

Übungen zu Informatik I

Aufgabe 12-1 Textnachrichten auf Mobiltelefonen (keine Abgabe)

Mobiltelefone bieten die Möglichkeit, Textnachrichten zu versenden; bei der Eingabe der Nachrichten werden Buchstaben als Zahlen nach folgendem Schema kodiert:

$a, b, c \rightsquigarrow 2$	$d, e, f \rightsquigarrow 3$	$g, h, i \rightsquigarrow 4$	$j, k, l \rightsquigarrow 5$
$m, n, o \rightsquigarrow 6$	$p, q, r, s \rightsquigarrow 7$	$t, u, v \rightsquigarrow 8$	$w, x, y, z \rightsquigarrow 9$

Diese Kodierung definiert eine Abbildung $C : \{a, \dots, z\} \rightarrow \{2, \dots, 9\}$. Eine Ziffernfolge $Z \in \{2, \dots, 9\}^*$ kodiert eine Zeichenreihe $W \in \{a, \dots, z\}^*$ (wir sagen dann: Z ist ein Kode von W), wenn W und Z die gleiche Länge haben und die elementweise Anwendung von C auf W die Ziffernfolge Z liefert. Zum Beispiel kodiert die Ziffernfolge 4636 die Zeichenreihe "info". Diese Kodierung ist nicht eindeutig; um Zweideutigkeiten aufzulösen, verfügen moderne Apparate über ein eingebautes Wörterbuch. Ziel der Aufgabe ist es, die Funktionalität für ein solches Wörterbuch bereit zu stellen.

- Geben Sie einen Datentyp *ziffer* an, der die Menge der Ziffern $\{2, \dots, 9\}$ repräsentiert.
- Wir stellen Zeichenreihen mit Hilfe des Typs **string** und Ziffernfolgen durch den Typ *ziffer list* dar. Geben Sie eine SML-Funktion *kodiere* an, welche zu einer Zeichenreihe Z diejenige Ziffernfolge bestimmt, die Z kodiert.
- Um auf effiziente Weise alle sich im Wörterbuch befindlichen Zeichenreihen zu finden, die von einer Ziffernfolge kodiert werden, verwendet man Bäume, bei denen jeder Knoten 8 Unterbäume besitzt; die zu den Unterbäumen führenden Kanten sind mit $2, \dots, 9$ markiert. Diese Bäume nennen wir O-Bäume. Jeder Knoten K speichert ein Datenelement vom Typ **string list**; dieses Datenelement bezeichnen wir mit $W(K)$. Geben Sie eine Typdefinition für O-Bäume an.
- Für einen Knoten K eines O-Baumes bezeichne $Z(K)$ die Ziffernfolge, die man erhält, wenn man sich von der Wurzel des O-Baumes (auf direktem Wege) zum Knoten K bewegt und dabei die Markierungen aller betretenen Kanten (in der Reihenfolge des Betretens) in einer Ziffernfolge sammelt.
Wir nennen einen O-Baum *schön*, wenn für jeden Knoten K gilt: Die Ziffernfolge $Z(K)$ ist ein Kode für jedes Element der Liste $W(K)$. Ein O-Baum B enthält eine Zeichenreihe S , wenn es einen Knoten K von B gibt, so dass $W(K)$ die Zeichenreihe S enthält.
Geben sie eine Funktion *einfüegen* an, die eine Zeichenreihe in einen O-Baum einfügt. Hierbei soll der resultierende Baum die einzufügende Zeichenreihe enthalten und – falls der Eingabebaum schön war – wieder schön sein.
- Geben Sie eine Funktion *extrahiere* an, die für einen schönen O-Baum B und einer Liste von Ziffern Z die Liste aller in B enthaltenen Zeichenreihen bestimmt, deren Kode Z ist.
- Alternativ können Wörterbücher auch als Listen verwaltet werden. Geben Sie eine Funktion *suche* an, die zu einer Liste L von Zeichenreihen und einer Ziffernfolge Z die Liste derjenigen Zeichenreihen bestimmt, die in L enthalten sind und Kode Z haben.
- Bestimmen Sie die asymptotische Komplexität der Algorithmen *extrahiere* und *suche* in Abhängigkeit von der Größe des Wörterbuchs und der Länge der Ziffernfolge.

Aufgabe 12-2**Ringpuffer**

(8 Punkte)

Um eine Anzahl n von Elementen zyklisch zu durchlaufen, verwendet man einen sogenannten *Ringpuffer*. Die in der Signatur für Ringpuffer enthaltenen Funktionen haben folgende Wirkung: Ein Aufruf `new_buffer([a1, ..., an])` erzeugt einen Ringpuffer mit den Elementen a_1, \dots, a_n (in dieser Reihenfolge). Ist p ein Ringpuffer mit Elementen a_1, \dots, a_n , so gibt der Aufruf `advance(p)` einen Ringpuffer mit den Elementen a_2, \dots, a_n, a_1 zurück. Der Aufruf `value(p)` gibt das Element a_1 zurück.

a) Gegeben sei die folgende Signatur und Implementierung für Ringpuffer:

```
signature BUFSig =
sig
  type 'a buf
  val new_buffer : 'a list -> 'a buf
  val advance    : 'a buf -> 'a buf
  val value     : 'a buf -> 'a
end

structure BUF : BUFSig =
struct
  type 'a buf = 'a list
  fun new_buffer xs = xs
  fun append [] ys = ys
    | append (x::xs) ys = x::(append xs ys)
  fun advance (x::xs) = append xs [x]
  fun value (x::_) = x
end
```

Welche asymptotische Zeitkomplexität haben die Funktionen `new_buffer`, `advance` und `value`?

b) Geben Sie eine Struktur an, die die Signatur `BUFSig` implementiert und bei der die asymptotische Zeitkomplexität aller Funktionen (`new_buffer`, `advance` und `value`) konstant, d.h., $O(1)$ ist.

c) Geben Sie ein SML-Programm an, das die von Ihnen erzeugte Struktur verwendet und einen Ringpuffer erzeugt, der die Elemente 1, 3, 7, 1, 2 in dieser Reihenfolge enthält.

Aufgabe 12-3**Rekursion**

(4 Punkte)

Die Funktion $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ ist folgendermaßen definiert:

$$f(n) = \begin{cases} 3 & \text{falls } n = 0 \\ 2 & \text{falls } n = 1 \\ 1 & \text{falls } n = 2 \\ f(n-1) + 2f(n-2) + 3f(n-3) & \text{sonst} \end{cases}$$

a) Geben Sie ein SML-Programm an, das f mit einer nicht-linearen Rekursion implementiert.

b) Geben Sie ein SML-Programm an, das f mit repetitiver (linearer) Rekursion implementiert.

Bitte beachten Sie, dass wir Ihre Lösung nur dann korrigieren und bewerten können, wenn sie als *Textdatei* abgegeben wird. Als Lösung ist ein *lauffähiges* SML-Programm abzugeben. Nicht zur Lösung gehörende Bemerkungen sind zwischen die Kommentarzeichen (* und *) einzuschließen. Alle hiervon abweichenden Hausaufgaben werden mit 0 Punkten bewertet.

Abgabe: Dienstag, 21.1.2003, 12:00 Uhr.