Sortieralgorithmen (Wiederholung)

37, 12, 16, 22, 11, 33, 32, 23

12 37

11, 22

16 37

Bubblesort, $O(n^2)$

Sortieren durch Auswahl (selectionsort)

37, ~~12~~, ~~16~~, 22, ~~11~~, 33, 32, 23

11, 12, 16, ... $O(n^2)$

Sortieren durch Einfügen (insertion sort)

37, 12, 16, 22, 11, 33, 32, 23

37

12 37

12 16 37

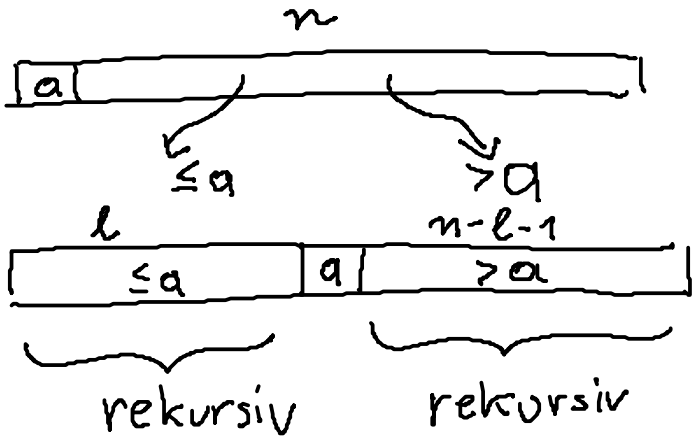
12 16 22 37

11 12 16 22 37

11 12 16 22 33 37

$O(n^2)$

Quicksort



Laufzeit

$T(n) = O(n) + T(l) + T(n-l-1)$

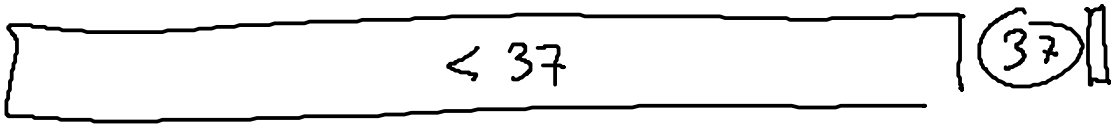
$0 \le l \le n-1$

$T(0) = T(1) = 0$

im besten Fall: a ist in der Mitte

$l \sim \frac{n}{2}$ $\quad T(n) = O(n) + 2T(\frac{n}{2}) \cdots = O(n \log n)$

37, 12, 16, 22, 11, 33, 32, 23



$l = n-1$

im schlimmsten Fall: $T(n) = O(n) + T(n-1) + T(0)$

$= O(n^2)$

Sortieren durch Verschmelzen (mergesort)

37, 12, 16, 22 , 11, 33, 32, 23

↓ rekursiv ↓ rekursiv

~~12~~, ~~16~~, ~~22~~, 37 ~~11~~, ~~23~~, ~~32~~, ~~33~~ } $O(n)$

11 12 16 22 23 32 33 37

$T(n) \le 2T(\lceil \frac{n}{2} \rceil) + O(n) = \ldots = O(n \log n)$

$T(2^k) = \ldots$

Teile und herrsche (divide-and-conquer)

	Quicksort	Merge-Sort
1.) Zerlege in Teilprobleme	Pivotelement $O(n)$	$O(1)$
2.) Löse Teilprobleme rekursiv	✓	✓
3.) Füge die Teillösungen zusammen	✗	$O(n)$

O-Notation (Wiederholung)

$$f, g : \mathbb{N} \to \mathbb{R}$$

$f(n) = O(g(n))$ $(n \to \infty)$

$\exists N_0 \; \exists C :$

$\forall n \geq N_0 : |f(n)| \leq C \cdot g(n)$

Vorteile:

- schneller oder langsamer Computer?
- Es kommt nur auf den dominanten Term an:

$$12 \underline{n \log_2 n} + \underline{4n^2} = \Theta(n^2)$$

$f(n) = \Omega(g(n))$ $(n \to \infty)$

$\exists N_0 \; \exists C > 0.$

$\forall n \geq N_0 : f(n) \geq C \cdot g(n)$

$f(n) = \Theta(g(n)) : f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))$

BEISPIEL

$T(n) = O(n \log n)$ für $n = 2^k$

$n \in \mathbb{N}$ beliebig

$n' = 2^k$ nächstgrößere Zweierpotenz $\quad \lceil \log_2 n \rceil = k$

$n' = 2^{\lceil \log n \rceil} \leq 2^{\log n + 1} = n \cdot 2$

$T(n') \leq C_0 \, n' \log_2 n'$ für $n' = 2^k$

$n \in \mathbb{N}$ beliebig

$T(n) \leq T(2^k) \leq C_0 \, 2^k \cdot \log_2 2^k$

$\leq C_0 \cdot 2n \cdot \log_2 2n$

$\log n + 1 \leq \log n$

$\leq 2 \cdot \log n$

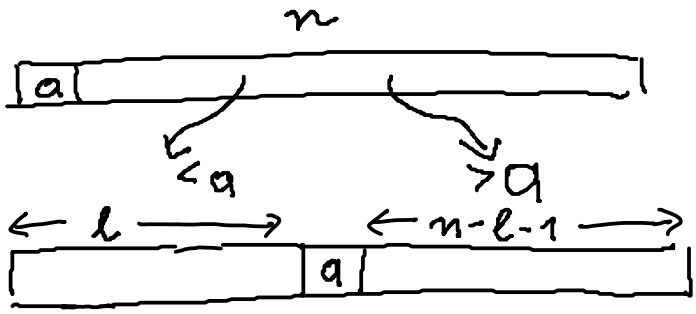
$n \geq 2 =: N_0$

$\leq C_0 \cdot 4 n \log_2 n$

$C \, n \log_2 n \quad \Rightarrow T(n) = O(n \log n)$

Quicksort mit zufälliger Pivotauswahl

- Alle Elemente verschieden



Laufzeit

$$T(n) \leq cn + T(\ell) + T(n-\ell-1) \qquad 0 \leq \ell \leq n-1$$

- Pivotelement a wird zufällig ausgewählt.

⇒ a ist mit Wahrscheinlichkeit $\frac{1}{n}$ das
kleinste, 2-t-kleinste, 3-kleinste, ... , größte

⇒ ℓ ist mit Wahrscheinlichkeit $\frac{1}{n}$ gleich $0, 1, 2, \ldots, \ell-1$

$$E[T(n)] \leq cn + \frac{1}{n} \sum_{\ell=0}^{n-1} \left(E[T(\ell)] + E[T(n-1-\ell)] \right)$$

$\leq S(n)$

$$S(n) \overset{=}{\leq} cn + \frac{1}{n} \sum_{\ell=0}^{n-1} \left(S(\ell) + S(n-1-\ell) \right)$$

$$= cn + \frac{2}{n} \sum_{\ell=0}^{n-1} S(\ell)$$

$$n\,S(n) = cn^2 + 2\left[S(0) + S(1) + \ldots + S(n-1)\right] \quad -$$

n+1 statt n eingesetzt.

$$(n+1)S(n+1) = c(n+1)^2 + 2\left[S(0) + S(1) + \ldots + S(n-1) + S(n)\right] \quad +$$

$$(n+1)\,S(n+1) - n\,S(n) = c(2n+1) + 2\,s(n)$$

$$(n+1)\,S(n+1) = (n+2)\,S(n) + c\cdot(2n+1) \qquad \Big| \times \frac{1}{(n+1)(n+2)}$$

$$\frac{S(n+1)}{n+2} = \frac{S(n)}{n+1} + C \cdot \frac{2n+1}{(n+1)(n+2)}$$

$U(n+1) =$ $\quad$ $U(n) =$ $\quad$ $S(n) = (n+1)\,U(n)$

$$U(n+1) = U(n) + C \cdot \frac{2n+1}{(n+1)(n+2)}$$

$$\leq \frac{2n+2}{(n+1)(n+2)} \cdot 2 = \frac{2}{n+2}$$

$$U(n+1) \leq \frac{2}{n+2} + U(n)$$

$$U(n) \leq \frac{2}{n+1} + U(n-1)$$

$$\leq \frac{2}{n} + U(n-2)$$

$$\cdots \leq \frac{2}{2} + U(0)$$

$$U(n+1) \leq 2\left[\frac{1}{n+2} + \frac{1}{n+1} + \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2}\right] + U(0)$$

$O(\log n)$ (harmonische Reihe)

$$E[T(n)] \leq S(n) = (n+1) \cdot O(\log n) = O(n \log n).$$

SATZ. RANDOMISIERTER ALGORITHMUS (Der Algorithmus macht Zufallsschritte)

Die erwartete Laufzeit für Quicksort mit zufälliger Pivotauswahl ist $O(n \log n)$.

SATZ. Die Eingabe ist zufällig. Analyse des durchschnittlichen Falls

Wenn die Eingabe in zufälliger Reihenfolge vorliegt (alle $n!$ Permutationen sind gleichwahrscheinlich), dann ist die mittlere Laufzeit von Quicksort $O(n \log n)$.

- Analyse im schlimmsten Fall (worst case)
  - ⊕ gilt für alle Eingaben der Länge n.
  - ⊖ zu pessimistisch
- Analyse im Durchschnitt / im Mittel (average case)
  - ⊖ benötigt eine Annahme über die Verteilung der Eingabe.
- Analyse in Abhängigkeit von Merkmalen der Daten (z.B. input-sensitive, output-sensitive, ...)

Beispiel: vorsortierte Listen

Randomisierte Algorithmen.

- Zufall passiert im Algorithmus.

IDEE: Zufall hilft.