Aufgabenblatt 4

Silke Trißl

Wissensmanagement in der Bioinformatik

Zuerst!

FRAGEN ?

Exercise 1

- Global alignment using dynamic programming
- Write a program to implement the algorithm based on dynamic programming.
- **BUT** – First let us repeat!

Abstandsmaße

- Approximatives Stringmatching sucht Ähnlichkeiten
  - Welcher String $T_1,...,T_n$ ist am ähnlichsten zu T?
- Voraussetzung dafür
  - Was heißt ähnlich?
  - Was heißt „am ähnlichsten“?
- Definition von Ähnlichkeit ist oft eine sehr schwierige Aufgabe
  - Ähnlichkeit ist abhängig vom Gegenstand und Aufgabe
    - Wann sind sich Gesichter ähnlich - Haarfarbe zählt weniger als Augenfarbe?
    - Wann sind sich Texte ähnlich – gleiche Wörter oder gleicher Inhalt?

Wie ähnlich sind sich zwei Sequenzen?

- „AGGTAG“ und
  - AGTAG — G zu wenig
  - AGGTAG — Identisch = überaus ähnlich
  - AGGATAG — A zu viel
  - AGTTCAG — G durch T ersetzen und C löschen
  - …TGAGGTAGGTT… — Sehr viel löschen
- Welche Sequenzen sind sich also besonders ähnlich?
  - „Ähnlichkeit“ muss quantifiziert werden
  - Idee: Wie sehr muss man eine Sequenz verändern, um die andere zu erzeugen
    - Man konnte auch Buchstaben zählen, Länge vergleichen, Anzahl GC nehmen, …

Editskripte

- Definition

  Ein Editskript e für zwei Strings A, B aus Σ = Σ U '_' ist eine Sequenz von Editieroperationen*

  - *I (Einfügen eines Zeichens c ε Σ* in A)*
    - *Dargestellt als Lücke in A; das neue Zeichen erscheint in B*
  - *D (Löschen eines Zeichens c in A)*
    - *Dargestellt als Lücke in B; das alte Zeichen erscheint in A*
  - *R (Ersetzen eines Zeichen in A mit einem anderen Zeichen in B)*
  - *M (Match, d.h., gleiche Zeichen in A und B an dieser Stelle)*

  so, dass e(A)=B

- Beispiel: A=„ATGTA“, B=„AGTGTC“

```
MIMMMR      IRMMMDI
A_TGTA      _ATGTA_
AGTGTC      AGTGT_C
```

Editabstand

- Offensichtlich gibt es für A,B ziemlich viele Editskripte
- Definiton
  - *Die Länge eines Editskript ist die Anzahl von Operationen o im Skript mit o ε {I, R, D}*
  - *Der Editabstand zweier Strings A, B ist die Länge des kürzesten Editskript für A, B*
- Bemerkung
  - Matchen zählt nicht – interessant sind nur die Änderungen
  - Anderer Name: Levenshtein-Abstand
  - Es gibt oft verschiedene kürzeste Editskripte

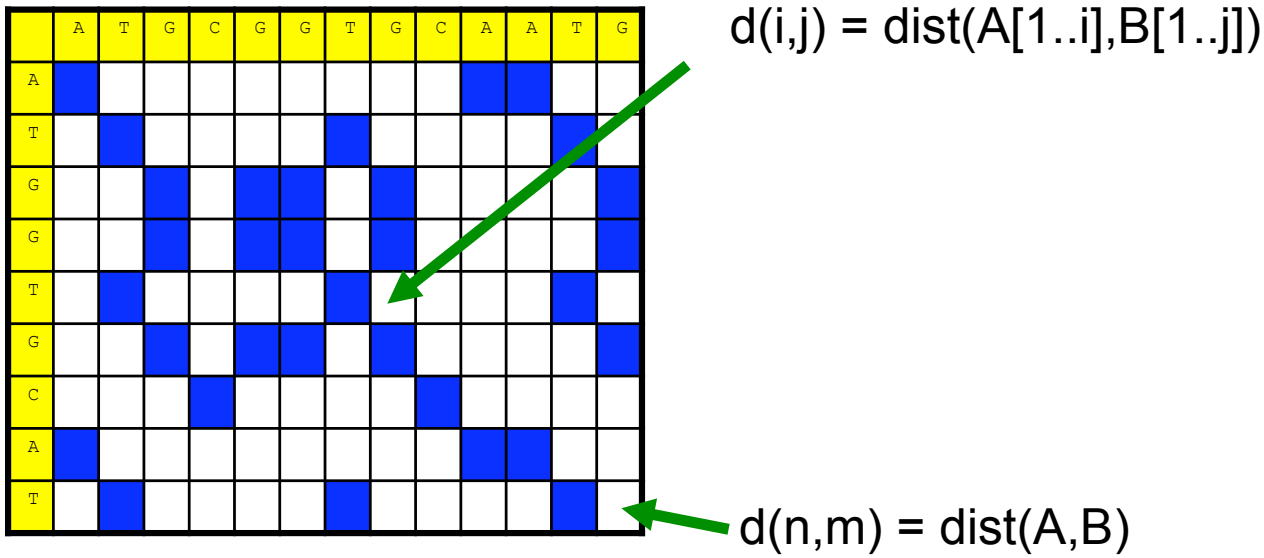
```
IMMMMMD        DMMMMMI
_AGAGAG        AGAGAG_
GAGAGA_        _GAGAGA
```

Berechnung des Editabstands

- Definition

  Gegeben zwei Strings A, B mit |A|=n, |B|=m

  - *Funktion dist(A,B) berechnet den Editabstand von A, B*
  - *Funktion d(i,j), 0≤i≤n und 0≤j≤m, berechnet den Editabstand zwischen A[1..i] und B[1..j]*



Rekursive Berechnung 1

- Wir betrachten die Berechnung von d(i,j) für A,B
  - Wir haben die optimalen Editskripte für $A[1..i_0]$ mit $B[1..j_0]$, $i_0 \leq i \wedge j_0 \leq j \wedge \neg(i_0=i \wedge j_0=j)$, berechnet
  - Wie kann das Editskript weitergeführt werden?
- Fallunterscheidung
  - I. Entspricht einer Insertion in A (oder Deletion in B)
    - Situation:

      ```
      ???I
      XXX_
      XXXT
      ```

    - Also benutzen wir ein Zeichen von B
    - d(i,j-1) ist der Editabstand von A[1..i] zu B[1..j-1]
      - Symbolisiert durch die XXX
    - Damit: d(i,j) = d(i, j-1) + 1

Rekursive Berechnung 2

- Wir betrachten die Berechnung von d(i,j) für A,B
  - Wir haben die optimalen Editskript für $A[1..i_0]$ mit $B[1..j_0]$, $i_0 \leq i \wedge j_0 \leq j \wedge \neg(i_0=i \wedge j_0=j)$, berechnet
  - Wie kann das Editskript weitergeführt werden?
- Fallunterscheidung
  - D. Entspricht einer Deletion in A (oder Insertion in B)
    - Situation:

      ```
      ???D
      XXXA
      XXX_
      ```

    - Umgekehrte Situation
    - Wir benutzen ein Zeichen von A
    - d(i-1,j) ist der Editabstand von A[1..i-1] zu B[1..j]
    - Damit: d(i,j) = d(i-1, j) + 1

Rekursive Berechnung 3

- Wir betrachten die Berechnung von d(i,j) für A,B
  - Wir haben die optimalen Editskript für $A[1..i_0]$ mit $B[1..j_0]$, $i_0 \leq i \wedge j_0 \leq j \wedge \neg(i_0=i \wedge j_0=j)$, berechnet
  - Wie kann das Editskript weitergeführt werden?
- Fallunterscheidung
  - M. Entspricht einem Match
    - Situation:

```
???M
XXXT
XXXT
```

    - Wir benutzen ein Zeichen von A und eines von B
    - Match kostet nichts
    - Damit: d(i,j) = d(i-1, j-1)

Rekursive Berechnung 4

- Wir betrachten die Berechnung von d(i,j) für A,B
  - Wir haben die optimalen Editskript für A[1..i_0] mit B[1..j_0], $i_0 \leq i \wedge j_0 \leq j \wedge \neg(i_0=i \wedge j_0=j)$, berechnet
  - Wie kann das Editskript weitergeführt werden?
- Fallunterscheidung
  - R. Entspricht einem Mismatch (Replace)
    - Situation:

```
???R
XXXA
XXXT
```

    - Wir benutzen ein Zeichen von A und eines von B
    - Mismatch kostet 1
    - Damit: d(i,j) = d(i-1, j-1) + 1

Rekursionsgleichung

- Wir leiten das nächste Symbol im Editskript aus schon bekannten Editabständen ab
- Wir suchen das kürzeste Skript, also

$$d(i,j) = \min \begin{cases} d(i,j-1)+1 \\ d(i-1,j)+1 \\ d(i-1,j-1)+t \end{cases}$$

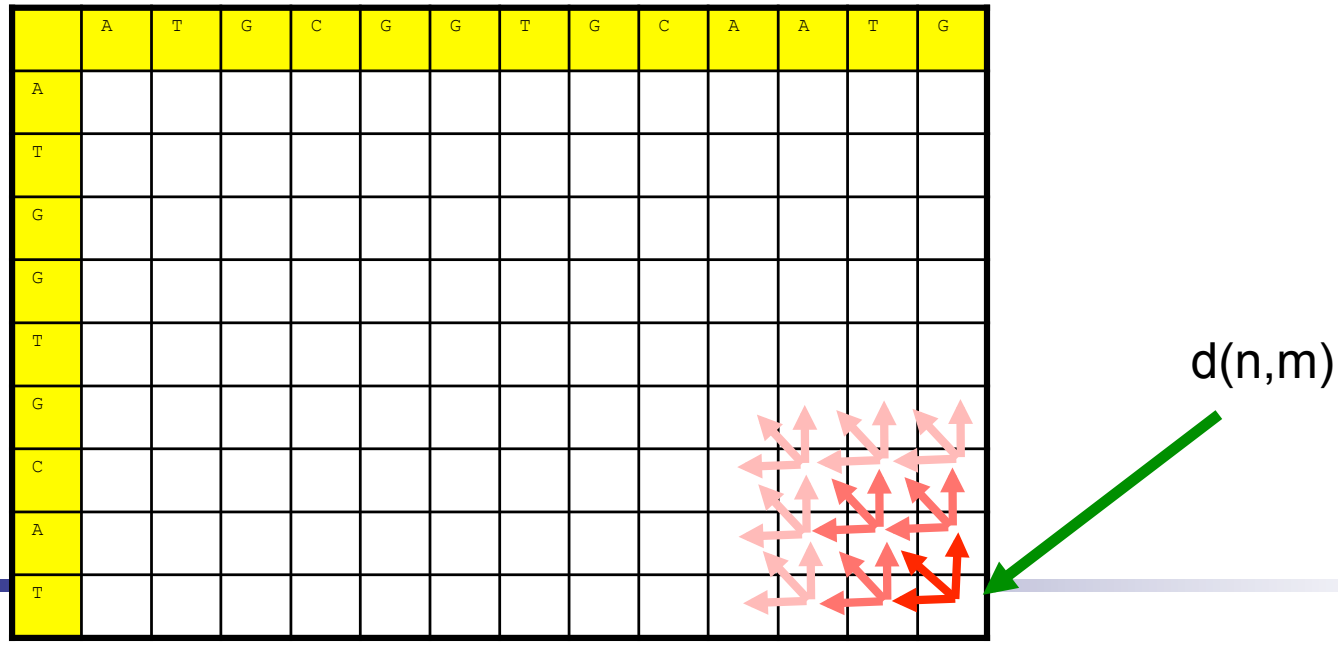
$$t(i,j) = \begin{cases} 1 : wenn\, A[i] \neq B[j] \\ 0 : sonst \end{cases}$$

Randbedingungen

- Randbedingungen nicht vergessen
  - d(i,0) = i
    - Um A[1..i] zu „_“ zu transformieren braucht man i Deletions
  - d(0,j) = j
    - Um A[1..0] zu B[1..j] zu transformieren braucht man j Insertions

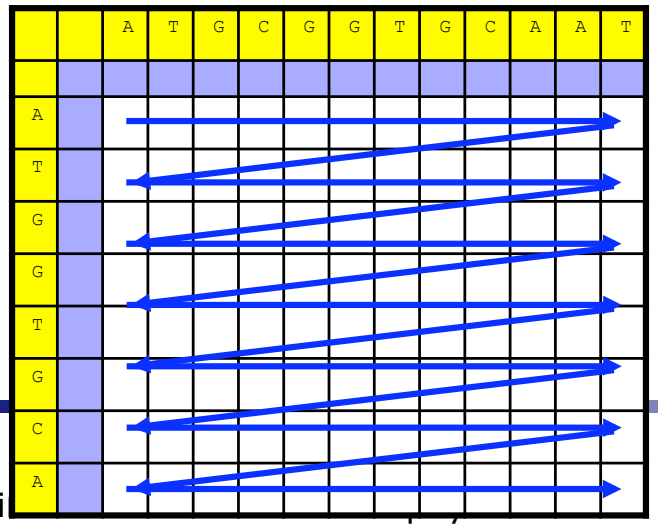
Berechnung des Editabstands

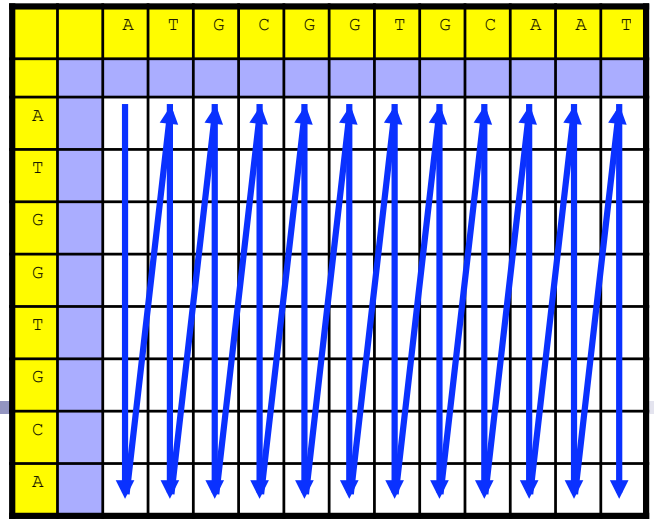
- Möglichkeit 1: Rekursiv
  - Ich will d(i,j) berechnen, dann berechne erst einmal
    - d(i-1,j)
    - d(i,j-1)
    - d(i-1,j-1)



Tabellarische Berechnung

- Grundidee
  - Speichern der Teillösungen in Tabelle
  - Bei Berechnung Wiederverwendung wo immer möglich
- Aufbau der Tabelle: Bottom-Up (statt rekursiv Top-Down)
  - Initialisierung mit festen Werten d(i,0) und d(0,j)
  - Sukzessive Berechnung von d(i,j) mit steigendem i,j
  - Für d(i,j) brauchen wir d(i,j-1), d(i-1,j) und d(i-1,j-1)
  - Verschiedene Reihenfolgen möglich





Erst mal von Hand

- A = ACGTTC
- B = ACCGT

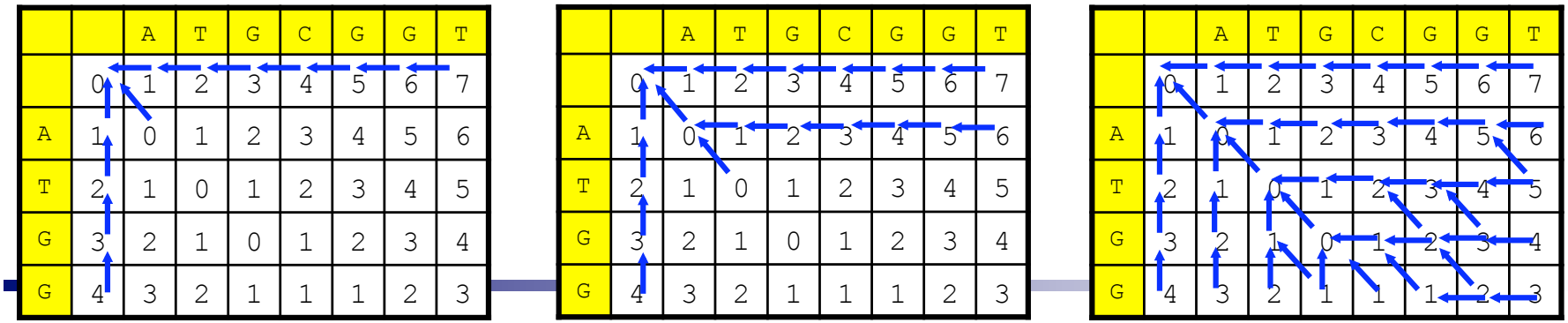
Jetzt programmiert

- Mit Programmiersprache Java
- In Entwicklungsumgebung Eclipse
  - Initialisierung des zwei-dimensionalen Arrays
  - Belegung der ersten Reihe und Spalte
  - Berechnen der d(i,j)-Werte

Von der Matrix zum Alignment

		A	T	G	C	G	G	T
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
T	2	1	0	1	2	3	4	5
G	3	2	1	0	1	2	3	4
G	4	3	2	1	1	1	2	3

- Editabstand ist 3
- Wir suchen aber ein Alignment, nicht nur den Abstand
- Extraktion aus der Tabelle durch „Tracebacking“
  - Bei Berechnung von d(i,j) behalte Pointer auf minimale Vorgängerzelle(n)
  - Die muss nicht eindeutig sein



Umsetzung in Java

- Pointer speichern
  - Drei-dimensionaler Array
    - dritte Dimension für die Richtung
- Berechnen beim durchlaufen, woher ich gekommen sein kann
  - von links oder oben
    - wenn ich noch nicht ganz links oder oben bin
    - wenn d(i-1,j)+gap = d(i,j) / d(i,j-1) + gap = d(i,j)
  - von diagonal
    - wenn ich weder ganz links noch oben bin
    - wenn d(i-1,j-1) + t(A[i], B[j]) = d(i,j)

Umsetzung in Java – cont. –

- Rekursive Funktion

```
function printAlignments(int x, int y, String A, String B)  {

  if (x == 0 && y == 0) {
    println(A);
    println(B);
  }
  else  {
    if ( x>0 && y>0 && d[x-1][y-1]+t(S1[x],S2[y]) == d[x][y] )  {
      printAlignment(x-1, y-1, S1[x]+A, S2[y]+B);
    }
    if (x>0 && d[x-1][y]+gap == d[x][y])  {
      printAlignment(x-1, y, S1[x]+A, "_"+B);
    }
    if (y>0 && d[x][y-1]+gap == d[x][y])  {
      printAlignment(x, y-1, "_"+A, S2[y]+B);
    }
    else { return; }
  }
}
```

Exercise 4c

- ACCGTTGACCACACACACAG + CACA
  - 73 Alignments
- TTTTTTTTTT + TTTTTTTTTT
  - 10 Alignments
- TTTTTTTTTTTT + TTTTTTT
  - 330 Alingments

Exercise 2

- String A, length n
- String B, length m
- Assumption m > n

- Lösung: Permutation mit Gruppen nicht unterscheidbarer Elemente

$$Anzahl = \frac{m!}{(m-n)!\,n!}$$

Exercise 3

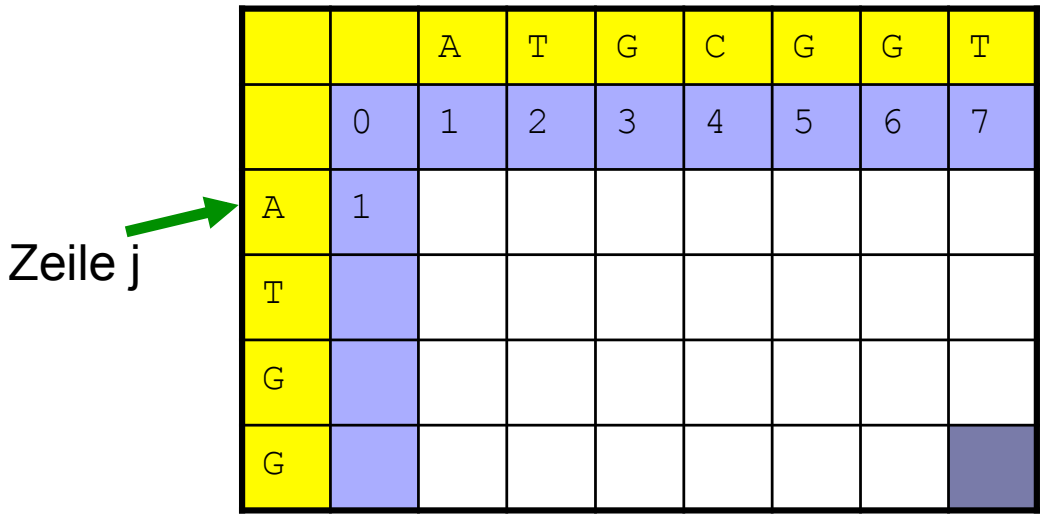
- Database search and alignment

- Alignment of sequences
  - AJ271740: 70,398 bp
  - AJ277892: 294,540 bp

- PROBLEM:
  - Die Matrix für die Alignments passt nicht mehr in den Hauptspeicher!
- LÖSUNG:
  - Dynamische Programmierung mit linearem Platzbedarf

Alignment mit linearem Platzbedarf

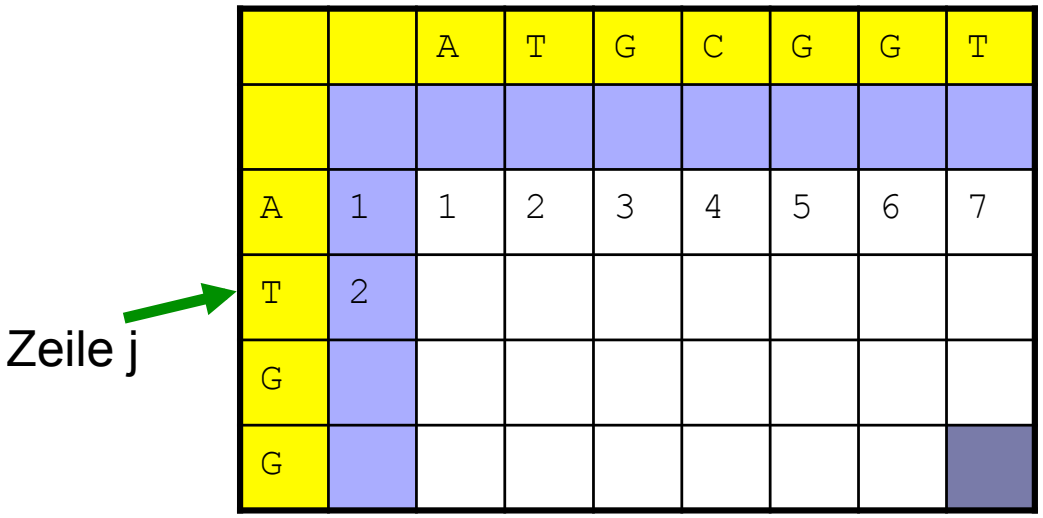
- Wenn ich Zeile j ausrechnen will brauche ich
  - Zeile j-1
  - Zelle d(0,j)



		A	T	G	C	G	G	T
	0	1	2	3	4	5	6	7
A	1							
T								
G								
G								

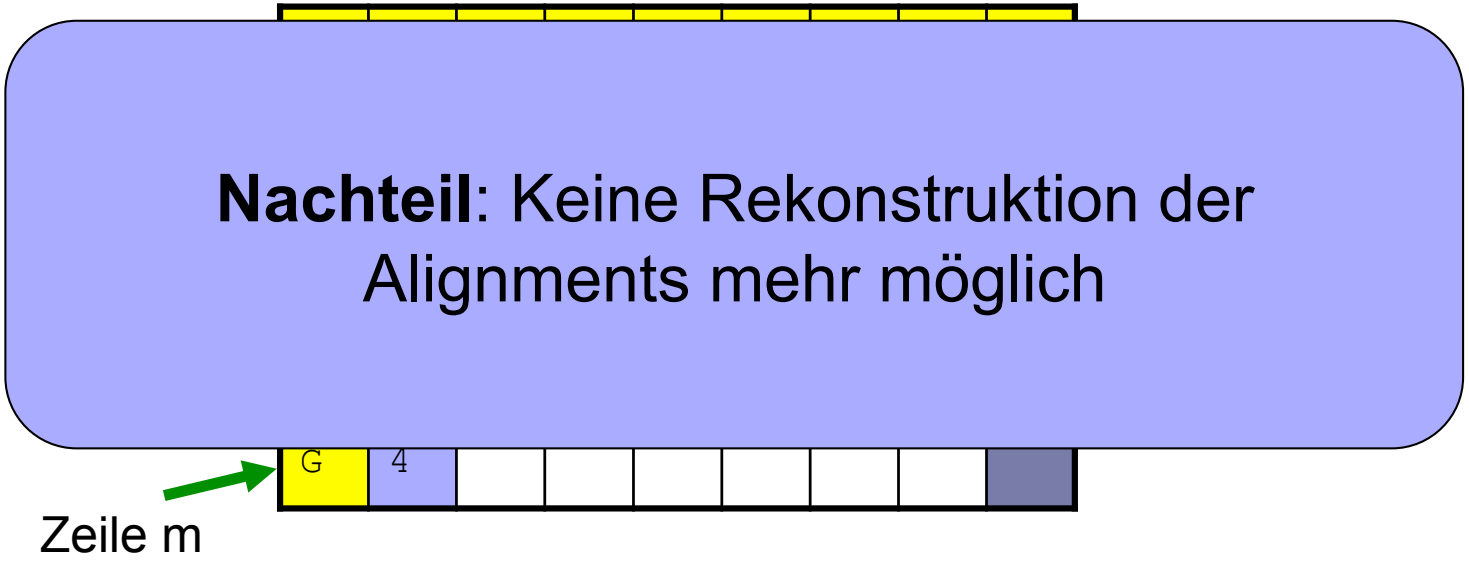
Alignment mit linearem Platzbedarf

- Wenn ich Zeile j ausrechnen will brauche ich
  - Zeile j-1
  - Zelle d(0,j)



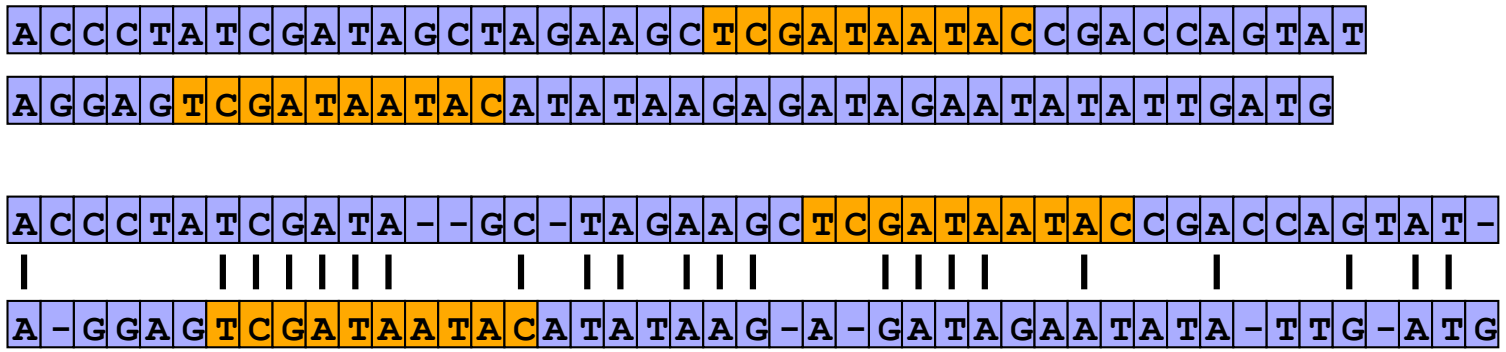
Alignment mit linearem Platzbedarf

- Wenn ich Zeile j ausrechnen will brauche ich
  - Zeile j-1
  - Zelle d(0,j)



Lokales und globales Alignment

- Bisher: Globale Alignments
  - Beide Sequenzen werden komplett betrachtet
- Das entspricht oft nicht der biologischen Frage
  - Funktion wird nur durch manche Sequenzblöcke bestimmt (Gene, Exons, Proteindomänen, ...)
- Wir suchen meistens ähnliche Teilsequenzen
  - Funktionstragende Sequenzblöcke
  - „Lokale“ Alignments

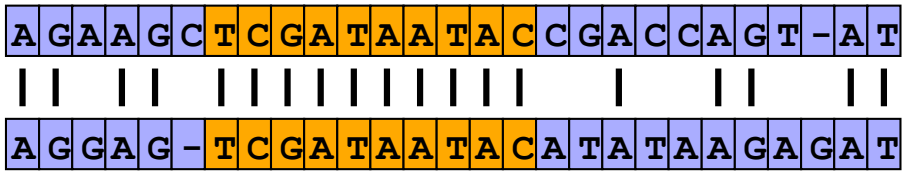


Formal: Lokale Alignments

- Definition. *Gegeben zwei Strings A, B.*
  - *Seien a,b Substrings mit a⊆A, b⊆B so dass*

$$sim(a,b) = \max_{\forall a' \in A, b' \in B} \left(sim(a',b') \right)$$

  - *Das vom (globalen) Alignment von a und b induzierte Alignment von A und B heißt lokales Alignment von A und B*
  - *Der lokale Ähnlichkeitsscore* $dist_{local}(A,B) = sim(a,b)$
- Bemerkung
  - Unempfindlich gegen unterschiedliche lange Strings
- Beispiel
  - Lokales A. findet den identischen Substring



Optimales lokales Alignment

- Theorem
  Gegeben Strings A,B. Mit der folgenden Funktion d(i,j), mit 0≤i≤n und 0≤j≤m

$$d(i,j) = \max \begin{Bmatrix} 0 \\ d(i,j-1) + gap \\ d(i-1,j) + gap \\ d(i-1,j-1) + t(A[i], B[j]) \end{Bmatrix}$$

  - *Gilt:*

$$dist_{local}(A,B) = \max_{i,j}(d(i,j))$$

- Traceback
  - Starte beim maximalen Wert in der Matrix
  - Verfolge beliebigen Pfad bis zu einer Zelle mit Wert 0

Globale vs. Lokale Alignments

- Minimieren oder Maximieren?
- Global (Editabstand)
  - Minimieren des Abstands
    - möglichst wenige I / D / R
- Lokal
  - Maximieren der Ähnlichkeit
    - möglichst viele M
- Austauschbar

Globale vs. Lokale Alignments

- Initialisierung

- Global
  - d(i,0) = i * gap;
  - d(0,j) = j * gap;

- Lokal
  - d(i,0) = 0
  - d(0,j) = 0

Globale vs. Lokale Alignments

- Ende des Alignments (Beginn des Tracebacks)

- Global
  - Start bei d(n,m)

- Lokal
  - Start bei den Zellen d(i,j), für die gilt, dass der Wert d(i,j) maximal innerhalb der Matrix ist

Globale vs. Lokale Alignments

- Beginn des Alignments (Ende des Tracebacks)
- Global
  - bei d(0,0)
- Lokal
  - sobald d(i,j) = 0

Zuletzt!

Weitere

Fragen ?