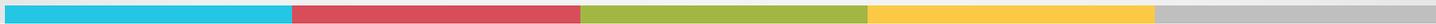


Suchen und Sortieren



Suchen: Visuelle Darstellung

The screenshot shows a web browser window with the URL <https://www.cs.usfca.edu/~galles/visualization/Search.html>. The page title is "Searching Sorted List".

At the top, there is a search input field containing "558". Below it are two tabs: "Linear Search" and "Binary Search". The "Binary Search" tab is selected. There are two radio buttons: "Small" (selected) and "Large".

The main content area displays the following information:

- Searching For: 558
- Result: 13
- Element found

Below this, there are three boxes representing the search range: "low" (12), "mid" (13), and "high" (14). The "mid" box is highlighted in green.

The sorted list of numbers is displayed in two rows:

7	135	193	198	205	280	292	304	316	432	447	526	549	558	576	587
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The numbers 12, 13, and 14 in the second row are circled in blue, green, and orange respectively.

Below the first row, there is another row of numbers:

659	726	727	771	772	774	774	787	803	862	887	891	927	928	932	962
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

<https://www.cs.usfca.edu/~galles/visualization/Search.html>

Big O: Vergleich von Algorithmen

Wann ist Algorithmus A besser als Algorithmus B?

- Algorithmus A verbraucht weniger Speicherplatz
Algorithmus B ist schneller
- Oft verbrauchen schneller Algorithmen mehr Hauptspeicher!
- Folgerung: Optimierung der Laufzeit!

Wie schnell ist ein Algorithmus?

- Best Case
- Worst Case
- Average Case

Big O: Lineares Suchen

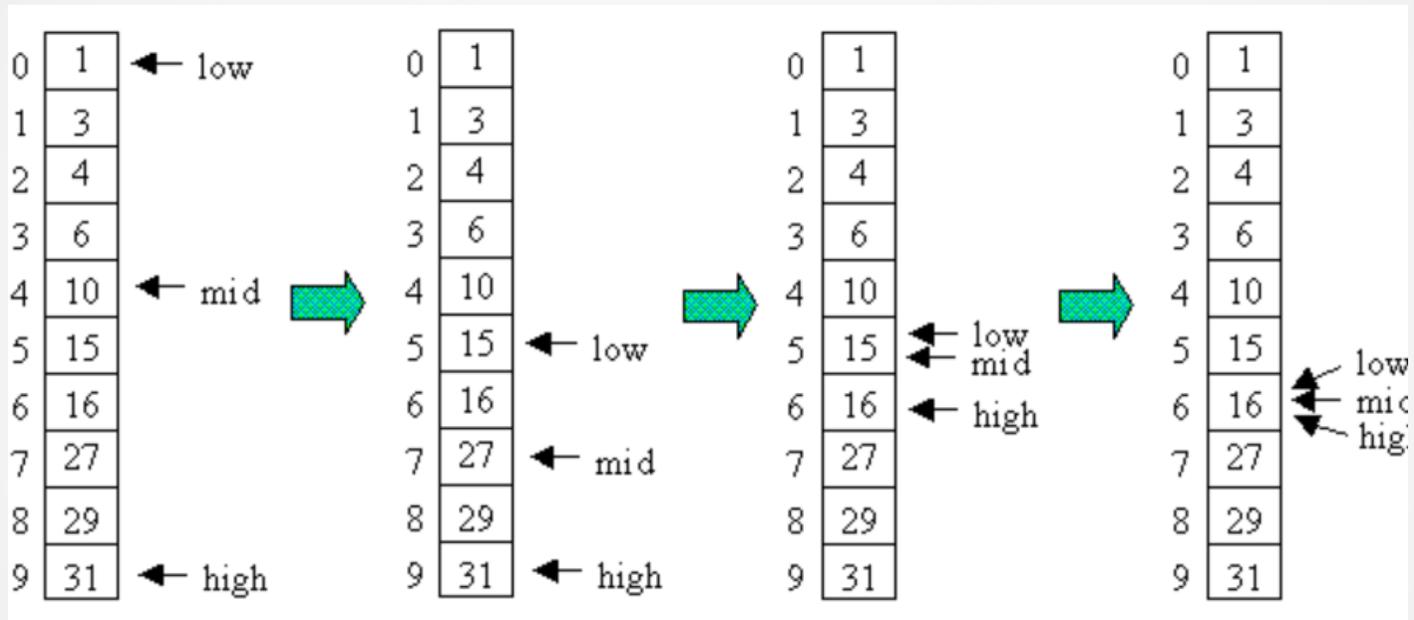
```
int linearSearch(int A[], int element) {  
    for (i=0; i<A.length; i++) {  
        if (A[i] == elem)  
            return i;  
    }  
    return false;  
}
```

Wann ist Algorithmus A besser als Algorithmus B?

- Algorithmus A verbraucht weniger Speicherplatz
Algorithmus B ist schneller
- Oft verbrauchen schneller Algorithmen mehr Hauptspeicher!
- Folgerung: Optimierung der Laufzeit!

Suchen: Binäres Suchen

Suche nach 16:



Suchen: **Binäres** vs sequentielles Suchen

Gegeben: 1000 Schachteln mit sortierten Zahlen. Es wird eine bestimmte Zahl gesucht z. B. 345

Lineare Suche

Im Mittel $(1+1000)/2$ Suchschritte

Im Mittel
 $=(1+n)/2$ Suchschritte

Binäre Suche

Im Mittel $\log_2 1000$ Suchschritte

Im Mittel $\log_2 n$ Suchschritte

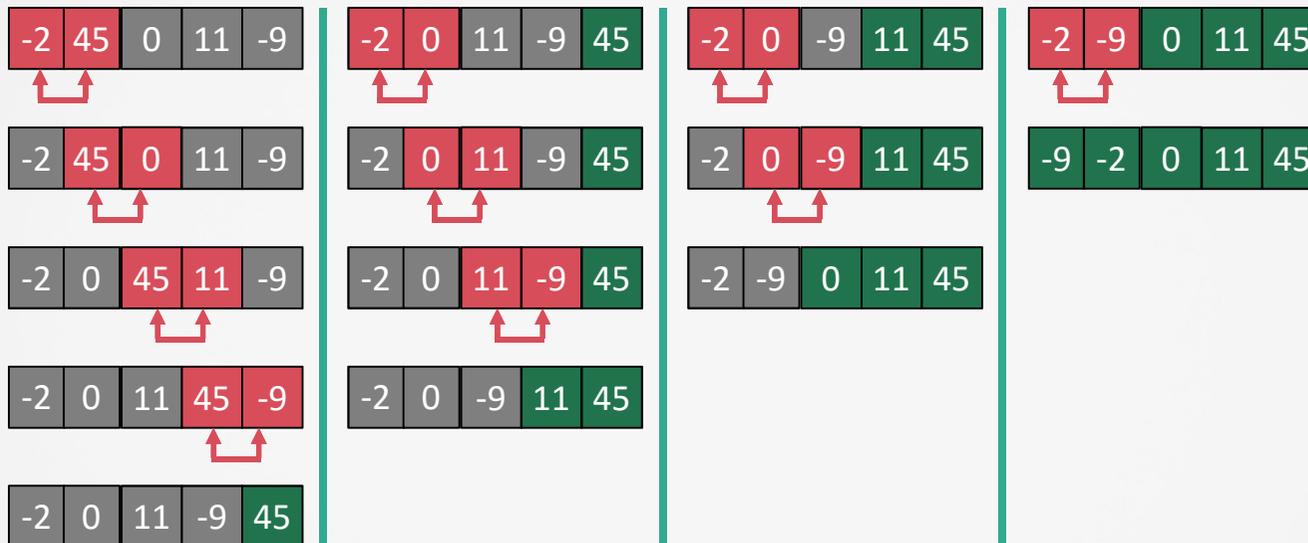


Sortieren: Bubble Sort I

Sortieren durch einfaches Vertauschen:

- Vergleiche das erste und zweite Element: Tausche es falls nötig!
- Vergleiche das zweite und dritte Element: Tausche es falls nötig!
- Vergleiche das dritte und vierte Element: Tausche es falls nötig!
- Mache die gleichen Schritte ab dem 2. Element!

Sortieren: Bubble Sort II



1.

2.

3.

4.

Sortieren: Bubble Sort III

Sortieren durch einfaches Vertauschen:

```
for(step=0;step<n-1;++step)
  for(i=0;i<n-step-1;++i) {
    if (data[i]>data[i+1]) {
      temp=data[i];
      data[i]=data[i+1];
      data[i+1]=temp;
    }
  }
```

Durchlauf durch die
Arrayelemente

Vergleich zweier
Elemente

Vertauschen zweier
Elemente

Sortieren: Insertion Sort I

Sortieren durch Einfügen:

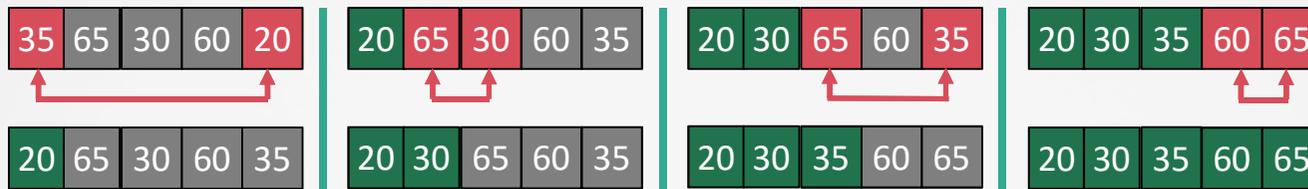
- Füge das zweite Element vor/hinter dem ersten an die richtige Stelle ein
- Füge das dritte Element an die richtige Stelle ein von den ersten zwei richtig sortierten Elementen ein
- Füge das vierte Element an die richtige Stelle ein von den ersten drei richtig sortierten Elementen ein

Sortieren: Insertion Sort II



Ergebnis: 

Sortieren: Selection Sort



1.

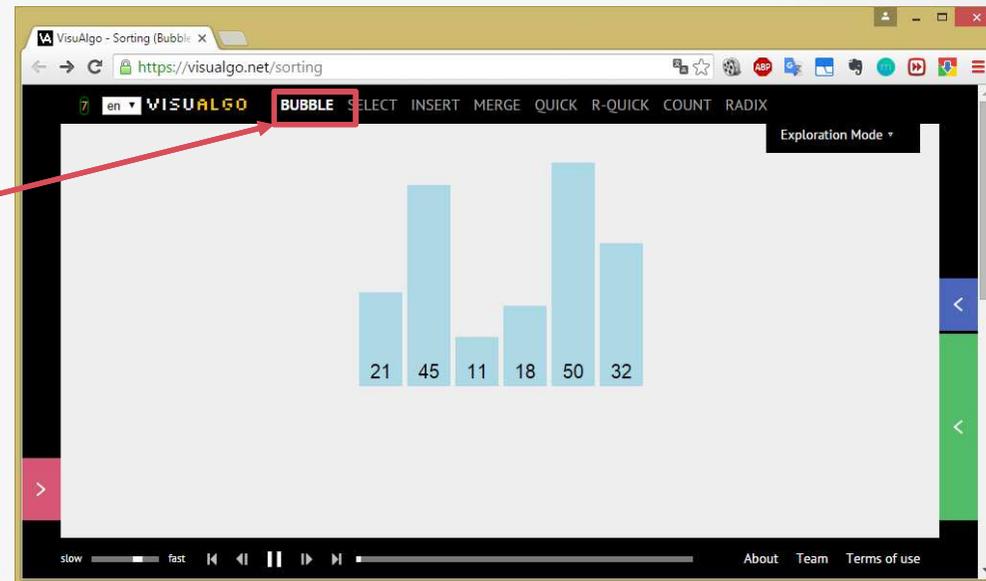
2.

3.

4.

Sortieren: Visuelle Darstellung I

1. Sortieralgorithmus wählen



<https://visualgo.net/sorting>

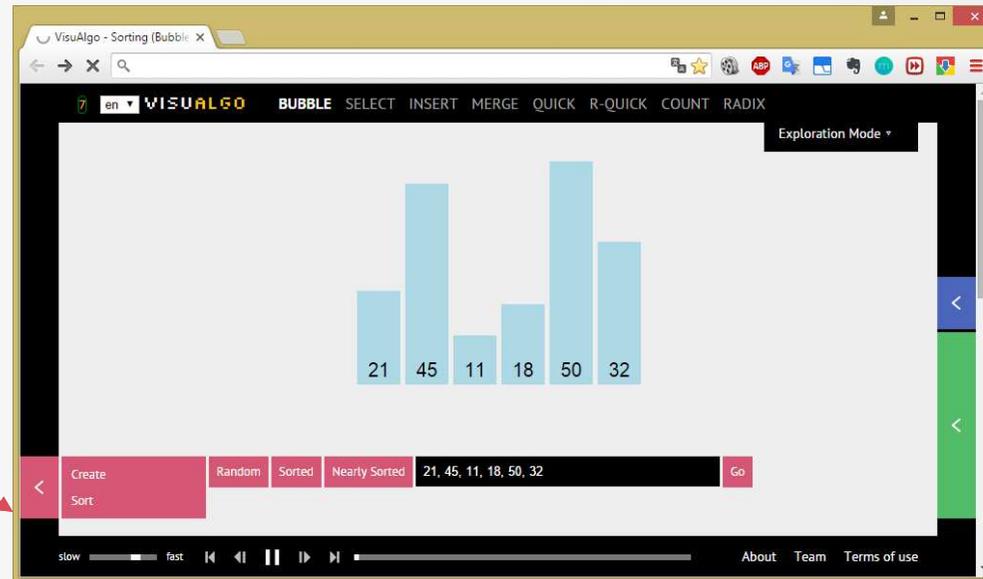
Sortieren: Visuelle Darstellung II

2. Daten eingeben



Sortieren: Visuelle Darstellung III

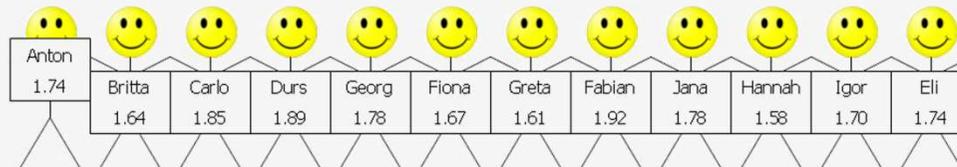
3. Sortieren



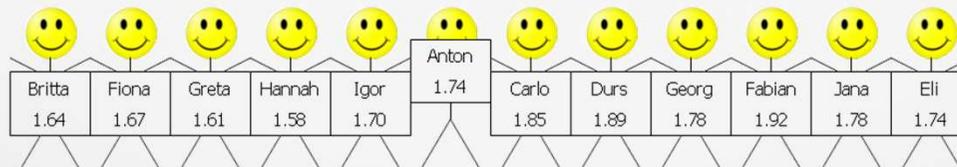
Sortieren: Quick Sort – Prinzip I

Anton, Britta, Carlo, ... wollen sich der Größe nach in einer Reihe aufstellen.
Zuerst werden alle genau vermessen.

Eine Person (z.B. die erste in der Reihe) wird als Vergleichsperson ausgewählt. Im vorliegenden Beispiel ist das Anton.

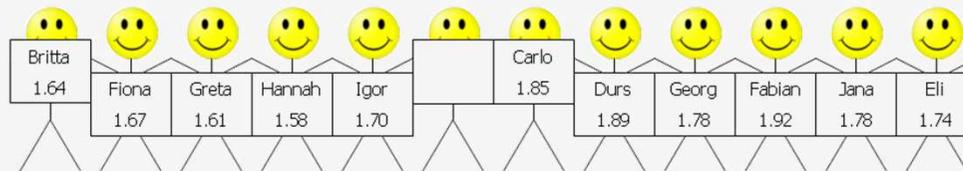


Alle anderen ordnen sich links bzw. rechts von der Vergleichsperson ein, je nachdem, ob sie kleiner oder größer gleich als die Vergleichsperson sind.



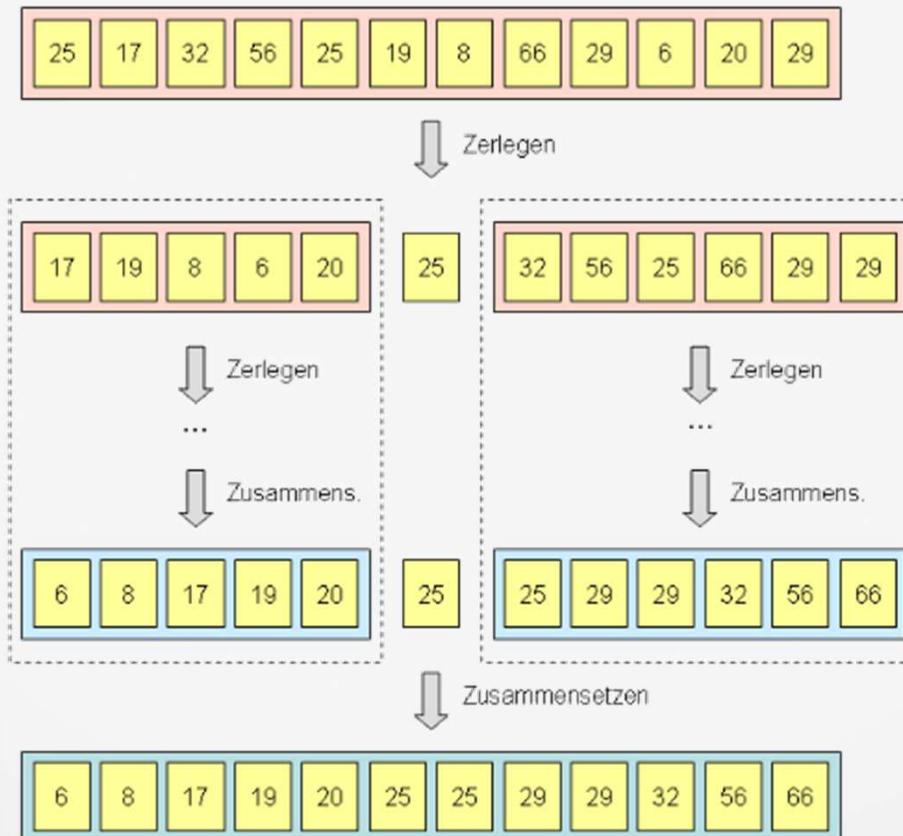
Sortieren: Quick Sort – Prinzip II

Anton bleibt jetzt auf seiner Position. Er nimmt nicht mehr an der Sortierung teil. Dasselbe Verfahren wird jetzt im linken und im rechten Bereich durchgeführt: Eine Person (z.B. die erste in der Reihe) wird wiederum als Vergleichsperson ausgewählt.



Und so weiter ...

Sortieren: Quick Sort – Beispiel



Sortieren: Quick Sort

Ebenso gut ist:

<http://www.w3resource.com/javascript-exercises/searching-and-sorting-algorithm/index.php>

Informationen zu Groß Oh:

<https://www.cs.usfca.edu/~galles/cs245S08/lecture/lecture11.pdf>

Graphische Darstellung von Quicksort:

<http://me.dt.in.th/page/Quicksort/>