Selection Sort

>> Beim **Selection Sort** (auf Deutsch: „Auswahlsortierung“) besteht die Idee des Algorithmus darin, das kleinste unsortierte Element zu finden und nach vorne zu bringen, also an den Anfang des noch unsortierten Bereichs.

Pseudocode

Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

    Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.

        Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel

Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
  - Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel

Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel

Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

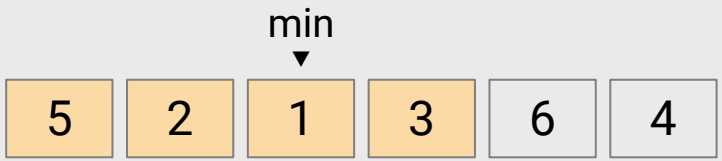
- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel

Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

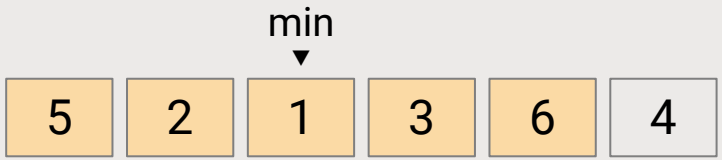
Beispiel



Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

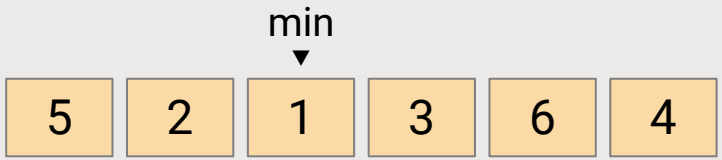
Beispiel



Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
  - Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel



Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel

Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
  - Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel

Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

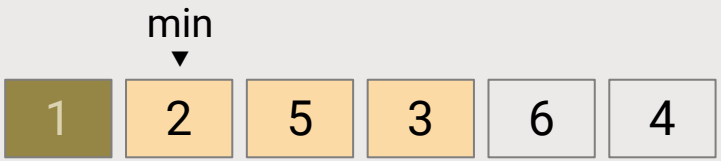
- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel

Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

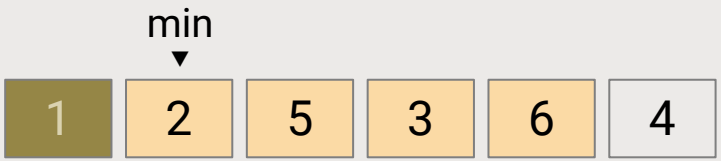
Beispiel



Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

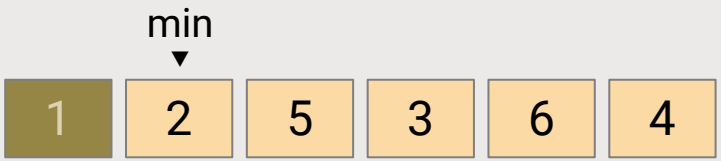
Beispiel



Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel



Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

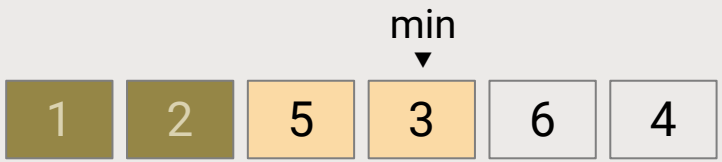
- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel

Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
  - Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

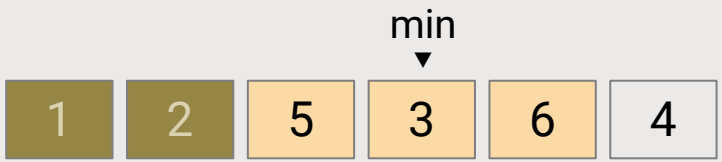
Beispiel



Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

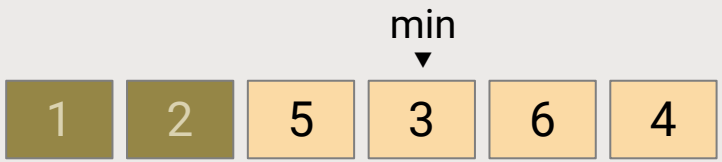
Beispiel



Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel



Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel

Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel

Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel

Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel

Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

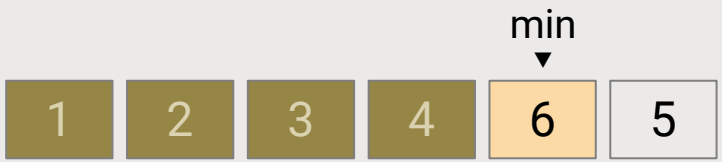
- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel

Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
  - Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

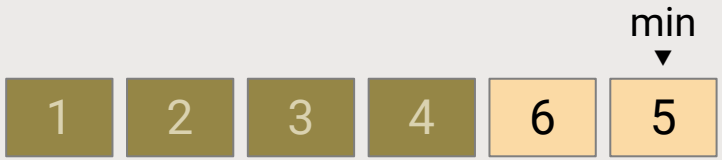
Beispiel



Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel



Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
  - Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel

Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
- Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Beispiel

Wiederhole, bis keine unsortierten Elemente mehr vorhanden sind:

- Durchsuche den unsortierten Teil der Daten, um den kleinsten Wert zu finden.
  - Vertausche den kleinsten gefundenen Wert mit dem ersten Element des unsortierten Teils.

Fälle

Worst Case:
Im schlimmsten Fall müssen wir bei jedem Durchgang alle *n* Elemente des Arrays durchsuchen, um das kleinste Element zu finden. Diesen Prozess wiederholen wir *n*-mal, da wir in jedem Durchgang nur ein Element sortieren.

Best Case:
Das Array ist bereits perfekt sortiert. Auch dann müssen wir bei jedem Durchgang alle *n* Elemente des Arrays durchsuchen, um das kleinste Element zu finden. Diesen Prozess wiederholen wir *n*-mal, da wir in jedem Durchgang nur ein Element sortieren.

Laufzeit-komplexität

$O(n^2)$

$\Omega(n^2)$

EXTRAS IN 3 MINUTEN

FRAGEN – ANTWORTEN – RÄTSEL
UND KURZE ZUSAMMENFASSUNG

F1: Was wird im Selection Sort „selected“?

F2: Wie programmiert man denn die Suche nach dem kleinsten Wert in einem unsortierten Array?

```
int find_index_of_smallest(int arr[], int length)
{
    int indexOfSmallest = 0;
    for (int i = 1; i < length; i++)
    {
        if (arr[i] < arr[indexOfSmallest])
        {
            indexOfSmallest = i;
        }
    }
    return indexOfSmallest;
}
```

Grundprinzip: Finde das kleinste Element im unsortierten Bereich und tausche es mit dem ersten Element des unsortierten Bereichs.

Vorteil: Einfach zu implementieren und zu verstehen; vorhersagbare Laufzeit.

Der sortierte Bereich wächst von links nach rechts, ein Element pro Durchgang.

Nachteil: Ineffizient bei großen Datenmengen aufgrund der quadratischen Laufzeit.

Laufzeitkomplexität: $O(n^2)$ im schlechtesten Fall, $\Omega(n^2)$ im besten Fall.

Praxistipp: Gut geeignet für kleine Arrays und Situationen, in denen Einfachheit wichtiger ist als Geschwindigkeit.