

ALGORITHMEN UND DATENSTRUKTUREN

ÜBUNG 9: SORTIEREN, SUCHEN & KORRIGIEREN

Eric Kunze

`eric.kunze@mailbox.tu-dresden.de`

TU Dresden, 19.12.2019

```
void ((*f[]))()
```

f	-- f
f[]	-- is an array
*f[]	-- of pointers ([] has higher precedence)
(*f[])()	-- to functions
((*f[])())	-- returning pointers
(((*f[])())())	-- to functions
void ((*f[])())();	-- returning void

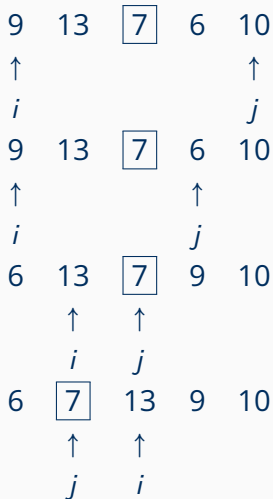
...NICHT ☺

QUICKSORT

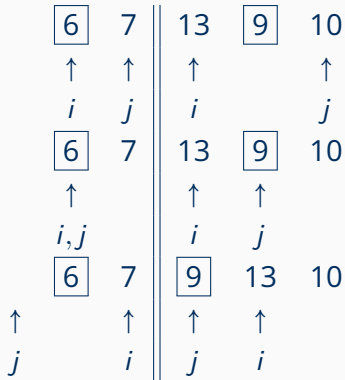
```
1 void quicksort(int a[], int L, int R) {
2     int i, j, w, x, k;
3
4     i = L;  j = R;  k = (L + R) / 2;
5     x = a[k];
6
7     do {
8         while (a[i] < x) i = i + 1;
9         while (a[j] > x) j = j - 1;
10        if (i <= j) {
11            w = a[i];    //
12            a[i] = a[j]; // swap a[i] and a[j]
13            a[j] = w;    //
14            i = i + 1;  j = j - 1;
15        }
16    } while (i <= j);
17
18    if (L < j) quicksort(a, L, j);
19    if (R > i) quicksort(a, i, R);
20 }
```

AUFGABE 1

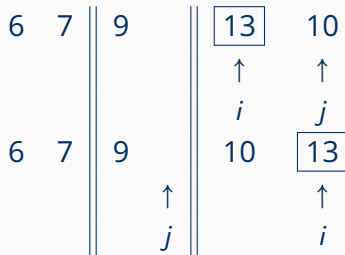
1. Durchlauf



2. Durchlauf



3. Durchlauf

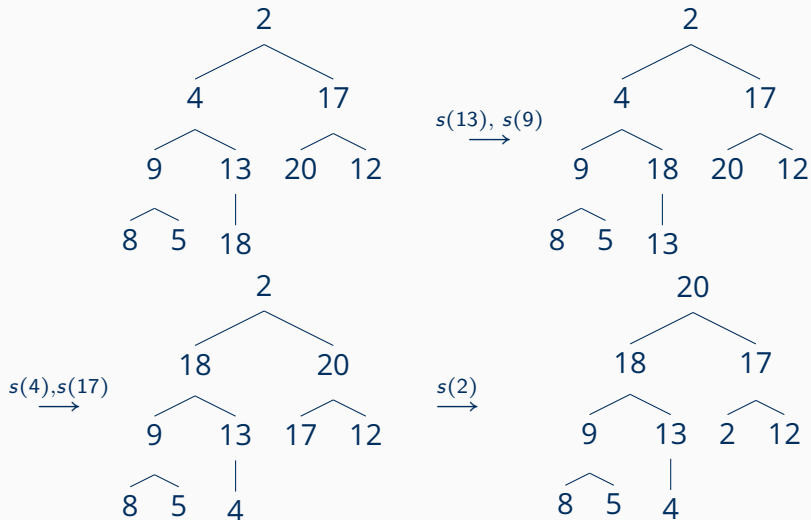


Ergebnis:

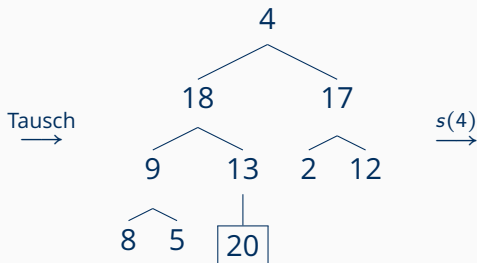
[6,7,9,10,13]

- ▶ Bäume sind nur Veranschaulichung
- ▶ Algorithmus arbeitet auf Listen
- ▶ zwei Phasen
 - ▷ **1. Phase:** Einsortieren in den Heap und Herstellen der Heap-Eigenschaft
 - ▷ **2. Phase:** Führe Sortierschritt wiederholt durch:
 - Tausch von Wurzel und "letztem" Element (tiefste Ebene, ganz rechts)
 - Fixiere dieses Element
 - Sinkenlassen des neuen Wurzelements

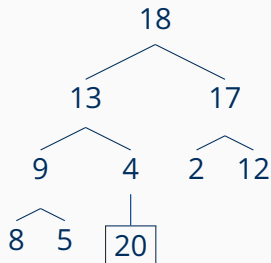
AUFGABE 2 — PHASE 1



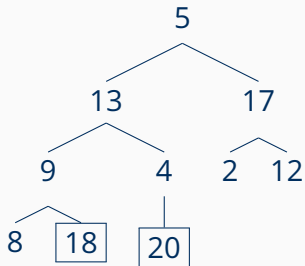
AUFGABE 2 — PHASE 2



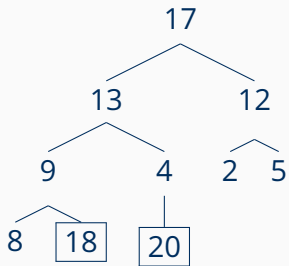
$s(4)$
→



Tausch
→



$s(5)$
→



- ▶ Mustersuche in (großen) Texten
- ▶ Ziel: Verschiebung des Musters um mehr als eine Position bei Nichtübereinstimmung.
- ▶ Methode: Ermittlung einer Verschiebetabelle $\text{Tab}[]$
- ▶ Bedeutung des Eintrags $\text{Tab}[i]=j$:
Bei Nichtübereinstimmung an Stelle i wird Position j des Musters an aktueller Vergleichsstelle angelegt.

KMP-ALGORITHMUS

Suche das Muster aaabaaaa im Text aaabaaabaaacaaabaaaa.

Position	0	1	2	3	4	5	6	7
Pattern	a	a	a	b	a	a	a	a
Tabelle	-1	-1	-1	2	-1	-1	-1	3

Erster Versuch:

```
aaabaaaabaaacaaabaaaa
aaabaaaaa
```

Tabelleneintrag an Position 7 ist 3, d.h. $\text{Tab}[7] = 3$ — Lege Position 3 des Musters an aktueller Vergleichsposition an:

```
aaabaaabaaaacaaabaaaa
aaabaaaaa
```

Erneute Verschiebung liefert schließlich:

```
aaabaaabaaacaaabaaaa
aaabaaaaa
```

Zwei Phasen:

- ▶ **1. Phase:** Markieren der längsten Teilwörter im Pattern, die mit einem Präfix übereinstimmen
 - ▷ ein Zyklus beginnt an einer Patternposition i falls $i \neq 0$ und $\text{Pat}[0] = \text{Pat}[i]$
 - ▷ ein Zyklus endet an der kleinsten Patternposition $i+m$, sodass $\text{Pat}[m+1] \neq \text{Pat}[i+m+1]$
- ▶ **2. Phase:** Bestimmung der Tabelleneinträge
 - ▷ $\text{Tab}[0] = -1$
 - ▷ Tabelleneinträge nach einem Zyklus:
Länge des längsten dort endenden Zyklus
 - ▷ Tabelleneinträgen in einem Zyklus:
Tabelleneintrag der derzeitigen Position im längsten laufenden Zyklus
 - ▷ verbleibende Einträge: 0

AUFGABE 3

Teil (a)

Pattern: aabaaacaab

Position	0	1	2	3	4	5	6	7	8	9
Pattern	a	a	b	a	a	a	c	a	a	b
Tabelle	-1	-1	1	-1	-1	2	2	-1	-1	1

Teil (b)

Position	0	1	2	3	4	5
Pattern	c	b	c	c	b	a
Tabelle	-1	0	-1	1	0	2

LEVENSHTEIN-DISTANZ

Kosten zur Überführung eines Wortes $w = w_1 \dots w_n$ in ein Wort $v = v_1 \dots v_k$; schreibe $d(w_1 \dots w_j, v_1 \dots v_i) = d(j, i)$.

$$d(0, i) = i$$

$$d(j, 0) = j$$

$$d(j, i) = \min \{ d(j, i-1) + 1, d(j-1, i) + 1, d(j-1, i-1) + \delta_{j,i} \}$$

für alle $1 \leq j \leq n$ und alle $1 \leq i \leq k$ wobei

$$\delta_{j,i} = \begin{cases} 1 & \text{wenn } w_j \neq v_i \\ 0 & \text{sonst} \end{cases}$$

Anschaulich: Überlagerung durch Pattern → Pfeile zeigen "Ursprung" des Minimums an

$w_j \neq v_i :$

+1	+1
+1	?

$w_j = v_i :$

+0	+1
+1	?

AUFGABE 4

$d(j, i)$		D	i	s	t	a	n	z
	0	→ 1	→ 2	→ 3	→ 4	→ 5	→ 6	→ 7
D	↓ ↘ 1	0	→ 1	→ 2	→ 3	→ 4	→ 5	→ 6
i	↓ ↘ 2	↓ ↘ 1	0	→ 1	→ 2	→ 3	→ 4	→ 5
n	↓ ↘ 3	↓ ↘ 2	↓ ↘ 1	1	→ 2	→ 3	↘ 3	→ 4
s	↓ ↘ 4	↓ ↘ 3	↓ ↘ 2	↓ ↘ 1	→ 2	→ 3	↘ 4	↘ 4
t	↓ ↘ 5	↓ ↘ 4	↓ ↘ 3	↓ ↘ 2	1	→ 2	→ 3	→ 4
a	↓ ↘ 6	↓ ↘ 5	↓ ↘ 4	↓ ↘ 3	↓ ↘ 2	1	→ 2	→ 3
s	↓ ↘ 7	↓ ↘ 6	↓ ↘ 5	↓ ↘ 4	↓ ↘ 3	↓ ↘ 2	↘ 2	↘ 3

AUFGABE 4

Alignments mit minimaler Levenshtein-Distanz:

D	i	n	s	t	a	*	s
D	i	*	s	t	a	n	z
		<i>d</i>				<i>i</i>	<i>s</i>

D	i	n	s	t	a	s	*
D	i	*	s	t	a	n	z
		<i>d</i>				<i>s</i>	<i>i</i>