Hochschule Braunschweig/Wolfenbüttel
Institut für Angewandte Informatik
Prof. Dr. F. Seutter

WS 2013/14

Laboraufgaben Algorithmen und Datenstrukturen

Labortermine und späteste Abgabetermine:

Aufgabe	Labor	Abgabe
1	01.10.13 15.10.13	15.10.13
2	29.10.13 12.11.13	12.11.13
3	26.11.13 10.12.13	10.12.13

Allgemeines:

- Sie arbeiten in Zweier-Gruppen.
- Jede Laborabgabe muss spätestens an dem angegebenen Termin erfolgen. Bei jeder Aufgabe checken Sie Ihre Lösung im SVN ein und stellen den Laborbetreuern die Lösung der Aufgabe vor, welche beide Mitglieder der Gruppe erläutern können.
- Pfad für das SVN: `code.ostfalia.de/svn/i-audws2013/`
- Sie melden sich für die Gruppeneinteilung unter folgender URL bis zum 10.10.2013 elektronisch an: `https://10.136.1.215/`
- Für jede Aufgabe erstellen Sie ein neues Projekt in Eclipse. Die Projektnamen lauten entsprechend `AlgoAufgabe1`, `AlgoAufgabe2` und `AlgoAufgabe3`.
- Die zu implementierenden Interfaces und die öffentlichen JUnit-Tests können unter

  ```
  L:\fbi\gruendel\Algorithmik\java\Aufgabe1
  L:\fbi\gruendel\Algorithmik\java\Aufgabe2
  L:\fbi\gruendel\Algorithmik\java\Aufgabe3
  ```

  heruntergeladen werden. Die nicht-öffentlichen JUnit-Tests laufen auf unserem Sever und überprüfen Ihre ins SVN hochgeladenen Lösungen.
- Alle JUnit-Tests (öffentlich und nicht-öffentlich) müssen mit Ihren Implementierungen ohne Fehler laufen. Ein Teil der Aufgabenstellungen ist dann damit gelöst. Die restlichen Teile der Aufgabenstellungen sind mit der Präsentation Ihrer Implementierungen und der jeweils zugehörigen Dokumentation zu lösen.
- Ein Beispielprogramm für das Ausgeben und das Einlesen von jeweils einer Zahl in/aus einer Textdatei kann unter

  ```
  L:\fbi\gruendel\Algorithmik\java\
  ```

  heruntergeladen werden.

- Testdateien für die Aufgaben 1 und 3 können unter

  `L:\fbi\gruendel\Algorithmik\Materialien\Aufgabe`X`\`

  heruntergeladen werden. $X \in \{1, 3\}$ entspricht der Aufgabennummer.

- Verwenden Sie für Ihre Implementierung die folgenden Namen und halten Sie auch die vorgegebene Verzeichnisstruktur ein, $X \in \{1, 2, 3\}$.

Projektname:	`AlgoAufgabe`X
Package :	`de\ostfalia\algo\aufgabe`X
Klassenname:	`Aufgabe`X
Interface :	`A`X`Interface`

  Verzeichnisstruktur für das jeweilige Eclipse-Projekt:

```
+---AlgoAufgabeX
    \---src
    |   \---de
    |       \---ostfalia
    |           \---algo
    |               \---aufgabeX
    |                   |   AXInterface.java
    |                   |   AufgabeX.java
    |                   |
    |                   \---test
    |                           AufgabeXTestPublic.java
    |
    +---Materialien
        |   dateiname1.txt
        |   dateiname2.txt
        |   ...
        |
        \---Ausgabe
                testdaten.txt
```

1. Analysieren Sie den Algorithmus MAXIMUM auf seine Komplexitäten (BC, WC, AC) bzgl. der Anzahl der auszuführenden Vergleiche und Zuweisungen.

   Schreiben Sie dazu ein Programm (Konsolen-Applikation), das entsprechend der theoretischen Analyse in der Vorlesung geeignete Testdatensätze erzeugt. Verwenden Sie als Testdatensätze n paarweise verschiedene, gleichverteilt zufällige natürlichen Zahlen von 1 bis m (jeweils einschließlich). Die Anzahl n und die obere Grenze des Zahlenintervalls m sind variabel zu halten.

   Die Implementierung des Algorithmus MAXIMUM soll neben einem einmaligen Durchlauf auch in einer Schleife k-mal hintereinander ausgeführt werden können. Die Anzahl der Durchläufe k ist variabel zu halten. Je Durchlauf werden ein Testdatensatz generiert, das Maximum, die Anzahlen der Vergleiche und der Zuweisungen bestimmt und diese ausgegeben. Abschließend sind die Mittelwerte der vorgenommenen Vergleiche und Zuweisungen zu berechnen und auszugeben. Vergleichen und dokumentieren Sie Ihre Ergebnisse mit den Ergebnissen der theoretischen Analyse.

   Bei einem einmaligen Durchlauf ist zusätzlich die Option vorzusehen, den generierten Testdatensatz in eine Datei auszugeben und/oder einen Testdatensatz aus einer Datei einzulesen. Ein Testdatensatz ist als eine Textdatei mit einer Zahl je Zeile anzulegen.

   Die Funktionsweise des Programms ist zu demonstrieren und die Dokumentation der Ergebnisse ist abzugeben. Hierfür sind die Werte $n = 1\,000$, $m = 10\,000$ und $k = 1, 10, 100, 1\,000$ zu verwenden.

   (4 Punkte)

2. Die Berechnung von Binominalkoeffizienten $\binom{n}{k}$ kann auf verschiedene Weise erfolgen. Seien $n, k \in \mathbb{N}_0$ und $k \leq n$.

   (a) Berechnen Sie zunächst theoretisch das größte n, für das für alle $0 \leq k \leq n$ der Binominalkoeffizient noch als Datentyp `int` darstellbar ist.

   (b) Berechnung über Fakultäten:

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

   (c) Iterative Berechnung:

$$\binom{n}{k} = \begin{cases} \frac{n(n-1)(n-2)\ldots(n-k+1)}{1 \cdot 2 \cdot 3 \ldots \cdot k}, & \text{falls } 1 \leq k \leq n \\ 1, & \text{falls } k = 0 \end{cases}$$

   (d) Rekursive Berechnung:

$$\binom{n}{k} = \begin{cases} \binom{n-1}{k-1} + \binom{n-1}{k}, & \text{falls } n \geq 1 \text{ und } 1 \leq k < n \\ 1, & \text{falls } n = 0 \text{ oder } k = 0 \text{ oder } k = n \end{cases}$$

   - Implementieren Sie die Varianten b), c) und d) mit dem Datentyp `int` für die Argumente und dem Ausgabewert und berechnen Sie einige Beispielwerte. Geben Sie die Wertebereiche von n an, so dass Ihre Implementierungen für alle k funktionieren?
   - In Variante b) kann der Wertebereich erheblich vergrößert werden, indem Zähler und Nenner vor ihrer Division nicht erst vollständig berechnet werden. Finden Sie dafür eine Lösung und begründen Sie diese.
   - Bestimmen Sie die Komplexitäten Ihrer Prozeduren (Varianten b), c), d)) durch Zählen der Schleifendurchläufe bzw. Rekursionsaufrufe. In welcher Beziehung stehen diese Anzahlen zu den Werten von n und/oder k?

   Die Funktionsweise Ihrer Programme ist zu demonstrieren und die Dokumentation der Ergebnisse ist abzugeben.

   (1+2+3+2 Punkte)

3. Analysieren Sie die Zeitkomplexitäten (BC, WC, AC) der Sortieralgorithmen Bublle-Sort II, Merge-Sort und Quick-Sort.

   Schreiben Sie dazu ein Programm (Konsolen-Applikation), das entsprechend der theoretischen Analyse in der Vorlesung geeignete Testdatensätze erzeugt. Verwenden Sie als Testdatensätze n paarweise verschiedene, gleichverteilt zufällige natürlichen Zahlen von 1 bis m (jeweils einschließlich). Die Anzahl n und die obere Grenze des Zahlenintervalls m sind variabel zu halten.

   Die Implementierungen der Algorithmen sollen neben einem einmaligen Durchlauf auch in einer Schleife k-mal hintereinander ausgeführt werden können. Die Anzahl der Durchläufe k ist variabel zu halten. Je Durchlauf werden ein Testdatensatz generiert, die Zahlen aufsteigend sortiert, die Anzahlen der Vergleiche und der Vertauschungen bestimmt und diese ausgegeben. Abschließend sind die Mittelwerte der vorgenommenen Vergleiche und Vertauschungen zu berechnen und auszugeben. Vergleichen und dokumentieren Sie Ihre Ergebnisse mit den Ergebnissen der theoretischen Analyse.

   Bei einem einmaligen Durchlauf ist zusätzlich die Option vorzusehen, den generierten Testdatensatz in eine Datei auszugeben und/oder einen Testdatensatz aus einer Datei einzulesen. Ein Testdatensatz ist als eine Textdatei mit einer Zahl je Zeile anzulegen.

   Die Funktionsweise der Programme ist zu demonstrieren und die Dokumentation der Ergebnisse ist abzugeben. Hierfür sind die Werte $n = 1\,000$, $m = 10\,000$ und $k = 1, 10, 100, 1\,000$ zu verwenden.

   (8 Punkte)

Algorithmen und Datenstrukturen
Punkteverteilung der Laboraufgaben WS 13/14

Aufgaben

			JUnit-Test
1)	Generierung Testdaten funktioniert	1 Punkt	x
	Implementierung Maximum-Suche richtig	1 Punkt	x
	Anzahlen Vergleiche, Zuweisungen, Mittelwerte richtig	0,5 Punkte	x
	Parametrierbarkeit gegeben	0,5 Punkte	-
	Ergebnisdokumentation liegt vor	1 Punkt	-
2a-d)	Ergebnisdokumentation liegt vor,		
	Wertebereiche und Komplexitäten richtig dokumentiert	1,5 Punkte	-
2b)	Implementierung richtig	1 Punkt	-
	Ergebnis und Zähler richtig	1 Punkt	x
2c)	Implementierung richtig	1 Punkt	-
	Ergebnis und Zähler richtig	1 Punkt	x
	Vergrößerung des Wertebereiches gelungen	0,5 Punkte	-
2d)	Implementierung richtig	1 Punkt	-
	Ergebnis und Zähler richtig	1 Punkt	x
3)	Implementierung Bubble-Sort-II richtig	1 Punkt	-
	Sortierung, Anzahlen Vergl., Vertausch., Mittelwerte richtig	1 Punkt	x
	Implementierung Merge-Sort richtig	1 Punkt	-
	Sortierung, Anzahlen Vergl., Vertausch., Mittelwerte richtig	1 Punkt	x
	Implementierung Quick-Sort richtig	1 Punkt	-
	Sortierung, Anzahlen Vergl., Vertausch., Mittelwerte richtig	1 Punkt	x
	Ergebnisdokumentation liegt vor und ist richtig	2 Punkte	-
	Summe	20 Punkte	