

4.8 Das Auswahlproblem

Ziel: Finde das Datum x , das in der sortierten Folge an Rang k steht.

(Es ist dann auch die Menge der Daten partitioniert in die Daten $> x$, $= x$, $< x$.)

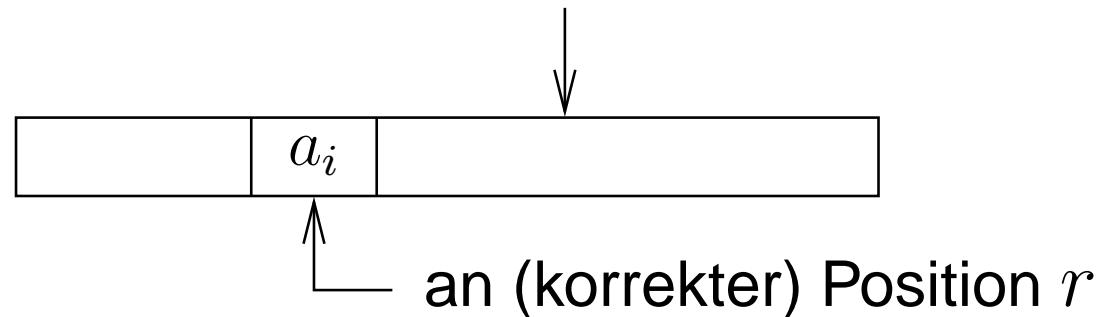
Algorithmen mit Worstcase-Zeit $O(n)$ sind kompliziert und praktisch nicht sehr effizient (große Konstante).

Was hilft? **Randomisierung!**

Quickselect:

Phase 1: Wie Phase 1 von Quicksort mit Zerlegungsstrategie 3:

(Zerlegungsdatum von zufälliger Position i)



Phase 2: Quickselect auf **einem** der beiden entstandenen Teilarrays...

Quickselect (Forts.):

Auswahl aus entstandenem Teilarray:

$r = k$: Glück gehabt, a_i ist das gesuchte Datum.

$r > k$: Suche auf dem linken Teil (Größe $r - 1$) nach dem Datum mit Rang k .

$r < k$: Suche auf dem rechten Teil (Größe $n - r$) nach dem Datum mit Rang $k - r$.

Im Gegensatz zu Quicksort nur ein Teilproblem.

Korrektheit klar.

Rechenzeitanalyse für den Averagecase (bei n verschiedenen Daten):

→ Rang des Zerlegungsdatum gleichverteilt auf $\{1, \dots, n\}$.

→ Größe des zu lösenden Teilproblems

$r = 1:$	$n - 1$	neuer Rang	$k - 1$
$r = 2:$	$n - 2$		$k - 2$
\vdots			\vdots
$r = k - 1:$	$n - (k - 1)$		1
$r = k:$	0		—
$r = k + 1:$	k		k
$r = k + 2:$	$k + 1$		k
\vdots			\vdots
$r = n:$	$n - 1$		k

Sei $V_k(n)$ Averagecase-Anzahl an wesentlichen Vergleichen bei n Daten und Rang k .

$$\begin{aligned} V_k(n) = n - 1 + \frac{1}{n} & (V_{k-1}(n-1) + V_{k-2}(n-2) + \dots \\ & + V_1(n - (k-1)) \\ & + 0 \\ & + V_k(k) + V_k(k+1) + \dots + V_k(n-1)) \end{aligned}$$

Das sieht ja schlimm aus!

$$V(n) := \max\{V_k(n) | 1 \leq k \leq n\}.$$

Sei k Rang (abh. von n), für den Maximum angenommen wird.

Dann:

$$\begin{aligned} V(n) &= V_k(n) = \dots \\ &\leq n - 1 + \frac{1}{n} \left(V(n-1) + V(n-2) + \dots + V(n-k+1) \right. \\ &\quad \left. + V(k) + V(k+1) + \dots + V(n-1) \right) \end{aligned}$$

$$V(n) \leq n - 1 + \frac{1}{n} \left(\sum_{i=n-k+1}^{n-1} V(i) + \sum_{i=k}^{n-1} V(i) \right).$$

Wir kennen keine Methode zum Lösen dieser Ungleichung.

Rechnen unter Annahmen,

Vermutung bilden,

Vermutung beweisen.

Vermutung 1: $V(\cdot)$ ist monoton wachsend.

Dann maximaler Wert bei $k = \lceil n/2 \rceil$.

$$\rightarrow V(n) \leq n - 1 + \frac{1}{n} \left(\sum_{\lfloor n/2 \rfloor + 1 \leq i \leq n-1} V(i) + \sum_{\lceil n/2 \rceil \leq i \leq n-1} V(i) \right)$$

Vermutung 2: $V(n) \leq cn$, aber welches c ?

Durchschnittliche Größe der Arrays in der 2.Phase:

$(3/4)n$, dann $(9/16)n$, ...

$$\rightarrow V(n) \leq n + \left(\frac{3}{4}\right)n + \left(\frac{9}{16}\right)n + \dots = \frac{1}{1 - 3/4} n = 4n.$$

Behauptung: $V(n) \leq 4n.$

Induktionsbeweis:

$n = 1$: $V(1) = 0 \leq 4.$

$1, \dots, n - 1 \rightarrow n$:

$4n$ ist monoton wachsend. Ungleichung und Ind.-Vor. liefern:

$$V(n) \leq n - 1 + \frac{1}{n} \left(\sum_{i=n-k+1}^{n-1} V(i) + \sum_{i=k}^{n-1} V(i) \right)$$

$$\stackrel{k=\lceil n/2 \rceil}{\leq} n - 1 + \frac{1}{n} \left(\sum_{\lfloor n/2 \rfloor + 1 \leq i \leq n-1} 4i + \sum_{\lceil n/2 \rceil \leq i \leq n-1} 4i \right)$$

$$\begin{aligned}
V(n) &\leq n - 1 + \frac{1}{n} \left(\sum_{\lfloor n/2 \rfloor + 1 \leq i \leq n-1} 4i + \sum_{\lceil n/2 \rceil \leq i \leq n-1} 4i \right) \\
&= n - 1 + \frac{4}{n} \left(\frac{1}{2} \cdot n \cdot (n - 1) - \frac{1}{2} \cdot (\lfloor n/2 \rfloor + 1) \cdot \lfloor n/2 \rfloor \right. \\
&\quad \left. + \frac{1}{2} \cdot n \cdot (n - 1) - \frac{1}{2} \cdot \lceil n/2 \rceil \cdot (\lceil n/2 \rceil - 1) \right) \\
&= n - 1 + \frac{4}{n} \left(n^2 - n - \frac{1}{2} \lceil n/2 \rceil^2 - \frac{1}{2} \lfloor n/2 \rfloor^2 + \frac{1}{2} \lceil n/2 \rceil - \frac{1}{2} \lfloor n/2 \rfloor \right) \\
&\quad \lceil n/2 \rceil^2 + \lfloor n/2 \rfloor^2 \geq (n/2)^2 + (n/2)^2 = n^2/2 \\
&\leq n - 1 + \frac{4}{n} \left(n^2 - n - \frac{1}{4} n^2 + 1 \right) \\
&\leq n + \frac{4}{n} \cdot \frac{3}{4} \cdot n^2 = 4n.
\end{aligned}$$



- Vermutlich ist $k = n/2$ der Worstcase, Mediansuche.
- Aber dann suchen wir in Phase 2 häufig nicht den Median!
- Mediansuche kann auch als Teilproblem auftreten, aber nicht in allen Teilproblemen.

→ Unsere Analyse ist pessimistisch.

Exaktes Ergebnis: $2 \cdot (1 + \ln 2) \cdot n + o(n) \approx 3,39n$.

4.9 Sortieren auf Parallelrechnern

Was hilft es, wenn wir viele Vergleiche gleichzeitig ausführen können?

Damit mehrfacher Zugriff auf Daten zu einem Zeitpunkt ausgeschlossen ist, darf a_i an maximal einem Vergleich zu einem Zeitpunkt beteiligt sein.

Also höchstens $\frac{n}{2}$ Vergleiche gleichzeitig.

⇒ **Allgemeine Sortierverfahren brauchen mind. $2 \log n - o(\log n)$ Zeittakte.**

Alle $O(\log n)$ Sortieralgorithmen in diesem Szenario sind sehr kompliziert und erst für sehr große n sinnvoll.

Praktisch effizienter Algorithmus:

Batchersort.

(Kenneth E. Batchner, 1968)

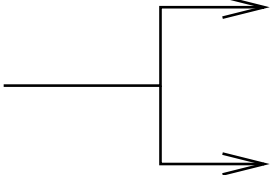
Welcher unserer Sortieralgorithmen ist „parallelisierbar“?

Mergesort.

Hier und im Folgenden alles für $n = 2^k$.

Batchersort, $\text{BS}(a_1, \dots, a_n)$:

- $n = 1$: Nichts zu tun.
- $n > 1$:

Gleichzeitig  $(b_1, \dots, b_{n/2}) := \text{BS}(a_1, \dots, a_{n/2})$
 $(c_1, \dots, c_{n/2}) := \text{BS}(a_{n/2+1}, \dots, a_n)$

$(d_1, \dots, d_n) := \text{BM}(b_1, \dots, b_{n/2}; c_1, \dots, c_{n/2})$.

 **Batchermerge**

$S(n) :=$ Anzahl der wesentliche Vergleiche für Batchersort.

$PS(n) :=$ Anzahl der Zeittakte für Batchersort.

$M(n) :=$ Anzahl der wesentlichen Vergleiche für Batchermmerge und zwei Folgen der Länge n .

$PM(n) :=$ Anzahl der Zeittakte für Batchermmerge und zwei Folgen der Länge n .

$S(1) = 0, PS(1) = 0.$

$M(1) = 1, PM(1) = 1$ (offensichtlich).

$$S(n) = 2 \cdot S(n/2) + M(n/2)$$

$$PS(n) = 1 \cdot PS(n/2) + PM(n/2)$$

 da die beiden rekursiven Aufrufe gleichzeitig stattfinden.

Aber wie können wir das Reißverschlussverfahren für das Mischen zweier sortierter Folgen parallelisieren?

Gar nicht, wir brauchen eine neue Idee.

Betrachte Aufruf

BM($a_1, \dots, a_n; b_1, \dots, b_n$) mit $a_1 \leq \dots \leq a_n$ und $b_1 \leq \dots \leq b_n$:

- Anzahl möglicher Rangplätze für $a_j : n + 1$,
nämlich $j, \dots, 2n - (n - j) = j + n$.
- Wenn $b_k \leq a_j \leq b_{k+2}$ bekannt ist, sind es nur noch
zwei Rangplätze: $j + k, j + k + 1$.
- **Randfälle:**
 - Falls $a_j \leq b_2$, sind es die Rangplätze: $j, j + 1$.
 - Falls $a_j \leq b_1$, ist es Rangplatz j .
 - Analog $b_{n-1} \leq a_j$ und $b_n \leq a_j$.

Wie bisher $n = 2^k$.

Batchermerge, $\text{BM}(a_1, \dots, a_n; b_1, \dots, b_n)$:

– $n = 1$: $z_1 = \min(a_1, b_1)$, $z_2 = \max(a_1, b_1)$, ein Vergleich

– $n > 1$:

Gleichzeitig $\left\{ \begin{array}{l} \rightarrow (v_1, \dots, v_n) = \text{BM}(a_1, a_3, \dots, a_{n-1}; b_1, b_3, \dots, b_{n-1}) \\ \rightarrow (w_1, \dots, w_n) = \text{BM}(a_2, a_4, \dots, a_n; b_2, b_4, \dots, b_n) \end{array} \right.$

Odd - Even - Merge

Gleichzeitig $\left\{ \begin{array}{l} \rightarrow v_2 \leftrightarrow w_1 \\ \rightarrow v_3 \leftrightarrow w_2 \\ \quad \vdots \\ \rightarrow v_{i+1} \leftrightarrow w_i \\ \quad \vdots \\ \rightarrow v_{n-1} \leftrightarrow w_{n-2} \\ \rightarrow v_n \leftrightarrow w_{n-1} \end{array} \right.$

$$z_1 = v_1$$

$$z_2 = \min(v_2, w_1), \quad z_3 = \max(v_2, w_1)$$

$$z_4 = \min(v_3, w_2), \quad z_5 = \max(v_3, w_2)$$

$$\vdots$$

$$z_{2i} = \min(v_{i+1}, w_i), \quad z_{2i+1} = \max(v_{i+1}, w_i)$$

$$\vdots$$

$$z_{2n-2} = \min(v_n, w_{n-1}), \quad z_{2n-1} = \max(v_n, w_{n-1})$$

$$z_{2n} = w_n.$$

Korrektheit:

Erinnerung: $a_1 \leq \dots \leq a_n, b_1 \leq \dots \leq b_n$.

Welche Rangplätze kann v_i haben?

Anfang, v_1 :

v_1 ist das kleinere Element von a_1 und b_1 , also das kleinste aller Elemente. Damit $z_1 = v_1$ korrekt.

Allgemeiner Fall, $v_i, i \geq 2$:

Dann $v_i = a_{2j-1}$ oder $v_i = b_{2j-1}$, o. B. d. A. $v_i = a_{2j-1}$.

Beobachtungen:

- (1) $2j - 2$ a -Daten kleiner als $v_i = a_{2j-1}$.
 - (2) In der v -Folge sind $i - 1$ Daten kleiner als v_i ,
davon $j - 1$ aus der a -Folge (ungerader Index).
- In der v -Folge sind $i - j$ b -Daten kleiner als v_i ,
dies sind $b_1, b_3, \dots, b_{2(i-j)-1}$.
- Wegen Sortierung dann aber sogar alle
 $b_1, b_2, b_3, \dots, b_{2(i-j)-1}$ kleiner als v_i .
- Mindestens $2i - 2j - 1$ b -Daten sind kleiner als v_i
und $2j - 2$ a -Daten sind kleiner als v_i .
- $\text{Rang}(v_i) \geq 2i - 2.$

Beobachtungen:

(3) $n - (2j - 1) = n - 2j + 1$ a -Daten größer als $v_i = a_{2j-1}$.

(4) In der v -Folge sind $n - i$ Daten größer als v_i ,
davon $n/2 - j$ aus der a -Folge.

→ In der v -Folge sind $n/2 - i + j$ b -Daten größer als v_i , dies
sind b_{2k-1} mit $k = i - j + 1, \dots, n/2$, explizit: $b_{2i-2j+1}, \dots, b_{n-1}$.

→ Wegen Sortierung automatisch auch
 $b_{2i-2j+1}, b_{2i-2j+2}, \dots, b_{n-2}, b_{n-1}$ größer als v_i .

→ Mindestens $n - 2i + 2j$ b -Daten sind größer als v_i und
 $n - 2j + 1$ a -Daten sind größer als v_i ,
insgesamt $2n - 2i + 1$ Daten.

→ $\text{Rang}(v_i) \leq 2i - 1.$

Also: $2i - 2 \leq \text{Rang}(v_i) \leq 2i - 1.$

Analog: $2i - 2 \leq \text{Rang}(w_{i-1}) \leq 2i - 1.$

→ $z_{2i-2} = \min(v_i, w_{i-1})$ und $z_{2i-1} = \max(v_i, w_{i-1}).$

→ Batchmerge arbeitet korrekt.



$$M(n) = 2 \cdot M(n/2) + n - 1 \text{ und } M(1) = 1.$$

→ $M(n) = n \log n + 1$ (siehe Analyse von Algo. 1.3.3),

$$PM(n) = 1 \cdot PM(n/2) + 1 \text{ und } PM(1) = 1$$

Gleichzeitige Ausführung der Mischvorgänge
und auch der letzten $n - 1$ Vergleiche.

→ $PM(n) = \log n + 1.$

Zur Erinnerung:

$$S(1) = 0 \text{ und}$$

$$S(n) = 2 \cdot S(n/2) + M(n/2) = 2 \cdot S(n/2) + \frac{n}{2} \log \frac{n}{2} + 1$$

$$\rightarrow S(n) = \frac{1}{4}n \cdot \log n \cdot (\log n - 1) + n - 1$$

(Lässt sich mit Induktionsbeweis verifizieren.)

$$PS(1) = 0 \text{ und}$$

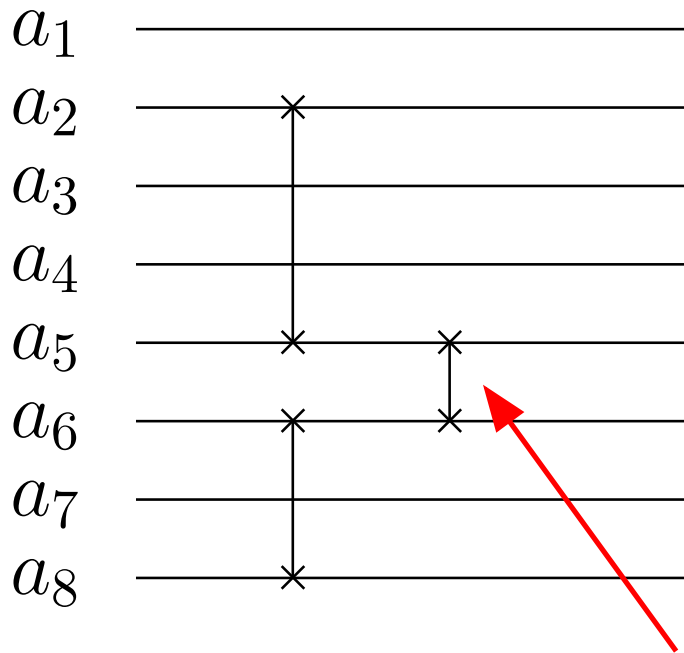
$$\begin{aligned} PS(n) &= PS(n/2) + PM(n/2) = PS(n/2) + \log n \\ &= \log n + \log(n/2) + \log(n/4) + \dots + \log(n/n) \\ &= \log n + (\log n - 1) + (\log n - 2) + \dots + 0 \\ &= \frac{1}{2} \cdot \log n \cdot (\log n + 1). \end{aligned}$$

Hardwaremäßige Realisierung – Sortiernetze:

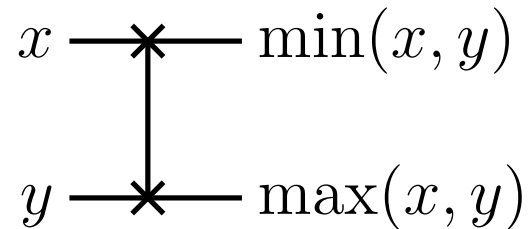
- Prozessoren P_1, \dots, P_n .
- Daten a_1, \dots, a_n eindeutig Prozessoren zugeordnet.
Zu Anfang hat P_i Datum a_i .
- Operationen: Vergleich (i, j) , $i < j$:
 - Prozessoren P_i und P_j vergleichen ihre Daten.
 - P_i erhält Minimum, P_j Maximum.

Sortiernetze (Forts.):

Graphische Darstellung von Algorithmen:

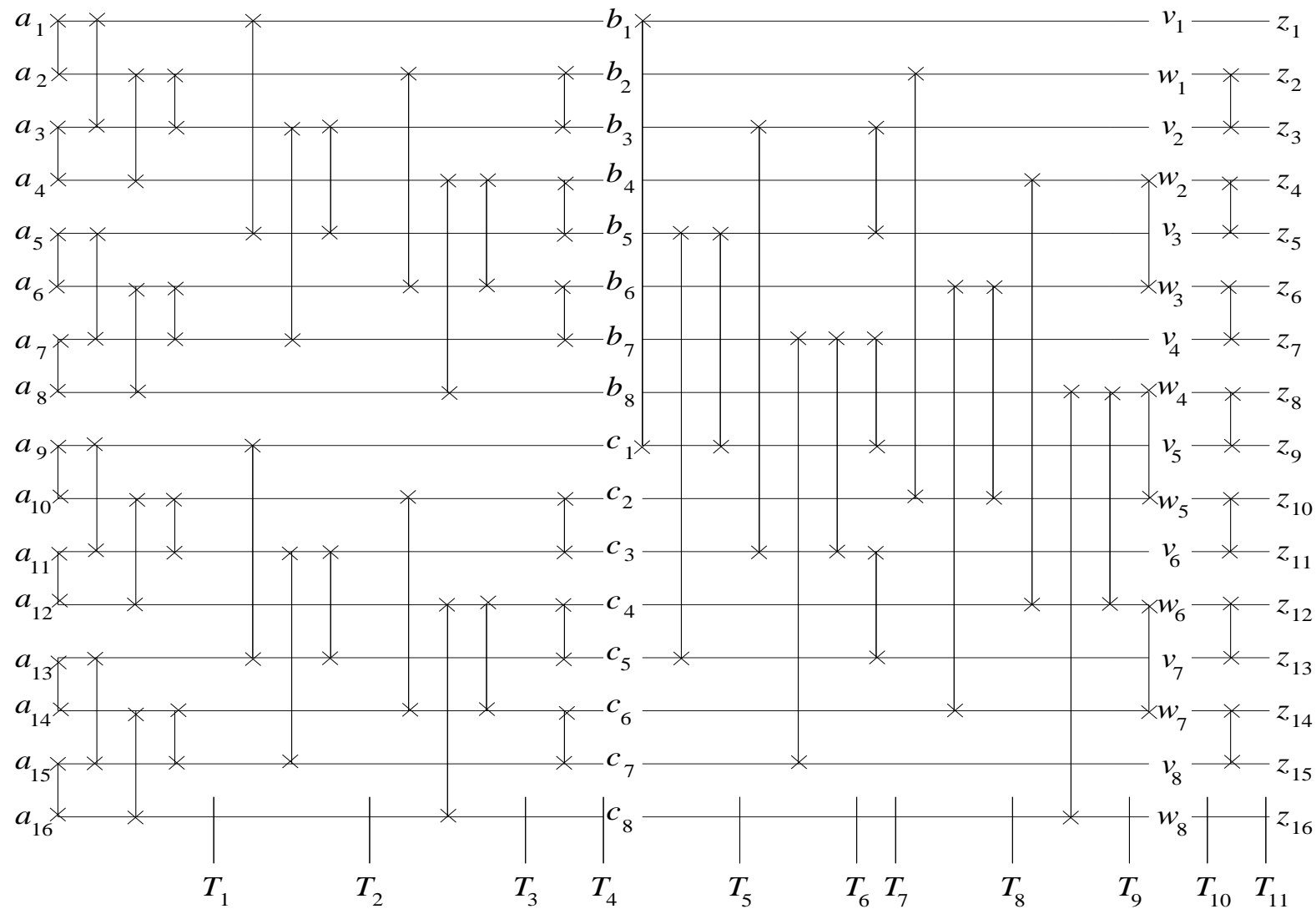


Vergleich (i, j) mit $i < j$
dargestellt durch Kante
zwischen P_i und P_j :



Vergleich von $\max(a_2, a_5)$ mit $\min(a_6, a_8)$.

Das Batcher-Sortiernetz für $n = 16$



P_i kommuniziert nur mit P_{i+2^k} und P_{i-2^k} !