Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Lehrstuhl für
INFORMATIK I
Effiziente Algorithmen und
wissensbasierte Systeme

Institut für Informatik

Algorithmen und Datenstrukturen

Wintersemester 2017/18

10. Vorlesung

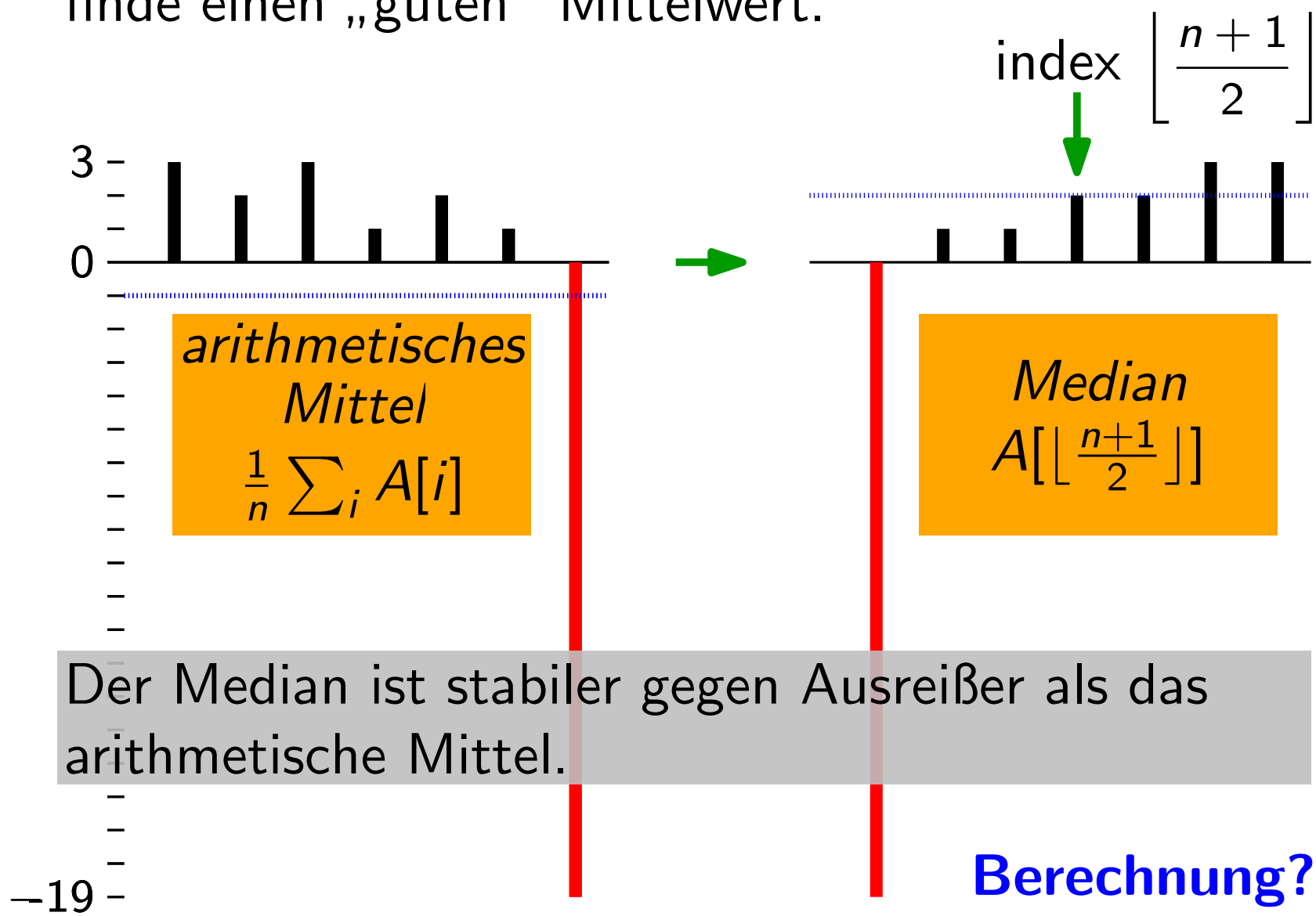
Das Auswahlproblem

Prof. Dr. Alexander Wolff *Lehrstuhl für Informatik I*

Analyse von Messreihen

Problem: Gegeben eine Reihe von n Messwerten $A[1..n]$, finde einen „guten“ Mittelwert.

Beispiel:



Beob.: Der Median ist stabiler gegen Ausreißer als das arithmetische Mittel.

Berechnung?

Das Auswahlproblem

Aufgabe: Gegeben ein Feld $A[1..n]$,
finde das i.-kleinste Element von A.

Lösung: Sortiere und gib $A[i]$ zurück!

Worst-Case-Laufzeit: $\Theta(n \log n)$ [wenn man nichts über die Verteilung der Zahlen weiß]

Geht das besser?

Spezialfälle

$i = \lfloor \frac{n+1}{2} \rfloor$: Median ← *Geht das auch in linearer Zeit??*

$i = 1$: Minimum

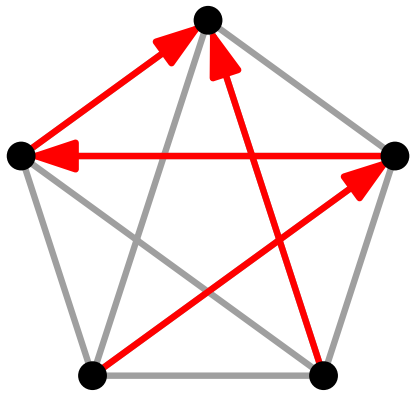
$i = n$: Maximum

} Laufzeit $\Theta(n)$ } Geht beides zusammen mit weniger als $2(n-1)$ Vergleichen?

```
Minimum(int[] A)
  min = A[1]
  for i = 2 to A.length do
    if min > A[i] then min = A[i]
  return min
```

Anzahl Vergleiche = $n - 1$

Ist das *optimal*?



Betrachte ein Turnier.

Bis ein Gewinner feststeht, muss *jeder* – außer dem Gewinner – mindestens einmal verlieren.

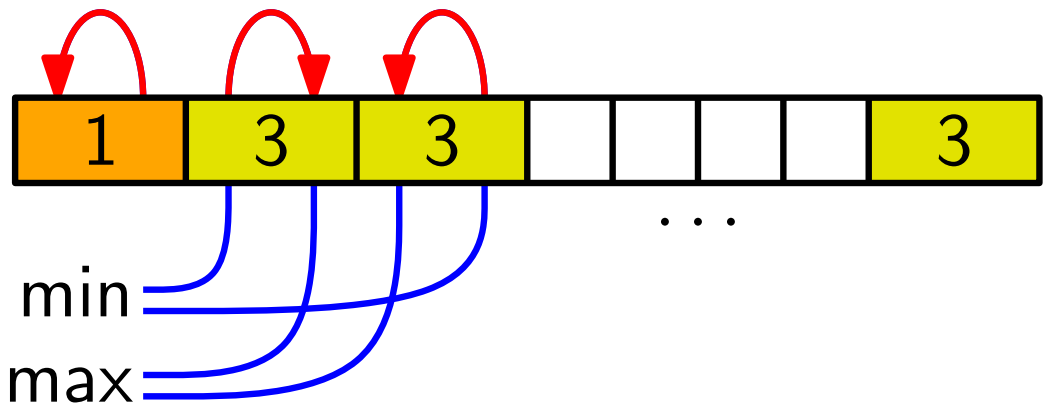
Also sind $n - 1$ Vergleiche optimal.

Eine Randbemerkung...

Def. Sei $V_{\text{minmax}}(n)$ die Anz. der Vgl., die man braucht um Minimum *und* Maximum von n Zahlen zu bestimmen.

Klar: $V_{\text{minmax}}(n) \leq 2 \cdot V_{\text{min}}(n) = 2(n-1)$

Frage: Geht es auch mit weniger Vergleichen? (n gerade)



$= 1 \cdot 1 + (n/2 - 1) \cdot 3$

$= 3n/2 - 2$

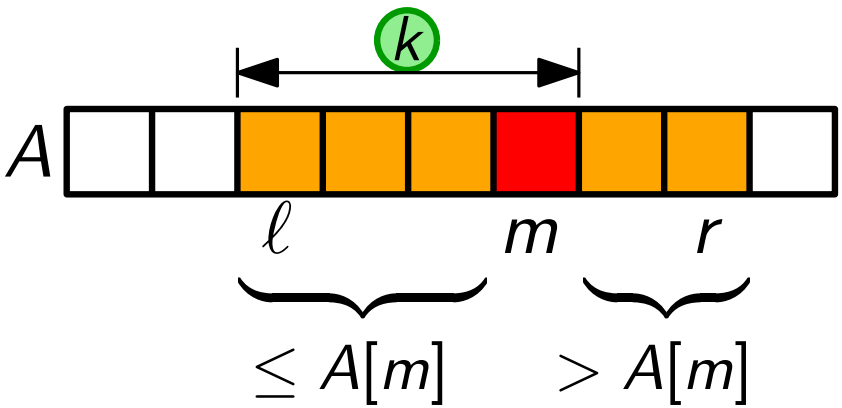
Ist das *optimal*?

Auswahl per Teile & Herrsche

Zur Erinnerung...

Randomized
QuickSort(int[] A, int ℓ, r)

if $\ell < r$ **then**
  $m =$ Randomized Partition(A, ℓ, r)
  QuickSort$(A, \ell, m-1)$
  QuickSort$(A, m+1, r)$



Finde i.-kleinstes Element in $A[\ell..r]$!

RandomizedSelect(int[] A, int ℓ, r, i)

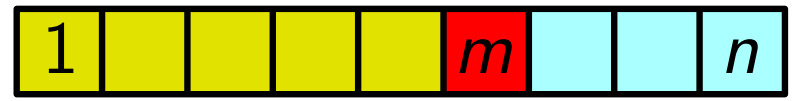
if $\ell == r$ **then return** $A[\ell]$
$m =$ RandomizedPartition(A, ℓ, r)
$k = m - \ell + 1$ // $A[m]$ k.-kleinstes El. von $A[\ell..r]$
if $i == k$ **then**
  return $A[m]$
else
  if $i < k$ **then**
    return RSelect$(A, \ell, m-1, i)$
  else
    return RSelect$(A, m+1, r, i-k)$

Ist Ihnen klar warum?

Laufzeit-Analyse

Anz. Vgl. von RandomizedSelect ist ZV; hängt von n und i ab.

Trick: Geh davon aus, dass das gesuchte i. Element immer im *größeren* Teilfeld liegt.



$\Rightarrow$ resultierende Zufallsvariable $V(n)$ ist

- obere Schranke für tatsächliche Anzahl von Vergleichen
- unabhängig von i

$$V(n) = \underbrace{V_{\text{Part}}(n)}_{= n-1} + \left\{ \begin{array}{ll} V(n-1) & \text{falls } m = 1 \\ V(n-2) & \text{falls } m = 2 \\ \cdots & \\ V(\lfloor \frac{n}{2} \rfloor) & \text{falls } m = \lfloor \frac{n}{2} \rfloor + 1 \\ \cdots & \\ V(n-2) & \text{falls } m = n-1 \\ V(n-1) & \text{falls } m = n \end{array} \right\}$$

Alle Fälle gleich wahrscheinlich!

[vorausgesetzt alle Elem. sind verschieden!]

$$\Rightarrow \boxed{E[V(n)] \leq n-1+2 \cdot \frac{1}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} E[V(k)]} \overset{?}{\leq} c \cdot n \quad \left(\begin{array}{l}\text{für ein} \\ c > 0\end{array}\right)$$

Substitutionsmethode

Wir schreiben $f(n)$ für $E[V(n)]$.

Dann gilt $f(n) \leq n + \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} f(k)$

Wir wollen prüfen, ob es ein $c > 0$ gibt, so dass $f(n) \leq cn$.

$$\text{Also:} \quad f(n) \leq n + \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} c \cdot k \qquad \text{[laut Annahme]}$$

Aufgabe:
Bestimmen Sie ein c, so dass $f(n) \leq cn$!
(Ignorieren Sie das Abrunden $\lfloor ... \rfloor$.)

$$E[V(n)] \leq n-1 + 2 \cdot \frac{1}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} E[V(k)]$$

Substitutionsmethode

Wir schreiben $f(n)$ für $E[V(n)]$.

Dann gilt $f(n) \leq n + \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} f(k)$

Wir wollen prüfen, ob es ein $c > 0$ gibt, so dass $f(n) \leq cn$.

Also:
$$\begin{aligned} f(n) &\leq n + \tfrac{2}{n} \textstyle\sum_{k=\lfloor n/2 \rfloor}^{n-1} c \cdot k \quad \text{[laut Annahme]} \\ &= n + \tfrac{2c}{n} \left(\textstyle\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lfloor n/2 \rfloor - 1} k \right) \\ &= n + \tfrac{2c}{n} \left(\tfrac{n(n-1)}{2} - \tfrac{\lfloor n/2 \rfloor (\lfloor n/2 \rfloor - 1)}{2} \right) \\ &\leq n + \tfrac{c}{n} \left(n(n-1) - (n/2-1)(n/2-2) \right) \\ &\leq n + c \cdot \tfrac{3n+2}{4} \quad = cn - \left(c \cdot \tfrac{n-2}{4} - n \right) \overset{?!}{\geq} 0 \\ &\leq cn \quad \text{falls } c \geq \tfrac{4n}{n-2} = \tfrac{4}{1-2/n} \xrightarrow[n \to \infty]{} 4^+ \end{aligned}$$

Für jedes $\varepsilon > 0$ gilt:

$$E[V(n)] \leq n-1+2 \cdot \tfrac{1}{n} \textstyle\sum_{k=\lfloor n/2 \rfloor}^{n-1} E[V(k)] \leq \overbrace{(4+\varepsilon)n}^{c :=} \left[\begin{matrix} \text{falls} \\ n \geq \frac{8}{\varepsilon} + 2 \end{matrix} \right]$$

Ergebnis und Diskussion

Satz. Das Auswahlproblem kann in erwartet linearer Zeit gelöst werden.

Genauer: Für jedes $\varepsilon > 0$ gilt, dass man in einer Folge von $n \geq \frac{8}{\varepsilon} + 2$ Zahlen die i.-kleinste Zahl $(1 \leq i \leq n)$ mit *erwartet* $(4 + \varepsilon)n$ Vergleichen finden kann.

Frage: Geht das auch *deterministisch*, d.h. ohne Zufall?

M.a.W.: Kann man das Auswahlproblem auch im *schlechtesten Fall* in linearer Zeit lösen?

Vorbereitung

Wir verwenden wieder Teile-und-Herrsche –
aber diesmal mit einer garantiert *guten* Aufteilung in Teilfelder.

d.h. *balanciert:*

jede Seite sollte $\geq \gamma n$ Elem. enthalten, für ein festes $0 < \gamma \leq \frac{1}{2}$.

Wir gehen für die Analyse wieder davon aus, dass alle Elemente verschieden sind.

Partition′$(A, \ell, r, pivot)$

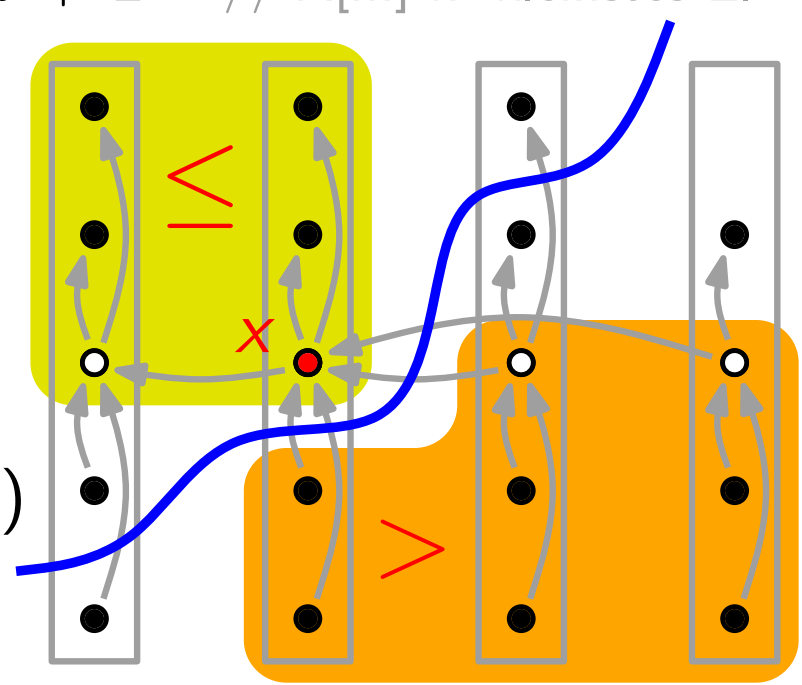
~~$pivot = A[r]$~~
$i = \ell - 1$
for $j = \ell$ **to** r ~~-1~~ **do**
  if $A[j] \leq pivot$ **then**
    $i = i + 1$
    Swap(A, i, j)
Swap$(A, i + 1, r)$
return $i + 1$

Select: deterministisch

Select(A, ℓ, r, i)

1. Teile die n Elem. der Eingabe in $\lfloor n/5 \rfloor$ 5er-Gruppen und eine Gruppe mit den restlichen (n mod 5) Elem.
2. Sortiere jede der $\lceil n/5 \rceil$ Gruppen und bestimme ihren Median.
3. Bestimme *rekursiv* den Median x der Gruppen-Mediane.
4. $m =$ Partition$'(A, \ell, r, x)$; $k = m - \ell + 1$ // $A[m]$ k.-kleinstes El.
5. **if** $i == k$ **then return** $\underbrace{A[m]}_{x}$
   else
   - **if** $i < k$ **then**
     - **return** Select$(A, \ell, m-1, i)$
   - **else**
     - **return** Select$(A, m+1, r, i-k)$

Anzahl(●) $\geq 3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right) \geq \frac{3n}{10} - 6$

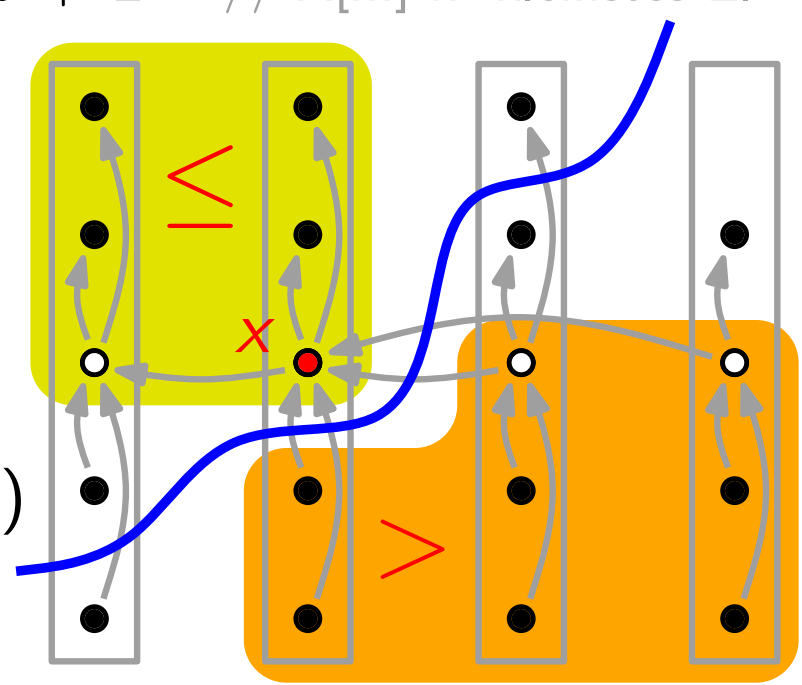


Select: deterministisch

Select(A, ℓ, r, i)

1. Teile die n Elem. der Eingabe in $\lfloor n/5 \rfloor$ 5er-Gruppen und eine Gruppe mit den restlichen (n mod 5) Elem.
2. Sortiere jede der $\lceil n/5 \rceil$ Gruppen und bestimme ihren Median.
3. Bestimme *rekursiv* den Median x der Gruppen-Mediane.
4. $m =$ Partition$'(A, \ell, r, x)$; $k = m - \ell + 1$ // $A[m]$ k.-kleinstes El.
5. **if** $i == k$ **then return** $\underbrace{A[m]}_{x}$
   else
   - **if** $i < k$ **then**
     - **return** Select$(A, \underbrace{\ell, m-1}_{\leq 7n/10+6 \text{ Elem.}}, i)$
   - **else**
     - **return** Select$(A, \overbrace{m+1, r}, i-k)$

Anzahl(●) $\geq 3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right) \geq \frac{3n}{10} - 6$



Laufzeit-Analyse

Beob. Es genügt wieder, Vergleiche zu zählen!

Partition′: $\approx 1n$, Sortieren: $\approx \frac{n}{5} \cdot V_{\mathsf{IS}}(5) = 2n$ Vgl.

Ansatz:

$$V(n) \leq \begin{cases} \overbrace{V(\lceil n/5 \rceil)}^{\text{Schritt 3}} + \underbrace{V(7n/10+6)}_{\text{Schritt 5}} + 3n & \text{falls } n \geq n_0, \\ O(1) & \text{sonst.} \end{cases}$$

Laufzeit-Analyse

Beob. Es genügt wieder, Vergleiche zu zählen!

Partition′: $\approx 1n$, Sortieren: $\approx \frac{n}{5} \cdot V_{\mathsf{IS}}(5) = 2n$ Vgl.

Ansatz:

$$V(n) \leq \begin{cases} V(\lceil n/5 \rceil) + V(7n/10 + 6) + 3n & \text{falls } n \geq n_0, \\ O(1) & \text{sonst.} \end{cases}$$

Behauptung:

Es gibt $c, n_0 > 0$, so dass für alle $n \geq n_0$ gilt: $V(n) \leq cn$.

$$\Rightarrow \quad V(n) \leq c \cdot (n/5 + 1) + c \cdot (7n/10 + 6) + 3n \qquad \overset{?!}{\geq} 0$$

$$= c \cdot (9n/10 + 7) + 3n = cn - \big(c \cdot (n/10 - 7) - 3n\big)$$

falls $c \geq \frac{3n}{n/10-7} = \frac{30}{1-70/n} \xrightarrow[n \to \infty]{}$

Laufzeit-Analyse

Beob. Es genügt wieder, Vergleiche zu zählen!

Partition′: $\approx 1n$, Sortieren: $\approx \frac{n}{5} \cdot V_{\mathsf{IS}}(5) = 2n$ Vgl.

Ansatz:

$$V(n) \leq \begin{cases} V(\lceil n/5 \rceil) + V(7n/10 + 6) + 3n & \text{falls } n \geq n_0, \\ O(1) & \text{sonst.} \end{cases}$$

Behauptung:

Es gibt $c, n_0 > 0$, so dass für alle $n \geq n_0$ gilt: $V(n) \leq cn$.

$$\Rightarrow \quad V(n) \leq c \cdot (n/5 + 1) + c \cdot (7n/10 + 6) + 3n$$

$$= c \cdot (9n/10 + 7) + 3n = cn - \big(c \cdot (n/10 - 7) - 3n\big)$$

falls $c \geq \frac{3n}{n/10-7} = \frac{30}{1-70/n} \xrightarrow[n \to \infty]{} 30^+$ bzw. $n \geq \frac{70c}{c-30}$.

$\Rightarrow$ für jedes $\varepsilon > 0$ und $n \geq \frac{2100}{\varepsilon} + 70$ gilt: $V(n) \leq \underbrace{(30 + \varepsilon)}_{c} \cdot n$

Laufzeit-Analyse

Kann man das verbessern?

Hausaufgabe!

Beob. Es genügt wieder, Vergleiche zu zählen!

Partition′: $\approx 1n$, Sortieren: $\approx \frac{n}{5} \cdot V_{\mathsf{IS}}(5) = 2n$ Vgl.

Ansatz:

$$V(n) \leq \begin{cases} V(\lceil n/5 \rceil) + V(7n/10+6) + 3n & \text{falls } n \geq n_0, \\ O(1) & \text{sonst.} \end{cases}$$

Behauptung:

Es gibt $c, n_0 > 0$, so dass für alle $n \geq n_0$ gilt: $V(n) \leq cn$.

$$\Rightarrow \quad V(n) \leq c \cdot (n/5+1) + c \cdot (7n/10+6) + 3n$$

$$= c \cdot (9n/10+7) + 3n = cn - \big(c \cdot (n/10-7) - 3n\big)$$

falls $c \geq \frac{3n}{n/10-7} = \frac{30}{1-70/n} \xrightarrow[n\to\infty]{} 30^+$ bzw. $n \geq \frac{70c}{c-30}$.

$\Rightarrow$ für jedes $\varepsilon > 0$ und $n \geq \frac{2100}{\varepsilon} + 70$ gilt: $V(n) \leq (30+\varepsilon) \cdot n$

Ergebnis und Diskussion

Satz: Das Auswahlproblem kann auch im schlechtesten Fall in linearer Zeit gelöst werden.

Genauer: Für jedes $\varepsilon > 0$ gilt, dass man in einer Folge von $n \geq 2100/\varepsilon + 70$ Zahlen die i.-kleinste Zahl mit höchstens $(30 + \varepsilon)n$ Vergleichen finden kann.

Literatur: *Randomized Algorithms* [Motwani+Raghavan, Cambridge U Press, '95]
Algorithmen und Zufall [Vorlesungsskript, Jochen Geiger, Uni KL]

- Der Algorithmus LazySelect [Floyd & Rivest, 1975] löst das Auswahlproblem mit WK $1 - O(1/\sqrt[4]{n})$ mit $\frac{3}{2}n + o(n)$ Vgl.
- Die besten deterministischen Auswahl-Algorithmen (*sehr kompliziert!*) benötigen $3n$ Vergleiche im schlechtesten Fall.
- *Jeder* deterministische Auswahl-Alg. benötigt im schlechtesten Fall mindestens $2n$ Vergleiche.