Prof. Bernd Krieg-Brückner
Axel Lankenau
Shi Hui
Kolyang
Christoph Lüth
Lutz Zegartowski

6. Übungsblatt

Ausgabe: 21.01.1999
Bearbeitungszeit: drei Praktikumstermine

---

1 *Arithmetik leicht gemacht* — *Wichtung: 40%*

Eine *Gruppe* ist ein mathematisches Gebilde, welches das Rechnen in anderen Datentypen als den ganzen Zahlen erlaubt. Gruppen wurden vor hundertachtzig Jahren, in den Nachwirren der französischen Revolution, von dem knapp zwanzigjährigen französischen Computerpionier Evariste Galois erfunden, der kurz danach in einem Duell erschossen wurde.

Implementieren Sie bitte eine Haskelltypklasse `Group` mit folgende Basisrechenoperationen:

```
one   :: a
inv   :: a -> a
plus  :: a -> a -> a
minus :: a -> a -> a
```

In den ganzen Zahlen, `one` wäre 1, `inv` die Negation, und `plus` und `minus` die Addition und Subtraktion. Welche der Operationen können durch *default*-Definitionen implementiert werden?

In einer Gruppe kann man leider nicht multiplizieren. Erweitern Sie deshalb Gruppen zu Ringen, indem Sie eine Unterklassen `Ring` mit folgenden Operationen definieren:

```
mult  :: a -> a -> a
pot   :: a -> Int-> a
```

In den ganzen Zahlen entsprechen diese der Multiplikation und Potenzierung. Welche von diesen Operationen können durch *default*-Definitionen implementiert werden?

Instantiieren sie die Klasse `Ring` bitte mit ganzen Zahlen, und mit Matrizen. Dafür definieren Sie einen Typ `Matrix` als Liste von Listen von ganzen Zahlen, und definieren Sie die nötigen Operationen darauf.

2 *Nicht immer, aber immer öfter...* — *Wichtung: 20%*

Gegeben ist eine Liste von Elementen deren Typ Instanz der `Eq`-Typklasse ist. Gesucht ist eine Funktion, die eine absteigend sortierte Häufigkeitsverteilung der Elemente der Liste erstellt. Implementieren Sie die Funktion

```
freqspec :: Eq a => [a] -> [(a,Int)]
```

und testen Sie sie mit folgenden Eingaben:

```
freqspec [31,28,31,30,31,30,31,31,30,31,30,31]
freqspec "My Mom always said, life is like a box of chocolate,
          you never know what you gonna get!"
```

Das Ergebnis des ersten Aufrufs ist dabei:

```
[(31,7), (30,4), (28,1)]
```

Erläuterung: Die `31` ist das häufigste Element (sieben Vorkommen), die `28` das seltenste (ein Vorkommen).

$\boxed{3}$ *Auflösung folgt.* *Wichtung: 40%*

Ein *lineares Gleichungsystem* ist ein System der Form

$$
\begin{aligned}
-x_1 + 8x_2 + 3x_3 &= 2 \\
2x_1 + 4x_2 - x_3 &= 1 \\
-2x_1 + x_2 + 2x_3 &= -1
\end{aligned}
\tag{1}
$$

Vor gut zweihundert Jahren ersann der deutsche Computerpionier C. F. Gauß ein Verfahren zur Lösung von linearen Gleichungssystemen dieser Form. Dieses beruht auf der Tatsache, daß die Lösungsmenge des Gleichungssystems nicht ändert, wenn von einer Zeile ein Vielfaches einer anderen abgezogen oder hinzuaddiert wird. Dadurch läßt sich ein Gleichungssystem in die sogenannte "Dreiecksform" transformieren, in dem die Variablen der Reihe nach ausgerechnet werden können.

Wir werden dieses Verfahren an dem Gleichungssystem 1 demonstrieren. Für eine übersichtlichere Darstellung schreiben wir nur die Koeffizienten:

$$
\begin{array}{rrrrl}
-1 & 8 & 3 & 2 & \\
2 & 4 & -1 & 1 & \\
-2 & 1 & 2 & -1 & \\
 & & & & \\
-1 & 8 & 3 & 2 & \\
 & 20 & 5 & 5 & \text{1. Zeile mult. mit 2, addiert zur 2. Zeile oben} \\
 & -15 & -4 & -5 & \text{1. Zeile mult. mit } -2\text{, addiert zur 3. Zeile oben} \\
 & & & & \\
-1 & 8 & 3 & 2 & \\
 & 20 & 5 & 5 & \\
 & & -\frac{1}{4} & -\frac{5}{4} & \text{2. Zeile multipliziert mit } \frac{3}{4}\text{, addiert zur 3. Zeile oben}
\end{array}
$$

Ein System in Dreiecksform läßt sich rekursiv auflösen: aus der letzen Gleichung $-\frac{1}{4}x_3 = -\frac{5}{4}$ ergibt sich $x_3 = 5$, welches in die 2. Zeile darüber eingesetzt $20x_2 + 25 = 5$ ergibt, also $x_2 = -1$, und schließlich nach Einsetzen in die erste Gleichung $x_1 = 5$.

Sowohl die Umformung wie auch die Auflösung können fehlschlagen; in diesem Fall hat das Gleichungssystem entweder keine oder unendlich viele Lösungen.

Wir stellen ein Gleichungssystem als Liste von Listen dar, also z.B ist das Gleichungssystem oben die Liste

```
[[-1,8,3,2], [2,4,-1,1], [-2,1,2,-1]]
```

Implementieren Sie eine Funktion

```
triangulize :: Fractional a=> [[a]]-> [[a]]
```

die ein lineares Gleichungssystem in die Dreicksform überführt. Dazu ist es hilfreich, Funktionen zur Vektoraddition und Skalarmultiplikation zu implementieren:

```
vecAdd   :: Num a=> [a] -> [a]-> [a]
smult    :: Num a=> a -> [a]-> [a]
```

Mit Hilfe von `triangulize` implementieren Sie eine Funktion

```
solve :: Fractional a => [[a]] -> [a]
```

die ein Gleichungssystem in Dreiecksform löst. Dazu muß geprüft werden, ob die Eingabe aus n Listen der Länge $n + 1$ besteht; ansonsten wird eine Fehlermeldung ausgegeben.