

Quicksort

Algorithmen und Datenstrukturen I

Prof. Dr. Oliver Braun

Letzte Änderung: 18.11.2019 07:38

- 1962: von C.A.R. (Tony) Hoare veröffentlicht
- eines der schnellsten, wenn nicht sogar *das* schnellste interne Sortierverfahren
- Divide-and-Conquer-Strategie
- im schlechtesten Fall $\Omega(N^2)$ viele Vergleichsoperationen, aber im Mittel nur $O(N \log N)$
- *In-situ-Sortierverfahren*, d.h. es wird zur (Zwischen-)Speicherung kein zusätzlicher Speicher benötigt, außer einer konstanten Anzahl von Hilfsspeicherplätzen für Tauschoperationen

- Sortieren durch rekursives Teilen
- Folge $F = k_1, \dots, k_N$ von N Schlüsseln sollen aufsteigend sortiert werden
- wähle ein beliebiges Element $k_p \in \{k_1, \dots, k_n\}$, genannt *Pivotelement*
- teile die Folge in zwei Teilfolgen F_1 und F_2 , für die gilt:

$$\forall k \in F_1 : k \leq k_p \text{ und } \forall k \in F_2 : k \geq k_p$$

- gilt $|F_1| = i - 1$ und $|F_2| = N - i$ so ist i die endgültige Position von k_p
- sortiere F_1 und F_2 rekursiv

```
1 void quicksort(int *a, int l, int r) {
2     if (r ≤ l) return;
3     int i = l - 1;
4     int j = r;
5     int p = a[r];
6     while (true) {
7         do { i++; } while (a[i] < p); // Schlüsselvergleich
8         do { j--; } while (a[j] > p); // Schlüsselvergleich
9         if (i ≥ j) break;
10        int t = a[i]; // Bewegung
11        a[i] = a[j]; // Bewegung
12        a[j] = t; // Bewegung
13    }
14    int t = a[i]; // Bewegung
15    a[i] = a[r]; // Bewegung
16    a[r] = t; // Bewegung
17    quicksort(a, l, i - 1);
18    quicksort(a, i + 1, r);
19 }
```

- im schlechtesten Fall
- zur Aufteilung eines Feldes der Länge N Vergleich aller Schlüssel im aufzuteilenden Bereich mit dem Pivotelement
 - in der Regel werden zwei Schlüssel je zweimal verglichen, d.h. $N + 1$ Vergleiche
- im ungünstigsten Fall wechseln alle Elemente je einmal ihren Platz
- d.h.

$$C_{max}(N) = \sum_{i=2}^N (i + 1) = \Omega(N^2)$$

- im günstigsten Fall haben die beiden Teilfolgen in jedem Schritt etwa die gleiche Länge
- damit hat der Baum der rekursiven Aufrufe die minimale Höhe, ca. $\log N$
- zur Aufteilung auf einem Niveau werden $\Theta(N)$ Schlüsselvergleiche durchgeführt
- d.h.

$$C_{min}(N) = \Theta(N \log N)$$

- bei einer Vertauschung im Aufteilungsschritt ist das eine Element kleiner und das andere größer als das Pivotelement
- am Schluß eines Aufteilungsschrittes ist das Pivotelement beteiligt
- d.h. Anzahl Vertauschungen höchstens so groß wie die Anzahl der Elemente in der kürzeren Teilfolge
- bei jedem rekursiven Schritt muss sich die Anzahl mindestens halbiert haben
- das kann höchstens $\log N$ mal passieren bis die kleinere Teilfolge nur noch aus einem Element besteht
- das gilt für jedes der N Elemente d.h. Anzahl der Bewegungen ist $O(N \log N)$

- man kann beweisen, dass Quicksort im Mittel

$$O(N \log N)$$

Zeit benötigt und das im Mittel

$$O(N \log N)$$

Schlüsselvergleiche notwendig sind

- Beweis in Ottmann & Widmayer: Algorithmen und Datenstrukturen, S. 98f

- vorgestelltes Verfahren benötigt für bereits sortierte oder fast sortierte Folgen quadratische Schrittzahl
- Verbesserung: *3-Median-Strategie* zur Wahl des Pivotelements
 - es werden drei Elemente gewählt
 - z.B. erstes, letztes und das in der Mitte
 - dann wird der Median der drei als Pivotelement verwendet
- anderer Ansatz: *Zufalls-Strategie*
 - das Pivotelement wird zufällig ausgewählt