Übungsblatt 4

Abgabe: 24.11.2008

Aufgabe 1 Sortieren ohne Bläschen und nicht quick

Neben dem in der Vorlesung vorgestellten *Bubblesort*-Verfahren und dem *Quicksort*-Algorithmus gibt es noch weitere Sortierverfahren – Ihr dürft ein weiteres implementieren.

Aufgabe 1.1 Sortiert?

Auch um während des Programmierens den Überblick zu behalten, ist es hilfreich eine Möglichkeit zu haben um zu entscheiden, ob ein Array sortiert ist. Implementiert eine Methode, die für einen als Parameter übergebenen Array entscheidet, ob dieser sortiert ist und `true` oder `false` zurückgibt.

Aufgabe 1.2 Diagnose

Sollte Eure Umsetzung des Algorithmus nicht ganz das gewünschte Ergebnis liefern, hilft es bei der Fehlersuche, sich das jeweils betrachtete Array ausgeben zu lassen, auch dies sollte in einer eigenen Methode passieren.

Aufgabe 1.3 Aus 2 mach 1

Als Eingabe werden zwei *int*-Arrays a und b, die *sortierte* Werte enthalten, übergeben. Aus diesen lässt sich ein sortiertes Array c bilden (dessen Länge dann natürlich der Summe der Längen von a und b entspricht), indem jeweils a und b aufsteigend durchlaufen werden und dabei jeweils die aktuellen Werte verglichen werden. Der kleinere Wert wird in das neue Array c eingetragen, und nur in dem Ursprungsarray, in dem dieser Wert steht, darf nun ein Feld weiter gegangen werden. Wenn eins der Ursprungsarrays abgearbeitet ist, wird c mit den restlichen Werten des anderen gefüllt. Implementiert diese Methode, die das Hilfsarray c zurückgibt.

Aufgabe 1.4 Sortieren in Abschnitten

Der folgende Algorithmus soll ein Array von *int*-Werten sortieren. Bei Start des Verfahrens ist ein Array a (mit Länge $n > 0$) mit *int*-Werten belegt.

In einem ersten Durchlauf werden jeweils Eintrag a_{2i} und a_{2i+1} für $i = 0, ..., \frac{n}{2} - 1$ miteinander verglichen und gegebenfalls vertauscht, so das $a_{2i} \leq a_{2i+1}$ gilt.

Im nächsten Durchlauf werden dann Abschnitte der Länge 2 wie folgt kombiniert: a_0, a_1 bilden den ersten Abschnitt, a_2, a_3 bilden den zweiten Abschnitt. Diese Abschnitte werden der in Aufgabenteil Aufgabe 1.3 geschriebenen Methode übergeben,

mit den Werten des zurückgegebenen Arrays werden dann $a_0 ... a_3$ überschrieben. Weiter geht es dann mit den Abschnitten a_4, a_5 und a_6, a_7 usw.

Im dritten Durchlauf werden dann analog Abschnitte der Länge 4 kombiniert, im vierten Durchlauf Abschnitte der Länge 8, usw.

Implementiert dieses Sortierverfahren und überlegt Euch geeignete Testfälle um zu prüfen, das dieses Verfahren seine Aufgabe erfolgreich erfüllt.

Terminierung Das Verfahren endet, wenn das Array kürzer ist, als die zu kombinierende Abschnittslänge. Unglücklicherweise gibt es häufig Arrays, deren Länge gerade nicht eine Zweierpotenz ist – in diesem Fall muss bei den Abschnittsgrenzen sehr sorgfältig programmiert werden.

Aufgabe 1.5 Kompliziert?

Dieser Algorithmus hört sich ausgesprochen kompliziert an, aber ist er tatsächlich aufwändiger als das Bubblesort-Verfahren aus der Vorlesung? Gebt eine Abschätzung für die Anzahl der Rechenschritte (Vergleiche bzw. Zuweisungen), die nötig sind, um ein Array der Länge n zu sortieren.