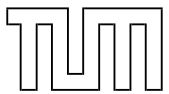
Algorithmische Bioinformatik 1

Dr. Hanjo Täubig

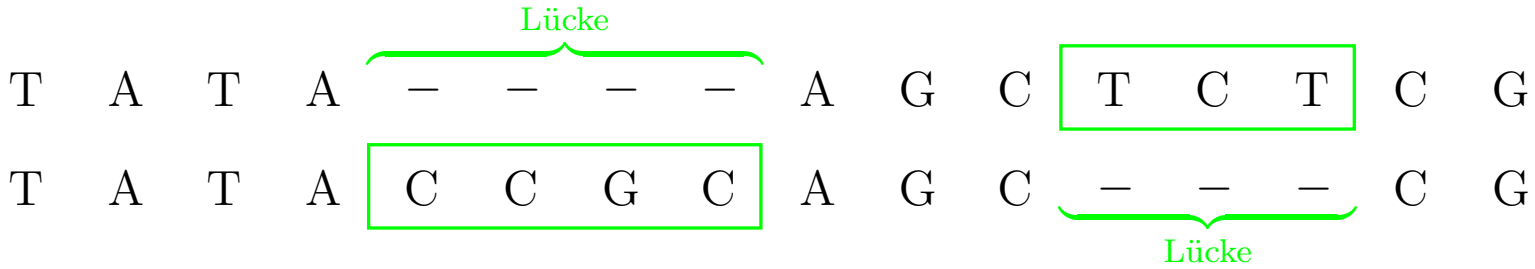
Lehrstuhl für Effiziente Algorithmen
(Prof. Dr. Ernst W. Mayr)
Institut für Informatik
Technische Universität München

Sommersemester 2009



Übersicht

Lückenstrafen



- Manchmal gibt es in Alignments lange Lücken (siehe Abb.)
- Lücken der Länge ℓ mit Kosten von ℓ Insertionen oder Deletionen zu bestrafen, ist nicht unbedingt angebracht (können durch eine einzige Mutation entstanden sein).
- Kurze Lücken sind aber wahrscheinlicher als lange.

$\Rightarrow$ Strafe sollte monoton in der Länge sein.

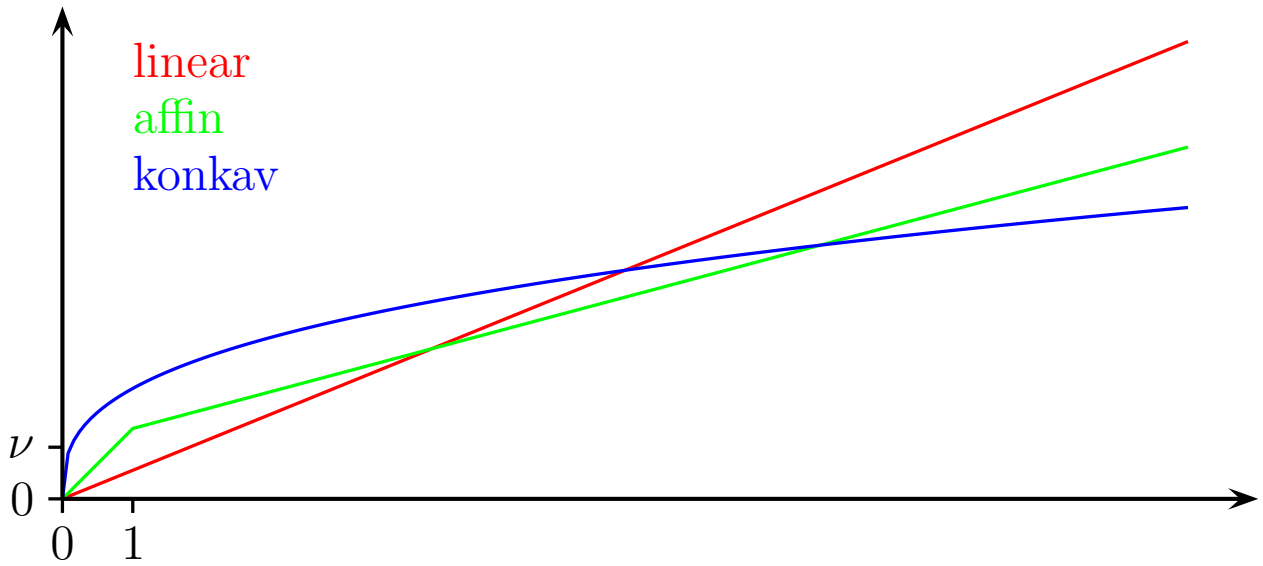
Lückenstrafen

- Lückenstrafe (engl. gap-penalty):

$$\text{Funktion} \quad g : \mathbb{N}_0 \to \mathbb{R}$$

- $g(k)$: Strafe für k konsekutive Insertionen bzw. Deletionen
- Für Distanzmaße ist g immer nichtnegativ und für Ähnlichkeitsmaße nichtpositiv.
- Dabei sollte immer $g(0) = 0$ gelten und $|g| : \mathbb{N}_0 \to \mathbb{R}_+ : k \mapsto |g(k)|$ eine monoton wachsende Funktion sein.
- Annahme: Lückenstrafe g ist sublinear, d.h. $g(k' + k'') \leq g(k') + g(k'')$ für alle $k', k'' \in \mathbb{N}_0$.
- Insertionen und Deletionen werden explizit immer gleich bewertet, unabhängig davon, welche Zeichen gelöscht oder eingefügt werden.

Lückenstrafen



- Abbildung: Funktionen für vernünftige Lückenstrafen
- Lineare Strafen haben wir bereits berücksichtigt, da ja die betrachteten Distanz- und Ähnlichkeitsmaße linear waren.
- Wir betrachten erstmal allgemeine, dann affine und konkave Lückenstrafen.

Allgemeine Lückenstrafen (Waterman-Smith-Beyer)

Rekursionsgleichung für allgemeine Lückenstrafen:

$$D(i,j) = \begin{cases} g(j) & \text{für } i = 0, \\ g(i) & \text{für } j = 0, \\ \min\limits_{k} \left\{ \begin{array}{lcl} D(i-1,j-1) & + & w(s_i, t_j), \\ D(i-k,j) & + & g(k), \\ D(i,j-k) & + & g(k) \end{array} \right\} & \text{für } (i > 0) \wedge (j > 0). \end{cases}$$

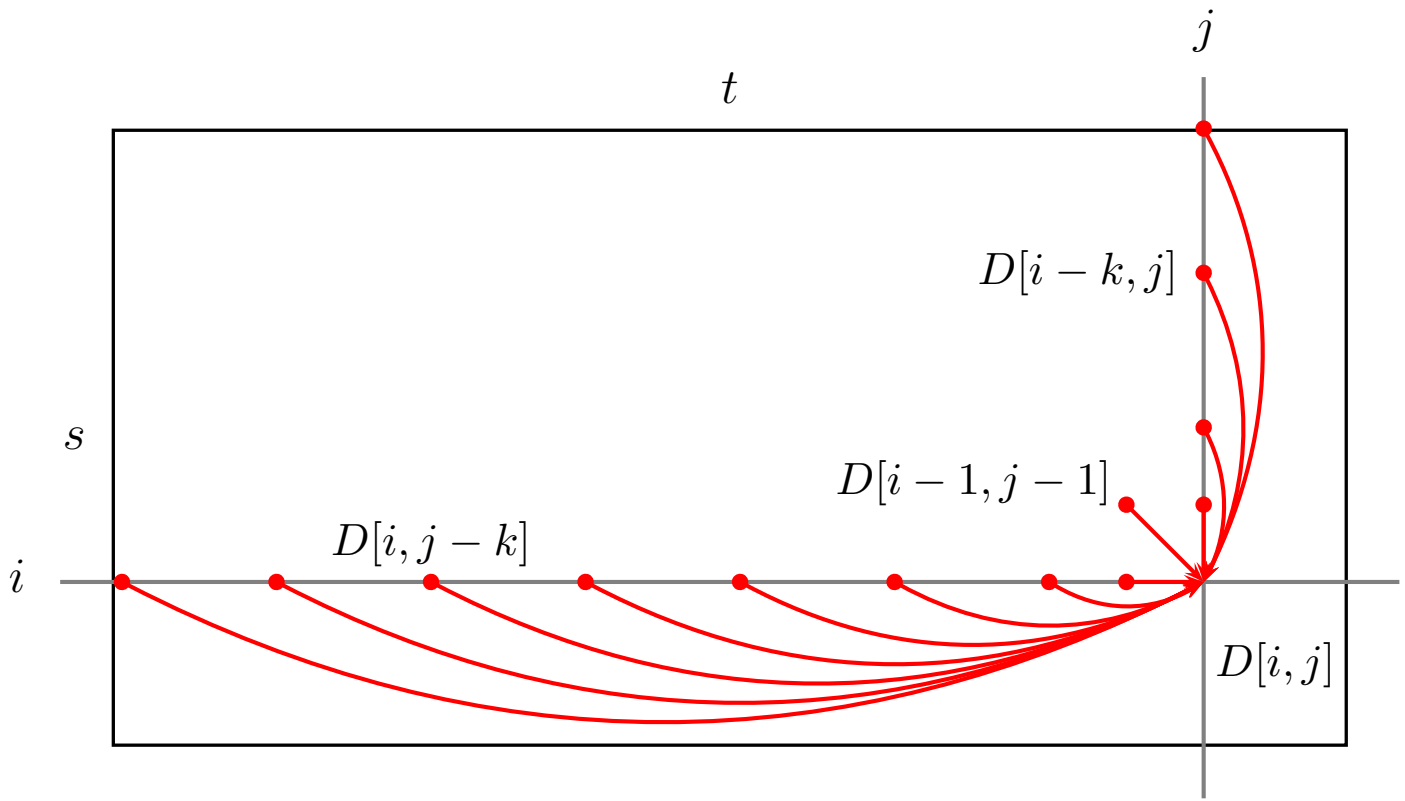
Allgemeine Lückenstrafen (Waterman-Smith-Beyer)

- Bei der Aktualisierung von $D(i,j)$ muss auf alle Werte in derselben Zeile bzw. Spalte bei Insertionen und Deletionen zurückgegriffen werden, da die Kosten der Lücken ja nicht linear sind und somit nur im ganzen und nicht einzeln berechnet werden können.
- Auch zwei unmittelbar aufeinander folgende Lücken werden berücksichtigt.

  Da aber die Strafe von zwei unmittelbar aufeinander folgenden Lücken der Länge k' und k'' größer als die einer Lücke der Länge $k' + k''$ ist, ist dies kein Problem.

  Dabei wird ausgenutzt, dass g sublinear ist, d.h. $g(k' + k'') \leq g(k') + g(k'')$ für alle $k', k'' \in \mathbb{N}_0$.

Allgemeine Lückenstrafen (Waterman-Smith-Beyer)



Skizze: Berechnung optimaler Alignments nach Waterman-Smith-Beyer

Allgemeine Lückenstrafen (Waterman-Smith-Beyer)

- Laufzeit jetzt größer (kubisch), weil für jeden Tabellen-Eintrag eine Minimumbildung von $\mathcal{O}(n+m)$ Elementen nötig ist:

$$\mathcal{O}(nm(n+m))$$

- Methode von Hirschberg nicht anwendbar, da alle vorherigen Zeilen benötigt werden

Theorem

Seien $s, t \in \Sigma^*$ mit $n = |s|$ und $m = |t|$.

Ein optimales globales paarweises Sequenzen-Alignment für s und t mit allgemeinen Lückenstrafen lässt sich in Zeit $\mathcal{O}(nm(n+m))$ mit Platz $\mathcal{O}(nm)$ berechnen.

Affine Lückenstrafen

- Laufzeit und Platzbedarf zu hoch für allgemeine Lückenstrafen
- deshalb: spezielle Lückenstrafen, zunächst affine, später konkave
- Affine Lückenstrafen:

$$g : \mathbb{N} \to \mathbb{R}_+ : \ k \mapsto \mu \cdot k + \nu$$

für Konstanten $\mu, \nu \in \mathbb{R}_+$

- $g(0)$ definieren wir, wie zu Beginn gefordert: $g(0) = 0$ (also ist eigentlich nur die Funktion auf $\mathbb{N}_+$ affin)

Affine Lückenstrafen

ν: Kosten, die für das Auftauchen einer Lücke berechnet werden (so genannte Strafe für Lückeneröffnung, engl. gap open penalty)

μ: Kosten für die Verlängerung der Lücke (so genannte Strafe für Lückenfortsetzung, engl. gap extension penalty)

- In der Literatur wird für die Lückenstrafe oft auch die Funktion $g(0) = 0$ und $g(k) = \mu(k-1) + \nu$ für alle $k \in \mathbb{N}$ verwendet.

Gotoh-Algorithmus für affine Strafen

- Algorithmus von Gotoh
- Rekursionsgleichungen etwas komplizierter, jetzt vier Tabellen:
  - $E[i,j]$ = Distanz eines optimalen Alignments von $s_1 \cdots s_i$ mit $t_1 \cdots t_j$, das mit einer Einfügung endet.
  - $F[i,j]$ = Distanz eines optimalen Alignments von $s_1 \cdots s_i$ mit $t_1 \cdots t_j$, das mit einer Löschung endet.
  - $G[i,j]$ = Distanz eines optimalen Alignments von $s_1 \cdots s_i$ mit $t_1 \cdots t_j$, das mit einer Substitution oder einem Match endet.
  - $D[i,j]$ = Distanz eines optimalen Alignments von $s_1 \cdots s_i$ mit $t_1 \cdots t_j$.

Gotoh-Algorithmus für affine Strafen

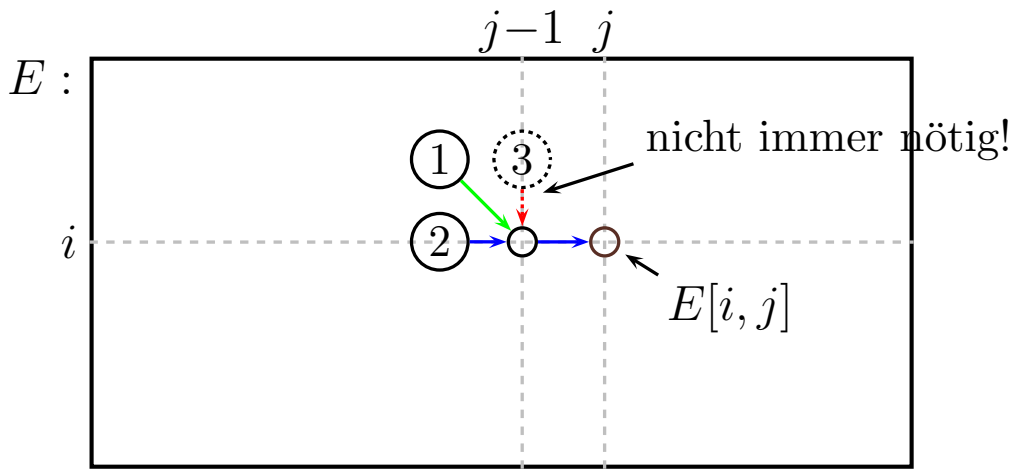
Rekursionsgleichungen:

- Tabelle E: Alignment endet mit einer Insertion
- davor: Substitution, Insertion oder Deletion
- 1. und 3. Fall: Lückeneröffnung (Kosten $\nu + \mu$),
  2. Fall: Lückenfortsetzung (Kosten μ)
- Somit erhalten wir:

$$E[i,j] = \min \left\{ \begin{array}{l} G[i,j-1] + \mu + \nu, \\ E[i,j-1] + \mu, \\ F[i,j-1] + \mu + \nu \end{array} \right\}.$$

(Der dritte Term $F[i,j-1] + \mu + \nu$ kann weggelassen werden, wenn eine Substitution (a,b) billiger als eine Deletion von a und eine Insertion von b ist. Allerdings ist dann nur D, aber nicht unbedingt E korrekt.)

Gotoh-Algorithmus für affine Strafen



1) $G[i, j-1] + \mu + \nu$
2) $E[i, j-1] + \mu$
3) $F[i, j-1] + \mu + \nu$

Skizze: Erweiterung eines Alignments mit einer Insertion

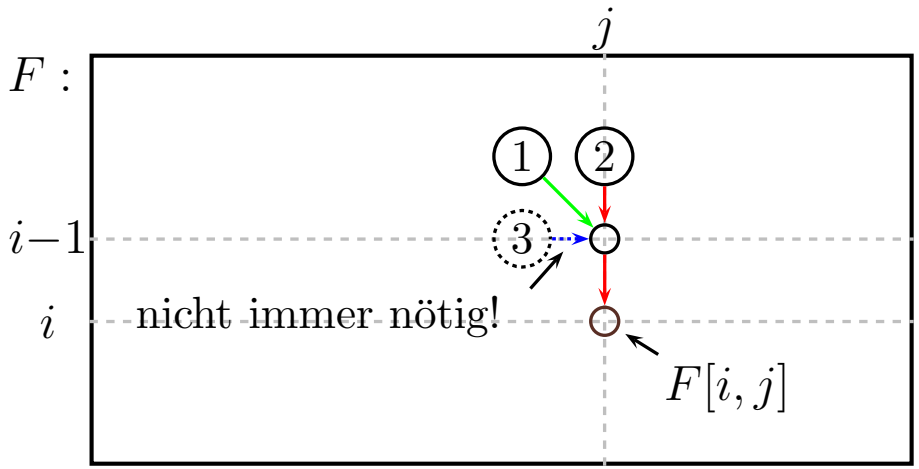
Gotoh-Algorithmus für affine Strafen

- Tabelle F: Alignment endet mit einer Deletion
- davor: Substitution, Deletion oder Einfügung
- 1. und 3. Fall: Lückeneröffnung (Kosten $\nu + \mu$),
  2. Fall: Lückenfortsetzung (Kosten μ)
- Somit erhalten wir:

$$F[i,j] = \min \left\{ \begin{array}{l} G[i-1,j] + \mu + \nu, \\ F[i-1,j] + \mu, \\ E[i-1,j] + \mu + \nu \end{array} \right\}.$$

(Der dritte Term $E[i-1,j] + \mu + \nu$ kann weggelassen werden, wenn eine Substitution (a,b) billiger als eine Deletion von a und eine Insertion von b ist. Allerdings ist dann nur D, aber nicht unbedingt F korrekt.)

Gotoh-Algorithmus für affine Strafen



1) $G[i-1,j]+\mu+\nu$
2) $F[i-1,j]+\mu$
2) $E[i-1,j]+\mu+\nu$

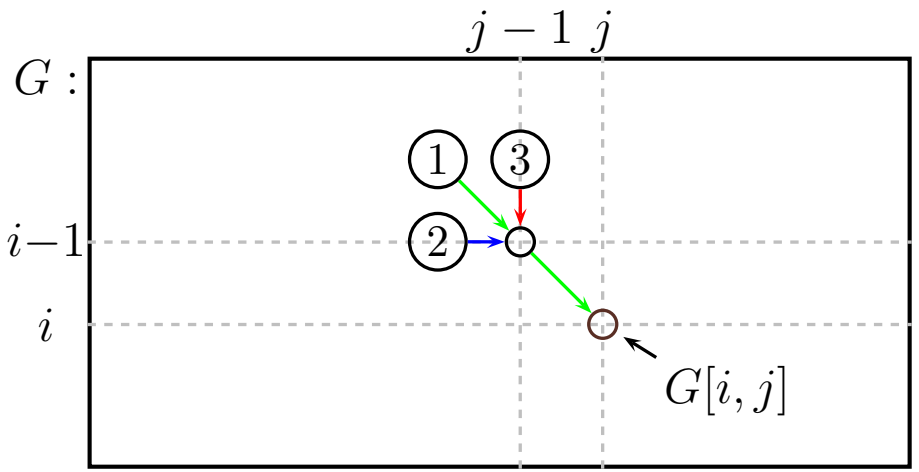
Skizze: Erweiterung eines Alignments mit einer Deletion

Gotoh-Algorithmus für affine Strafen

- Tabelle G: Alignment endet mit einer Substitution
- davor: Substitution, Deletion oder Insertion
- Dann erhalten wir:

$$G[i,j] = \min \left\{ \begin{array}{l} G[i-1,j-1] + w(s_i, t_j), \\ E[i-1,j-1] + w(s_i, t_j), \\ F[i-1,j-1] + w(s_i, t_j) \end{array} \right\}.$$

Gotoh-Algorithmus für affine Strafen



1) $G[i-1,j-1]+w(s_i,t_j)$
2) $E[i-1,j-1]+w(s_i,t_j)$
3) $F[i-1,j-1]+w(s_i,t_j)$

Skizze: Erweiterung eines Alignments mit einer Substitution

Gotoh-Algorithmus für affine Strafen

- Tabelle D: Minimum aller drei Tabellen

$$D[i,j] = \min \left\{ \begin{array}{l} E[i,j], \\ F[i,j], \\ G[i,j] \end{array} \right\}.$$

- bei Ähnlichkeitsmaßen:

  Rekursionsgleichungen im Wesentlichen gleich,
  Minimum wird durch Maximum ersetzt

  Für die Tabellen E und F müssen alle Deletionen und Insertionen berücksichtigt werden, da bei Ähnlichkeitsmaßen aufgrund der fehlenden Dreiecksungleichung auch Insertionen und Deletionen unmittelbar benachbart sein dürfen.

Gotoh-Algorithmus für affine Strafen

Werte in der 1. Zeile bzw. in der 1. Spalte der Matrizen?
Es gilt für $i > 0$ und $j > 0$:

$$\begin{aligned} E[0,j] &= j * \mu + \nu, \\ E[i,0] &= \infty, \\ E[0,0] &= \infty, \end{aligned}$$

$$\begin{aligned} F[i,0] &= i * \mu + \nu, \\ F[0,j] &= \infty, \\ F[0,0] &= \infty, \end{aligned}$$

$$\begin{aligned} G[i,0] &= \infty, \\ G[0,j] &= \infty, \\ G[0,0] &= 0. \end{aligned}$$

Gotoh-Algorithmus für affine Strafen

- Eigentlich müsste Auch $G[0,0] = \infty$ sein, da es kein Alignment leerer Sequenzen gibt, die mit einer Substitution bzw. Match enden.
- Dann würde jedoch $G[1,1]$ falsch berechnet werden.
- Also interpretieren wir das Alignment von ϵ mit sich selbst als Match.
- $E[0,0]$ bzw. $F[0,0]$ darf jedoch nicht mit 0 initialisiert werden, da sonst die Eröffnung einer ersten Lücke von Deletionen bzw. Insertionen straffrei bleiben würde.

Gotoh-Algorithmus für affine Strafen

Auch hier kann man wieder die Methode von Hirschberg zur Platzreduktion anwenden.

Theorem

Seien $s, t \in \Sigma^*$ mit $n = |s|$ und $m = |t|$.

Ein optimales globales paarweises Sequenzen-Alignment für s und t mit affinen Lückenstrafen lässt sich in Zeit $\mathcal{O}(nm)$ mit Platz $\mathcal{O}(\min(n, m))$ berechnen.

Konkave Lückenstrafen

- Eine Funktion $f : \mathbb{N}_0 \to \mathbb{R}$ heißt konkav, wenn gilt:

$$\forall n \in \mathbb{N} : \; f(n) - f(n-1) \geq f(n+1) - f(n).$$

- anschaulich: Funktion wächst immer langsamer
- Äquivalent in der kontinuierlichen Analysis: erste Ableitung monoton fallend bzw. zweite Ableitung kleiner gleich Null (sofern die Funktion zweimal differenzierbar ist)
- bekanntes Beispiel konkaver Funktionen: Logarithmus-Funktion

Konkave Lückenstrafen

Lemma

Sei $g : \mathbb{N} \to \mathbb{R}$ *eine konkave Funktion.*

Dann gilt für alle $q \leq q' \in \mathbb{N}$ *und* $d \in \mathbb{N}$*:*

$$g(q+d) - g(q) \geq g(q'+d) - g(q').$$

Beweis.

Durch wiederholtes Anwenden der Definition einer konkaven Funktion gilt für alle $k' \geq k \in \mathbb{N}$:

$$g(k) - g(k-1) \geq g(k') - g(k'-1).$$

Konkave Lückenstrafen

Beweis.

Addiere Ungleichungen:
linke Seite für alle $k \in [q+1 : q+d]$ und
rechte Seite für alle $k' \in [q'+1 : q'+d]$

Man erhält für alle $q \leq q'$:

$$\begin{aligned} g(q+d) - g(q) &= \sum_{k=q+1}^{q+d} g(k) - g(k-1) \\ &\geq \sum_{k'=q'+1}^{q'+d} g(k') - g(k'-1) \\ &= g(q'+d) - g(q') \end{aligned}$$

Konkave Lückenstrafen

Wiederholung: leicht modifizierte Fassung der Rekursionsgleichungen für eine beliebige Lückenstrafe g (basierend auf der Variante für affine Lückenstrafen):

$$
\begin{aligned}
D[i,j] &:= \min\{E[i,j], F[i,j], G[i,j]\},\\
G[i,j] &:= D[i-1,j-1] + w(s_i, t_j),\\
E[i,j] &:= \min\{D[i,k] + g(j-k) \;:\; k \in [0:j-1]\},\\
F[i,j] &:= \min\{D[k,j] + g(i-k) \;:\; k \in [0:i-1]\},\\
D[i,0] &:= g(i),\\
D[0,j] &:= g(j),\\
E[i,0] &:= g(i),\\
F[0,j] &:= g(j).
\end{aligned}
$$

Nicht definierte Einträge: ∞