Einführung in die Objektorientierte Programmierung
Vorlesung 15: Sortieralgorithmen

Sebastian Küpper

Suchen und Sortieren

- Aufgabenstellung: Finden eines Elements in einem Feld
- Suchen nach bestimmten Elementen, Minima oder Maxima in beliebigen Feldern ist zeitaufwendig
- Daher interessant: Sortierte Felder

Suchen und Sortieren

- Aufgabenstellung: Finden eines Elements in einem Feld
- Suchen nach bestimmten Elementen, Minima oder Maxima in beliebigen Feldern ist zeitaufwendig
- Daher interessant: Sortierte Felder
- Hier: Grundlegende Algorithmen
- Später (Vorlesung 16) auch fortgeschrittene Algorithmen

Lineare Suche in einem Feld

- Gegeben: Ein Element `wert` und ein Feld `feld`.
- Gesucht: `true`, falls `wert` in `feld` ist, sonst `false`

Lineare Suche in einem Feld

- Gegeben: Ein Element `wert` und ein Feld `feld`.
- Gesucht: `true`, falls `wert` in `feld` ist, sonst `false`

Algorithmus:

1. Falls das Feld leer ist, gib `false` zurück
2. Setze den aktuellen Eintrag `i` auf das erste Element des Feldes
3. Falls das Ende des Feldes erreicht wurde, gib `false` zurück
4. Falls `i=wert` gilt, dann gib `true` zurück
5. Setze den aktuellen Eintrag `i` auf den Nachfolger vom bisherigen aktuellen Eintrag in `feld` und setze bei Schritt 3 fort.

Implementierung Lineare Suche in einem Feld

```
boolean istEnthalten(int wert, int[] feld) {
   for (int i : feld) {
      if (wert == i) {
         return true;
      }
   }
   return false;
}
```

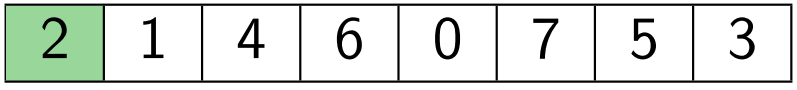
Beispiel lineare Suche

Wir wollen überprüfen ob die folgende Liste den Eintrag 0 hat:

2	1	4	6	0	7	5	3

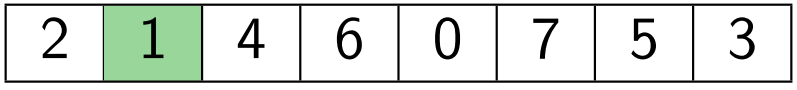
Beispiel lineare Suche

Wir wollen überprüfen ob die folgende Liste den Eintrag 0 hat:



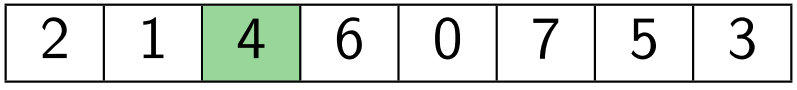
Beispiel lineare Suche

Wir wollen überprüfen ob die folgende Liste den Eintrag 0 hat:



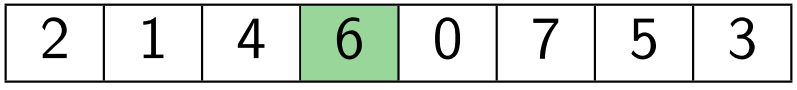
Beispiel lineare Suche

Wir wollen überprüfen ob die folgende Liste den Eintrag 0 hat:



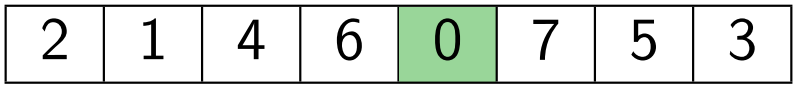
Beispiel lineare Suche

Wir wollen überprüfen ob die folgende Liste den Eintrag 0 hat:



Beispiel lineare Suche

Wir wollen überprüfen ob die folgende Liste den Eintrag 0 hat:



Gib `true` zurück.

Optimierung falls Feld sortiert ist

Man kann bereits abbrechen, wenn man ein größeres Element gefunden hat:

1. Falls das Feld leer ist, gib `false` zurück
2. Setze den aktuellen Eintrag `i` auf das erste Element des Feldes
3. Falls das Ende des Feldes erreicht wurde, gib `false` zurück
4. Falls `i=wert` gilt, dann gib `true` zurück
5. Falls `i>wert` gilt, dann gib `false` zurück
6. Setze den aktuellen Eintrag `i` auf den Nachfolger vom bisherigen aktuellen Eintrag in `feld` und setze bei Schritt 3 fort.

Implementierung Lineare Suche (Sortiert)

```
boolean istEnthalten(int wert, int[] feld) {
   for (int i : feld) {
      if (wert == i) {
         return true;
      }
      if (wert < i) {
         return false;
      }
   }
   return false;
}
```

Beispiel lineare Suche (Sortiert)

Wir wollen überprüfen ob die folgende Liste den Eintrag 5 hat:

0	1	2	3	8	9	14	20

Beispiel lineare Suche (Sortiert)

Wir wollen überprüfen ob die folgende Liste den Eintrag 5 hat:

0	1	2	3	8	9	14	20

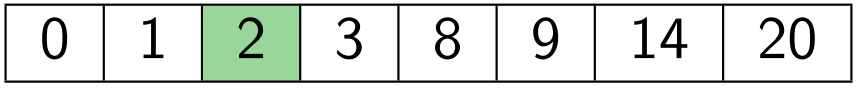
Beispiel lineare Suche (Sortiert)

Wir wollen überprüfen ob die folgende Liste den Eintrag 5 hat:

0	1	2	3	8	9	14	20

Beispiel lineare Suche (Sortiert)

Wir wollen überprüfen ob die folgende Liste den Eintrag 5 hat:



Beispiel lineare Suche (Sortiert)

Wir wollen überprüfen ob die folgende Liste den Eintrag 5 hat:

0	1	2	3	8	9	14	20

Beispiel lineare Suche (Sortiert)

Wir wollen überprüfen ob die folgende Liste den Eintrag 5 hat:

0	1	2	3	8	9	14	20

Gib `false` zurück.

Verwandte Fragestellung: Suche Minimum / Maximum

- Trivial in sortierter Liste: Gib ersten / letzten Eintrag zurück
- Komplizierter bei unsortierter Liste: Kleinstes / größtes Element in Variable merken

Finden des Minimums

Man kann bereits abbrechen, wenn man ein größeres Element gefunden hat:

1. Falls das Feld leer ist, gib ∞ zurück
2. Setze `minimum` auf ∞
3. Setze den aktuellen Eintrag `i` auf das erste Element des Feldes
4. Falls das Ende des Feldes erreicht wurde, gib `minimum` zurück
5. Falls `i`$<$`minimum` gilt, setze `minimum` auf `i`
6. Setze den aktuellen Eintrag `i` auf den Nachfolger vom bisherigen aktuellen Eintrag in `feld` und setze bei Schritt 4 fort.

Implementierung Finden des Minimums

```
boolean istEnthalten(int wert, int[] feld) {
   int minimum = Integer.MAX_VALUE
   for (int i : feld) {
      if (minimum > i) {
         minimum = i;
      }
   }
   return minimum;
}
```

Beispiel Finden des Minimums

Wir wollen das Minimum des folgenden Feldes finden:

2	1	4	6	0	7	5	3

Beispiel Finden des Minimums

Wir wollen das Minimum des folgenden Feldes finden:

2	1	4	6	0	7	5	3

`Minimum`: ∞

Beispiel Finden des Minimums

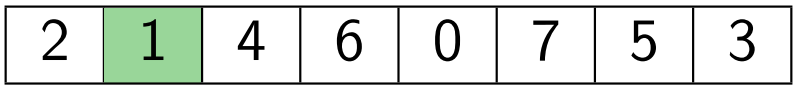
Wir wollen das Minimum des folgenden Feldes finden:

2	1	4	6	0	7	5	3

`Minimum`: 2

Beispiel Finden des Minimums

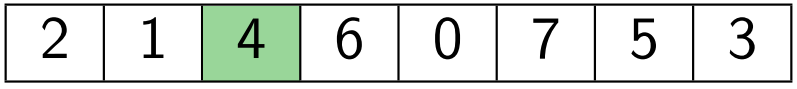
Wir wollen das Minimum des folgenden Feldes finden:



`Minimum`: 1

Beispiel Finden des Minimums

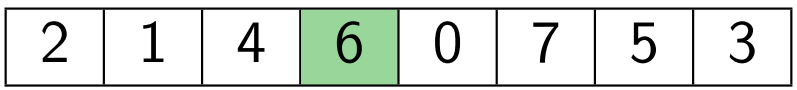
Wir wollen das Minimum des folgenden Feldes finden:



`Minimum`: 1

Beispiel Finden des Minimums

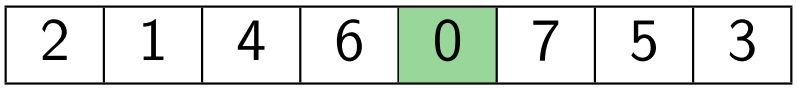
Wir wollen das Minimum des folgenden Feldes finden:



`Minimum`: 1

Beispiel Finden des Minimums

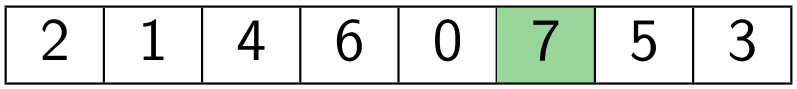
Wir wollen das Minimum des folgenden Feldes finden:



`Minimum`: 0

Beispiel Finden des Minimums

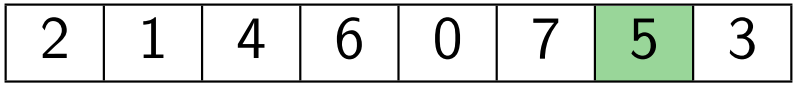
Wir wollen das Minimum des folgenden Feldes finden:



`Minimum`: 0

Beispiel Finden des Minimums

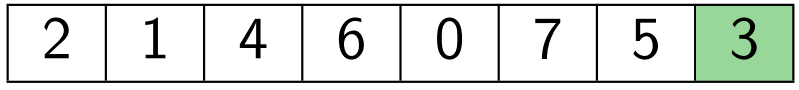
Wir wollen das Minimum des folgenden Feldes finden:



`Minimum`: 0

Beispiel Finden des Minimums

Wir wollen das Minimum des folgenden Feldes finden:



`Minimum`: 0
Gib 0 zurück.

Verwandte Fragestellung: Suche Teilsequenz

- Gegeben: Eine Sequenz `gesuchteFolge` und ein Feld `feld`.
- Gesucht: `true`, falls `gesuchteFolge` Teilefolge in `feld` ist, sonst `false`

Verwandte Fragestellung: Suche Teilsequenz

- Gegeben: Eine Sequenz `gesuchteFolge` und ein Feld `feld`.
- Gesucht: `true`, falls `gesuchteFolge` Teilefolge in `feld` ist, sonst `false`

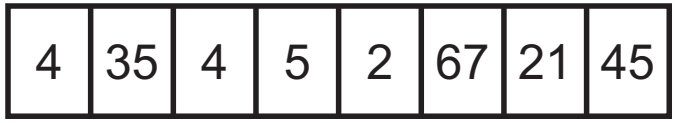
Algorithmus:

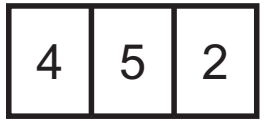
1. Falls das Feld leer ist, gib `false` zurück
2. Setze den aktuellen Eintrag `i` auf das erste Element des Feldes
3. Falls das Ende des Feldes erreicht wurde, gib `false` zurück
4. Falls `i=gesuchteFolge[0]` und der Nachfolger von `i=gesuchteFolge[1]` gilt, und (...) dann gib `true` zurück
5. Setze den aktuellen Eintrag `i` auf den Nachfolger vom bisherigen aktuellen Eintrag in `feld` und setze bei Schritt 3 fort.

Implementierung Finden des Minimums

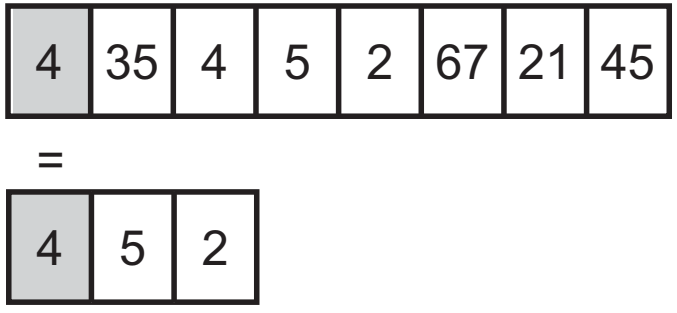
```
boolean istTeilfolge(int[] feld, int[] gesuchteFolge) {
   for (int i = 0; i < feld.length - gesuchteFolge.length
       + 1; i++) {
      for (int j = 0; j < gesuchteFolge.length; j++) {
         if (feld[i + j] != gesuchteFolge[j]) { // Folge
             nicht an Position i
            break;
         } else if (j == gesuchteFolge.length - 1) { //
             Folge gefunden
            return true;
         }
      }
   }
   return false;
}
```

Beispiel: Suche einer Teilfolge

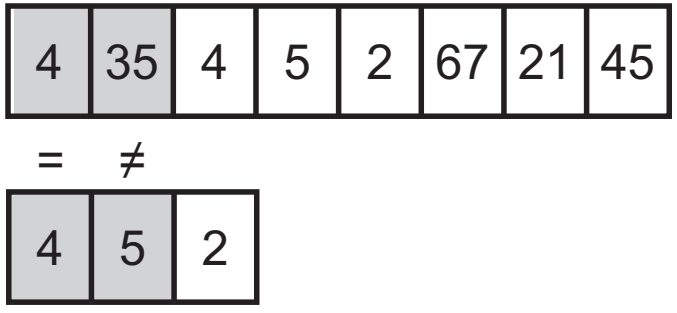




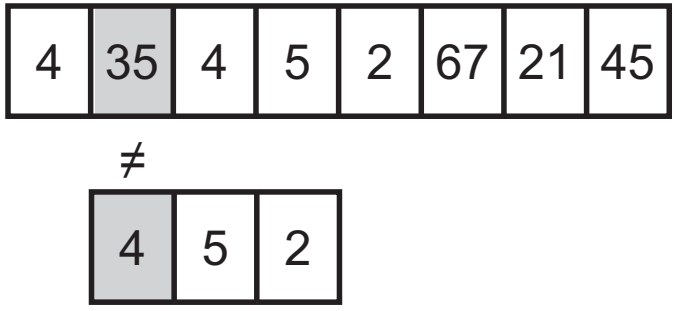
Beispiel: Suche einer Teilfolge



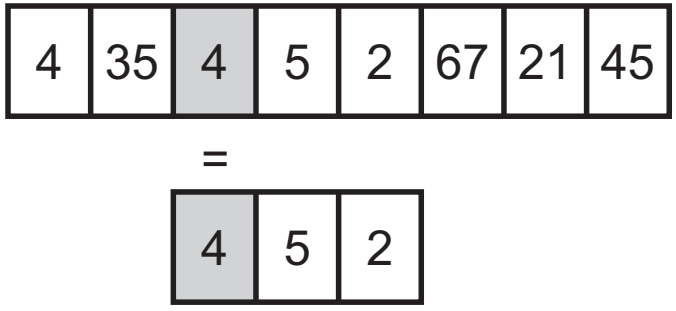
Beispiel: Suche einer Teilfolge



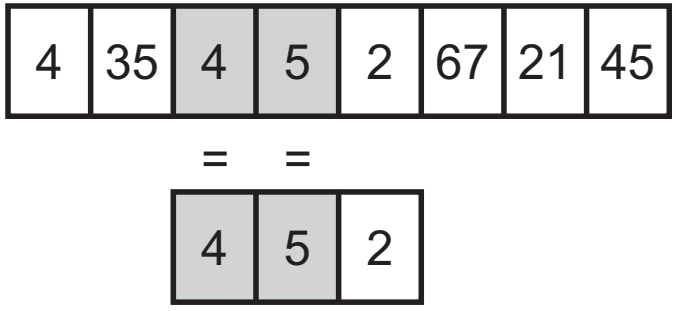
Beispiel: Suche einer Teilfolge



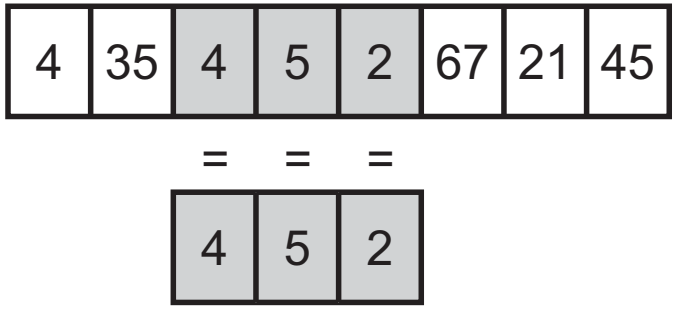
Beispiel: Suche einer Teilfolge



Beispiel: Suche einer Teilfolge



Beispiel: Suche einer Teilfolge



Gib `true` zurück.

Sortieren beim Einfügen

Sortieren wie beim Kartenspiel üblich. Ziel: Sortiere das Feld `feld`:

1. definiere leeres Feld `sortiertesFeld`
2. Falls `feld` leer ist, gibt `sortiertesFeld` zurück
3. Entferne das erste Element `aktuell` aus `feld`
4. Durchlaufe `feld` von links nach rechts bis das Ende von `feld` erreicht ist, oder ein Element gefunden wird, das größer ist als `aktuell`
5. Füge `aktuell` zu `feld` an der gefundenen Stelle an – als letztes Element oder direkt vor dem ersten größeren.
6. Gehe zu Schritt 2.

Implementierung Sortieren beim Einfügen

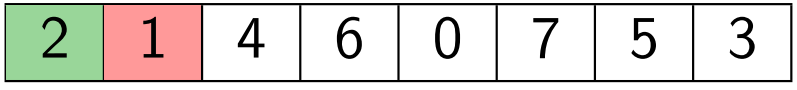
In der Implementation: links von Index *i* sortierte Teilliste, rechts: zu sortierende Teilliste

```
void sortiereAufsteigend(int[] feld) {
for (int i = 1; i < feld.length; i++) {
   for (int j = i; j > 0; j--) {
      if (feld[j - 1] > feld[j]) { // vertausche solange
          links groesser
         int temp = feld[j];
         feld[j] = feld[j - 1];
         feld[j - 1] = temp;
      } else {
         break; // Element an der richtigen Stelle
      }
   }
}
```

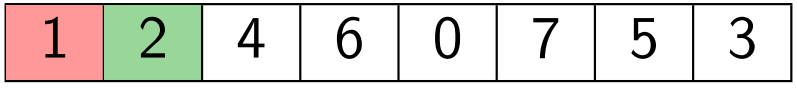
Beispiel: Sortieren beim Einfügen

2	1	4	6	0	7	5	3

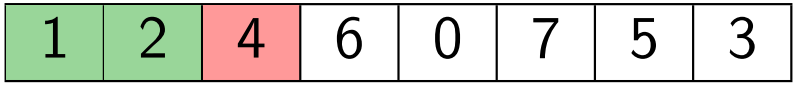
Beispiel: Sortieren beim Einfügen



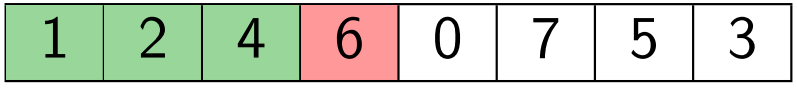
Beispiel: Sortieren beim Einfügen



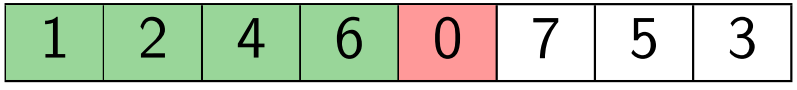
Beispiel: Sortieren beim Einfügen



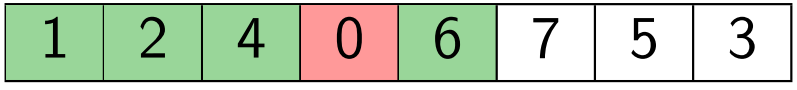
Beispiel: Sortieren beim Einfügen



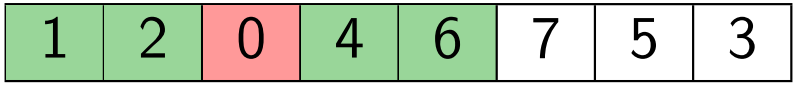
Beispiel: Sortieren beim Einfügen



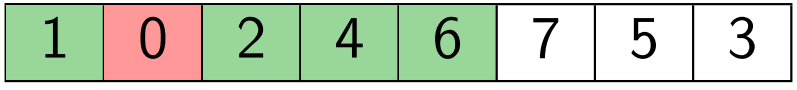
Beispiel: Sortieren beim Einfügen



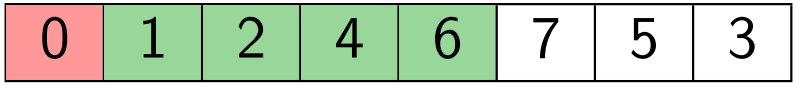
Beispiel: Sortieren beim Einfügen



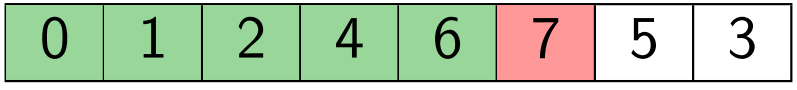
Beispiel: Sortieren beim Einfügen



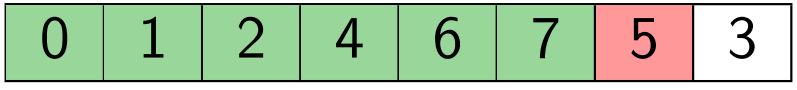
Beispiel: Sortieren beim Einfügen



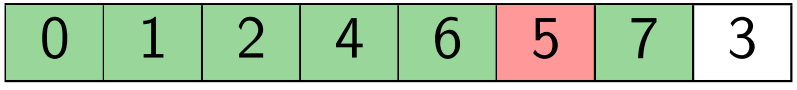
Beispiel: Sortieren beim Einfügen



Beispiel: Sortieren beim Einfügen

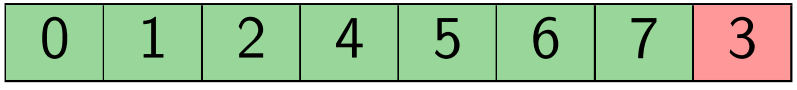


Beispiel: Sortieren beim Einfügen

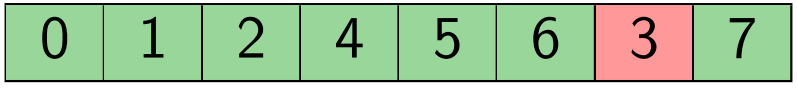


Beispiel: Sortieren beim Einfügen

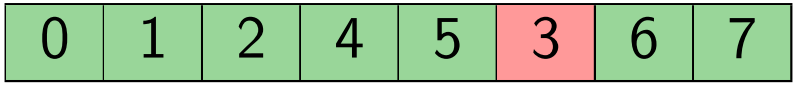
Beispiel: Sortieren beim Einfügen



Beispiel: Sortieren beim Einfügen



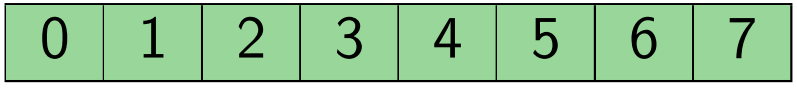
Beispiel: Sortieren beim Einfügen



Beispiel: Sortieren beim Einfügen

Beispiel: Sortieren beim Einfügen

Beispiel: Sortieren beim Einfügen



Bubblesort

Idee: große Elemente steigen wie Blasen in einer Flüssigkeit auf.
Ziel: Sortiere das Feld `feld`:

1. Durchlaufe das Feld von links nach rechts
2. Falls ein größeres Element links von einem kleineren steht, vertausche die beiden Elemente
3. Falls das Ende des Felds erreicht wird, ohne eine Vertauschung vorgenommen zu haben, terminiere
4. Falls das Ende des Feldes nach einer Vertauschung erreicht wird, gehe zu Schritt 1.

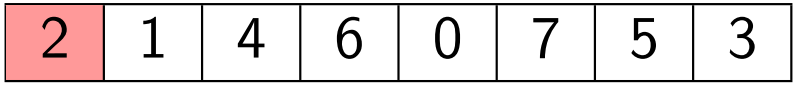
Implementierung Bubblesort

```
void bubblesort(int[] feld) {
   for (int i = 0; i < feld.length - 1; i++) {
      boolean sorted = true; // Vertauschen merken
      for (int j = 0; j < feld.length - 1 - i; j++) {
         if (feld[j] > feld[j + 1]) {
            int temp = feld[j];
            feld[j] = feld[j + 1];
            feld[j + 1] = temp;
            sorted = false;
         }
      }
      if (sorted) break;
   }
}
```

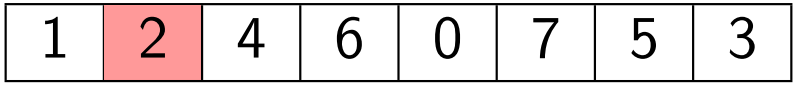
Beispiel: Bubblesort

2	1	4	6	0	7	5	3

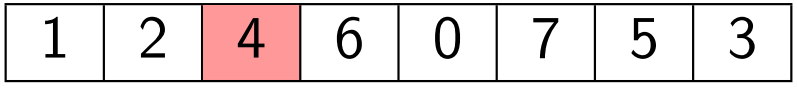
Beispiel: Bubblesort



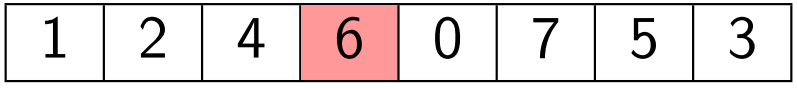
Beispiel: Bubblesort



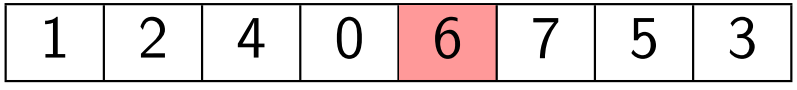
Beispiel: Bubblesort



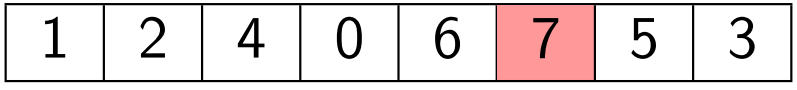
Beispiel: Bubblesort



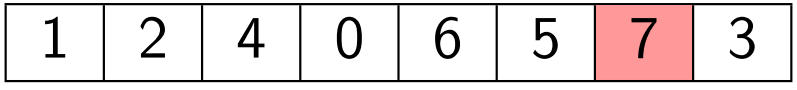
Beispiel: Bubblesort



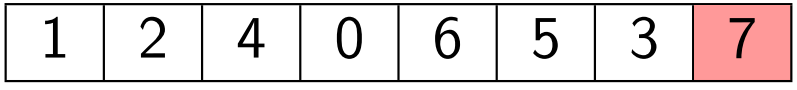
Beispiel: Bubblesort



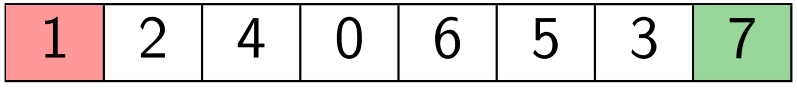
Beispiel: Bubblesort



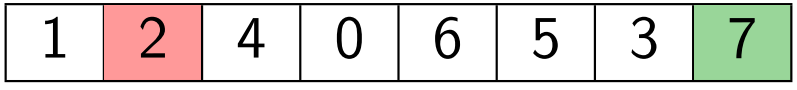
Beispiel: Bubblesort



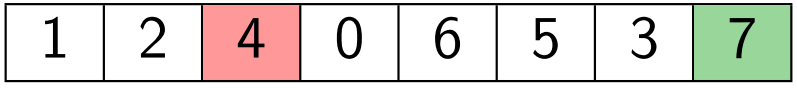
Beispiel: Bubblesort



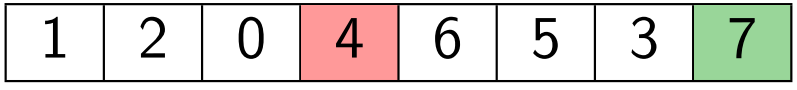
Beispiel: Bubblesort



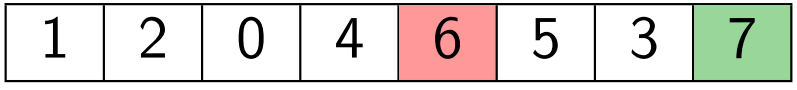
Beispiel: Bubblesort



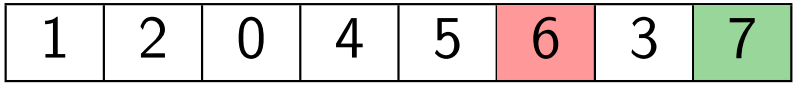
Beispiel: Bubblesort



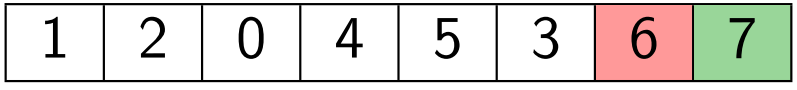
Beispiel: Bubblesort



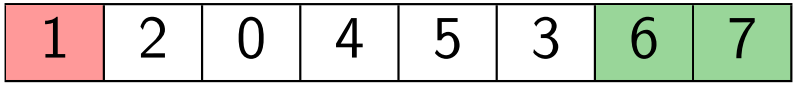
Beispiel: Bubblesort



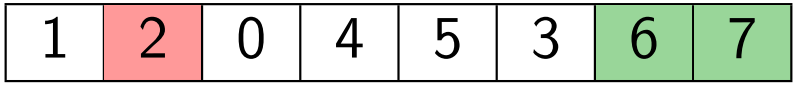
Beispiel: Bubblesort



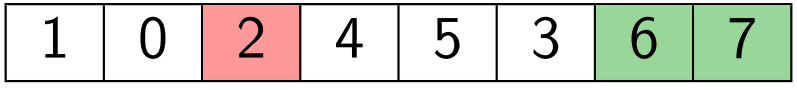
Beispiel: Bubblesort



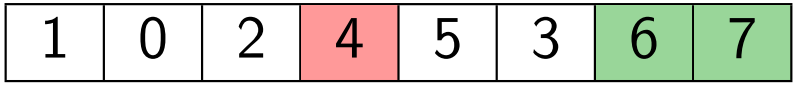
Beispiel: Bubblesort



Beispiel: Bubblesort

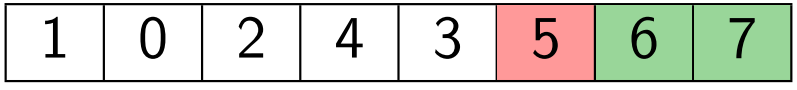


Beispiel: Bubblesort

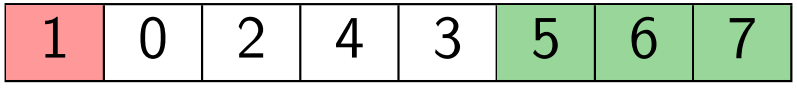


Beispiel: Bubblesort

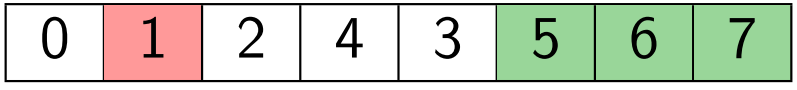
Beispiel: Bubblesort



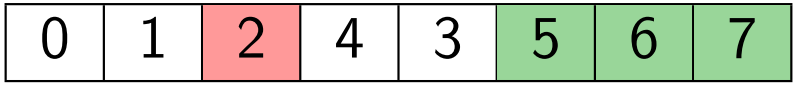
Beispiel: Bubblesort



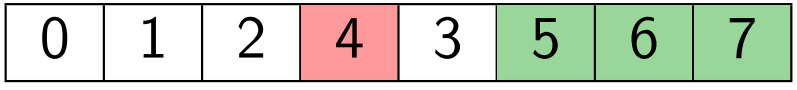
Beispiel: Bubblesort



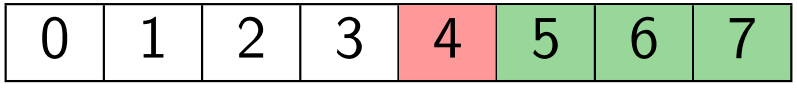
Beispiel: Bubblesort



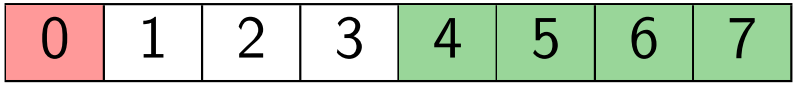
Beispiel: Bubblesort



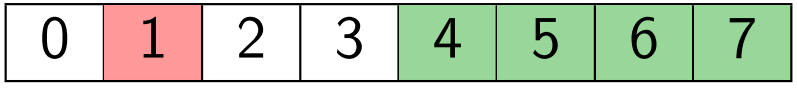
Beispiel: Bubblesort



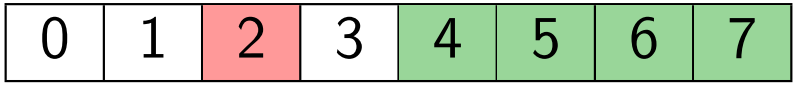
Beispiel: Bubblesort



Beispiel: Bubblesort

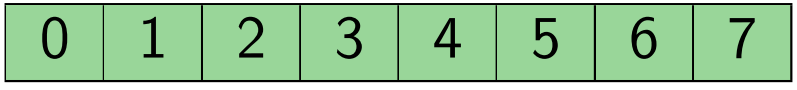


Beispiel: Bubblesort



Beispiel: Bubblesort

Beispiel: Bubblesort



Fragen zur Vorlesungseinheit

1. Wie wird die lineare Suche und die Suche nach Maximum / Minimum durch sortiere Felder beschleunigt?
2. Welche Rolle spielt die Variable `minimum` beim Algorithmus zur Suche nach dem Minimum?
3. Wie funktioniert Bubblesort?