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

Sommersemester 2012
Übungsblatt 3
7. Mai 2012

Grundlagen: Algorithmen und Datenstrukturen

Abgabetermin: Jeweilige Tutorübung in der Woche vom 14. bis 18. 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 $O(h(n))$ liegt. In diesem Kontext stellt sich die Frage, wie man zeigt, dass die Worst-Case Laufzeit eines Algorithmus nicht in $O(h(n))$ 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 $O(n^{3/2})$ 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

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) Wenn $f(n) \in \omega(g(n))$, dann gilt $f(n) \notin O(g(n))$.

(b) Wenn $f(n) \in o(g(n))$, dann gilt $f(n) \notin \Omega(g(n))$.

(c) Wenn $f(n) \in \Omega(g(n))$ und $h(n) \in o(g(n))$, dann gilt $f(n) \in \omega(h(n))$.

(d) Wenn $f(n) \in O(g(n))$ und $h(n) \in \omega(g(n))$, dann gilt $f(n) \in o(h(n))$.

Anmerkung: Die Rückrichtungen dieser Gesetze gelten im Allgemeinen nicht!

Hausaufgabe 2

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 $f(n) \notin O(g(n))$ und $g(n) \notin O(f(n))$ 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 3

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

Hausaufgabe 4

Betrachten Sie das folgende Problem: Gegeben ist eine Zahl $a \in \{0, 1\}$ und eine Zahlenfolge $(x_1, x_2, \ldots, x_k)$, wobei $x_i \in \{0, 1\}$ für alle $i \in \{1, \ldots, k\}$ gilt. Gefragt ist, ob a in der Zahlenfolge vorkommt.

Algorithmus 3 löst offensichtlich dieses Problem. Berechnen Sie die erwartete Anzahl der Vergleiche $x_i = a$, die dieser Algorithmus durchführt, wenn jede Eingabe mit den oben beschriebenen Eigenschaften mit derselben Wahrscheinlichkeit auftritt. Bestimmen Sie zudem die asymptotisch erwartete Laufzeit.

Algorithmus 3: IsElementOf

```
   Input : int a, int[] (x_1, x_2, ..., x_k)
1  int i = 1
2  while i ≤ k do
3  |  if x_i = a then
4  |  |  return Ja
5  |  i = i + 1
6  return Nein
```

Hinweis: Verwenden Sie, wie in der Vorlesung demonstriert binäre Zufallsvariablen (sogenannte Indikatorvariablen). Verwenden Sie zudem, dass für eine binäre Zufallsvariable Y mit $\mathbf{Pr}[Y = 1] = c$ und $\mathbf{Pr}[Y = 0] = 1 - c$ für ein $c \in [0, 1]$, der Erwartungswert von Y genau c ist, d.h. $\mathbb{E}[Y] = \mathbf{Pr}[Y = 1] = c$. Sie können zudem für $c \neq 1$ von der Gleichung $\sum_{i=0}^{n} c^i = \frac{c^{n+1}-1}{c-1}$ Gebrauch machen.