

In []:



Quicksort

1 Grundidee

Quicksort ist ein **Divide-and-Conquer-Algorithmus**, der ein **Pivot-Element** wählt und die Liste so partitioniert, dass:

- alle kleineren Elemente links vom Pivot liegen
- alle grösseren Elemente rechts vom Pivot liegen

Die Teillisten werden anschliessend rekursiv sortiert.

2 Voraussetzungen

- ✗ keine
 - funktioniert auf **beliebigen vergleichbaren Elementen**
-

3 Laufzeiten & Eigenschaften

Eigenschaft	Wert
Best Case	$O(n \log n)$
Average Case	$O(n \log n)$
Worst Case	$O(n^2)$
Speicherbedarf	$O(\log n)$ (rekursiv)
In-place	ja
Stabil	nein

Hinweis: Der Worst Case tritt auf bei **schlechter Pivot-Wahl** (z. B. bereits sortierte Liste).



4 Schritt-für-Schritt-Beispiel

Ausgangsliste:

[5, 3, 4, 1]

Pivot = 3

```
links: [1]
Pivot: [3]
rechts: [5, 4]
```

Rekursives Sortieren

```
[1] + [3] + [4, 5]
```

Ergebnis:

```
[1, 3, 4, 5]
```

5 Besonderheiten / Prüfungsrelevante Hinweise

- Sehr schnell in der Praxis
 - Pivot-Wahl ist entscheidend
 - Oft mit **Random-Pivot** oder **Median-of-Three** optimiert
-

6 Vor- und Nachteile

Vorteile

- sehr schnell im Durchschnitt
- in-place
- cache-freundlich

Nachteile

- nicht stabil
 - Worst Case $O(n^2)$
-

🧠 Merksatz für die Prüfung

Quicksort partitioniert die Liste um ein Pivot und ist im Mittel sehr schnell, aber nicht stabil.

7 Python-Implementierung

```
In [1]: def quicksort(arr):
    if len(arr) <= 1:
        return arr

    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]

    return quicksort(left) + middle + quicksort(right)
```