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

Übung 1

Hinweise:

- Die Übungsblätter können in 3er Gruppen bearbeitet werden. Übung 1 soll bis zum 14.04.2020 um 8:00 Uhr online in RWTHmoodle abgegeben werden. Abgaben alleine oder zu zweit sind nicht zulässig. Die Abgabepartner müssen im gleichen Tutorium sein.
- 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 Klausurzulassung relevant ist nur das Bestehen der Präsenzübung, welche am 29. Mai stattfinden wird.
- 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.

Aufgabe 1 (Anmelden): ★10+10+10+10=40 Punkte

Um Zugang zu den Lernmaterialien zu bekommen und um einer Tutorgruppe zugeteilt zu werden, melden Sie sich jetzt sofort und **vor dem 8. April** über RWTHonline zur Veranstaltung *Datenstrukturen und Algorithmen (Übung)* an. Gehen Sie dazu wie folgt vor:

a) Loggen Sie sich auf `https://online.rwth-aachen.de` ein.

b) Suchen Sie unter Lehrveranstaltungen nach *Datenstrukturen und Algorithmen* oder klicken Sie hier.

c) Klicken Sie auf den Button *LV-Anmeldung*.

d) Vergeben Sie Präferenzen für Tutoriumstermine und vervollständigen Sie die Anmeldung.

Hinweise:

- Aus technischen Gründen wird der Zugriff zum RWTHmoodle Lernraum erst nach dem 8.4. freigeschaltet. Vorher können Sie die Vorlesungsmaterialien und weitere Informationen auf `https://moves.rwth-aachen.de/teaching/ss-20/datenstrukturen-und-algorithmen/` finden.
- Die Anmeldung zur Vorlesung bzw. Globalübung in RWTHonline ist nicht zwingend notwendig und hat keinen weiteren Effekt.
- Die Veranstaltung *Algorithmen und Datenstrukturen (Service)*, welche ebenfalls in RWTHonline zu finden ist, hat nichts mit der Veranstaltung *Datenstrukturen und Algorithmen* zu tun.
- Die Anmeldung zur Klausur ist erst ab dem 1. Mai möglich.

Aufgabe 2 (Vollständige Induktion): 10+10=20 Punkte

Beweisen Sie mithilfe von vollständiger Induktion,

a) dass für alle $a, n \in \mathbb{N}$ gilt

$$\sum_{i=0}^{n}(a \cdot i) = a \cdot \sum_{i=0}^{n} i,$$

b) dass es für natürliche Zahlen $2 \leq n \in \mathbb{N}$ die folgende Darstellung gibt:

$$n = 2a + 3b \text{ für } a, b \in \mathbb{N}, a + b > 0.$$

Hinweise:

- Aus $a + b > 0$ folgt, dass $a > 0$ oder $b > 0$ gilt.
- Wie in allen Informatikvorlesungen gilt auch hier: $0 \in \mathbb{N}$.

Aufgabe 3 (Datenstrukturen): 2+2+6=10 Punkte

Wir wollen uns in dieser Übung mit verschiedenen Datenstrukturen beschäftigen und ihre Unterschiede diskutieren. Dazu betrachten wir die drei Strukturen k-Tupel, Array und Liste.

a) Nennen Sie mindestens eine Gemeinsamkeit von k-Tupel, Array und Liste.

b) Nennen Sie mindestens einen Unterschied zwischen Array und Liste.

c) Datenstrukturen haben stets auch bestimmte Zugriffsoperationen. Nennen Sie die Operationen, die Sie von den drei Strukturen jeweils erwarten würden.

Aufgabe 4 (Laufzeitanalyse): 4+4+16+6=30 Punkte

Gegeben sei ein Algorithmus, der für ein Array von Integern überprüft, ob die Einträge aufsteigend sortiert sind:

```
1  bool isSorted(int [] E) {
2      int i = 1;
3      while (i < E.length) {
4          if (E[i - 1] > E[i]) {
5              return false;
6          }
7          i++;
8      }
9      return true;
10 }
```

Bei Betrachtung der Laufzeit wird angenommen, dass die Vergleiche in den Zeilen 3 und 4 jeweils eine Zeiteinheit benötigen. Die Laufzeit aller weiteren Operationen wird vernachlässigt.
Sei $n \in \mathbb{N}$ die Länge des Arrays E.

a) Bestimmen Sie in Abhängigkeit von n die Best-case Laufzeit $B(n)$. Begründen Sie Ihre Antwort. Geben Sie für jedes n ein Beispielarray an, bei dem diese Laufzeit erreicht wird.

b) Bestimmen Sie in Abhängigkeit von n die Worst-case Laufzeit $W(n)$. Begründen Sie Ihre Antwort. Geben Sie für jedes n ein Beispielarray an, bei dem diese Laufzeit erreicht wird.

c) Bestimmen Sie in Abhängigkeit von n die Average-case Laufzeit $A(n)$. Begründen Sie Ihre Antwort. Hierzu nehmen wir für $n \geq 2$ folgende Verteilung der Eingaben an:

- $\Pr(\mathtt{E}[0] = 1) = 1$,

- $\Pr(\texttt{E}[\texttt{n}-1]=0) \;=\; 1$ und
- für alle $\texttt{i} \in \{1,\ldots,n-2\}$ gilt: $\Pr(\texttt{E}[\texttt{i}]=0) \;=\; \Pr(\texttt{E}[\texttt{i}]=1) \;=\; 0.5$.

Der erste Eintrag ist also immer 1 und der letzte Eintrag ist immer 0. Die übrigen Einträge sind mit gleicher Wahrscheinlichkeit entweder 0 oder 1. Insbesondere ist die Wahrscheinlichkeit, dass das Array aufsteigend sortiert ist immer 0. Für $n < 2$ ist die Verteilung der Eingaben nicht relevant.

d) Geben Sie einen äquivalenten Algorithmus an, dessen Average-case Laufzeit ab einer gewissen Eingabelänge kleiner ist. Nehmen Sie hierzu die gleiche Wahrscheinlichkeitsverteilung der Eingaben wie bei **c)** an. Begründen Sie Ihre Antwort kurz. Sie müssen nicht die Average-case Laufzeit ihres Algorithmus berechnen.

Hinweise:

- Wir gehen davon aus, dass die Indizierung von Arrays mit 0 beginnt.
- Geben Sie die geforderten Laufzeiten in geschlossener Form (d.h. ohne Summenzeichen, $\Pr(\cdots)$ oder ähnliches) an.
- Wir nennen zwei Algorithmen äquivalent, wenn sie mit der gleichen Eingabe die gleiche Ausgabe produzieren. Hierzu zählen auch Eingaben, die wir bei der Average-case Analyse nicht berücksichtigen!