Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Dr. Hanjo Täubig
Jeremias Weihmann

Sommersemester 2013
Übungsblatt 3
6. Mai 2013

Grundlagen: Algorithmen und Datenstrukturen

Abgabetermin: Jeweilige Tutorübung in der Woche vom 13. bis 17. Mai

Tutoraufgabe 1

Sei $h : \mathbb{N}_0 \to \mathbb{R}$ eine Funktion mit $\exists n_0 \in \mathbb{N} : \forall n > n_0 : h(n) > 0$. In der Vorlesung haben wir einige Methoden kennengelernt, mit denen wir zeigen können, dass die Worst-Case Laufzeit eines Algorithmus in $\mathcal{O}(h(n))$ liegt. In diesem Kontext stellt sich die Frage, wie man zeigt, dass die Worst-Case Laufzeit eines Algorithmus nicht in $\mathcal{O}(h(n))$ liegt.

Dazu betrachten wir das folgende Problem: Gegeben sei eine Folge von natürlichen Zahlen $(x_1, x_2, \ldots, x_n)$. Gefragt ist, ob es eine Zahl gibt, die in dieser Folge (mindestens) zweimal vorkommt.

Wir betrachten den Algorithmus ExistsPair für dieses Problem. Zeigen Sie, dass die Laufzeit dieses Algorithmus nicht in $\mathcal{O}(n^{3/2})$ liegt.

Algorithmus 1: ExistsPair

```
   Input : int[] (x_1, x_2, ..., x_n)
 1 int i := 1
 2 while i ≤ n do
 3 |   int j := 1
 4 |   while j ≤ n do
 5 |   |   if x_j = x_i and i ≠ j then
 6 |   |   |   return Ja
 7 |   |   j := j + 1
 8 |   i := i + 1
 9 return Nein
```

Tutoraufgabe 2

Wir betrachten nochmals den Minimum-Algorithmus aus der vorigen Woche (siehe Algorithmus 2). Nachdem wir bereits wissen, dass die Worst-Case Laufzeit in $\mathcal{O}(n)$ liegt, wollen wir dieses Mal die *genaue* Average-Case Laufzeit berechnen (nicht die asymptotische Average-Case Laufzeit). Sie dürfen davon ausgehen, dass jede Eingabe der Länge n aus n unterschiedlichen Zahlen besteht.

Algorithmus 2: 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
```

Hausaufgabe 1

Zeigen Sie, dass es Funktionen $f, g : \mathbb{N}_0 \to \mathbb{R}$ mit $\exists n_0 \in \mathbb{N} : \forall n > n_0 : f(n), g(n) > 0$ gibt, sodass $f(n) \notin \mathcal{O}(g(n))$ und $f(n) \notin \Omega(g(n))$ gilt.

Hinweis: Verwenden Sie hierfür die wie folgt definierten Funktionen:

$$f(n) = n^2 \quad \text{und} \quad g(n) = \begin{cases} n, & \text{falls n gerade} \\ n^3, & \text{falls n ungerade} \end{cases}$$

Hausaufgabe 2

Ordnen Sie die folgenden Funktionen so an, dass für zwei in der Anordnung aufeinander folgende Funktionen f und g gilt: $f(n) \in o(g(n))$ oder $f(n) \in \mathcal{O}(g(n))$. Beweisen Sie Ihre Anordnung. Benutzen Sie gegebenenfalls die Monotonie- und Konvergenzeigenschaften elementarer Funktionen ohne Beweis. Dabei sind $c, \epsilon > 0$ von n unabhängige Konstanten.

$$\ln \operatorname{ld} n, \quad n^n, \quad e^{n \operatorname{ld} n}, \quad \sqrt{\ln n}, \quad n!, \quad \epsilon n^c, \quad n^{c+\epsilon}, \quad n^c$$

Hinweis: Verwenden Sie in gegebenenfalls die Rechenregeln aus der Vorlesung, die einen Zusammenhang zwischen dem Wachstumsverhalten von $f(n)$ und $g(n)$ und dem Wachstumsverhalten von $f'(n)$ und $g'(n)$ herstellen. Alternativ können Sie auch eine Regel von L'Hospital verwenden, die besagt:

Seien $f, g : \mathbb{R} \to \mathbb{R}$ zwei Funktionen mit entweder $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = 0$ oder $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = \infty$. Dann gilt $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$ sofern der Limes $\lim_{n\to\infty} \frac{f'(n)}{g'(n)}$ existiert.

An vielen Stellen ist es auch hilfreich, die Transitivitätsregeln von Übungsblatt 2 zu verwenden.

Hausaufgabe 3

Seien $f, g, h : \mathbb{N}_0 \to \mathbb{R}$ Funktionen mit $\exists n_0 \in \mathbb{N} : \forall n > n_0 : f(n), g(n), h(n) > 0$. Zeigen Sie die folgenden Aussagen:

(a) $f(n) \in \omega(g(n)) \Rightarrow f(n) \notin \mathcal{O}(g(n))$.

(b) $f(n) \in o(g(n)) \Rightarrow f(n) \notin \Omega(g(n))$.

(c) $f(n) \in \Omega(g(n))$ und $h(n) \in o(g(n)) \Rightarrow f(n) \in \omega(h(n))$.

(d) $f(n) \in \mathcal{O}(g(n))$ und $h(n) \in \omega(g(n)) \Rightarrow f(n) \in o(h(n))$.

Anmerkung: Die Rückrichtungen dieser Gesetze gelten im Allgemeinen nicht!

Hausaufgabe 4

Betrachten Sie das Problem und den Algorithmus ExistsPair von Tutoraufgabe 1. Zeigen Sie, dass für jede Funktion $h(n) \in o(n^2)$ mit $\exists n_0 \in \mathbb{N} : \forall n > n_0 : h(n) > 0$ die Worst-Case Laufzeit nicht in $\mathcal{O}(h(n))$ liegt.