Universität Mannheim
Lehrstuhl für Praktische Informatik IV
Prof. Dr. W. Effelsberg
Christoph Kuhmünch, Gerald Kühne

Praktische Informatik I
WS 1999/2000

Übungsblatt 11 **Ausgabe: Mi, 19.01.00** **Abgabe: Di, 25.01.00, 18 Uhr**

Aufgabe 1: Rekurrenzrelationen [4 Punkte]

Bestimmen Sie die Rekurrenzrelation der Kostenfunktion $T(n,a)$ der folgenden Java-Funktion und geben Sie mit Hilfe des O-Kalküls das asymptotische Verhalten der Kostenfunktion an. Bitte ausführlich begründen!

```
int f (int n, int a) {
  if (n <= 1)
    return 0;
  else
    return (f(n-1,a) + f(n-1, a/2));
}
```

Lösung:

Der Aufwand der Methode f ergibt sich aus der Rekursion in der Return-Anweisung. Hier wird die Funktion f rekursiv mit $n-1$ aufgerufen. Da der Parameter a den Abbruch der Rekursion nicht beeinflußt und auch keine Schleifen von a abhängen, spielt dieser Parameter in der Kostenfunktion $T(n,a)$ keine Rolle.

Die Kostenfunktion wird alleine durch die Rekursion in der return Anweisung der Methode beeinflußt, da der restliche Rumpf von f in konstanter Zeit C ausgeführt werden kann. Falls $n = 1$, kann die Funktion in konstanter Zeit k ausgeführt werden.

Daraus ergibt sich folgende rekursive Definition der Kostenfunktion:

$$T(n,a) = T(n) = \begin{cases} k & : \quad \text{für } n \leq 1 \\ T(n-1)+T(n-1)+c = 2T(n-1)+c & : \quad \text{für } n \geq 2 \end{cases}$$

Abschätzung der geschlossenen Darstellung:

Um eine Vermutung über die geschlossene Darstellung aufstellen zu können, ist es sinnvoll, einige Rekursionsstufen zu betrachten (i gibt die Rekursionstiefe an):

$$\begin{array}{lll} T(n) & = & = 2T(n-1)+c \quad (i=1) \\ & = 2(2T(n-2)+c)+c & = 4T(n-2)+3c \quad (i=2) \\ & = 4(2T(n-3)+c)+3c & = 8T(n-3)+7c \quad (i=3) \\ & = 8(2T(n-4)+c)+7c & = 16T(n-4)+15c \quad (i=4) \\ & \ldots & \end{array}$$

Durch „scharfes Hinsehen“ erkennt man, daß die Kosten proportional zu 2^i verhalten, wobei i die aktuelle Rekursionstiefe angibt.

Behauptung: *Es gilt:*

$T(n,a) = T(n) = 2^i T(n-i) + (2^i - 1)c$

Beweis durch vollständige Induktion über i:

Induktionsanfang $i = 1$:

$T(n) = 2T(n-1) + c = 2^1 T(n-1) + (2^1 - 1)c$

Induktionsannahme es gelte die obige Behauptung für i.

Induktionsschritt i auf $i+1$ für $i \geq 1$:

z.z.: die obige Behauptung gilt auch für $i+1$, d.h. $T(n) = 2^{i+1}T(n-(i+1)) + 2^i c$

$$\begin{aligned} T(n) &= 2^i T(n-i) + (2^i - 1)c \\ &= 2^i(2T(n-(i+1)) + c) + (2^i - 1)c \\ &= 2^{i+1}T(n-(i+1)) + (2^{i+1} - 1)c \end{aligned}$$

Bestimmung des O-Kalküls:

Betrachte nun nur solche n für die gilt: $n = i+1$ bzw. $i = n-1$. Da – wie auch im Beispiel in der Vorlesung – der Zeitaufwand des Algorithmus mindestens linear wächst (müßte eigentlich noch bewiesen werden), gilt für alle $n \leq n_2$:

$T(n) \leq T(n_2) mit n_2 := i+1$, *so daß* $i < n \leq i+1 = n_2$

$$\begin{array}{lllll} T(n) & = 2^i & T(n-i) & +(2^i - 1)c \\ & = 2^{n-1} & T(1) & +(2^{n-1} - 1)c \\ & = 2^{n-1} & k & +(2^{n-1} - 1)c \end{array}$$

Es folgt: $T(i) = O(2^n)$

Aufgabe 2: Komplexitätsberechnung [6 Punkte]

Den drei unten gegebenen Programmstücken gehe folgender Vereinbarungsteil voraus:

```
const int c = 5;
int  s,i,j,n;
```

Bestimmen Sie die Zeitkomplexität dieser Programmstücke (die Additionsoperationen seien in konstanter Zeit durchführbar).

1.
```
s = 0;
for (i = 1; i <= n; i++)
  for (j = 1; j <= 2*n; j++)
    s = s+i+j;
```

2.
```
s = 0;
for (i = 1; i <= n; i++)
  for (j = i; j <= 2*n; j++ )
    s = s+i+j;
```

3.
```
s = 0;
for (i = 1; i <= n; i++)
  for (j = 1; j <= c; j++ )
    s = s+i+j;
```

Lösung:

1. $t = \sum_{i=1}^{n} \sum_{j=1}^{2n} k = n \cdot 2n \cdot k = 2kn^2 = O(n^2)$

2.

$$
\begin{aligned}
t &= \sum_{i=1}^{n}\sum_{j=i}^{2n} k && (1)\\
&= \sum_{i=1}^{n}\left(\sum_{j=1}^{2n} k - \sum_{j=1}^{i-1} k\right) && (2)\\
&= \sum_{i=1}^{n}(2nk-(i-1)k) && (3)\\
&= \sum_{i=1}^{n}(2n+1-i)k && (4)\\
&= k\left(\sum_{i=1}^{n}(2n+1) - \sum_{i=1}^{n} i\right) && (5)\\
&= k\left(2n^2+n-\frac{n(n+1)}{2}\right) && (6)\\
&= k(\frac{3}{2}n^2+\frac{n}{2}) && (7)\\
&= \frac{k}{2}(3n^2+n) && (8)\\
&= O(n^2) && (9)
\end{aligned}
$$

3. $t = \sum_{i=1}^{n}\sum_{j=1}^{c} k = \sum_{i=1}^{n} ck = nck = O(n)$

Aufgabe 3: B-Bäume (Abgabe per abox) [10 Punkte]

Entwickeln Sie eine Java Applikation zur Verwaltung von B-Bäume beliebiger Ordnung! Die Applikation soll in form einer Klasse BTree zur Sortierung natürlicher Zahlen $\mathbf{N}^+$ realisiert werden, die folgende Methoden implementiert:

```
public class BTree
{
    // Konstruktor erzeugt einen B-Baum n-ter Ordnung (mit n>0)
    public BTree(int n);

    // Sortiert eine Zahl z in den Baum ein, sofern diese noch nicht im Baum
    // enthalten ist.
    public void insert(int z);

    // Prueft, ob eine Zahl z im Baum enthalten ist.
    public boolean contains(int z);

    // Gibt alle im Baum enthaltenen Zahlen in sortierter Folge als String
    // aus. Einzelne Zahlen werden durch Leerzeichen getrennt.
    public String toString();

    // Berechnet die Höhe des Baumes.
    public int height();
}
```

Schreiben Sie eine statische main-Methode, die eine rudimentäre Kommandosprache zum Zugriff auf die Zahlen im Baum ermöglicht. In dieser Sprache gibt es die folgenden Befehle (s. auch Übungsblatt 7):

`create zahl`	Initialisiert einen B-Baum entsprechender Ordnung.
`add zahl`	Einfügen eines neuen Eintrages, Duplikate sind nicht erlaubt. Keine Ausgabe.
`search zahl`	Ausgabe “true”, falls Zahl im Baum sonst “false”
`height`	Ausgabe der Höhe des Baumes
`print`	aufsteigend sortierte Ausgabe aller Einträge.

Das Programm beginnt per default mit einem Baum der Ordnung zwei. Wird der Befehl create ausgeführt wird der vorherige Baum verworfen.

Anmerkung: Das Programm muß als Kommandozeilenparameter den Namen einer Datei erwarten, die eine Folge von Befehlen der Kommandosprache enthält. Ein Beispiel für eine solche Kommandodatei finden Sie im Web[1].

Hier ein Beispiel für einen Kommandozeilenaufruf und der Ergebnisausgabe des von Ihnen zu entwickelnden Programmes:

```
kommandos.txt:
create 2
add 1
add 2
add 3
print
search 1
height
search 4
```

```
 > java BTree kommandos.txt
1 2 3
true
2
false
```

[1] http://www.informatik.uni-mannheim.de/informatik/pi4/stud/veranstaltungen/ws199900/pi1/ueb/blatt11/kommandos.txt