

Teil 3:

# Das Auswahlproblem

# Das Auswahlproblem

Im Folgenden wollen wir ein Problem behandeln, das eng mit dem Sortierproblem verwandt ist.

## SELECT

**Eingabe:** Eine Folge von paarweise verschiedenen Elementen  $S_1, \dots, S_n$  aus einer geordneten Menge und ein Index  $i$  mit  $1 \leq i \leq n$ .

Finde das  $i$ -t kleinste Element, d.h. einen Schlüssel  $S_j$ , so dass für  $i - 1$  Schlüssel  $S_k < S_j$  und für  $n - i$  Schlüssel  $S_k > S_j$  gilt.

## Bemerkung

*Das Problem SELECT ist eng mit dem Problem MEDIAN verwandt:*

- $LOWER\_MEDIAN(S_1, \dots, S_n) = SELECT(S_1, \dots, S_n, \lceil \frac{n}{2} \rceil)$
- $UPPER\_MEDIAN(S_1, \dots, S_n) = SELECT(S_1, \dots, S_n, \lceil \frac{n+1}{2} \rceil)$

# Eine einfache Lösung

Eine mögliche Lösung ist natürlich:

## Algorithmus (SELECT mittels SORT)

**Eingabe:** Eine Folge von *paarweise verschiedenen* Elementen  $S_1, \dots, S_n$  aus einer geordneten Menge und ein Index  $i$  mit  $1 \leq i \leq n$

**Ausgabe:** Das  $i$ -t kleinste Element

*Sortiere die Eingabe*

*Wähle aus der Sortierten Liste das  $i$ -te Element aus*

Die Laufzeit dieses Verfahrens ist im schlechtesten und mittleren Fall mindestens  $\Omega(n \log n)$  bei allgemeinen Schlüsseln.

Es gibt jedoch eine weitere einfache Lösung, die im Mittel sogar **linear** ist.

## Algorithmus (FIND)

**Eingabe:** Eine Menge  $M$  von *paarweise verschiedenen* Elementen  $S_1, \dots, S_n$  aus einer geordneten Menge und ein Index  $i$  mit  $1 \leq i \leq n$   
**Ausgabe:** Das  $i$ -t kleinste Element

Wähle ein zufälliges Element  $S$  aus

$M_1 := \{S_k \mid S_k < S\}$

$M_2 := \{S_k \mid S_k > S\}$

**wenn**  $|M_1| = i - 1$  **dann** gebe  $S$  aus

**wenn**  $|M_1| \geq i$  **dann** gebe  $FIND(M_1, i)$  aus

**wenn**  $|M_1| < i - 1$  **dann** gebe  $FIND(M_2, i - |M_1| - 1)$  aus

Die Korrektheit des Algorithmus ist sofort intuitiv klar. Es bleibt die Laufzeit zu untersuchen.

## Die Laufzeit von FIND

Im schlechtesten Fall wählen wir in jeder Runde den kleinsten oder größten Schlüssel. Das führt zu insgesamt  $n - 1$  Runden mit den Laufzeiten  $O(n - 1), O(n - 2), \dots, O(1)$ , denn in der  $k$ -ten Runde muss der gewählte Schlüssel mit  $n - k$  anderen verglichen werden. Also ergibt sich insgesamt  $O(n^2)$  im schlechtesten Fall.

Betrachten wir die Laufzeit im mittleren Fall.

Wir haben  $n$  Elemente und nehmen an, dass das gewählte Element  $S$  mit Wahrscheinlichkeit  $\frac{1}{n}$  das  $k$ -t kleinste ist, d.h.

$$P(S \text{ ist das } k\text{-t kleinste Element}) = \frac{1}{n} \text{ für } 1 \leq j \leq n.$$

## Die mittlere Laufzeit von FIND

Sei  $T(n, i)$  die mittlere Laufzeit von  $\text{FIND}(S_1, \dots, S_n; i)$  und  $T(n) := \max_i(T(n, i))$ .

Offensichtlich gilt  $T(1) \leq C$  für eine positive Konstante  $C$ .

Sei  $S$  das  $k$ -t kleinste Element, dann gilt

$$|M_1| = k - 1 \text{ und } |M_2| = n - 1 - |M_1| = n - k.$$

Damit ergeben sich die folgenden Fälle:

- Falls  $k = i$  wird nur  $S$  zurück gegeben.
- Falls  $k \geq i + 1$  wird  $\text{FIND}(M_1, i)$  aufgerufen.
- Falls  $k < i$  wird  $\text{FIND}(M_2, i - k)$  aufgerufen.

# Die mittlere Laufzeit von FIND

Damit ergibt sich die folgende Abschätzung:

$$T(n, i) \leq \underbrace{Cn}_{\text{Teilen}} + \frac{1}{n} \left( \underbrace{\sum_{k=1}^{i-1} T(n-k, i-k)}_{k < i} + \underbrace{D}_{k=i} + \underbrace{\sum_{k=i+1}^n T(k-1, i)}_{k \geq i+1} \right)$$

Und somit

$$T(n) \leq Cn + \frac{1}{n} \max_i \left( \sum_{k=1}^{i-1} T(n-k) + D + \sum_{k=i}^{n-1} T(k) \right)$$

## Lemma

$$T(n) \leq 4Cn + D \text{ für } n \geq 1$$

## Beweis

Für  $n = 1$  gilt  $T(1) \leq C \leq 4Cn + D$ .

Für  $n > 1$  ergibt sich:

$$\begin{aligned} T(n) &\leq Cn + \frac{1}{n} \max_i \left( \sum_{k=1}^{i-1} T(n-k) + D + \sum_{k=i}^{n-1} T(k) \right) \\ &\leq Cn + \frac{D}{n} + \frac{1}{n} \max_i \left( \sum_{k=1}^{i-1} (4C(n-k) + D) + \sum_{k=i}^{n-1} (4Ck + D) \right) \\ &= Cn + \frac{D}{n} + \frac{1}{n} \max_i \left( \sum_{k=n-i+1}^{n-1} 4Ck + \sum_{k=i}^{n-1} 4Ck + (n-1)D \right) \end{aligned}$$

## Beweis (Fortsetzung)

$$\begin{aligned} T(n) &= Cn + \frac{D}{n} + \frac{1}{n} \max_i \left( \sum_{k=n-i+1}^{n-1} 4Ck + \sum_{k=i}^{n-1} 4Ck + (n-1)D \right) \\ &= Cn + D + \frac{4C}{n} \max_i \left( \sum_{k=n-i+1}^{n-1} k + \sum_{k=i}^{n-1} k \right) \\ &= Cn + D + \frac{4C}{n} \max_i \left( \frac{n(n-1)}{2} - \frac{(n-i+1)(n-i)}{2} \right. \\ &\quad \left. + \frac{n(n-1)}{2} - \frac{i(i-1)}{2} \right) \\ &= Cn + D + \frac{4C}{n} \max_i \left( n(n-1) - \frac{(n-i+1)(n-i)}{2} - \frac{i(i-1)}{2} \right) \end{aligned}$$

## Beweis (Fortsetzung)

$$\begin{aligned} T(n) &= Cn + D + \frac{4C}{n} \max_i \left( n(n-1) - \frac{(n-i+1)(n-i)}{2} - \frac{i(i-1)}{2} \right) \\ &= Cn + D + \frac{4C}{n} \max_i \left( \frac{n^2}{2} - \frac{3n}{2} + ni - i^2 + i \right) \end{aligned}$$

Um das Maximum abzuschätzen, betrachten wir die Funktion

$$f(x) = \frac{n^2}{2} - \frac{3n}{2} + nx - x^2 + x$$

auf dem Intervall  $[1, n]$ . Es ergibt sich

$$f'(x) = n - 2x + 1 \text{ und } f''(x) = -2$$

D.h. an den Nullstellen von  $f'$  liegt ein lokales Maximum vor.

## Beweis (Fortsetzung)

Es gilt

$$f'(x) = n - 2x + 1 = 0 \quad \Leftrightarrow \quad x = \frac{n+1}{2}.$$

Für  $n \geq 1$  liegt dieser Wert im Intervall  $[1, n]$ .

Damit gilt

$$f(x) \leq \frac{n^2}{2} - \frac{3n}{2} + \frac{n(n+1)}{2} - \frac{(n+1)^2}{4} + \frac{n+1}{2} = \frac{3}{4}n^2 - \frac{1}{2}(n-1)$$

uns somit

$$\begin{aligned} T(n) &\leq Cn + D + \frac{4C}{n} \left( \frac{3}{4}n^2 - \frac{1}{2}(n-1) \right) \\ &\leq Cn + D + \frac{4C}{n} \frac{3}{4}n^2 \\ &= Cn + D + 3Cn = 4Cn + D \quad \square \end{aligned}$$

## Satz

*Der Algorithmus FIND hat auf  $n$  Elementen eine mittlere Laufzeit von  $O(n)$  und benötigt im schlechtesten Fall  $O(n^2)$ .*

Die Laufzeit im schlechtesten Fall kann aber noch weiter verbessert werden.

Ähnlich wie bei der Verbesserung von QUICKSORT, können wir statt nur eines Elementes mehrere Elemente wählen und das Element  $S$ , an dem wir die Menge aufteilen, aus diesen auswählen.

Eine Möglichkeit hierfür wäre  $S$  als Median von  $k$  zufälligen Elementen zu wählen.

Das reicht aber nicht um im schlechtesten Fall lineare Laufzeit zu erhalten.

## Algorithmus (SELECT)

**Eingabe:** Eine Menge  $M$  von *paarweise verschiedenen* Elementen  $S_1, \dots, S_n$  aus einer geordneten Menge und ein Index  $i$  mit  $1 \leq i \leq n$

**Ausgabe:** Das  $i$ -te kleinste Element

**wenn**  $n \leq 100$  **dann** Sortiere  $M$  und bestimme das  $i$ -te Element

**sonst**

**für**  $i = 1, \dots, k := \lceil \frac{n}{5} \rceil$  **tue**

$M_i := \{S_{5(i-1)+1}, \dots, S_{5i}\}$  (soweit wie möglich)

Sortiere  $M_i$  und setze  $m_i := \text{MEDIAN}(M_i)$

**Ende**

$S := \text{SELECT}(\{m_1, \dots, m_k\}, \lceil \frac{k}{2} \rceil)$

$A := \{S_j \mid S_j \leq S\}$

$B := \{S_j \mid S_j > S\}$

**wenn**  $|A| \geq i$  **dann** gebe  $\text{SELECT}(A, i)$  aus

**wenn**  $|A| < i$  **dann** gebe  $\text{SELECT}(B, i - |A|)$  aus

**Ende**

# SELECT

## Lemma

$$|A|, |B| \geq \frac{3}{11}n \text{ für } n \geq 100$$

## Beweis

Sei  $n \geq 100$  und  $n = 10x + y$  mit  $0 \leq y < 10$ , d.h.  $x \geq 10$ .

Damit gilt

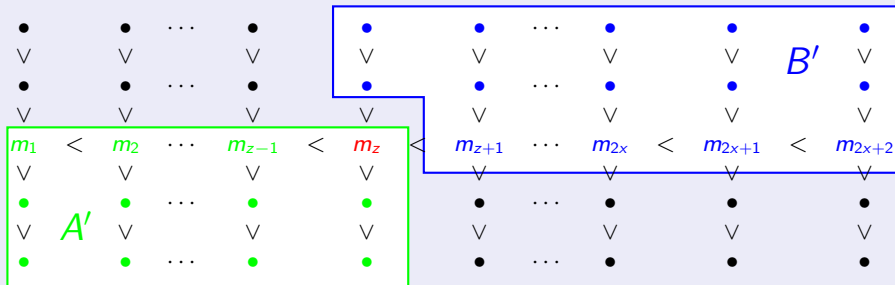
$$\frac{n}{5} = 2x + \frac{y}{5} \text{ und somit } 2x \leq \left\lceil \frac{n}{5} \right\rceil = k \leq 2x + 2.$$

Dies führt zu

$$x \leq \left\lceil \frac{\left\lceil \frac{n}{5} \right\rceil}{2} \right\rceil = \left\lceil \frac{k}{2} \right\rceil =: z \leq x + 1$$

## Beweis (Fortsetzung)

Ordnen wir die Mengen  $M_j$  nach ihrem Median, können wir sie folgendermaßen anordnen:

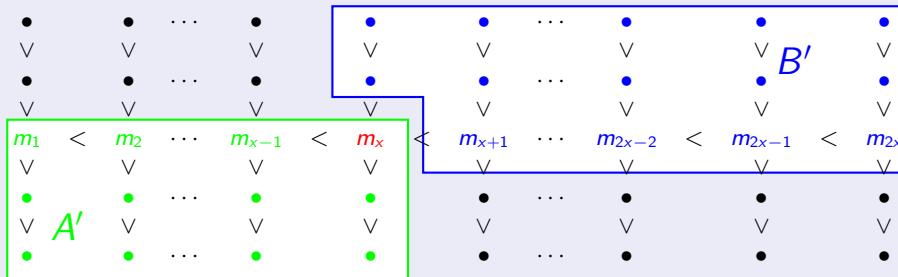


Dabei können die letzten beiden Spalten leer sein!

$A'$  enthält nur Elemente  $\leq m_z$ , d.h.  $A' \subseteq A$

$B'$  enthält nur Elemente  $> m_z$ , d.h.  $B' \subseteq B$

## Beweis (Fortsetzung)



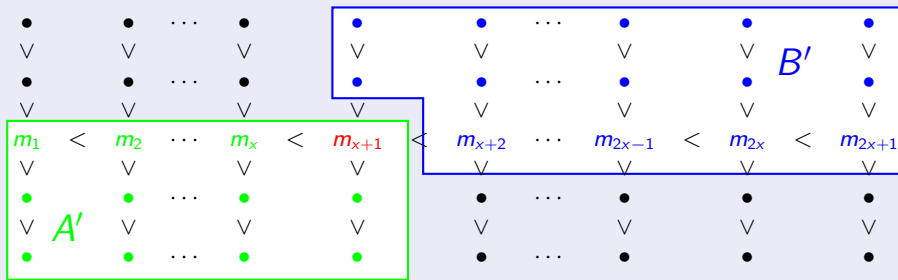
Falls  $y = 0$  gilt, gilt  $z = x$ ,  $k = 2x$  und alle  $|M_i| = 5$  für alle  $i$ . Damit gilt

$$|A| \geq |A'| = 3x \text{ und } |B| \geq |B'| = 3x + 2 \geq 3x.$$

Da  $n = 10x$  gilt, ergibt sich

$$|A|, |B| \geq 3x.$$

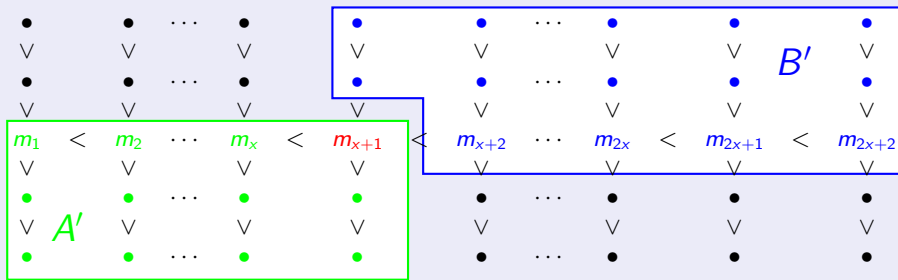
## Beweis (Fortsetzung)



Falls  $1 \leq y \leq 5$  gilt, gilt  $z = x + 1$  und  $k = 2x + 1$ . Da höchstens eine Menge weniger als fünf Elemente enthält, sind unter den ersten  $x + 1$  Mengen mindestens  $x$  mit fünf Elementen. Daher gilt  $|A| \geq |A'| \geq 3x$ .

Von den Mengen  $M_{x+2}, \dots, M_{2x+1}$  sind mindestens  $(x - 1)$  vollständig und höchstens eine unvollständig. Außerdem enthält  $B'$  zwei Elemente aus  $M_{x+1} = M_z$ . Somit gilt  $|B| \geq |B'| \geq 3(x - 1) + 1 + 2 = 3x$ .

## Beweis (Fortsetzung)



Falls  $6 \leq y \leq 9$  gilt, gilt  $z = x + 1$  und  $k = 2x + 2$ . Da höchstens eine Menge weniger als fünf Elemente enthält, sind unter den ersten  $x + 1$  Mengen mindestens  $x$  mit fünf Elementen. Daher gilt  $|A| \geq |A'| \geq 3x$ .

Von den Mengen  $M_{x+2}, \dots, M_{2x+2}$  sind mindestens  $(x - 1)$  vollständig, höchstens eine unvollständig. Außerdem enthält  $B'$  zwei Elemente aus  $M_{x+1} = M_z$ . Somit gilt  $|B| \geq |B'| \geq 3(x - 1) + 1 + 2 = 3x$ .

## Beweis (Fortsetzung)

In allen Situationen erhalten wir somit

$$|A|, |B| \geq 3x.$$

Wegen  $n \geq 100$  und  $x = \frac{n-y}{10}$  führt dies zu

$$\begin{aligned} |A|, |B| \geq 3x &= \frac{3}{10}(n-y) \geq \frac{3n}{10} - \frac{27}{10} = \frac{33n - 297}{110} \\ &= \frac{30n + 3n - 297}{110} = \frac{30}{110}n + \frac{3n - 297}{110} \\ &\geq \frac{3}{11}n + \frac{300 - 297}{110} = \frac{3}{11}n + \frac{3}{110} \\ &\geq \frac{3}{11}n \end{aligned}$$



## Lemma

Es gibt Konstanten  $a$  und  $b$ , so dass

$$T(n) \leq \begin{cases} an & \text{für } n \leq 100 \\ T\left(\frac{21n}{100}\right) + T\left(\frac{8n}{11}\right) + bn & \text{für } n > 100. \end{cases}$$

## Beweis

Für  $n \leq 100$  ist die Behauptung offensichtlich erfüllt, da dann mit  $O(100 \log 100)$  sortiert wird.

Für  $n > 100$  wird *SELECT* zweimal aufgerufen:

Einmal für die Mediane, d.h. für

$$\left\lceil \frac{n}{5} \right\rceil \leq \frac{n}{5} + \frac{4}{5} < \frac{n}{5} + \frac{1}{100}n = \frac{21}{100}n$$

Elemente.

## Beweis (Fortsetzung)

Das zweite Mal für höchstens  $\frac{8n}{11}$  Elemente, denn

$$\max(|A|, |B|) \leq n - \min(|A|, |B|) \leq n - \frac{3}{11}n = \frac{8}{11}n.$$

Die übrigen Operationen, auch das Sortieren aller Fünfer-Gruppen, kosten höchstens  $O(n)$  Schritte.

Das ergibt insgesamt:

$$T(n) \leq bn + T\left(\frac{21}{100}n\right) + T\left(\frac{8}{11}n\right).$$



## Satz

*SELECT* findet das  $i$ -t kleinste Element in  $O(n)$  Schritten.

## Beweis

O.B.d.A. können wir annehmen, dass  $b \geq \frac{69}{1100}a$  gilt. Sei

$$c := \frac{1100}{69}b \text{ und somit } a \leq c \text{ und } b = \frac{69}{1100}c.$$

Wir beweisen induktiv, dass  $T(n) \leq c$ .

Für  $n \leq 100$  gilt die Behauptung offensichtlich, denn  $a \leq c$ .

Für  $n > 100$  gilt:

$$\begin{aligned} T(n) &\leq T\left(\frac{21}{100}n\right) + T\left(\frac{8}{11}n\right) + bn \leq c\frac{21}{100}n + c\frac{8}{11}n + c\frac{69}{1100}n \\ &= cn\left(\frac{231}{1100} + \frac{800}{1100} + \frac{69}{1100}\right) = cn \quad \square \end{aligned}$$

## Auswahl in linearer Zeit

Mit dem Entwurf und der Analyse von SELECT zeigten Blum, Floyd, Pratt, Rivest und Tarjan 1973, dass das  $i$ -t kleinste Element schneller direkt gefunden werden kann, als durch die Sortierung.

Dabei verwendeten Sie wieder nur Vergleiche zwischen unstrukturierten Schlüsseln.

Für den Spezialfall des Medians ist immer noch unbekannt, wieviel Vergleiche benötigt werden. Bent und John bewiesen 1985 eine untere Schranke von mindestens  $2n$  Vergleichen. 1995 verbesserten Dor und Zwick die untere Schranke auf einen Wert leicht oberhalb von  $2n$  und bewiesen eine obere Schranke von  $2.95n$ .