Quick Sort

„divide and conquer“

Rupert J. Hartung

Warum Sortieralgorithmen?

- Konzepte (z.B. Datenstrukturen)
- Analyse
- „Optimalität“

	# Vergleiche wc	# Vergleiche ac	zus. Speicher
QuickSort	$\frac{1}{2}N^2 + \mathcal{O}(N)$	$1.4\, N \log N + \mathcal{O}(N)$	$\mathcal{O}(\log N)$
HeapSort	$2\, N \log N + \mathcal{O}(N)$	$\Theta(N \log N)$	$\mathcal{O}(1)$
MergeSort	$N \log N + \mathcal{O}(N)$	$\Theta(N \log N)$	$N + \mathcal{O}(1)$
Insertion Sort	$\frac{1}{2}N^2 + \mathcal{O}(N)$	$\frac{1}{4}N^2 + \mathcal{O}(N)$	$\mathcal{O}(1)$

QuickSort (A,L, R){ QuickSort (A, 1, N);

$l := L$; $r := R + 1$;

$V := A[L]$;

while ($l < r$) {

while ($l < r$ und $A[r] > V$) $r--$;

while ($l < r$ und $A[l] < V$) $l++$;

if ($l < r$) { $A[l] \leftrightarrow A[r]$ };

}

if ($A[r] < V$) $r--$;

$A[r] \leftrightarrow A[L]$;

($*$)

QuickSort (A, L, $r - 1$);

QuickSort (A, $r + 1$, R);

}

Korrektheit

Behauptung: Bei $(*)$ gilt

$$A_i \leq V = A_r \leq A_j$$

für alle

$$i < r < j$$

Analyse

(Schlüssel-)Zuweisungen $\leq$ # Vergleiche

$Q([A_1, \ldots, A_N])$: Laufzeit bei Eingabe $[A_1, \ldots, A_N]$

$Q^{wc}(N)$: worst-case Laufzeit auf N Objekten ($= \max_A Q([A_1, \ldots, A_N])$)

$Q^{ac}(N)$: average-case Laufzeit auf N Objekten ($= \mathbb{E}_A\, Q([A_1, \ldots, A_N])$)

$$Q([A_1, \ldots, A_N]) = N + Q([A'_1, \ldots, A'_{r-1}]) + Q([A'_{r+1}, \ldots, A'_N]) + 1$$

a) worst case:

Konzept:

(i) worst case finden,

(ii) Laufzeit berechnen,

(iii) Maximalität zeigen.

Im worst case ist r groß oder klein

(ausprobieren oder überlegen)

b) average case:

Jede der $N!$ möglichen Reihenfolgen (Permutationen) sei gleich wahrscheinlich.

Kennzeichen der Gleichverteilung (*uniform distribution*):

$$\mathbf{Pr}(A_{\pi(1)} \leq \ldots \leq A_{\pi(N)}) = \frac{1}{N!}$$

oder

$$\mathbf{Pr}(A_{\pi(1)} \leq \ldots \leq A_{\pi(N)}) = \mathbf{Pr}(A_{\sigma(1)} \leq \ldots \leq A_{\sigma(N)})$$

Gesucht:

$$Q^{ac}(N) = \frac{1}{n!} \sum_{\pi \in \mathfrak{S}_n} Q([\pi(1), \ldots, \pi(N)])$$

Wir wissen nur:

$$\begin{aligned} Q(A_1, \ldots, A_N) = N \quad &+ \quad Q(A'_1, \ldots, A'_{r-1}) \\ &+ \quad Q(A'_{r+1}, \ldots, A'_N) \; + \; 1 \end{aligned}$$

Wir müssen herausfinden:

- Verteilung von r
- Verteilung von $[A'_1, \ldots, A'_{N-1}]$

Ziel: $Q^{ac}(N) = g(Q^{ac}(1), Q^{ac}(2), \ldots, Q^{ac}(N-1))$

$\rightsquigarrow$ Rekursion auflösen

Behauptung: r ist gleichverteilt auf $\{1, \ldots, N\}$

Beweis: Sei $1 \leq r_0 \leq N$. Dann

$$\begin{aligned}
\mathbf{Pr}\ (\quad & r = r_0\,) = \\
= \ & \mathbf{Pr}(r_0 - 1 \text{ der } A_i \text{ sind kleiner als } V(= A_1)) \\
= \ & \mathbf{Pr}(\,A_1 = r_0\,) = \frac{1}{N}
\end{aligned}$$

Behauptung: Wenn auf der Eingabe die Gleichverteilung vorliegt, so auch auf den danach zu sortierenden Teillisten $(A'_1, \ldots, A'_{r-1})$, $(A'_{r+1}, \ldots, A'_N)$

Daher:

$$Q^{ac}(N) = N + Q^{ac}(r-1) + Q(N-r) + 1$$

Ergebnis

Satz: Quicksort benötigt zum Sortieren von N Schlüsseln höchstens $\frac{1}{2}N^2+\frac{3}{2}N+\mathcal{O}(1)$ Vergleiche.

Im Durchschnitt werden nur $2 \ln 2\, N \log N + \mathcal{O}(N)$ Vergleiche benötigt ($2 \ln 2 \approx 1.386\ldots$).

Verbesserung

Herabsetzten der wc-Komplexität durch Randomisierung