Ludwig-Maximilians-Universität München
Department "Institut für Informatik"
Prof. Dr. Christian Böhm
Elke Achtert, Arthur Zimek

München, 01.12.2006

Informatik 1

WS 2006/07

Übungsblatt 7: Endrekursion, asymptotische Laufzeit-Komplexität

Besprechung: 11.12.–15.12.2006

Abgabe aller mit **Hausaufgabe** markierten Aufgaben bis Freitag, 08.12.2006, 18:00 Uhr

Aufgabe 7-1 *Endrekursion* **(Hausaufgabe)**

Für eine frühere Übungsaufgabe waren die Funktionen `vorgaenger`, `summe`, `ist_gerade`, `haelfte`, `doppelt` wie in der Datei `7-1.sml` vorgegeben. Damit wurde eine Funktion `produkt' : int * int -> int` definiert, die das Produkt ihrer beiden Argumente so berechnet, dass die Rechenzeit in vielen Fällen nur logarithmisch vom zweiten Parameter abhängt.

Geben Sie in einer Datei `7-1.sml` eine Definition `produkt'' : int * int -> int` der Produkt-Funktion an, die mit rein-funktionalen Mitteln einen iterativen Berechnungsprozess auslöst (mit einer lokalen Hilfsfunktion).

Aufgabe 7-2 *Endrekursion* **(Hausaufgabe)**

In einer früheren Übungsaufgabe wurde die Funktion `quadrat : int -> int` nur mit Hilfe von Addition und Subtraktion definiert.

Geben Sie in einer Kopie der Datei `7-2.sml` eine Definition `quadrat' : int -> int` der Quadrat-Funktion an, die ebenfalls nur mit Hilfe von Addition und Subtraktion das Quadrat einer Zahl mit rein-funktionalen Mitteln durch einen iterativen Berechnungsprozess berechnet.

Alle Hilfsfunktionen sollen lokal sein.

Aufgabe 7-3 *Endrekursion* **(Hausaufgabe)**

In der Vorlesung wurde die Potenz-Funktion effizient implementiert, aber nicht endrekursiv.

Geben Sie in einer Datei `7-3.sml` eine endrekursive Definition `potenz''(a, b)` der Potenz-Funktion an, die mit rein-funktionalen Mitteln durch einen iterativen Berechnungsprozess für $(a, b) \in \mathbb{N} \times \mathbb{N}$ die Zahl a^b berechnet.

Alle Hilfsfunktionen sollen lokal sein.

Aufgabe 7-4 *Komplexität* **(Hausaufgabe)**

Der Binomialkoeffizient von zwei natürlichen Zahlen n, k mit $0 \leq k \leq n$ ist definiert als

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Sie kennen den Binomialkoeffizienten aus der Analysis und vom Lottospielen.

(a) Eine direkte Implementierung der Definition des Binomialkoeffizienten sieht wie folgt aus:

```
fun fak(n) = if n = 0 then 1 else n * fak(n-1);

fun binom_naiv(n, k) =
    fak(n)
    div
    (fak(k) * fak(n-k));
```

In welcher Komplexitätsklasse (asymptotisches Laufzeitverhalten) liegt die Funktion `binom_naiv`? Warum ist diese Implementierung "naiv"? Geben Sie Ihre Überlegungen in einer Datei `7-4a.txt` ab.

(b) Ebenso aus der Analysis ist bekannt, dass für $n \geq k \geq 1$ gilt:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

Für $k = 0$ oder $n = k$ ist $\binom{n}{k} = 1$. Damit ist eine rekursive Definition des Binomialkoeffizienten möglich. Geben Sie in einer Datei `7-4b.sml` eine Implementierung `binom : int * int -> int` des Binomialkoeffizienten gemäß dieser rekursiven Definition an.

(c) In welcher Komplexitätsklasse (asymptotisches Laufzeitverhalten) liegt die Funktion `binom`? Geben Sie Ihre Überlegungen in einer Datei `7-4c.txt` ab.

(d) Eine bessere Implementierung könnte sich auf die folgende Beobachtung stützen. Die eigentliche Definition von $\binom{n}{k}$ läßt sich geschickt kürzen, so dass sich wieder eine rekursive Definition ergibt:

$$\frac{n!}{k!(n-k)!} = \begin{cases} 1 & \text{falls} \quad k = 0 \\ \frac{(n)}{1} \cdot \frac{(n-1)}{2} \cdot \frac{(n-2)}{3} \cdots \frac{(n-k+2)}{(k-1)} \cdot \frac{(n-k+1)}{k} & \text{falls} \quad k > 0 \end{cases}$$

Geben Sie in einer Datei `7-4d.sml` eine Implementierung `binom' : int * int -> int` des Binomialkoeffizienten gemäß dieser (effizienteren) rekursiven Definition an.

(e) In welcher Komplexitätsklasse (asymptotisches Laufzeitverhalten) liegt die Funktion `binom'`? Geben Sie Ihre Überlegungen in einer Datei `7-4e.txt` ab.

Hinweis: Zur Einordnung in eine Komplexitätsklasse genügt hier eine anschauliche oder argumentative Begründung Ihrer Entscheidung. Ein Beweis ist nicht gefordert.