

## 5.4 Quicksort

- Benutzt wie Mergesort eine divide-and-conquer-Strategie.
- Rekursion
- Komplexität / Laufzeit
  - im Worst-Case:  $O(n^2)$
  - im Mittel (Average-Case):  $O(n \log n)$
- Überblick
  - 5.4.1 Beschreibung
  - 5.4.2 Worst-Case
  - 5.4.3 Best-Case
  - 5.4.4 Average-Case

# 5.4.1 Beschreibung

## Beispiel

A:	0	1	2	3	4	5	6	7	8	9	10	↔ Index
	✓	0	6	4	7	2	1	3	9	8	5	↔ Inhalt
	↑	↑								↑		
	P <sub>1</sub>	P <sub>2</sub>								Referenzelement		

Idee: Sortiere die „kleinen“ Elemente nach vorne.

↑  
Im Bezug auf das Referenzelement.

P<sub>2</sub> läuft von A[1] nach A[n]

P<sub>1</sub> merkt sich das Ende der kleinen Elemente

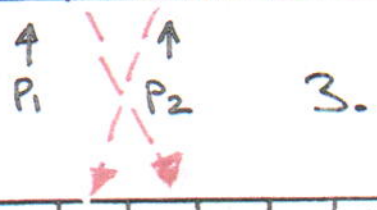
1. 0 ist klein. → bleibt vorne.  $P_2++$ .  $P_1++$ .

2. 6 ist groß. → gehe weiter  $P_2++$ .

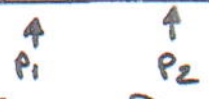
3. 4 ist klein. → tausche an das Ende der kleinen Elemente.  
 $A[P_1+1] \leftrightarrow A[P_2]$

gehe weiter.  $P_2++$ .  $P_1++$ .

0	1	2	3	4	5	6	7	8	9	10
/	0	6	4	7	2	1	3	9	8	5



0	1	2	3	4	5	6	7	8	9	10
/	0	4	6	7	2	1	3	9	8	5



4. 7 ist groß. → gehe weiter.  $P_2++$

5. 2 ist klein → Tausch:  
 $A[P_1+1] \leftrightarrow A[P_2]$

gehe weiter.  $P_2++$ .  $P_1++$ .



In Pseudocode:

Quicksort( $A, p, r$ )

- 1 IF  $p < r$
- 2      $q \leftarrow \text{Partition}(A, p, r)$
- 3     Quicksort( $A, p, q-1$ )
- 4     Quicksort( $A, q+1, r$ )

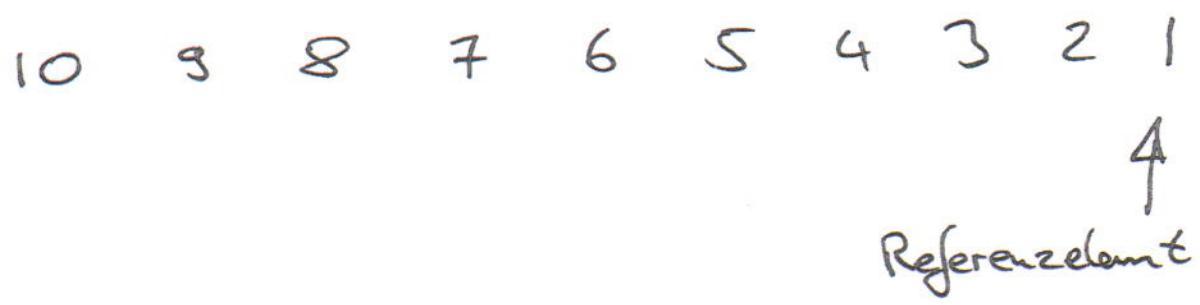
Partition( $A, p, r$ )

- 1  $x \leftarrow A[r]$      "Referenzelement"
- 2  $i \leftarrow p-1$
- 3 FOR  $j \leftarrow p$  TO  $r-1$  DO
- 4     IF  $A[j] \leq x$
- 5          $i \leftarrow i+1$
- 6         tausche  $A[i] \leftrightarrow A[j]$
- 7     tausche  $A[i+1] \leftrightarrow A[r]$
- 8 return  $i+1$ .

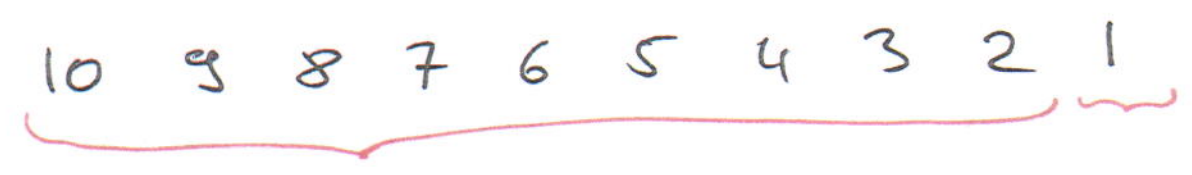


# 5.2 Worst Case

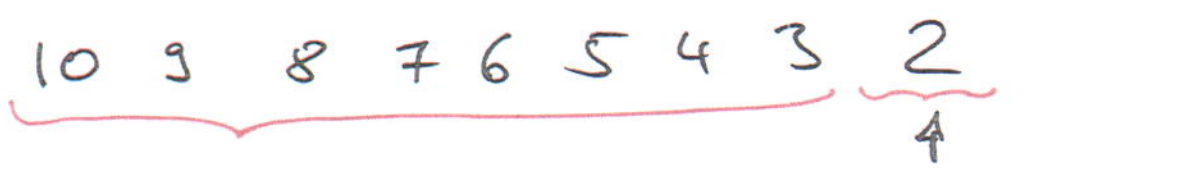
Ein Beispiel:



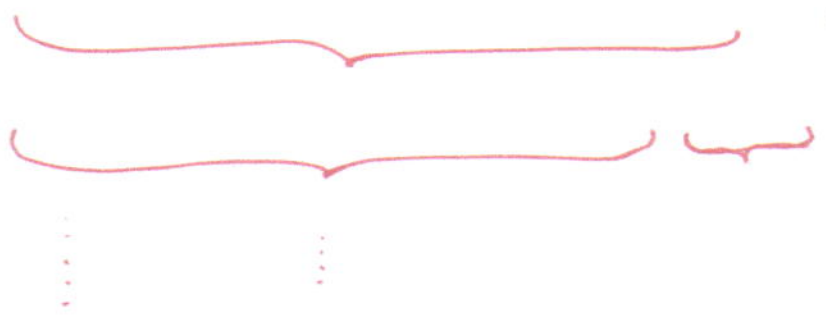
Am Ende von Partition:



Quicksort



Referenzelement





### 5.4.3 Best Case

- Im günstigsten Fall ist das Referenzelement der Median, d.h. es teilt die Elemente <sup>in</sup> zwei Mengen, die sich in der Größe um höchstens 1 unterscheiden.

↳ Bisektion

- Damit erhalten für folgende Rekursionsgleichung für die Anzahl  $V(n)$  der Vergleiche:

$$V(n) = (n-1) + 2 \cdot V\left(\frac{n}{2}\right)$$

mit  $V(1) = 1$ .

- Lösung:  $V(n) = n \cdot \log n + O(n)$   
und damit  $V(n) \in O(n \log n)$   
(im besten Fall).



## 5.4.4 Average Case

(131)

- Durchschnittliche Laufzeit.

↳ worüber?

- Zahlenfolge:  $a_1, a_2, \dots, a_n$

↳ Wie viele Permutation dieser Zahlen gibt es?

$$\underbrace{n \cdot (n-1) \cdot (n-2) \cdot \dots \cdot 1}_{= n!}$$

- Jede Permutation kann mit gleicher Wahrscheinlichkeit vorkommen:  $\frac{1}{n!}$

• Zurück zu Quicksort:

- Eingabe:  $a_1, a_2, \dots, a_n$

- Ausgabe:  $s_1, s_2, \dots, s_n$  sortierte Zahlenfolge

### Satz 5.9

Die Average-Case-Laufzeit von Quicksort ist  $O(n \log n)$ .

### Beweis:

Beobachtung: Zwei Elemente  $s_i$  und  $s_j$  werden höchstens einmal miteinander verglichen.

(Genau dann, wenn  $s_i$  oder  $s_j$  Referenzelement ist)

Zufallsvariable

$$X_{ij} = \begin{cases} 1 & , s_i \text{ und } s_j \text{ werden verglichen} \\ 0 & , \text{sonst} \end{cases}$$

$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^n X_{ij}$  ist die  
gesamte Anzahl an Vergleichen.

Damit:

$$E[X] = E\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^n X_{ij}\right]$$

$$(*) = \sum_{i=1}^{n-1} \sum_{j=i+1}^n \underbrace{E[X_{ij}]}$$

$$= 1 \cdot P(X_{ij}=1) + \underbrace{0 \cdot P(X_{ij}=0)}_{=0}$$

Wir müssen also <sup>die</sup> Wahrscheinlichkeit  
bestimmen, dass  $s_i$  und  $s_j$  verglichen  
werden.