Informatik I WS 06/07
Prof. Dr. C. Damm
Dipl.-Inform. Marc Njoku

Übungsblatt 9

Ausgegeben am:	20.12.2006
Abgabe bis:	12.01.2007

Thema: Iteration, asymptotische Laufzeit + *O*-Notation, Mastertheorem, Java

Die Theorieaufgaben auf diesem Blatt sind bis zum Freitag, 12. Januar um 14.00 Uhr in die Info-I-Briefkästen im Erdgeschoss der NAM einzuwerfen. Die Kästen sind nach Übungsgruppen geordnet. Achten Sie bitte darauf dass Sie Ihren Zettel in den richtigen Kasten werfen, und dass **Name und Gruppe** auf **jedem Blatt** stehen! Falls Ihre Lösung mehrere Blätter umfasst, heften Sie diese bitte zusammen. Die Praxis-Aufgaben bearbeiten Sie bitte am Rechner und führen Sie Ihrem Tutor am Rechner vor. Eine Bearbeitung in Zweier-Teams innerhalb Ihrer Übungsgruppe ist möglich.

Aufgabe 1 (Theorie: 15 Punkte):

_O_-Notation

1. Geben Sie eine Java-Funktion an, die zwei Vektoren gleicher Länge als Parameter nimmt und den Summenvektor zurückgibt. Beispiele für Vektoren sind (1 2 3 4), (-1 0 1) oder (-10 - 23). Überlegen Sie sich eine geeignete Datenstruktur für Vektoren.
2. Leiten Sie die Funktionsklasse Ihrer Funktion in *O*-Notation her.

Aufgabe 2 (Theorie: 10 Punkte):

***O* -Notation**

Beweisen Sie: für $f(n) \in O(s(n))$ und $g(n) \in O(r(n))$ gilt: $f(n) + g(n) \in O(s(n) + r(n))$

Aufgabe 3 (Theorie: 12 Punkte):

_O_-Notation

Betrachten sie folgendes Kostenmaß, das für eine einfache Anweisung 1 Schritt berechnet, für eine Sequenz von Programmen die Summe ihrer Schritte, für eine Alternative das Maximum der möglichen Schritte und für eine Schleife die Anzahl der Schleifendurchläufe multipliziert mit der Anzahl der Schritte im Schleifen-Inneren:

T(Prog) = ?	
ZuweisungProg = Zuweisung	1
SequenzProg = $Prog_1$;$Prog_2$;$Prog_3$; . . . $Prog_n$;	$T(Prog_1) + . . . + T(Prog_n)$
Auswahl einer AlternativeProg = if(Bedingung) $Prog_1$; else $Prog_2$	$max(T(Prog_1) , T(Prog_2))$
SchleifeProg = for(i=0;i<k;i++) $Prog_1$;	$k * T(Prog_1)$

Außerdem betrachten sie diese Java-Methode:

```
long f(int n) {
  long result;
  if(n < 0) result = -1;
  else {
    result = 1;
    for(int i=0; i<n; i++) {
      int j = i+1;
      result = result * j;
    }
  }
  return result;
}
```

1. geben sie eine Laufzeitfunktion T(n) für obige Java-Methode an, basierend auf obigem Kostenmaß*.
2. Geben Sie die asymptotische Laufzeit in *O*-Notation (mit Herleitung) an.
3. Gehen Sie nun von einem verändertes Kostenmaß aus, in dem nun auch Additionen mit 1 und Multiplikationen mit 4 bewertet werden. Wie ändert sich die Laufzeitfunktion T(n), und wie verändert sich die asymptotische Laufzeit?

*: Sie können hierbei vernachlässigen, dass in jedem Schleifendurchlauf implizit der Wert der Laufvariablen i durch eine Zuweisung/Addition erhöht wird.

Aufgabe 4 (Theorie: 13 Punkte):

Divide and Conquer: Ermittlung des Maximums

Berechnen Sie mit Hilfe des Mastertheorems die asymptotische Laufzeit des gegebenen Programms.

```
public class Max {
        public static int compute(int[] array) {
                return computeRec(array,0,array.length-1);
        }
        public static int computeRec(int[] array, int from, int to){
                if(from==to) return array[from];
                else {
                        int mid = (from+to) / 2;
                        int ret1 = computeRec(array, from, mid);
                        int ret2 = computeRec(array, mid+1, to);
                        if (ret1<ret2) return ret2;
                        else return ret1;
                }
        }
}
```

Aufgabe 5 (Theorie: 20 Punkte):

Rekurrenzrelation - Mastertheorem

Gegeben sei die Rekurrenzrelation eines Algorithmus als

$T(n) = 4\ T(n/2) + n^2$, $T(1) = 1$.

1. Geben Sie die asymptotische Laufzeit des Algorithmus an.
2. Veranschaulichen Sie sich das Ergebnis durch Betrachtung der Höhe des Aufrufbaumes sowie des Aufwandes auf jeder einzelnen Ebene. Betrachten Sie dabei die inneren Ebenen des Baumes (Conquer-Schritte) sowie die Ebene der Basisfälle.
3. Geben Sie eine geschlossene Form fuer T(n) = f(n) an (Hinweis: Vergleichen Sie die Werte von T(n) mit den Werten der in (1.) erhaltenen Funktion).

Aufgabe 6 (Praktisch: 20 Punkte):

Email-Recycling in Java

Da die Anzahl der verschickten Emails weltweit in den letzten Jahren rasant angestiegen ist, wird es langsam Zeit für Strategien um mit dem anfallenden Datenmüll fertig zu werden. Insbesondere werden Verfahren benötigt, um die anfallenden Zeichenmengen sauber zu trennen und so einer Wertstoff erhaltenden Wiederaufbereitung zuführen zu können.
Helfen sie mit den Planeten zu retten und stellen Sie eine Klasse "Email" bereit die

1. die Eigenschaften "sender" (String), "recipients" (beliebig viele, jeweils ein String), "subject" (String) und "content" (String) hat (4 P.)
2. einen Konstruktor hat der diese Eigenschaften beim Aufruf setzt (3 P.)
3. eine Methode "print()", der die Email mit allen Eigenschaften auf den Bildschirm ausgibt (3 P.)
4. eine Methode "sort()", die den Text der Email (content) zeichenweise alphanumerisch mittels Bubble-Sort sortiert (und somit nach Buchstaben getrennt zur Wertstoff erhaltenden Wiederaufbereitung bereit stellt). (12 P.)
5. Eine Testmethode, die eine Email anlegt, ausgibt, den Text sortiert, und wieder ausgibt. (3 P.)

Bearbeitungshinweis: Sie können einen String in ein Array von char-Werten umwandeln mit der Methode

```
char[] cArr = content.toCharArray();.
```

Zurück wandeln können sie mit

```
String newContent = new String(cArr);
```

Aufgabe 7 (Praktisch: 10 Punkte):

Iteratives Sortierverfahren in Java

Implementieren Sie die iterative Variante des *SelectionSort*-Algoirthmus.