8. Übung **Informatik A** **WS 07/08**

Klaus Kriegel **Abgabe: 18.12.2007, 12:00 Uhr**

Aufgabe 1: **Laufzeitanalyse** (2 + 3 Punkte)

a) Beweisen Sie mit vollständiger Induktion, dass die folgende Funktion `magic` für Listen der Länge n mindestens exponentielle Laufzeit (genauer $\Omega(2^n)$) hat. Die Laufzeit wird hier durch die Anzahl von Aufrufen der Funktionen `magic` und `max` und gemessen.

```
magic ::  [Int] -> Int
magic [] = 0
magic (x:xs) = max (max x (magic xs)) (max (-x) (magic xs))
```

b) Entwerfen Sie eine andere Implementierung der gleichen Funktion mit linearer Laufzeit und beweisen Sie diese obere Schranke auch mit vollständiger Induktion.

Aufgabe 2: **Strings** (8 Punkte)

Implementieren Sie die folgenden Listenfunktionen
`ignoreDoublings, deleteRepetitions, onlySingles ::  String -> String`.

a) `ignoreDoublings` soll alle Doppel- und Mehrfachzeichen durch ein einzelnes ersetzen, also z.B. für die Eingabe `"abbacxxxax"` den String `"abacxax"` ausgeben.

b) `deleteRepetitions` soll jedes Zeichen, dass im Eingabestring vorkommt, nur einmal in den Ausgabestring setzen also z.B. für die Eingabe `"abbacxxxax"` den String `"abcx"` (oder eine Permutation dieses Strings) ausgeben.

c) `onlySingles` soll nur die Zeichen in den Ausgabestring setzen, die im Eingabestring genau einmal vorkommen, also z.B. für die Eingabe `"abbacxxxax"` den String `"c"` ausgeben.

d) Implementieren Sie die Listenfunktion `countSymbols ::  String -> [(Char,Int)]` mit der für alle in der Liste auftretenden Symbole deren Vielfachheit bestimmt wird, z.B. soll für die Eingabe `"abbacxxxax"` die Liste `[('a',3),('b',2),('c',1),('x',4)]` ausgegeben werden.

Aufgabe 3: **Euklidischer Algorithmus** 5 Punkte

Sie haben die Umkehrung des Euklidischen Algorithmus schon geübt, jetzt soll diese Methode implementiert werden. Die Eingabe sind zwei ganze Zahlen m, n für die man $m \geq n > 0$ voraussetzen kann. Gesucht ist ein Paar (s, t), das die Bedingung $s \cdot m + t \cdot n = ggT(m, n)$ erfüllt. Sie können einen eigenen Lösungsansatz entwickeln oder die folgende Idee realisieren:
Ist `mod m n == 0` so ist $ggT(m, n) = n$ und man kann das Paar $(0, 1)$ ausgeben. Anderenfalls arbeitet der Euklidische Algorithmus rekursiv mit n und $r = mod\ m\ n$ weiter, aber um die Rückrechnung in die Rekursion integrieren zu können, muss man weitere Informationen zwischenspeichern und das kann man mit einem 7-Tupel machen. Ein solches Tupel (a, b, c, d, e, f, g) repräsentiert eine Runde des Euklidischen Algorithmus, wobei a und b die aus der letzten Runde übernommenen Werte sind, $c = mod\ a\ b$ und die Paare (d, e) und (f, g) die Koeffizienten sind, um b bzw. c als Linearkombination von m und n darzustellen, d.h. $b = d \cdot m + e \cdot n$ und $c = f \cdot m + g \cdot n$.

Wir demonstrieren das am Beispiel der Berechnung des $ggT(50, 22)$. Hier startet man mit dem Tupel $(50, 22, 6, 0, 1, 1, -2)$. Die letzten Koeffizienten ergeben sich aus dem Divisionsalgorithmus $50 = 2 \cdot 22 + 6$, umgestellt $6 = 1 \cdot 50 - 2 \cdot 22$. Die ersten 5 Komponenten des Folgetupels sind einfach zu finden, nämlich $(22, 6, 4, 1, -2, \ldots)$. Um nun noch die $4 = mod\ 22\ 6$ als Linearkombination von 36 und 10 darzustellen, muss man $22 = 3 \cdot 6 + 4$ nach $4 = 22 - 3 \cdot 6$ umstellen und dann die Kombinationen von 22 und 6 nutzen:
$4 = 0 \cdot 50 + 1 \cdot 22 - 3 \cdot (1 \cdot 50 - 2 \cdot 22) = -3 \cdot 50 + 7 \cdot 22$.
Es ergibt sich also die Tupelfolge

$$\begin{array}{rrrrrrr}
(\ 50 & 22 & 6 & 0 & 1 & 1 & -2) \\
(\ 22 & 6 & 4 & 1 & -2 & -3 & 7) \\
(\ 6 & 4 & 2 & -3 & 7 & 4 & -9) \\
(\ 4 & 2 & 0 & 4 & -9 & -11 & 25)
\end{array}$$

Wie man sieht, kann man bereits vor der Berechnung des letzten Tupels abbrechen, denn an `mod 4 2 == 0` erkennt man, dass die 2 der gesuchte ggT ist und deren Kombination $2 = 4 \cdot 50 - 9 \cdot 22$ ist bereits Bestandteil des vorletzten Tupels. Mit diesem Ansatz führt man das Problem auf eine rekursive Funktion

```
rekLinComb ::  (Int,Int,Int,Int,Int,Int,Int) -> (Int,Int)
```

zurück.