Dynamische Programmierung (4)

Editierdistanz
Approximative Zeichenkettensuche
Sequence Alignment

Prof. Dr. S. Albers

Prof. Dr. Th Ottmann

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

Zwei verschiedene Ansätze

Bottom-up:

+ kontrollierte effiziente Tabellenverwaltung, spart Zeit
+ spezielle optimierte Berechnungsreihenfolge, spart Platz
- weitgehende Uncodierung des Originalprogramms erforderlich
- möglicherweise Berechnung nicht benötigter Werte

Top-down: (Memoisierung, Notizblockmethode)

+ Originalprogramm wird nur gering oder nicht verändert
+ Nur tatsächlich benötigte Werte werden berechnet
- separate Tabellenverwaltung benötigt Zeit
- Tabellengröße oft nicht optimal

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

Probleme der Ähnlichkeiten von Zeichenketten

Approximative Zeichenkettensuche

Finde für einen gegebenen Text *T*, ein Muster *P* und eine Distanz *d* alle Teilketten *P´* in *T* mit $D(P,P´) \leq d$

Sequence Alignment

Finde optimale Alignments zwischen DNA-Sequencen

G A G C A - C T T G G A T T C T C G G
- - - C A C G T G G - - - - - - - - -

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

Editier-Distanz

Einheitskostenmodell:

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

$$a = \varepsilon,\ b = \varepsilon \ \text{möglich}$$

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

A = b a a c a a b c

B = a b a c b c a c

oder mit Indels

A = - b a a c a - a b c

B = a b a - c b c a - c

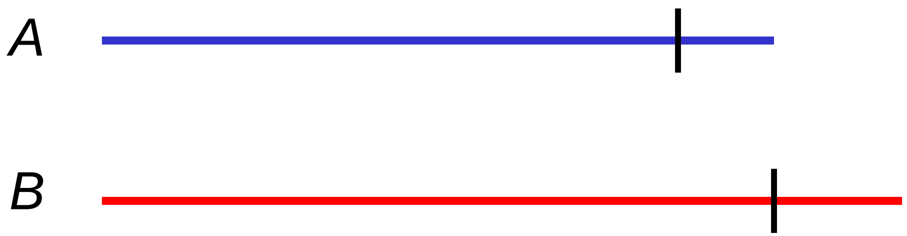
Editier-Distanz (Kosten) : 5

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:
   $D_{m,n} = \mathrm{D}_{\mathrm{m-1,n-1}} + c(a_m, b_n)$

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

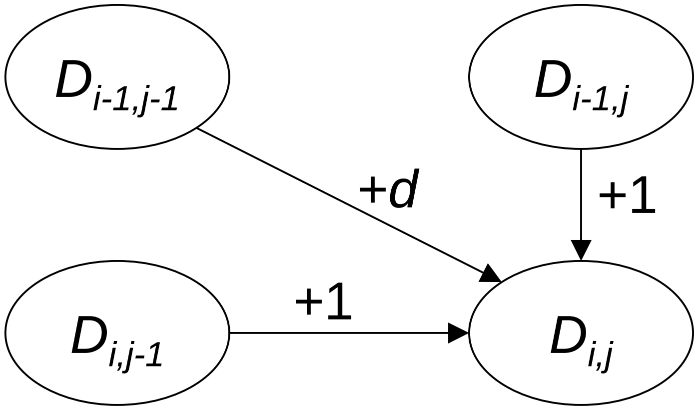
3. b_n wird eingefügt: $D_{m,n} = D_{m,n-1} + 1$

Berechnung der Editier-Distanz

Rekursionsgleichung, falls *m,n* ≥ 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, $0 \le i \le m$, $0 \le j \le n$.



Rekursiongleichung für die Editier-Distanz

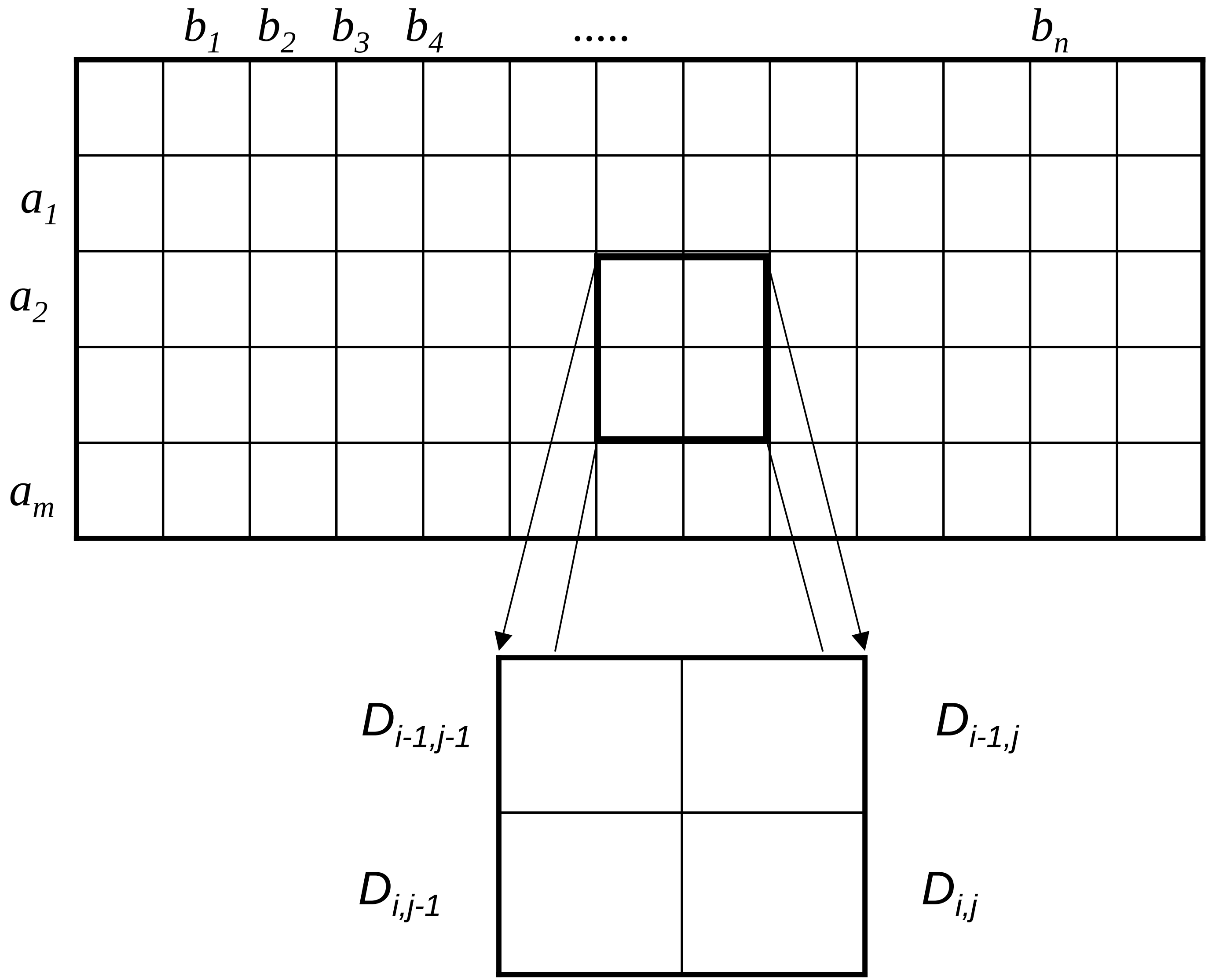
Anfangsbedingungen:

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

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

$D[0,0] := 0$
for $i := 1$ **to** m **do** $D[i,0] = i$
for $j := 1$ **to** n **do** $D[0,j] = j$
for $i := 1$ **to** m **do**
    for $j := 1$ **to** n do
        $D[i,j] := \min(D[i - 1,j] + 1,$
            $D[i,j - 1] + 1,$
            $D[i - 1, j - 1] + c(a_i,b_j))$

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

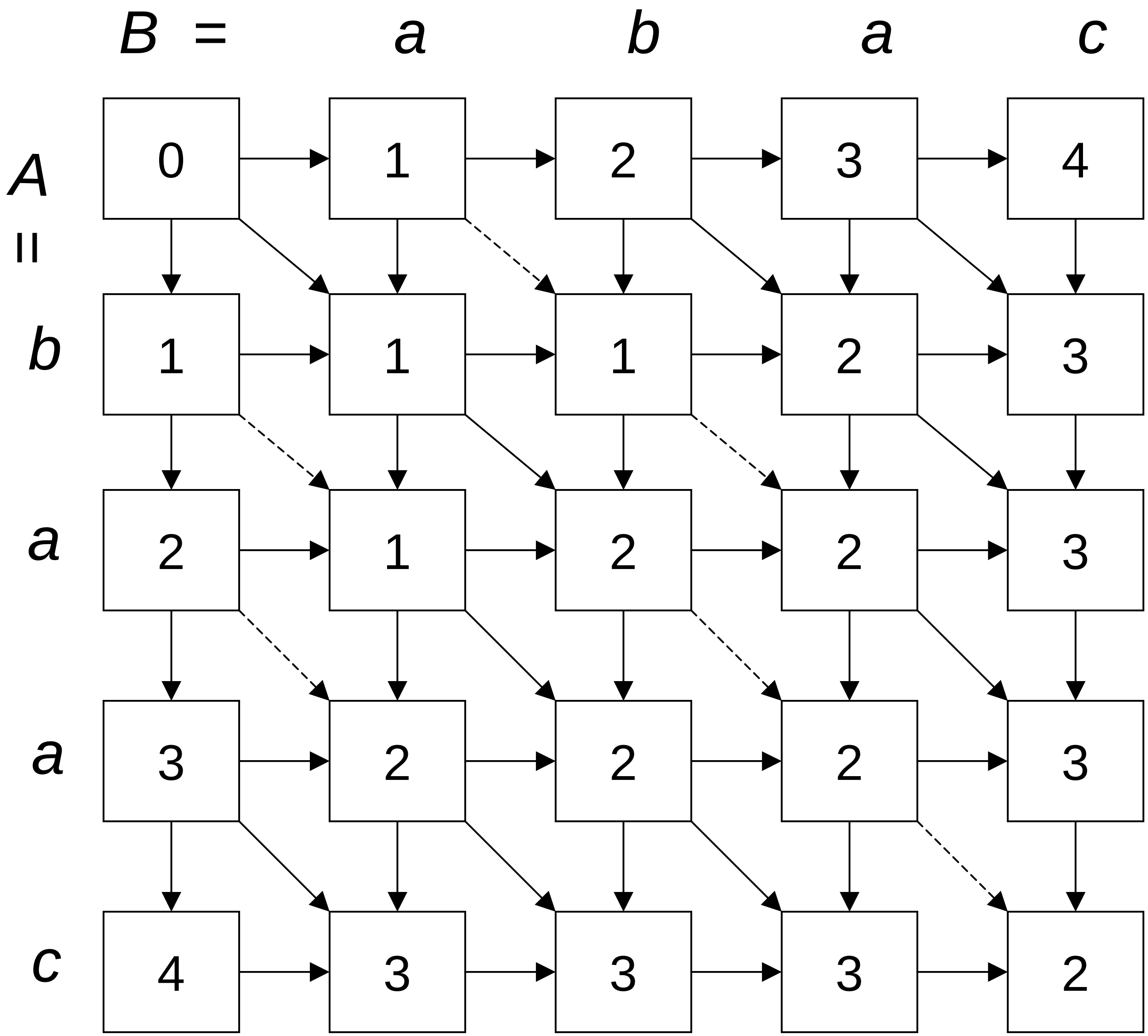
Algorithmus Editieroperationen (*i*,*j*)

Input: Berechnet Matrix *D*

if *i* = 0 **and** j = 0 **then return**
if *i* ≠ 0 and *D*[*i*,*j*] = *D*[*i* – 1 , *j*] + 1
  then „lösche *a*[*i*]“
      Editieroperationen (*i* – 1, *j*)
else if *j* ≠ 0 and *D*[*i*,*j*] = *D*[*i*, *j* – 1] + 1
  then „füge *b*[*j*] ein“
      Editieroperationen (*i*,*j* – 1)
else
  / D[i,j] = D[i – 1, j – 1] + c(a[i], b[j]) */*
      „ersetze *a[i]* durch *b[j]* “
      Editieroperationen (*i* – 1, *j* – 1)

Aufruf: Editieroperationen(*m*,*n*)

Spurgraph der Editieroperationen



Subgraph 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 entsprechen den Editierkosten.

Kosten nehmen entland eines optimalen Weges monoton zu.

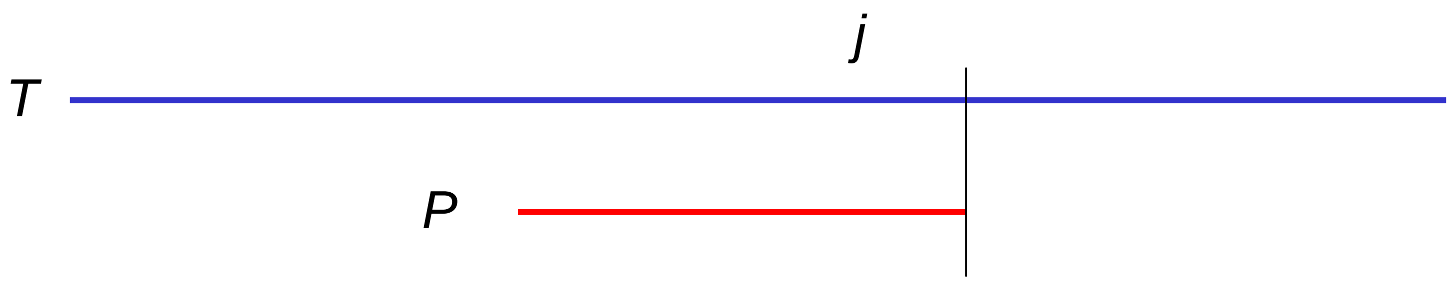
Jeder Weg mit monoton wachsenden Kosten von der linken oberen Ecke zu rechten unteren Ecke entspricht einer optimalen Spur.

Approximative Zeichenkettensuche

Gegeben: zwei Zeichenketten $P = p_1 p_2 \ldots p_m$ (Muster) und
$T = t_1 t_2 \ldots t_n$ (Text)

Gesucht: Ein Intervall $[j', j]$, $1 \leq j' \leq j \leq n$, so dass das Teilwort
$T_{j',j} = t_{j'} \ldots t_j$ das dem Muster P
ähnlichste Teilwort von T ist, d.h. für alle anderen
Intervalle $[k', k]$, $1 \leq k' \leq k \leq n$, gilt:

$$D(P, T_{j',j}) \leq D(P, T_{k',k})$$



Approximative Zeichenkettensuche

Naives Verfahren:

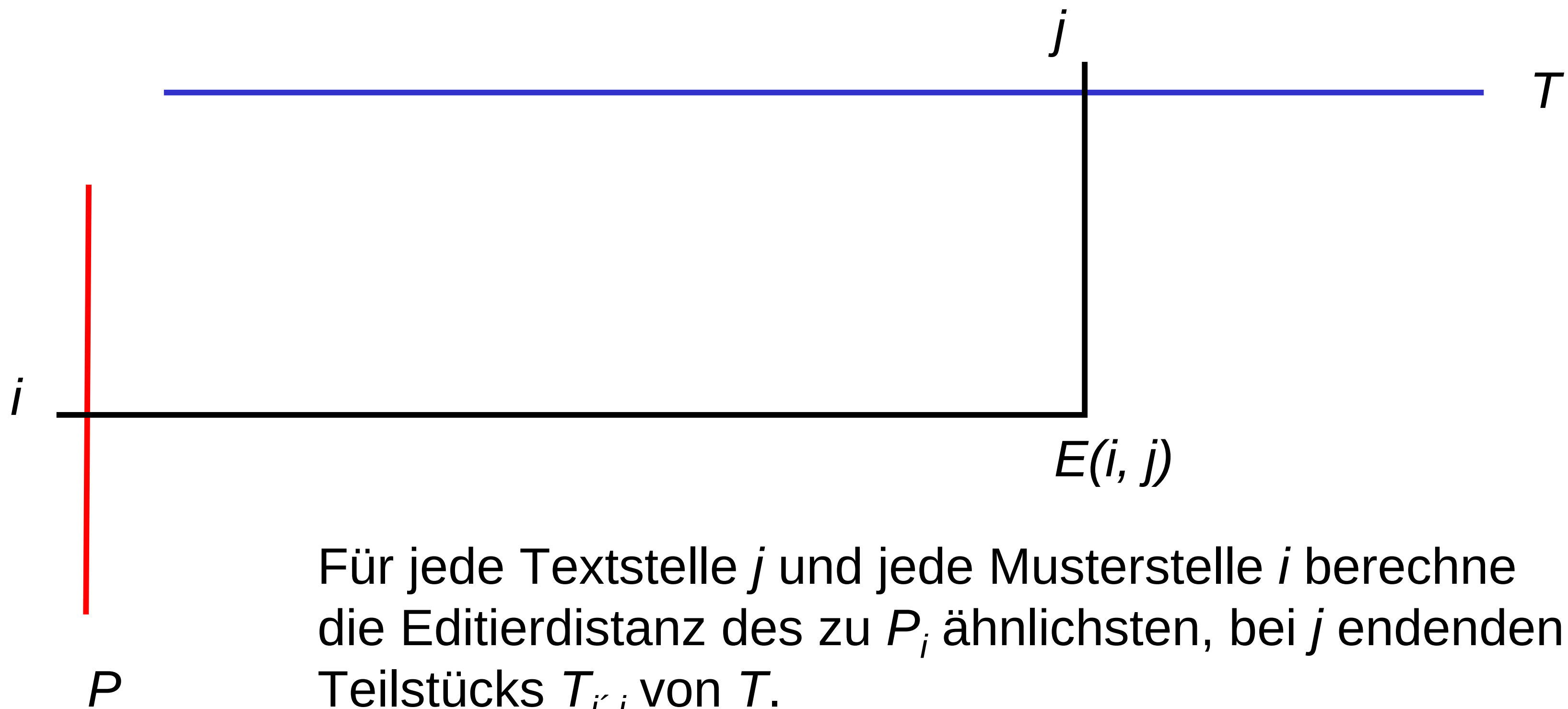
for all $1 \leq j' \leq j \leq n$ **do**

Berechne $D(P, T_{j', j})$

wähle Minimum

Approximative Zeichenkettensuche

Betrachte verwandtes Problem:



Für jede Textstelle j und jede Musterstelle i berechne die Editierdistanz des zu P_i ähnlichsten, bei j endenden Teilstücks $T_{j',j}$ von T.

Approximative Zeichenkettensuche

Methode:

for all $1 \leq j \leq n$ ***do***

Berechne j', so dass $D(P,T_{j',j})$ minimal ist

Für $1 \leq i \leq m$ und $0 \leq j \leq n$ *sei:*

$$E_{i,j} = \min_{1 \leq j' \leq j+1} D(P_i, T_{j',j})$$

Optimale Spur:

$P_i =$ b a a c a a b c

$T_{j',j} =$ b a c b c a c

Approximative Zeichenkettensuche

Rekursionsgleichung:

$$E_{i,j} = \min \begin{Bmatrix} E_{i-1,j-1} + c(p_i, t_j), \\ E_{i-1,j} + 1, \\ E_{i,j-1} + 1 \end{Bmatrix}$$

Bemerkung:

j' kann für $E_{i-1,\,j-1}$, $E_{i-1,j}$ und $E_{i,\,j-1}$ ganz verschieden sein.

Teilspur einer optimalen Spur ist eine optimale Teilspur.

Approximative Zeichenkettensuche

Anfangsbedingungen:

$$E_{0,0} = E(\varepsilon, \varepsilon) = 0$$
$$E_{i,0} = \mathrm{E}(P_j, \varepsilon) = i$$

aber

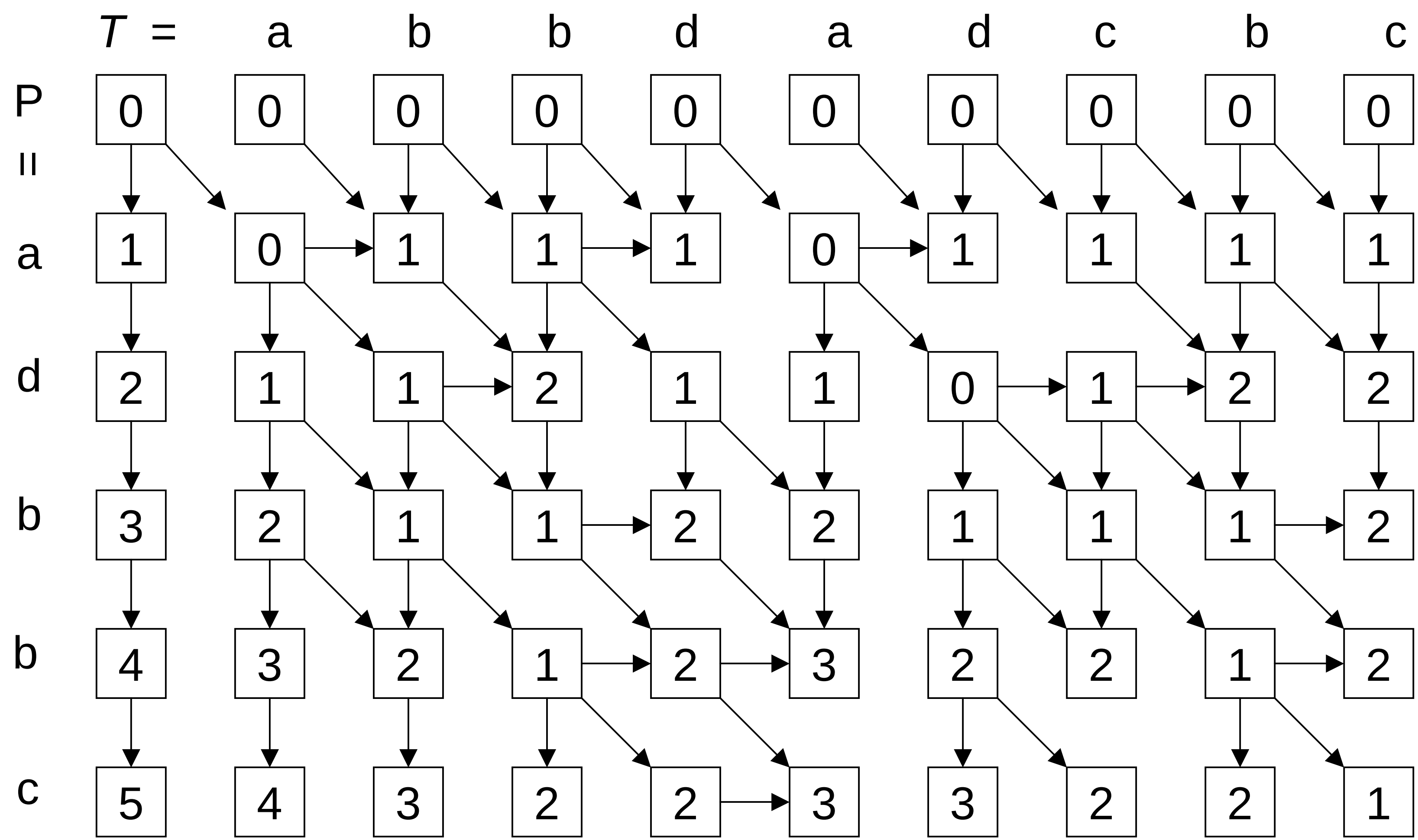
$$E_{0,j} = E(\varepsilon, T_j) = 0$$

Beobachtung:

Die optimale Editiersequenz von P nach $T_{j',j}$ beginnt nicht mit einer Einfügung von $t_{j'}$.

Approximative Zeichenkettensuche

Abhängigkeitsgraph



Approximative Zeichenkettensuche

Satz

Gibt es im Abhängigkeitsgraphen einen Weg von $E_{0, j'-1}$ nach $E_{i, j}$, so ist $T_{j', j}$ ein zu P_i ähnlichstes, bei j endendes Teilstück von T mit

$$D(P_i, T_{j',j}) = E_{i, j}$$

Ähnlichkeit von Zeichenketten

Sequence Aligment:

Finde für zwei DNA – Sequenzen Einfügestellen von Leerzeichen, so dass die Sequenzen möglichst ähnlich sind

```
G A - C G G A T T A G
G A T C G G A A T A G
```

Ähnlichkeit von Zeichenketten

Ähnlichkeitsmaß für Zeichenpaare:

Bsp.	Situation	allgemein
+ 1	für Match	} *s*(a,b)
- 1	für Mismatch	
- 2	FürLeerzeichen	- c

Ähnlichkeitsmaß für Sequencen:

$$S(A,B) = \sum_{alleZeichenpaare} \text{Ähnlichkeit des Zeichenpaars}$$

Ziel: Finde Alignment mit optimaler Ähnlichkeit

Ähnlichkeit von Zeichenketten

Ähnlichkeiten zweier Zeichenketten *S(A,B)*

Operationen:

1. Ersetzen eines Zeichens *a* durch ein Zeichen *b* :
   Gewinn: *s*(*a*,*b*)

2. Löschen eines Zeichens von *A*, Einfügen eines Zeichens von *B*
   Verlust: – *c*

Aufgabe:
Finde eine Folge von Operationen zur Unwandlung von *A* in *B*,
so dass die Summe der Gewinne maximiert wird.

Ähnlichkeit von Zeichenketten

$S_{i,j} = S(A_i, B_j)$, $0 \leq i \leq m$, $0 \leq j \leq n$

Rekursionsgleichung:

$$S_{m,n} = \max \left(S_{m-1,n-1} + s(a_m, b_n),\; S_{m-1,n} - c,\; S_{m,n-1} - c \right)$$

Anfangsbedingung:

$$S_{0,0} = S(\varepsilon, \varepsilon) = 0$$
$$S_{0,j} = S(\varepsilon, B_j) = -jc$$
$$S_{i,0} = S(A_i, \varepsilon) = -ic$$

Ähnlichste Teilzeichenketten

Gegeben: zwei Zeichenketten $A = a_1 .... a_m$ und $B = b_1 ... b_n$

Gesucht: Ein zwei Intervalle $[i', i] \subseteq [1, m]$ und $[j', j] \subseteq [1, n]$,
so dass:

$$\mathrm{S}(A_{i',i} , B_{j',j}) \geq \mathrm{S}(A_{k',k} , B_{l',l}),$$

für alle $[k',k] \subseteq [1, m]$ und $[l',l] \subseteq [1, n]$.

Naives Verfahren:

for all $[i', i] \subseteq [1, m]$ **and** $[j', j] \subseteq [1, n]$ **do**

Berechne $S(A_{i',i} , B_{j',j})$

Ähnlichste Teilzeichenketten

Methode:

for all $1 \leq i \leq m$, $1 \leq j \leq n$ **do**

Berechne $i´$ und $j´$, so dass $S(A_{i´,i}, B_{j´,j})$ maximal ist

Für $0 \leq i \leq m$ und $0 \leq j \leq n$ sei:

$$H_{i,j} = \max_{\substack{1 \leq i' \leq i+1,\\ 1 \leq j' \leq j+1}} S(A_{i',i}, B_{j',j})$$

Optimale Spur:

$$\begin{array}{lccccccccc} A_{i´,i} = & b & a & a & c & a & - & a & b & c \\ & | & | & & | & | & & | & & | \\ B_{j´,j} = & b & a & - & c & b & c & a & - & c \end{array}$$

Ähnlichste Teilzeichenketten

Rekursionsgleichung:

$$H_{i,j} = \max \begin{Bmatrix} H_{i-1,j-1} + s(a_i, b_j), \\ H_{i-1,j} - c, \\ H_{i,j-1} - c, \\ 0 \end{Bmatrix}$$

Anfangsbedingung:

$$H_{0,0} = H(\varepsilon, \varepsilon) = 0$$
$$H_{i,0} = H(A_j, \varepsilon) = 0$$
$$H_{0,j} = H(\varepsilon, B_j) = 0$$