

7 Anwendung: Sortieren

Gegeben: eine Folge von ganzen Zahlen.

Gesucht: die zugehörige aufsteigend sortierte Folge.

Idee:

- ▶ speichere die Folge in einem Feld ab;
- ▶ lege ein weiteres Feld an;
- ▶ füge der Reihe nach jedes Element des ersten Felds an der richtigen Stelle in das zweite Feld ein!

⇒ Sortieren durch Einfügen (↑InsertionSort)

7 Anwendung: Sortieren

```
1 public static int[] sort(int[] a) {
2     int n = a.length;
3     int[] b = new int[n];
4     for (int i = 0; i < n; ++i)
5         insert(b, a[i], i);
6         // b    = Feld, in das eingefuegt wird
7         // a[i] = einzufuegendes Element
8         // i    = Anzahl von Elementen in b
9     return b;
10 } // end of sort ()
```

Sortieren durch Einfügen

Teilproblem: wie fügt man ein?

7 Anwendung: Sortieren

Gegeben: eine Folge von ganzen Zahlen.

Gesucht: die zugehörige aufsteigend sortierte Folge.

Idee:

- ▶ speichere die Folge in einem Feld ab;
- ▶ lege ein weiteres Feld an;
- ▶ füge der Reihe nach jedes Element des ersten Felds an der richtigen Stelle in das zweite Feld ein!

⇒ Sortieren durch Einfügen (↑InsertionSort)

Beispiel

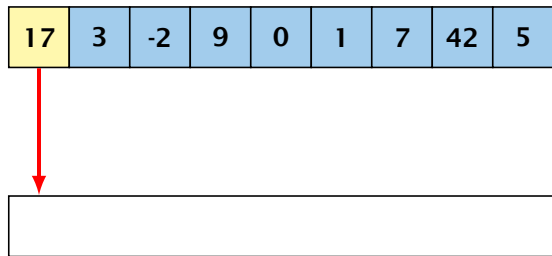
17	3	-2	9	0	1	7	42	5
----	---	----	---	---	---	---	----	---

--

Beispiel



Beispiel

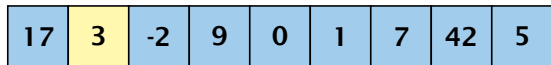


Beispiel

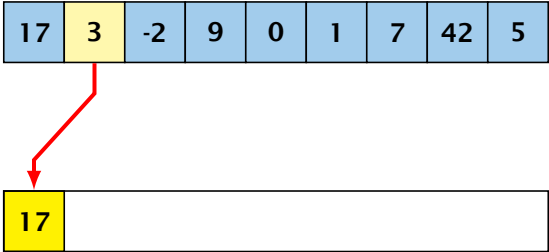
17	3	-2	9	0	1	7	42	5
----	---	----	---	---	---	---	----	---

17								
----	--	--	--	--	--	--	--	--

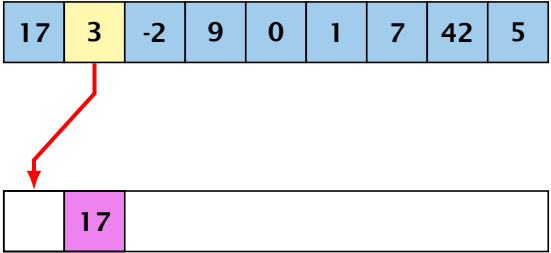
Beispiel



Beispiel



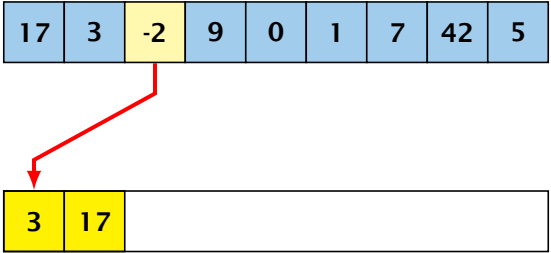
Beispiel



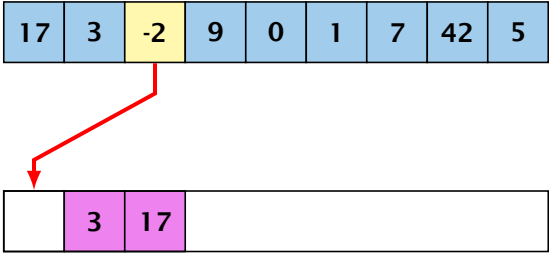
Beispiel



Beispiel



Beispiel

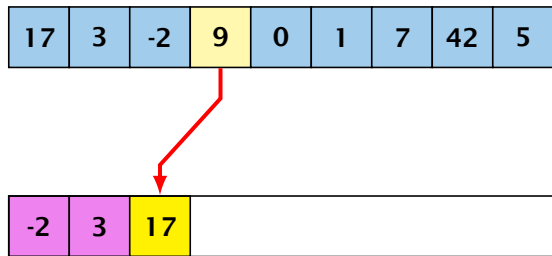


Beispiel

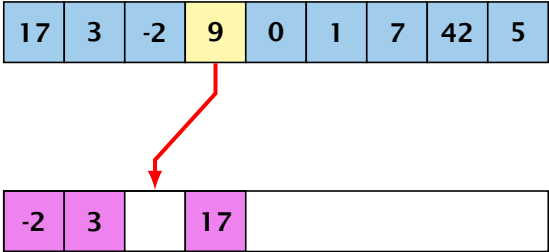
17	3	-2	9	0	1	7	42	5
----	---	----	---	---	---	---	----	---

-2	3	17						
----	---	----	--	--	--	--	--	--

Beispiel



Beispiel

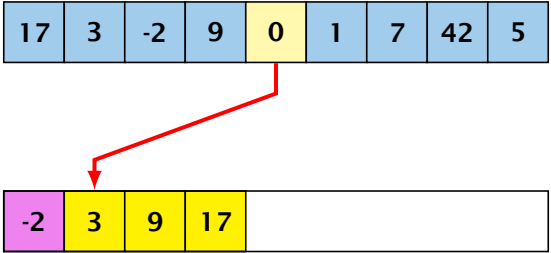


Beispiel

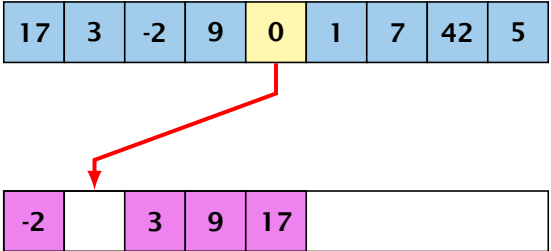
17	3	-2	9	0	1	7	42	5
----	---	----	---	---	---	---	----	---

-2	3	9	17					
----	---	---	----	--	--	--	--	--

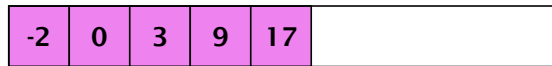
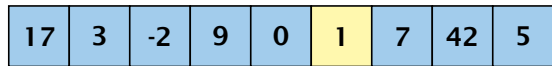
Beispiel



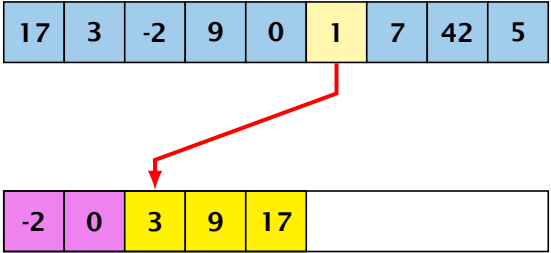
Beispiel



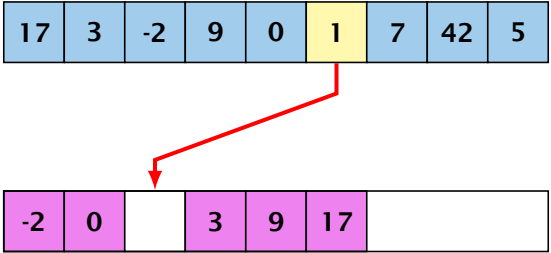
Beispiel



Beispiel



Beispiel

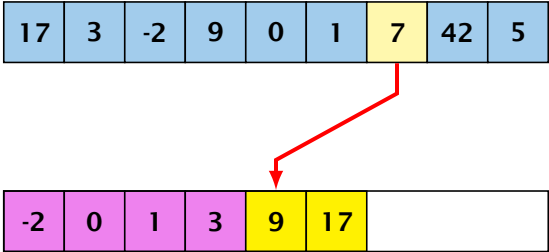


Beispiel

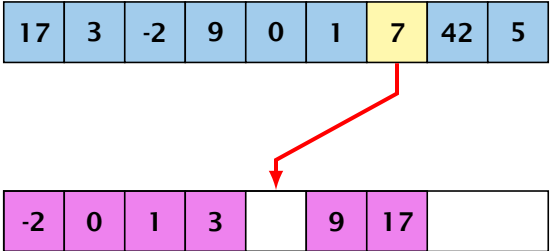
17	3	-2	9	0	1	7	42	5
----	---	----	---	---	---	---	----	---

-2	0	1	3	9	17	
----	---	---	---	---	----	--

Beispiel



Beispiel

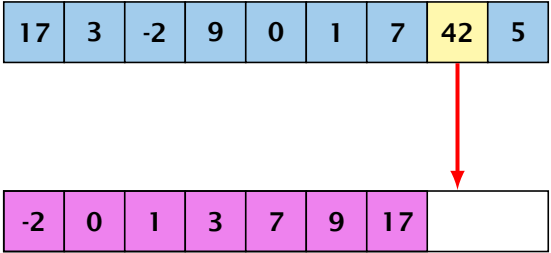


Beispiel

17	3	-2	9	0	1	7	42	5
----	---	----	---	---	---	---	----	---

-2	0	1	3	7	9	17	
----	---	---	---	---	---	----	--

Beispiel

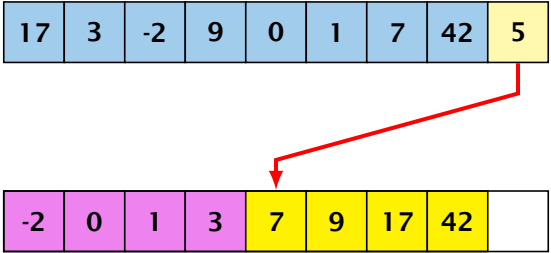


Beispiel

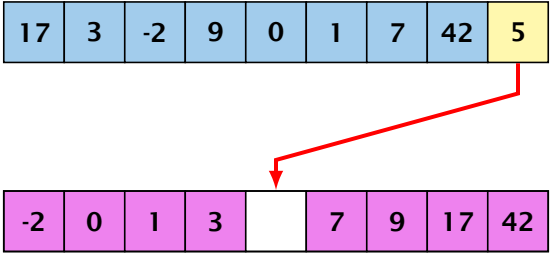
17	3	-2	9	0	1	7	42	5
----	---	----	---	---	---	---	----	---

-2	0	1	3	7	9	17	42	
----	---	---	---	---	---	----	----	--

Beispiel



Beispiel



Beispiel

17	3	-2	9	0	1	7	42	5
----	---	----	---	---	---	---	----	---

-2	0	1	3	5	7	9	17	42
----	---	---	---	---	---	---	----	----

7 Anwendung: Sortieren

```
1 public static void insert(int[] b, int x, int i) {  
2     // finde Einfuegestelle j fuer x in b  
3     int j = locate(b,x,i);  
4     // verschiebe in b Elemente b[j],...,b[i-1]  
5     // nach rechts  
6     shift(b,j,i);  
7     b[j] = x;  
8 }
```

Einfügen

- ▶ Wie findet man Einfügestelle?
- ▶ Wie verschiebt man nach rechts?

7 Anwendung: Sortieren

```
public static int locate(int[] b, int x, int i) {
    int j = 0;
    while (j < i && x > b[j]) ++j;
    return j;
}
public static void shift(int[] b, int j, int i) {
    for (int k = i-1; k >= j; --k)
        b[k+1] = b[k];
}
```

- ▶ Warum läuft Iteration in `shift()` von `i-1` **abwärts** nach `j`?

7 Anwendung: Sortieren

```
1 public static void insert(int[] b, int x, int i) {
2     // finde Einfuegestelle j fuer x in b
3     int j = locate(b,x,i);
4     // verschiebe in b Elemente b[j],...,b[i-1]
5     // nach rechts
6     shift(b,j,i);
7     b[j] = x;
8 }
```

Einfügen

- ▶ Wie findet man Einfügestelle?
- ▶ Wie verschiebt man nach rechts?

7 Anwendung: Sortieren

Erläuterungen

- ▶ Das Feld `b` ist (ursprünglich) lokale Variable von `sort()`.
- ▶ Lokale Variablen sind nur im eigenen Funktionsrumpf sichtbar, nicht in den aufgerufenen Funktionen.
- ▶ Damit die aufgerufenen Hilfsfunktionen auf `b` zugreifen können, muss `b` explizit als Parameter übergeben werden!

Achtung:

Das Feld wird nicht kopiert. Das Argument ist der Wert der Variablen `b`, also nur eine Referenz!

- ▶ Deshalb benötigen weder `insert()`, noch `shift()` einen separaten Rückgabewert. . .
- ▶ Weil das Problem so klein ist, würde eine erfahrene Programmiererin hier keine Unterprogramme benutzen...

7 Anwendung: Sortieren

```
public static int locate(int[] b, int x, int i) {
    int j = 0;
    while (j < i && x > b[j]) ++j;
    return j;
}
public static void shift(int[] b, int j, int i) {
    for (int k = i-1; k >= j; --k)
        b[k+1] = b[k];
}
```

- ▶ Warum läuft Iteration in `shift()` von `i-1` abwärts nach `j`?

7 Anwendung: Sortieren

```
1 public static int[] sort(int[] a) {
2     int[] b = new int[a.length];
3     for (int i = 0; i < a.length; ++i) {
4         // begin of insert
5         int j = 0;
6         while (j < i && a[i] > b[j]) ++j;
7         // end of locate
8         for (int k = i-1; k >= j; --k)
9             b[k+1] = b[k];
10        // end of shift
11        b[j] = a[i];
12        // end of insert
13    }
14    return b;
15 } // end of sort
```

7 Anwendung: Sortieren

Erläuterungen

- ▶ Das Feld **b** ist (ursprünglich) **lokale** Variable von **sort()**.
- ▶ Lokale Variablen sind nur im eigenen Funktionsrumpf sichtbar, nicht in den aufgerufenen Funktionen.
- ▶ Damit die aufgerufenen Hilfsfunktionen auf **b** zugreifen können, muss **b** explizit als Parameter übergeben werden!

Achtung:

Das Feld wird nicht kopiert. Das Argument ist der Wert der Variablen **b**, also nur eine **Referenz!**

- ▶ Deshalb benötigen weder **insert()**, noch **shift()** einen separaten Rückgabewert. . .
- ▶ Weil das Problem so klein ist, würde eine **erfahrene** Programmiererin hier keine Unterprogramme benutzen...

Diskussion

- ▶ Die Anzahl der ausgeführten Operationen wächst quadratisch in der Größe des Felds **a**.
- ▶ Glücklicherweise gibt es Sortierverfahren, die eine bessere Laufzeit haben ([↑Algorithmen und Datenstrukturen](#)).

```
1 public static int[] sort(int[] a) {
2     int[] b = new int[a.length];
3     for (int i = 0; i < a.length; ++i) {
4         // begin of insert
5         int j = 0;
6         while (j < i && a[i] > b[j]) ++j;
7         // end of locate
8         for (int k = i-1; k >= j; --k)
9             b[k+1] = b[k];
10        // end of shift
11        b[j] = a[i];
12        // end of insert
13    }
14    return b;
15 } // end of sort
```