

# Kapitel 5

## Ausrichten

Sucht man in einer Datenbank nach Zeichenketten (Wörtern, Namen, Gensequenzen), möchte man oft anders als in den vorangegangenen Kapiteln unterstellt nicht nur exakte Treffer, sondern auch ähnliche Vorkommen finden. Wir behandeln hier das Basisproblem, die Ähnlichkeit zweier Zeichenketten zu bestimmen, und nehmen dazu Bewertungen ganz bestimmter Formen der Unähnlichkeit vor.

Um Problemstellung und Bewertungen eindeutig zu beschreiben, werden folgende Bezeichnungen verwendet, die in der Vorlesung „Theoretische Grundlagen der Informatik“ (4. Semester) vertieft werden.

- $\Sigma$  ist ein endliches *Alphabet*, d. h. eine Menge von *Symbolen* (auch: *Zeichen*),
- $s = a_1 a_2 \cdots a_n \in \Sigma^* = \bigcup_{k \in \mathbb{N}_0} \Sigma^k$  heißt *Wort* (auch: *Sequenz*) der *Länge*  $|s| = n$ ,
- das *leere Wort*  $\varepsilon$  ist das Wort der Länge Null,
- eine *Teilsequenz*  $s'$  von  $s = a_1 \cdots a_n$  erfüllt  $s' = a_i \cdots a_j$  für  $1 \leq i, j \leq n$  (im Unterschied zu einem *Teilwort*, bei dem auch zwischendrin noch Zeichen ausgelassen werden können)

### 5.1 Beispiel (Alphabete)

Wichtige Alphabete für die symbolische Darstellung von Information im Rechner sind

$\Sigma_{\text{binär}}$	=	$\{0, 1\}$
$\Sigma_{\text{ASCII}}$	enthält 128 Steuerzeichen, Ziffern und Buchstaben	American Standard Code for Information Interchange
$\Sigma_{\text{Unicode}}$	enthält mehr als 100 000 Zeichen (vgl. <a href="http://www.unicode.org">www.unicode.org</a> )	

In Molekularbiologie und Bioinformatik werden z.B. Molekülketten als Folge ihrer wesentlichen Elemente (Basenpaare, Aminosäuren) repräsentiert. Von besonderer Bedeutung sind

$\Sigma_{\text{DNS}}$	=	$\{A, C, G, T\}$	Desoxyribonukleinsäuren
$\Sigma_{\text{RNS}}$	=	$\{A, C, G, U\}$	Ribonukleinsäuren
$\Sigma_{\text{P}}$	=	$\{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$	Proteine (EiweiÙe)

Um zwei Wörter über einem gegebenen Alphabet zu vergleichen, muß festgelegt werden, wie Unterschiede zu behandeln sind. Wir gehen hier davon aus, dass Wörter aus anderen Wörtern durch drei Operationen entstehen können:

**Substitution:** ein Zeichen wird gegen ein anderes ausgetauscht

**Einfügung:** ein Zeichen wird hinzugefügt

**Löschung:** ein Zeichen wird weggelassen

Die Unähnlichkeit zweier Wörter kann dann z.B. als die minimale Anzahl von Operationen dieses Typs, die nötig sind, um ein Wort in das andere zu überführen, definiert werden. Da offensichtlich sinnlose Mehrfachersetzungen ausgeschlossen werden können, ermöglicht die Verwendung eines Platzhalterzeichens -, das nicht aus dem verwendeten Alphabet stammt, die Darstellung einer Folge von Operationen, die aus dem einen Wort das andere macht, durch Untereinanderschreiben. Die beiden Beispiele

ACCTG	-ACCTG
AAC-G	AACG--

$\bar{\Sigma} := \Sigma \cup \{-\}$

illustrieren, dass man die Sequenz AACG aus der Sequenz ACCTG zum Beispiel durch eine Substitution und eine Löschung, oder durch eine Einfügung, eine Substitution und zwei Löschungen erhält.

### 5.2 Definition (Ausrichtung)

Ein Paar  $\bar{s}, \bar{t} \in \bar{\Sigma}^* \times \bar{\Sigma}^*$  heißt Ausrichtung von  $s, t \in \Sigma^*$ , falls

engl.  
alignment

- $\bar{s}|_{\Sigma} = s, \bar{t}|_{\Sigma} = t$  (Weglassen aller Leerzeichen ergibt die Ausgangssequenzen)
- $|\bar{s}| = |\bar{t}|$  (beide Sperrung gleich lang)

Da sie per Definition gleiche Länge haben, können Ausrichtungen stellenweise bewertet werden.

### 5.3 Definition (Bewertung)

Eine Funktion  $\sigma : \bar{\Sigma} \times \bar{\Sigma} \rightarrow \mathbb{R}$  heißt Bewertung. Sie wird auf Ausrichtungen  $\text{score}$  ( $\bar{s} = \bar{a}_1 \cdots \bar{a}_\ell, \bar{t} = \bar{b}_1 \cdots \bar{b}_\ell$ ) erweitert durch

$$\sigma(\bar{a}_1 \cdots \bar{a}_\ell, \bar{b}_1 \cdots \bar{b}_\ell) = \sum_{i=1}^{\ell} \sigma(\bar{a}_i, \bar{b}_i)$$

. Bewertungen  $\sigma$  mit

$$\sigma(a, b) = \begin{cases} 0 & \text{falls } a = b \in \Sigma & (\text{Abgleich, match}) \\ 1 & \text{falls } a \neq b \in \Sigma & (\text{Substitution, mismatch}) \\ 1 & \text{falls } a = -, b \in \Sigma & (\text{Einfügung}) \\ 1 & \text{falls } a \in \Sigma, b = - & (\text{Löschung}) \\ \infty & \text{falls } a = - = b & (\text{unzulässig}) \end{cases}$$

für alle  $a, b \in \bar{\Sigma}$  heißen Editierdistanz.

edit distance

Abhängig vom Kontext kommen verschiedene Bewertungsfunktionen zum Einsatz, in denen entweder eine Ähnlichkeit oder eine Unähnlichkeit, aber immer stellenweise bewertet wird. Für unsere Algorithmen wird es egal sein, wie die konkreten Bewertungen aussehen, sodass wir in diesem Abschnitt der Einfachheit halber immer von der Editierdistanz ausgehen. Das Finden einer Ausrichtung mit optimaler Bewertung  $\sigma$  ist daher ein *Minimierungsproblem*.

Die Zahl der möglichen Ausrichtungen ist beim weitem zu groß, um ein Minimum durch Aufzählen zu suchen. Das folgende Lemma erlaubt es, sich auf deutlich weniger zu beschränken.

ÜBUNG

**5.4 Lemma**

Zu  $s, t \in \Sigma^*$  definiere  $opt(s, t) = \min\{\sigma(\bar{s}, \bar{t}) : (\bar{s}, \bar{t}) \text{ ist Ausrichtung von } s, t\}$ .  
 Dann gilt

$$opt(\underbrace{a_1 \cdots a_n}_{=s}, \underbrace{b_1 \cdots b_m}_{=t}) = \min \left\{ \begin{array}{l} opt(a_1 \cdots a_{n-1}, b_1 \cdots b_{m-1}) + \sigma(a_n, b_m), \\ opt(a_1 \cdots a_n, b_1 \cdots b_{m-1}) + \sigma(-, b_m), \\ opt(a_1 \cdots a_{n-1}, b_1 \cdots b_m) + \sigma(a_n, -) \end{array} \right\}$$

*Beweis.* In einer Ausrichtung gibt es nur drei Kombinationsmöglichkeiten für die beiden Zeichen an der jeweils letzten Stelle, da es sich entweder um Abgleich bzw. Substitution, um eine Einfügung oder um eine Löschung handelt.

$$\begin{array}{l} \bar{s} : \quad \boxed{\bar{a}_1 \quad \cdots \quad \bar{a}_{\ell-1}} \quad \boxed{\bar{a}_\ell} \\ \bar{t} : \quad \boxed{b_1 \quad \cdots \quad b_{\ell-1}} \quad \boxed{b_\ell} \end{array}$$

$\underbrace{\hspace{10em}}$   
 muss optimal sein  
 $\underbrace{\hspace{10em}}$   
 damit optimal sein kann

Wäre in einer optimalen Ausrichtung das Anfangsstück ohne die Zeichen der letzten Stelle nicht optimal, so könnte sie gegen eine andere ausgetauscht und so die Bewertung zusammen mit derjenigen an der letzten Stelle verringert werden. □

Die unmittelbare Umsetzung dieser Beobachtung in einem rekursiven Algorithmus ist in Algorithmus 24 angegeben, führt aber zu inakzeptabler Laufzeit.

**5.5 Satz**

Die Laufzeit von Algorithmus 24 ist in  $\Omega(3^{\min\{n,m\}})$ .

*Beweis.* ... □

Dass der naive Ansatz zu exponentieller Laufzeit führt, liegt an der wiederholten Auswertung für die gleichen Argumente und lässt sich daher leicht vermeiden.

**5.6 Beispiel**

---

**Algorithmus 24:** Optimale Ausrichtung (näiver Ansatz)

---

**Eingabe** : Sequenzen  $s = a_1 \cdots a_n, t = b_1 \cdots b_m \in \Sigma^*$   
 Bewertung  $\sigma : \bar{\Sigma} \times \bar{\Sigma} \rightarrow \mathbb{R}$

**Ausgabe** : Minimale Bewertung  $\text{opt}(s, t)$  einer Ausrichtung von  $s, t$

```

proc D(i, j) begin
  if i = 0 then return  $\sum_{k=1}^j \sigma(-, b_k)$ 
  if j = 0 then return  $\sum_{k=1}^i \sigma(a_k, -)$ 
  return min {
     $D(i - 1, j - 1) + \sigma(a_i, b_j),$ 
     $D(i, j - 1) + \sigma(-, b_j),$ 
     $D(i - 1, j) + \sigma(a_i, -)$ 
  }

```

Aufruf:  $D(n, m)$

---

	t		A	A	C	G			
s	0	→	1	→	2	→	3	→	4
		↓							
A	1	↘	0	→	1	→	2	→	3
		↓							
C	2		1	↘	1	↘	1	→	2
		↓							
C	3		2	↘	2	↘	1	→	2
		↓							
T	4		3	↘	3	↘	2	↘	2
		↓							
G	5		4	↘	4	↘	3	↘	2

AAC-G  
ACCTG

**5.7 Satz**

Algorithmus 25 bestimmt eine optimale Ausrichtung in  $\Theta(nm)$  Zeit mit  $\Theta(nm)$  Platz. Soll nur deren Bewertung bestimmt werden, kann der Platzbedarf auf  $\Theta(\min\{n, m\})$  reduziert werden.

*Beweis.* ... □

Zwei Beobachtungen: Aufteilung, Spiegelung

**5.8 Satz**

Eine optimale Ausrichtung kann in  $\mathcal{O}(nm)$  Zeit mit  $\mathcal{O}(\min\{n, m\})$  Platz

---

**Algorithmus 25:** Optimale Ausrichtung

---

**Eingabe** : Sequenzen  $s = a_1 \cdots a_n, t = b_1 \cdots b_m \in \Sigma^*$   
Bewertung  $\sigma : \bar{\Sigma} \times \bar{\Sigma} \rightarrow \mathbb{R}$

**Ausgabe** : Optimale Ausrichtung von  $s, t$

$D[0, 0] \leftarrow 0$

**for**  $j = 0, \dots, m$  **do**

$D[0, j] \leftarrow D[0, j - 1] + \sigma(-, b_j)$

$\text{pred}[0, j] \leftarrow \text{left}$

**for**  $i = 1, \dots, n$  **do**

$D[i, 0] \leftarrow D[i - 1, 0] + \sigma(a_i, -)$

$\text{pred}[i, 0] \leftarrow \text{above}$

**for**  $j = 1, \dots, m$  **do**

$D[i, j] \leftarrow D[i - 1, j - 1] + \sigma(a_i, b_j)$

$\text{pred}[i, j] \leftarrow \text{diagonal}$

**if**  $D[i, j - 1] + \sigma(-, b_j) < D[i, j]$  **then**

$D[i, j] \leftarrow D[i, j - 1] + \sigma(-, b_j)$

$\text{pred}[i, j] \leftarrow \text{above}$

**if**  $D[i - 1, j] + \sigma(a_i, -) < D[i, j]$  **then**

$D[i, j] \leftarrow D[i - 1, j] + \sigma(a_i, -)$

$\text{pred}[i, j] \leftarrow \text{left}$

$i \leftarrow n; \quad \bar{s} \leftarrow \varepsilon$

$j \leftarrow m; \quad \bar{t} \leftarrow \varepsilon$

**while**  $i + j > 0$  **do**

**if**  $\text{pred}[i, j] = \text{diagonal}$  **then**

$\bar{s} \leftarrow a_i \bar{s}; \quad i \leftarrow i - 1$

$\bar{t} \leftarrow b_j \bar{t}; \quad j \leftarrow j - 1$

**else**

**if**  $\text{pred}[i, j] = \text{left}$  **then**

$\bar{t} \leftarrow b_j \bar{t}; \quad j \leftarrow j - 1$

**else**

$\bar{s} \leftarrow a_i \bar{s}; \quad i \leftarrow i - 1$

**print**  $(\bar{s}, \bar{t}), D[n, m]$ 

---

bestimmt werden.

*Beweis.* Bestimme (mit modifiziertem Alg. 25) ein  $k \in \{0, \dots, m\}$  so, dass Algorithmus  
von Hirschberg

$$\text{opt}(a_1 \cdots a_{\lfloor \frac{n}{2} \rfloor}, b_1 \cdots b_k) + \text{opt}(a_n \cdots a_{\lfloor \frac{n}{2} \rfloor + 1}, b_m \cdots b_{m-k+1})$$

minimal

Berechne rekursiv optimale Ausrichtungen von  $a_1 \cdots a_{\lfloor \frac{n}{2} \rfloor}$  mit  $b_1 \cdots b_{\hat{k}}$  sowie von  $a_n \cdots a_{\lfloor \frac{n}{2} \rfloor + 1}$  mit  $b_m \cdots b_{m-\hat{k}+1}$  und gib als Ergebnis deren Aneinanderreihung zurück □

## 5.9 Beispiel

...

Ausblick: semi-globales und lokales Ausrichten