Sortieren

> Ein Sortierverfahren ist ein Algorithmus, der dazu dient, ein Tupel (i. A. ein Array) zu sortieren. Voraussetzung ist, dass auf der Menge der Elemente eine strenge schwache Ordnung definiert ist, z. B. die lexikographische Ordnung von Zeichenketten oder die numerische Ordnung von Zahlen.
>
> Es gibt verschiedene Sortierverfahren, die unterschiedlich effizient arbeiten bzgl. der Zeitkomplexität (Anzahl der nötigen Operationen) sowie der Platzkomplexität (zusätzlich zum Eingabe-Array benötigter weiterer Speicherplatz). Die Komplexität eines Algorithmus wird üblicherweise in der Landau-Notation dargestellt (s. u. Ausdrücke wie O(n*log(n)). Die Zeitkomplexität hängt bei einigen Sortierverfahren von der anfänglichen Anordnung der Werte im Array ab, man unterscheidet dann zwischen Best Case (bester Fall), Average Case (Durchschnittsverhalten) und Worst Case (schlechtester Fall ~ die Werte sind „maximal ungünstig vorgeordnet“). Häufig sind zusätzliche Faktoren zu beachten, die Einfluss auf Zeit- oder Platzkomplexität haben, zum Beispiel langsamer Zugriff auf extern liegende Daten, begrenzte Größe des Arbeitsspeichers oder ähnliches.
>
> *Quelle: Wikipedia,CC-by-sa-3.0*

Sortierverfahren werden mit zunehmender Zahl der zu sortierenden Elemente schnell sehr komplex. Aus diesem Grund eignen sich Algorithmen zum Sortieren sehr gut, um über die Effizienz eines Verfahrens nachzudenken - es gibt viel Optimierungspotential.

Sortieren

Der Mistkäfer Willi möchte Ordnung in seine Mistkugelsammlung bringen. Am liebsten hätte er sie schön der Grösse nach sortiert, die kleinste Kugel links, die grösste rechts. Leider hat Willi keine Ahnung, wie er das anstellen könnte. Er setzt sich auf eine der Mistkugeln und überlegt, aber es fällt ihm keine Lösung ein.

Da wählt er in seinem Frust zufällig zwei Mistkugeln aus, bei denen die Reihenfolge nicht stimmt, und vertauscht die beiden. Willi ist natürlich klar, dass er damit seine Kugelreihe noch lange nicht sortiert hat. In seiner wachsenden Verzweiflung wählt er ein weiteres Kugelpaar, das nicht richtig geordnet ist, und vertauscht auch dieses.

Diesen Vorgang wiederholt er nun laufend. Nach vielen solchen Vertauschungen stutzt Willi plötzlich: Er findet kein einziges "falsches" Paar mehr! Enttäuscht wendet er sich ab und versinkt wieder ins Grübeln. Nach einer Weile schaut er auf und betrachtet die Reihe seiner Mistkugeln…[1)]

Wozu sortieren wir?

In unserer welt sind allemöglichen Sachen sortiert: Rechnungen, Adressen, Kleider, Bankbelege, Personen und vieles mehr. Wozu eigentlich?

Logisch: Sortieren erleichtert das Wiederfinden. Ein Telefonbuch, in welchem die Einträge in zufälliger Reihenfolge abgedruckt sind, wäre nahezu nutzlos.

Aufgabe

Beschreibe den Unterschied zwischen einem Inhaltsverzeichnis und dem Index eines Buches bezüglich der Reihenfolge der Einträge.

Sortierkriterien

Das **Kriterium**, nach dem wir etwas sortieren ist von wesentlicher Bedeutung: Ein nach den Nachnamen sortiertes Telefonbuch ist von großem Nutzen, sortiert man die Einträge nach den Vornamen ist der Nutzen geringer - oder zumindest ein anderer, ebenso wie eine Sortierung nach den Telefonnummern ein vollkommen anderes Verzeichnis zur Folge hat.

Wie man am Beispiel des Telefonbuchs sieht, gibt es auch kombinierte Sortierkriterien: Die Einträge sind in erster Linie nach der Ortschaft sortiert, innerhalb einer Ortschaft nach Name, bei gleichen Namen nach Vorname.

Aufgabe

In welcher Reihenfolge stehen folgende Namen im Telefonbuch?

- Heinrich Vonberg
- Hubert Vogel
- Franziska Völker
- Marian Völkermann
- Uwe Vockler
- Hans-Herrmann Vollenbirker
- Mark von Salis
- Wilhelm Vogel

Vergleichbarkeit

Um eine Liste nach einem bestimmten Kriterium sortieren zu können, müssen je zwei Elemente gemäss diesem Kriterium mit einander **verglichen** werden können. Von zwei beliebigen Elementen muss entscheidbar sein, welches das grössere bzw. das kleinere ist. Natürlich gibt es auch den Fall der Gleichheit zweier Elemente.

Diese Vergleichbarkeit ist z.B. bei Zahlen selbstverständlich. Bei Buchstaben ebenfalls, da gilt die alphabetische Ordnung. Der Vergleich zweier Buchstabenfolgen (also Wörter) gemäss alphabetischer Ordnung ist hingegen schon etwas komplizierter.

Um dieses Problem etwas zu entschärfen, werden wir in unseren Überlegungen stets **Arrays mit ganzen Zahlen** sortieren, da hier die Vergleichbarkeit durch die Operatoren `<`,`>`,`=` eindeutig gegeben ist.

Aufgabe

Welche Probleme ergeben sich, wenn man eine Schulklasse nach

- Haarfarbe
- Geschlecht
- Hobbies

sortieren möchte?

Sortierrichtung

Man kann für jedes Kriterium, nach dem sortiert werden soll (und kann) die Richtung der Sortierung festlegen: *Aufsteigend* oder *Absteigend*. Bei Aufsteigender Sortierung befinden sich die nach der Vergleichsmethode kleineren Elemente am Beginn der Liste, bei absteigender Sortierung ist es umgekehrt.

Sortierte Arrays

Im folgenden ist ein unsortiertes Array zu sehen. Die Reihenfolge der Elemente ist durch den Index (in eckigen Klammern) festgelegt, der Wert der jeweiligen Array-Variablen durch die Zuweisung:

```
int[] zahlen = new int[5];
zahlen[0]=7
zahlen[1]=3
zahlen[2]=15
zahlen[3]=5
zahlen[4]=12
```

Nun die sortierte Variante:

```
zahlen[0]=3
zahlen[1]=5
zahlen[2]=7
zahlen[3]=12
zahlen[4]=15
```

Die Werte sind nun aufsteigend sortiert, die Reihenfolge noch immer durch den Index gegeben.

Wann ist ein Array sortiert?

Um einzusehen, welche Bedingungen ein Array erfüllen muss, damit es sortiert ist, hilft uns die Geschichte von Willi und seinen Mistkugeln weiter: Nachdem er viele Kugelpaare vertauscht hat, bei

denen die linke Kugel grösser war als die rechte, fand er irgendwann kein solches "falsches" Paar mehr. Natürlich waren zu diesem Zeitpunkt seine Kugeln vollständig sortiert.

Warum funktioniert das eigentlich? Jede Vertauschung bringt eine grössere Kugel ein Stück nach rechts und eine kleinere Kugel ein Stück nach links. Auf diese Weise trägt jede Vertauschung ein kleines Stück zur Sortierung bei. Jede Vertauschung macht also die Sortierung ein bisschen besser. Die Sortierung ist perfekt, wenn es keine falschen Paare mehr gibt.

Ein Array ist also sortiert, wenn es keine zwei Elemente mit falscher Reihenfolge gibt.

Es ist leicht einzusehen, dass auch die folgende Aussage richtig ist: **Ein Array ist sortiert, wenn es keine zwei benachbarten Elemente mit falscher Reihenfolge gibt.**

Klicken, um den Quellcode zu sehen

App.java

```
/**
 * Erzeugt eine Zufallsreihe und ermöglicht Abfragen darüber.
 *
 * @author Rainer Helfrich
 * @author Frank Schiebel
 * @version 1.0
 */
class Zufallsreihe
{
    private int[] daten;
    int anzahl;

    public Zufallsreihe(int anzahl)
    {
        this.anzahl = anzahl;
        daten = new int[anzahl];
        for (int i = 0; i < daten.length; i++)
        {
            // Für manche Aufgaben sollte man die 6 durch z.B. 1000
ersetzen
            daten[i] = getZufallszahl(6);
        }
    }

    public int aufgabe01Summe()
    {
        return 0;
    }

    public int aufgabe02ZaehleNullen()
    {
        return 0;
```

```
    }

    public int aufgabe03FindeLetzteNull()
    {
        return 0;
    }

    public int aufgabe04FindeErsteNull()
    {
        return 0;
    }

    public boolean aufgabe05Enthaelt1()
    {
        return false;
    }

    public boolean aufgabe06Enthaelt2Und5()
    {
        return false;
    }

    public boolean aufgabe07EnthaeltFixpunkt()
    {
        return false;
    }

    public int aufgabe08ZaehleWiederholungen()
    {
        return 0;
    }

    public int aufgabe09ZaehleDreierWiederholungen()
    {
        return 0;
    }

    public int aufgabe10LaengsteSerie()
    {
        return 0;
    }

    public int aufgabe11Zweitgroesste()
    {
        return 0;
    }

    public void aufgabe12Plus1()
    {

    }
```

```
    public void aufgabe13NullZuHundert()
    {

    }

    public void aufgabe14Rotation()
    {

    }

    public void aufgabe15Umdrehen()
    {

    }


     /** dient zum Anzeigen der Reihung am Bildschirm;
     * kann durch INSPECT ersetzt werden */
    public void anzeigen() {
        for (int i=0; i< anzahl; i++) {
             System.out.println( i + " :  " + daten[i]);
        }
    }

    /**
     * Gibt eine Zufallszahl zwischen 0 und grenze-1 zurück.
     */
    private int getZufallszahl(int grenze)
    {
        return (int)(grenze*Math.random()+1);
    }

}


/* App Klasse. Steuerklasse für unser Programm */
public class App {

    public static void main(String[] args) {
        Zufallsreihe reihe1 = new Zufallsreihe(100);
        reihe1.anzeigen();
    }

}
```

Aufgabe

- Speichere die Datei array_ausgeben.php[2] auf deinem Webspace.
- Teste das Programm mit unterschiedlichen Zahlenreihen und beobachte, was es macht
- Bearbeite den Quelltext. Du findest 3 Aufgaben im Quelltext. Beantworte die Fragen schriftlich in der Datei.

Aufgabe

- Speichere die Datei sortiert_test.php[3] auf deinem Webspace.
- Teste das Programm mit unterschiedlichen Zahlenreihen und beobachte, was es macht
- Bearbeite den Quelltext. Du findest 3 Aufgaben im Quelltext. Beantworte die Fragen soweit möglich schriftlich in der Datei.

Weiter zu BubbleSort

[1] Diese Geschichte und weitere Ideen der Einführung sind dem Skript der ETH Zürich entnommen

[2], [3] Zip-Datei auspacken!