

# Sequenz - Alignment

10.11.03

Vorlesung Bioinformatik 1

Molekulare Biotechnologie

Dr. Rainer König

Besonderen Dank an Mark van der Linden, Mechthilde Falkenhahn und  
der Husar Biocomputing Service Gruppe für die Unterstützung bei  
meinen Folien

# Warum Sequenz-Vergleich?

## Biologische Hintergrund

Viele Gene und Proteine sind Mitglieder von Familien mit ähnlicher biochemischer Funktion und/oder gemeinsamer evolutionärer Herkunft

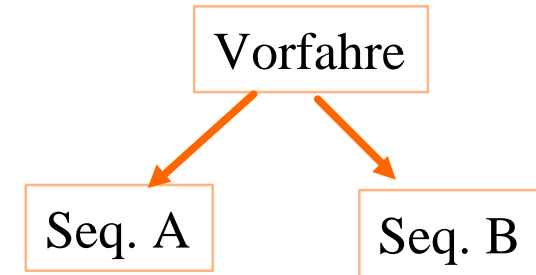
## Der Sequenzvergleich wird verwendet,

- um funktionelle, strukturelle oder evolutionäre Beziehungen zu entdecken  
z.B. Ähnliche Sequenz => ähnliche Funktion, Struktur, ...
- um konservierte Muster zu identifizieren
- wenn man über ein unbekanntes Protein etwas wissen will:  
Finden von Proteinen mit ähnlichen Domänen  
=> Schließen auf ähnliche Funktion

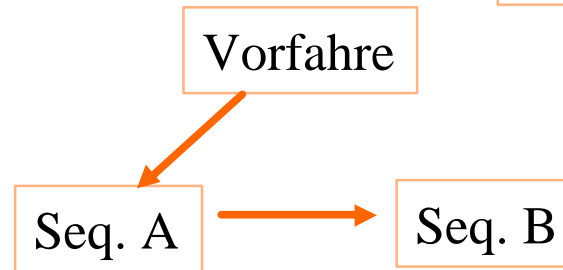
Ein Vergleich kann Ausgangsbasis zu weiteren Untersuchungen sein

# Ursache der Sequenzähnlichkeit

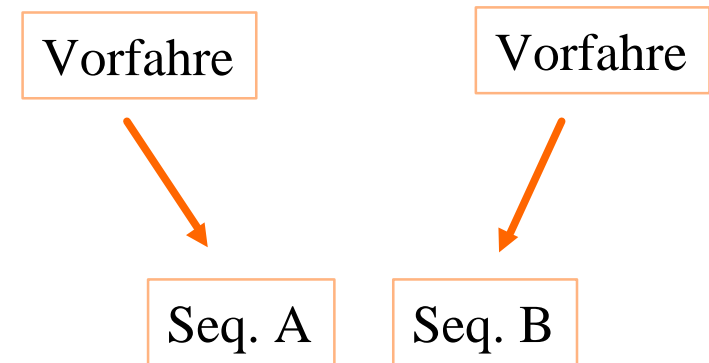
- evolutionärer Zusammenhang (Orthologe)



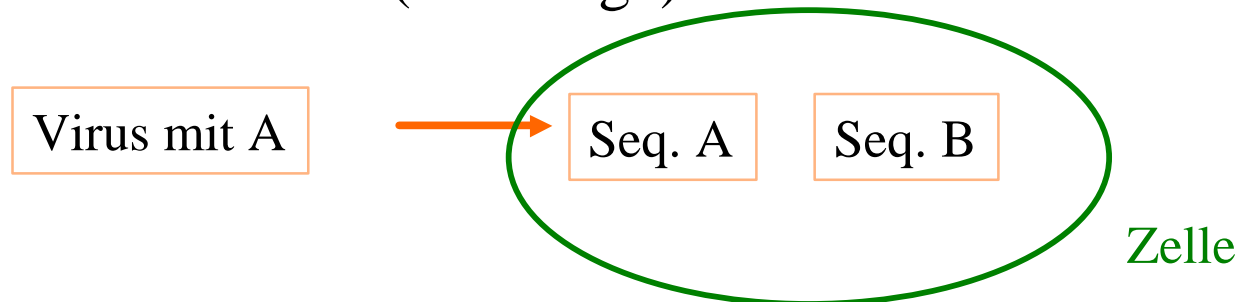
- Gen-Duplizierung (Paraloge)



- Evolutionäre Konvergenz (Analoge)

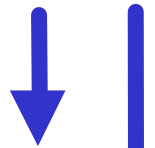


- Gen-Transfer (Xenologe)



# Typen von Sequenzvergleichen

- Paarweises Alignment



- Multiples Alignment



- Datenbank-Suchen

# viele Möglichkeiten, Sequenzen zu Alignieren ...

```
actaccagttcatttgatacttctcaaa
taccattaccggtgtaactgaaaggacttaaagact
```

```
actaccagttcatttgatacttctcaaa
taccattaccggtgtaactgaaaggacttaaagact
```

1. Sequenz
2. Sequenz

```
actaccagttcatttgatacttctcaaa
      | |       |       ||
taccattaccggtgtaactgaaaggacttaaagact
```

```
actaccagttcatttgatacttctcaaa
| | | | | | | | | | | | | | | |
taccattaccggtgtaactgaaaggacttaaagact
```

```
actaccagttcatttgatacttctcaaa
| | | | | | | | | | | | | | | |
taccattaccggtgtaactgaaaggacttaaagact
```

```
actaccagttcatttgatacttctcaaa
| | | | | | | | | | | | | | | |
taccattaccggtgtaactgaaaggacttaaagact
```

```
actaccagttcatttgatacttctcaaa
| | | | | | | | | | | | | | | |
taccattaccggtgtaactgaaaggacttaaagact
```

```
actaccagttcatttgatacttctcaaa
| | | | | | | | | | | | | | | |
taccattaccggtgtaactgaaaggacttaaagact
```

```
actaccagttcatttgatacttctcaaa
| | | | | | | | | | | | | | | |
taccattaccggtgtaactgaaaggacttaaagact
```

```
actaccagttcatttgatacttctcaaa
| | | | | | | | | | | | | | | |
taccattaccggtgtaactgaaaggacttaaagact
```

# Erste Erkenntnisse zur Sequenz- Alignierung

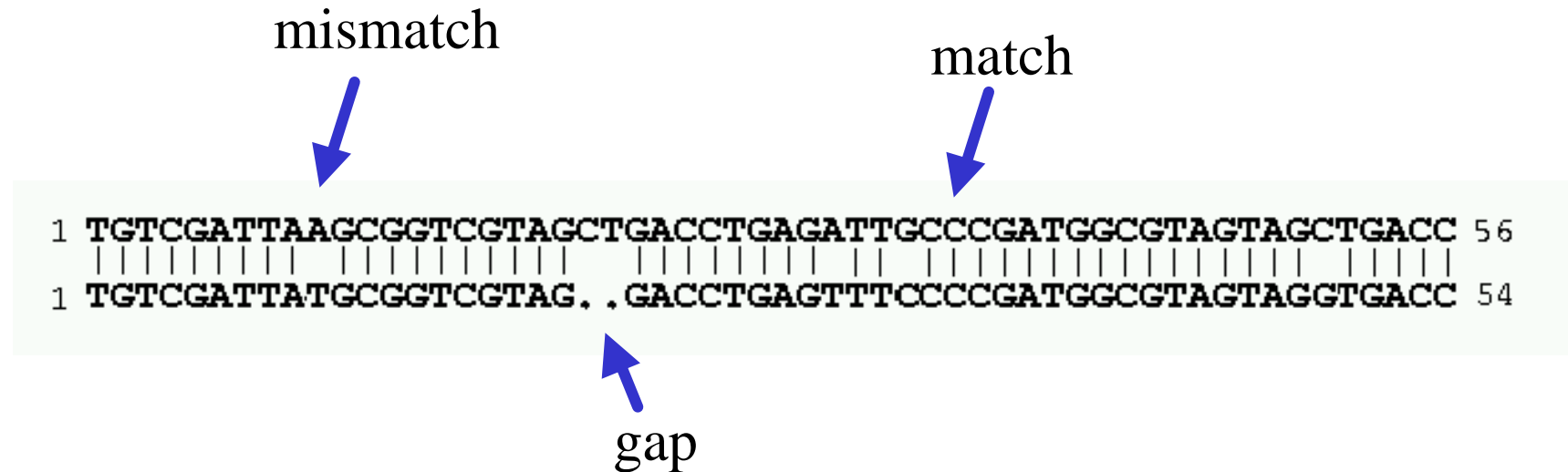
- Viele Alignierungen sind denkbar
- Zwei Sequenzen lassen sich *immer* alignieren
- => Sequenz-Alignments müssen **bewertet** werden ("Score")
- Oft kann es mehr als eine Lösung zur gleichen Bewertung geben

# Paarweiser Sequenzvergleich

- Globale Alignments

- Lokale Alignments

# Globale Alignments



- es wird ein end-to-end Alignment erzeugt
- Beispiel-Programm: "**GAP**" (*Needleman & Wunsch*)



# Lokales Alignment

- finde Region mit höchster lokaler Ähnlichkeit
- Programm dazu: **Bestfit** (*Smith-Waterman*)

```
14 TCAGAAAGCAGCTAAAGCGT 32
   ||| ||| ||| ||| ||| |||
42 TCAGAAAGCA . CTAAAGCGT 59
```

# Lokales Alignment

- wenn man an allen Regionen mit hoher Ähnlichkeit interessiert ist:
- **SIM, Similarity** (*X. Huang*)

```
14 TCAAGAAGCAGCTAAAGCGT 32
   ||||| |||||
42 TCAAGAAGCA . CTAAAGCGT 59
```

```
1 AGGATTGGAATGCT 14
   ||||| |||||
1 AGGATTGGAATGCT 14
```

```
39 AGGATTGGAAT 49
   ||||| |||||
1 AGGATTGGAAT 11
```

```
62 AGACCG 67
   |||||
66 AGACCG 71
```

# Einige Bemerkungen zu globalen und lokalen Alignments

- global:
- die gesamte Sequenz wird aligniert
  - so viele Matches wie möglich werden gefunden
  - Sequenzen sollten ähnliche Länge aufweisen

- lokal:
- Sequenzabschnitte mit hoher Match-Dichte werden gefunden
  - Sequenzen müssen nur in Teilabschnitten ähnlich sein (z.B. Domänen, konservierte Bereiche)
  - Sequenzen können verschieden lang sein
  - Es existieren statistische Signifikanz-Methoden (siehe Vorlesung Alignment-Statistik)



**lokales Alignment wird sehr viel häufiger verwendet**

# Wie komme ich zu einem guten Alignment ?

- Was ist ein gutes Alignment => **Bewertung**
- Algorithmus

# Scoring für Nukleotid-Sequenzen

1. Sequenz

2. Sequenz

```
actaccagttcatttgatacttctcaaa
```

```
      | |      |      ||  
taccattaccgtgttaactgaaaggacttaaagact
```

Match: 1

Mismatch: 0

Score = 5

# Scoring für Nukleotid-Sequenzen

1. Sequenz

2. Sequenz

```
actaccagttcatttgatacttctcaaa
      | |      |      ||
taccattaccgtgttaactgaaaggacttaaagact
```

Match: 1  
Mismatch: 0  
Score = 5

	<b>A</b>	<b>G</b>	<b>C</b>	<b>T</b>
<b>A</b>	1	0	0	0
<b>G</b>	0	1	0	0
<b>C</b>	0	0	1	0
<b>T</b>	0	0	0	1



# Scoring für Proteine

1. Sequenz

2. Sequenz

PTHPLASKTQILPEDLASEDLTI  
 ||||| | | ||  
 PTHPLAGERAIGLARLAEEDFGM

Scoring matrix

	C	S	T	P	A	G	N	D	.	.
C	9									
S	-1	4								
T	-1	1	5							
P	-3	-1	-1	7						
A	0	1	0	-1	4					
G	-3	0	-2	-2	0	6				
N	-3	1	0	-2	-2	0	5			
D	-3	0	-1	-1	-2	-1	1	6		
.										
.										

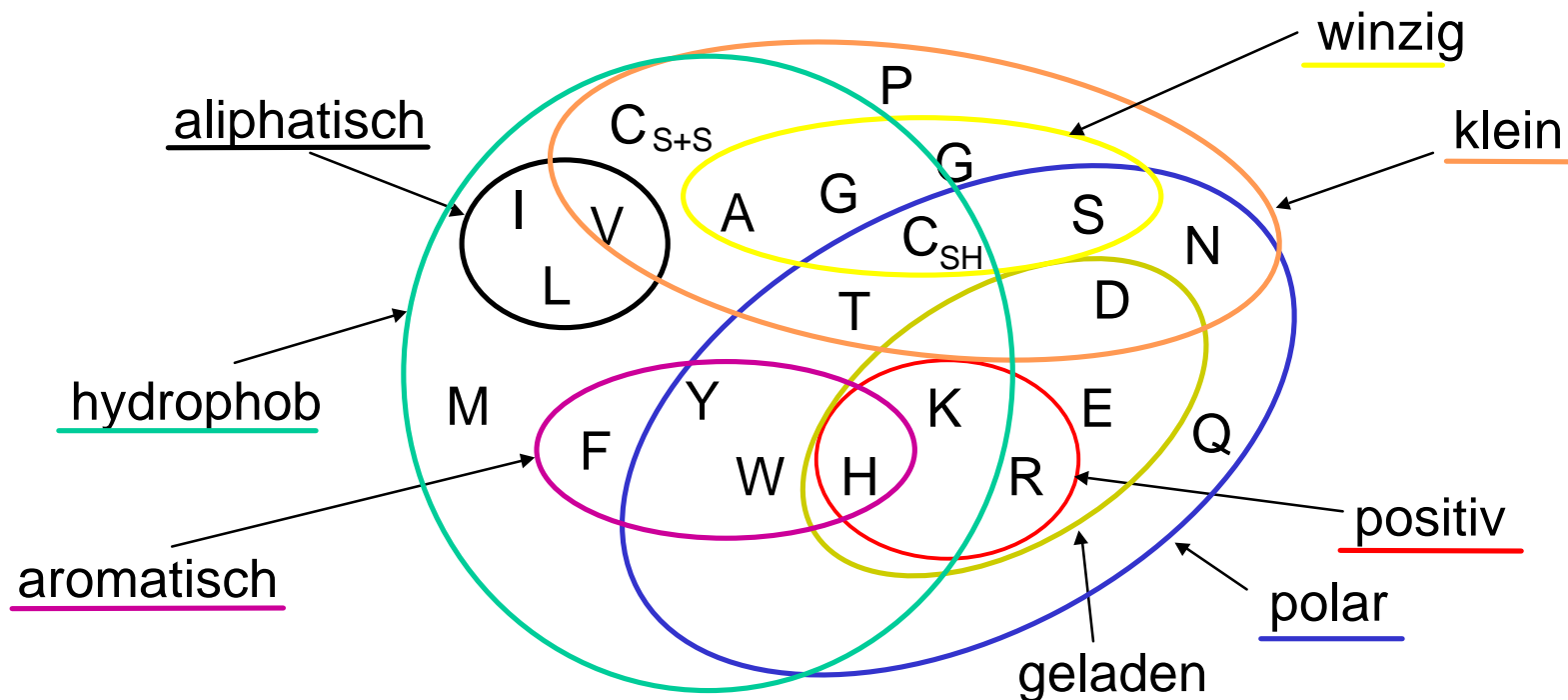
T:G = -2

T:T = 5

Score = 48

# Scoring für Proteine

- Aminosäuren haben verschiedene biochemische und physikalische Eigenschaften, wodurch ihre relative Vertauschbarkeit im Laufe der Evolution beeinflusst wird



=> z.B. Score von I - L > Score von L - Q (Glutamin)

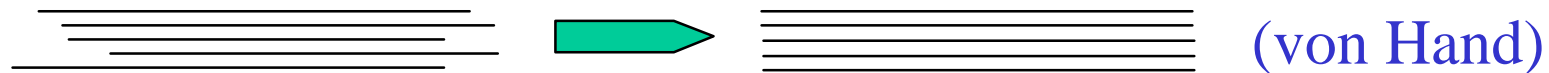
# Scoring für Proteine

- Aminosäuren haben verschiedene biochemische und physikalische Eigenschaften, wodurch ihre relative Vertauschbarkeit im Laufe der Evolution beeinflusst wird
- Score-Matrizen enthalten
  - Wahrscheinlichkeit gegenseitiger Vertauschung,
  - Wahrscheinlichkeit, dass die jeweiligen Aminosäuren überhaupt auftreten (seltene Aminosäuren entstehen selten)
- Weit verbreitet:
  - PAM
  - BLOSUM

# PAM-Matrix (Percent Accepted Mutations)

- berechnet durch globale Alignments von Proteinfamilien

(nahe verwandt: mindestens 85% identitisch) *(Dayhoff et al., 1978)*



- Anzahl der Austausche für jedes Aminosäurepaar wird berechnet und verglichen

z.B.

**L**   **V**   **V**   **V**

⇒ L <-> I sehr wahrscheinlich ⇒

**I**   **I**   **T**   **V**

(L,I) hoher Scorewert

**I**   **L**   **L**   **I**

⇒ V <-> T selten ⇒ (V,T)

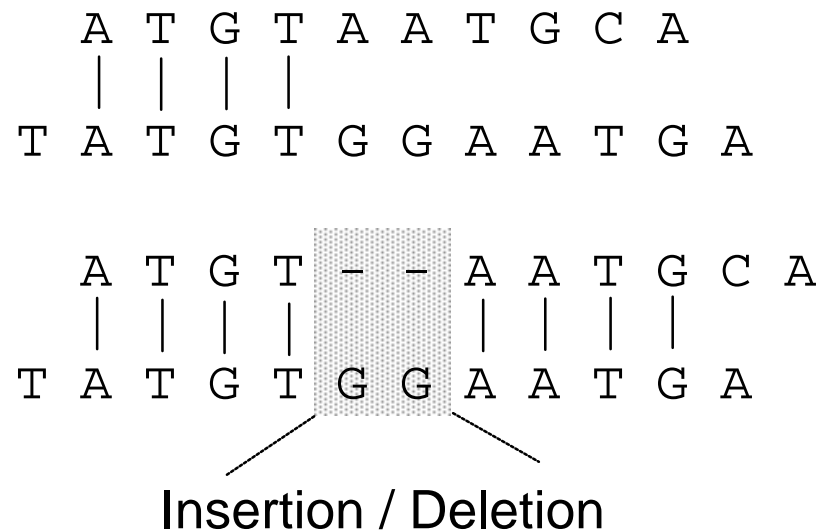
niedriger Scorewert

**I**   **I**   **V**   **L**

# PAM 250

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z
A	2	-2	0	0	-2	0	0	1	-1	-1	-2	-1	-1	-3	1	1	1	-6	3	0	2	1
R	-2	6	0	-1	-4	1	-1	-3	2	-2	-3	3	0	-4	0	0	-1	2	-4	-2	1	2
N	0	0	2	2	-4	1	1	0	2	-2	-3	1	-2	-3	0	1	0	-4	-2	-2	4	3
D	0	-1	2	4	-5	2	3	1	1	-2	-4	0	-3	-6	-1	0	0	-7	-4	-2	5	4
C	-2	-4	-4	-5	12	-5	-5	-3	-3	-2	-6	-5	-5	-4	-3	0	-2	-8	0	-2	-3	-4
Q	0	1	1	2	-5	4	2	-1	3	-2	-2	1	-1	-5	0	-1	-1	-5	-4	-2	3	5
E	0	-1	1	3	-5	2	4	0	1	-2	-3	0	-2	-5	-1	0	0	-7	-4	-2	4	5
G	1	-3	0	1	-3	-1	0	5	-2	-3	-4	-2	-3	-5	0	1	0	-7	-5	-1	2	1
H	-1	2	2	1	-3	3	1	-2	6	-2	-2	0	-2	-2	0	-1	-1	-3	0	-2	3	3
I	-1	-2	-2	-2	-2	-2	-2	-3	-2	5	2	-2	2	1	-2	-1	0	-5	-1	4	-1	-1
L	-2	-3	-3	-4	-6	-2	-3	-4	-2	2	6	-3	4	2	-3	-3	-2	-2	-1	2	-2	-1
K	-1	3	1	0	-5	1	0	-2	0	-2	-3	5	0	-5	-1	0	0	-3	-4	-2	2	2
M	-1	0	-2	-3	-5	-1	-2	-3	-2	2	4	0	6	0	-2	-2	-1	-4	-2	2	-1	0
F	-3	-4	-3	-6	-4	-5	-5	-5	-2	1	2	-5	0	9	-5	-3	-3	0	7	-1	-3	-4
P	1	0	0	-1	-3	0	-1	0	0	-2	-3	-1	-2	-5	6	1	0	-6	-5	-1	1	1
S	1	0	1	0	0	-1	0	1	-1	-1	-3	0	-2	-3	1	2	1	-2	-3	-1	2	1
W	1	-1	0	0	0	0	0	-1	0	-2	0	-1	-3	0	1	3	0	1	3	0	2	1
Y	-6	-2	-4	-7	-8	-7	-7	-3	-5	-2	-3	-4	0	-6	-2	-5	-5	0	10	-2	-2	-3
V	-3	-4	-2	-4	-4	-4	-5	0	-1	-1	-4	-2	7	-5	-3	-3	0	10	-2	-2	-3	
B	0	-2	-2	-2	-2	-2	-2	-1	-2	4	2	-2	2	-1	-1	-1	0	-6	-2	4	0	0
Z	2	1	4	5	-3	3	4	2	3	-1	-2	2	-1	-3	1	2	2	-4	-2	0	6	5
Z	1	2	3	4	-4	5	5	1	3	-1	-1	2	0	-4	1	1	1	-4	-3	0	5	6

## Score von Einfügungen und Auslassungen (Gaps)



Einfügen eines Gaps wird mit einem negativen Score-Wert bestraft

# Höhe der Gapkosten

1. Gaps zu teuer => "karge" Ausbeute

```
1 GTGATAGACACAGACCGGTGGCATTGTGGA 29
  |||   |   |                   |   ||
1 GTGTCGGGAAGCAGATAACTCCGATGGTTG 29
```

2. Gaps zu billig => gedehntes Alignment, u.U. unspezifisch

```
1 GTG.ATAG.ACA.CAGA..CCGGT..GGCCTTACGG 29
  ||| || | | | ||| || | | | |
1 GTGTAT.GGA.AGCAGATACC..TCCG...T....G 29
```

# Affine Gap-Kosten

Score mit linearer Bestrafung:

$$\gamma(g) = -gd$$

Score mit affiner Bestrafung:

$$\gamma(g) = -d - (g - 1)e$$

$\gamma(g)$  = Bestrafung für Gap mit Länge  $g$

$d$  = Bestrafung bei Öffnen des Gaps

$e$  = Bestrafung bei Verlängern des Gaps

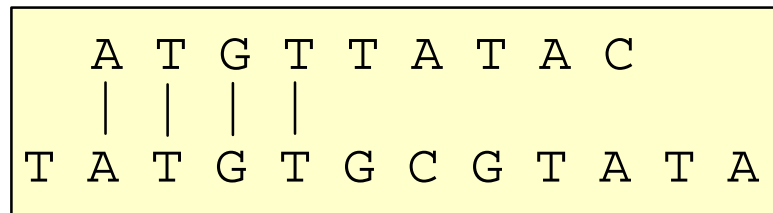
$g$  = Gap-Länge

# Affine Gap-Kosten

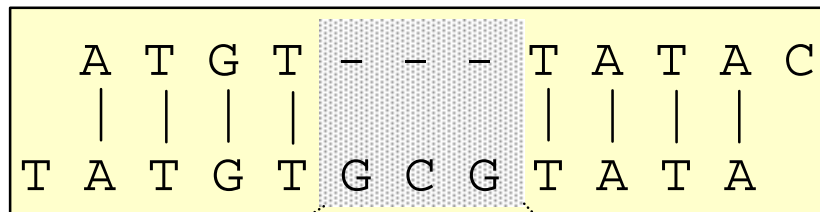
Match = 1

Mismatch = 0

Gesamt-Score: 4



Gesamt-Score:  $8 - 3.2 = 4.8$



insertion / deletion

Gap-Parameter:

$d = 3$  (gap opening)

$e = 0.1$  (gap extension)

$g = 3$  (gap length)

$\gamma(g) = -3 - (3 - 1) 0.1 = -3.2$

# Methoden des Sequenz-Alignments

- Dynamisches Programmieren  
gibt optimales Alignment,  
alle möglichen Kombinationen,  
cpu-intensiv
- Dotplot-Analyse  
erster Eindruck,  
zeigt Einfügungen/Auslassungen, Repeats
- "Wort"-Methoden  
Sammeln von "Inseln",  
schnell, aber heuristisch,  
wird für DB-Suchen verwendet (spezielle Vorlesung)

# Dynamisches Programmieren

Automatisierter Ablauf, Algorithmus, der das beste Alignment (mit optimalem Score) findet, abhängig von den gegebenen Parametern (Gap-Kosten, Scorematrix)

- Needleman - Wunsch Algorithmus
  - **Globales** Alignment
- Smith - Waterman Algorithmus
  - **Lokales** Alignment

# Needleman - Wunsch

(globales Alignment)

1. Sequenz: H E A G A W G H E E

2. Sequenz: P A W H E A E

Score-Matrix: Blosum-50-Matrix

Gap-Kosten: Lineare Gap-Kosten von 8

# Werte aus der Score-Matrix (BLOSUM 50)

	<b>H</b>	<b>E</b>	<b>A</b>	<b>G</b>	<b>A</b>	<b>W</b>	<b>G</b>	<b>H</b>	<b>E</b>	<b>E</b>
<b>P</b>	-2	-1	-1	-2	-1	-4	-2	-2	-1	-1
<b>A</b>	-2	-1	<b>5</b>	0	<b>5</b>	-3	0	-2	-1	-1
<b>W</b>	-3	-3	-3	-3	-3	<b>15</b>	-3	-3	-3	-3
<b>H</b>	<b>10</b>	0	-2	-2	-2	-3	-2	<b>10</b>	0	0
<b>E</b>	0	<b>6</b>	-1	-3	-1	-3	-3	0	<b>6</b>	<b>6</b>
<b>A</b>	-2	-1	<b>5</b>	0	<b>5</b>	-3	0	-2	-1	-1
<b>E</b>	0	<b>6</b>	-1	-3	-1	-3	-3	0	<b>6</b>	<b>6</b>

## **Dynamisches Programmieren geschieht in zwei Schritten:**

- 1. Erstellen der Alignment-(Pfad)-Matrix
- 2. Backtracking in der Matrix und aufaddieren der Score-Werte zum Gesamtscore-Wert

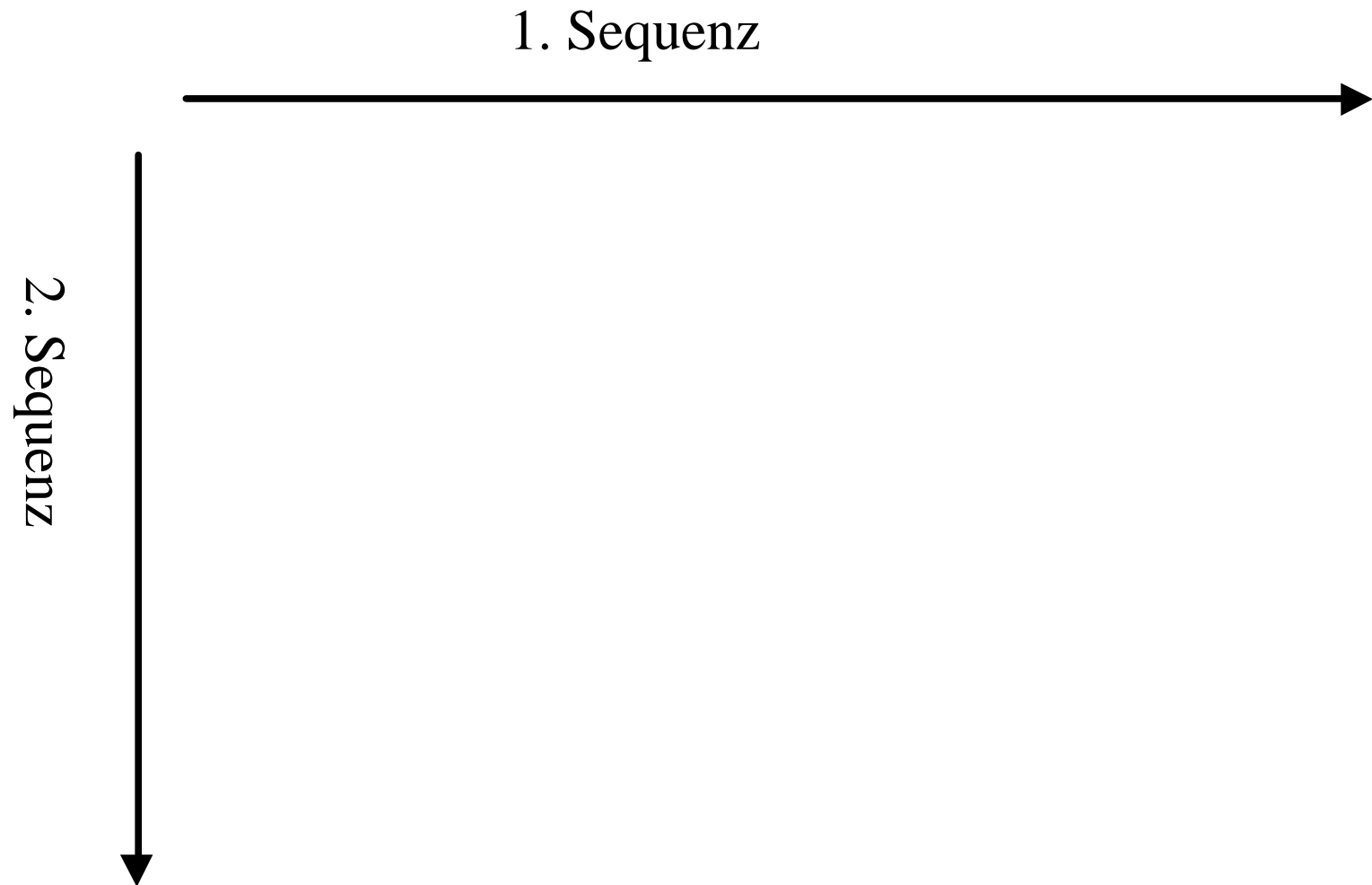
Idee: bilde das optimale Alignment durch:

- starten an einem Ende der Sequenzen
- sukzessives Anfügen einzelner Symbole zu optimalen Teillösungen, solange bis alle Symbole abgehandelt sind

Das Prinzip:

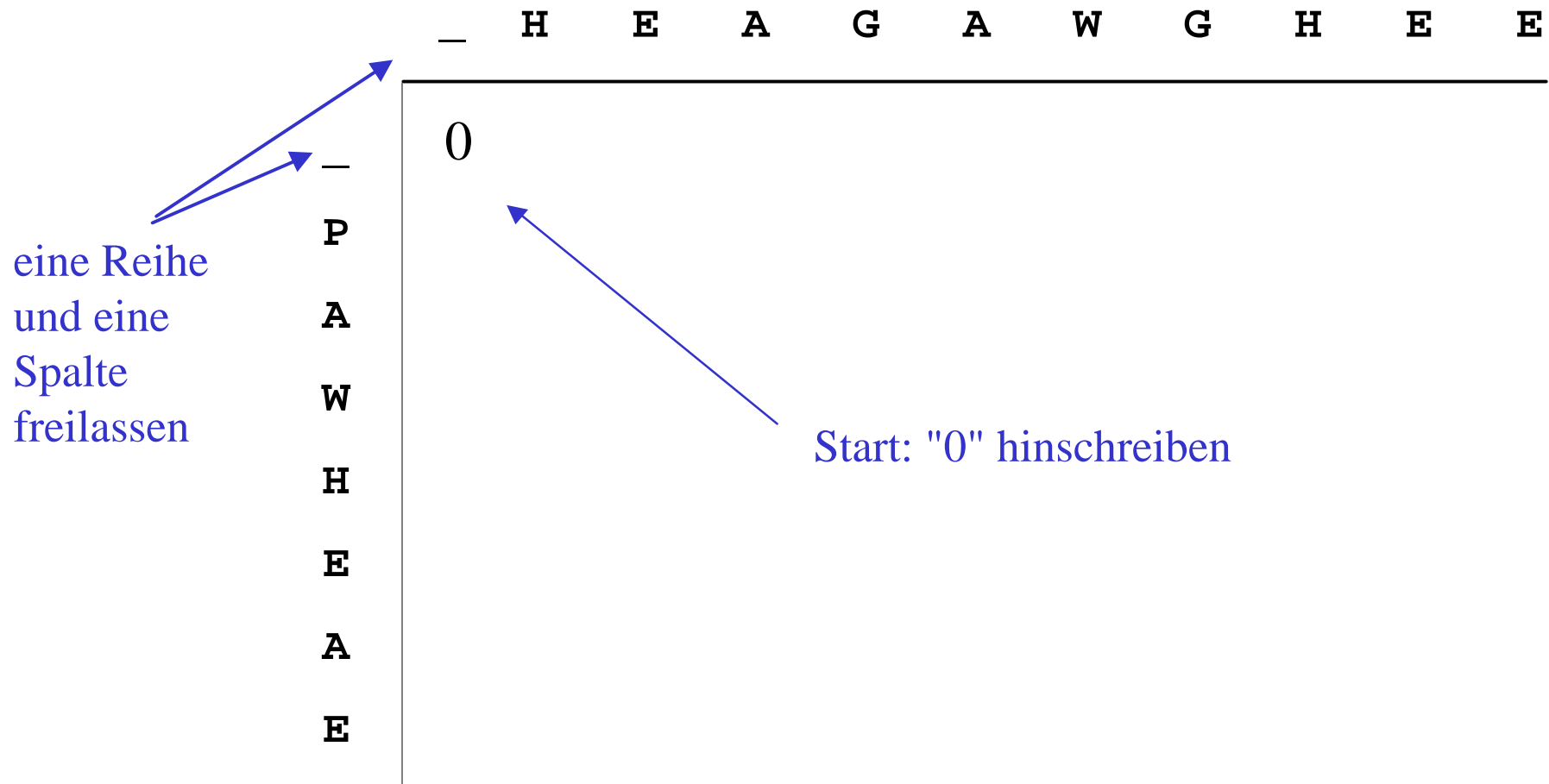
1. Die Alignment-Matrix

hinschreiben ...



# 1. Die Alignment-Matrix

hinschreiben ...



# 1. Die Alignment-Matrix

Wie kann das Alignment an der linken Seite beider Sequenzen aussehen?

=> Es gibt nur 3 Möglichkeiten:

1.     **H**                   Gap für 2. Sequenz, Score = -8

—

2.     **H**                   Mismatch, Score = -2

**P**

3.     —                    Gap für 1. Sequenz, Score = -8

**P**

# 1. Die Alignment-Matrix

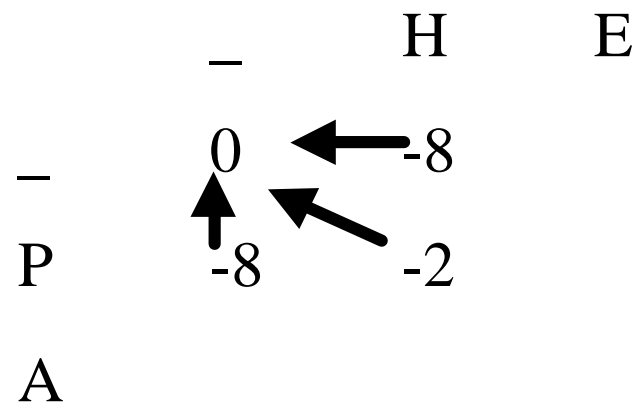
Eintragen aller Möglichkeiten in die Matrix, mit den jeweils (aufaddierten) Score-Werten,

	–	H	E
–	0	-8	
P	-8	-2	
A			

# 1. Die Alignment-Matrix

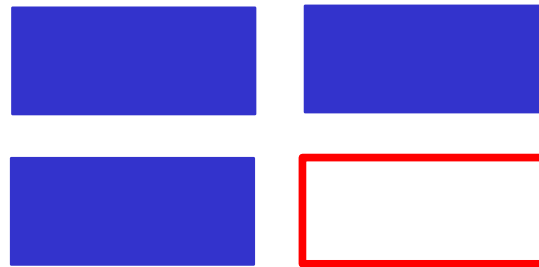
-zeichne die Richtung rein, von wo die neue Zahl berechnet wurde

- der Pfeil wird "falschrum" reingeschrieben:



# 1. Die Alignment-Matrix

jetzt wird die Matrix von links oben bis rechts unten nacheinander aufgebaut, jeweilige Ecken können berechnet werden:



blau: gegeben, mit diesen Werten wird der Wert für das rote Kästchen berechnet

rot: kann berechnet werden

Bem. zu den Rändern: wenn es zu einem zu berechnenden Kästchen (rot) keine linken oder rechten Kästchen gibt, dann nehme nur die (blauen) Kästchen, die es gibt



# Die fertige Matrix

		H	E	A	G	A	W	G	H	E	E
	0	← -8	← -16	← -24	← -32	← -40	← -48	← -56	← -64	← -72	← -80
P	-8	↗ -2	↘ -9	↙ -17	↘ -25	↙ -33	↘ -42	↙ -49	↘ -57	↙ -65	↘ -73
A	-16	↑ -10	↗ -3	↘ -4	↙ -12	↘ -20	↙ -28	↘ -36	↙ -44	↘ -52	↙ -60
W	-24	↑ -18	↗ -11	↘ -6	↙ -7	↘ -15	↙ -5	↘ -13	↙ -21	↘ -29	↙ -37
H	-32	↑ -14	↗ -18	↘ -13	↙ -8	↘ -9	↙ -13	↘ -7	↙ -3	↘ -11	↙ -19
E	-40	↑ -22	↗ -8	↘ -16	↙ -16	↘ -9	↙ -12	↘ -15	↙ -7	↘ 3	↙ -5
A	-48	↑ -30	↗ -16	↘ -3	↙ -11	↘ -11	↙ -12	↘ -12	↙ -15	↘ -5	↙ 2
E	-56	↑ -38	↗ -24	↘ -11	↙ -6	↘ -12	↙ -14	↘ -15	↙ -12	↘ -9	↙ 1

## Berechnen der Alignment-Matrix:

### Mathematische Formulierung für zu Hause

- Construct matrix  $F$  indexed by  $i$  and  $j$  (one index for each sequence)
- $F(i,j)$  is the score of the best alignment between the initial segment  $x_{1\dots i}$  of  $x$  up to  $x_i$  and the initial segment  $y_{1\dots j}$  of  $y$  up to  $y_j$
- Build  $F(i,j)$  recursively beginning with  $F(0,0) = 0$

## Berechnen der Alignment-Matrix:

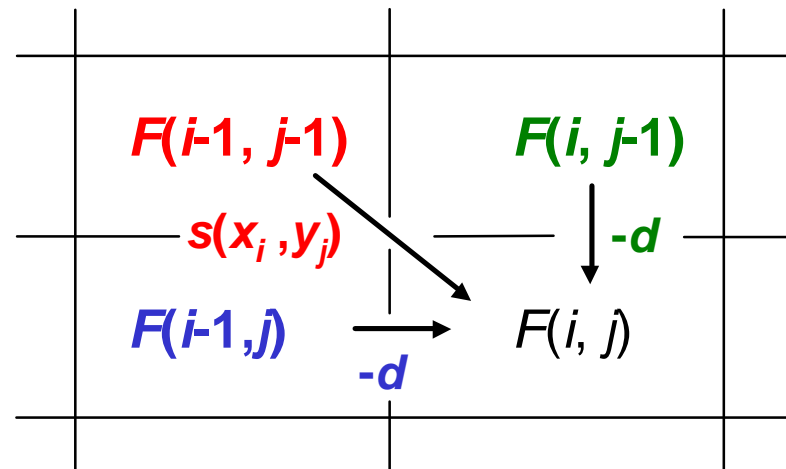
### Mathematische Formulierung für zu Hause

- If  $F(i-1,j-1)$ ,  $F(i-1,j)$  and  $F(i,j-1)$  are known we can calculate  $F(i,j)$
- Three possibilities:
  - $x_i$  and  $y_j$  are aligned,  $F(i,j) = F(i-1,j-1) + s(x_i, y_j)$
  - $x_i$  is aligned to a gap,  $F(i,j) = F(i-1,j) - d$
  - $y_j$  is aligned to a gap,  $F(i,j) = F(i,j-1) - d$
- The best score up to  $(i,j)$  will be the **largest** of the three options

## Berechnen der Alignment-Matrix:

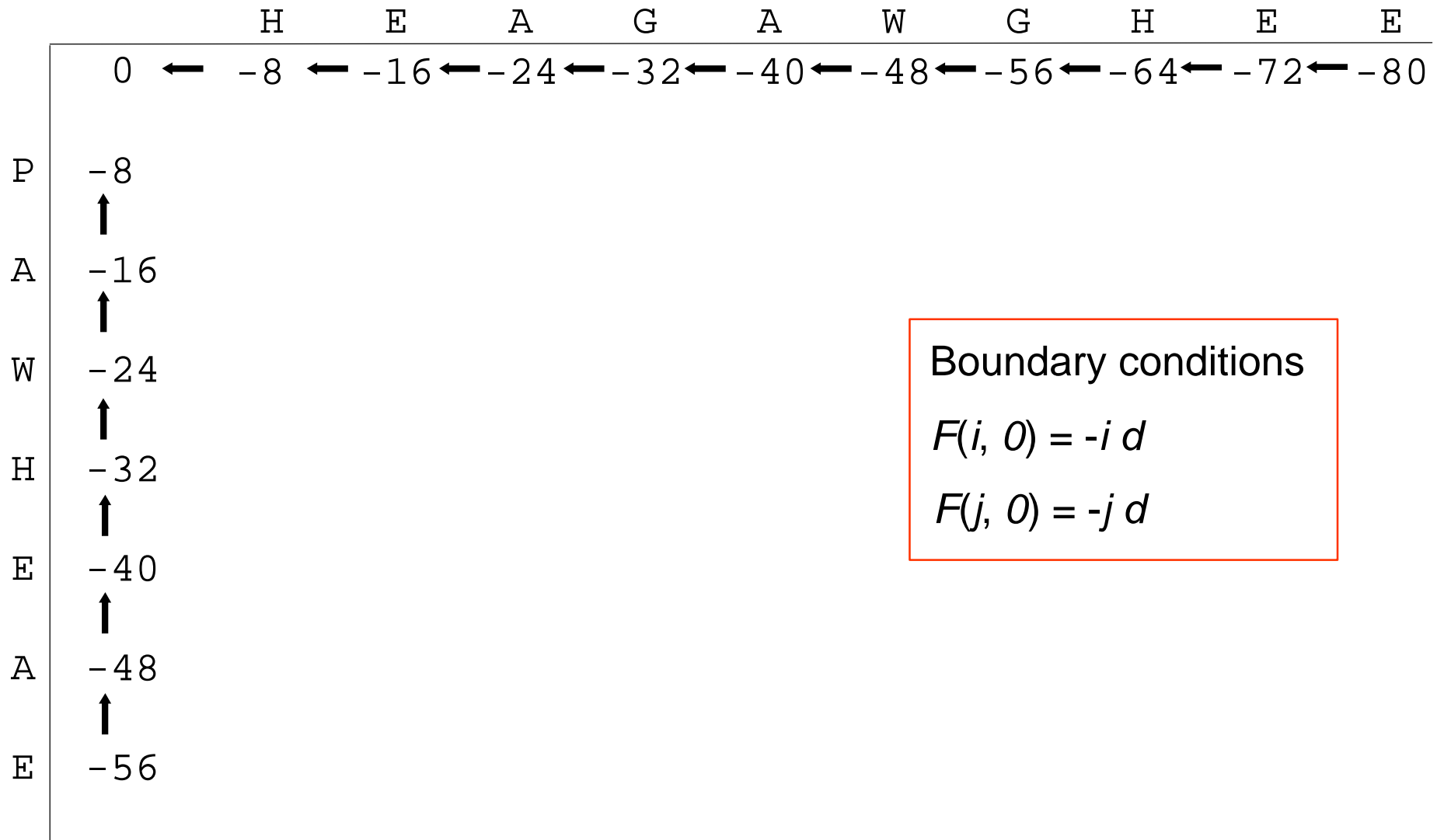
### Mathematische Formulierung für zu Hause

$$F(i, j) = \max \begin{cases} F(i, j) = F(i-1, j-1) + s(x_i, y_j) \\ F(i, j) = F(i-1, j) - d \\ F(i, j) = F(i, j-1) - d \end{cases}$$



# Berechnen der Alignment-Matrix:

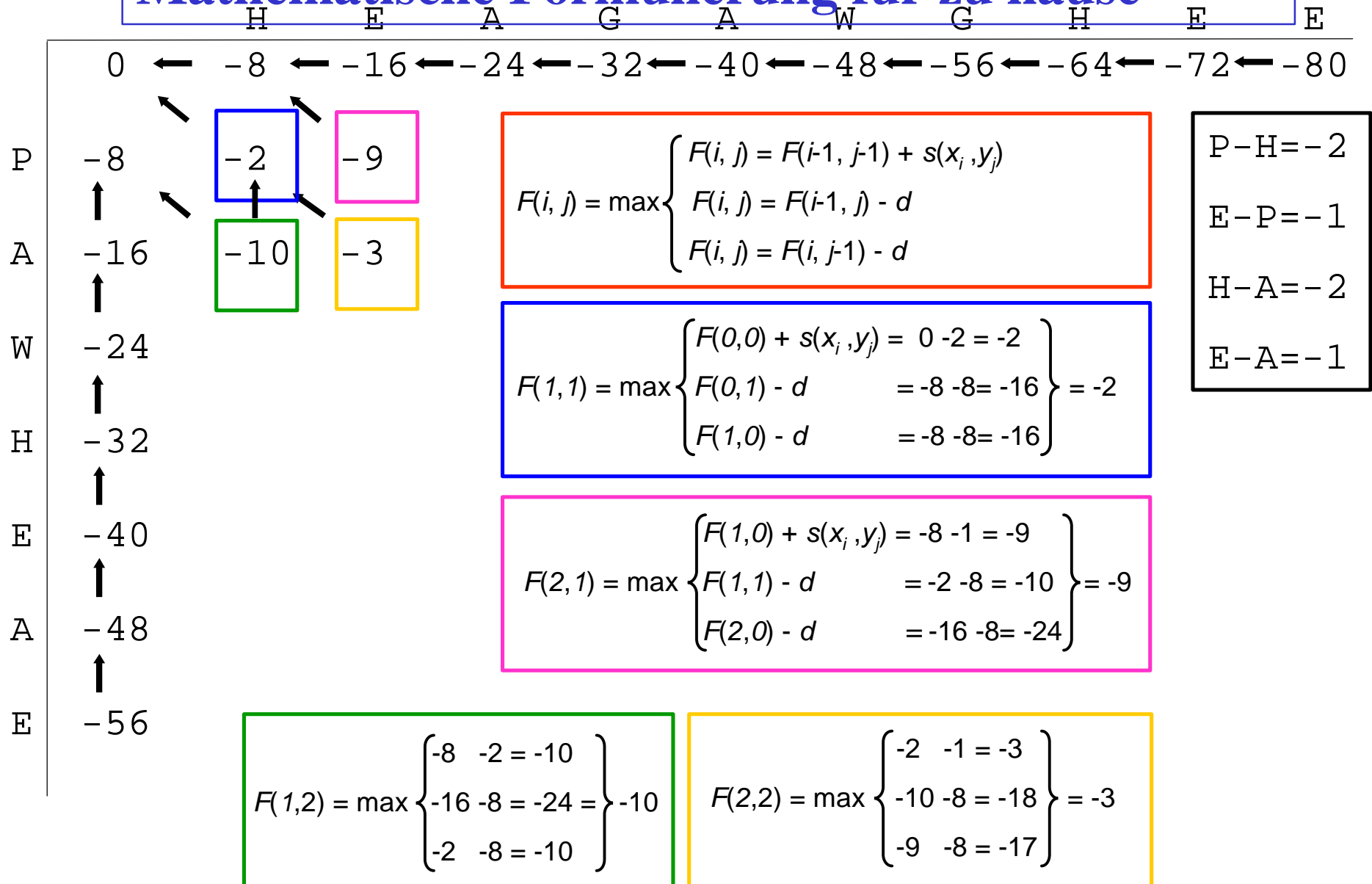
## Mathematische Formulierung für zu hause



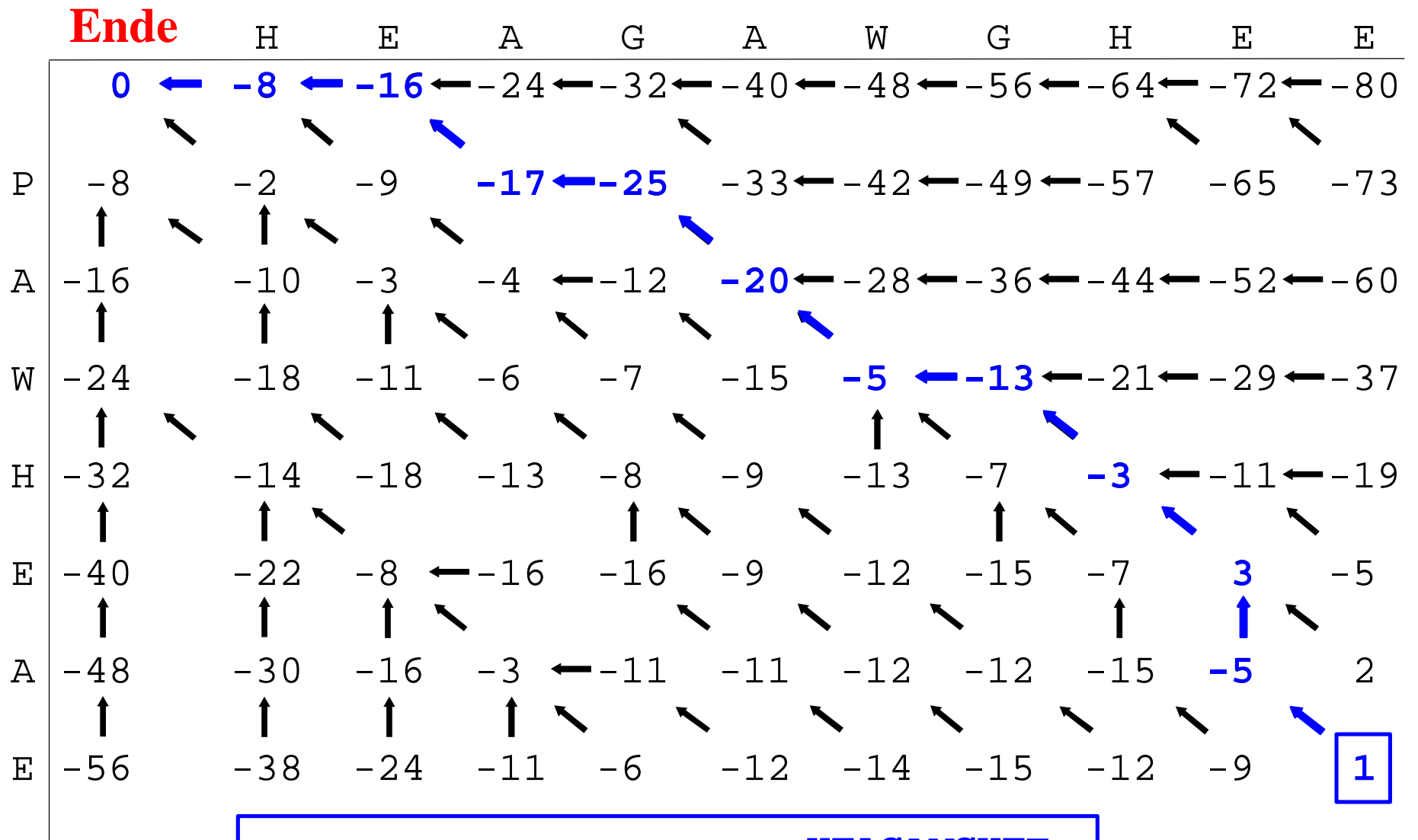
Boundary conditions  
 $F(i, 0) = -i d$   
 $F(j, 0) = -j d$

# Berechnen der Alignment-Matrix:

## Mathematische Formulierung für zu Hause



## 2. Das Backtracking



Optimales globales Alignment: HEAGAWGHEE  
--P-AW-HEE

Start

# Smith - Waterman (lokales Alignment)

Zwei simple Unterschiede:

1. Wenn der Wert, den man in die Matrix eintragen möchte negativ wird, dann schreibe eine Null
2. Ein Alginment kann überall in der Matrix starten und enden (=> suche die größte Zahl in der Backtracking-Matrix als Startpunkt)

## Berechnen der Alignment-Matrix:

### Mathematische Formulierung für zu Hause

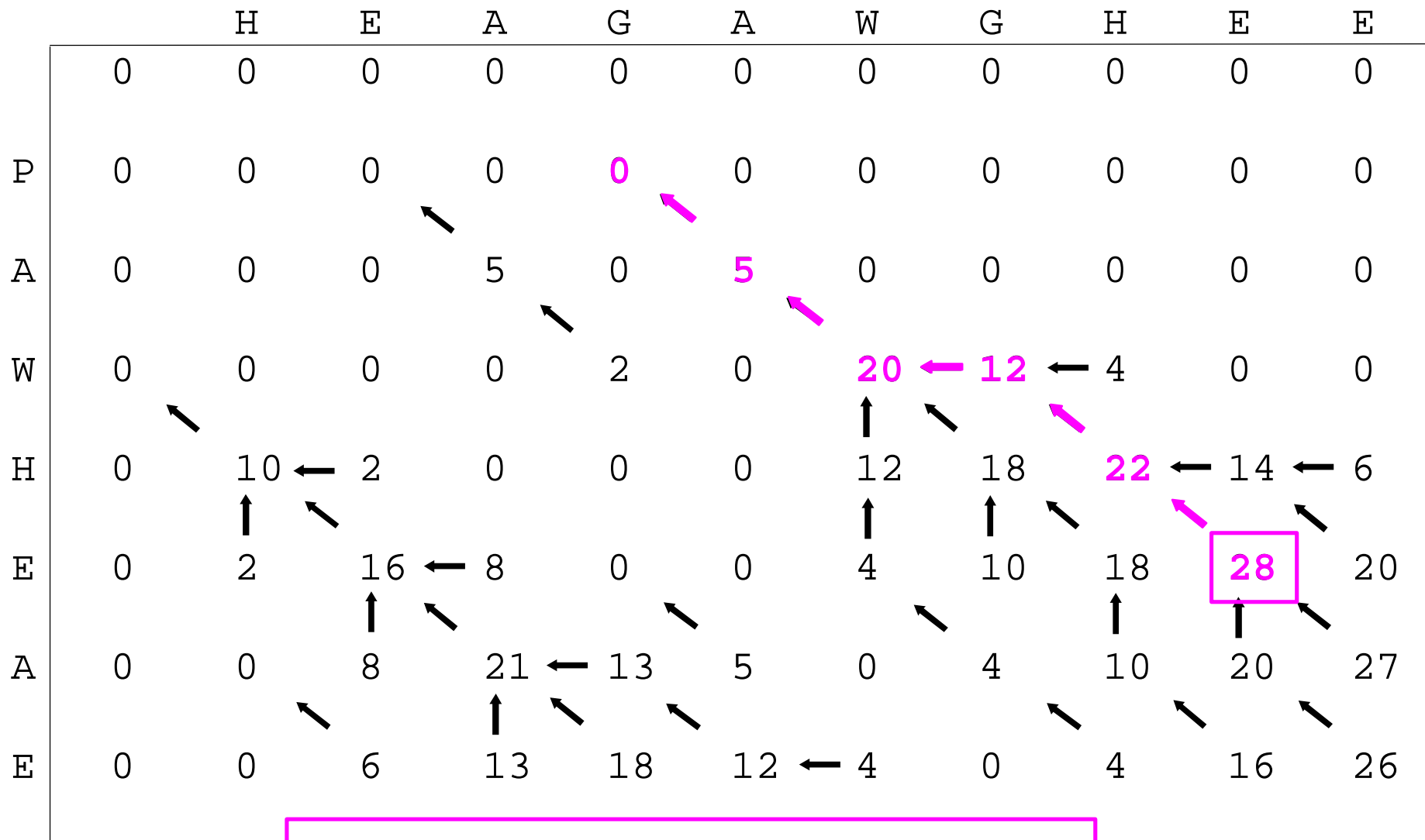
## Smith - Waterman (lokales Alignment)

mathematisch:

$$1. F(i, j) = \max \left\{ \begin{array}{l} 0 \\ F(i, j) = F(i-1, j-1) + s(x_i, y_j) \\ F(i, j) = F(i-1, j) - d \\ F(i, j) = F(i, j-1) - d \end{array} \right.$$

2. -

# Smith-Waterman Alignment

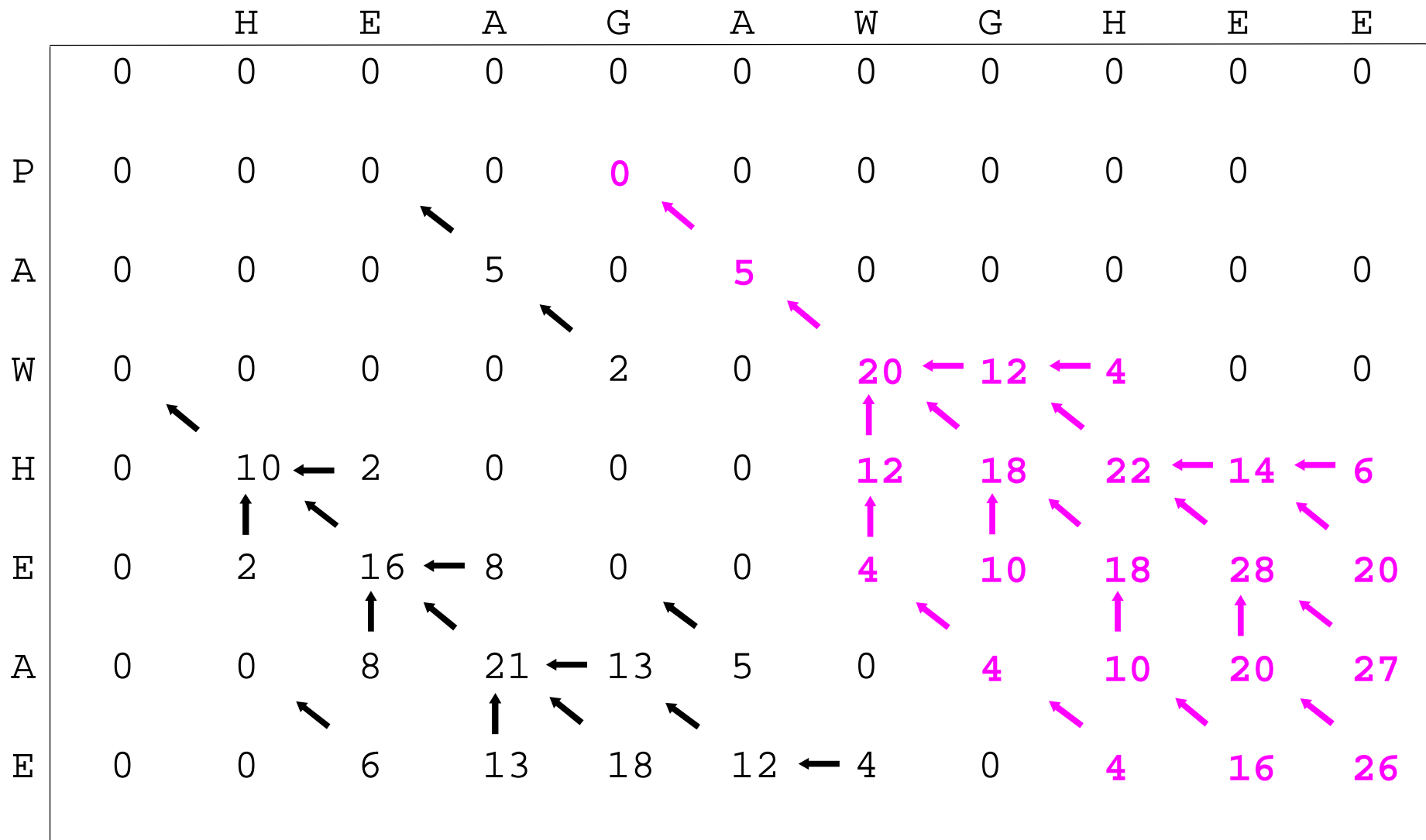


Optimales lokales Alignment: **AWGHE**  
**AW-HE**

## Smith - Waterman für mehrere lokale Alignments

- ermittle das beste lokale Alignment
- lösche alle Einträge dafür und drumherum
- ermittle das jetzt beste Alignment
- u.s.w.

# Smith-Waterman Alignment



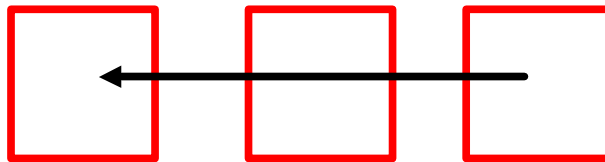
# Smith-Waterman Alignment

		H	E	A	G	A	W	G	H	E	E
	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	5	0	0	0	0	0	0	0
W	<b>0</b>	0	0	0	2	0	0	0	0	0	0
H	0	<b>10</b>	2	0	0	0	0	0	0	0	0
E	0	2	<b>16</b>	8	0	0	0	0	0	0	0
A	0	0	8	<b>21</b>	13	5	0	0	0	0	0
E	0	0	6	13	18	12	4	0	0	0	0

Zweitbestes Alignment: **HEA**  
**HEA**

# Alignment mit affinen Gapkosten

- wenn ausgedehnte Gaps billiger sind, als ihre Summe (affine Gapkosten), dann werden auch längere Pfeile, entsprechend einem längeren Gap, in die Matrix eingetragen  
=> wesentlich höherer Rechenaufwand



# Algorithmische Komplexität

Wieviel Zeit und Speicher braucht so ein Algorithmus bezogen auf seine Eingabedaten?

## Beispiel Needleman - Wunsch

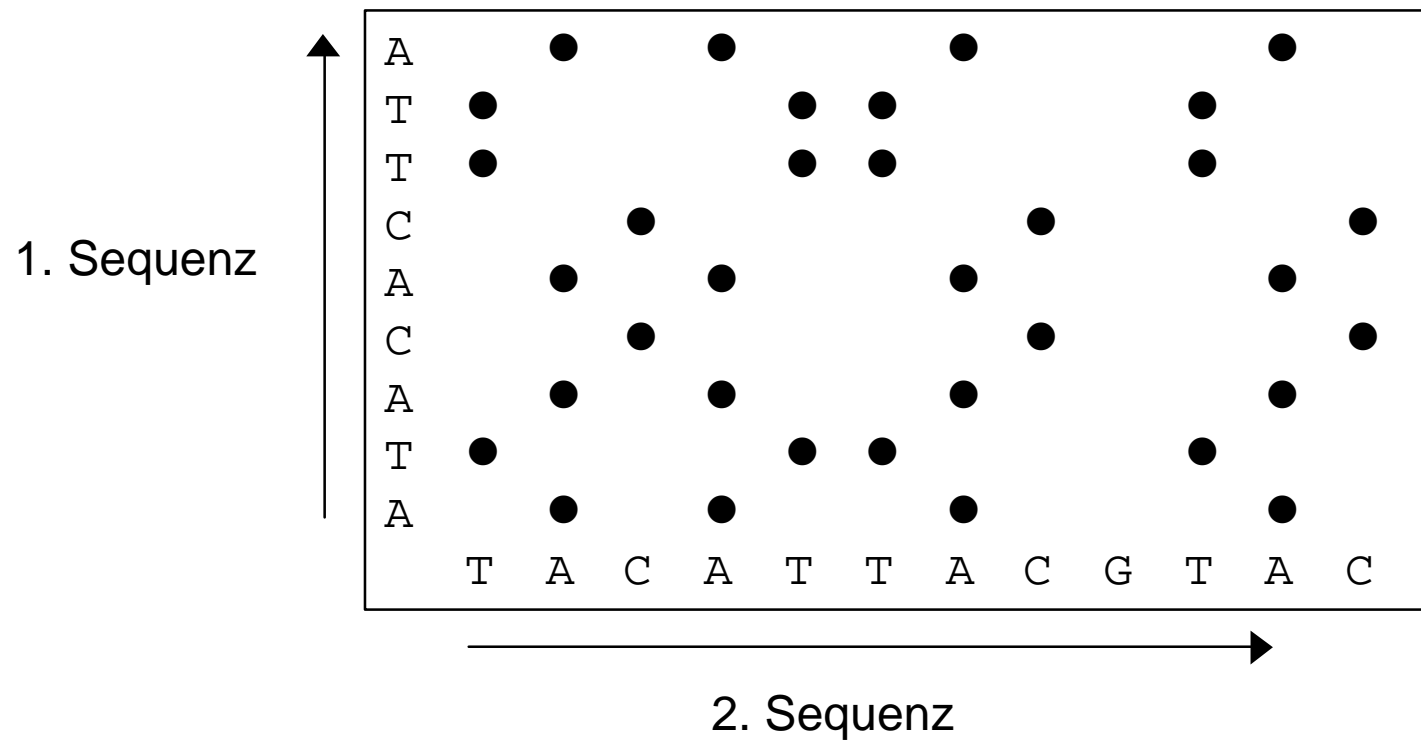
- speichert  $(n+1) \times (m+1)$  Zahlen
- Jede Zahl kostet eine konstante Anzahl von Berechnungen (drei Summen und ein Maximum)
- $\Rightarrow$  Algorithmus braucht  $O(nm)$  Speicher und  $O(nm)$  Zeit
- $n$  und  $m$  sind meist ähnlich groß  $\Rightarrow$  Rechenzeit und Speicher  $\sim n^2$

# Methoden des Sequenz-Alignments

- Dynamisches Programmieren  
gibt optimales Alignment,  
alle möglichen Kombinationen,  
cpu-intensiv
- Dotplot-Analyse  
erster Eindruck,  
zeigt Einfügungen/Auslassungen, Repeats
- "Wort"-Methoden  
Sammeln von "Inseln",  
schnell, aber heuristisch,  
wird für DB-Suchen verwendet (spezielle Vorlesung)

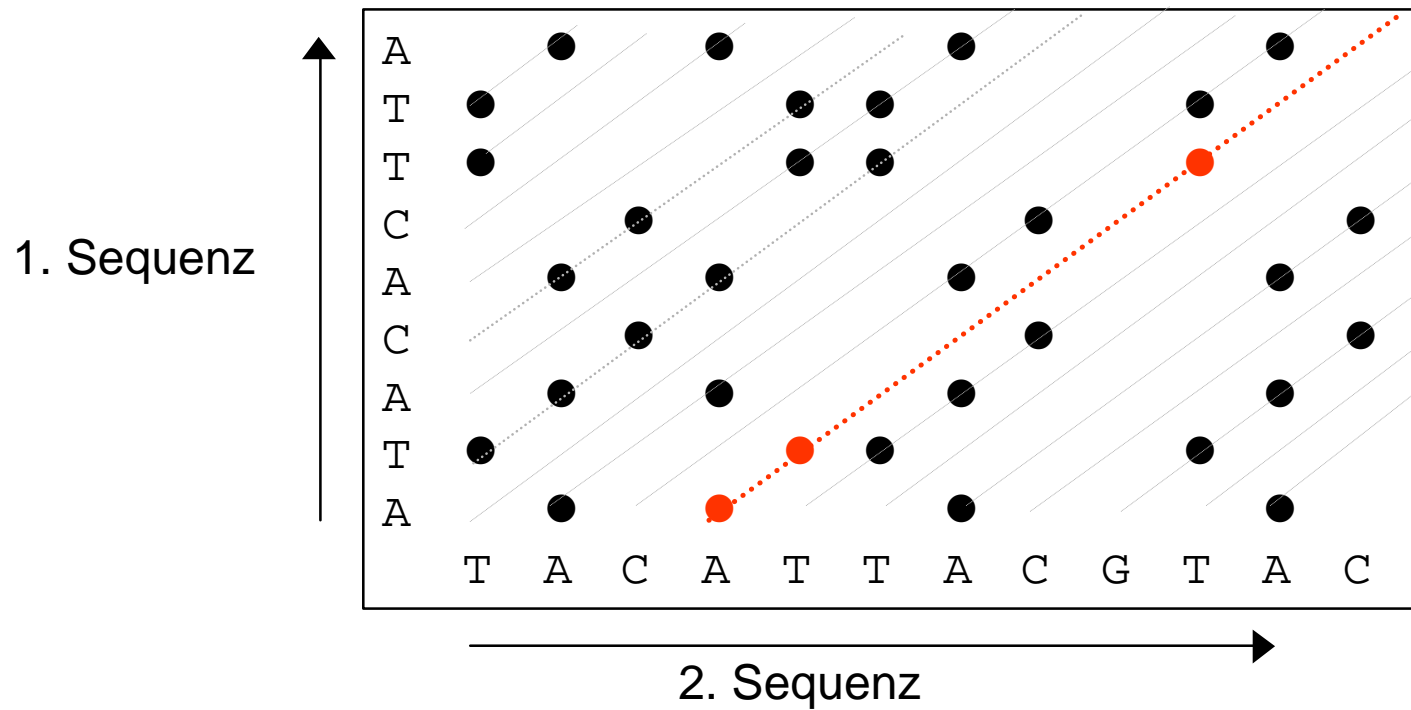
# Dotplot

- Überblick über alle möglichen Alignments



# Dotplot

- jede Diagonale entspricht einem möglichen Alignment (ohne Gaps)



Ein mögliches Alignment:

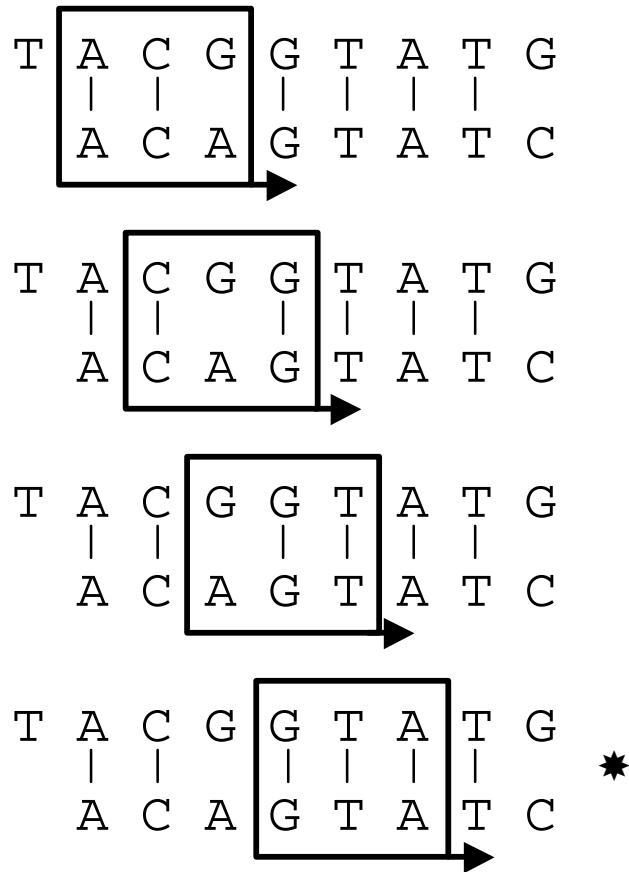
T	A	C	A	T	T	A	C	G	T	A	C
			A	T	A	C	A	C	T	T	A

# Erweiterung des Dotplots durch Einführung von "Fenster"

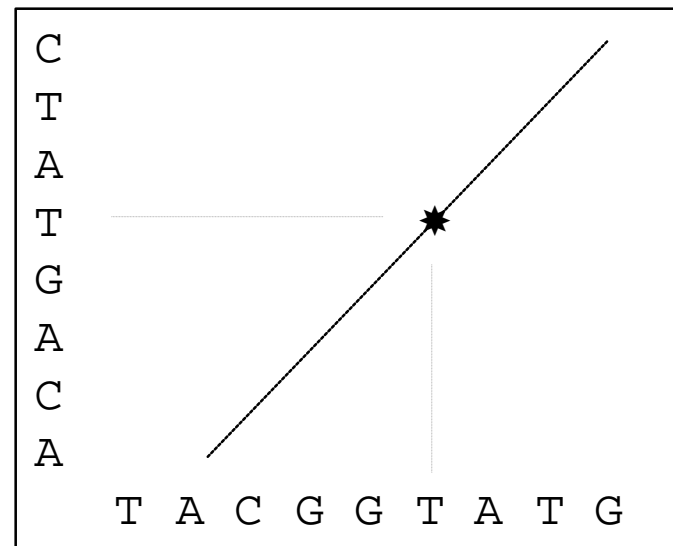
Parameter:

- Wort-Größe
- Fenstergröße, Stringenz

# Word Size Algorithm

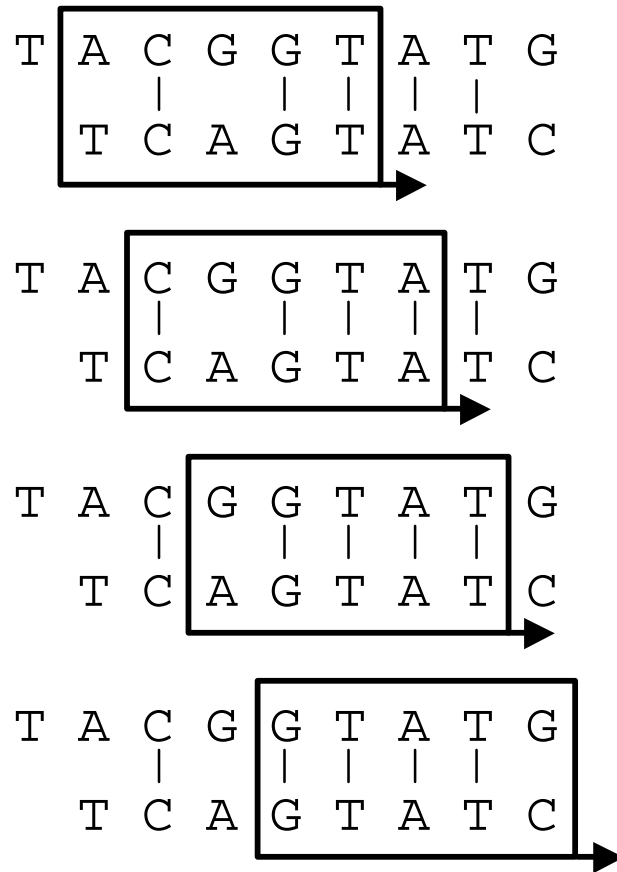


Wortgröße = 3 = Fenstergröße

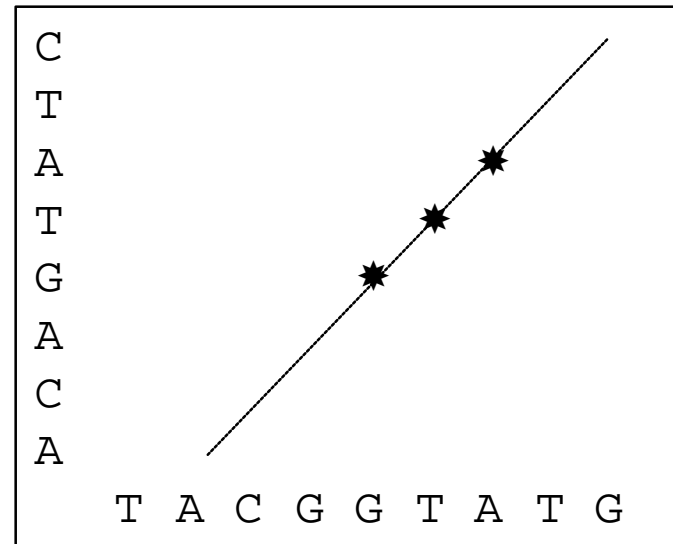


etwas lockerer:

## Fenster / Stringenz

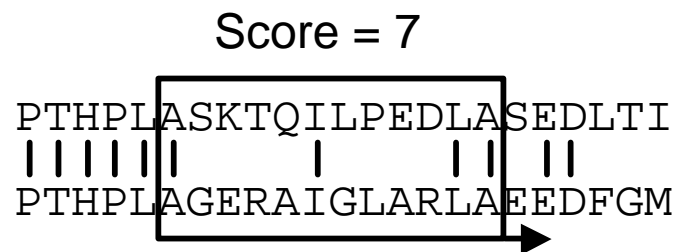
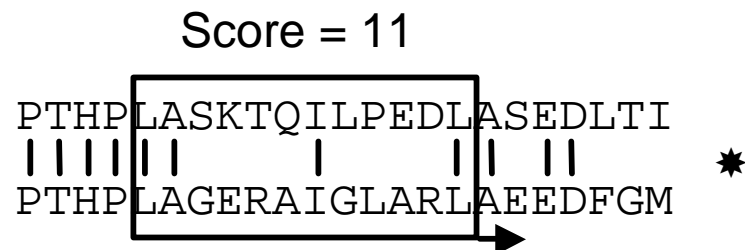
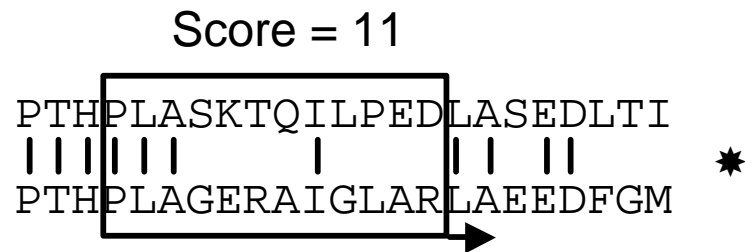


Fenster = 5 / Stringenz = 4



# Fenster / Stringenz

... geht auch für Aminosäure-Sequenzen ...



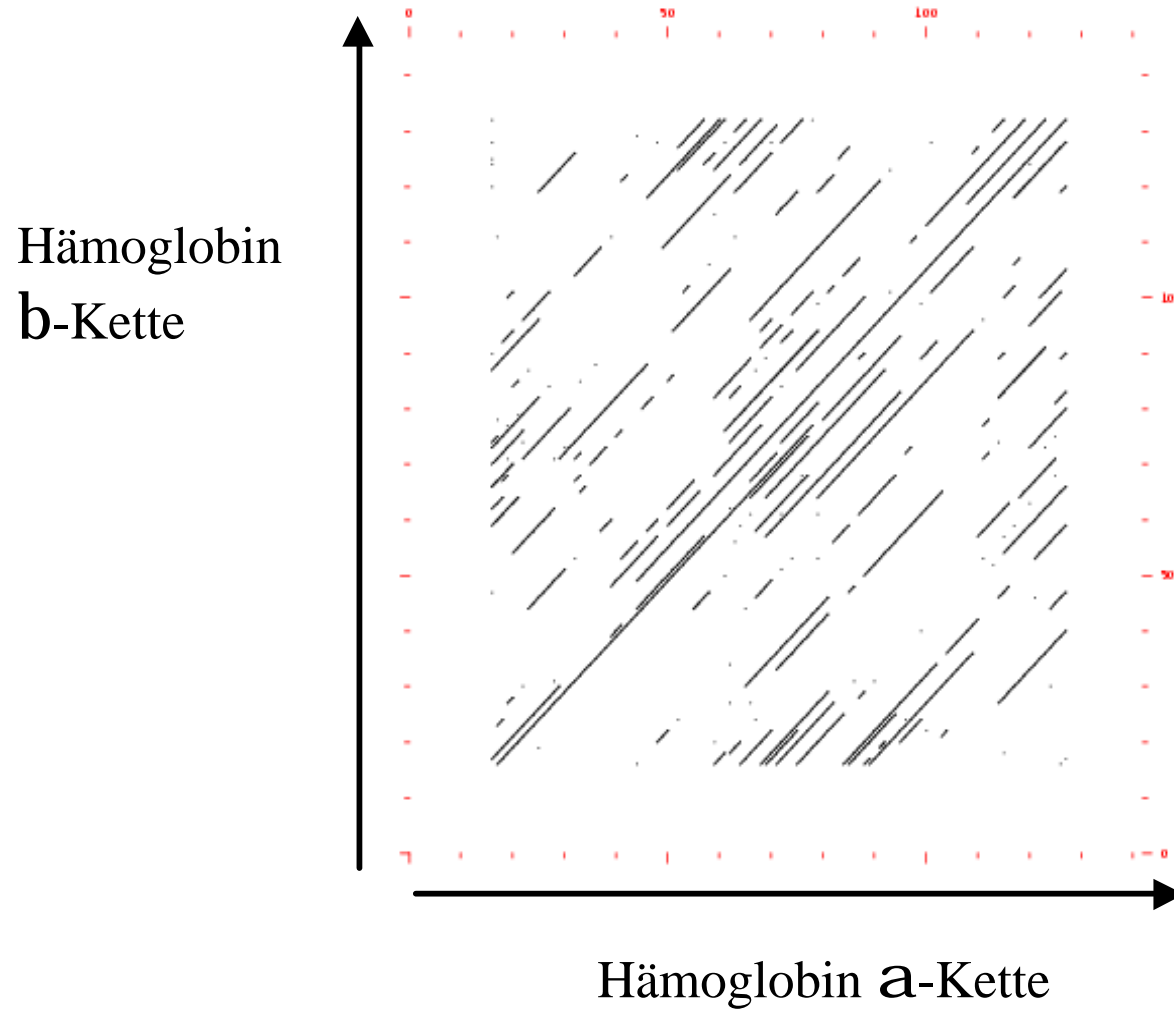
Matrix: PAM250

Fenster = 12

Stringenz = 9

Beispiel Hämoglobin, mit Fenster = 30 / Stringenz = 9

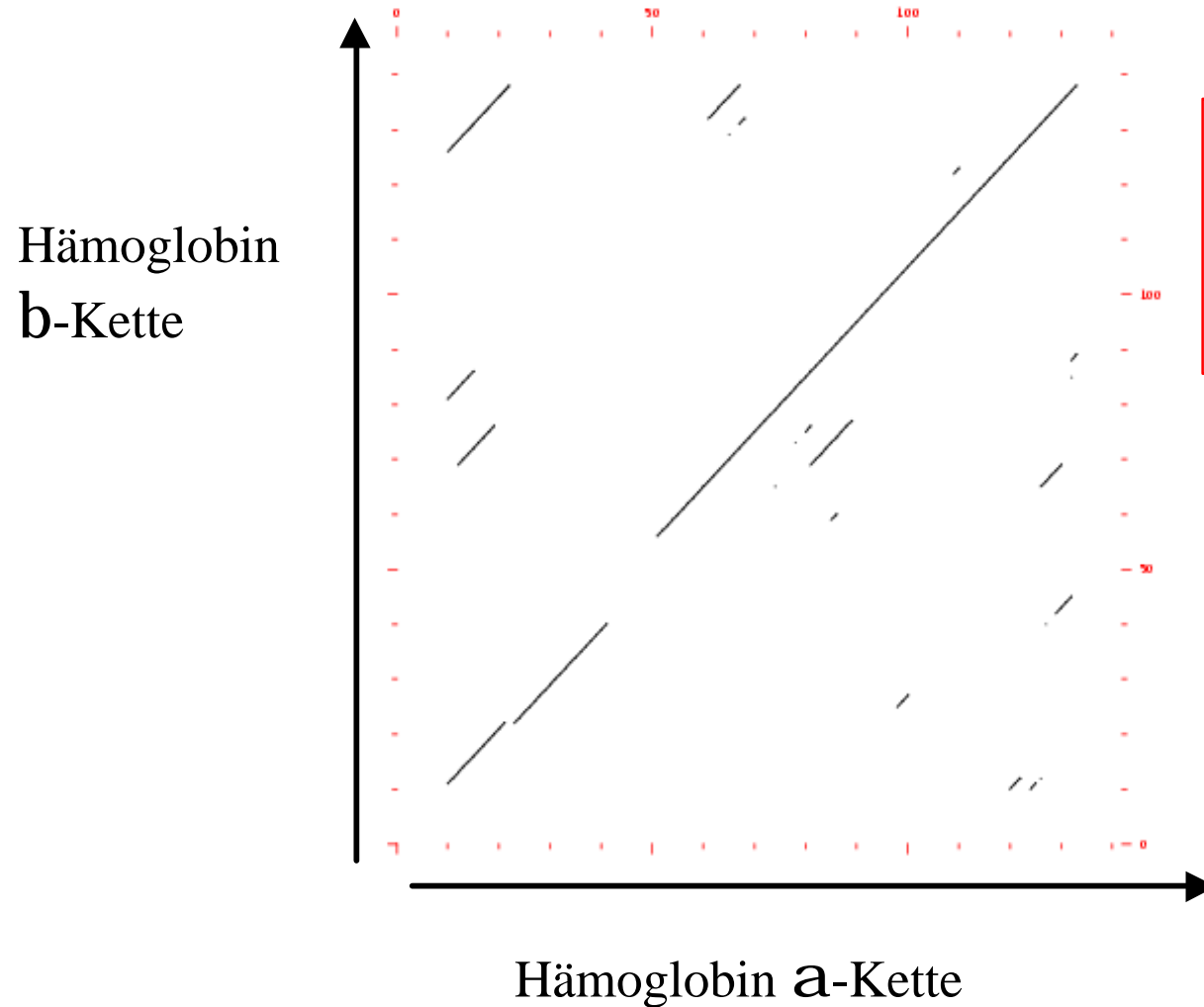
... Stringenz zu niedrig ...



Grafiken aus den  
Programmen  
*Compare* und  
*DotPlot2*

Beispiel Hämoglobin, mit Fenster = 18 / Stringenz = 10

... Stringenz o.k. ...

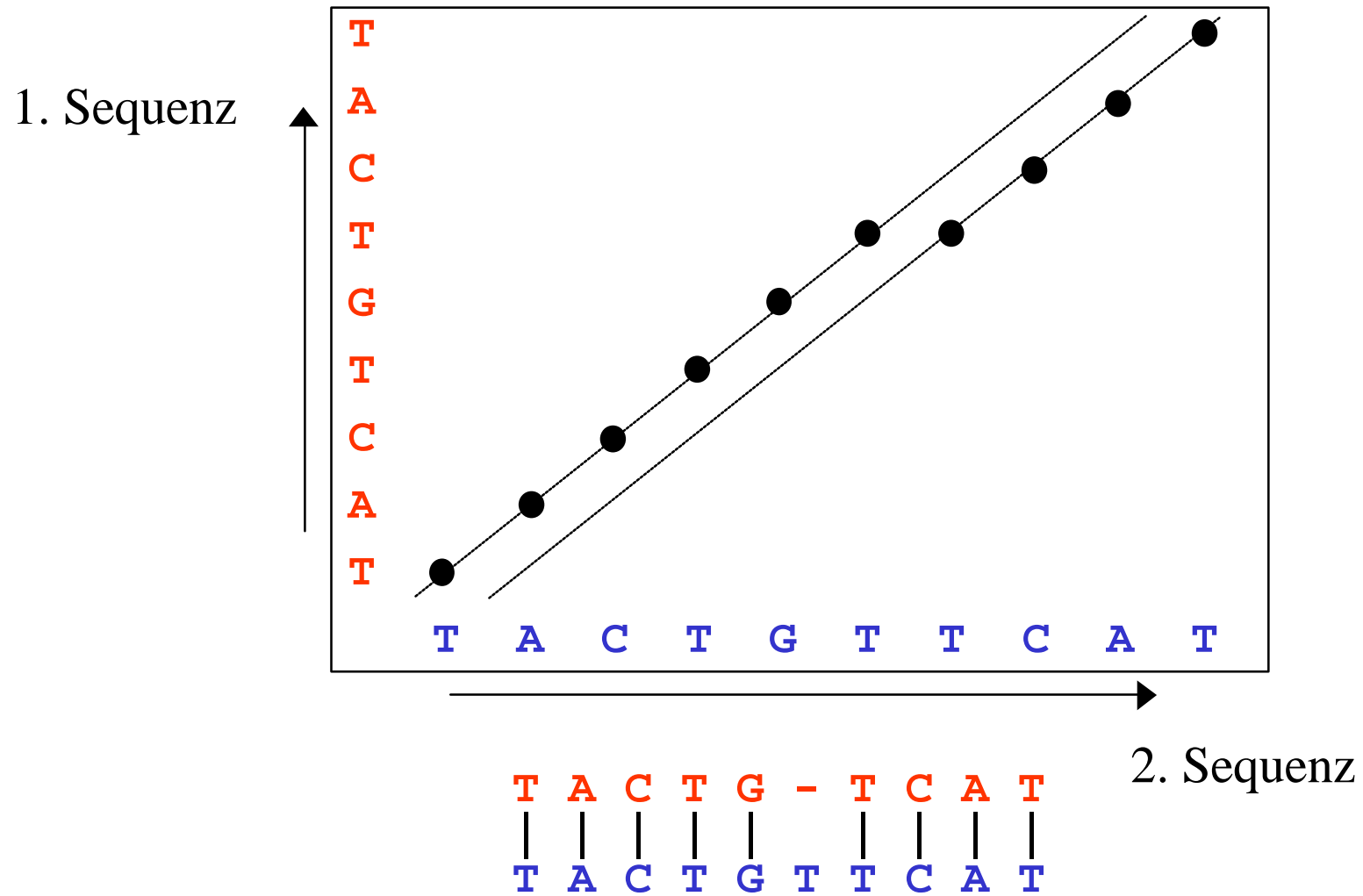


Grafiken aus den  
Programmen  
*Compare* und  
*DotPlot2*

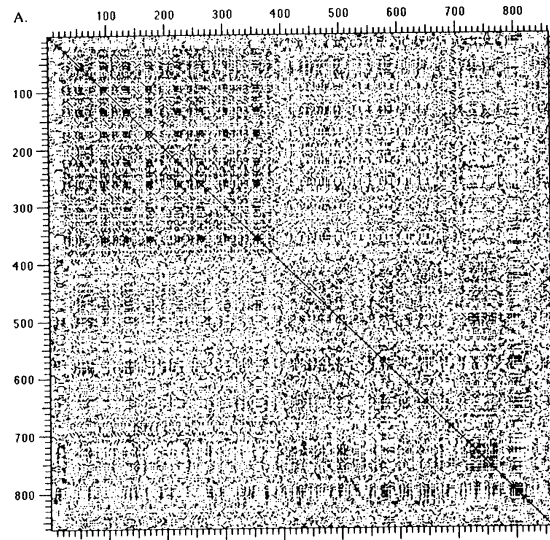
# Bemerkungen

- Fenster/Stringenz-Methode (lässt Mismatches bzw. unspezifische Matches zu), ist sensitiver als reine Wort-Methode
- je kleiner das Fenster, desto stärkeres statistisches Gewicht haben Mismatches => statistische Schwankungen werden auch angezeigt
- aber: große Fenster verringern die Sensitivität für kleine Sequenzen  
=> optimale Fenster-Stringenz- Einstellung muss (durch probieren) ermittelt werden
- Einfügungen, Auslassungen werden nicht direkt angezeigt, aber indirekt:

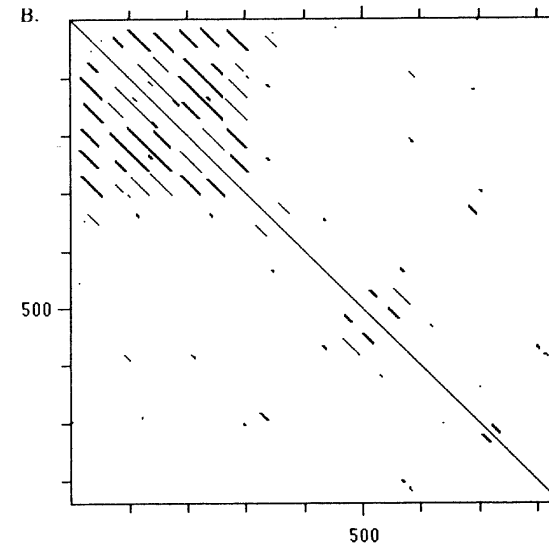
# Erkennen von Einfügungen/Auslassungen in einem Dotplot



# Repeats



Fenster = 1, Stringenz = 1,



Fenster = 23, Stringenz = 7,

- rechts: Diagonale: Sequenz auf sich selbst abgebildet
- Plot ist symmetrisch
- Repeats sind kleinere Diagonalen
  - hier: z.B. 6 Repeats in der oberen linken Ecke
  - Repeats sind nicht alle gleich lang => enthalten Mutationen
  - wieder: nur beobachtbar bei korrekter Wahl der Fenstergröße und Stringenz  
=> ausprobieren der Einstellungen notwendig

# Methoden des Sequenz-Alignments

- **Dynamische Programmieren**
- **Dotplot-Analyse**
- **Wort-Methoden für DB-Suchen**

# Datenbanksuchen: BLAST

- Fenster-basiert, Wortgröße = 11 (Nukleotide), 3 (Aminosäuren)
- Wenn ein Wort über diesem Schwellenwert gefunden wurde, wird es nach Möglichkeit verlängert
- Verlängerung bricht ab, wenn der Score-Wert zu stark abfällt
- Neuere BLAST-Versionen können auch Gaps verarbeiten

**Heuristische Methode, ohne dynamisches Programmieren, schnell aber nicht so sensitiv**



# Datenbanksuchen: FastA

- Fenster-basierend, Wortgröße 4-6 (Nukleotide), 1-2 (Aminosäuren)
- berechnet Scorewerte für Diagonale, Diagonale mit den besten Scores werden behalten
- Diagonalen werden verlängert
- Diagonalen, die schlecht zu einem Alignment mit der besten Diagonalen passen, werden verworfen
- vollständiges Dynamisches Programmieren innerhalb einer Zone/Band, in dem die verbliebenen Diagonalen sind (rechenintensiv)

**Heuristische Methode, enthält dynamisches Programmieren. Ist langsamer als BLAST, aber dafür sensitiver.**

## FASTA Algorithm

