Prof. Dr. Ir. Dr. h. c. Joost-Pieter Katoen Stefan Dollase, Ira Fesefeldt, Tim Quatmann, Jip Spel, Tobias Winkler

Übung 5

Hinweise:

- Die Übungsblätter sollen in 3er Gruppen bearbeitet werden.
- Die Bearbeitung der Übungsblätter ist nicht zwingend notwendig um die Klausurzulassung zu erhalten. Es werden zwar Punkte vergeben, diese dienen jedoch nur zur eignen Einschätzung.
- Zur Vorbereitung auf Präsenzübung und Klausur empfehlen wir trotzdem alle Übungsblätter semesterbegleitend zu bearbeiten. Besonders relevant hierfür sind Aufgaben, die mit einem ★ markiert sind.
- Die Lösung des Übungsblattes wird in RWTHmoodle veröffentlicht. Solange die Tutorien nicht stattfinden können, werden außerdem Videos zu jeder Aufgabe angefertigt.
- Wir haben in RWTHmoodle ein Forum eingerichtet, welches als erste Anlaufstelle für **Fragen** dienen soll.
- Relevante Vorlesungen für dieses Übungsblatt: 1, 2, 5, 6

Aufgabe 1 (Master Theorem): ★5·8=40 Punkte

Geben Sie für die folgenden Rekursionsgleichungen an, ob diese mit dem Master-Theorem aus der Vorlesung gelöst werden können. Begründen Sie ihre Antwort! Geben Sie auch die resultierende Komplexitätsklasse an, sofern das Master-Theorem angewendet werden kann.

a) $T(n) = 8 \cdot T(\frac{n}{4}) + 2^n$

b) $T(n) = 64 \cdot T(\frac{n}{4}) + (n^2 + 1) \cdot (n + 7)$

c) $T(n) = 27 \cdot T(\frac{n}{3}) + n \cdot (n^2 \log(n) + n)$

d) $T(n) = 16 \cdot T(\frac{n}{4}) + n \log(n) + n$

e) $T(n) = T(\frac{n}{3}) + f(n)$, wobei $f(n) = \begin{cases} 3n + 2^{3n} & \text{falls } n \in \{2^i | i \in \mathbb{N}\} \\ 3n & \text{sonst} \end{cases}$

Aufgabe 2 (Rekursionsgleichungen): 5+7+8+10=30 Punkte

Wir betrachten folgenden Algorithmus.

```
1  int foo(int A[], int start, int end) {
2      int n = end - start
3      if (n == 0) return 0;
4      if (n == 1) return A[start];
5      int pos1 = start + ⌈ n/4 ⌉; // runde auf
6      int pos2 = start + ⌈ n/2 ⌉; // runde auf
7      int pos3 = start + ⌈ 3*n/4 ⌉; // runde auf
8      return foo(A,start, pos1)
9           + foo(A, pos1, pos3)
10          + foo(A, pos3, end)
11          + foo(A, start, pos2)
12          + foo(A, pos2, end);
13 }
```

a) Geben Sie die (rekursiven) Aufrufe des Algorithmus beim Aufruf von `foo([0,1,2,3,4], 0, 5)` an. Geben Sie dazu für jeden Aufruf von `foo` die Arrayeinträge `A[start], A[start + 1], ..., A[end - 1]` an.

b) Geben Sie eine Rekursionsgleichung $T(n)$ an, welche in Abhängigkeit der Länge $n = \texttt{end} - \texttt{start}$ die Anzahl der rekursiven Aufrufe beschreibt. Vereinfachen Sie die rekursive Gleichung, indem Sie ⌊Gauss⌋ ⌈Klammern⌉ weglassen. Geben Sie auch den Basisfall bzw. die Basisfälle an

c) Zeichnen Sie einen Rekursionsbaum für Ihre Rekursionsgleichung aus **b)**. Dieser soll mindestens die obersten 3 Ebenen (inklusive Wurzel) enthalten, sowie Ebenen für die Blätter.

d) Wir wollen nun experimentell die Rekursionsgleichung aus **b)** mit der tatsächlichen Anzahl der rekursiven Aufrufe vergleichen. Implementieren Sie dazu den Algorithmus in einer beliebigen Programmiersprache und fügen Sie zusätzlich eine Zählvariable ein, welche die Anzahl der Aufrufe von `foo` zählt.

Geben Sie jeweils für die Eingabelängen $n \in \{4, 5, 32, 33, 64, 65\}$ an, welchen Wert $T(n)$ hat und wie oft `foo` tatsächlich aufgerufen wird. Woher kommen die Abweichungen? Experimentieren Sie auch mit anderen Eingabelängen. Würden Sie, anhand ihrer experimentellen Ergebnisse, vermuten, dass die Anzahl der tatsächlichen Aufrufe in $\Theta(T(n))$ liegt? Begründen Sie Ihre Antwort!

Hinweise:

- Achten Sie auf korrektes Runden, insbesondere in den Zeilen 5 bis 7 und bei der Berechnung von $T(n)$ (Integer Division sollte vermieden werden).
- Es empfiehlt sich, auch eine Methode zum Ausrechnen von $T(n)$ zu implementieren.
- Für Ihre Lösung müssen Sie keinen Quellcode einreichen.

Aufgabe 3 (Exkurs: Groß-O-Notation in Gleichungen): 6+4+20=30 Punkte

Wir haben in der Vorlesung bisher die Landau-Symbole immer als Menge von Funktionen aufgefasst. Es gibt jedoch auch eine andere Interpretation, die es ermöglicht, die Notation in Gleichungen zu benutzen. Diese werden selten formal eingeführt und stattdessen als Missbrauch von Notation bezeichnet. In diesem Exkurs wollen wir dennoch versuchen, dieser Notation eine sinnvolle Bedeutung zu geben. Wir beschränken uns in dieser Aufgabe auf das Symbol O. Um Verwirrungen mit der Gleichheit von Mengen auszuschließen, definieren wir $=\circ$ unter Termen, die auch den einstelligen Funktionsterm O enthalten können. In Realität wird statt $=\circ$ aber tatsächlich das Symbol $=$ benutzt. Das Symbol $=\circ$ ist als eine so genannte *Einweg-Gleichheit*[1] zu sehen. Ein Term τ kann nun so verstanden werden, dass wir jedes vorkommen eines $O(g(n))$ in τ mit einer Funktion $f(n) \in O(g(n))$ ersetzen. Falls wir ein Term $O(\tau)$ haben, so müssen wir erst τ in eine Funktion verwandeln, bevor wir $O(\tau)$ auswerten können. Wir sagen nun zwei Terme τ_1 und τ_2 sind gleich, geschrieben $\tau_1 =\circ \tau_2$, wenn es für jede Ersetzung aller O Subterme im linken Teil der Gleichung τ_1 auch eine Ersetzung aller O Subterme im rechten Teil der Gleichung τ_2 gibt, sodass die Terme zwei (echt) gleiche Funktionen bilden. Die Einweg-Gleichheit $=\circ$ entspricht damit der Teilmengenrelation $\subseteq$ in der Mengeninterpretation. Zum Beispiel:

$$(n+1)^2 =\circ n^2 + O(n) \text{ gilt,}$$

da $(n+1)^2 = n^2 + 2n + 1$ und $2n + 1 \in O(n)$.

$$O((n+1)^2) =\circ n^2 + O(n) \text{ gilt } \textbf{nicht},$$

da es die Ersetzung $2n^2 \in O((n+1)^2)$ gibt. Wir brauchen also eine Funktion $f(n) \in O(n)$, sodass $2n^2 = n^2 + f(n)$. Dafür muss $f(n) = n^2$ sein. Jedoch ist $n^2 \notin O(n)$.

a) Zeigen oder widerlegen Sie die folgenden Gleichungen:

i) $n^2 \log_2(O(n^5)) =\circ O(n^2 \log(n))$

ii) $O(n^3 + O(n^2)) =\circ O(n^3)$

[1] Siehe Donald Knuth, The Art of Computer Programming Vol 1, Mai 1997, Seite 107-110

iii) $O(n^4) =\circ\ n^4 + n^3 + n^2 + n$

b) Zeigen Sie (mit einem Gegenbeispiel), dass $=\circ$ nicht symmetrisch ist.

c) Häufig sieht man die Notation $2^{O(f(n))}$ statt $O(2^{f(n)})$. Wir wollen in diesen Aufgaben nun analysieren, warum dies häufig getan wird. Zeigen Sie dazu, dass für $f(n) \in \Omega(1)$ folgendes gilt:

i) $2^{O(f(n))} =\circ\ O(2^{O(f(n))})$

ii) $O(2^{O(f(n))}) =\circ\ 2^{O(f(n))}$

iii) $O(2^{f(n)}) =\circ\ O(2^{O(f(n))})$

iv) $2^{O(f(n))} \neq\circ\ O(2^{f(n)})$

Hinweise:

- Sie können für die Beweise dieser Aufgabe natürlich auf die Mengeninterpretation zurückgreifen.
- Sie dürfen nutzen, dass $\tau =\circ\ O(\tau)$ für alle Terme τ gilt.