
Algorithieren und Programmieren

Sommersemester 2019

Prof. Dr. rer. nat. habil. Petra Hofstedt

Sven Löffler M. Sc.

Sonja Breuß, Deborah Buckenauer, Johannes Kuhn, Julius Schöning, Carlo Bückert



Brandenburgische
Technische Universität
Cottbus - Senftenberg

Übungsblatt 6

Abgabedatum: 29.05.2019

Hinweise

- Beachten Sie, dass die Tutoren unkommentierte Programme nicht als Lösung akzeptieren, selbst wenn die Programme richtig funktionieren. Zu einer richtigen Lösung gehören immer sinnvolle Kommentare, deren Umfang der Komplexität des Programms angemessen ist.
- **Halten Sie sich an die in den Übungsblättern vorgegebenen Namen von Funktionen und Dateien, Funktionstypen (Typsignaturen), Reihenfolge der Parameter und verwenden Sie - sofern vorhanden - die Vorgaben!**
- Auf den Übungsblättern finden Sie einige Haskell-Quelltextfragmente. Diese sind der besseren Lesbarkeit wegen unter Nutzung einiger mathematischer Sonderzeichen wiedergegeben.
- Für diese Veranstaltung wird die Verwendung der **Haskell-Plattform** (<https://www.haskell.org/platform/>) empfohlen.
- Als **Tutoriumsaufgabe** markierte (Teil-)aufgaben werden in den Übungen ausführlicher besprochen. Die schriftliche bzw. elektronischen Lösungen müssen jedoch trotzdem mit abgegeben werden. Bitte schauen Sie sich diese Aufgaben im Vorfeld der Übung an und bereiten Sie sich darauf vor.
- Die Abgabe Ihrer Lösungen erfolgt vor Ablauf der Abgabefrist digital über die Moodle-Plattform an Ihren Tutor. Erstellen Sie dazu ein PDF-Dokument, das die Lösungen Ihrer schriftlichen Aufgaben enthält. Laden Sie dieses PDF-Dokument und die erzeugten Haskell-Dateien, mit den in den Aufgaben vorgegebenen Namen, bei Moodle hoch.

Informationsquellen

- Sie finden unter <http://haskell.org/> sehr viele Informationen über die Programmiersprache Haskell. Von besonderem Interesse sind dabei sicherlich die Übersicht über zahlreiche online verfügbare Haskell-Tutorials (<http://haskell.org/haskellwiki/Tutorials>) sowie die Suchmaschine Hoogle (<http://haskell.org/hoogle/>) für die Haskell-API, die Ihnen mit zunehmender Haskell-Erfahrung wertvolle Dienste leisten wird.

Sie können maximal **(8 Punkte)** mit diesem Übungsblatt erreichen.

Aufgabe 1 (Aufwand)

3 Punkte

Für jeweils 3 gelöste Aufgaben gibt es einen Punkt.

Geben Sie für die folgenden Haskell-Funktionen die Aufwände T^{worst} und T^{best} an. Gehen Sie bei der Bestimmung der Aufwände analog zu den in der Vorlesungen präsentierten Beispielen vor. Geben Sie ebenfalls die Aufwandklasse an.

Beschreiben Sie zudem kurz, was die einzelnen Funktionen machen.

1. `fun1 :: el → [el] → [el]`
`fun1 el liste = el : liste`

2. (Tutoriumsaufgabe)

`fun2 :: [el] → el → [el]`
`fun2 [] el' = [el']`
`fun2 (el:els) el' = el : fun2 els el'`

3. `fun3 :: Integer → Integer → Integer`
`fun3 x y`
 | `(x > y) && (y > 0) = fun3 (x - y) y`
 | `otherwise = x`

4. `fun4 :: Integer → Bool`
`fun4 x`
 | `fun3 x 2 == 0 = True`
 | `otherwise = False`

5. `fun5 :: Int → [a] → [a]`
`fun5 0 _ = []`
`fun5 n (x : xs) = x : fun5 (n - 1) xs`

6. `fun6 :: [a] → [b] → [(a, b)]`
`fun6 [] _ = []`
`fun6 _ [] = []`
`fun6 (x : xs) (y : ys) = (x, y) : fun6 xs ys`

7. `fun7 :: (Ord el) ⇒ [el] → el`
`fun7 [el] = el`
`fun7 (el : els) =`
 `let`
 `vonRestliste = fun7 els`
 `in`
 `if el > vonRestliste`
 `then el`
 `else vonRestliste`

8. `fun8 :: (Eq el) ⇒ el → [el] → [el]`
`fun8 _ [] = error "nicht da"`
`fun8 el0 (el : els)`
 | `el0 == el = els`
 | `otherwise = el : fun8 el0 els`

9. `fun9 :: Ord el ⇒ [el] → [el]`
`fun9 [] = []`
`fun9 liste =`
 `let`
 `kleinstes = fun7 liste`
 `in`
 `kleinstes : fun9 (fun8 kleinstes liste)`

Aufgabe 2 (Aufwandsklassen)

2 Punkte

1. Beweisen Sie oder widerlegen Sie die folgenden Beziehungen:

a) (**Tutoriumsaufgabe**) $(n + 1)^2 \in O(n^2)$

b) $\log(n) + n^2 + n^4 \in O(n^5)$

c) $3n * \log(n) + 7n \in O(n * \log(n))$

d) $|\sin(n)| \in O(1)$

e) (**Tutoriumsaufgabe**) $3^n \in O(2^n)$

f) $3^n \in 2^{O(n)}$

2. Sortieren Sie die folgenden Aufwandsklassen bezüglich der Inklusionsrelation \subseteq :

$$O(\sqrt{n}), O(n^3), O(n^2), O(n * \log(n)), O(2^n), O(n^2 * \log(n)), O(n), O(\log(n)), O(21)$$

Aufgabe 3 (Termination)

3 Punkte

Es gibt für jede Teilaufgabe jeweils einen Punkt.

1. Implementieren Sie die Funktion `bubbleSort :: (Ord e1) => [e1] -> [e1]`, welche eine Liste übergeben bekommt und eine mittels BubbleSort-Algorithmus sortierte Liste zurückgibt. Beweisen Sie, dass Ihre Implementierung für beliebige Eingabelisten terminiert.

2. Implementieren Sie die Funktion `selectionSort :: (Ord e1) => [e1] -> [e1]`, welche eine Liste übergeben bekommt und eine mittels SelectionSort-Algorithmus sortierte Liste zurückgibt.

Beweisen Sie, dass Ihre Implementierung für beliebige Eingabelisten terminiert. Beweisen Sie dazu zunächst die Termination der Hilfsfunktionen `minimum :: (Ord a) => [a] -> a` und `loesche :: (Ord a) => a -> [a] -> [a]`.

3. Beweisen Sie, dass die im Folgenden angegebene Haskell-Funktion `ackermann` für positive Argumente terminiert:

`ackermann :: Integer -> Integer -> Integer`

`ackermann 0 m = m + 1`

`ackermann n 0 = ackermann (n - 1) 1`

`ackermann n m = ackermann (n - 1) (ackermann n (m - 1))`