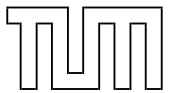
Algorithmische Bioinformatik 1

Dr. Hanjo Täubig

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

Sommersemester 2009



Übersicht

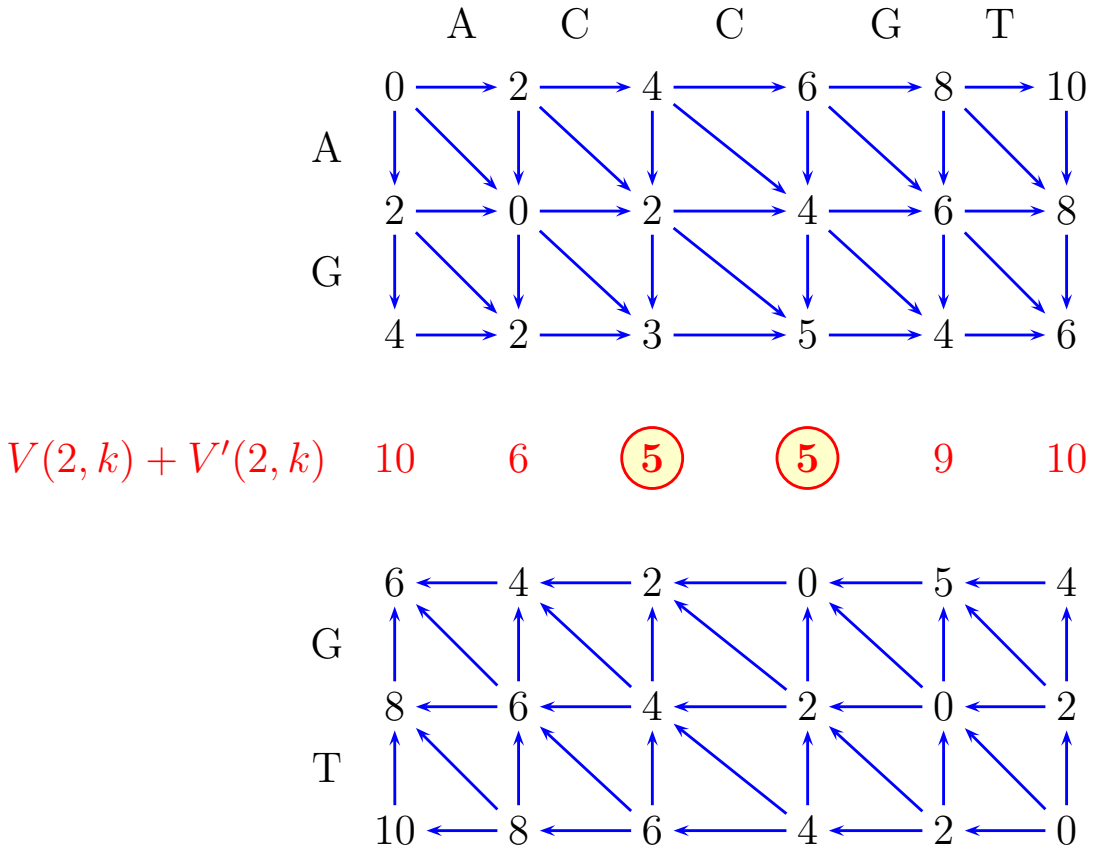
Hirschberg-Verfahren

- Beispiel für das Hirschberg-Verfahren zur Bestimmung eines optimalen Sequenzen-Alignments anhand von zwei Sequenzen:

$$\begin{aligned} s &= AGGT \quad \text{und} \\ t &= ACCGT \end{aligned}$$

- Zuerst wird der Wert des optimalen Alignments für $s_1 \cdots s_2$ mit $t_1 \cdots t_k$ und für $s_3 \cdots s_4$ mit $t_{k+1} \cdots t_5$ für alle $k \in [0:5]$ berechnet.

Hirschberg-Verfahren



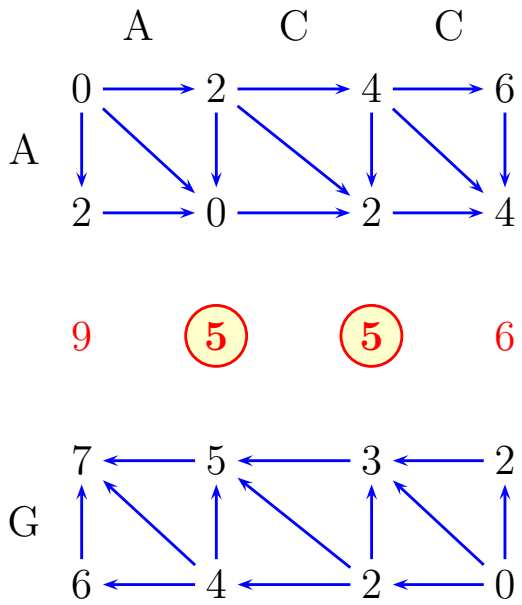
$V(2,k) + V'(2,k)$ 10 6 **5** **5** 9 10

Bestimmung von m'

Hirschberg-Verfahren

- Der nächste Schritt besteht nun darin, m' zu bestimmen.
- In unserem Fall sind zwei verschiedene Werte möglich, da zweimal der Wert 5 auftritt.
- Für den weiteren Verlauf entscheiden wir uns für $m' = 3$.
- Jetzt müssen wir rekursiv die beiden Teile bearbeiten.

Hirschberg-Verfahren



- Zuerst betrachten wir den oberen linken Teil.
- Wieder haben wir zwei Schnittpunkte zur Wahl, nämlich 1 und 2.
- Wir entscheiden uns für 1.

Hirschberg-Verfahren

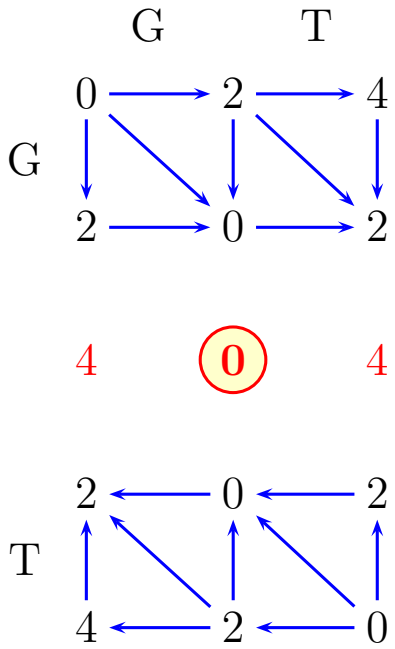
- Damit erhalten wir jetzt Probleme, bei denen die erste Sequenz Länge 1 hat.
- Wir müssen jetzt also ein Alignment für A mit A und für G mit CC finden.
- Offensichtlich wählt man $\binom{A}{A}$ und $\binom{G-}{CC}$.
- Dieses wird dann zu $\binom{AG-}{ACC}$ zusammengesetzt.

Hirschberg-Verfahren

- Jetzt fehlt noch der zweite rekursive Aufruf für $m' = 3$, d.h. der untere rechte Teil (siehe wieder nachfolgende Abbildung).
- Hier ist der Aufteilungspunkt eindeutig und die Zeichen stimmen ja auch überein, so dass wir zuerst zwei kurze Alignments $\binom{G}{G}$ und $\binom{T}{T}$ erhalten, die dann zu $\binom{GT}{GT}$ zusammengesetzt werden.

- Setzt man nun die beiden Alignments aus dem ersten Aufruf, nämlich $\binom{AG-}{ACC}$, und dem zweiten rekursiven Aufruf, nämlich $\binom{GT}{GT}$, zusammen, so erhalten wir als gesamtes Alignment $\binom{AG-GT}{ACCGT}$, das auch die schon berechnete Distanz 5 besitzt.

Hirschberg-Verfahren



Beispiel: Zweiter rekursiver Aufruf im Hirschberg-Algorithmus

Hirschberg-Verfahren: Platzbedarf

- Wir haben bereits die Korrektheit der Variante von Hirschberg bewiesen. Es bleibt noch zu zeigen, dass der Platzbedarf wirklich linear ist und wie groß die Laufzeit ist.
- Platzbedarf: Dazu betrachten wir noch einmal den angegebenen Algorithmus.
- Schritte 1 und 2 können wir, wie bereits erläutert, in linearem Platz $O(m)$ berechnen.
- Schritt 3 benötigt keinen weiteren Platz.
- Im letzten Schritt rufen wir zweimal rekursiv die Prozeduren auf und benötigen für die erste Rekursion Platz $O(m')$ sowie für die zweite Rekursion Platz $O(m'')$ und somit wieder Platz von $O(m' + m'') = O(m)$.
- Da wir den Platz aus Schritt 1 und 2 wiederverwenden können, benötigen wir insgesamt nur Platz $O(m)$.

Hirschberg-Verfahren: Laufzeitanalyse

- Laufzeitanalyse: Wir bezeichnen hierzu mit $T(n, m)$ den Zeitbedarf für die Hirschberg-Variante für zwei Zeichenreihen mit Längen n und m.
- Wir stellen zuerst eine Rekursionsformel für T auf.
- Hierfür zählen wir die Anzahl besuchter Einträge der virtuellen Tabelle D.

Hirschberg-Verfahren: Laufzeitanalyse

- Schritt 1 besucht $(\lfloor \frac{n}{2} \rfloor + 1) \cdot (m+1)$ Felder, Schritt 2 $(\lfloor \frac{n}{2} \rfloor + 1 + 1) \cdot (m+1)$ Felder.
- Schritt 3 besucht dieselben $2(m+1)$ Felder, wie in Schritt 1 und 2.
- Die Schritte 1 bis 3 besuchen also maximal $(n+2)(m+1)$ Felder.
- Aufgrund der beiden rekursiven Aufrufe im Schritt 4, ist der Zeitbedarf hierfür durch $T(\lfloor n/2 \rfloor, k) + T(\lceil n/2 \rceil, m-k)$ gegeben, wobei $k \in [0:m]$.
- Somit erhalten wir folgende Rekursionsformel für den Zeitbedarf:

$$T(n,m) = (n+2)(m+1) + T\left(\left\lfloor \frac{n}{2} \right\rfloor, k\right) + T\left(\left\lceil \frac{n}{2} \right\rceil, m-k\right).$$

Hirschberg-Verfahren: Laufzeitanalyse

- Wir könnten diese Rekursionsgleichung mit aufwendigen Mitteln direkt lösen.
- Wir machen es uns aber hier etwas leichter und verifizieren eine geratene Lösung mittels Induktion.

Beobachtung

Es gilt $T(n, m) \leq 3n(m+1) + 8m\log(n)$.

Hirschberg-Verfahren: Laufzeitanalyse

Beweis.

Wir beweisen die Behauptung mittels vollständiger Induktion nach n.

Induktionsanfang ($n = 1$): $T(1, m)$ ist sicherlich durch $3(m + 1)$ beschränkt, da wir nur ein Zeichen gegen eine Zeichenreihe der Länge m optimal ausrichten müssen, was eine Inspektion von $2(m + 1)$ Feldern bedeutet.

Induktionsschritt ($\longrightarrow n$): Wir setzen nun die Behauptung als Induktionsvoraussetzung in die Rekursionsformel ein (da $\lceil n/2 \rceil < n$ für $n \geq 2$) und formen zunächst für $m \geq 2$ um:

Hirschberg-Verfahren: Laufzeitanalyse

Beweis.

$$
\begin{aligned}
T(n,m) &= (n+2)(m+1) + T\left(\left\lfloor \frac{n}{2} \right\rfloor, k\right) + T\left(\left\lceil \frac{n}{2} \right\rceil, m-k\right) \\
&\leq nm + n + 2m + 2 \\
&\quad + 3\left\lfloor \frac{n}{2} \right\rfloor (k+1) + 8k \log\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \\
&\quad + 3\left\lceil \frac{n}{2} \right\rceil (m-k+1) + 8(m-k) \log\left(\left\lceil \frac{n}{2} \right\rceil\right) \\
&\leq nm + n + 2m + 2 + 3n + 3\left\lceil \frac{n}{2} \right\rceil m + 8m \log\left(\left\lceil \frac{n}{2} \right\rceil\right) \\
&\quad \text{da } n \leq 2\lfloor n/2 \rfloor + 1,\ m \geq 2 \text{ und } \left\lceil \tfrac{n}{2} \right\rceil \leq \tfrac{2}{3} n \\
&\leq \left(2\left\lfloor \frac{n}{2} \right\rfloor + 1\right) m + \left(m\left\lfloor \frac{n}{2} \right\rfloor + 1\right) + 2m + 2 + 3n \\
&\quad + 3\left\lceil \frac{n}{2} \right\rceil m + 8m \log\left(\frac{2n}{3}\right)
\end{aligned}
$$

Hirschberg-Verfahren: Laufzeitanalyse

Beweis.

$$
\begin{aligned}
T(n,m) &\leq 3\left\lfloor \frac{n}{2} \right\rfloor m + 3m + 3 + 3n + 3\left\lceil \frac{n}{2} \right\rceil m + 8m\log\left(\frac{2n}{3}\right) \\
&\leq 3n(m+1) + 3m + 3 + 8m\log(n) + 8m\log\left(\frac{2}{3}\right) \\
&\leq 3n(m+1) + 8m\log(n) + m\left[3 + \frac{3}{m} + 8\log\left(\frac{2}{3}\right)\right] \\
&\quad \text{da } [9/2 + 8\log(2/3)] < 0 \\
&\leq 3n(m+1) + 8m\log(n).
\end{aligned}
$$

Beachte, dass wir auch den Fall $m \in [0:1]$ untersuchen müssen. Für $m = 0$ gilt aber offensichtlich $T(n,0) = n \leq 8n - 2$ für alle $n \geq 2$. Für $m = 1$ gilt weiter $T(n,1) = 2(n+1) \leq 6n + 8 \leq 6n + 8\log(n)$ für alle $n \geq 2$. □

Hirschberg-Verfahren

Da man die Sequenzen s und t vertauschen kann, bestimmt die kürzere der beiden Sequenzen den benötigten Platzbedarf.

Theorem

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

Der Algorithmus von Hirschberg berechnet ein optimales globales paarweises Sequenzen-Alignment für s und t in Zeit $\mathcal{O}(nm)$ und Platz $\mathcal{O}(\min\{n, m\})$.

Spezielle Lückenstrafen

- In diesem Abschnitt: teilweise Alignments und Strafen für längere Lücken genauer untersuchen.
- Lücke (gap): aufeinander folgende Folge von Edit-Operationen, die entweder nur aus Deletionen oder nur aus Insertionen bestehen (jedoch nicht abwechselnd)
- im Alignment: entspricht einem Teilwort, das nur aus Leerzeichen besteht
- Solche zusammenhängenden Lücken der Länge ℓ haben ihre Ursache meist aus einer einzigen Mutation, die eine ganze Teilsequenz entfernt bzw. eingefügt hat.
- Aus diesem Grund ist eine Bestrafung, die proportional zur Länge der Lücke ist, nicht realistisch, und sollte daher eher sublinear in der Länge der Lücke sein.

Semiglobale Alignments

- Im Falle semiglobaler Alignments wollen wir Lücken, die am Anfang oder am Ende eines Wortes auftreten, nicht berücksichtigen.
- Semiglobale Alignments werden oft auch als Free-Shift Alignments bezeichnet.
- Dies ist insbesondere dann von Interesse, wenn die Wörter sehr unterschiedlich lang sind oder wenn klar ist, dass diese Sequenzen zwar eine Ähnlichkeit besitzen, aber man nicht weiß, ob man die Sequenzen korrekt aus einer großen Sequenz herausgeschnitten hat.
- Dann können an den Enden Fehler aufgetreten sein (etwas zu kurze oder zu lange Sequenzen gewählt).

Semiglobale Alignments: Beispiel

Betrachten wir die beiden Sequenzen *CGTACGTGATGA* und *CGATTA*. Wenn wir hierfür die optimale Alignment-Distanz berechnen (mit $w(x, y) = 3$ für $x \neq y \in \Sigma$ und $w(x, -) = 2$ für $x \in \Sigma$), so erhalten wir das folgende optimale Alignment:

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

Dieses hat einen Alignment-Distanz von $7 * 2 = 14$.
Alternativ betrachten wir folgendes Alignment:

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

Dieses hat natürlich eine größere Alignment-Distanz von $9 * 2 + 1 * 3 = 21$.

Semiglobale Alignments

- Berücksichtigen wir jedoch die Deletionen am Anfang und Ende nicht, da diese vermutlich nur aus einer zu lang ausgewählten ersten (oder zu kurz ausgewählten zweiten) Sequenz herrühren, so erhalten wir eine Alignment-Distanz von $1 * 2 + 1 * 3 = 5$.

- Aus diesem Grund werden bei einem semiglobalen Alignment Folgen von Insertionen bzw. Deletionen zu Beginn oder am Ende nicht berücksichtigt.

Semiglobale Alignments: Problem

- Es gibt jedoch noch ein kleines Problem:
  Man kann nämlich dann immer ein Alignment mit Alignment-Distanz 0 basteln:

$$CGTACGTGAGTGA(\epsilon)------$$
$$--------------(\epsilon)CGATT$$

- Hier wurde also eigentlich ϵ mit ϵ aligniert.
- Bei solchen Distanzen sollte man den Wert der Distanz im Verhältnis zur Länge des Bereiches in Beziehung setzen, in dem das eigentliche, bewertete Alignment steht.
- Man kann jetzt die Distanz bezüglich der wirklich ausgerichteten Zeichen um jeweils einen konstanten Betrag erniedrigen.
- Einfacher geht es, wenn wir stattdessen Ähnlichkeitsmaße verwenden.

Semiglobale Alignments

- Wir verwenden jetzt als Kostenfunktion für ein Ähnlichkeitsmaß für Matches $+2$, für Insertionen sowie Deletionen -1 und für Substitutionen -1.
- Dieses Kostenmaß wurde aus der Kostenfunktion für das Distanzmaß mittels $2 - w(x, y)$ bzw. $1 - w(x, a)$ gewonnen.
- Somit erhält man für das erste globale Alignment

$$6 * (+2) + 7 * (-1) = +5.$$

- Für das zweite Alignment erhält man als globales Alignment einen Ähnlichkeitswert von

$$4 * (+2) + 9 * (-1) + 1 * (-1) = -2$$

- und als semiglobales-Alignment einen Score von

$$4 * (+2) + 1 * (-1) + 1 * (-2) = +6.$$

- Für das künstliche Alignment jedoch einen Score von 0.

Semiglobale Alignments

Problem

Semiglobales Alignment

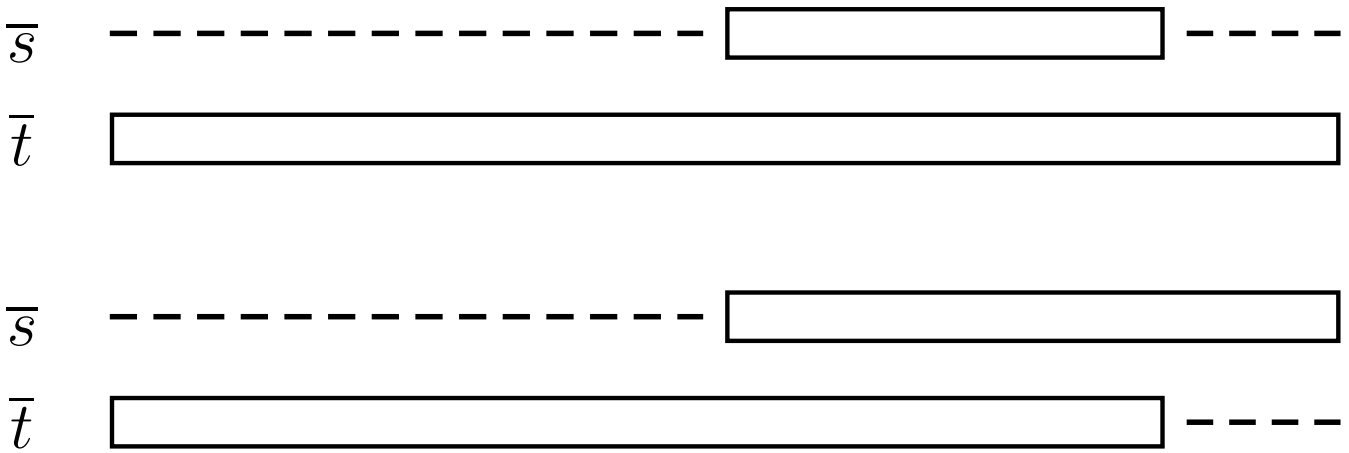
Eingabe: *Seien $s \in \Sigma^n$ und $t \in \Sigma^m$ und σ ein Ähnlichkeitsmaß.*

Gesucht: *Zwei Teilwörter $s' = s_{i_1} \cdots s_{j_1}$ und $t' = t_{i_2} \cdots t_{j_2}$ mit einem optimalen Ähnlichkeitswert $\sigma(s', t')$, wobei entweder*

- $i_1 = 1$ *und* $i_2 = 1$,
- $i_1 = 1$ *und* $j_2 = m$,
- $j_1 = n$ *und* $i_2 = 1$,
- $j_1 = n$ *und* $j_2 = m$,
- $i_1 = 1$ *und* $j_1 = n$ *oder*
- $i_2 = 1$ *und* $j_2 = m$ *ist.*

Semiglobale Alignments

- Wie äußert sich jetzt die Nichtberücksichtigung von Lücken am Anfang und Ende eines Alignments in der Berechnung mit Hilfe der Dynamischen Programmierung nach Needleman-Wunsch?



Skizze: semiglobale Alignments

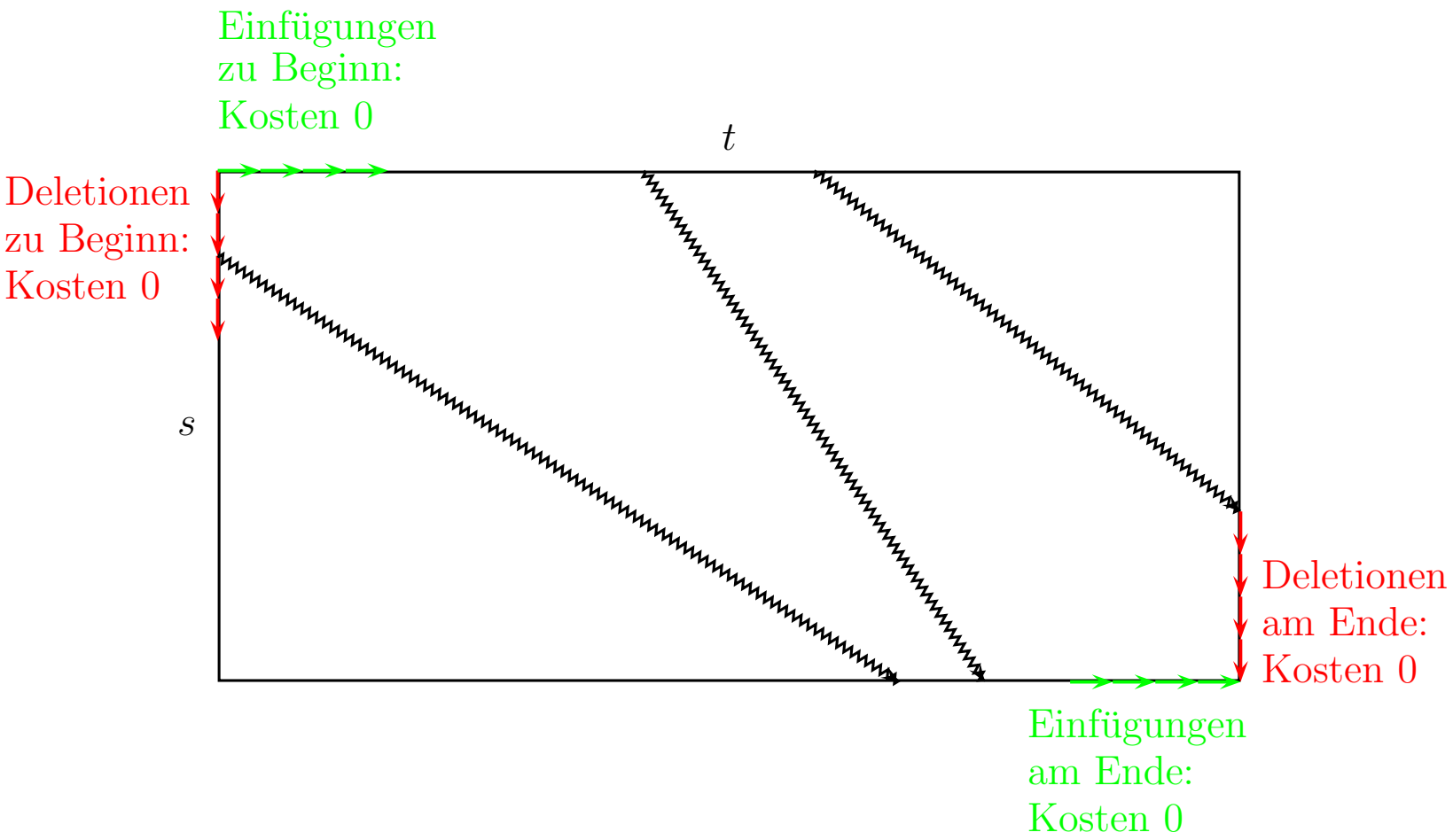
Semiglobale Alignments: Lücken am Anfang

- Wenn in der ersten Sequenz (s) am Anfang Lücken auftreten dürfen, bedeutet dies, dass wir in der zweiten Sequenz (t) Einfügungen gemacht haben.
- Damit diese nicht zählen, dürfen diese Einfügungen zu Beginn nicht gewertet werden.
- Daher werden wir die erste Zeile der Tabelle mit 0 initialisieren.
- Analoges gilt für Lücken zu Beginn von t.
- Dann dürfen die Deletionen von s nicht bewertet werden und wir initialisieren auch die erste Spalte mit 0.

Semiglobale Alignments: Lücken am Ende

- Tritt am Ende von s eine Lücke auf, dann dürfen wir die letzten Insertionen von Zeichen in t nicht berücksichtigen.
- Dazu betrachten wir die letzte Zeile der Tabelle.
- Wenn die letzten Insertionen nicht zählen sollen, dann hört ein solches semiglobales Alignment irgendwo in der letzten Zeile auf.
- Wenn wir nun ein semiglobales Alignment mit maximaler Ähnlichkeit wollen, müssen wir einfach nur in der letzten Zeile den maximalen Wert suchen.
- Die Spalten dahinter können wir für unser semiglobales Alignment dann einfach vergessen.
- Dasselbe gilt für Deletionen in s.
- Dann hört das semiglobale Alignment irgendwo in der letzten Spalte auf und wir bestimmen für ein optimales semiglobales Alignment den maximalen Wert in der letzten Spalte und vergessen die Zeilen danach.

Semiglobale Alignments



Skizze: Pfade semiglobaler Alignments in der Ähnlichkeitstabelle

Semiglobale Alignments

- Optimales semiglobales Alignment:
  - erste Zeile und erste Spalte auf Null setzen und
  - den maximalen Ähnlichkeitswert, der in der letzten Zeile oder Spalte auftritt, bestimmen

- Das Alignment selbst erhalten wir dann genauso wie im Falle des globalen Alignments, indem wir einfach von diesem Maximalwert rückwärts das Alignment bestimmen.

- Wir hören auf, sobald wir auf die erste Spalte oder die erste Zeile treffen.

Semiglobale Alignments

Damit ergibt sich für die Tabelle S:

$$S(i,j) = \begin{cases} 0 & \text{für } (i=0) \vee (j=0), \\ \max \begin{cases} S(i-1,j-1) + w(s_i,t_j), \\ S(i-1,j) + w(s_i,-), \\ S(i,j-1) + w(-,t_j) \end{cases} & \text{für } (i>0) \wedge (j>0). \end{cases}$$

Auch für semiglobale Alignments lässt sich das Verfahren von Hirschberg zur Platzreduktion anwenden.

Theorem

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

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

Lokale Alignments

- weitere Einschränkung: lokale Alignments
- gesucht: zwei möglichst ähnliche Teilwörter in zwei Sequenzen
- Damit wir nicht wieder zwei leere Teilwörter mit Alignment-Distanz 0 bekommen, verwenden wir auch hier wieder Ähnlichkeitsmaße.

Skizze: Lokales Alignment

Lokale Alignments

Problem

Lokales Alignment

Eingabe: $s \in \Sigma^n$ *und* $t \in \Sigma^m$, *Ähnlichkeitsmaß* σ

Gesucht: *Zwei Teilwörter* $s' = s_{i_1} \cdots s_{j_1}$ *und* $t' = t_{i_2} \cdots t_{j_2}$ *mit einem optimalen Ähnlichkeitswert* $\sigma(s', t')$.

Lokale Alignments

- Beispiel:
  lokales Alignment zwischen den Sequenzen
  $s = ACGATTATTT$ und $t = TAGTAATCG$.
- Das lokale Alignment besteht aus den beiden Teilwörtern, die in dem Rahmen eingefasst sind, und hat den Ähnlichkeitswert 7 (hierbei ist $w(x, x) = 3$, $w(x, y) = -3$ und $w(x, -) = -2$ für $x \neq y \in \Sigma$).

s:	A	C	G	A	T	T	A	T	T	T
t:	T	A	G	–	T	A	A	T	C	G

Beispiel: Ein lokales Alignment zwischen s und t

Lokale Alignments

- Berechnung: auch hier ähnlich zur Methode von Needleman-Wunsch
- $S(i,j)$:
  Wert eines besten lokalen Alignments von zwei Teilwörtern von $s_{i'} \cdots s_i$ und $t_{j'} \cdots t_j$ mit $i' \leq i+1$ und $j' \leq j+1$
- Rekursionsgleichung:

$$S(i,j) = \begin{cases} 0 & \text{für } (i=0) \vee (j=0), \\ \max \begin{cases} S(i-1,j-1) + w(s_i,t_j), \\ S(i-1,j) + w(s_i,-), \\ S(i,j-1) + w(-,t_j), \\ 0 \end{cases} & \text{für } (i>0) \wedge (j>0). \end{cases}$$

Lokale Alignments

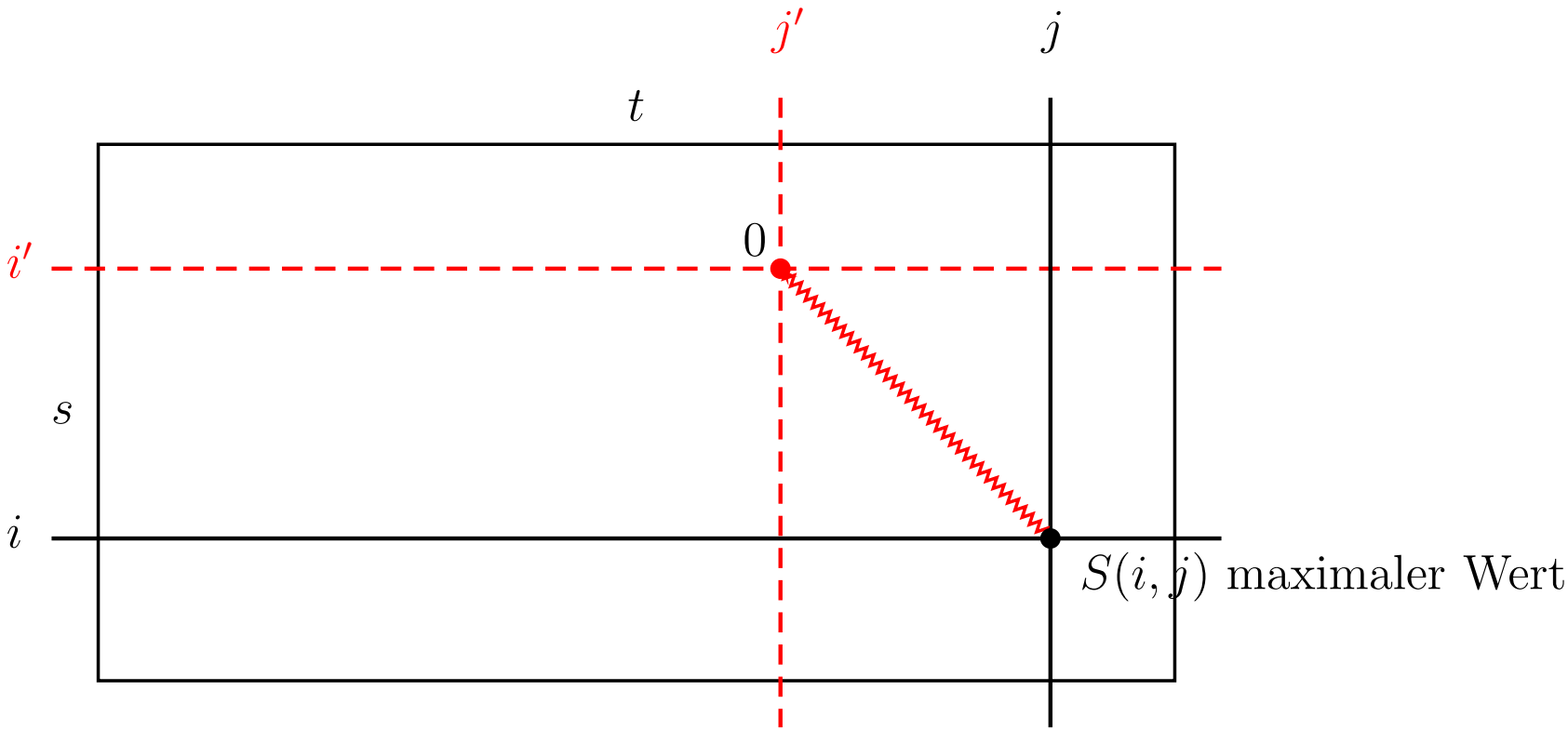
- Die Rekursionsgleichung sieht fast so aus, wie im Falle der semiglobalen Alignments.
- Wir müssen hier nur in der Maximumsbildung im Falle von $i \neq 0 \neq j$ den Wert 0 berücksichtigen, weil ein lokales Alignment ja an jeder Stelle innerhalb der beiden gegebenen Sequenzen s und t beginnen kann.

- Wie finden wir nun ein optimales lokales Alignment?
- Da ein lokales Alignment ja an jeder Stelle innerhalb der Sequenzen s und t enden darf, müssen wir einfach nur den maximalen Wert innerhalb der Tabelle S finden. Dies ist dann der Ähnlichkeitswert eines optimalen lokalen Alignments.

Lokale Alignments

- Die Korrektheit folgt wiederum aus der Tatsache, dass ein optimales lokales Alignment, das das Suffix von $s_1 \cdots s_i$ bzw. $t_1 \cdots t_j$ umfasst, entweder mit einer Substitution, einem Match, einer Insertion, einer Deletion endet oder ganz und gar leer ist und somit den Wert 0 erhält.
- Das Alignment selbst finden wir dann wieder durch Rückwärtsverfolgen der Sieger aus der Maximumbildung.
- Ist der Sieger letztendlich der Wert 0 in der Maximumsbildung, so haben wir den Anfangspunkt eines optimalen lokalen Alignments gefunden.
- Die auf dieser Rekursionsgleichung basierende Methode wird oft auch als Algorithmus von Smith-Waterman bezeichnet.

Lokale Alignments



Skizze: Pfad des lokalen Alignments in der Tabelle

Lokale Alignments

- Auch für das lokale paarweise Sequenzen-Alignment lässt sich die Methode von Hirschberg zur Platzreduktion anwenden.

Theorem

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

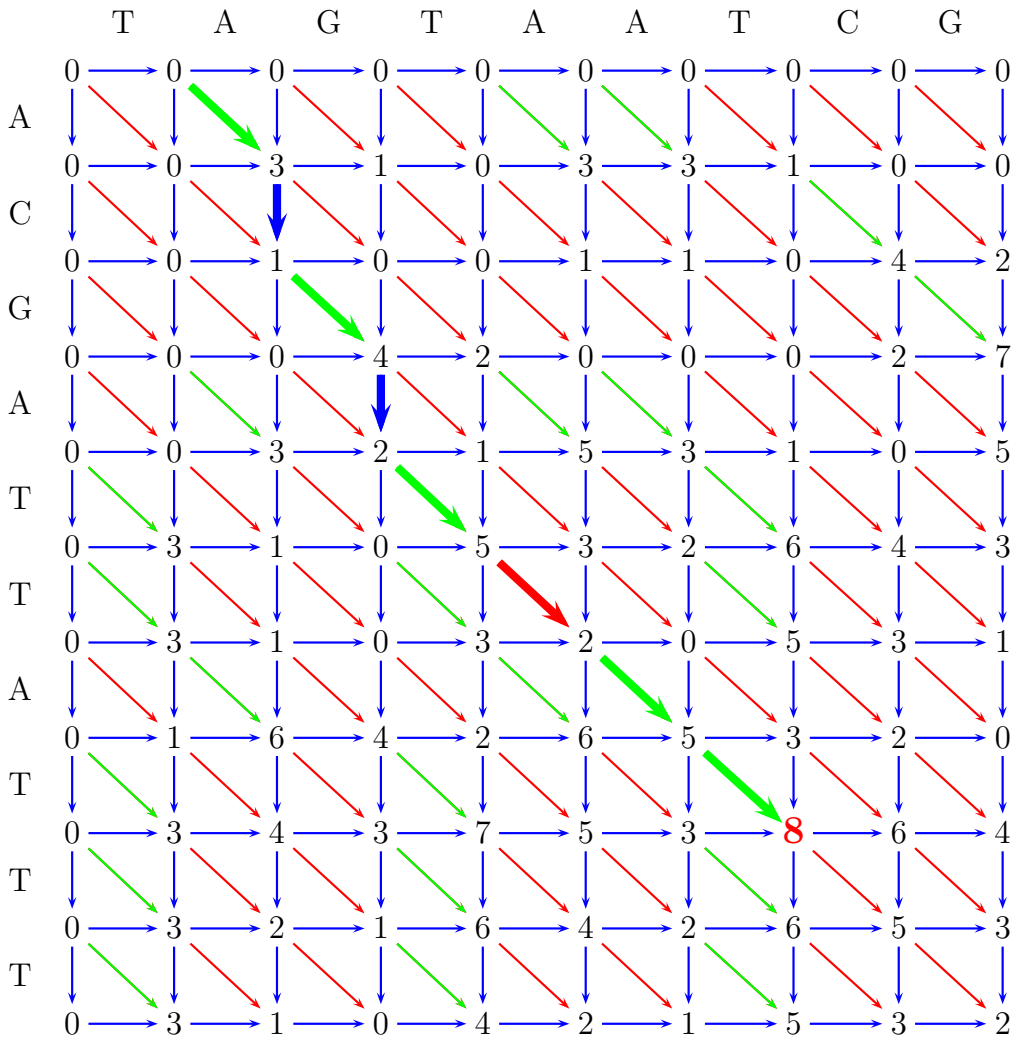
Ein optimales lokales paarweises Sequenzen-Alignment für s und t sowie der zugehörige Ähnlichkeitswert lässt sich in Zeit $\mathcal{O}(nm)$ mit Platz $\mathcal{O}(\min\{n, m\})$ berechnen.

Lokale Alignments

- Kehren wir noch einmal zu unserem konkreten Beispiel vom Anfang des Abschnitts zurück und berechnen die Tabelle S für die Sequenzen $s = ACGATTATTT$ und $t = TAGTAATCG$.
- Die Tabelle mit den zugehörigen Werten ist in der nächsten Abbildung angegeben.

- Der maximale Wert ist 8 (siehe Position $(8, 7)$ in der Tabelle für S).
- Der zurückverfolgte Weg für das optimale lokale Alignment ist in der Abbildung durch die dicken Pfeile dargestellt.

Lokale Alignments



Beispiel: Tabelle für lokale Alignments zwischen s und t

Lokale Alignments

- Auf die Null trifft man an der Position $(0, 1)$ in der Tabelle S.
- Aus diesem Pfad lässt sich wie üblich wieder das zugehörige lokale Alignment ablesen.
- Das zu Beginn angegebene lokale Alignment war also nicht optimal, aber schon ziemlich nahe dran.
- Durch eine Verlängerung kommt man auf das optimale lokale Alignment.

s:	−	A	C	G	A	T	T	A	T	T	T
t:	T	A	−	G	−	T	A	A	T	C	G

Beispiel: Ein optimales lokales Alignment zwischen s und t