Dynamische Programmierung Editierdistanz

Dynamisches Programmieren

- Algorithmenentwurfstechnik, oft bei Optimierungsproblemen angewandt.
- Allgemein einsetzbar bei rekursiven Problemlöseverfahren, wenn Teillösungen mehrfach benötigt werden.
- Lösungsansatz: Tabellieren von Teilergebnissen
- Vorteil: Laufzeitverbesserungen, oft polynomiell statt exponentiell

Probleme der Ähnlichkeiten von Zeichenketten

Editier-Distanz

Berechne für zwei gegebene Zeichenfolgen *A* und *B* möglichst effizient die Editier-Distanz *D(A,B)* und eine minimale Folge von Editieroperationen, die *A* in B überführt.

i	n	f	-	-	-	o	r	m	a	t	i	k	-
i	n	t	e	r	p	o	l	-	a	t	i	o	n

Editier-Distanz

Gegeben: Zwei Zeichenketten A = $a_1 a_2 .... a_m$ und $B = b_1 b_2 ... b_n$

Gesucht: Minimale Kosten $D(A,B)$ für eine Folge von Editieroperationen, um A in B zu überführen.

Editieroperationen:

1. Ersetzen eines Zeichens von A durch ein Zeichen von B
2. Löschen eines Zeichens von A
3. Einfügen eines Zeichens von B

i	n	f	-	-	-	o	r	m	a	t	i	k	-
i	n	t	e	r	p	o	l	-	a	t	i	o	n

Editier-Distanz

Einheitskostenmodell:

$$c(a,b) = \begin{cases} 1 & \text{falls } a \neq b \\ 0 & \text{falls } a = b \end{cases}$$

$a = \varepsilon,\ b = \varepsilon$ möglich

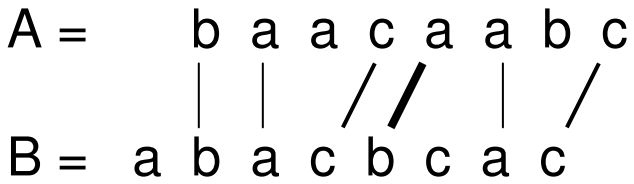
Dreiecksungleichung soll für *c* im allgemeinen gelten:

$$c(a,c) \leq c(a,b) + c(b,c)$$

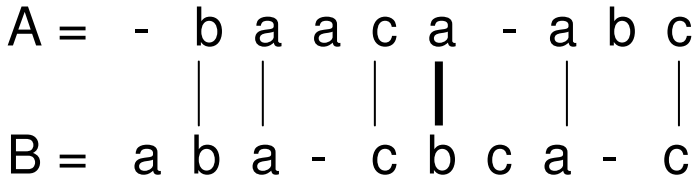
→ ein Buchstabe wird höchstens einmal verändert

Editier-Distanz

Spur als Repräsentation von Editiersequenzen



oder mit Indents



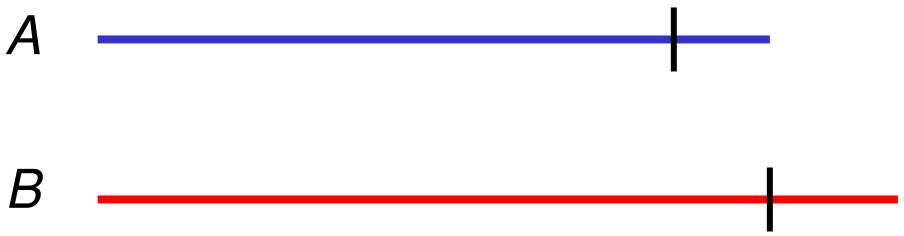
Editier-Distanz (Kosten) : 5

Aufteilung einer optimalen Spur ergibt zwei optimale Teilspuren
→ dynamische Programmierung anwendbar

Berechnung der Editier-Distanz

Sei $A_i = a_1 ... a_i$ und $B_j = b_1 .... b_j$

$$D_{i,j} = D(A_i, B_j)$$



Berechnung der Editier-Distanz

Drei Möglichkeiten, eine Spur zu beenden:

1. a_m wird durch b_n ersetzt:
   $D_{m,n} = D_{m-1,n-1} + c(a_m, b_n)$

2. a_m wird gelöscht: $D_{m,n} = D_{m-1,n} + 1$

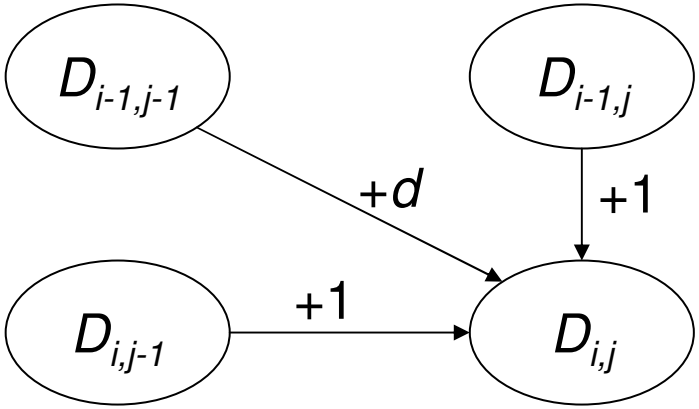
3. b_n wird eingefügt: $D_{m,n} = D_{m,n-1} + 1$

Berechnung der Editier-Distanz

Rekursionsgleichung, falls $m,n \geq 1$:

$$D_{m,n} = \min \begin{Bmatrix} D_{m-1,n-1} & + & c(a_m, b_n) \\ D_{m-1,n} & + & 1 \\ D_{m,n-1} & + & 1 \end{Bmatrix}$$

→ Berechnung aller $D_{i,j}$ erforderlich, $0 \leq i \leq m$, $0 \leq j \leq n$.



Rekursiongleichung für die Editier-Distanz

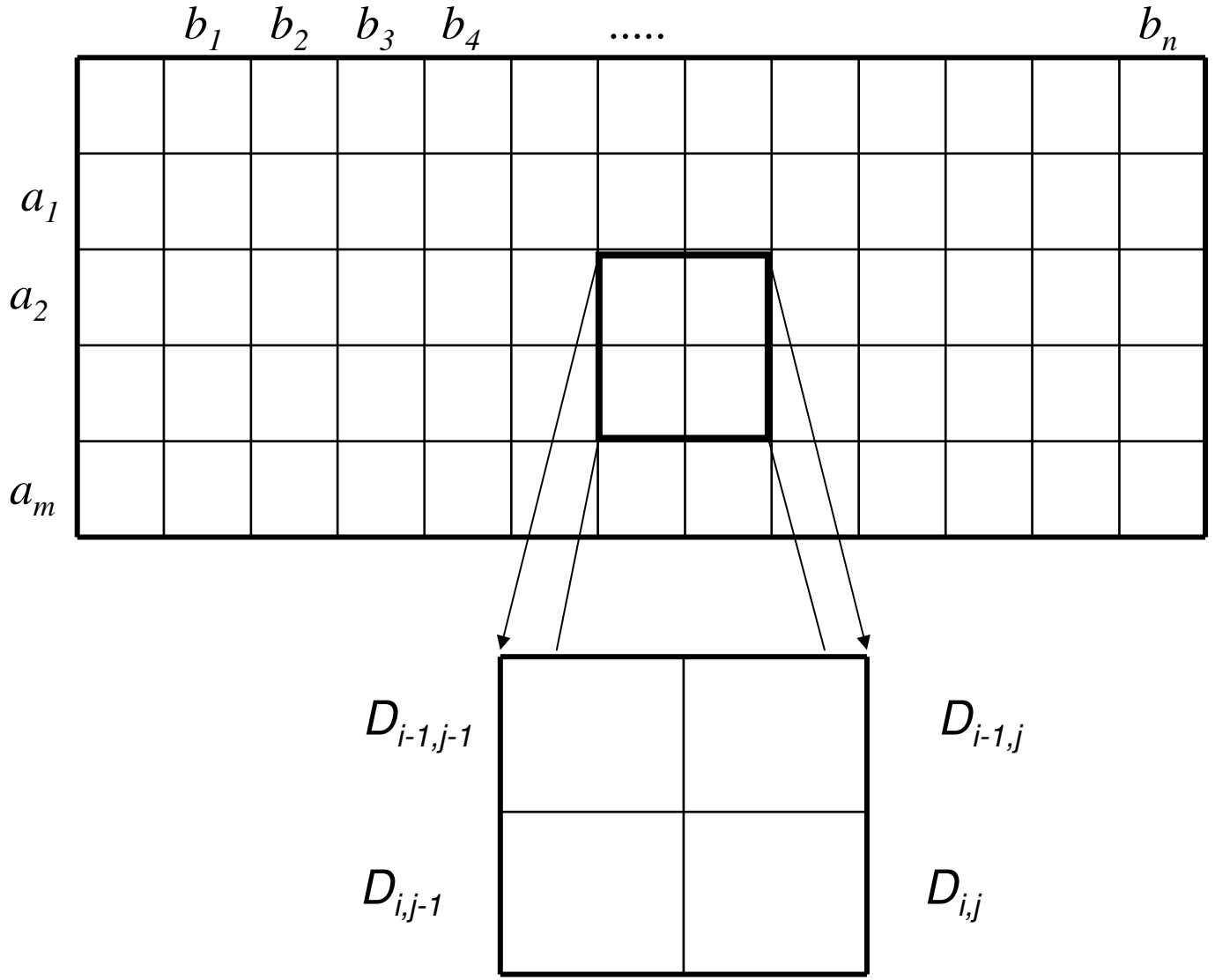
Anfangsbedingungen:

$$D_{0,0} = D(\varepsilon, \varepsilon) = 0$$
$$D_{0,j} = D(\varepsilon, B_j) = j$$
$$D_{i,0} = D(A_i, \varepsilon) = i$$

Rekursionsgleichung:

$$D_{i,j} = \min \begin{Bmatrix} D_{i-1,j-1} & + & c(a_i, b_j) \\ D_{i-1,j} & + & 1 \\ D_{i,j-1} & + & 1 \end{Bmatrix}$$

Berechnungsreihenfolge für die Editier-Distanz



Algorithmus für die Editier-Distanz

Algorithmus Editierdistanz

Input: Zwei Zeichenketten $A = a_1 .... a_m$ und $B = b_1 ... b_n$

Output: Matrix $D = (D_{ij})$

```
1 D[0,0] := 0
2 for i := 1 to m do D[i,0] = i
3 for j := 1 to n do D[0,j] = j
4 for i := 1 to m do
5    for j := 1 to n do
6          D[i,j] := min( D[i - 1,j] + 1,
7                         D[i,j - 1] + 1,
8                         D[i –1, j – 1] + c(a_i ,b_j ))
```

Beispiel für die Editier-Distanz

		a	b	a	c
	0	1	2	3	4
b	1				
a	2				
a	3				
c	4				

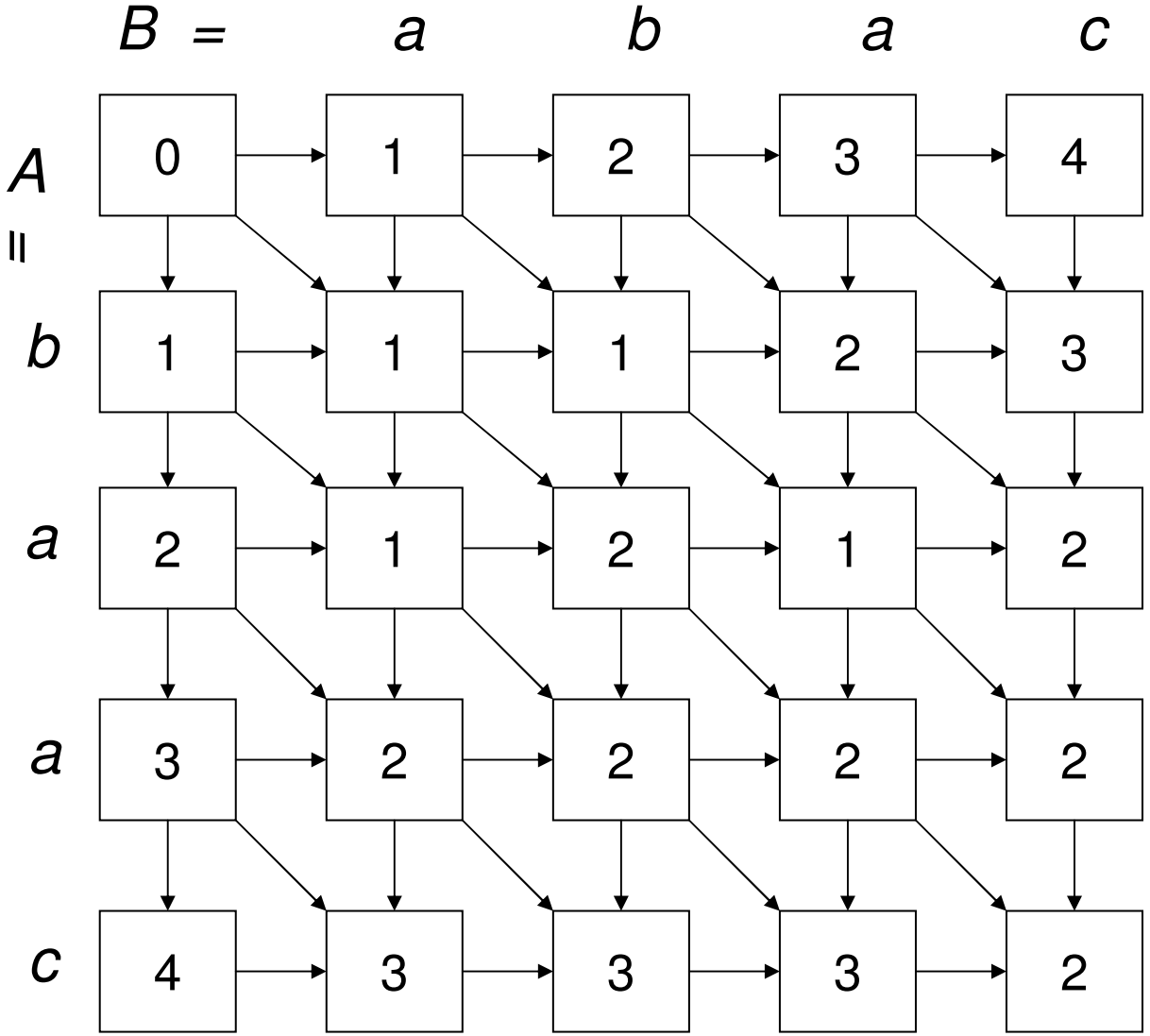
Berechnung der Editieroperation

Algorithmus Editieroperationen (*i,j*)
Input: Berechnete Matrix *D*
Output: Sequenz der Editieroperationen

```
1   if i = 0 and j = 0 then return
2   if i ≠ 0 and D[i,j] = D[i – 1, j] + 1
3     then Editieroperationen (i – 1, j)
4          „lösche a[i]“
5   else if j ≠ 0 and D[i,j] = D[i, j – 1] + 1
6      then Editieroperationen (i,j – 1)
7           „füge b[j] ein“
8   else  /* D[i,j] = D[i – 1, j – 1 ] + c(a[i], b[j])  */
9           Editieroperationen (i – 1, j – 1)
10         „ersetze a[i] durch b[j] “
```

Aufruf: Editieroperationen(*m,n*)

Spurgraph der Editieroperationen



Spurgraph der Editieroperationen

Spurgraph: Übersicht über alle möglichen Spuren zur Transformation von A in B, gerichtete Kanten von Knoten (i, j) zu $(i + 1, j)$, $(i, j + 1)$ und $(i + 1, j + 1)$.
Gewichtung der Kanten entspricht den Editierkosten.

Kosten nehmen entlang eines optimalen Weges monoton zu.

Jeder Weg mit monoton wachsenden Kosten von der linken oberen Ecke zu rechten unteren Ecke entspricht einer optimalen Spur.