

ALGORITHMEN UND DATENSTRUKTUREN

ÜBUNG 10: SUCHEN & KORRIGIEREN, AVL-BÄUME

Eric Kunze

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

TU Dresden, 09.01.2020

Die Methode beruht auf der Gleichung

$$\begin{aligned} & \text{Tab}[j] \\ &= \max \{-1\} \cup \{m \mid 0 \leq m \leq j-1 \wedge b_0 \dots b_{m-j} = b_{j-m} \wedge b_{j-1} \wedge b_m \neq b_j\} \end{aligned}$$

Daraus ergibt sich nach Initialisierung von $\text{Tab}[0] = -1$ für jeden folgenden Eintrag $\text{Tab}[i]$ folgendes Verfahren:

- ▶ wähle $m < i$ in absteigender Reihenfolge (also $i-1, i-2, \dots$), sodass $\text{Pat}[i] \neq \text{Pat}[m]$
- ▶ wenn $\text{Pat}[0 \dots m-1] = \text{Pat}[i-m \dots i-1]$, dann fülle $\text{Tab}[i] = m$.
- ▶ wenn keine passende Position m gefunden werden kann, dann fülle $\text{Tab}[i] = -1$.

AUFGABE 1

Teil (a)

Pattern: abbabbaa

AUFGABE 1

Teil (a)

Pattern: abbabbaa

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

AUFGABE 1

Teil (a) Pattern: abbabbaa

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

Teil (b)

Position	0	1	2	3	4	5
Pattern	<i>b</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>c</i>
Tabelle	-1	?	?	0	?	3

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

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

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 \rightarrow Pfeile zeigen "Ursprung" des Minimums an

$w_j \neq v_i :$

+1	+1
+1	?

$w_j = v_i :$

+0	+1
+1	?

AUFGABE 9.4

$d(j, i)$		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 9.4

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>

AUFGABE 2

$d(j, i)$		s	c	h	ü	r	z	e
	0	→ 1	→ 2	→ 3	→ 4	→ 5	→ 6	→ 7
b	↓ ↘ 1	↓ ↘ 1	↓ ↘ 2	↓ ↘ 3	↓ ↘ 4	↓ ↘ 5	↓ ↘ 6	↓ ↘ 7
ü	↓ ↘ 2	↓ ↘ 2	↓ ↘ 2	↓ ↘ 3	↓ ↘ 3	↓ ↘ 4	↓ ↘ 5	↓ ↘ 6
r	↓ ↘ 3	↓ ↘ 3	↓ ↘ 3	↓ ↘ 3	↓ ↘ 4	↓ ↘ 3	↓ ↘ 4	↓ ↘ 5
s	↓ ↘ 4	↓ ↘ 3	↓ ↘ 4	↓ ↘ 4	↓ ↘ 4	↓ ↘ 4	↓ ↘ 4	↓ ↘ 5
t	↓ ↘ 5	↓ ↘ 4	↓ ↘ 4	↓ ↘ 5	↓ ↘ 5	↓ ↘ 5	↓ ↘ 5	↓ ↘ 5
e	↓ ↘ 6	↓ ↘ 5	↓ ↘ 5	↓ ↘ 5	↓ ↘ 6	↓ ↘ 6	↓ ↘ 6	↓ ↘ 5

AUFGABE 2

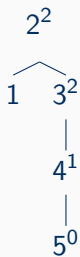
$d(j, i)$		s	c	h	ü	r	z	e
	0	→ 1	→ 2	→ 3	→ 4	→ 5	→ 6	→ 7
b	↓ ↘ 1	↓ ↘ 1	↓ ↘ 2	↓ ↘ 3	↓ ↘ 4	↓ ↘ 5	↓ ↘ 6	↓ ↘ 7
ü	↓ ↘ 2	↓ ↘ 2	↓ ↘ 2	↓ ↘ 3	↓ ↘ 3	↓ ↘ 4	↓ ↘ 5	↓ ↘ 6
r	↓ ↘ 3	↓ ↘ 3	↓ ↘ 3	↓ ↘ 3	↓ ↘ 4	↓ ↘ 3	↓ ↘ 4	↓ ↘ 5
s	↓ ↘ 4	↓ ↘ 3	↓ ↘ 4	↓ ↘ 4	↓ ↘ 4	↓ ↘ 4	↓ ↘ 4	↓ ↘ 5
t	↓ ↘ 5	↓ ↘ 4	↓ ↘ 4	↓ ↘ 5	↓ ↘ 5	↓ ↘ 5	↓ ↘ 5	↓ ↘ 5
e	↓ ↘ 6	↓ ↘ 5	↓ ↘ 5	↓ ↘ 5	↓ ↘ 6	↓ ↘ 6	↓ ↘ 6	↓ ↘ 5

$$d(\text{bürste}, \text{schürze}) = 5$$

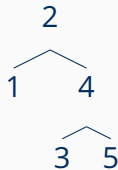
AUFGABE 3



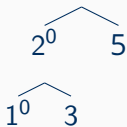
$i(5)$
→



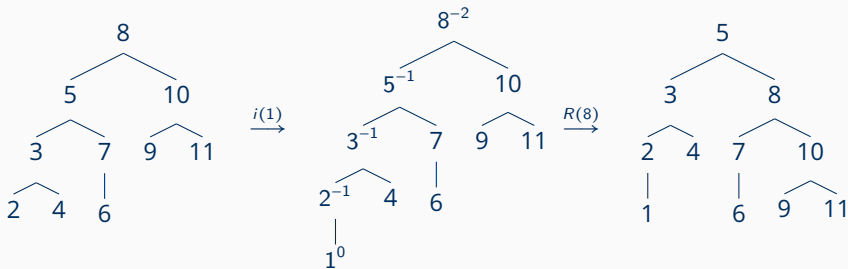
$L(3)$
→



$i(1)$
→



AUFGABE 3



AUFGABE 3

