Institut für Programmstrukturen und Datenorganisation
Prof. Dr.-Ing. Gregor Snelting — gregor.snelting@kit.edu
Prof. Dr. Ralf Reussner — reussner@kit.edu

Programmierparadigmen – WS 2021/22

https://pp.ipd.kit.edu/lehre/WS202122/paradigmen/uebung

Blatt 3: Streams, Laziness

Abgabe: 12.11.2021, 14:00
Besprechung: 15.11. – 16.11.2021

Reichen Sie Ihre Abgabe bis zum 12.11.2021 um 14:00 in unserer Praktomat-Instanz unter https://praktomat.cs.kit.edu/pp_2021_WS ein.

1 Streams [Klausuraufgabe vom SS 2011] [7 Punkte]

Geben Sie ihre Lösung als Modul `Fibs` ab.

1. Definieren Sie die unendliche Liste `fibs :: [Integer]` aller Fibonacci-Zahlen:

   ```
   [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ... ]
   ```

   Überlegen Sie sich, welche Zeitkomplexität ein Zugriff auf das n-te Element hat (angenommen, Ganzzahladdition lässt sich in konstanter Zeit durchführen).

   Hinweis: Mittels `zipWith` lässt sich `fibs` sehr kompakt definieren.

2 Collatz-Vermutung

Geben Sie ihre Lösung als Modul `Collatz` ab.

Die Collatz-Vermutung[1] ist ein ungelöstes mathematisches Problem, in dessen Zentrum eine Familie an Zahlenfolgen steht, deren Folgenglieder durch eine einfache Vorschrift definiert sind:

$$a_{n+1} = \begin{cases} \frac{a_n}{2} & \text{falls } a_n \text{ gerade} \\ 3a_n + 1 & \text{sonst} \end{cases}$$

Berechnet man die Folgenglieder für einige Startwerte a_0, so stellt man fest, dass irgendwann die 1 erreicht wird. Die Collatz-Vermutung besagt, dass dies für alle positiven ganzen Zahlen gilt.

1. Definieren Sie eine Funktion `collatz :: Int -> [Int]`, sodass `collatz a0` für den Startwert `a0` die unendliche Liste aller Folgenglieder berechnet.
   Hinweis: Verwenden Sie `iterate` mit geeigneter Hilfsfunktion.

2. Definieren Sie eine Funktion `num :: Int -> Int`, sodass `num m` für den Startwert `m` das kleinste n mit $a_n = 1$ bestimmt. Es ist also `num 4 == 2`, denn $a_0 = 4$, $a_1 = 2$ und $a_2 = 1$.

[1] Siehe auch https://xkcd.com/710/.

3. Definieren Sie eine Funktion `maxNum :: Int -> Int -> (Int, Int)` zur Suche nach dem Maximum von `num` auf einem Intervall. Es soll also `maxNum a b` ein Tupel `(m, num m)` liefern, sodass `num m` maximal ist für alle $m \in [a, b]$.
   Für welchen Wert $m \in [1, 1000]$ ist `num m` maximal und wie groß ist `num m`?

3 Stream-Kombinatoren [Klausuraufgabe vom WS 2010/11] [15 Punkte]

Geben Sie Ihre Lösung als Modul `Merge` ab.

1. Definieren Sie eine Haskellfunktion [5 Punkte]

   ```
   merge :: [Integer] -> [Integer] -> [Integer],
   ```

   die zwei sortierte, möglicherweise unendliche Listen zu einer sortierten, möglicherweise unendlichen Liste vereinigt. Verwenden Sie Pattern Matching.

   Hinweis: Für die Bearbeitung dieses Übungsblattes können Sie Ihre Implementierung von `merge` von Blatt 1 wiederverwenden, falls diese auch für unendliche Listen funktioniert.

2. Definieren Sie eine Haskellfunktion [10 Punkte]

   ```
   primepowers :: Integer -> [Integer],
   ```

   die für einen gegebenen Parameter n die unendliche Liste der ersten n Potenzen aller Primzahlen berechnet, aufsteigend sortiert. D.h., `primepowers n` enthält in aufsteigender Reihenfolge genau die Elemente der Menge

   $$\{p^i \mid p \text{ Primzahl}, \ 1 \leq i \leq n\}.$$

   Verwenden Sie die unendliche, aufsteigend sortierte Liste `primes :: [Integer]` aller Primzahlen aus der Vorlesung *(musste in der Klausur nicht mit angegeben werden)* und `merge`.
   Hinweis: `a ^ b` berechnet die `b`-te Potenz von `a` in Haskell.