwischen 10 und 60 eingeordnet werden, allerdings nicht genau bestimmt werden. Die Konstante konnte sich aufgrund der Leistung unserer Rechner nicht bestimmen lassen (in Nanosekunden kamen mitunter sogar negative Laufzeiten zu stande o.O).