Algorithmen und Datenstrukturen

Dr. Martin Moritz Kleine

Probeklausur

Juni 2013

- **Lesen Sie sich zunächst diese Hinweise durch.**
- **Erlaubte Hilfsmittel**: 2 doppelseitig beschriftete Seiten mit Notizen, Stift, leere Seiten, ggf. Wörterbuch – sonst nichts.
- Tipp: Lesen Sie erst alle Aufgaben und bearbeiten Sie zunächst jene, die Ihnen leicht fallen.
- Dauer: Für die Bearbeitung sind **60 Minuten** vorgesehen.

Name (in Druckschrift): ....

Vorname (in Druckschrift): ....

Matrikelnummer: ....

Unterschrift: ....

Erreichte Punkte	Leistungspunkte

1. Analyse von Algorithmen

Aufgabe 1.1: Gegeben sei der folgende Algorithmus $\mathsf{fun}(n)$:

Algorithm 1 $\mathsf{fun}(n)$, $n \in \mathbb{N}$

```
if n < 3 then
    return 42
else
    return 2 * fun(n − 1) + 4 * fun(n − 2) + 8 * fun(n − 3)
end if
```

Wir interessieren uns für die Anzahl $A(n)$ der Aufrufe der Prozedur fun, wenn wir mit dem Aufruf $\mathsf{fun}(n)$ starten.

1. Stellen Sie eine Rekursionsgleichung für den Zeitbedarf $A(n)$ auf.

von 3

$$A(n) =$$

2. Erläutern Sie, wie Sie auf die Gleichung gekommen sind:

von 2

3. Was vermuten Sie für das asymptotische Wachstum von $A(n)$?

$$A(n) \in \underline{\qquad\qquad\qquad\qquad}$$

von 2

2. Rekurrenzen

Aufgabe 2.1: Betrachte die folgende Rekurrenz:

$$f(n) = \begin{cases} 4 & \text{falls } n = 1 \\ (f(\frac{1}{2}n))^3 & \text{falls } n > 1 \end{cases}$$

Beweisen Sie (per Induktion über k) die Vermutung:

von
8

$$f(n) = \left(f\left(\frac{n}{2^k}\right)\right)^{(3^k)} \quad \text{für alle } k \text{ mit } 2^k < n$$

- Induktionsanfang:

- Induktionsvoraussetzung:

- Induktionsschritt:

  Zu zeigen ist:

  Dies sieht man wie folgt:

Aufgabe 2.2: Gegeben sei die folgende Rekurrenz:

$$T(n) = \begin{cases} 1 & \text{falls } n = 1 \\ 4T(n/2) + \frac{n^2}{log\ n} & \text{falls } n > 1 \end{cases}$$

Bestimmen Sie durch Anwendung des Mastertheorems eine Funktion, die asymptotisch gleich T ist, sofern das Mastertheorem anwendbar ist. Begründen Sie anderenfalls, warum das Mastertheorem nicht anwenbar ist.

von
8

3. Sortieren

Aufgabe 3.1: Betrachte das folgende Sortierverfahren:

```
Sort(A[1...n])

1: for i = n downto 2 do
2:    idx = idxMax(A[1..i])
3:    swap(A[i], A[idx])
4: end for
```

```
idxMax(A[1...n])
1: idx = 1
2: for i = 2 to n do
3:    if A[i] > A[idx] then
4:       idx = i
5:    end if
6: end for
7: return idx;
```

1. Wir wollen die Arbeitsweise von `Sort` für das folgende Array A nachvollziehen. Notieren Sie dazu die Werte, die die Variablen jeweils am Ende der For-Schleife (d.h. in Zeile 4) besitzen.
   Bitte notieren Sie nur Werte, die sich im Vergleich zur letzten Zeile verändert haben. In der letzten Zeile notieren Sie bitte alle Werte.

von
4

i	idx	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
-	-	6	8	3	5	1	7	2	9	4
9										
8										
7										
6										
5										
4										
3										
2										
2										

2. In der Vorlesung haben wir die Korrektheit von `Sort` mit Hilfe einer Invarianten bewiesen, d.h. einer Eigenschaft, die vor dem Schleifeneintritt und auch nach dem -austritt gilt.
   Formulieren Sie diese Invariante mit Ihren eigenen Worten!

von
3

3. Erläutern Sie knapp, wie die Korrektheit des Sortieralgorithmus aus der Invariante folgt.

von
2

4. Erläutern Sie knapp, warum die Invariante vor dem allerersten Schleifeneintritt gilt.

von
2

5. Erläutern Sie knapp, warum die Invariante, die zu Beginn der Schleife gilt, nach einem Schleifendurchlauf weiterhin gilt.

von
2

4. Algorithmentechniken

Aufgabe 4.1:

1. Beschreiben Sie die funktionsweise von *Divide-And-Conquer*-Algorithmen.

von
2

2. Geben Sie ein Beispiel für einen *Divide-And-Conquer*-Algorithmus an.

von
1

3. Notieren Sie den genannten Algorithmus im Pseudo-Code

von
3

5. Graphen

Aufgabe 5.1: Bestimmen Sie bei den maximalen Fluss durch folgendes Netzwerk indem Sie die Ford-Fulkerson Methode anwenden. Geben Sie für jeden Schritt das jeweiligen Restnetzwerk und den gewählten Erweiterungspfad an.

von
6