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_1 ↑ P_2 ↑ Referenzelement

Idee: Sortiere die „kleinen" Elemente nach vorne.

↑ Im Bezug auf das Referenzelement.

P_2 läuft von $A[1]$ nach $A[n]$

P_1 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

↑ p_1 ↑ p_2 3.

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

↑ p_1 ↑ p_2

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++.

⋮

Am Ende: $p_2 = 1$, $p_1 = 5$

0	1	2	3	4	5	6	7	8	9	10
/	0	4	2	1	3	7	6	9	8	5
					↑ p_1					↑ p_2

Tausche: $A[p_1 + 1] \Longleftrightarrow A[p_2]$

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

Sortiere mit Quicksort

Sortiere mit Quicksort

In Pseudocode:

```
Quicksort(A, p, r)
    IF p < r
        q ← Partition(A, p, r)
        Quicksort(A, p, q-1)
        Quicksort(A, q+1, r)

Partition(A, p, r)
    x ← A[r]                          „Referenzelement"
    i ← p-1
    FOR j ← p TO r-1 DO
        IF A[j] ≤ x
            i ← i+1
            tausche A[i] <=> A[j]
    tausche A[i+1] <=> A[r]
    return i+1.
```

5.4.2 Worst Case

Ein Beispiel:

10 9 8 7 6 5 4 3 2 1

Referenzelement

Am Ende von Partition:

$$\underbrace{10\ \ 9\ \ 8\ \ 7\ \ 6\ \ 5\ \ 4\ \ 3\ \ 2}_{\text{Quicksort}}\ \ \underbrace{1}$$

$$\downarrow$$

$$\underbrace{10\ \ 9\ \ 8\ \ 7\ \ 6\ \ 5\ \ 4\ \ 3}\ \ \underbrace{2}$$

Referenzelement

⋮ ⋮

- Im schlechtesten Fall ist das Referenzelement jedesmal die kleinste (oder größte) Zahl.
- Dann ist die Aufteilung: $\underbrace{1 \ldots n-1}\ \underbrace{n}$

Satz: 5.7

Die Worst-Case-Komplexität von Quicksort ist $O(n^2)$.

Beweis:

Partition benötigt für einen k-elementigen Array k-1 Vergleiche.

n	Elemente:	$n-1$
$n-1$	"	$n-2$
$\vdots$		
1		0

$$\sum_{i=0}^{n-1} i = \frac{(n-1)\cdot n}{2} \in O(n^2)$$

$\square$

5.4.3 Best Case

- Im günstigsten Fall ist das Referenzelement der Median, d.h. es teilt die Elemente in 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 $V(n) \in \Theta(n \log n)$ (im besten Fall).

5.4.4 Average Case

- Durchschnittliche Laufzeit.

  ↳ worüber?

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

  ↳ Wie viele Permutation dieser Zahlen gibt es?

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

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

- Zurück zu Quicksort:
  - Eingabe: $a_1\ a_2 \ldots a_n$
  - Ausgabe: $s_1\ s_2 \ldots s_n$ 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 die Wahrscheinlichkeit bestimmen, dass s_i und s_j verglichen werden.