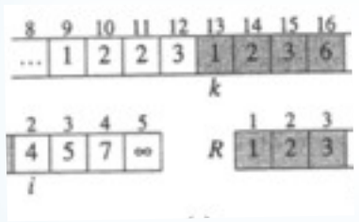


Kapitel 5.5: Quicksort

Algorithmen und Datenstrukturen
WS 2012/13

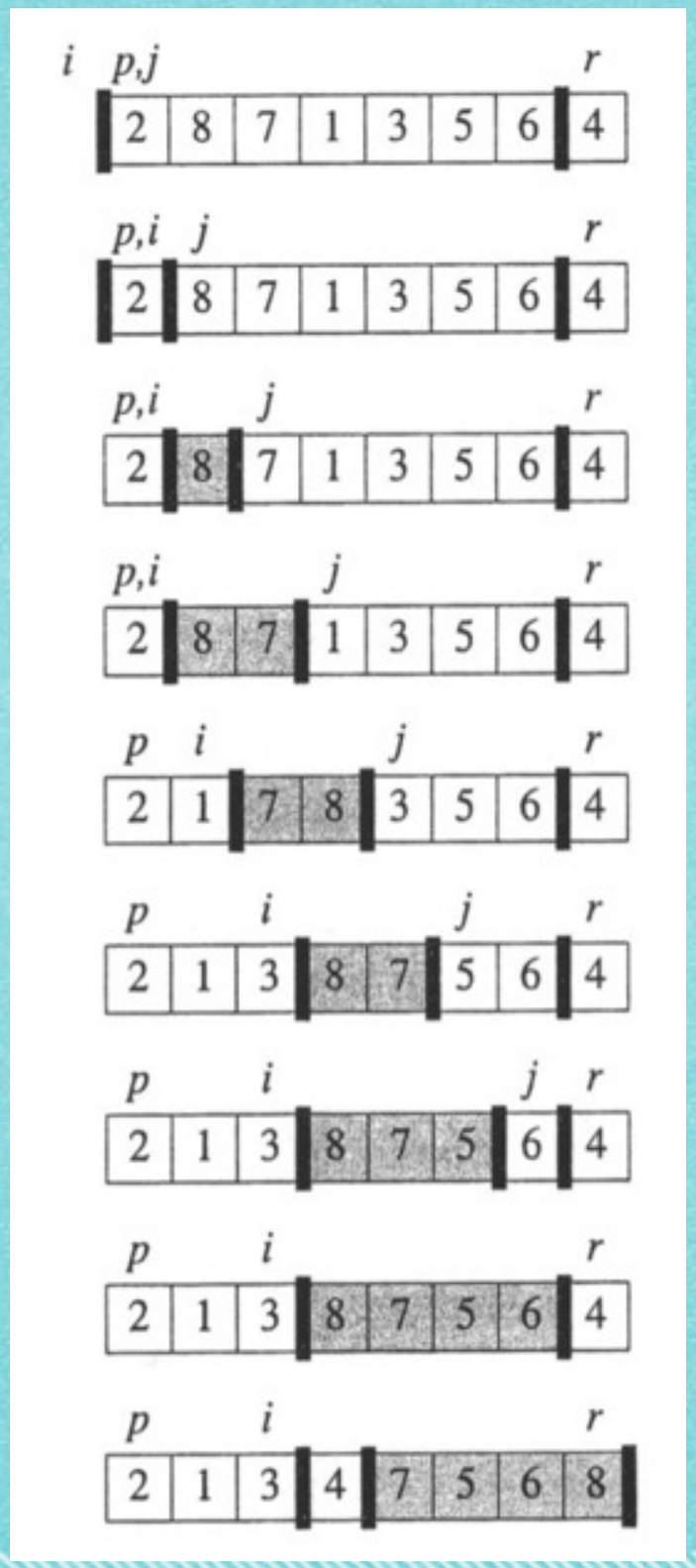
Prof. Dr. Sándor Fekete

5.5 Quicksort

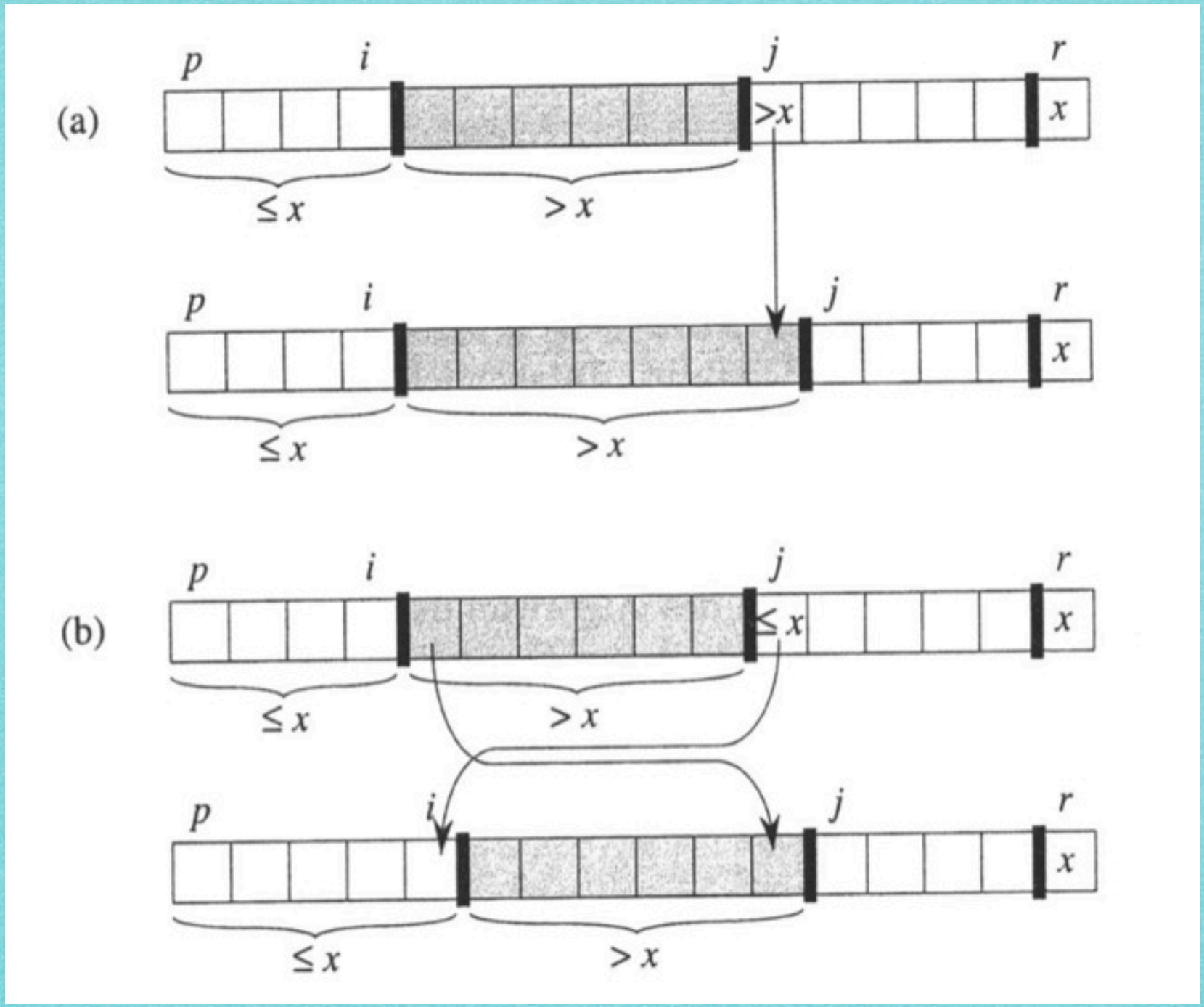
<u>Grundideen:</u>

- **Divide and Conquer**
- **Jeweils Aufteilung in zwei Teilarrays**
- **Rekursiv: Sortieren der Teilarrays**
- ***Kein* Merge-Schritt!**
- **Stattdessen Aufteilung der Teilarrays anhand eines “Pivot”-Elements, das die Menge in kleinere und größere Elemente teilt.**
- **Balance der Aufteilung vorher nicht absehbar.**

5.5.1 Ablauf Quicksort



5.5.1 Ablauf Quicksort



5.5.2 Algorithmische Beschreibung

Algorithmus 5.1 1

<u>INPUT:</u> Subarray von A=[1,...,n],
der bei Index p beginnt und bei Index r endet, d.h. A[p,...,r]

<u>OUTPUT:</u> Sortierter Subarray

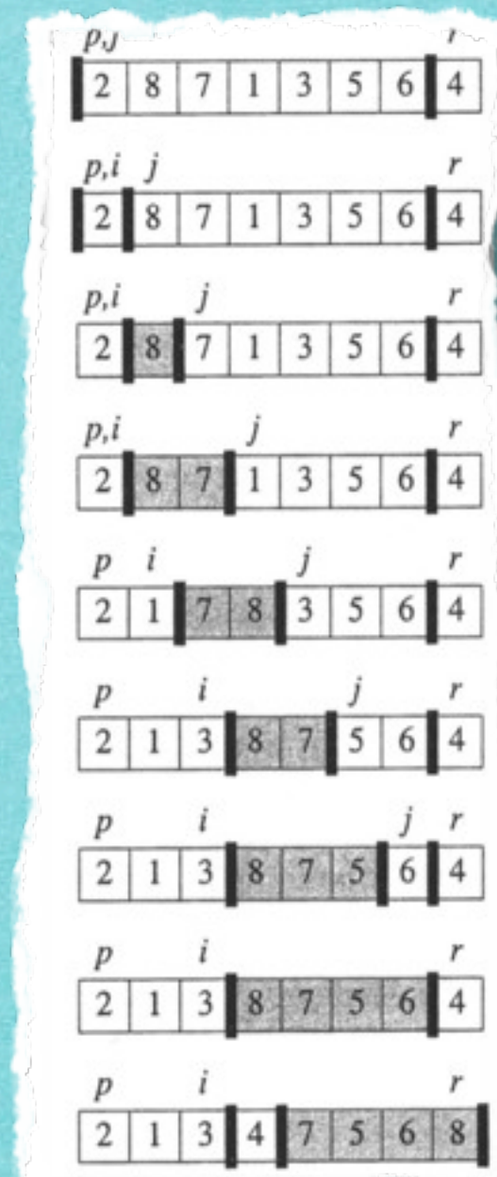
```
QUICKSORT(A, p, r)
if p < r
    then q ← PARTITION(A, p, r)
         QUICKSORT(A, p, q − 1)
         QUICKSORT(A, q + 1, r)
```

Subroutine 5.1.2

<u>INPUT:</u> Subarray von A=[1,...,n], d.h. A[p,...,r]

<u>OUTPUT:</u> Zwei Subarrays A[p,...,q-1] und A[q+1,...,r]
mit A[i]≤A[q] und A[q]<A[j] für i=p,...,q-1 und j=q+1,...,r

```
PARTITION(A, p, r)
x ← A[r]
i ← p − 1
for j ← p to r − 1
        do if A[j] ≤ x
                then i ← i + 1
                        vertausche A[i] ↔ A[j]
vertausche A[i + 1] ↔ A[r]
return i + 1
```



5.5.3 Laufzeit von Quicksort

Wie viele Schritte benötigt Quicksort für einen Array der Länge n?

<u>Unterscheidung:</u>

- **Worst Case**
- **Best Case**
- **Average Case**

5.5.3 Quicksort: Worst Case

Schlechtester Fall: Pivot liegt extremal, d.h. nach PARTITION ist ein Teilarray n-1 lang, der andere hat Länge 0.

$$\begin{aligned} T(n) &= T(n-1) + T(0) + \Theta(n) \\ &= T(n-1) + \Theta(n) \ . \end{aligned}$$

Man sieht (und beweist leicht):

$$\sum_{i=0}^{n} \Theta(n-i) = \Theta(n^2)$$

5.5.3 Quicksort: Best Case

Bester Fall: Pivot liegt genau in der Mitte, d.h. nach PARTITION haben beide Teilarrays i.W. die Länge n/2.

$$T(n) \leq 2T(n/2) + \Theta(n)$$

Man sieht, z.B. mit dem Mastertheorem:

$$\Theta(n \log n)$$

5.5.3 Quicksort: Best Case

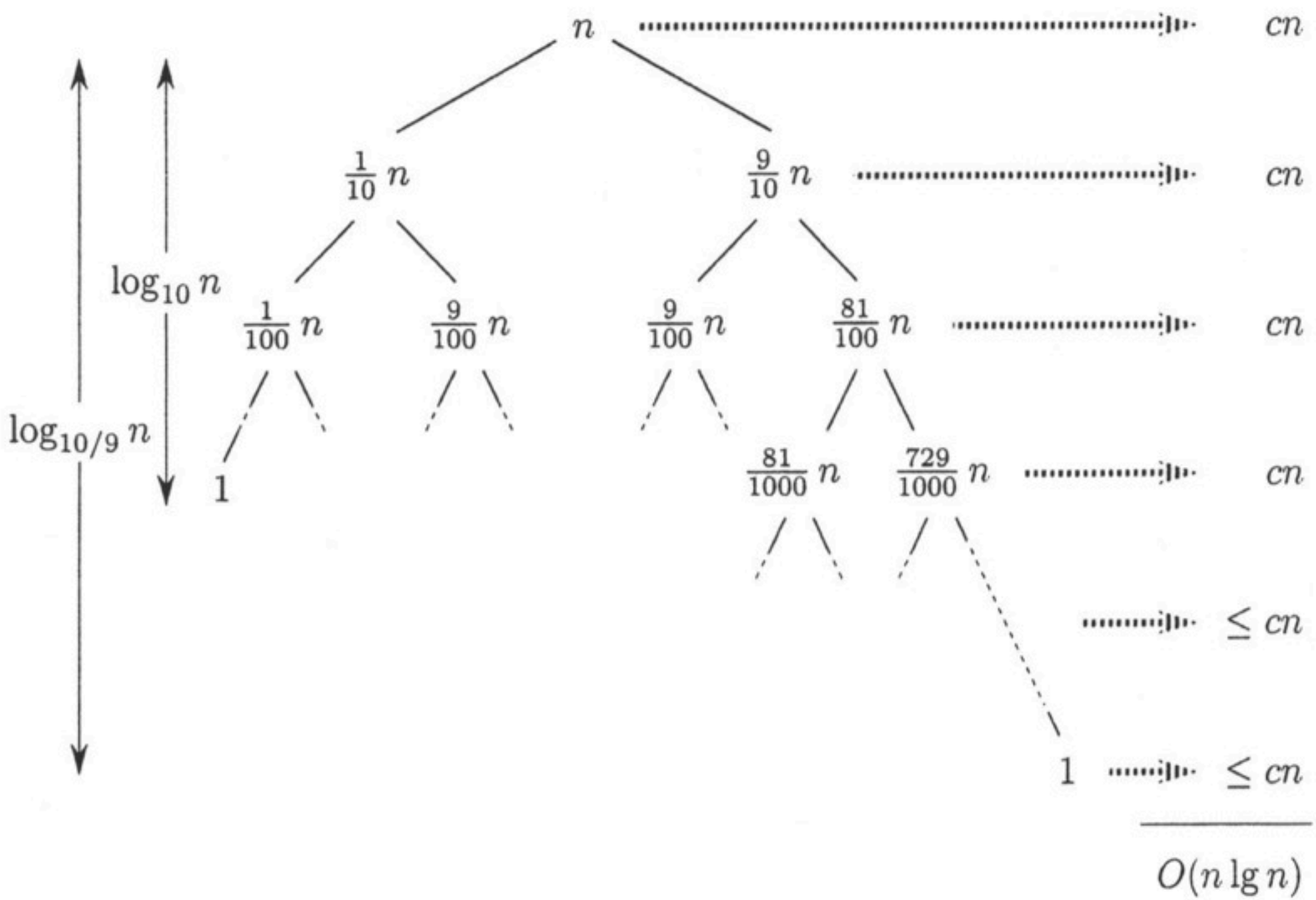
Das funktioniert auch noch, wenn PARTITION ein *annähernd* balanciertes Ergebnis liefert:

$$T(n) \leq T(9n/10) + T(n/10) + cn$$

Man sieht wieder mit dem Mastertheorem:

$$\Theta(n \log n)$$

5.5.3 Quicksort: Best Case



5.5.3 Quicksort: Average Case

<u>Satz 5.13 (Durchschnittliche Laufzeit von Quicksort)</u>
Für einen n-elementigen Array A hat Quicksort eine erwartete Laufzeit von O(n log n).

5.5.3 Quicksort: Average Case

Warum interessiert das?

*** 1987**

Sebastian Wild

*** 1981**

Zlatan Ibrahimovic

5.5.3 Quicksort: Average Case

Warum interessiert das?

*** 1987**

Sebastian Wild

	Comparisons
Classic Quicksort (Algorithm 1)	$2(n+1)\mathcal{H}_{n+1} - \frac{8}{3}(n+1)$ $\approx 2n\ln n - 1.51n + \mathcal{O}(\ln n)$
Sedgewick (Algorithm 2)	$\frac{32}{15}(n+1)\mathcal{H}_{n+1} - \frac{856}{225}(n+1) + \frac{3}{2}$ $\approx 2.13n\ln n - 2.57n + \mathcal{O}(\ln n)$
Yaroslavskiy (Algorithm 3)	$\frac{19}{10}(n+1)\mathcal{H}_{n+1} - \frac{711}{200}(n+1) + \frac{3}{2}$ $\approx 1.9n\ln n - 2.46n + \mathcal{O}(\ln n)$

“Best Paper Award”

Mehr an der Tafel!

s.fekete@tu-bs.de