Florian Erhard
LFE Bioinformatik
Institut für Informatik
Ludwig-Maximilians-Universität München

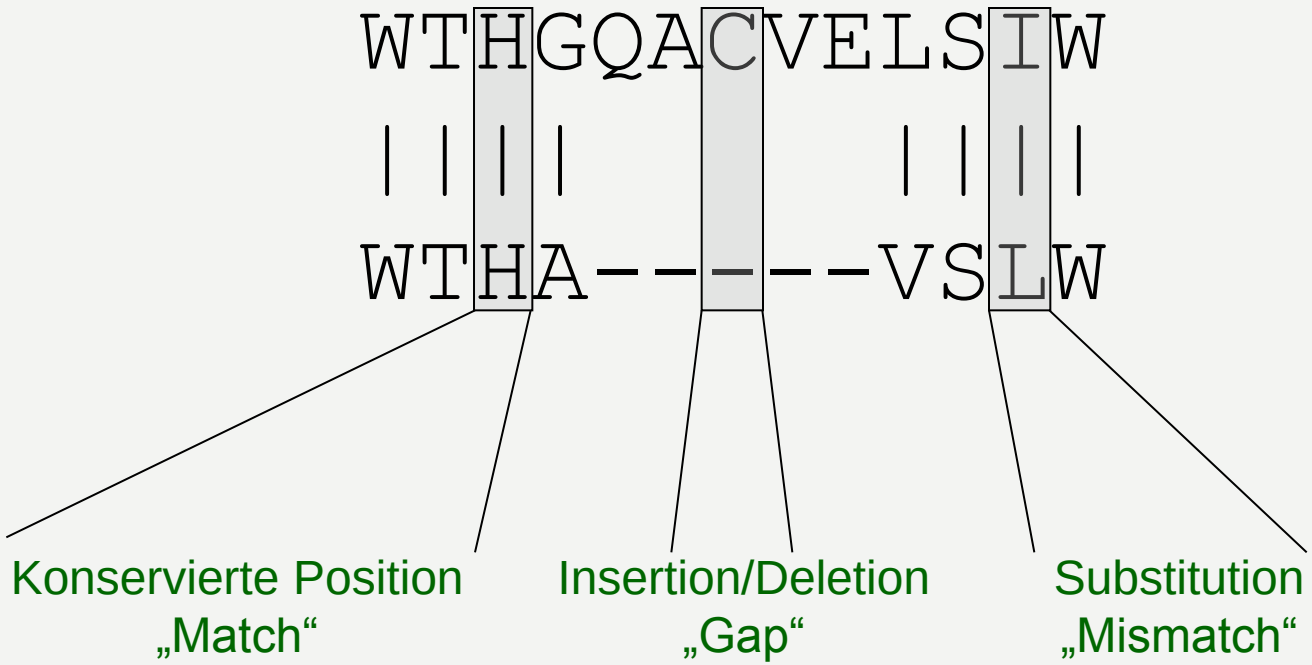
Sequenzen-Alignment
für Fortgeschrittene

GoBi WS 2013/14

Einführung Alignment

Für zwei Wörter s,t ∈ Σ* soll eine Ähnlichkeit berechnet werden.
Dazu wird die bestmögliche Zuordnung ihrer Buchstaben gesucht. Solche Zuordnungen nennt man Alignment.

z.B. s=WTHGQACVELSIW, t=WTHAVSLW, $s,t \in \Sigma_{AS}^*$



Alignment-Bewertung

Gültige Alignments: Reihenfolge muss beibehalten werden. Gaps dürfen beliebig eingefügt werden, so dass beide Sequenzen gleich lang sind. Es darf nur nie – über – stehen.

```
WTHGQACVELSIW-
| | | | | | |
W-T-H-G-V-S-LW
```

```
WTHGQACVELSIW
||||      ||||
WTHA-----VSLW
```

„biologisch sinnvolles Alignment“

```
WTHGQACVELSIW
|||   | |||   |
WTH--A-VSL--W
```

Alignment mit den meisten Matches

Gap-Kosten

```
WTHGQACVELSIW
||||      ||||
WTHA-----VSLW
```

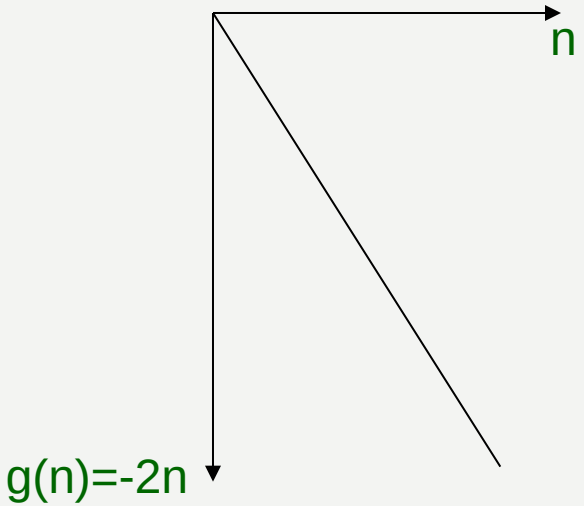
Bei BLOSUM50 und g=-2:

s=40.9

```
WTHGQACVELSIW
|||   |  |||   |
WTH--A-VSL--W
```

s=45.9

Gap-Kosten-Funktion:
g: Gaplänge → Kosten

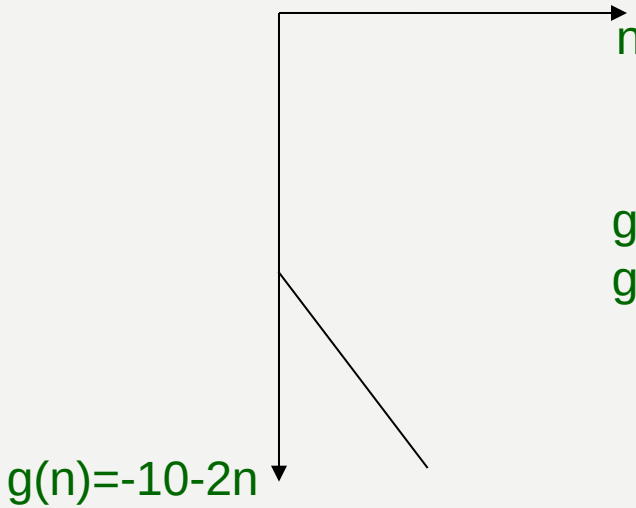


$g_1 = g(5) = -10$
$g_2 = g(2)+g(1)+g(2) = -10$

Affine Gap-Kosten

g(n)=go + ge*n

für n>0



$g_1 = g(5) = -20$
$g_2 = g(2)+g(1)+g(2) =$
$= -14 \quad -12 \quad -14 = -40$

```
WTHGQACVELSIW
||||      ||||
WTHA-----VSLW
```

Bei BLOSUM50 und g(n)=-10-2n:

s=30.9 (optimal!)

```
WTHGQACVELSIW
|||   |  |||   |
WTH--A-VSL--W
```

s=15.9

Problemstellung

Gegeben:

- **Zwei Sequenzen** **s, t** $\in \Sigma^*$
- **Substitutionsmatrix** **S** $: \Sigma \times \Sigma \rightarrow R$
- **Gap-Kosten-Funktion** **g** $: n \rightarrow R$

Gesucht:

Alignment mit maximalen score

Algorithmus von Gotoh

Merke nicht nur eine Tabelle A, sondern drei Tabellen A,I,D:

$$I_{i,j} = \max \{ A_{i-1,j} + go + ge, I_{i-1,j} + ge \}$$
$$D_{i,j} = \max \{ A_{i,j-1} + go + ge, D_{i,j-1} + ge \}$$
$$A_{i,j} = \max \{ A_{i-1,j-1} + S(s_i,t_j), D_{i,j} , I_{i,j} \}$$

A_{ij}: score des optimalen Alignments der Präfixe $s^{(i)}$ und $t^{(j)}$

D_{ij}: score des optimalen Alignments der Präfixe $s^{(i)}$ und $t^{(j)}$,das mit einem Gap in s endet (Deletion)

I_{ij}: score des optimalen Alignments der Präfixe $s^{(i)}$ und $t^{(j)}$,das mit einem Gap in t endet (Insertion)

Zeitbedarf: O(n*m)
Platzbedarf: O(n*m)

Gotoh in a nutshell

Initialisierung

- $A_{0,k} = A_{k,0} = g(k)$
- $I_{0,j} = D_{i,0} = -Inf$

Gotoh-Initialisierung

A		W	T	H	G	Q	A
	0.00	-12.00	-14.00	-16.00	-18.00	-20.00	-22.00
W	-12.00						
T	-14.00						
H	-16.00						
A	-18.00						

… oder 0 für Freeshift / Local

Gotoh-Initialisierung

Spalte 0 wird nie benutzt!

I		W	T	H	G	Q	A
		-Inf	-Inf	-Inf	-Inf	-Inf	-Inf
W							
T							
H							
A							

$I_{0,j}$ ist nicht definiert!

„-Inf bewirkt, dass für Zeile 1 immer der Wert aus A benutzt wird“

$I_{i,j}$: score des optimalen Alignments der Präfixe $s^{(i)}$ und $t^{(j)}$, das mit einem Gap in t endet (Insertion)

Gotoh-Initialisierung

Zeile 0 wird nie benutzt!

D		W	T	H	G	Q	A
W	-Inf						
T	-Inf						
H	-Inf						
A	-Inf						

$D_{i,0}$ ist nicht definiert!

„-Inf bewirkt, dass für Spalte 1 immer der Wert aus A benutzt wird“

$D_{i,j}$: score des optimalen Alignments der Präfixe $s^{(i)}$ und $t^{(j)}$, das mit einem Gap in s endet (Deletion)

Gotoh in a nutshell

Initialisierung

- $A_{0,k} = A_{k,0} = g(k)$
- $I_{0,j} = D_{i,0} = -Inf$

Rekursion

$$I_{i,j} = \max \{ A_{i-1,j} + go + ge, I_{i-1,j} + ge \}$$
$$D_{i,j} = \max \{ A_{i,j-1} + go + ge, D_{i,j-1} + ge \}$$
$$A_{i,j} = \max \{ A_{i-1,j-1} + S(s_i,t_j), D_{i,j}, I_{i,j} \}$$

Gotoh-Rekursion

I		W	T	H	G	…
		-Inf	-Inf	-Inf	-Inf	…
W		-24.00				
T						
H						
A						

A		W	T	H	G	…
	0.00	-12.00	-14.00	-16.00	-18.00	…
W	-12.00					
T	-14.00					
H	-16.00					
A	-18.00					

$I_{1,1}$ = max { -12.00-10-2,

$$I_{i,j} = \max \{ A_{i-1,j} + go + ge, I_{i-1,j} + ge \}$$

Gotoh-Rekursion

I		W	T	H	G	…
		-Inf	-Inf	-Inf	-Inf	…
W		-Inf				
T						
H						
A						

A		W	T	H	G	…
	0.00	-12.00	-14.00	-16.00	-18.00	…
W	-12.00					
T	-14.00					
H	-16.00					
A	-18.00					

$I_{1,1}$ = max { -12.00-10-2, -Inf-2 }

$$\mathbf{I_{i,j} = \max \{ A_{i-1,j} + go + ge, I_{i-1,j} + ge \}}$$

Gotoh-Rekursion

I		W	T	H	G	…
		-Inf	-Inf	-Inf	-Inf	…
W	-24.00					
T						
H						
A						

A		W	T	H	G	…
	0.00	-12.00	-14.00	-16.00	-18.00	…
W	-12.00					
T	-14.00					
H	-16.00					
A	-18.00					

$I_{1,1}$ = max { -12.00-10-2, -Inf-2 }

$$\mathbf{I_{i,j} = \max \{ A_{i-1,j} + go + ge, I_{i-1,j} + ge \}}$$

Gotoh-Rekursion

D		W	T	H	G	…
W	-Inf	-24.00				
T	-Inf					
H	-Inf					
A	-Inf					

A		W	T	H	G	…
	0.00	-12.00	-14.00	-16.00	-18.00	…
W	-12.00					
T	-14.00					
H	-16.00					
A	-18.00					

$D_{1,1}$ = max { -12.00-10-2,

$$D_{i,j} = \max \{ A_{i,j-1} + go + ge, D_{i,j-1} + ge \}$$

Gotoh-Rekursion

D		W	T	H	G	…
W	-Inf	-Inf				
T	-Inf					
H	-Inf					
A	-Inf					

A		W	T	H	G	…
	0.00	-12.00	-14.00	-16.00	-18.00	…
W	-12.00					
T	-14.00					
H	-16.00					
A	-18.00					

$D_{1,1}$ = max { -12.00-10-2,-Inf-2 }

$$\mathbf{D_{i,j} = \max \{ A_{i,j-1} + go + ge, D_{i,j-1} + ge \}}$$

Gotoh-Rekursion

D		W	T	H	G	…
W	-Inf	-24.00				
T	-Inf					
H	-Inf					
A	-Inf					

A		W	T	H	G	…
	0.00	-12.00	-14.00	-16.00	-18.00	…
W	-12.00					
T	-14.00					
H	-16.00					
A	-18.00					

$D_{1,1}$ = max { -12.00-10-2,-Inf-2 }

$D_{i,j}$ = max { $A_{i,j-1}$ + go + ge, $D_{i,j-1}$ + ge }

Gotoh-Rekursion

	A	W	T	H	G	Q	A
	0.00	-12.00	-14.00	-16.00	-18.00	-20.00	-22.00
W	-12.00	17.30					
T	-14.00						
H	-16.00						
A	-18.00						

$A_{1,1} = \max \{ 0 + 17.30$

$$A_{i,j} = \max \{ A_{i-1,j-1} + S(s_i,t_j), D_{i,j}, I_{i,j} \}$$

Gotoh-Rekursion

A		W	T	H	G	…
	0.00	-12.00	-14.00	-16.00	-18.00	
W	-12.00	-24.00				
T	-14.00					
H	-16.00					
A	-18.00					

D		W	T	H	G	…
W	-Inf	-24				
T	-Inf					
H	-Inf					
A	-Inf					

$A_{1,1} = \max \{ 0 + 17.30, -24$

$$\mathbf{A_{i,j} = \max \{ A_{i-1,j-1} + S(s_i,t_j), D_{i,j}, I_{i,j} \}}$$

Gotoh-Rekursion

A		W	T	H	G	…
	0.00	-12.00	-14.00	-16.00	-18.00	
W	-12.00	-24.00				
T	-14.00					
H	-16.00					
A	-18.00					

I		W	T	H	G	…
		-Inf	-Inf	-Inf	-Inf	…
W		-24				
T						
H						
A						

$A_{1,1} = \max \{ 0 + 17.30, -24, -24 \}$

$$\mathbf{A_{i,j} = \max \{ A_{i-1,j-1} + S(s_i,t_j), D_{i,j}, I_{i,j} \}}$$

Gotoh-Rekursion

A		W	T	H	G	Q	A
	0.00	-12.00	-14.00	-16.00	-18.00	-20.00	-22.00
W	-12.00	17.30					
T	-14.00						
H	-16.00						
A	-18.00						

$$A_{1,1} = \max \{ 0 + 17.30, -24, -24 \}$$

… oder 0 für local

Gotoh-Rekursion

I		W	T	H	G	Q	A
		-Inf	-Inf	-Inf	-Inf	-Inf	-Inf
W		-24.00	-26.00	-28.00	-30.00	-32.00	-34.00
T		5.30	-6.70	-8.70	-10.70	-12.70	-14.70
H		3.30	7.90	-4.10	-6.10	-8.10	-10.10
A		1.30	5.90	14.50	2.50	0.50	-1.50

$$I_{i,j} = \max \{ A_{i-1,j} + go + ge, I_{i-1,j} + ge \}$$

Gotoh-Rekursion

D		W	T	H	G	Q	A
W	-Inf	-24.00	5.30	3.30	1.30	-0.70	-2.70
T	-Inf	-26.00	-6.70	7.90	5.90	3.90	1.90
H	-Inf	-28.00	-8.70	-4.10	14.50	12.50	10.50
A	-Inf	-30.00	-10.70	-6.10	2.50	15.80	13.80

$$D_{i,j} = \max \{ A_{i,j-1} + go + ge, D_{i,j-1} + ge \}$$

Gotoh-Rekursion

A		W	T	H	G	Q	A
	0.00	-12.00	-14.00	-16.00	-18.00	-20.00	-22.00
W	-12.00	17.30	5.30	3.30	1.30	-0.70	-2.70
T	-14.00	5.30	19.90	7.90	5.90	3.90	1.90
H	-16.00	3.30	7.90	26.50	14.50	12.50	10.50
A	-18.00	1.30	5.90	14.50	27.80	15.80	14.30

$$A_{i,j} = \max \{ A_{i-1,j-1} + S(s_i,t_j), D_{i,j} , I_{i,j} \}$$

Gotoh in a nutshell

Initialisierung

- $A_{0,k} = A_{k,0} = g(k)$
- $I_{0,j} = D_{i,0} = -Inf$

Rekursion

$I_{i,j} = \max \{ A_{i-1,j} + go + ge, I_{i-1,j} + ge \}$
$D_{i,j} = \max \{ A_{i,j-1} + go + ge, D_{i,j-1} + ge \}$
$A_{i,j} = \max \{ A_{i-1,j-1} + S(s_i,t_j), D_{i,j}, I_{i,j} \}$

Backtracking-Schritt von $A_{i,j}$:

- $A_{i,j} = A_{i-1,j-1} + S(s_i,t_j)$: weiter bei $A_{i-1,j-1}$
- $A_{i,j} = I_{i,j}$: suche k, so dass $A_{i-k,j} + g(k) = A_{i,j}$
- $A_{i,j} = D_{i,j}$: suche k, so dass $A_{i,j-k} + g(k) = A_{i,j}$

Gotoh-Backtracking

A		W	T	H	G	Q	A
	0.00	-12.00	-14.00	-16.00	-18.00	-20.00	-22.00
W	-12.00	17.30	5.30	3.30	1.30	-0.70	-2.70
T	-14.00	5.30	19.90	7.90	5.90	3.90	1.90
H	-16.00	3.30	7.90	26.50	14.50	12.50	10.50
A	-18.00	1.30	5.90	14.50	27.80	15.80	14.30

$$A_{i,j} = \max \{ A_{i-1,j-1} + S(s_i,t_j), D_{i,j}, I_{i,j} \}$$

Gotoh-Backtracking

A		W	T	H	G	Q	A
	0.00	-12.00	-14.00	-16.00	-18.00	-20.00	-22.00
W	-12.00	17.30	5.30	3.30	1.30	-0.70	-2.70
T	-14.00	5.30	19.90	7.90	5.90	3.90	1.90
H	-16.00	3.30	7.90	26.50	14.50	12.50	10.50
A	-18.00	1.30	5.90	14.50	27.80	15.80	14.30

$$A_{i,j} = \max \{ A_{i-1,j-1} + S(s_i,t_j), D_{i,j}, I_{i,j} \}$$

Gotoh-Backtracking

D		W	T	H	G	Q	A
	0.00	-12.00	-14.00	-16.00	-18.00	-20.00	-22.00
W	-Inf	-24.00	5.30	3.30	1.30	-0.70	-2.70
T	-Inf	-26.00	-6.70				
H	-Inf	-28.00	-8.70	-4.10	14.50	12.50	10.50
A	-Inf	-30.00	-10.70	-6.10	2.50	15.80	13.80

In D steht hier auch 12.50.
Der Eintrag in A kam also von dort.

$$D_{i,j} = \max \{ A_{i,j-1} + go + ge, D_{i,j-1} + ge \}$$

Gotoh-Backtracking

D		W	T	H	G	Q	A
	0.00	-12.00	-14.00	-16.00	-18.00	-20.00	-22.00
W	-Inf	-24.00	5.30	3.30	1.30	-0.70	-2.70
T	-Inf	-26.00	-6.70	Hier wurde nur gap-extend bezahlt!			
H	-Inf	-28.00	-8.70	-4.10	14.50	12.50	10.50
A	-Inf	-30.00	-10.70	-6.10	2.50	15.80	13.80

$$D_{i,j} = \max \{ A_{i,j-1} + go + ge, D_{i,j-1} + ge \}$$

Gotoh-Backtracking

D		W	T	H	G	Q	A
	0.00	-12.00	-14.00	-16.00	-18.00	-20.00	-22.00
W	-Inf	-24.00	5.30	3.30	1.30	-0.70	-2.70
T	-Inf	-26.00	-6.70				0
H	-Inf	-28.00	-8.70	-4.10	14.50	12.50	10.50
A	-Inf	-30.00	-10.70	-6.10	2.50	15.80	13.80

Gehe so lange nach links, bis gap-open bezahlt wurde!

$$D_{i,j} = \max \{ A_{i,j-1} + go + ge, D_{i,j-1} + ge \}$$

Gotoh-Backtracking

	A	W	T	H	G	Q	A
	0.00	-12.00	-14.00	-16.00	-18.00	-20.00	-22.00
W	-12.00	17.30	5.30	3.30	1.30	-0.70	-2.70
T	-14.00	5.30	19.90				
H	-16.00	3.30	7.90	26.50		12.50	10.50
A	-18.00	1.30	5.90	14.50	27.80	15.80	14.30

Gehe so lange nach links, bis gap-open bezahlt wurde!

Gotoh-Backtracking

A		W	T	H	G	Q	A
	0.00	-12.00	-14.00	-16.00	-18.00	-20.00	-22.00
W	-12.00	17.30	5.30	3.30	1.30	-0.70	-2.70
T	-14.00	5.30	19.90	7.90	5.90	3.90	1.90
H	-16.00	3.30	7.90	26.50	[illegible]	12.50	10.50
A	-18.00	1.30	5.90	14.50	27.80	15.80	14.30

Gotoh in a nutshell

Initialisierung

- $A_{0,k} = A_{k,0} = g(k)$
- $I_{0,j} = D_{i,0} = -Inf$

Rekursion

$I_{i,j} = \max \{ A_{i-1,j} + go + ge, I_{i-1,j} + ge \}$
$D_{i,j} = \max \{ A_{i,j-1} + go + ge, D_{i,j-1} + ge \}$
$A_{i,j} = \max \{ A_{i-1,j-1} + S(s_i,t_j), D_{i,j}, I_{i,j} \}$

Backtracking-Schritt von $A_{i,j}$:

- $A_{i,j} = A_{i-1,j-1} + S(s_i,t_j)$: weiter bei $A_{i-1,j-1}$
- $A_{i,j} = I_{i,j}$: suche k, so dass $A_{i-k,j} + g(k) = A_{i,j}$
- $A_{i,j} = D_{i,j}$: suche k, so dass $A_{i,j-k} + g(k) = A_{i,j}$

Natürlich ebenfalls als lokale/Freeshift- Variante!

Gotoh-Backtracking

I	W	T	H	G	Q	A
	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf
W	-24.00	-26.00	-28.00	-30.00	-32.00	-34.00
T	5.30	-6.70	-8.70	-10.70	-12.70	-14.70
H	3.30	7.90	-4.10	-6.10	-8.10	-10.10
A	1.30	5.90	14.50	2.50	0.50	-1.50

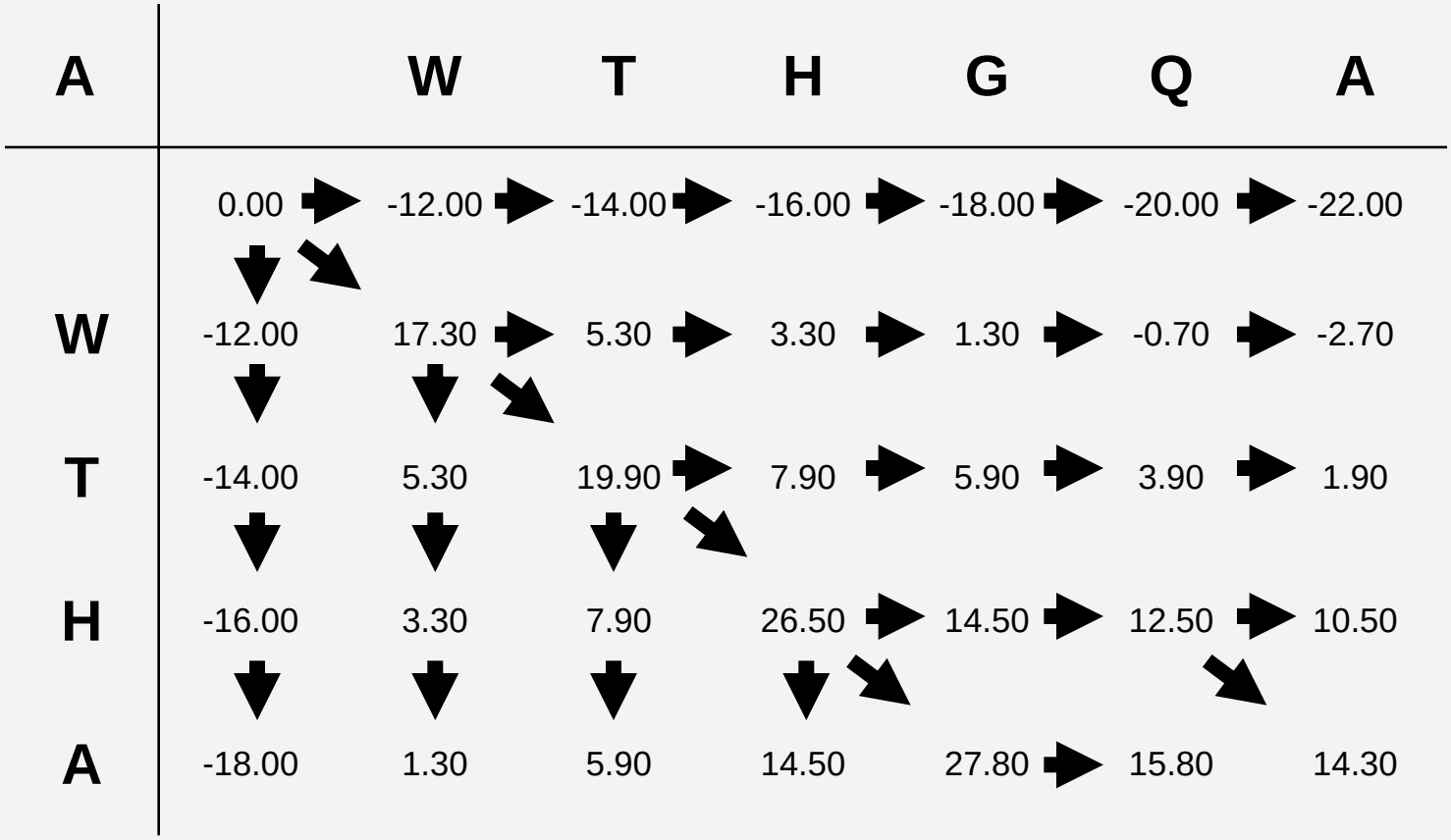
$$I_{i,j} = \max \{ A_{i-1,j} + go + ge, I_{i-1,j} + ge \}$$

Gotoh-Backtracking

D		W	T	H	G	Q	A
W	-Inf	-24.00	5.30	3.30	1.30	-0.70	-2.70
T	-Inf	-26.00	-6.70	7.90	5.90	3.90	1.90
H	-Inf	-28.00	-8.70	-4.10	14.50	12.50	10.50
A	-Inf	-30.00	-10.70	-6.10	2.50	15.80	13.80

$$D_{i,j} = \max \{ A_{i,j-1} + go + ge, D_{i,j-1} + ge \}$$

Gotoh-Backtracking



$$A_{i,j} = \max \{ A_{i-1,j-1} + S(s_i,t_j), D_{i,j} , I_{i,j} \}$$

Zusammenfassung

	Lineare Gapkosten-Funktion	Affine Gapkosten-Funktion	Allgemeine Gapkosten-Funktion	Besonderheiten	Backtracking
Globales Alignment	Needleman-Wunsch	Gotoh	Waterman-Smith-Beyer		**Zelle (|s|,|t|) bis Zelle (0,0)**
Lokales Alignment	Smith-Waterman			**Waterman-Trick**	**Zelle mit max.Eintrag bis erste 0-Zelle**
Freeshift-Alignment				**Waterman-Trick nur in Zellen (0,.)/(.,0)**	**max.Zelle in (|s|,.)/(.,|t|) bis (0,.)/(.,0)**
Zeit	**$O(n^2)$**	**$O(n^2)$**	**$O(n^3)$**		

Korrektheit:

- Alignmentscore kann man mit Referenzimplementierung vergleichen.
- Korrektheit des Backtracking nicht so einfach überprüfbar (warum?)
- Implementiere checkscore: Berechne score aus Alignment und vgl. mit Eintrag in A

Effizienz:

- Zugriff auf S
- Integer/Float Arithmetik
- Unnötige teure Operationen

Zugriff auf S

Aufgabe: Implementiere eine Funktion $\Sigma \times \Sigma \rightarrow R$

Lösung 1: Eine Hash-Tabelle

```
HashMap<String,Double> S = readMatrix();

double S(char a, char b) {
   return S.get(String.valueOf(new char[] {a,b}));
}
```

Probleme:
Jedesmal wird ein Array erstellt und ein Hashwert berechnet.

Zugriff auf S

Aufgabe: Implementiere eine Funktion $\Sigma \times \Sigma \rightarrow R$

Lösung 2: Verschachtelte Hash-Tabellen

```
HashMap<Character,HashMap<Character,Double>> S =
                                          readMatrix();

double S(char a, char b) {
   return S.get(a).get(b);
}
```

Probleme:
Autoboxing, relativ viele Funktionsaufrufe

Zugriff auf S

Aufgabe: Implementiere eine Funktion $\Sigma \times \Sigma \rightarrow R$

Lösung 3: char-indizierte Arrays

```
double[][] S = readMatrix();

double S(char a, char b) {
   return S[a][b];
}
```

Probleme:
?

Zugriff auf S

Aufgabe: Implementiere eine Funktion $\Sigma \times \Sigma \rightarrow R$

Lösung 4: int-indizierte Arrays

```
int[][] sequences = encode(readSequences());
double[][] S = readMatrix();

double S(int a, int b) {
   return S[a][b];
}
```

Vorteil:
S ist kleiner, möglicherweise (!) cache-effizienter

Was wird schneller sein:

```
double S(char a, char b) { … }
double[][] A;
A[i][j] = max(max(I[i][j],D[i][j]),
              A[i-1][j-1]+S(a[i],b[j]));

int S(char a, char b) { … }
int[][] A;
A[i][j] = max(max(I[i][j],D[i][j]),
              A[i-1][j-1]+S(a[i],b[j]));
```

Int-Arithmetik ist schneller als Float-Arithmetik.
Lösung: Multipliziere Substitutionskosten/Gapkosten mit geeignetem (!) Faktor!

Unnötig teure Operationen

Was läuft hier falsch:

```
for (int i=1; i<n; i++) {
   for (int j=1; j<m; j++) {
      S = readMatrix();
      D = …; I = …; A = …;
   }
}
```

Vermeide unnötige Operationen, v.a. wenn sie quadratisch oft ausgeführt werden!

Unnötig teure Operationen

Was läuft hier falsch:

```
double align(char[] a, char[] b) {
   int n = a.length+1, m = b.length+1;
   double[][] A = new double[n][m];
   …
   for (int i=1; i<n; i++) {
      for (int j=1; j<m; j++) {
         D = …; I = …; A = …;
      }
   }
}
```

`new double[n][m]` kostete O(nm) Zeit und produziert viele Objekte, die irgendwann durch die GC wieder freigegeben werden müssen!

Unnötig teure Operationen

Spare so viele Operationen in der inneren Schleife wie möglich!

```
for (int i=1; i<n; i++)
   for (int j=1; j<m; j++)
      D[i][j]=max(D[i][j-1]+ge,A[i][j-1]+go+ge);
```

- go+ge kann vorberechnet werden
- Die Referenzen D[i] und A[i] können vor der inneren Schleife aufgelöst werden

Moderne Compiler machen solche Optimierungen aber teils automatisch!

Gib dem Compiler Tipps! (z.B. deklariere go,ge,n,m als final!)

Das wars!

Fragen?

… dann viel Spaß im GoBi !