Metody programowania 2016

Lista zadań nr 10

Na zajęcia 12 i 16 maja 2016

Zadanie 1 (1 pkt). Zdefiniuj listę

```
nat2 :: [(Integer,Integer)]
```

zawierającą wszystkie pary nieujemnych liczb całkowitych w kolejności określonej przez — znany z dowodu tw. Cantora — porządek

$$(x_1, y_1) \prec (x_2, y_2) \iff x_1 + y_1 < x_2 + y_2 \lor (x_1 + y_1 = x_2 + y_2 \land x_1 < x_2)$$

tj. takiej, że

```
nat2  =  [(0,0),(0,1),(1,0),(0,2),(1,1),(2,0),
          (0,3),(1,2),(2,1),(3,0),(0,4),...]
```

Wskazówka: definicja powinna zmieścić się w jednym wierszu długości mniejszej niż 45 znaków. Zauważ, że suma współrzędnych punktów leżących na tej samej przekątnej jest stała.

Zadanie 2 (1 pkt). Zaprogramuj w Haskellu algorytm Mergesort. Napisz dwie funkcje:

```
halve :: [a] → ([a],[a])
merge :: Ord a => ([a], [a]) → [a]
```

tak, by można je było wykorzystać w programie:

```
msort [] = []
msort [x] = [x]
msort xs =
   merge . cross (msort,msort) . halve $ xs
```

gdzie

```
cross :: (a → c, b → d) → (a,b) → (c,d)
cross (f,g) = pair (f . fst, g . snd)
pair :: (a → b, a → c) → a → (b,c)
pair (f,g) x = (f x, g x)
```

Zadanie 3 (1 pkt). Zaprogramuj, a następnie wykorzystaj funkcję

```
merge_unique :: Ord a => [a] -> [a] -> [a]
```

do rozwiązania problemu 2-3-5 Dijkstry: zdefiniuj nieskończony rosnący ciąg liczb całkowitych

```
d235 :: [Integer]
```

zawierający liczbę 1 i taki, że jeśli n jest elementem tego ciągu, to są nimi też $2n$, $3n$ i $5n$. *Uwaga:* w odróżnieniu od funkcji wykorzystanej w algorytmie Mergesort, ta funkcja `merge_unique` powinna usuwać duplikaty.

Zadanie 4 (1 pkt). Aby wyeliminować kopiowanie podczas dzielenia w algorytmie Mergesort listy na połowy możemy zdefiniować ogólniejszą funkcję „posortuj n początkowych elementów podanej listy":

```
msortn :: Ord a => [a] → Int → [a]
```

Np.

```
msortn [7,4,6,9,3,5,2] 4  =  [4,6,7,9]
```

Zaprogramuj ten algorytm. Użyj funkcji `length` by otrzymać zwykły algorytm sortowania.

Zadanie 5 (1 pkt). Niech

```
class Monoid a where
   (***) :: a -> a -> a
   e :: a
infixl 6 ***
```

będzie klasą monoidów (półgrup z jednością). Chociaż nie możemy tego wyrazić wprost w Haskellu, jednak zakładamy, że metody tej klasy spełniają następujące aksjomaty:

$$\forall a :: a.\; e *** a = a$$

$$\forall a :: a.\; a *** \mathtt{e} = a$$

$$\forall a, b, c :: a.\; (a *** b) *** c = a *** (b *** c)$$

Niech

$$\begin{aligned} a^0 &= \mathtt{e} \\ a^{n+1} &= a *** a^n,\; n \in \mathbb{N} \end{aligned}$$

tj. w notacji Haskellowej:

```
infixr 7 ^^^
(^^^) :: Monoid a => a -> Integer -> a
a ^^^ 0 = e
a ^^^ (n+1) = a *** a ^^^ n
```

Powyższe równości są dobrą specyfikacją, ale nie implementacją potęgowania w monoidzie. Aby szybko wyliczać potęgę korzystamy raczej z następujących wzorów:

$$\begin{aligned} a^0 &= \mathtt{e} \\ a^n &= \begin{cases} a *** a^{n/2} *** a^{n/2}, & n \text{ is odd,} \\ a^{n/2} *** a^{n/2}, & \text{otherwise.} \end{cases} \end{aligned}$$

Wyprowadź powyższe równości ze specyfikacji funkcji potęgowania. Przepisz powyższe równości w Haskellu. Zadbaj o to, by funkcja zachowywała się rozsądnie dla ujemnych wykładników. Skorzystaj z powyższej definicji by obliczyć $1234567890^{1234567890}$ mod 9876543210. Zdefiniuj monoid liczb całkowitych z dodawaniem i zerem.

Zadanie 6 (1 pkt). Poniższa deklaracja jest świetną specyfikacją, ale nie implementacją ciągu Fibonacciego:

```
fib :: Num a => Integer -> a
fib 0 = 1
fib 1 = 1
fib (n+2) = fib (n+1) + fib n
```

Zwykle używamy takiej implementacji:

```
fib :: Num a => Integer -> a
fib = aux 1 1 where
   aux f1 f2 n =
      case n `compare` 0 of
         LT -> 0
         EQ -> f1
         GT -> aux f2 (f1 + f2) (n-1)
```

która jednak okazuje się niewystarczająco dobra, by policzyć np.

$$\texttt{fib 123456789}.$$

Istnieje jednak szybszy sposób. Udowodnij, że dla dowolnego $n \in \mathbb{N}$ mamy

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n \quad = \quad \begin{bmatrix} \texttt{fib}\,(n-1) & \texttt{fib}\,n \\ \texttt{fib}\,n & \texttt{fib}\,(n+1) \end{bmatrix},$$

przy czym rozważamy tu zwyczajne potęgowanie macierzy 2×2. Niech więc

```
data Mtx2x2 a = Mtx2x2 a a a a
```

gdzie wyrażenie `Mtx2x2` a_{11} a_{12} a_{21} a_{22} reprezentuje macierz

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}.$$

Zainstaluj typ `Mtx2x2` w klasie `Monoid` i wykorzystaj powyższą równość do zdefiniowania efektywnej implementacji ciągu Fibonacciego.

Zadanie 7 (1 pkt). Udowodnij, że

$$\texttt{map}\,f\ .\ \texttt{concat} \quad = \quad \texttt{concat}\ .\ \texttt{map}\,(\texttt{map}\,f)$$