SORTIERVERFAHREN

→ wichtiges Problem

Formal: INPUT: Folge von Datensätzen s_1, s_2, s_n (DS)

Schlüssel → k_1, k_2, k_n

totale Ordnung "$\leq$"

OUTPUT: Permutation $\Pi_i \{1, \dots n\} \to \{1, \dots n\}$

$k_{\Pi}(1) \leq k_{\Pi}(2) \leq \dots \leq k_{\Pi}(n)$

Internes Problem: Alle Daten im HS (appl. speicher)

Feld $A[1], \dots, A[n]$

$A[i].key$, $A[i].info$

Laufzeitmessung:

1) Schlüsselvergleiche

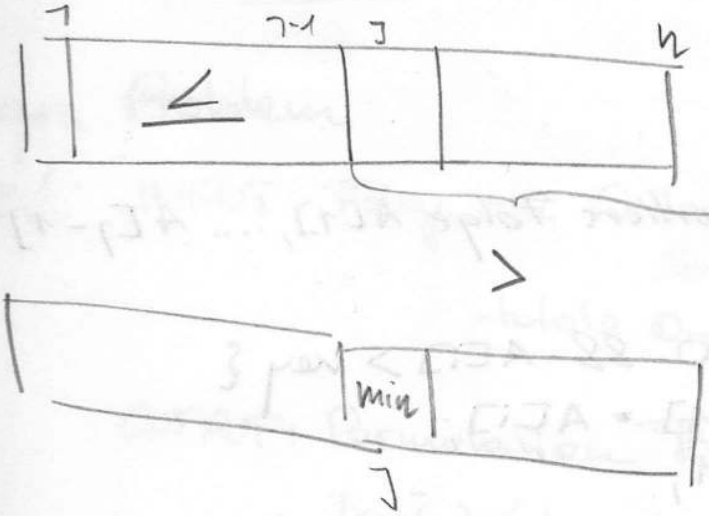
$C_{best}(n)$, $C_{worst}(n)$, $C_{avg}(n)$

2) Datenbewegungen

Bewegungen von DS

$M_{best}(n)$, $M_{worst}(n)$, $M_{Avg}(n)$

Sortieren durch Auswahl (Selection Sort)



j=1	5	2	4	3	①
j=2	1	②	4	3	5
j=3	1	2	4	③	5
j=4	1	2	3	④	5
j=5	1	2	3	4	5

Alg. Selection-Sort (var A)
Eingabe: Folge in Feld A
Ausgabe: sortiere Folge in Feld A
Variablen: Indizes minpos, i, j

```
1  für j = 1, 2, ..., n-1 {
2      // Bestimme Position des Minim. aus A[j] ... A[n]
3      minpos = j;
4      für i = j+1, ..., n {
5          falls A[i].key < A[minpos].key dann {
6              minpos = i;
7          }
8      }
9      falls minpos > j dann {
10         Vertausche A[minpos] mit A[j];
11     }
12 }
```

Schleifen werden immer durchlaufen $\Rightarrow C_W = C_A$

← kopiert wird nur hier

Aufwand für Selection-Sort:

C:

$C_{best}(n) = \Theta(n^2)$

$C_{worst}(n) = \Theta(n^2)$

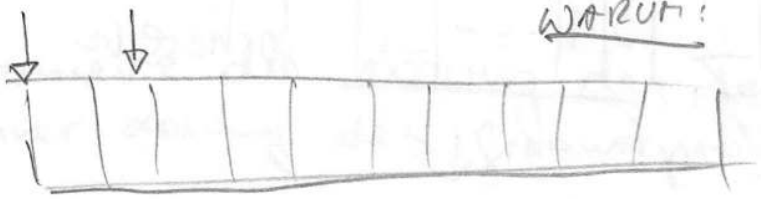
$C_{AVG}(n) = \Theta(n^2)$

M:

$M_{best}(n) = \emptyset$

$M_{worst}(n) = \Theta(n)$

$M_{AVG}(n) = \Theta(n)$ ← WARUM?



$n \quad n$

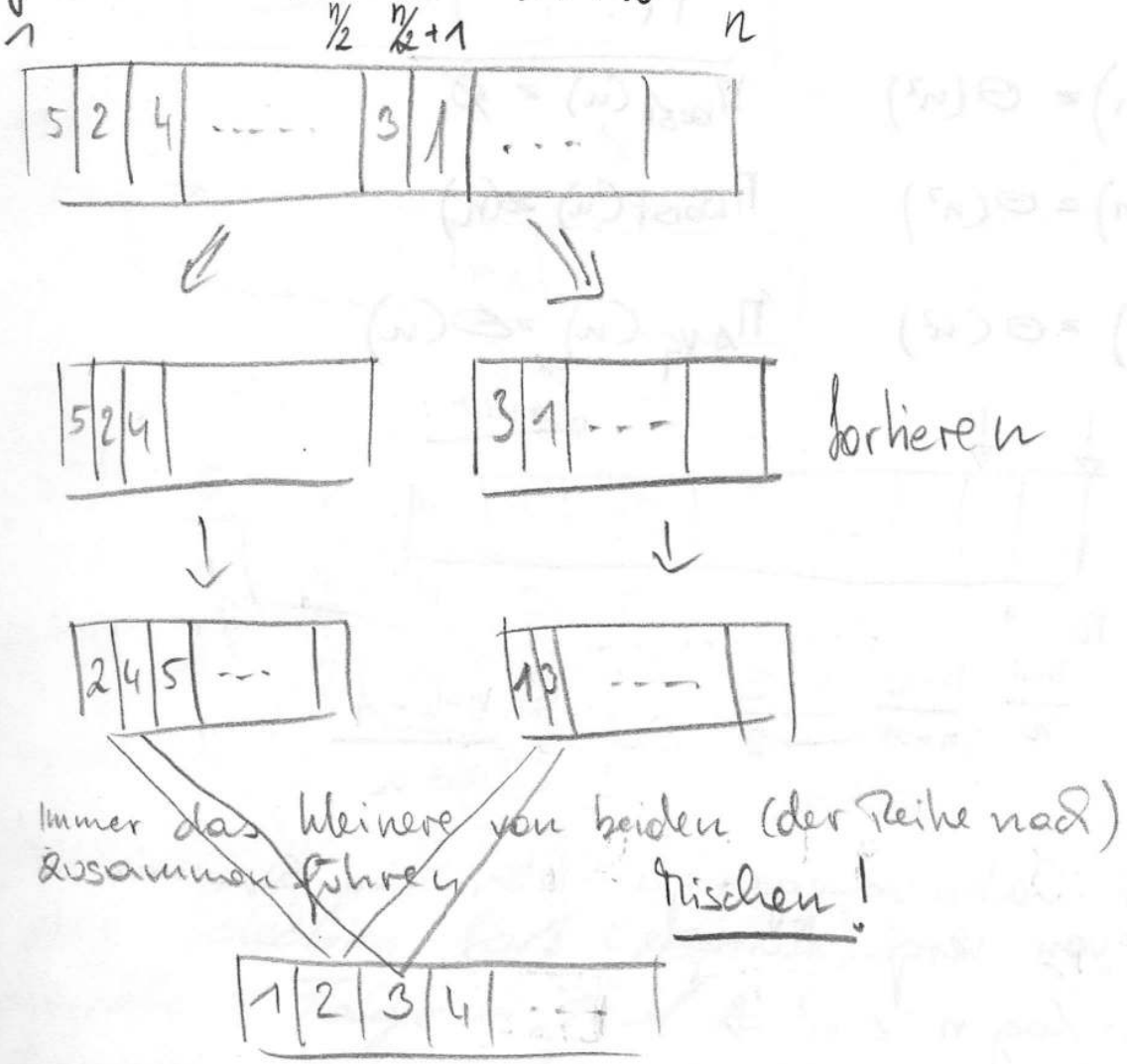
$\frac{n-1}{n} \quad \frac{n-2}{n-1} \quad \frac{n-3}{n-2} \quad \ldots \quad \frac{n-k-1}{n-k}$

Wenige Datenbewegungen über eine fixe Anzahl von Vergleichungen (n^2).

$n < n \cdot \log n < n^2$

MergeSort / Sortieren u. Mischen

J. v. Neumann ca 1945



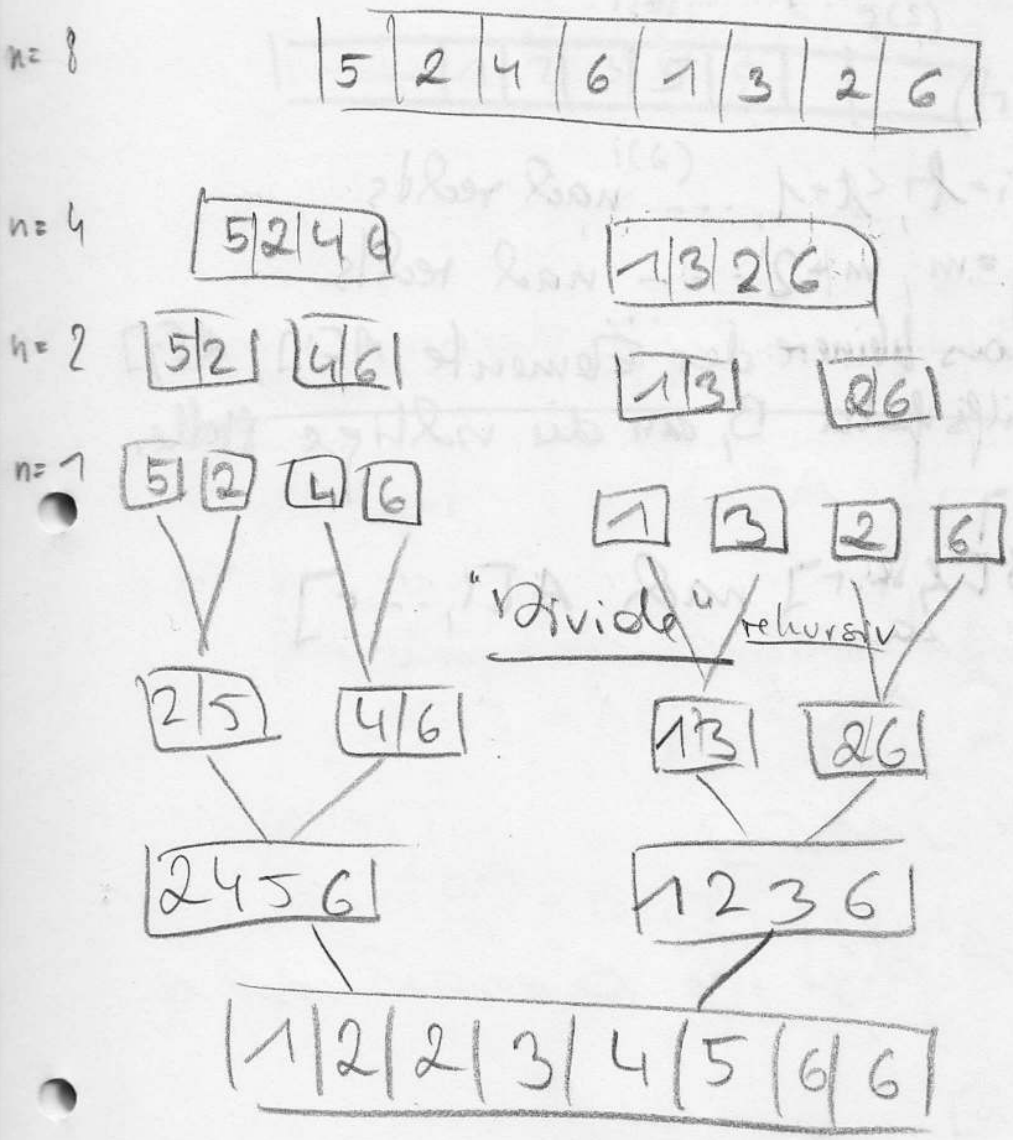
1.) Mischen u. Aufteilen kann mit wenig Aufwand
Mischen ... linear
Bei Zerlegung ... rekursiv vorgehen

Prinzip des Divide and Conquer

↳ Latein!

- Teile ein Problm. in Teilprobleme auf
- Erobere die Teilprobl. durch rekursives Lösen
  Wenn ein Teilproblem klein genug ist, löse es direkt.
- Kombiniere die Lösung der Teilprobleme zu einer Lösung des Gesamtproblem

Gut für sequenziell...



Merge_sort (A, l, r) A[l, ... r]
Bereich der sortiert wird

Eingabe: Folge A
Indexgrenzen l u. r (falls l ≥ r ist nichts zu tun)
Ausgabe: sortiere in A[l] .. A[r]
Variablen: Index m
falls l < r dann {
  m = [(l+r)/2]
  Merge-Sort (A, l, m)
  Merge-Sort (A, m+1, r);
  Merge (A, l, m, r);
}

Merge (var A, l, r)
Idee: gehe von i = l, l+1, ... nach rechts
  – u – j = m, m+2 – – – nach rechts
  kopiere das kleinere der Elemente A[i], A[j]
  in ein Hilfsfeld B, an die richtige Stelle.
  Erhöhe i,j
  kopiere B[l .. r] nach A[l .. r]

MergeSort

```
i = l  // läuft durch A[l] bis A[m] der 1. Teilfolge
j = m+1 // läuft durch A[m+1] bis A[r] der 2. TF
k = l; // das nächste Element der Resultatsfolge ist B[k]

solange (i ≤ m) && (j ≤ r) {
    falls (A[i].key ≤ A[j].key) dann { // beide Teilfolgen nicht erschöpft
        B[k] = A[i]; i = i+1;
    }
    sonst {
        B[k] = A[j]; j = j+1;
    }
    k = k+1
}
falls (i > m) dann {
    // erste Teilfolge erschöpft, übernimm 2.te
    für h = j,...,r {
        B[k+h-j] = A[h];
    }
}
sonst { // 2. Teilfolge erschöpft übernimm 1te
    für h = i...,m [
    ]   B[k+h-i] = A[h];
}
// speichere sortierte Folge von B zurück nach A
für h = l,...,r {
    A[h] = B[h]
}
```

Vergleiche sind hier

Mergesort-Aufwand:

$T(n) = \Theta(n)$

$n = r - l + 1$

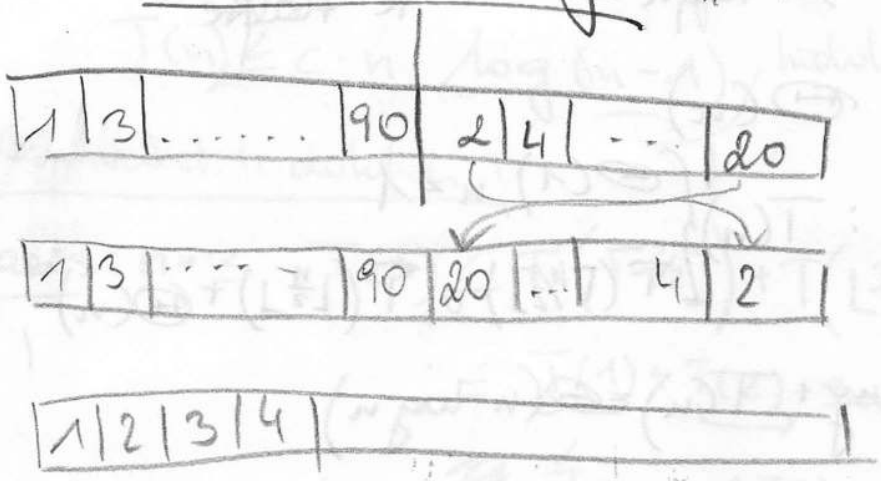
$C(n) = \Theta(n)$

$C_{best}(n) = \min \Theta(m-l, r-m)$

m in der Mitte

$C_{best}(n) = \Theta(n/2)$

Verbessertes Verg // mid Trick




```
B[l,...,m] = A[l,...,m]
B[m+1,...r] = A[r,...,m+1];
p = l; q = r;
für i = l,...,r{
   falls B[p] ≤ B[q] dann {
   }    A[i] = B[p]; p = p+1
   SONST{
        A[i] = B[q]; q = q-1;
   }
}
```

Analyse von Mergesort

Worstcase: Laufzeit $T(n)$

Divide $\Theta(1)$

Conquer $T\left(\lceil \frac{n}{2} \rceil\right)$ li Hälfte und $T\left(\lfloor \frac{n}{2} \rfloor\right)$ re Hälfte

$\lfloor \frac{n+1}{2} \rfloor = \lceil \frac{n}{2} \rceil$

$n - \lfloor \frac{n+1}{2} \rfloor = \lfloor \frac{n}{2} \rfloor$

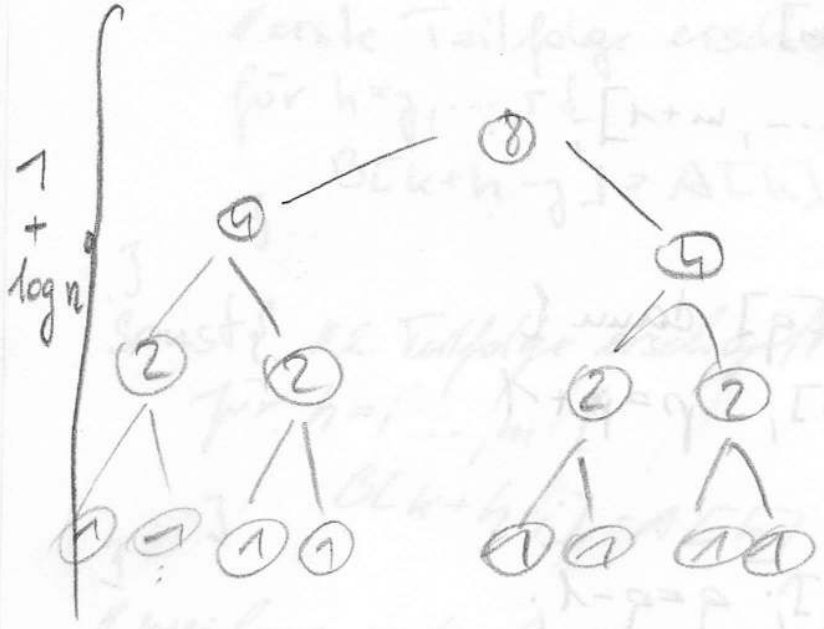
Combine $\Theta(n)$

Rekursion:

$$T(n) \begin{cases} \Theta(1), & n = 1 \\ T(\lceil n/2 \rceil) \text{ u. } T(\lfloor \frac{n}{2} \rfloor) + \Theta(n), & n > 1 \end{cases}$$

Behauptung: $T(n) = \Theta(n \cdot \log n)$

$$n = a^{\log_a n} = b^{\log_b n}$$



Anz Instanz	Zeit/Instanz	Gesamtzeit
$2^0 = 1$	$2^3 = 8$	$2^3 = 8$
$2^1 = 2$	$2^2 = 4$	$2^3 = 8$
$2^2 = 4$	$2^1 = 2$	$2^3 = 8$
$2^3 = 8$	$2^0 = 1$	$2^3 = 8$

Beweis: 1) $T(n) = O(n \log n)$

2) $T(n) = \Omega(n \log n)$

$$T(n) \leq \begin{cases} a, & n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + a \cdot n \end{cases} + f(n) \in \Theta(n)$$

~~Für~~ $n \geq 3$ und $c = 3a$ gilt:

$T(n) \leq c \cdot n \cdot \log(n-1)$ Induktionsbehauptung

Bsp vollst. Induktion:

Basis: $n = 3$ $T(3) \leq T(\lceil 3/2 \rceil) + T(\lfloor 3/2 \rfloor) + 3a$

$\leq T(1) + T(2) + 3a$

$\leq a \quad \leq T(1) + 2a$

$\leq 3T(1) + 5a$

$\leq 8a \overset{a \leq c/3}{=} \frac{8}{3} \cdot c \leq 3 \cdot c =$

$3 \cdot c \cdot \log(3-1)$

$\log(2) = 1$

Induktionsschritt: Ang. Behauptung gilt für alle $n' < n$

$\Rightarrow$ gilt auch für n

Wir müssen zeigen...

$$T(n) \leq c \cdot n \cdot \log(n-1)$$

$$T(n) \leq T(\lceil \tfrac{n}{2} \rceil) + T(\lfloor \tfrac{n}{2} \rfloor) + a \cdot n$$

$$T(n) \leq c \cdot \underbrace{\lceil \tfrac{n}{2} \rceil}_{\leq \frac{n+1}{2}} \cdot \log\Big(\underbrace{\lceil \tfrac{n}{2} \rceil}_{\leq \frac{n+1}{2}} - 1\Big) + c \cdot \underbrace{\lfloor \tfrac{n}{2} \rfloor}_{\leq \frac{n}{2}} \log\Big(\underbrace{\lfloor \tfrac{n}{2} \rfloor}_{\leq \frac{n}{2}} - 1\Big) + an$$

$$T(n) \leq c \cdot \frac{n+1}{2} \cdot \log\left(\frac{n+1}{2} - 1\right) + c \cdot \frac{n}{2} \cdot \log\left(\frac{n}{2} - 1\right) + an$$

$$\leq c \cdot \frac{n+1}{2} \cdot \log\left(\frac{n-1}{2}\right) + c \frac{n}{2} \log\left(\frac{n-1}{2}\right) + an$$

haben wir grobes gemacht

$$\leq c \cdot \frac{n}{2} \cdot 2 \log\left(\frac{n-1}{2}\right) + \frac{c}{2} \underbrace{\log\left(\frac{n-1}{2}\right)}_{\leq n} + an$$

$$\leq cn \log(n-1) - cn + \frac{c}{2} + \underbrace{\frac{c}{3} \cdot n}_{a = \frac{c}{3}}$$

$$\leq c \cdot n \cdot \log(n-1) - cn\left(1 - \frac{1}{2} - \frac{1}{3}\right)$$

$$\leq c \cdot n \cdot \log(n-1)$$

$T(n) = \Omega(n \cdot \log n)$ $\quad T_n = \Theta(n \cdot \log n)$

Was zu beweisen ist...

$T_{best}(n) = T_{worst}(n) = \Theta(n \log n)$

- $T_{avg}(n) = \Theta(n \cdot \log n)$

Allg.: Wenn $T_{worst}(n) = O(f(n))$ nach oben beschränkt ist
u. $T_{best}(n) = \Omega(f(n))$
$\Rightarrow T_{worst}(n) = T_{best}(n) = T_{avg}(n) = \Theta(f(n))$

- **In Bits:** Problem wird auf der Stelle gelöst

Aufgabe: 1.000.000 Zahlen zu sortieren

	Algor	Implementierung	Geschw.
Supercomp	Insert-Sort	$2n^2$	1000 Mio Ops/s
PC	Merge-Sort	$50 n \log_2 n$	10 Mio Ops/s

- Supercomputer: $\frac{2 \cdot 10^6)^2 \text{ Ops}}{10^9 \text{ Ops/sec}}$

$= 2000$ sec
≈ 33 min

PC $= \frac{50 \cdot 10^6 \cdot \log_2 \cdot 10^6 \text{ Ops}}{10^7 \text{ Ops/sec}}$

- 100 sec

Quicksort – Laufzeitverh. im WC nicht mehr als kann entarten mit quad. Verhalten
AVG sieht nicht so gut aus

folgt aus Prinzip Div

Teile: Wähle ein Pivotelement $x \in A$ und teile
A in 2 Teilfolgen

- A_1 enthält Elemente $\leq x$
- A_2 $\geq x$

Partition

```
falls l < r dann {
    x = A[r].key
    i quicksort
```