
Algorithmen und Datenstrukturen

Prof. Dr. Ralf Möller

Universität zu Lübeck

Institut für Informationssysteme

Tanya Braun (Übungen)

sowie viele Tutoren



Danksagung

Die nachfolgenden 4 Präsentationen übernommen aus der Vorlesung „Effiziente Algorithmen und Datenstrukturen“ (Kapitel 5: Sortieren und Selektieren) gehalten von Christian Scheideler an der TUM

<http://www14.in.tum.de/lehre/2008WS/ea/index.html.de>

Selektion

Problem: finde k -kleinstes Element in einer Folge von n Elementen

Lösung: sortiere Elemente (z.B. Mergesort), gib k -tes Element aus ! Zeit $O(n \log n)$

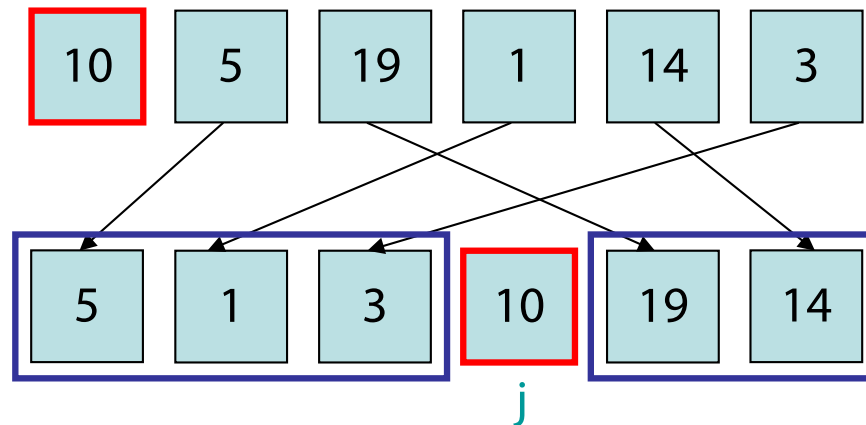
Geht das auch schneller??

Ganz sicher für $k=1$ (min) und $k=n$ (max)

Ausschluss von $n-1$ Elementen nötig, also $O(n)$

Selektion

Ansatz: Verfahren ähnlich zu Quicksort



- j : Position des Pivotelements nach Partitionierung
- $k < j$: mach mit linker Teilfolge weiter
- $k > j$: mach mit rechter Teilfolge weiter

Selektion

```
function QUICKSELECT(A, l, r, k)
  // a[l..r]: Restfeld, k: k-kleinstes Element,  $l \leq k \leq r$ 
  if r = l then return A[l]
  z := zufällige Position in {l,..,r}
  temp := A[z]; A[z] := A[r]; A[r] := temp
  v := A[r]; i := l-1; j := r
  repeat // ordne Elemente in [l,r-1] nach Pivot v
    repeat i := i + 1 until A[i] ≥ v
    repeat j := j - 1 until A[j] < v or j = l
    if i < j then temp := A[i]; A[i] := A[j]; A[j] := temp
  until j ≤ i
  temp := A[i]; A[i] := A[r]; A[r] := temp
  if k < i then e := Quickselect(A, l, i-1, k)
  if k > i then e := Quickselect(A, i+1, r, k)
  if k = i then e := A[k]
  return e
```

Quickselect: Analyse

- Aufwand $T(n)$: erwartete Anzahl Vergleiche

Behauptung: $T(n)$ in $O(n)$

Beweis:

- Pivot ist **gut**: keine der Teilfolgen länger als $n \cdot 2/3$
- Sei $p =$ Anteil der Fälle, in denen gilt: Pivot ist gut



- $p=1/3$
- Pivot **gut**: Restaufwand $\leq T(2n/3)$
- Pivot **schlecht**: Restaufwand $\leq T(n)$

Quickselect: Analyse

$$\begin{aligned}T(n) &\leq cn + p \cdot T(n \cdot 2/3) + (1 - p) \cdot T(n) \\p \cdot T(n) &\leq cn + p \cdot T(n \cdot 2/3) \\T(n) &\leq cn/p + T(n \cdot 2/3) \\&\leq cn/p + c \cdot (n \cdot 2/3)/p + T(n \cdot (2/3)^2) \\&\dots \text{wiederholtes Einsetzen} \\&\leq (cn/p)(1 + 2/3 + 4/9 + 8/27 + \dots) \\&\leq \frac{cn}{p} \cdot \sum_{i \geq 0} (2/3)^i \\&\leq \frac{cn}{1/3} \cdot \frac{1}{1 - 2/3} = 9cn \in O(n)\end{aligned}$$

geometrischen Reihe mit $a_0 = 1$:

$$\sum_{k=0}^{\infty} a_0 q^k = \frac{a_0}{1 - q}$$

Quickselect: Analyse

$$T(n) \leq cn + p \cdot T(n \cdot 2/3) + (1-p) \cdot T(n)$$

$$p \cdot T(n) \leq cn + p \cdot T(n \cdot 2/3)$$

$$T(n) \leq cn/p + T(n \cdot 2/3)$$

$$\leq cn/p + c \cdot (n \cdot 2/3)/p + T(n \cdot (2/3)^2)$$

... wiederholtes Einsetzen

$$\leq (cn/p)(1 + 2/3 + 4/9 + 8/27 + \dots)$$

$$\leq \frac{cn}{p} \sum_{i \geq 0} (2/3)^i$$

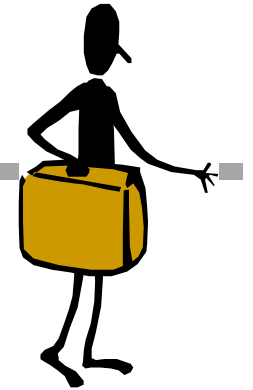
$$\leq \frac{cn}{1/3} \frac{1}{1 - 2/3}$$

geometrische Reihe mit $a_0 = 1$:

$$\sum_{k=0}^{\infty} a_0 q^k = \frac{a_0}{1 - q}$$

$$= 9cn \in O(n)$$

Überblick



- Bisher behandelt:
 - Sortieren durch Vergleichen (vorige Sitzungen)
 - Lineares Sortieren
 - Prioritätswarteschlangen
 - Selektion von Elementen aus einem Feld (z.B. Median)
- Es kommt:
 - Realisierung von Mengen
 - Assoziation von Objekten (über sog. Hashtabellen)