technische universität
dortmund

Fakultät für Informatik
Lehrstuhl 1
Dr. Lars Hildebrand
Frank Thorsten Breuer, Karsten Klein,
Marcel Martin, Sascha Plazar

Praktikum zu

Einführung in die Informatik für LogWings und WiMas

Wintersemester 2011/12

Übungsblatt 9

Bearbeitungszeit:
09.–13.01.2012

Aufgabe 9.1 – Quicksort

Der Algorithmus Quicksort ist ein Sortieralgorithmus. Ein Beispiel soll erklären, wie er funktioniert: Gegeben ist ein Array der Länge 7, das die Zahlen {25, 5, 8, 9, 1, 2, 7} enthält.

Zunächst sucht man sich irgendein Element des zu sortierenden Arrays aus, dieses wird auch *Pivotelement* genannt. Als Pivotelement wird ab jetzt immer das am weitesten rechts stehende Element genommen, dies ist hier die 7. Dann ordnet man die Zahlen so um, dass im Array zuerst alle Zahlen stehen, die kleiner als oder gleich dem Pivotelement sind, danach steht das Pivotelement selbst und schließlich alle Zahlen, die größer als das Pivotelement sind: {5, 1, 2, **7**, 25, 8, 9}

Das Pivotelement (unterstrichen) steht jetzt schon an seinem richtigen Platz innerhalb des Arrays und wird nicht mehr angefasst. Die Teile des Arrays, die links und rechts vom Pivotelement stehen, sind noch unsortiert, daher wird der Algorithmus rekursiv auf diese beiden Teilarrays (zuerst links, dann rechts) angewandt, um sie auch zu sortieren. Zuerst wird {5, 1, 2} sortiert, indem wieder das Pivotelement gewählt wird (die 2) und das Array wie beschrieben umsortiert wird: {1, **2**, 5}

Jetzt wird wieder rekursiv das linke Teilarray sortiert. Dies besteht nur aus dem Element 1, daher wird dieses als Pivotelement gewählt und es passiert nichts weiter. Das gleiche passiert jetzt mit dem rechten Teilarray, das nur aus der 5 besteht.

Die Elemente {1, 2, 5, 7} liegen jetzt schon sortiert vor. Als nächstes wird das Teilarray {25, 8, 9} sortiert und der Algorithmus ist fertig.

Aufgabe Gegeben ist folgendes Array mit ganzen Zahlen: {39, 47, 41, 42, 89, 92, 48, 5, 90, 33, 45}

Geben Sie an, in welcher Reihenfolge Quicksort hier die Pivotelemente wählt. Berücksichtigen Sie: Es wird immer das letzte Element eines Teilarrays als Pivotelement gewählt. Es wird zuerst der Teil links vom Pivotelement, dann der Teil rechts vom Pivotelement rekursiv sortiert.

Aufgabe 9.2 – Heaps

Bearbeiten Sie diese Aufgabe auf dem Papier. Beachten Sie die Definition eines Heaps auf der Folie 10 in Kapitel 5, Teil 2, und die Anmerkungen zum Einfügen von Werten in einen Heap ab Folie 19.

a) Fügen Sie nacheinander (mit Hilfe der Einfüge-Operation, die Sie aus der Vorlesung kennen) in einen zu Beginn leeren Heap die Werte 17, 47, 32, 12, 34, 4, 14, 50, 1 und 80 ein.

b) Eine aufsteigend sortierte Folge der Zahlen kann erzeugt werden, indem in jedem Schritt das minimale Element aus dem Heap entfernt wird und dann die Heapeigenschaft wieder hergestellt wird. Führen Sie dies für den gefüllten Heap durch und notieren Sie jedes Vertauschen und jedes Entfernen von Knoten. Sie können aufhören, wenn die 17 entfernt wurde.

Aufgabe 9.3 – Wiederholung: Arrays

Programmieren Sie eine Methode `fibonacciArray`, die eine ganze Zahl n als Parameter bekommt und ein Array von ganzen Zahlen zurückliefert. Das zurückgelieferte Array enthält die ersten n Fibonacci-Zahlen. Fibonacci-Zahlen wurden auf den Übungsblättern 4 und 5 besprochen. Die Methode muss auch für $n = 0$ und für $n = 1$ korrekt funktionieren. Falls $n < 0$, soll `null` zurückgeliefert werden.

Aufgabe 9.4 – Noch mal Arrays: Sieb des Eratosthenes

Um alle Primzahlen bis zu einer bestimmten Größe zu finden, gibt es ein Verfahren namens *Sieb des Eratosthenes*, das Sie eventuell schon aus der Schule kennen. Angenommen, alle Primzahlen bis zur Größe n sollen bestimmt werden, dann werden zunächst alle Zahlen von eins bis n aufgeschrieben. Danach werden alle Zahlen weggestrichen, die durch zwei teilbar sind, außer der zwei selbst, also $4, 6, 8$ usw. Nun wird mit der nächsten Zahl fortgefahren, die noch nicht gestrichen ist, in diesem Fall der drei. Es werden also alle Vielfachen dieser Zahl weggestrichen, also $6, 9, 12, \ldots$. Dieses Vorgehen wird solange wiederholt, bis keine weiteren Zahlen mehr gestrichen werden können.

Implementieren Sie eine Methode `siebDesEratosthenes`, die als Parameter eine ganze Zahl n erhält und ein Array von Booleans der Länge $n + 1$ zurückliefert, das für jede Zahl Auskunft gibt, ob es sich um eine Primzahl handelt.

Tipp: Legen Sie in Ihrer Methode als erstes ein Array von Booleans der Länge $n + 1$ an und setzen Sie alle Elemente des Arrays auf `true`.

Ergänzende Aufgaben

Aufgabe 9.5 – Pascalsches Dreieck

Beim Ausmultiplizieren von Binomen (Terme der Form $(x + y)^n$) treten Koeffizienten auf, die sich nach einem einfachen Schema berechnen lassen. Zum Beispiel erhält man für $n = 3$ folgendes: $(x + y)^3 = x^3 + 3x^2y^1 + 3xy^2 + y^3$. Die Koeffizienten lauten in diesem Fall $1, 3, 3, 1$; sie heißen *Binomialkoeffizienten*. Schreibt man die Koeffizienten, die sich für ein bestimmtes n ergeben, in eine Zeile und notiert alle Zeilen bis zu einem bestimmten n, so erhält man ein Pascalsches Dreieck. Für $n = 6$ sieht es so aus:

```
n
0                           1
1                       1       1
2                   1       2       1
3               1       3       3       1
4           1       4       6       4       1
5       1       5       10      10      5       1
6   1       6       15      20      15      6       1
```

Schreiben Sie nun eine Methode `pascalschesDreieck`, die einen ganzzahligen Parameter n hat und ein Pascalsches Dreieck bis zur Zeile n ausgibt. Auf eine schöne grafische Darstellung kommt es hier nicht an, daher reicht es, wenn die Ausgabe für $n = 5$ ungefähr so aussieht:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
```

Es gibt mehrere Möglichkeiten, diese Aufgabe zu lösen. Sie können zum Beispiel für jede Zeile des Dreiecks ein neues Array anlegen und die jeweils vorherige Zeile benutzen, um die aktuelle Zeile zu füllen. Sie können auch von Anfang an nur ein Array anlegen, das bereits groß genug ist, um alle Koeffizienten der letzten Zeile aufzunehmen, und dieses Array immer wieder überschreiben.

Aufgabe 9.6 – Sortieren: Bubblesort

Informieren Sie sich (zum Beispiel unter `http://de.wikipedia.org/wiki/Bubblesort`) über das Sortierverfahren „Bubblesort“. Programmieren Sie es dann in Java; benutzen Sie dabei eine `do-while`- und eine `for`-Schleife.