Friedrich-Schiller-Universität Jena Fakultät für Mathematik und Informatik Institut für Informatik Lehrstuhl für Programmiersprachen und Compiler	Höhere Programmierung SS 2001	Praktikumblatt 5 Ausgabe: 28.05.2001 Termin: 31.05.2001

Aufgabe 1: Sortieralgorithmen

In diesem Praktikum werden drei Sortieralgorithmen für Arrays von Zahlen vorgestellt. Alle Sortierverfahren sortieren ein Array in aufsteigender Reihenfolge. Überlegen Sie selbst, was geändert werden muss um Arrays in absteigender Reihenfolge zu sortieren.

I. Sortieren durch vertauschen

Für den Sortieralgorithmus soll ein Programm erstellt werden, das bis zu 20 beliebige Zahlen vom Typ FLOAT einliest und in sortierter Reihenfolge wieder ausgibt. Der Benutzer soll durch die Eingabe des Strings “ende“ das Ende der Eingabe markieren. Implementieren Sie entweder Algorithmus A. Bubblesort oder Algorithmus B. Selectionsort.

A. Bubblesort

Der Bubblesort Algorithmus vergleicht immer eine Zahl mit ihrem Nachfolger. Ist der Nachfolger kleiner, dann werden die beiden Zahlen getauscht, sonst bleiben die Zahlen unverändert. Dann wird die nächste Zahl mit ihrem Nachfolger überprüft, bis eine Zahl keinen Nachfolger mehr hat. Auf diese Weise wird die größte Zahl an das Ende verschoben. Die letzte Zahl braucht also nun nicht weiter beachtet werden. Die restlichen Zahlen können aber noch unsortiert sein. Der Algorithmus des Vergleichens und Tauschen muss also noch einmal angewendet werden, dieses Mal braucht die letzte Zahl nicht mehr untersucht zu werden. Nach

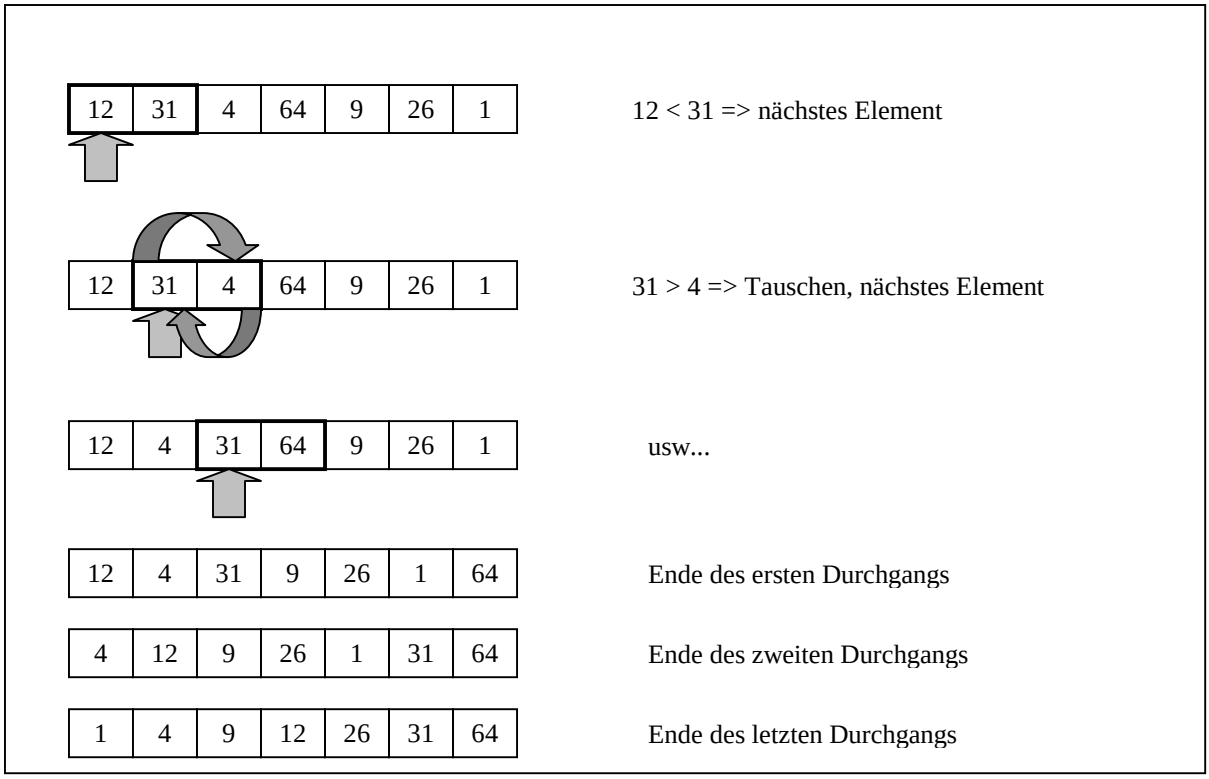

Abbildung 1: Bubblesort-Verfahren

dem zweiten anwenden ist auf dem letzten Platz des Arrays die größte und auf dem vorletzten Platz die zweitgrößte Zahl. Beim nächsten Durchlauf müssen also die letzten beiden Plätze nicht mehr untersucht werden.

Das Verfahren wird solange angewendet, bis nur noch eine Folge mit einer Zahl übrig ist, diese Folge ist dann jedoch sortiert, da eine Folge von einer Zahl immer sortiert ist. Damit ist das Array in aufsteigender Reihenfolge sortiert. Abbildung 1 zeigt noch einmal wie das Verfahren funktioniert.

B. Selectionsort

Ein Nachteil des Bubblesort-Verfahrens sind die vielen Tauschvorgänge. Das Verfahren kann leicht so abgeändert werden, dass weniger Tauschvorgänge notwendig sind.

In diesem Fall benötigen wir eine zusätzliche Variable für die Position des maximalen Wertes. Mittels einer Schleife wird jeder Wert untersucht. Jeder Wert wird mit dem momentanen maximalen Wert verglichen. Ist der untersuchte Wert größer wird seine Position als Position des maximalen Wertes übernommen. Ist der Durchgang beendet wird der letzte Wert mit dem maximalen Wert getauscht. Die restlichen Werte bildet eine neue Folge, die unsortiert sein kann. Der zweite Durchlauf sucht wieder die größte Zahl in der Restfolge und tauscht sie mit dem letzten Wert der Restfolge. Dann wird aus der gerade untersuchten Folge durch Abschneiden des letzten Wertes die neue Restfolge gebildet. Das wird wie beim Bubblesort Algorithmus solange durchgeführt bis nur noch eine Folge von einer Zahl übrig ist, die dann wie unter A. schon erwähnt immer sortiert ist.

Abbildung 2 zeigt einen Überblick über das Verfahren. Dabei stellt der Pfeil die Variable max_position dar, der eingerahmte Wert ist der aktuell analysierte und der dicke Balken zwischen zwei Werten zeigt das Ende des Analysedurchgangs an.

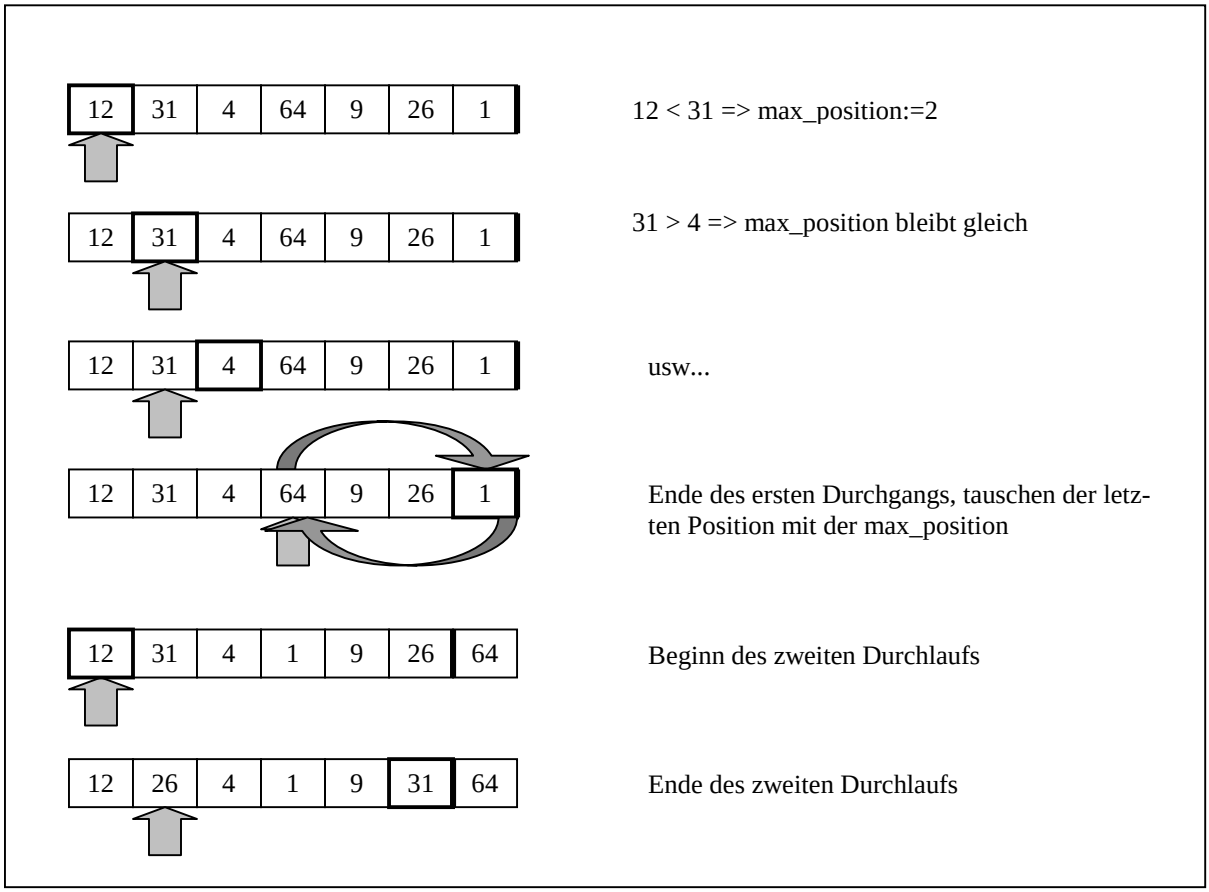

Abbildung 2: Selectsort-Verfahren mit nur einem Tauschvorgang pro Durchlauf

II. Countingsort

Der Countingsort-Algorithmus kann angewendet werden, wenn die zu sortierenden Daten bestimmte Bedingungen erfüllen. Die möglichen Werte müssen diskret und der Wertebereich muss stark beschränkt sein.

Die Aufgabe ist, einen String von Kleinbuchstaben zu sortieren. Der String wird vom Benutzer eingegeben und soll eine Länge von 100 Zeichen nicht überschreiten. Die Kleinbuchstaben werden dann sortiert wieder ausgegeben.

Für den Countingsort-Algorithmus definieren wir ein Array AH für die Häufigkeit eines jeden zu sortierenden Werte. Der Elementtyp des Arrays ist ein Untertyp von Integer, der von 0 bis zur maximalen Eingabelänge definiert ist. Nun wird durch eine Schleife jedes einzelne Element der Eingabe analysiert. Da jedes einzelne Element als Index für das Häufigkeitsarray AH genutzt werden kann, muss keine Abfrage über das Element durchgeführt werden. Das

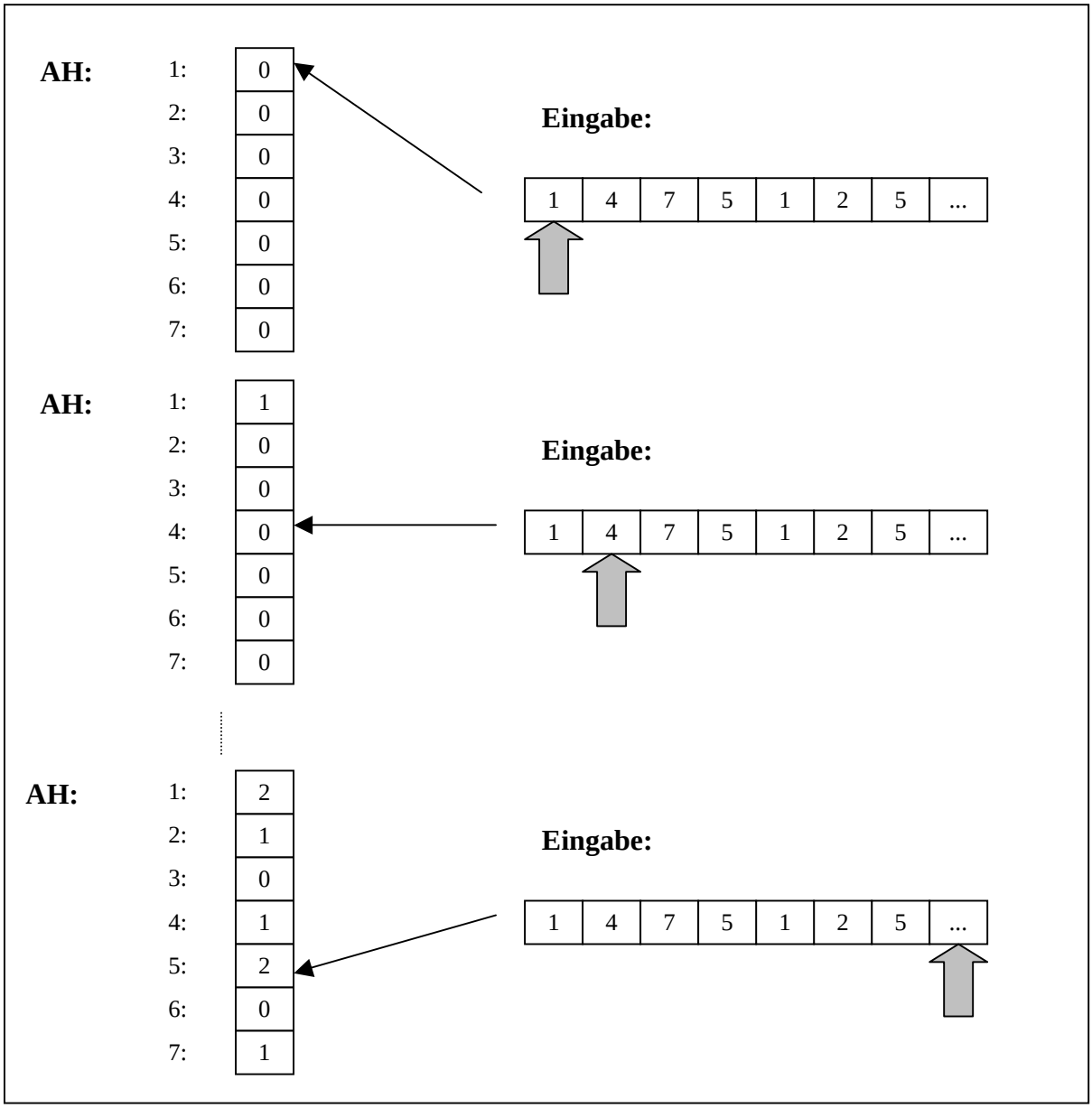

Abbildung 3: Countingsort-Verfahren für Zahlen von 1 bis 7

Element kann direkt als Index zur Erhöhung der entsprechenden Häufigkeit genutzt werden.

Nach dem Durchlauf können die Elemente durch AH in sortierter Reihenfolge ausgegeben werden. Das geschieht durch zwei Schleifen. Die erste Schleife geht die möglichen Elemente in aufsteigender Reihenfolge durch, die zweite Schleife geht die Anzahl durch und gibt je Durchlauf ein Element aus. Die Elemente stehen nach Anwendung des Algorithmus in sortierter Reihenfolge am Bildschirm.

In Abbildung 3 wird das Vorgehen beim Countingsort-Algorithmus dargestellt.