Ludwig-Maximilians-Universität München WS 2005/2006
Institut für Informatik Blatt 11
Prof. Dr. F. Kröger, Dr. M. Hölzl, F. Hacklinger

Übungen zu Informatik I

Aufgabe 11-1 Permutationen

Eine Liste $m = [m_1, \ldots, m_n]$ heißt eine *Permutation* einer Liste $l = [l_1, \ldots l_n]$, wenn es eine Bijektion $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ gibt, so dass $m_{\pi(i)} = l_i$ für alle $1 \leq i \leq n$. Das folgende SML-Programm überprüft, ob m eine Permutation von l ist:

```
fun member x nil = false
  | member x (y::ys) = x = y orelse member x ys
exception not_found
fun remove x nil = raise not_found
  | remove x (y::ys) = if x = y then ys else y::(remove x ys)
fun perm l m =
    if null l then null m
    else if member (hd l) m then perm (tl l) (remove (hd l) m)
    else false
```

Welche asymptotische Komplexität hat die Funktion *perm*?

Aufgabe 11-2 Kleinstes gemeinsames Vielfaches

Das *kleinste gemeinsame Vielfache* zweier natürlicher Zahlen $m > 0$ und $n > 0$ ist die kleinste positive natürliche Zahl, die sowohl durch m als auch durch n teilbar ist. Das folgende SML-Programm berechnet das kleinste gemeinsame Vielfache von m und n:

```
fun kgv(m,n) =
    let fun loop i =
            if (i mod m) = 0 andalso (i mod n) = 0
            then i
            else loop(i+1)
    in loop m
    end
```

a) Zeigen Sie: *kgv* terminiert für alle $m, n > 0$.

b) Welche asymptotische Komplexität hat *kgv* in Abhängigkeit von m und n im schlechtesten Fall? Begründen Sie Ihre Antwort.
Hinweis: Der schlechteste Fall tritt für voneinander verschiedene Primzahlen m, n ein.

c) Geben Sie ein SML-Programm *kgv2* : **int** $*$ **int** $\to$ **int** an, das das kleinste gemeinsame Vielfache zweier Zahlen mit geringerer asymptotischer Komplexität berechnet als *kgv*.

Aufgabe 11-3 **Revertieren von Listen**

In der Vorlesung wurde die folgende Funktion zum Revertieren von Listen angegeben:

```
fun rev nil = nil
  | rev (x::xs) = (rev xs)@[x];
```

a) Zeigen Sie: Die Funktion *rev* hat Zeitkomplexität $O(n^2)$. (Setzen Sie dabei voraus, dass die Listenkonkatenation `@` : $'a$ **list** $*$ $'a$ **list** $\rightarrow$ $'a$ **list** lineare Zeitkomplexität in der Länge der ersten Eingabeliste hat.)

b) Geben Sie mit einer geeigneten Einbettung eine repetitiv rekursive SML-Funktion

$$rev' : 'a \textbf{ list} \rightarrow 'a \textbf{ list}$$

an, die eine Liste l mit geringerer Zeitkomplexität als *rev* revertiert.

c) Bestimmen Sie die Zeitkomplexität von rev'.

Aufgabe 11-4 **Anzahl der Elemente einer endlichen Menge**

Die SML-Funktion *setsize* ermittelt die Anzahl der Elemente einer endlichen Menge:

```
fun setsize set = if isemptyset(set) then 0
  else let val element = any(set) in
   1 + setsize(delete(element, set))
       end;
```

Geben Sie eine repetitiv rekursive SML-Funktion *setsizerep* an, die die Anzahl der Element einer endlichen Menge durch eine geeignete Einbettung berechnet.

Aufgabe 11-5 **Suchen in binären Bäumen**

(Diese Aufgabe ist etwas schwieriger; für besonders interessierte Studierende gedacht.)

Die SML-Funktion *enthalten2* aus der Vorlesung, Abschnitt 4.7, berechnet, ob ein Element in einem Binärbaum enthalten ist. Geben Sie eine repetitiv rekursive Implementierung dieser Funktion an.