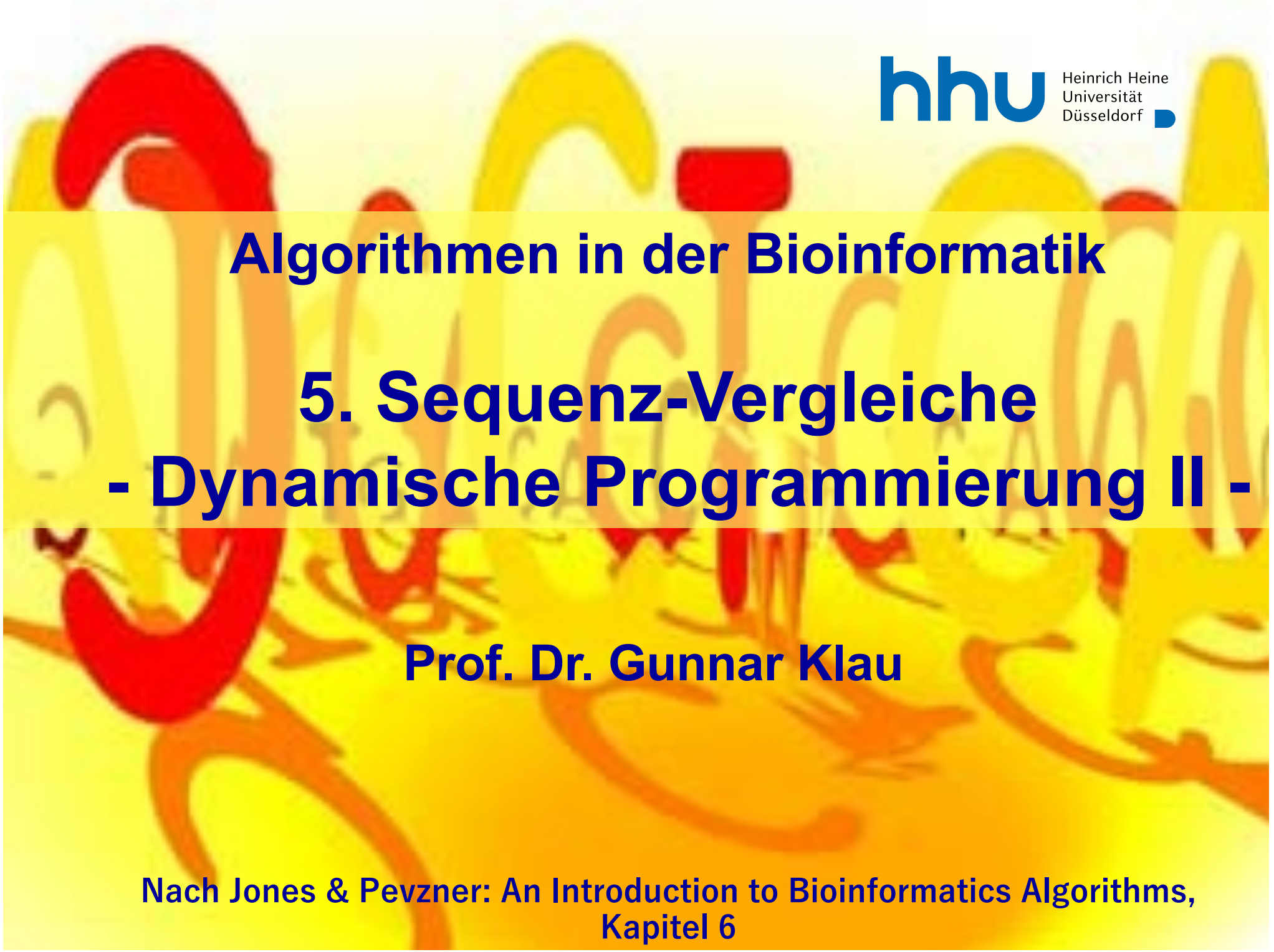


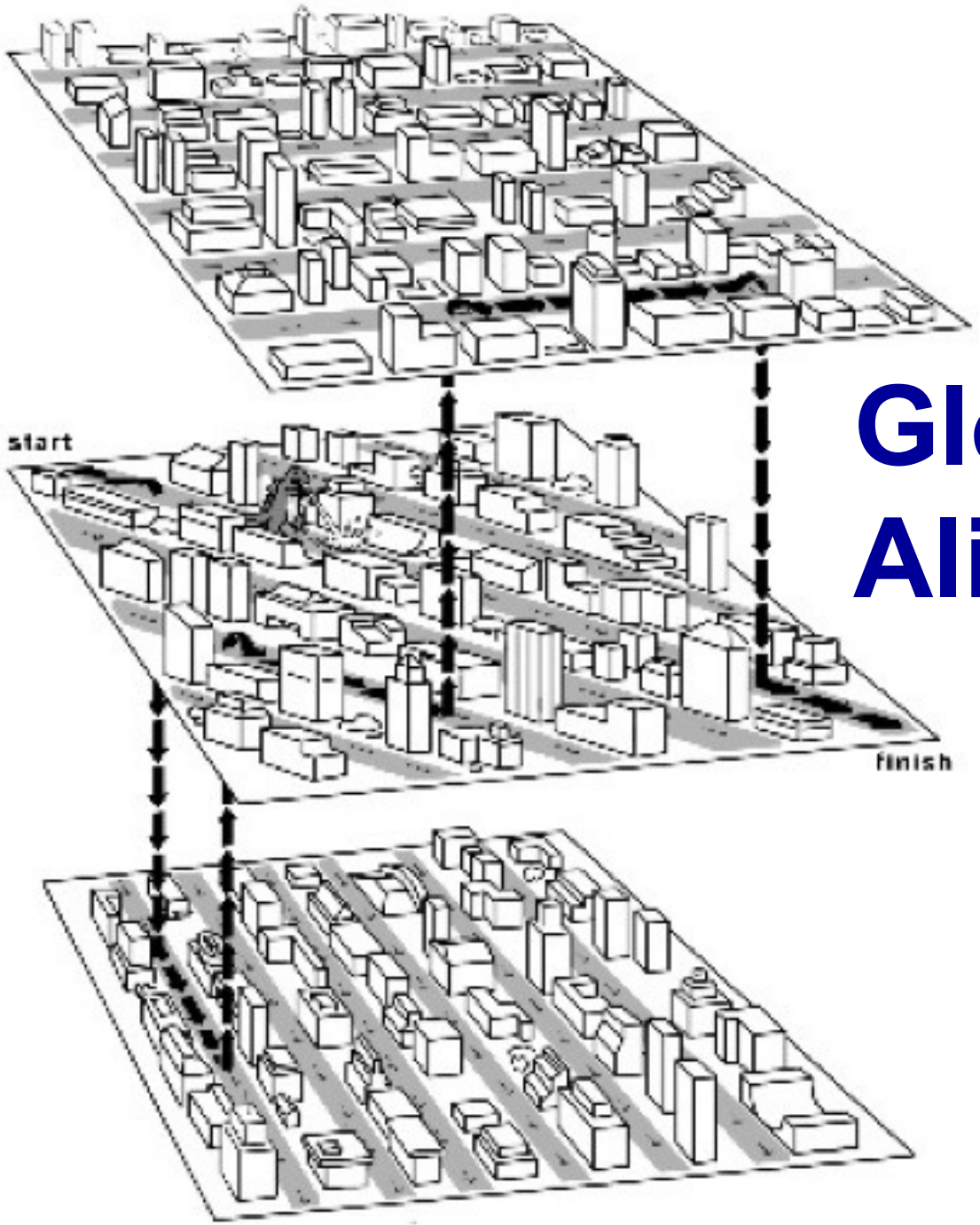
Heinrich Heine Universität Düsseldorf

Algorithmen in der Bioinformatik

5. Sequenz-Vergleiche - Dynamische Programmierung II -

Prof. Dr. Gunnar Klau

Nach Jones & Pevzner: An Introduction to Bioinformatics Algorithms, Kapitel 6



Globales Alignment

LGT → Globales Alignment

- LGT erlaubt nur Matches & Indels (keine Mismatches)
- Score = 1 für Matches, 0 für Indels
- Wir können Indels mit negativem Score bestrafen
- Einfachstes *Scoringschema*:
  - *+1* : Belohnung für **Match**
  - *-μ* : Strafe für **Mismatch**
  - *-σ* : Strafe für **Indel**

Score = # *Matches* – μ (# *Mismatches*) – σ (# *Indels*)

Rekurrenz:

$$s_{i,j} = \mathbf{max} \begin{cases} s_{i-1,j-1} + 1 & \text{wenn } v_i = w_j \\ s_{i-1,j-1} - \mu & \text{wenn } v_i \neq w_j \\ s_{i-1,j} - \sigma \\ s_{i,j-1} - \sigma \end{cases}$$

Scoring-Matrizen

DNA [ATGC-]:
(4+1) x (4+1) **Scoring-Matrix** $\delta(v_i, w_j)$

Aminosäuren [ARNDCEQGHILKMFPSTWYV-]:
(20+1) x (20+1) **Scoring-Matrix** $\delta(v_i, w_j)$

$\Rightarrow$ Vereinfachte Rekurrenz-Relation:

$$s_{i,j} = \mathbf{max} \begin{cases} s_{i-1,j-1} + \delta(v_i, w_j) \\ s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \end{cases}$$

Herstellung von Scoring-Matrizen

- Basierend auf biologischen Daten
- Alinierte Sequenzen sind durch Mutationen aus einem gemeinsamen Vorfahren entstanden
- Mutationen können sehr verschiedene Auswirkungen auf Fitness haben → Scores $\delta(v_i, w_j)$ können sehr unterschiedlich sein
- Aminosäure-Austausche sind umso wahrscheinlicher, je ähnlicher die Codons sind.
- Aminosäure-Veränderungen sind weniger kritisch, wenn sie physiko-chemische Eigenschaften erhalten, z.B.:
  - polar (Aspartat → Glutamat)
  - nicht-polar (Alanin → Valin)
  - pos. Ladung (Arginin → Lysin)

AKRANR

KAAANK

-1-1-2+5+7+3

	Alanin	Arginin	Asparagin	Lysin
	A	R	N	K
A	5	-2	-1	-1
R	-	7	-1	3
N	-	-	7	0
K	-	-	-	6

Alignment von kodierender DNA

Der genetische Code ist redundant

First letter	Second letter: U	Second letter: C	Second letter: A	Second letter: G	Third letter
U	UUU, UUC: Phenylalanine	UCU, UCC, UCA, UCG: Serine	UAU, UAC: Tyrosine	UGU, UGC: Cysteine	U C A G
	UUA, UUG: Leucine		UAA: Stop codon; UAG: Stop codon	UGA: Stop codon; UGG: Tryptophan	
C	CUU, CUC, CUA, CUG: Leucine	CCU, CCC, CCA, CCG: Proline	CAU, CAC: Histidine; CAA, CAG: Glutamine	CGU, CGC, CGA, CGG: Arginine	U C A G
A	AUU, AUC, AUA: Isoleucine	ACU, ACC, ACA, ACG: Threonine	AAU, AAC: Asparagine; AAA, AAG: Lysine	AGU, AGC: Serine; AGA, AGG: Arginine	U C A G
	AUG: Methionine; start codon				
G	GUU, GUC, GUA, GUG: Valine	GCU, GCC, GCA, GCG: Alanine	GAU, GAC: Aspartic acid; GAA, GAG: Glutamic acid	GGU, GGC, GGA, GGG: Glycine	U C A G

- DNA ist weniger konserviert als Aminosäuren
- Alignments von DNA werden besser, wenn man **zuerst Aminosäuren aliniert**, und dann das Alignment zurück in DNA übersetzt!

PAM - Scoring-Matrix für Aminosäuren

PAM = Point **A**ccepted **M**utation (Dayhoff et al.)

- 1 PAM = PAM_1 = 1% aller Positionen unterschiedlich
- $PAM_{100} = (PAM_1)^{100}$ (Mutationen wirken auf das schon mutierte!)
  → nicht jede Position ist verändert
  - manche Positionen mutieren mehrmals, andere selten

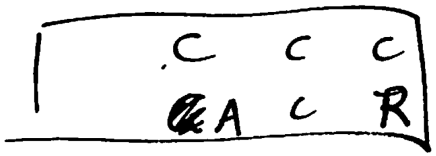
PAM_{250}

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W
A	2																	
R	-2	6																
N	0	0	2															
D	0	-1	2	4														
C	-2	-4	-4	-5	4													
Q	0	1	1	2	-5	4												
E	0	-1	1	3	-5	2	4											
G	1	-3	0	1	-3	-1	0	5										
H	-1	2	2	1	-3	3	1	-2	6									
I	-1	-2	-2	-2	-2	-2	-2	-3	-2	5								

Aminosäure	3-Buchstaben Code	1-Buchstaben Code	Molekular-gewi
Alanin	Ala	**A**	89
Arginin	Arg	**R**	174
Aspargin	Asn	**N**	132
Asparginsäure	Asp	**D**	133
Cystein	Cys	**C**	121
Glutamine	Gln	**Q**	146
Glutaminsäure	Glu	**E**	147
Glycin	Gly	**G**	175
Histidin	His	**H**	155
Isoleucin	Ile	**I**	131
Leucin	Leu	**L**	131
Lysin	Lys	**K**	146
Methionin	Met	**M**	149
Phenylanalin	Phe	**F**	165
Prolin	Pro	**P**	115
Serin	Ser	**S**	105
Threonin	Thr	**T**	119
Tryptophan	Trp	**W**	204
Tyrosin	Tyr	**Y**	181
Valin	Val	**V**	117

$M(i,j) \approx$ wie häufig ~~ersetzt~~ mutiert i-te AA zu j-te AA in verwandten Alignm. Seq.

$M(i,j)$ = Anteil aller AA i, die mit AA j alinie~~t~~ sind



$f(j)$ = Anteil AA j in der Seq

$$PAM^1(i,j) = \log\left(\frac{M(i,j)}{f(j)}\right)$$

eigentlich 10 $\log_{10}$(fraction), hier 10 $\log_{10}(1.5)$ = ca. 10 * 0.176...

$$PAM^1(C,A) = \log\left(\frac{1/4}{1/6}\right) = + \ldots$$

BLOSUM - Scoring-Matrix für Aminosäuren

BLOSUM = BLOck **SU**bstitution **M**atrix

- Scores aus beobachteten Häufigkeiten von Substitutionen in Blöcken von **lokalen** Alignments verwandter Proteine
- Subskript gibt an, wie ähnlich die Proteine einander waren (BLOSUM62 aus Sequenzen mit ca. 62% Identität)
- nicht extrapoliert
- PAM für verwandte, BLOSUM für divergente Sequenzen

$BLOSUM_{50}$

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	5	-2	-1	-2	-1	-1	-1	0	-2	-1	-2	-1	-1	-3	-1	1	0	-3	-2	0	-2	-1	-1	-5
R	-2	7	-1	-2	-4	1	0	-3	0	-4	-3	3	-2	-3	-3	-1	-1	-3	-1	-3	-1	0	-1	-5
N	-1	-1	7	2	-2	0	0	0	1	-3	-4	0	-2	-4	-2	1	0	-4	-2	-3	4	0	-1	-5
D	-2	-2	2	8	-4	0	2	-1	-1	-4	-4	-1	-4	-5	-1	0	-1	-5	-3	-4	5	1	-1	-5
C	-1	-4	-2	-4	13	-3	-3	-3	-3	-2	-2	-3	-2	-2	-4	-1	-1	-5	-3	-1	-3	-3	-2	-5
Q	-1	1	0	0	-3	7	2	-2	1	-3	-2	2	0	-4	-1	0	-1	-1	-1	-3	0	4	-1	-5
E	-1	0	0	2	-3	2	6	-3	0	-4	-3	1	-2	-3	-1	-1	-1	-3	-2	-3	1	5	-1	-5
G	0	-3	0	-1	-3	-2	-3	8	-2	-4	-4	-2	-3	-4	-2	0	-2	-3	-3	-4	-1	-2	-2	-5
H	-2	0	1	-1	-3	1	0	-2	10	-4	-3	0	-1	-1	-2	-1	-2	-3	2	-4	0	0	-1	-5
I	-1	-4	-3	-4	-2	-3	-4	-4	-4	5	2	-3	2	0	-3	-3	-1	-3	-1	4	-4	-3	-1	-5
L	-2	-3	-4	-4	-2	-2	-3	-4	-3	2	5	-3	3	1	-4	-3	-1	-2	-1	1	-4	-3	-1	-5
K	-1	3	0	-1	-3	2	1	-2	0	-3	-3	6	-2	-4	-1	0	-1	-3	-2	-3	0	1	-1	-5
M	-1	-2	-2	-4	-2	0	-2	-3	-1	2	3	-2	7	0	-3	-2	-1	-1	0	1	-3	-1	-1	-5
F	-3	-3	-4	-5	-2	-4	-3	-4	-1	0	1	-4	0	8	-4	-3	-2	1	4	-1	-4	-4	-2	-5
P	-1	-3	-2	-1	-4	-1	-1	-2	-2	-3	-4	-1	-3	-4	10	-1	-1	-4	-3	-3	-2	-1	-2	-5
S	1	-1	1	0	-1	0	-1	0	-1	-3	-3	0	-2	-3	-1	5	2	-4	-2	-2	0	0	-1	-5
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	2	5	-3	-2	0	0	-1	0	-5
W	-3	-3	-4	-5	-5	-1	-3	-3	-3	-3	-2	-3	-1	1	-4	-4	-3	15	2	-3	-5	-2	-3	-5
Y	-2	-1	-2	-3	-3	-1	-2	-3	2	-1	-1	-2	0	4	-3	-2	-2	2	8	-1	-3	-2	-1	-5
V	0	-3	-3	-4	-1	-3	-3	-4	-4	4	1	-3	1	-1	-3	-2	0	-3	-1	5	-4	-3	-1	-5
B	-2	-1	4	5	-3	0	1	-1	0	-4	-4	0	-3	-4	-2	0	0	-5	-3	-4	5	2	-1	-5
Z	-1	0	0	1	-3	4	5	-2	0	-3	-3	1	-1	-4	-1	0	-1	-2	-2	-3	2	5	-1	-5
X	-1	-1	-1	-1	-2	-1	-1	-2	-1	-1	-1	-1	-1	-2	-2	-1	0	-3	-1	-1	-1	-1	-1	-5
*	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	1

Globales Alignment: Problem

Ziel: Finde das beste Alignment für 2 Sequenzen
bei vorgegebener Scoring-Matrix.

Eingabe: Strings **v, w** und eine Scoring-Matrix δ

Ausgabe: Ein Alignment zwischen **v** und **w** mit maximalem Score

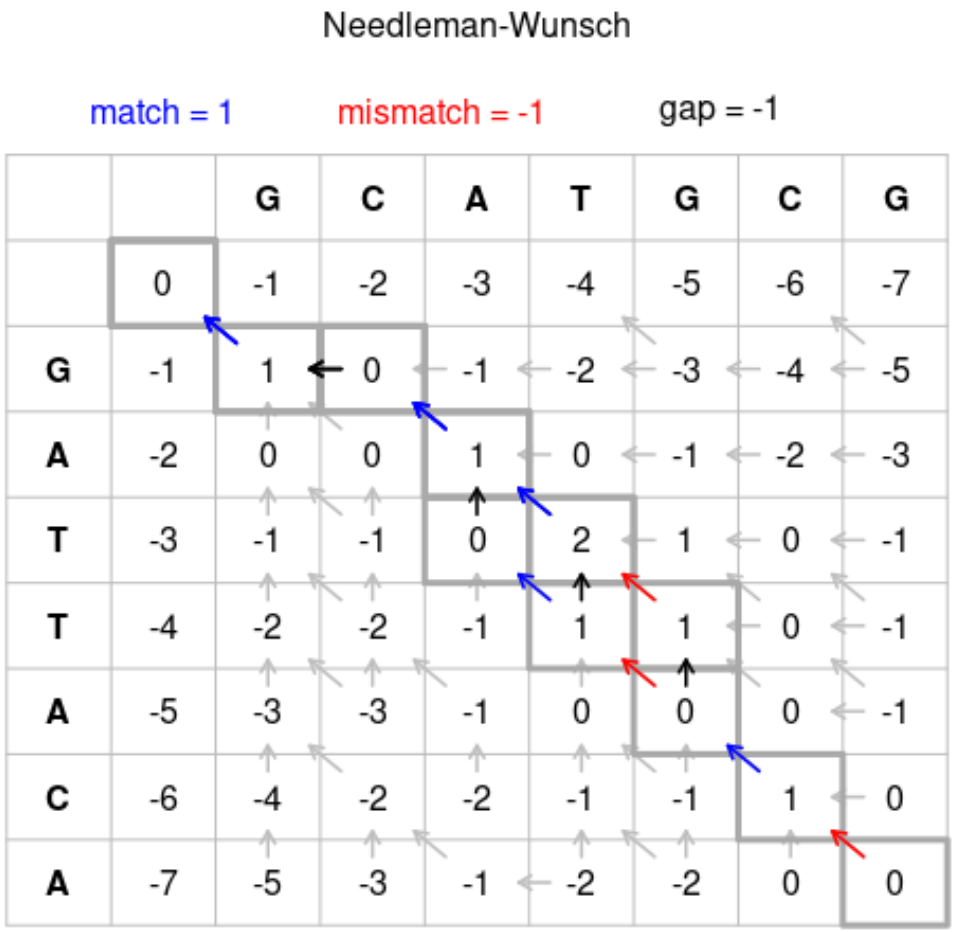
Lösung: Dynamische Programmierung (à la LGT) mit Rekurrenz

$$s_{i,j} = \mathbf{max} \begin{cases} s_{i-1,j-1} + \delta(v_i, w_j) \\ s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \end{cases}$$

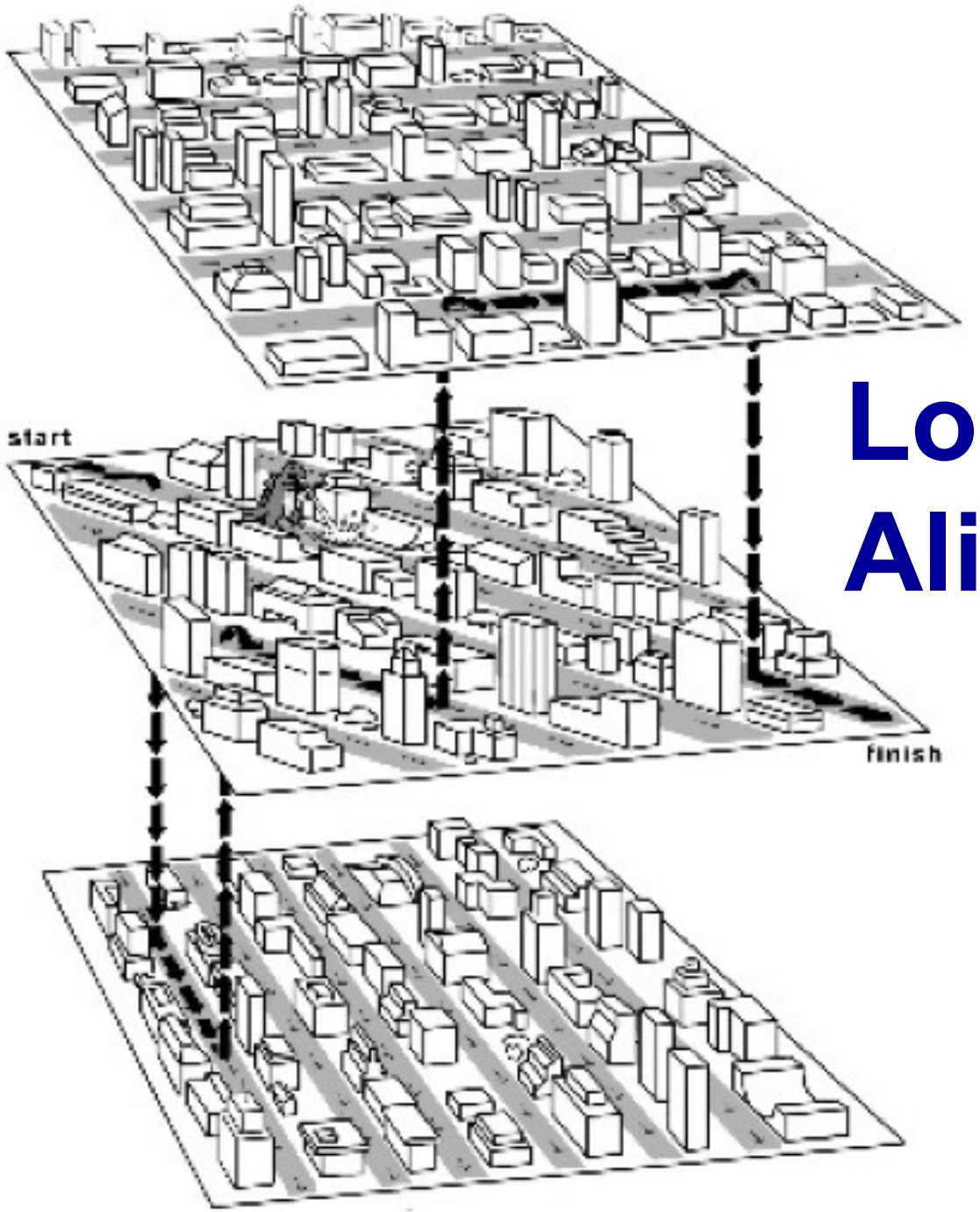
Needleman-Wunsch-Algorithmus

Globales Alignment: Beispiel

- v = GCATGCG, w = GATTACA. Match = 1, Mismatch = Gap = -1
- Siehe https://en.wikipedia.org/wiki/Needleman–Wunsch_algorithm



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



Lokales Alignment

Lokales vs. Globales Alignment

- Globales Alignment sucht den längsten Pfad zwischen (0,0) und (*n,m*) im Edit-Graphen

```
--T--CC-C-AGT--TCTGT-CAGGGGACACG-A-GCATGCAGA-GAC
  |  || |  ||  | | | |||    || | | |  | ||||   |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG-T-CAGAT--C
```

- Lokales Alignment sucht den längsten Pfad zwischen beliebigen Knoten (*i,j*) und (*i', j'*) im Edit-Graphen

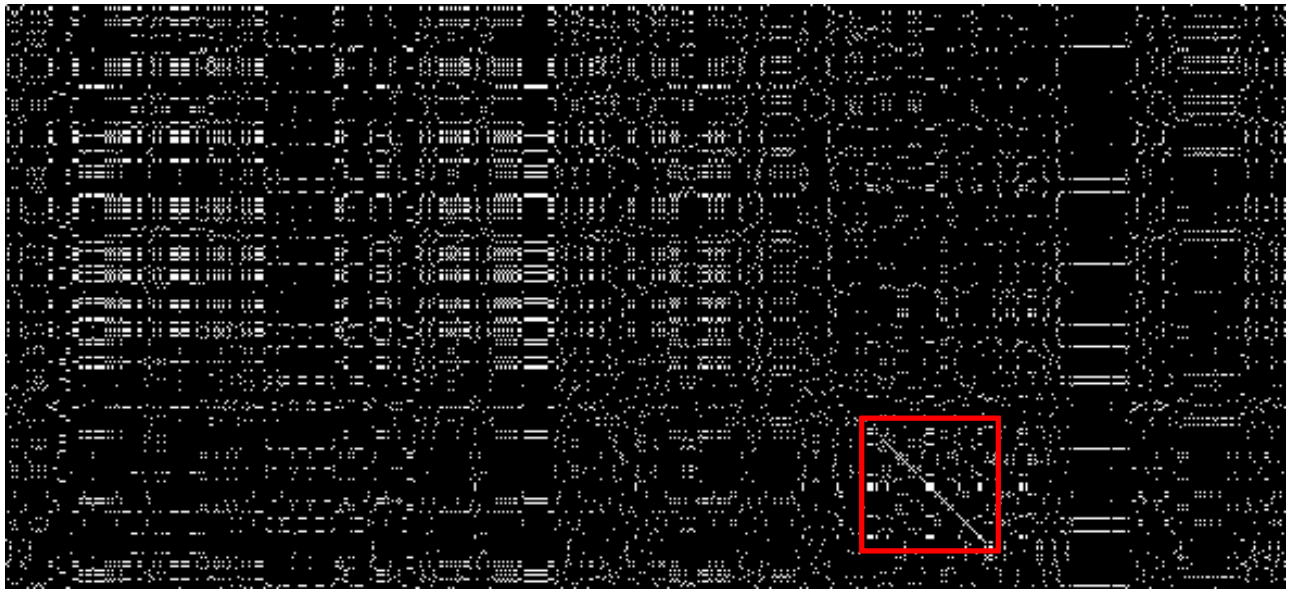
```
                    tccCAGTTCTGTCAGgggacacgagcatgcagagac
                       ||||| ||||||
aattgccgccgtcgttttcagCAGTTATGTCAGatc
```

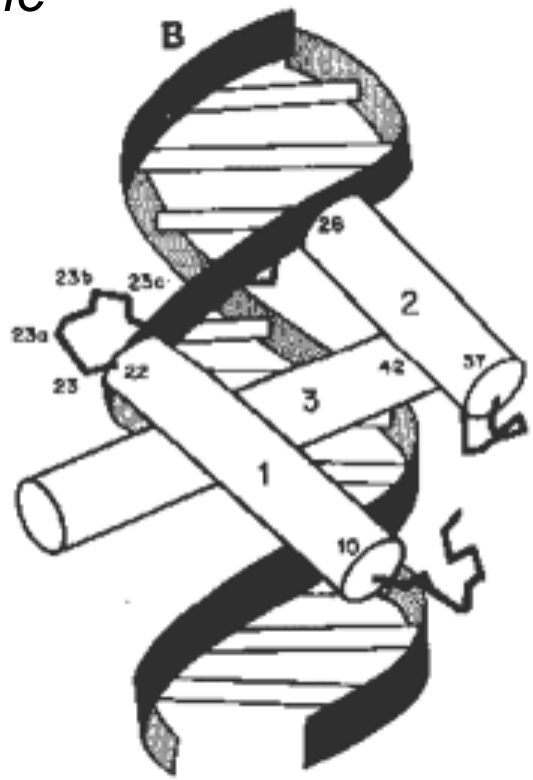
Durch negative Scores δ (v_i, w_j) kann ein lokales Alignment den höheren Score haben!

Warum lokale Alignments?

Zwei entfernt verwandte Gene mit konservierter Funktion können anhand von funktionell wichtigen Regionen erkannt werden

Beispiel: Homöobox-Gene haben einen *Homöo-Domäne*
- globales Alignment würde die übersehen!



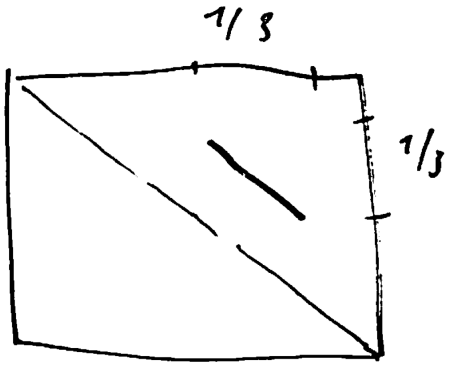


Homeobox

YTRFQTLELEKEFHFNHYLTRRRRIEIAHALCLTERQIKIWFQNRRMKLKK

YTR-QTLELEKEFHFN-YLTRRRRIEIAHALCL-ERQIKIWFQNRRMK-KK

Fiktives Bsp.:



Perfektes lok. Ali, 1/3 de Länge

Score (lok. Ali) ≥

$1/3\,(1) + \frac{2}{3}(-\sigma) + \frac{2}{3}(-\sigma) =$

(1): Match; $(-\sigma)$: Gap

$\frac{1}{3} - \frac{4}{3}\sigma$

Score (glob) ≥

$1/4 + \frac{3}{4}(-\mu) = \frac{1}{4} - \frac{3}{4}\mu$

$\frac{3}{4}(-\mu)$: Mismatch.

$\frac{1}{3} - \frac{4}{3}\sigma < \frac{1}{4} - \frac{3}{4}\mu$

$\longleftrightarrow \frac{4}{3}\sigma - \frac{1}{3} > \frac{3}{4}\mu - \frac{1}{4}$

$\longleftrightarrow \frac{4}{3}\sigma > \frac{3}{4}\mu + \frac{1}{12}$

$\longleftrightarrow \sigma > \frac{9}{16}\mu + \frac{1}{16}$

Meistens $\mu \ll 1$.!

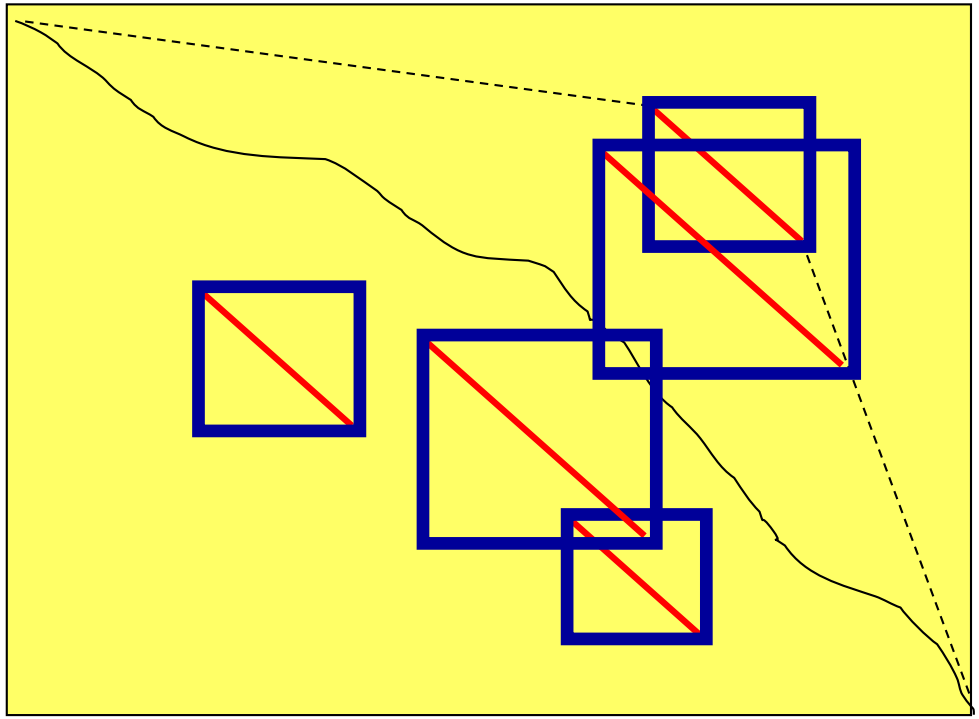
Lokales Alignment: Problem

Ziel: Finde das beste lokale Alignment zwischen 2 Sequenzen

Eingabe: Sequenzen **v, w** und Scoring-Matrix δ

Ausgabe: Alignment von Teilsequenzen von **v** und **w** mit maximalem Score (verglichen mit allen Alignments aller Teilsequenzen)

Lokales Alignment: Laufzeit



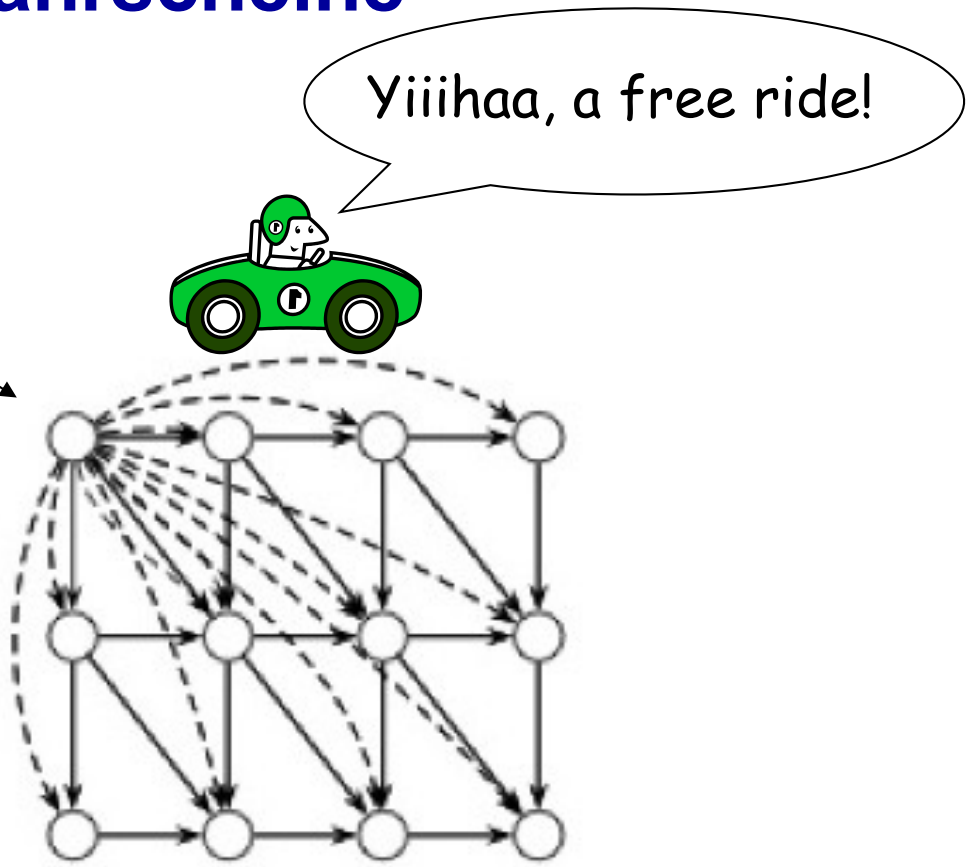
Laufzeit $O(n^4)$

- im n x n Gitter gibt es n^2 Knoten (i,j), die als Quellen dienen können
- Berechnung aller Alignments von diesem Startpunkt braucht $O(n^2)$

Schnellere Lösung?

Freifahrscheine!

Lokales Alignment: Freifahrscheine



Gestrichelte Kanten sind Freifahrten von **(0,0)** zu allen Knoten

max. $s_{i,j}$ im Edit-Graphen entspricht dem besten lokalen Alignment

Rekurrenz:

$$s_{i,j} = \mathbf{max} \begin{cases} 0 \\ s_{i-1,j-1} + \delta(v_i, w_j) \\ s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \end{cases}$$

Unterschied zur Rek. globalem Alignment

Needleman-Wunsch → Smith-Waterman

Smith-Waterman-Algorithmus

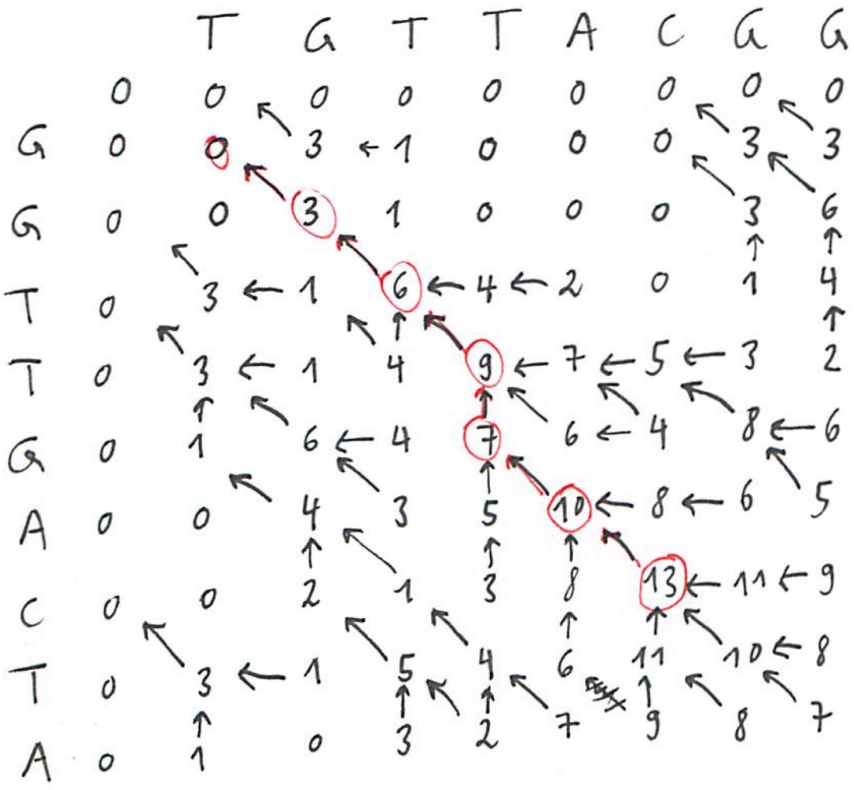
S. auch https://en.wikipedia.org/wiki/Smith–Waterman_algorithm#Example

Sequenzen T G T T A C G G und
G G T T G A C T A

$$\delta(a,b) = \begin{cases} +3 & a=b \\ -3 & a \neq b \end{cases}$$

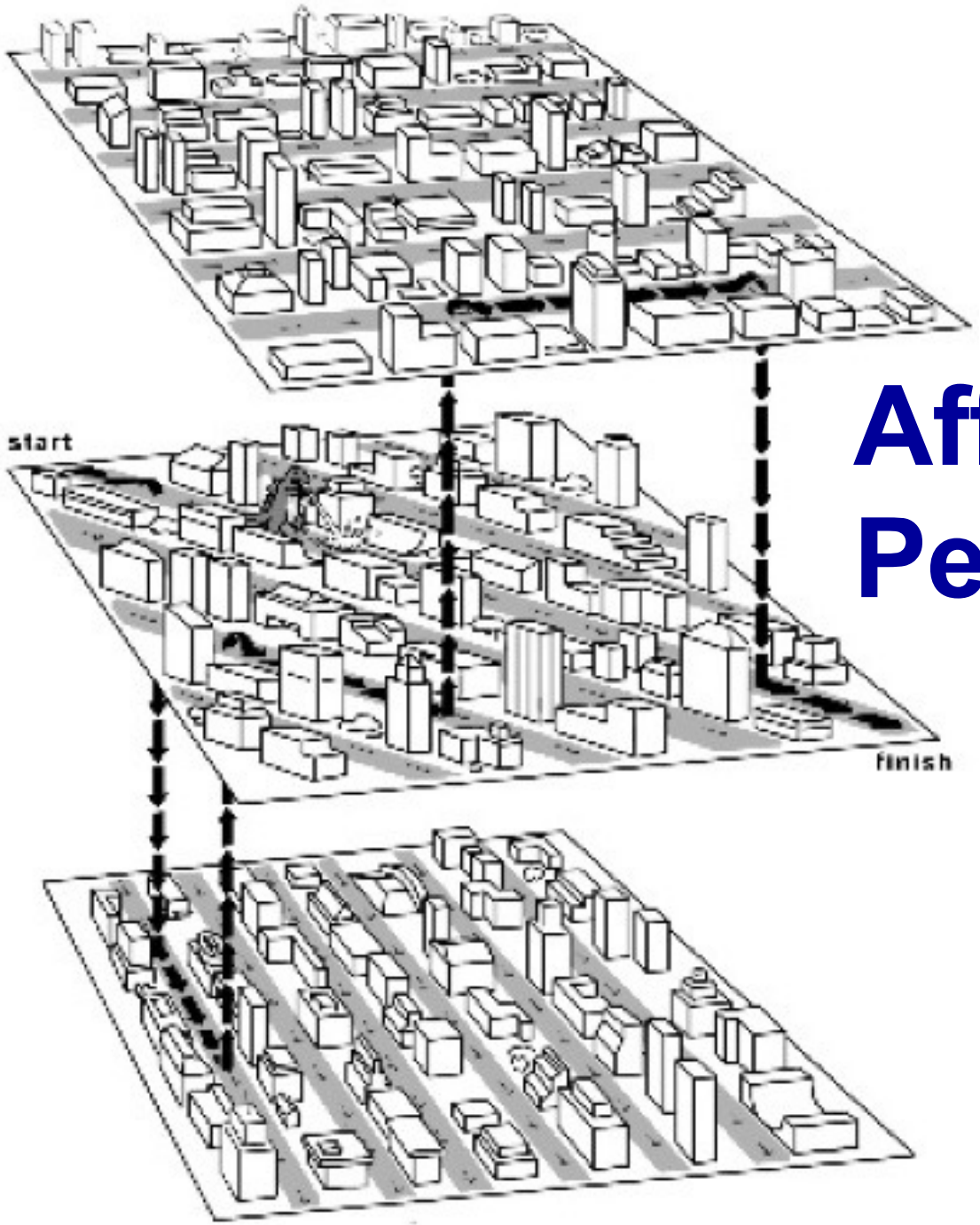
$\delta(a,-) =$

$\delta(-,a) = -2$



G T T – A C

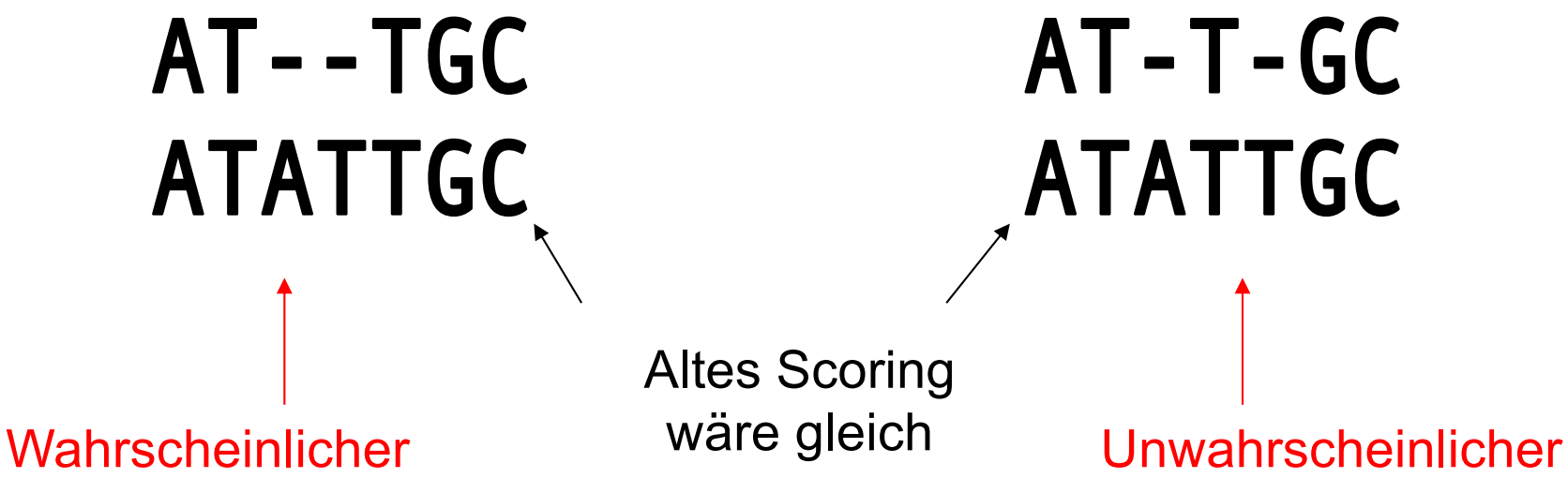
G T T G A C



Affine Lücken-Penalties

Affine Lücken-Penalties

- Bisher: Strafkosten σ für jedes Indel
- Biologie: eine Mutation mit *k* Indels ist wahrscheinlicher als *k* einzelne Mutationen:



Affine Lücken-Penalties

Score für eine Lücke der Länge *k*:

$$-(\sigma + \varepsilon(k-1))$$

mit $\sigma > 0$ Strafe für das Öffnen der Lücke:

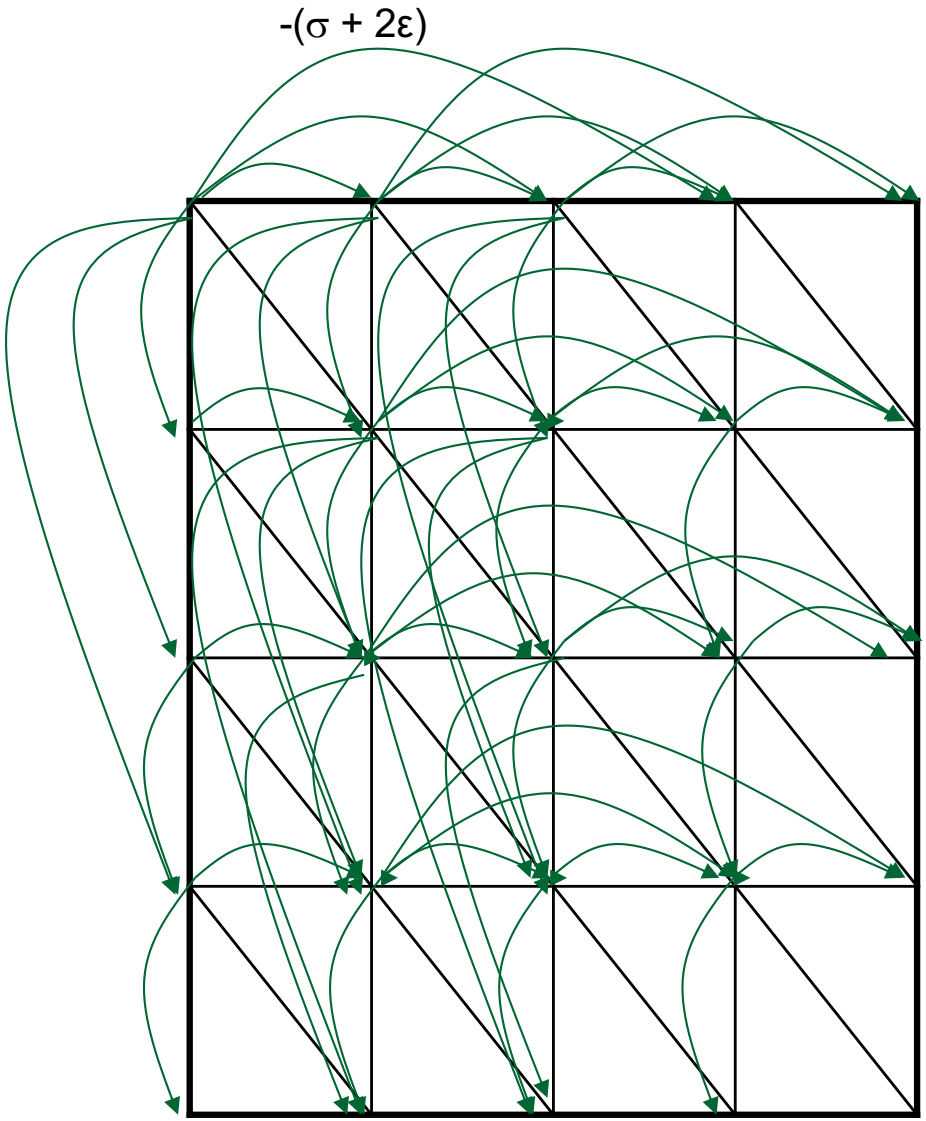
Gap opening penalty

und $\varepsilon > 0$ Strafe für das weitere Vergrößern der Lücke:

Gap extension penalty

In der Regel $\sigma > \varepsilon$: Vergrößern ist billiger (biologisch wahrscheinlicher) als das Öffnen einer neuen Lücke

Affine Gaps und der Edit-Graph



Affine Gaps entsprechen langen (k) horizontalen und vertikalen Kanten mit Gewicht $-(\sigma + \varepsilon(k-1))$

Davon gibt es viele

$\rightarrow$ Komplexität steigt von $O(n^2)$ auf $O(n^3)$

- Außer wir sind schlau!

Anzahl grüne Kanten:

Betrachte Knoten (i,j): $(n-j)$ horiz. Kanten + $(n-i)$ vertikale Kanten.

Insgesamt $\sum_{i=1}^{n} \sum_{j=1}^{n} (n-i) + \sum_{i=1}^{n} \sum_{j=1}^{n} (n-j)$

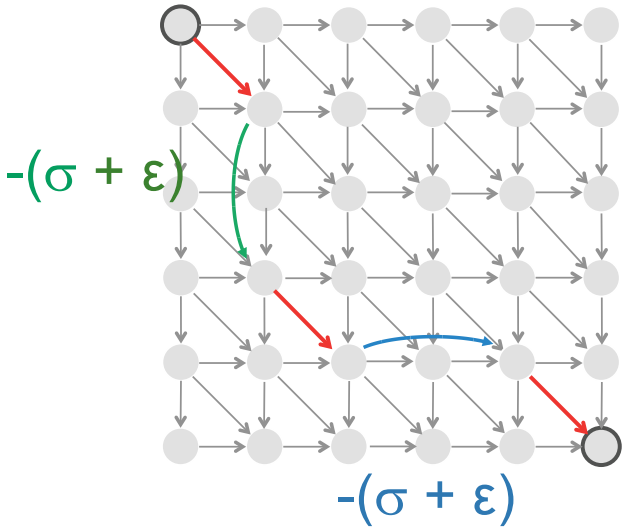
$$= 2 \sum_{i=1}^{n} \sum_{j=1}^{n} (n-i) = 2n^3 - \sum_{i=1}^{n} \sum_{j=1}^{n} i$$

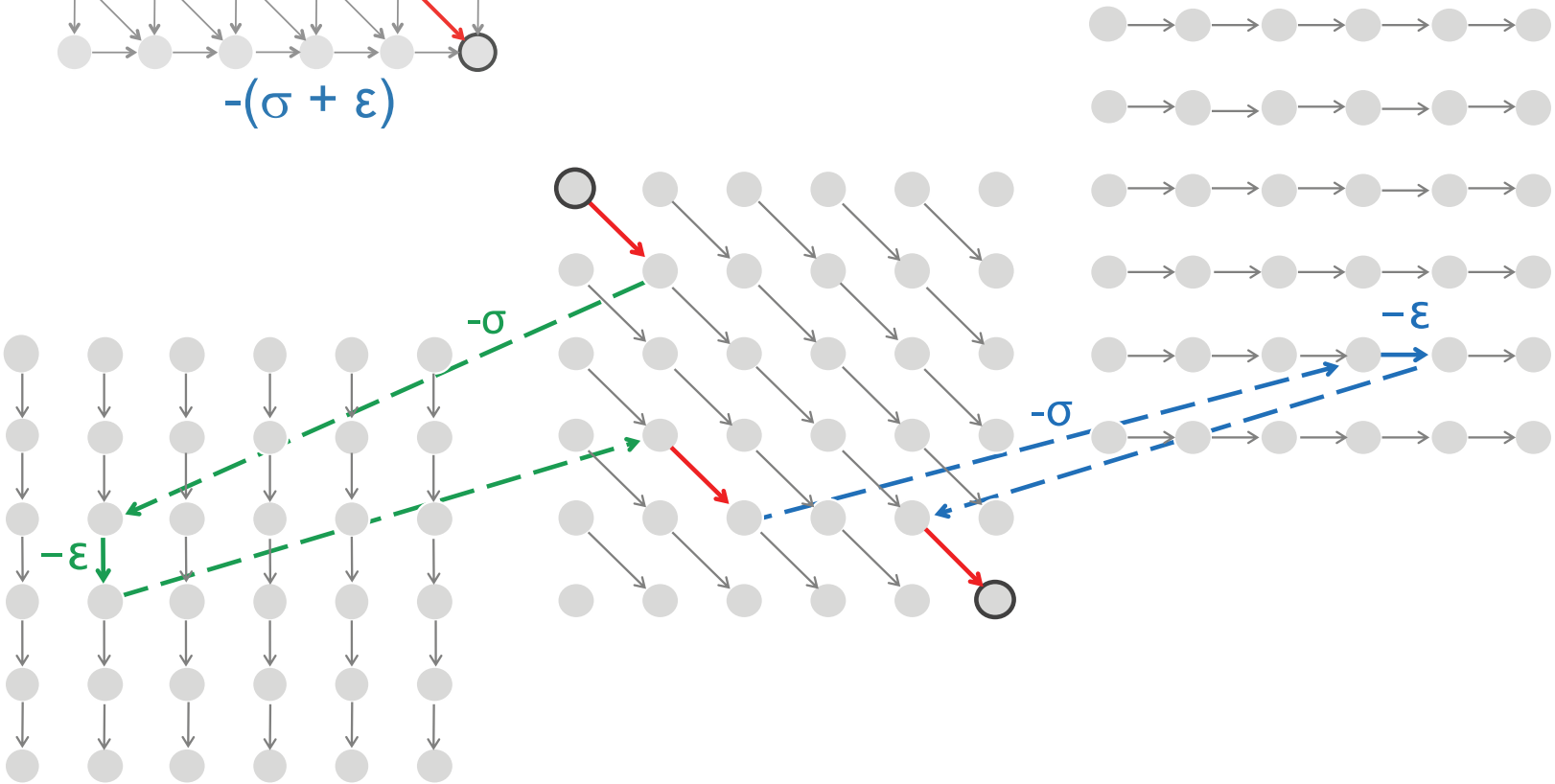
$$= 2n^3 - \sum_{i=1}^{n} \frac{n(n+1)}{2}$$

$$= 2n^3 - \frac{n \cdot n \cdot (n+1)}{2}$$

$$= 2n^3 - \frac{n^3 + n^2}{2} = \Theta(n^3)$$

Beispiel und Idee



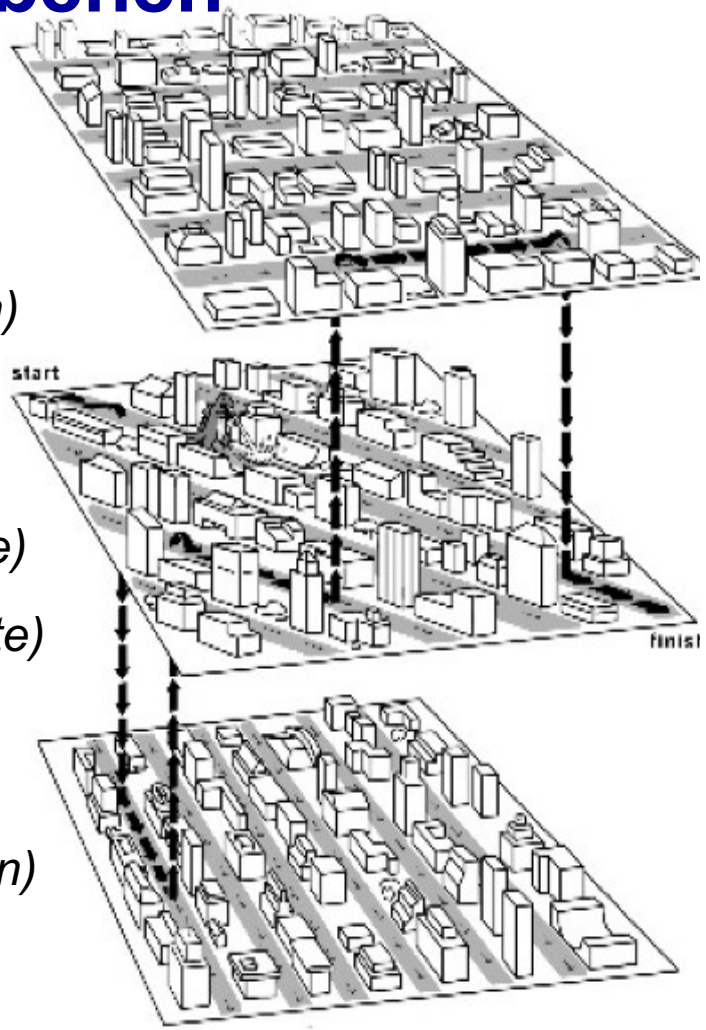


Affine Lücken: Manhattan in 3 Ebenen

$$\downarrow s_{i,j} = \max \begin{cases} \downarrow s_{i,j-1} - \varepsilon & \textit{Setze Lücke in w fort} \\ s_{i,j-1} - \sigma & \textit{Öffne Lücke in w (Mitte} \rightarrow \textit{ oben)} \end{cases}$$

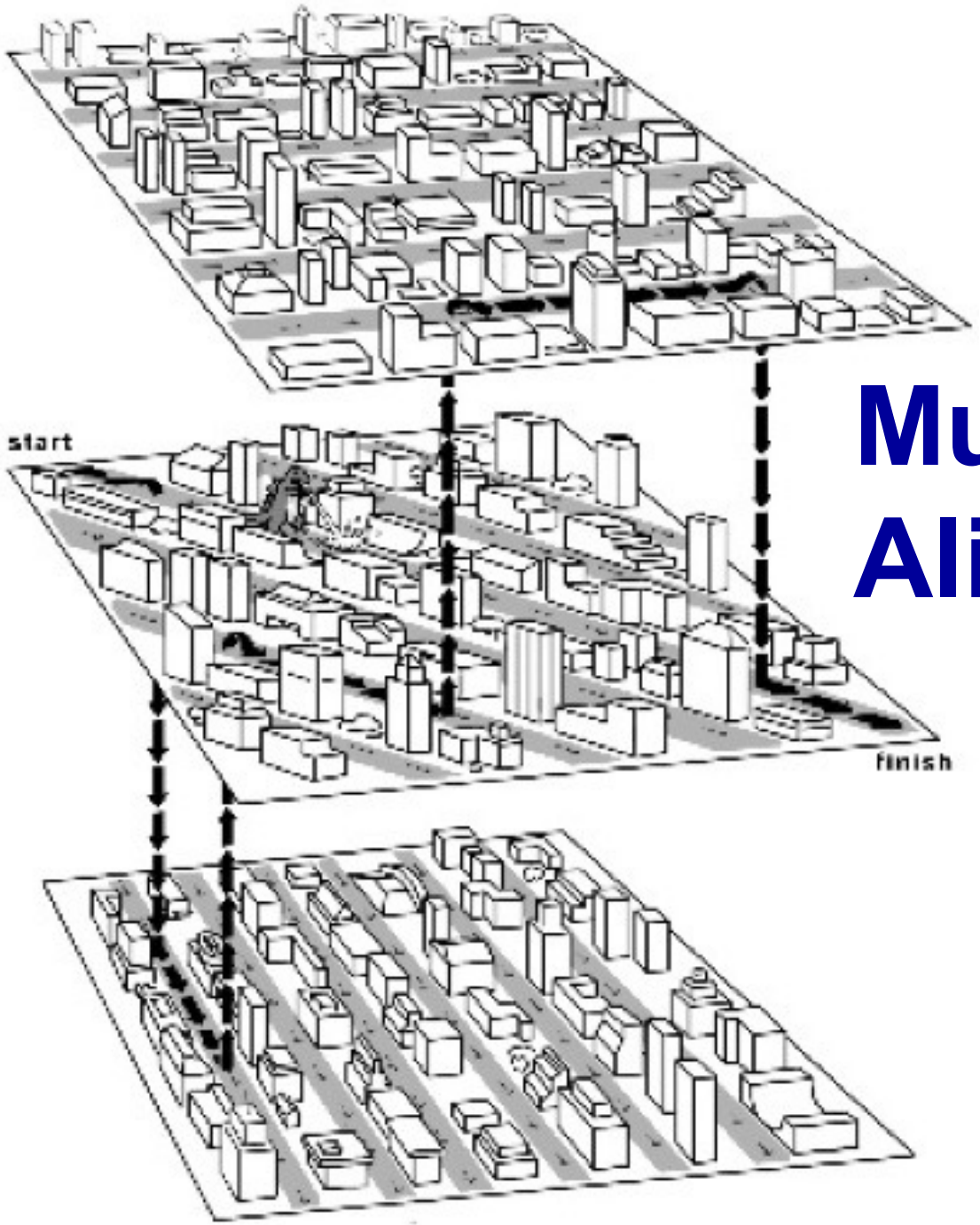
$$s_{i,j} = \max \begin{cases} s_{i-1,j-1} + \delta(v_i, w_j) & \textit{Match oder Mismatch} \\ \downarrow s_{i,j} & \textit{Ende Lücke in w (oben} \rightarrow \textit{ Mitte)} \\ \rightarrow s_{i,j} & \textit{Ende Lücke in v (unten} \rightarrow \textit{Mitte)} \end{cases}$$

$$\rightarrow s_{i,j} = \max \begin{cases} \rightarrow s_{i-1,j} - \varepsilon & \textit{Setze Lücke in v fort} \\ s_{i-1,j} - \sigma & \textit{Öffne Lücke in v (Mitte} \rightarrow \textit{unten)} \end{cases}$$



- Sprung-Strafe -σ für Bewegungen vom Hauptlevel nach oben oder unten
- Fortsetzungs-Strafe -ε für Schritte im oberen oder unteren Level

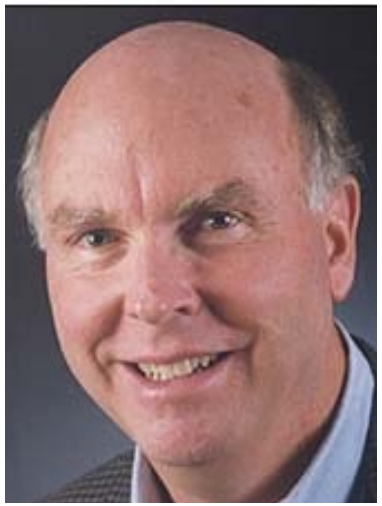
Noch weitere Änderungen für Initialisierung → Übung.

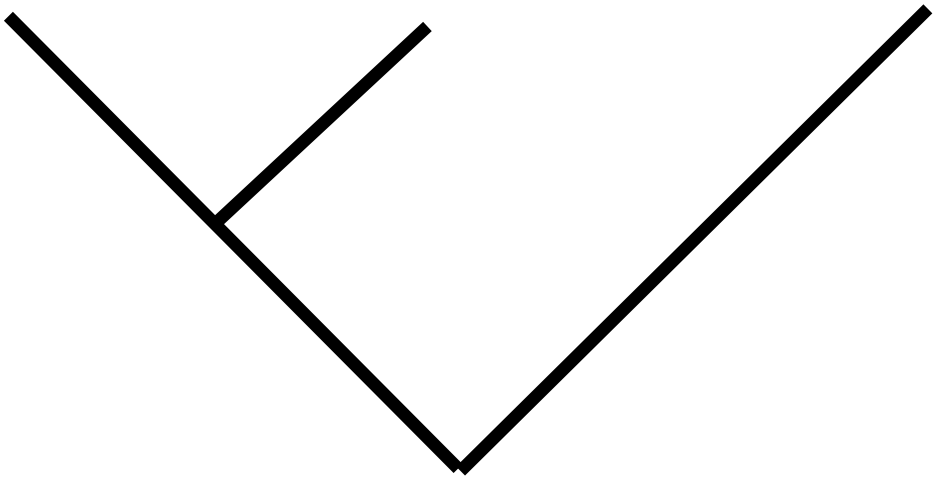


Multiples Alignment

Multiples Alignment

Um Veränderungen zu verstehen, brauchen wir >2 Sequenzen

Verallgemeinerung von paarweisem Alignment

Alignment von 2 Sequenzen ~ 2-reihige Matrix

```
ATGCG-
ACGT-A
```

Alignment von 3 Sequenzen ~ 3-reihige Matrix (etc.)

```
AT-GCG-
A-CGT-A
ATCAC-A
```

...

Score: mehr Info (mehr konservierte Positionen)
→ besseres Alignment

Alignment-Pfade

0	1	1	2	3	4
	A	-	T	G	C
0	1	2	3	3	4
	A	A	T	-	C
0	0	1	2	3	4
	-	A	T	G	C

x-Koordinate

y-Koordinate

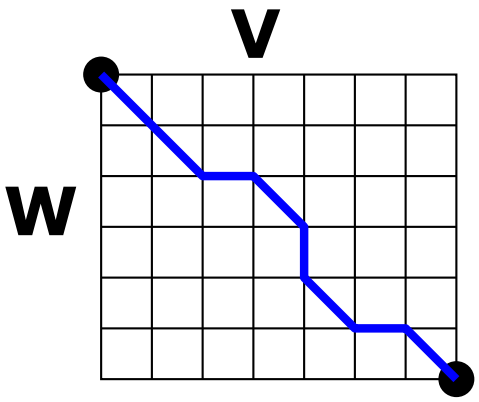
z-Koordinate

Pfad in 3-D:

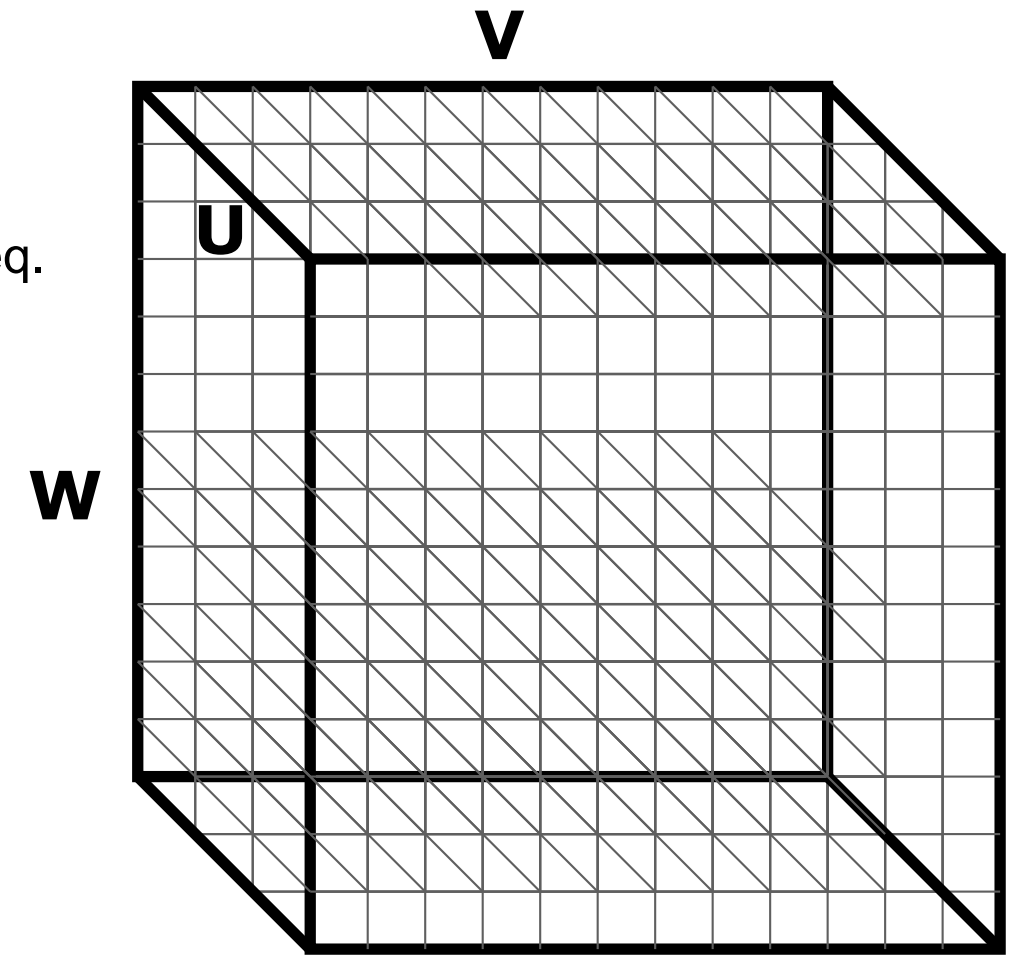
(0,0,0) → (1,1,0) → (1,2,1) → (2,3,2) → (3,3,3) → (4,4,4)

2-D vs. 3-D Alignment-Gitter

- Selbe Strategie wie in 2-D
- 3-D “Manhattan-Würfel”,
  jede Achse repräsentiert 1 Seq.
- Gehe von Quelle zu Senke
  (für globales Alignment)

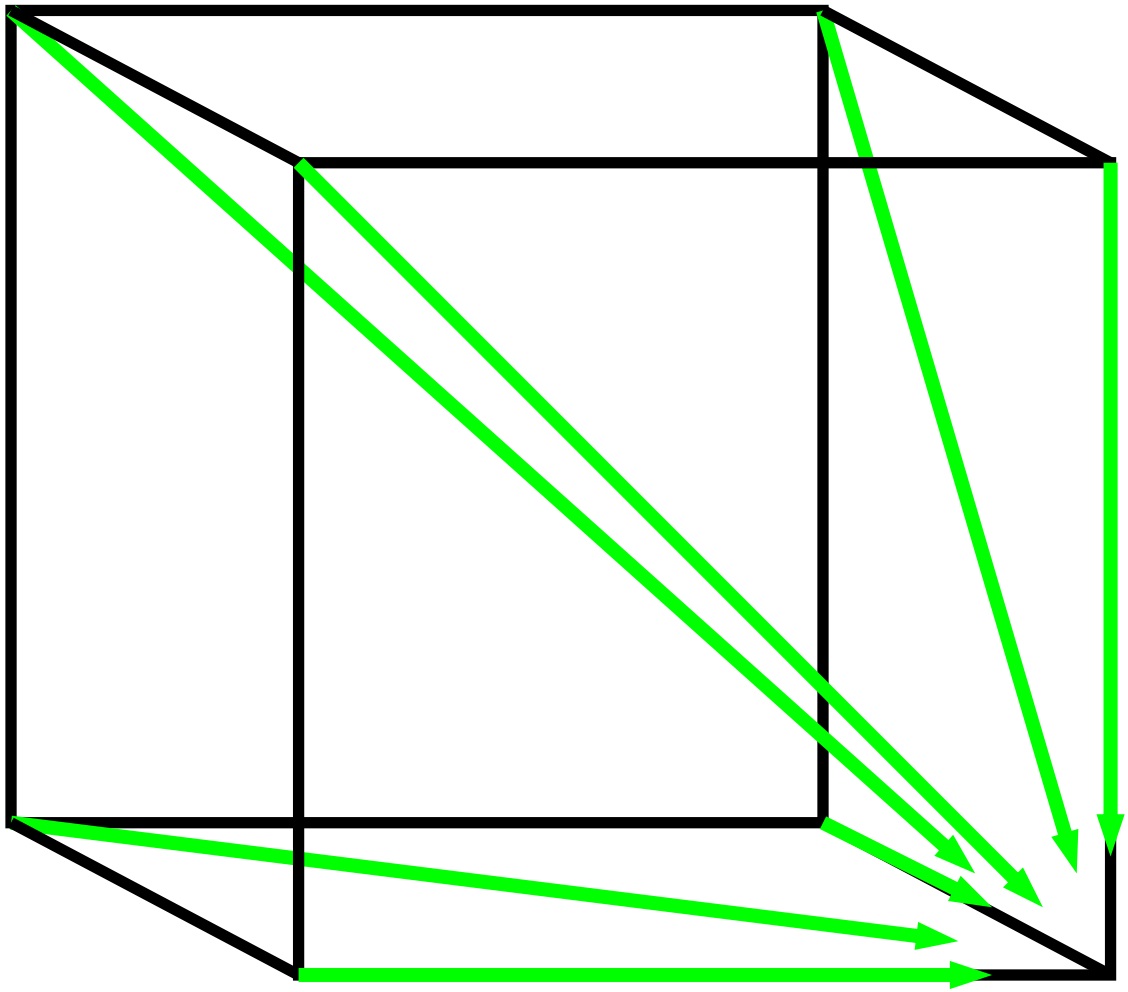


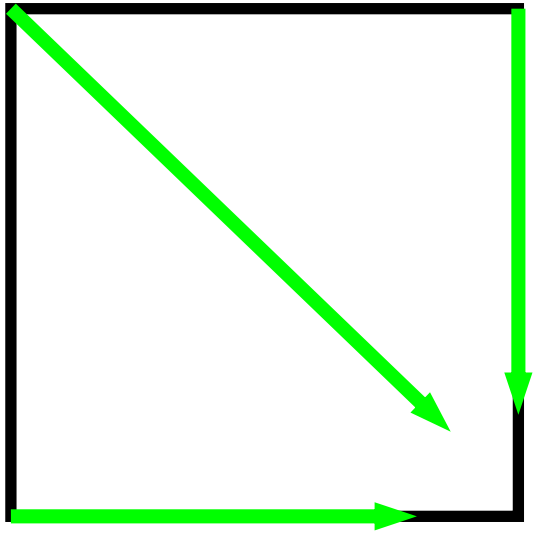
2-D edit graph



3-D edit graph

3-D vs. 2-D Alignment-Zellen





In **2-D**, 3 Kanten pro Quadrat

In **3-D**, 7 Kanten pro Würfel

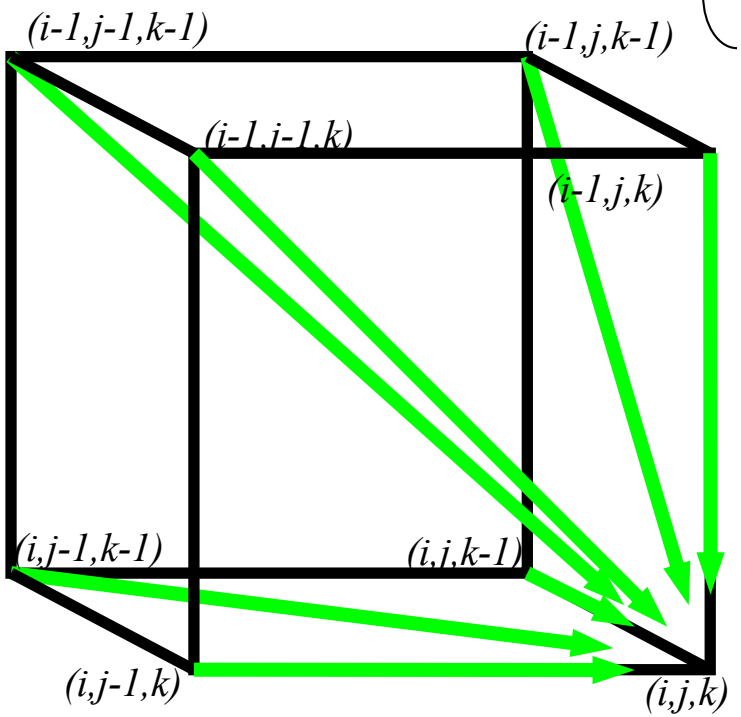
Multiples Alignment: Rekurrenz

$$s_{i,j,k} = \max \begin{cases} s_{i-1,j-1,k-1} + \delta(v_i, w_j, u_k) \\ s_{i-1,j-1,k} + \delta(v_i, w_j, \_) \\ s_{i-1,j,k-1} + \delta(v_i, \_, u_k) \\ s_{i,j-1,k-1} + \delta(\_, w_j, u_k) \\ s_{i-1,j,k} + \delta(v_i, \_, \_) \\ s_{i,j-1,k} + \delta(\_, w_j, \_) \\ s_{i,j,k-1} + \delta(\_, \_, u_k) \end{cases}$$

Würfel-Diagonale: 0 Indels

Flächen-Diagonale: 1 Indel

Würfelkante: 2 Indels



$\delta(x, y, z)$ ist ein Eintrag der 3-D Scoring-Matrix

Multiples Alignment: Laufzeit

Für 3 Sequenzen der Länge ***n:***

$7n^3 \rightarrow O(n^3)$

Für ***k*** Sequenzen: ***k***-dimensionales Manhattan:

$(2^k-1)(n^k) \rightarrow O(2^k n^k)$

Dynamische Programmierung kann leicht auf ***k*** Sequenzen verallgemeinert werden, aber Laufzeit ist inpraktikabel

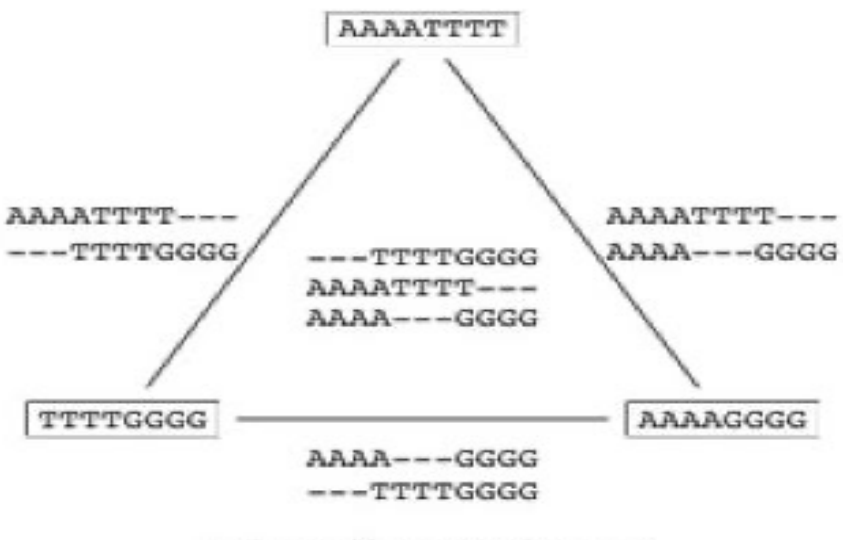
Multiples Alignment ↔ paarweises Alignment

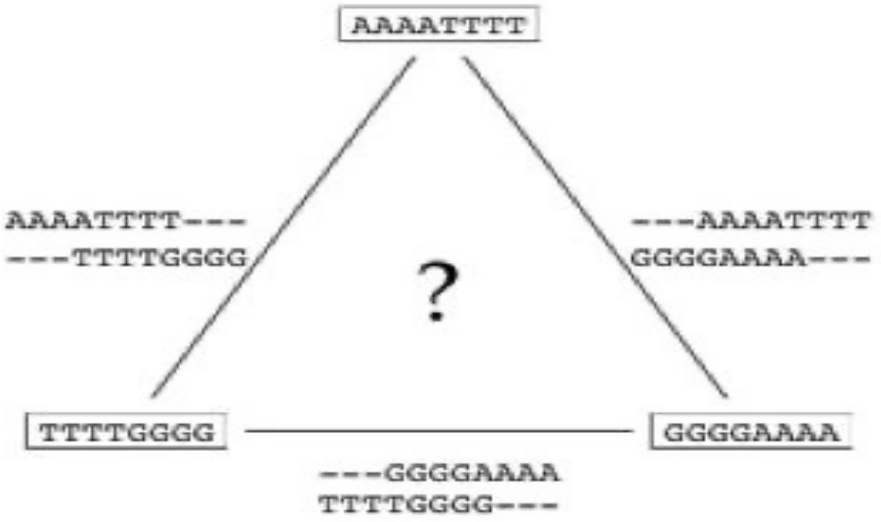
Multiples Alignment induziert 3 paarweise Alignments:

```
x: AC-GCGG-C
y: AC-GC-GAG
z: GCCGC-GAG
```

→ `x: ACGCGG-C;` `y: ACGC-GAC;` `x: AC-GCGG-C;` `z: GCCGC-GAG;` `y: AC-GCGAG` `z: GCCGCGAG`

Induzieren 3 paarweise Alignments ein multiples Alignment?

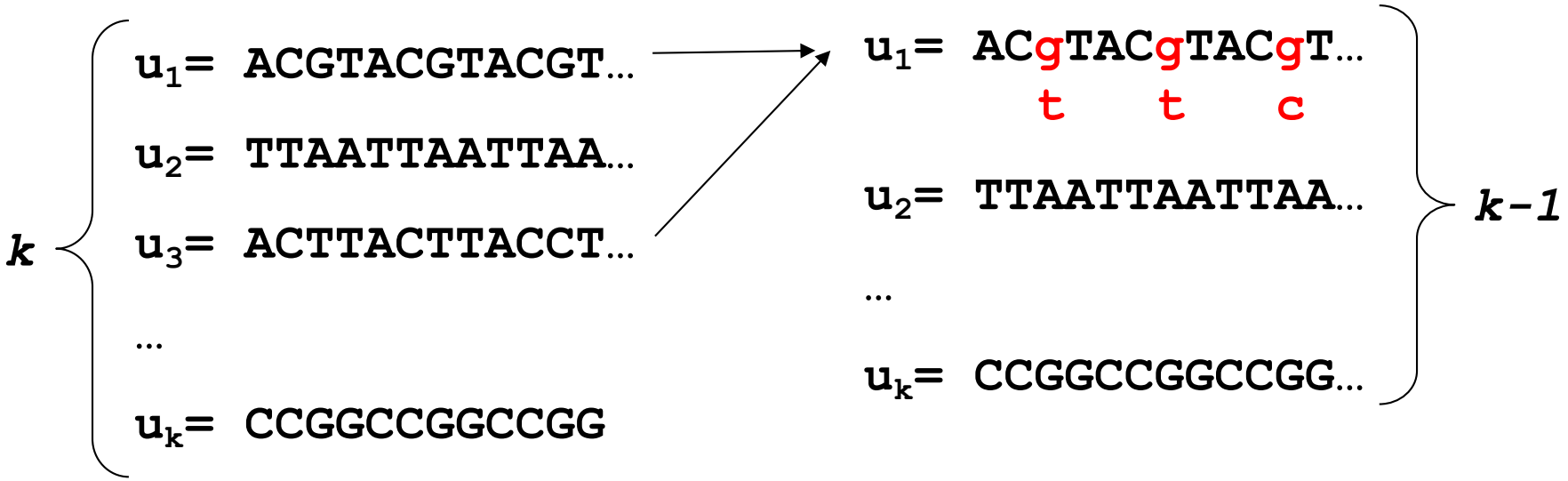




Multiples Alignment: Gieriger Algorithmus

Wähle die 2 ähnlichsten* Sequenzen und kombiniere sie in ein Alignment:
$k \rightarrow k$ - 1 Sequenzen/Profile

* besser: die am nächsten verwandten



Profil-Repräsentation eines multiplen Alignments

Alignment:

```
-  A  G  G  C  T  A  T  C  A  C  C  T  G
T  A  G  -  C  T  A  C  C  A  -  -  -  G
C  A  G  -  C  T  A  C  C  A  -  -  -  G
C  A  G  -  C  T  A  T  C  A  C  -  G  G
C  A  G  -  C  T  A  T  C  G  C  -  G  G
```

Profil:

```
A      1           1        .8
C   .6        1        .4  1     .6 .2
G            1 .2                 .2        .4  1
T   .2                 1     .6                .2
-   .2           .8                       .4 .8 .4
```

Progressives Alignment: Integriere die Sequenzen schrittweise in das Profil, so dass das Profil nach *i* Schritten *i*+1 Sequenzen repräsentiert

ClustalW

1. Paarweise Alignments
2. → Führungs-Baum (für Reihenfolge)
3. Progressives Alignment (Gieriger Alg. mit Profilen)

Probleme:

- Gaps im Profil können nicht korrigiert werden
- Profile berücksichtigen Verwandtschaft nicht mehr