Bubble Sort

>> Bubble Sort ist ein einfacher Algorithmus zum Sortieren von Elementen.

Beim Bubble Sort besteht die Idee darin, Elemente mit höherem Wert nach rechts und Elemente mit niedrigerem Wert nach links zu verschieben.

Der Name *Bubble Sort*, auf Deutsch Blasensortierung, soll andeuten, dass die großen Elemente wie Luftblasen im Wasser aufsteigen.

Pseudocode

Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

    Setze Tauschzähler auf 0.

    Iteriere über alle benachbarten Paare von Elementen:

        Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Tauschzähler | −1

5	2	1	3	6	4

Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

> Setze Tauschzähler auf 0.
>
> Iteriere über alle benachbarten Paare von Elementen:
>
> > Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Tauschzähler 0

Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

Setze Tauschzähler auf 0.

Iteriere über alle benachbarten Paare von Elementen:

Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Tauschzähler 1

Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

Setze Tauschzähler auf 0.

Iteriere über alle benachbarten Paare von Elementen:

Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Tauschzähler | 1

2	5	1	3	6	4

Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

> Setze Tauschzähler auf 0.
>
> Iteriere über alle benachbarten Paare von Elementen:
>
> > Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Tauschzähler | 2 |

2	1	5	3	6	4

Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

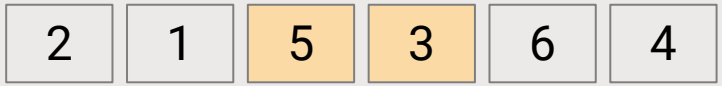
Setze Tauschzähler auf 0.

Iteriere über alle benachbarten Paare von Elementen:

Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Tauschzähler 2



Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

Setze Tauschzähler auf 0.

Iteriere über alle benachbarten Paare von Elementen:

Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Tauschzähler 3

Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

Setze Tauschzähler auf 0.

Iteriere über alle benachbarten Paare von Elementen:

Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Tauschzähler 3

Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

Setze Tauschzähler auf 0.

Iteriere über alle benachbarten Paare von Elementen:

Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Tauschzähler 3

Setze *Tauschzähler* auf einen Wert ungleich Null.

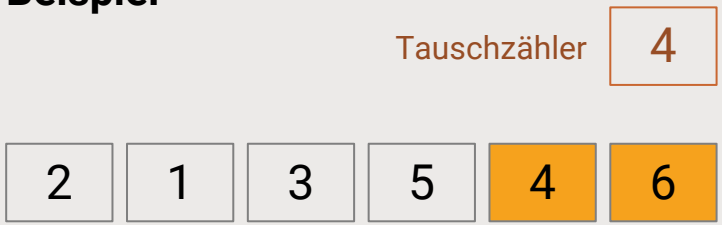
Wiederhole, bis Tauschzähler 0 ist:

Setze Tauschzähler auf 0.

Iteriere über alle benachbarten Paare von Elementen:

Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel



Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

- Setze Tauschzähler auf 0.
- Iteriere über alle benachbarten Paare von Elementen:
  - Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Tauschzähler 4

Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

Setze Tauschzähler auf 0.

Iteriere über alle benachbarten Paare von Elementen:

Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Tauschzähler 0

Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

Setze Tauschzähler auf 0.

Iteriere über alle benachbarten Paare von Elementen:

Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Tauschzähler 1

Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

Setze Tauschzähler auf 0.

Iteriere über alle benachbarten Paare von Elementen:

Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Tauschzähler 1

Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

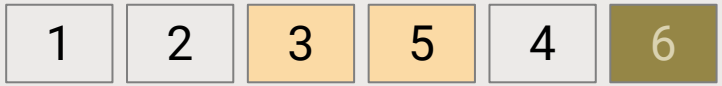
Setze Tauschzähler auf 0.

Iteriere über alle benachbarten Paare von Elementen:

Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Tauschzähler 1



Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

Setze Tauschzähler auf 0.

Iteriere über alle benachbarten Paare von Elementen:

Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Tauschzähler 1

Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

Setze Tauschzähler auf 0.

Iteriere über alle benachbarten Paare von Elementen:

Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

Setze Tauschzähler auf 0.

Iteriere über alle benachbarten Paare von Elementen:

Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Tauschzähler 2

Setze *Tauschzähler* auf einen Wert ungleich Null.

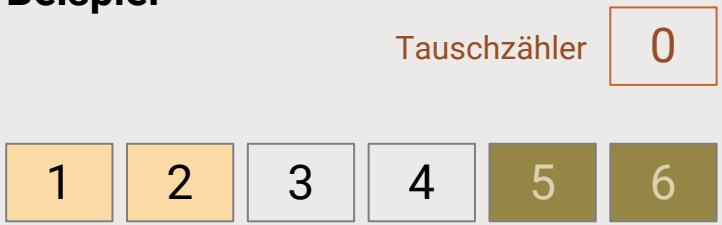
Wiederhole, bis Tauschzähler 0 ist:

Setze Tauschzähler auf 0.

Iteriere über alle benachbarten Paare von Elementen:

Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel



Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

Setze Tauschzähler auf 0.

Iteriere über alle benachbarten Paare von Elementen:

Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

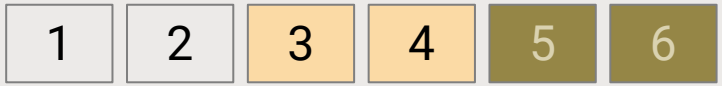
Setze Tauschzähler auf 0.

Iteriere über alle benachbarten Paare von Elementen:

Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Tauschzähler 0



Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

Setze Tauschzähler auf 0.

Iteriere über alle benachbarten Paare von Elementen:

Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Beispiel

Tauschzähler 0

Setze *Tauschzähler* auf einen Wert ungleich Null.

Wiederhole, bis Tauschzähler 0 ist:

> Setze Tauschzähler auf 0.
>
> Iteriere über alle benachbarten Paare von Elementen:
>
> > Wenn zwei benachbarte Elemente nicht in der richtigen Reihenfolge sind, vertausche ihre Position und erhöhe den Tauschzähler um 1.

Fälle

Worst Case:
Array ist in umgekehrter Reihenfolge.

Dann müssen wir jedes der *n* Elemente quer durch das Array schieben, und da wir nur ein Element pro Durchgang vollständig an die richtige Position bringen können, müssen wir dies *n* mal tun.

Best Case:
Das Array ist bereits perfekt sortiert.

Dann müssen wir schon im ersten Durchlauf keine Vertauschungen vornehmen.

Laufzeit-komplexität

$O(n^2)$

$\Omega(n)$

EXTRAS IN 3 MINUTEN

FRAGEN – ANTWORTEN – RÄTSEL
UND KURZE ZUSAMMENFASSUNG

F1: Was ist die Aufgabe des *Tauschzählers*?

F2: Weshalb ist die Best-Case-Laufzeit $\Omega(n)$ und nicht $\Omega(1)$?

F3: Welches Array kann schneller mit dem Bubble Sort Algorithmus sortiert werden? A = [3, 5, 2, 4, 1] oder B = [2, 4, 3, 1, 5]?

A3: Array B kann schneller sortiert werden.

Nur 4 Tauschoperationen notwendig:

Tausche 4 und 3.

Tausche 4 und 1.

Tausche 3 und 1.

Tausche 2 und 1.

Dazu insgesamt 4 Durchläufe erforderlich.

Array A: definitiv mehr Tauschoperationen nötig. Schon für die „1“ brauchen wir 4, weil die von ganz hinten nach ganz vorne muss. Weitere kommen dazu.

F4: Warum ist die Komplexität von Bubble Sort $O(n^2)$? Es sind doch nur 4 Durchläufe und 4 Tauschoperationen nötig bei 5 Elementen. Sollte die Komplexität dann nicht $O(n)$ sein?

A4: Die Komplexität wird durch die Operationen bestimmt, die bei steigendem n am schnellsten wachsen – und das sind hier die *Vergleiche*. Dabei ist es egal, ob der Rechner die Vergleiche schneller oder langsamer ausführt als die Vertauschungen.

1. Durchlauf: $n - 1$ Vergleiche

2. Durchlauf: $n - 2$ Vergleiche usw.

Gesamtanzahl der Vergleiche:
$(n - 1) + (n - 2) + \ldots + 1 = n\,(n - 1)/2$

Ignorieren der Konstanten ergibt $O(n^2)$.

Der Algorithmus vergleicht benachbarte Elemente und tauscht sie, wenn sie in falscher Reihenfolge sind.

Bubble Sort ist einfach zu implementieren, aber für große Datenmengen ineffizient.

Die Zeitkomplexität ist $O(n^2)$ wegen der notwendigen Vergleiche, nicht wegen der Tauschoperationen.

Der Name "Bubble Sort" kommt daher, dass große Elemente wie Blasen nach oben (bzw. rechts) steigen.

Best-Case Laufzeit ist $\Omega(n)$ bei bereits sortiertem Array.

In der Praxis wird Bubble Sort selten eingesetzt, da es effizientere Alternativen gibt.