Einführung in das Programmieren (2)

7. Sortieren

7.1. Einführung

Beim **Sortieren** von Datenbeständen geht es stets darum, diese in eine gewisse Reihenfolge zu bringen. Bei Zahlen sortiert man i. d. R. nach der Größe, bei Texten nach dem Alphabet usw.

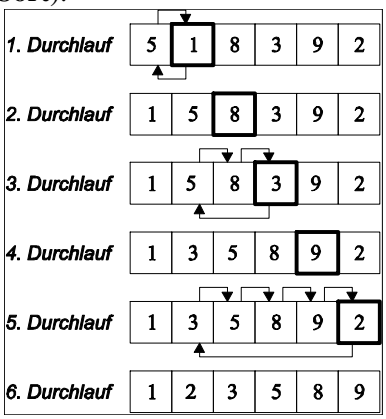
Bei den Sortierverfahren unterscheidet man
- Interne Verfahren: Sortierung von Datensätzen (z.B. Arrays, Listen), die im Hauptspeicher gehalten werden können und
- Externe Verfahren: Sortierung von Massendaten, die auf externen Speichermedien gehalten werden, wie z. B. Datenbestände in Datenbanken

Im Folgenden sollen die bekanntesten Sortierverfahren vorgestellt werden.

7.2. Sortieren durch Einfügen: InsertionSort

Beispiel: Sortieren eines Stapels von Spielkarten
Das typische Vorgehen ist wohl: man startet mit der ersten Karte des Stapels, nimmt jeweils die nächste Karte des Originalstapels und fügt diese an der richtigen Stelle in den neuen Stapel ein.

Dieses Verfahren heißt **Sortieren durch Einfügen** (= **InsertionSort**).
Beispiel: Sortieren einer Zahlenfolge



Kommentar:

Ein möglicher Java-Code für den Insertsort-Algorithmus ist:

```
public void insertionSort(int[] a)
    {
        int i;
        int j;
        int v;

        //Betrachte Element für Element und füge jedes an seinen richtigen
         //Platz zwischen die bereits betrachteten ein.

        for (i = 0; i < a.length; i++)
        {
            v = a[i];
            j = i;

            //1. breche while Schleife ab, wenn j-1 einen Wert kleiner Null hat.
            //2. breche while Schleife ab, wenn das Element bei j-1 größer als das
            //   Element bei i ist.

            while ((j > 0) && (a[j - 1] > v))
            {
                a[j] = a[j - 1];
                j--;
            }
            a[j] = v;
        }
    }
```

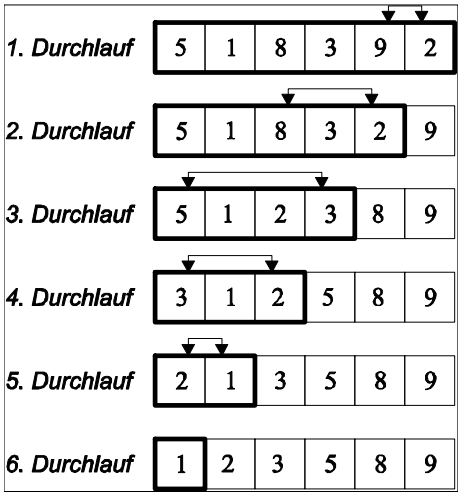
7.3. Sortieren durch Selektion: SelectionSort

Beim **Sortieren durch Selektion** (**= SelectionSort**) liegt folgendes Verfahren zu Grunde:

– Wähle das größte Element der Folge und bringe es ans Ende.
– Verfahre weiter so, wobei die Folge in jedem Schritt um eine Position verkürzt wird

Beispiel: Sortieren einer Zahlenfolge

Kommentar:



7.4. Sortieren durch Vertauschen: BubbleSort

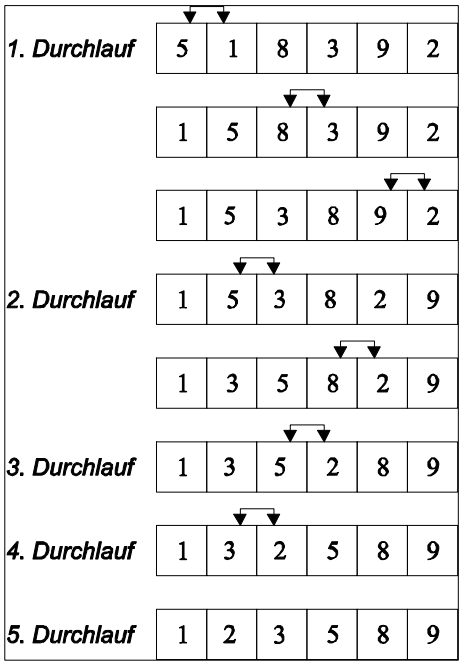
Vorgehen beim **Sortieren durch Vertauschen** (**= BubbleSort**):

– Folge immer wieder durchlaufen und dabei benachbarte Elemente in die richtige Reihenfolge bringen
– Größere Elemente überholen so die kleineren und drängen ans Ende der Folge

Zur Verdeutlichung: Bei einer vertikalen Anordnung der Elemente der Folge sortieren sich diese selbst wie aufsteigende Blasen in einer Flüssigkeit, da die großen die kleinen „überholen“

Beispiel: Sortieren einer Zahlenfolge

Kommentar:



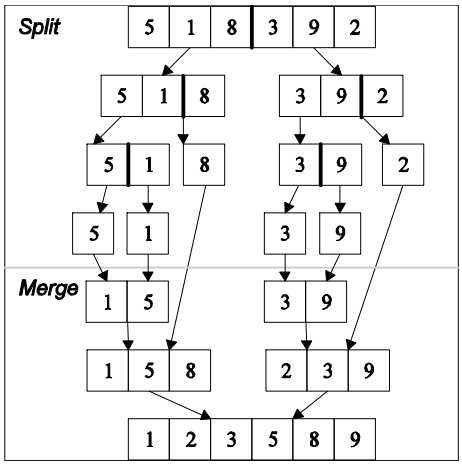
7.5. Sortieren durch Mischen: MergeSort

Die bisherige Verfahren benötigten direkten Zugriff auf die Daten und sind daher nur für internes Sortieren geeignet. Was ist zu machen, wenn die Daten nicht in den Hauptspeicher passen?

Dann kann auf das **Sortieren durch Mischen** (= **MergeSort**) zurück gegriffen werden.

Hierbei wird in zwei Phasen sortiert:
- Die Folge wird in Teile zerlegt, die jeweils in den Hauptspeicher passen und daher getrennt voneinander mit internen Verfahren sortiert werden können. Diese sortierten Teilfolgen werden wieder in Dateien ausgelagert.
- Anschließend werden die Teilfolgen parallel eingelesen und gemischt, indem jeweils das kleinste Element aller Teilfolgen gelesen und in die neue Folge (d.h. wieder in eine Datei) geschrieben wird.

Dieses Verfahren ist auch für internes Sortieren anwendbar.
Das Mischen der Teilfolgen erfolgt durch eine rekursiven Aufruf von MergeSort.
Beispiel: Sortieren einer Zahlenfolge



Kommentar:

7.6. Quicksort

Der **Quicksort** ist ein rekursiver Algorithmus, der von C. A. R. Hoare erfunden wurde (1962).
Er ist ein sehr häufig eingesetztes Sortierverfahren und arbeitet in den meisten Fällen sehr effizient.
Er besitzt ein ähnliches Prinzip wie der Mergesort: „Teile und herrsche“, stellt dem gegenüber jedoch eine Verbesserung dar, der Mischvorgang vermieden wird.