Sortieralgorithmen

Sortieren ist das Ändern der Reihenfolge der Elemente einer Folge, sodass nachher alle Elemente in aufsteigender oder absteigender Reihenfolge sortiert sind. Die Elementfolge nennt man allgemein einen "*Datensatz*".

Sortieralgorithmen setzen voraus, dass je zwei Elemente nach irgendeinem Kriterium **vergleichbar** sind. Für die Sortieralgorithmen spielt die Art der verglichenen Information keine Rolle.

Die **Qualität** von Sortieralgorithmen wird an der Geschwindigkeit (Anzahl Elementvergleiche und Anzahl Kopieraktionen) gemessen.

Einfache Sortieralgorithmen

Es werden folgende Sortieralgorithmen besprochen:

- **Bubble Sort**
- **Selection Sort**
- **Insertion Sort**

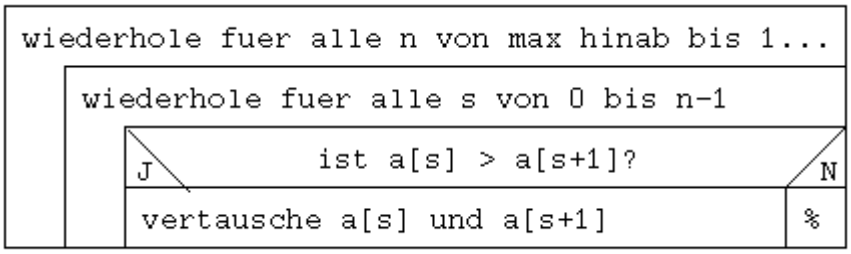
Ziel ist jeweils, dass der Datensatz aufsteigend sortiert wird, d. h. nach dem Sortieren steht das kleinste Element am Anfang und das größte am Ende.

Bubblesort

Der Datensatz wird vom Anfang bis zum Ende durchkämmt und die Elemente dabei paarweise verglichen. Wenn beide in der richtigen Reihenfolge stehen (das kleinere zuerst, das größere hinterher), dann wird mit dem nächsten Paar fortgefahren. Wenn beide in der falschen Reihenfolge stehen, werden sie zuerst vertauscht. Das größte Element wandert dabei an das Ende des Datensatzes.

Dann wird der Prozess wiederholt, aber ohne das letzte Element. Im nächsten Durchgang werden die letzten beiden Elemente ausgelassen usw. Der Datensatz ist vollständig sortiert, wenn nur noch ein Element zu "sortieren" ist.

Das folgende Struktogramm zeigt den Algorithmus schematisch:



Für dieses und alle weiteren Struktogramme wird angenommen, dass der Datensatz (max + 1) Elemente umfasst und in einem Vektor `a` mit Indizes 0 bis `max` gespeichert ist.

Das folgende Beispiel (4 Elemente, d.h. `max` = 3) zeigt die Arbeitsweise. Ein – zwischen zwei Elementen bedeutet, dass ihre Reihenfolge passt. Ein × zwischen zwei Elementen bedeutet, dass sie in der falschen Reihenfolge stehen und von Bubblesort vertauscht werden.

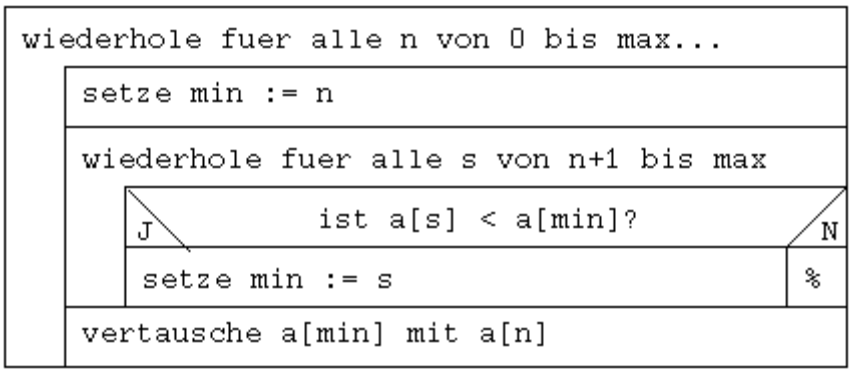
n = 3
44 - 55 12 42
44 55 × 12 42
44 12 55 × 42
n = 2
44 × 12 42 55
12 44 × 42 55
n = 1
12 - 42 44 55

Selection Sort

Der Datensatz wird nach dem kleinsten Element durchsucht, dann wird dieses an den Anfang gesetzt. Im nächsten Schritt wird das erste Element ausgelassen und der gleiche Prozess mit dem Rest des Datensatzes wiederholt. Der Algorithmus kommt zum Ende, wenn nur noch 1 Element übrig ist.

Der Vorteil von Selection Sort gegenüber Bubblesort liegt darin, wie ein Element an seinen Zielort transportiert wird: Bei Bubblesort wird ein Element an seinem Quellort herausgenommen, alle Elemente zwischen dem Quellort und dem Zielort um eine Position verschoben und dann das Element an seinem Zielort eingefügt. Selection Sort vertauscht nur die beiden Elemente am Quell- und Zielort und lässt alle Elemente dazwischen stehen.

Das folgende Struktogramm zeigt den Algorithmus schematisch:



Das folgende Beispiel zeigt die Arbeitsweise. Die jeweils farbig dargestellten Elemente wurden vertauscht. Der senkrechte Strich markiert den Anfang des bearbeiteten Bereichs.

n = 0
44 55 12 42
min = 12
12 55 **44** 42

n = 1
12 | 55 44 42
min = 42
12 | **42** 44 **55**

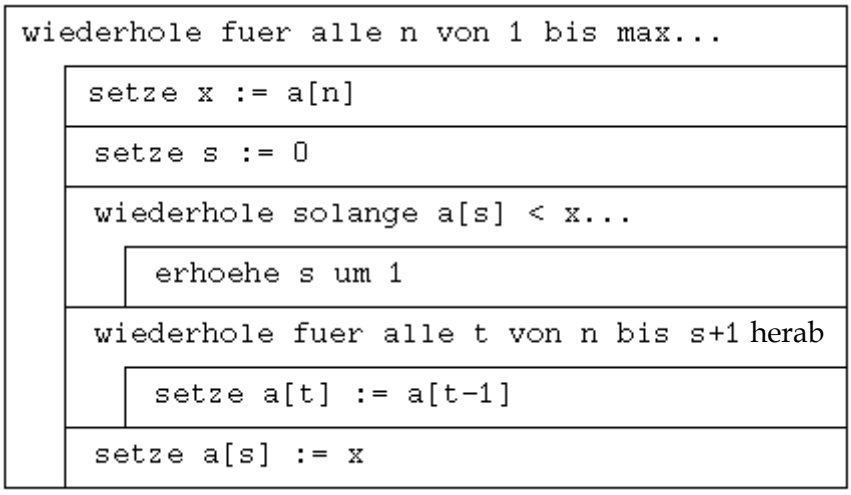
n = 2
12 42 | 44 55
min = 44
keine Änderung

n = 3
keine Änderung

Insertion Sort

Insertion Sort geht in gewissem Sinne den umgekehrten Weg von Selection Sort: Von vorne nach hinten im Datensatz wird ein Element nach dem anderen ausgewählt. Für ein ausgewähltes Element wird im vorderen, schon sortierten Teil des Datensatzes die passende Position gesucht. Dann wird an dieser Position durch Verschieben des restlichen Abschnittes Platz geschaffen und das ausgewählte Element eingefügt (daher der Name des Algorithmus).

Das folgende Struktogramm zeigt den Algorithmus schematisch:



Das folgende Beispiel zeigt die Arbeitsweise. Das jeweils farbig gedruckte Element ist das ausgewählte Element. Das Kreuz markiert die gefundene Einfügestelle.

(n = 0)
x**44** 55 12 42
44 55 12 42
n = 1
44 ×**55** 12 42
44 55 12 42
n = 2
×44 55 **12** 42
12 44 55 42
n = 3
12 ×44 55 **42**
12 42 44 55