Prof. Dr. Gerd Stumme, Folke Eisterlehner, Dominik Benz
Fachgebiet Wissensverarbeitung

8.06.2009

9. Hausübung „Algorithmen und Datenstrukturen“

Sommersemester 2009

Abgabetermin: 3.07.2009, 23:59:00Uhr MESZ
Mit Hilfe dieses Zusatzaufgabenblattes können eine verpasste Abgabe von Aufgaben und fehlende Punkte ausgeglichen werden. Korrekte Abgaben zur letzten Teilaufgabe nehmen automatisch an einem Programmierwettbewerb teil.

Comparator: Schreiben Sie einen Comparator **`IntegerComp`** für die Klasse **`Integer`**, so dass für alle Integer-Zahlen i, j gilt:

$$\texttt{compare}(i,j) = \begin{cases} 1, & \text{falls } i > j \\ 0, & \text{falls } i = j \\ -1, & \text{falls } i < j \end{cases}$$

Ferner soll die Klasse **`IntegerComp`** ein Attribut **`counter`** enthalten, welches die Anzahl von Vergleichsoperationen zählt, die mithilfe einer Instanz dieser Klasse durchgeführt wurden. Dieses Attribut soll mit der Methode **`getCounter`** ausgelesen werden können.

(4 Punkte)

Pivotisierung: Schreiben Sie zu jedem der folgenden Verfahren zur Auswahl eines Pivotelements **`p`** aus einem Array **`a`** jeweils eine Java-Klasse, welche das auf der Webseite der Vorlesung zur Verfügung gestellte Interface **`DataPivot`** implementiert (dabei bezeichnet n die Länge von **`a`**):

(1) Setze **`p`**:= max(**`a[1]`**, ..., **`a[`**n**`]`**)
(2) Wähle $i \in \{1, \ldots, n\}$ *zufällig* und setze **`p`**:= **`a[`**i**`]`**.
(3) Wähle $i_1, i_2, i_3 \in \{1, \ldots, n\}$ *zufällig* und setze **`p`**:= min(**`a[`**i_1**`]`**,**`a[`**i_2**`]`**,**`a[`**i_3**`]`**).

Alle Vergleichsoperationen auf Elementen von **`a`** sollen unter Verwendung des als Parameter übergebenen Comparators durchgeführt werden. Benennen Sie die Klassen so, dass die i-te Pivotisierungsmethode in der Klasse „**`Pivot`**i“ implementiert wird. (12 Punkte)

QuickSort: Schreiben Sie eine Java-Klasse **`QuickSort`**, welche den QuickSort-Algorithmus umsetzt. Die Klasse soll das Interface **`DataSort`** von der Webseite zur Vorlesung implementieren und zur Auswahl des Pivot-Elements eine Pivotisierungsstrategie vom Typ **`DataPivot`** verwenden. Die Pivotisierungsstrategie soll in einem Attribut „**`pivotSelector`**“ gespeichert sein und mittels der Methoden **`getPivotSelector`** und **`setPivotSelector`** gelesen bzw. gesetzt werden können. (10 Punkte)

Laufzeitmessung: Schreiben Sie eine Klasse **ComparisonCounter**, welche die Anzahl der Schlüsselvergleiche obiger Klasse **QuickSort** unter Verwendung der verschiedenen Pivotisierungsstrategien **DataPivot1**,. . ., **DataPivot3** und des Comparators **IntegerComp** vergleicht.

Erzeugen Sie hierzu zehn verschiedene Arrays mit 10.000 Zufallszahlen mittels der Methode **getRandomData** der Klasse **DataSource2** von der Webseite zur Lehrveranstaltung. Führen Sie dann jeweils die Methode **doSort** der Klasse **QuickSort** unter Verwendung der verschiedenen Pivotisierungsstrategien aus und speichern Sie die Anzahl der jeweils benötigten Schlüsselvergleiche ab.

Geben Sie zum Schluss für jede QuickSort-Variante den jeweiligen Mittelwert aus. Ihre Ausgabe soll wie folgt aussehen:

```
Anzahl von Schluesselvergleichen fuer:
   DataPivot1: <X1>
   DataPivot2: <X2>
   DataPivot3: <X3>
```

wobei <X*i*> für den vorher bestimmten Mittelwert der Schlüsselvergleiche von QuickSort mit Pivotisierungsstrategie **DataPivot***i* steht. Vergleichen Sie Ihre Ergebnisse anschließend mit der Average-Case- und Worst-Case-Komplexität von QuickSort. (8 Punkte)
Hinweis: Starten Sie Ihr Programm mittels „**java -Xss1024k**“

Wettbewerb: Überlegen Sie sich eine eigene möglichst effiziente Pivotisierungsstrategie und implementieren Sie diese in einer Klasse **PivotOpt**, welche das Interface **DataPivot** implementiert. Listen Sie in den Kommentaren Ihres Quelltextes die Überlegungen auf, die zu der Konstruktion Ihres Verfahrens geführt haben.

Hinweis: Achten bei der Implementierung Ihrer Pivotisierungsstrategie darauf, dass jeder Schlüsselvergleich in **PivotOpt** mit Hilfe des übergebenen Comparators durchgeführt wird. Anderenfalls droht der Ausschluss vom Programmierwettbewerb. (6 Punkte)

Allgemeine Hinweise zum Programmierwettbewerb:

- Grundlage des Wettbewerbs ist die Anzahl von Schlüsselvergleichen, die unsere **QuickSort**-Implementierung unter Verwendung Ihrer Pivotisierungsstrategie mit unserer Implementierung der **IntegerComp**-Klasse zur Sortierung eines Feldes benötigt.
- Getestet wird auf 10 geheimen Datensätzen mit jeweils 10.000 Elementen.
- Es zählt die durchschnittliche Anzahl von Schlüsselvergleichen bei Sortierung unserer Datensätze (wir berechnen das arithmetisches Mittel aus den zehn Sortierdurchläufen).
- Gewinner des Wettbewerbs ist die Pivotisierungsstrategie mit minimaler durchschnittlichen Anzahl von Schlüsselvergleichen.

- Die Abgabe einer Lösung erfolgt wie üblich über das Abgabesystem zu diesem Aufgabenblatt. Zwischenlösungen können jederzeit hochgeladen und aktualisiert werden.
- Wir werden täglich aktuelle Zwischenergebnisse auf unsere Webseite bekanntgeben.
- Dafür ist es wichtig, dass die von uns vorgegebenen Interfaces strikt umgesetzt und nicht angepasst werden. Ferner müssen alle Klassen im Standard-Paket liegen (d.h. ohne Package-Angabe).
- Zur Bestimmung des Pivotelements darf nur auf das übergebene Array, Attribute ihrer Klasse und auf Zufallszahlen der Klasse `DataSource3` von der Vorlesungsseite zurückgegriffen werden.

Viel Spaß und viel Erfolg!