Algorithms and Tools in Bioinformatics

Algorithms: Sequence Alignment
(adapted from Prof. Stephan Winkler)

Julia Vetter
julia.vetter@fh-hagenberg.at

SS2024

(3) Global/Local Alignments

Dynamic Programming

Methods of Sequence Alignment

- Dotplot Analysis
  - first impressions
  - shows indels, repeats

- Dynamic Programming
  - optimal alignment
  - checks all possible combinations
  - cpu intensive

- Word methods
  - collect “islands”
  - fast, heuristic
  - for DB searches

Global Pairwise Alignment

Exact comparison of similar sequences with approximately the same sequence length

– e.g., characterization of protein families

– e.g., determination of a consensus sequence for multiple sequence alignment

A global alignment of two strings S1 and S2 is obtained by inserting spaces in and/or at the ends of S1 and S2 so that the resulting strings have the same length;

Then you place one string on top of the other so that each character in one string faces exactly one character in the other string.

Examples:

```
GCTACTAG-T-T--CGC-T-TAGC
GCTACTAGCTCTAGCGCGTATAGC

GCTACTAGTT------CGCTTAGC
GCTACTAGCTCTAGCGCGTATAGC
```

Local Pairwise Alignment

- finds interesting regions in unknown sequences, e.g., different species, different genes, ...
- finds conserved (=functional) subsequences, first step to the protein families; later also within the family
- finds conserved areas to study lineage/evolutionary processes

Let S1, S2 be two character strings. Find substrings α and β of S1 and S2 that are most similar among all possible pairs of substrings.

Example for local alignment:

```
S1: HEAG  AWGHE    E    AWGHE
S2: P     AW-HE   AE    AW-HE
```

Comments

- Global alignments:
  - easy to find
  - score grows with length of sequences

- High value local alignments are not always found
  - in random sequences, e.g.

- Algorithms for sequence alignment calculation:

  dynamic programming

Dynamic Programming

1. Characterize the solution space and the structure of the desired optimal solution
2. Define recursively how an optimal solution (and its associated value) is composed of smaller optimal solutions (and their values).
3. Conceive the algorithm in a bottom-up way in such a way that for n=1,2,3, ... tabular optimal partial solutions (and their associated values) are found. When finding a specific optimal partial solution of size k, all optimal partial solutions of size < k must be used.

Prerequisite: The optimal solution to a problem of size n consists of optimal subsolutions of smaller size. (Bellmann's optimality principle)

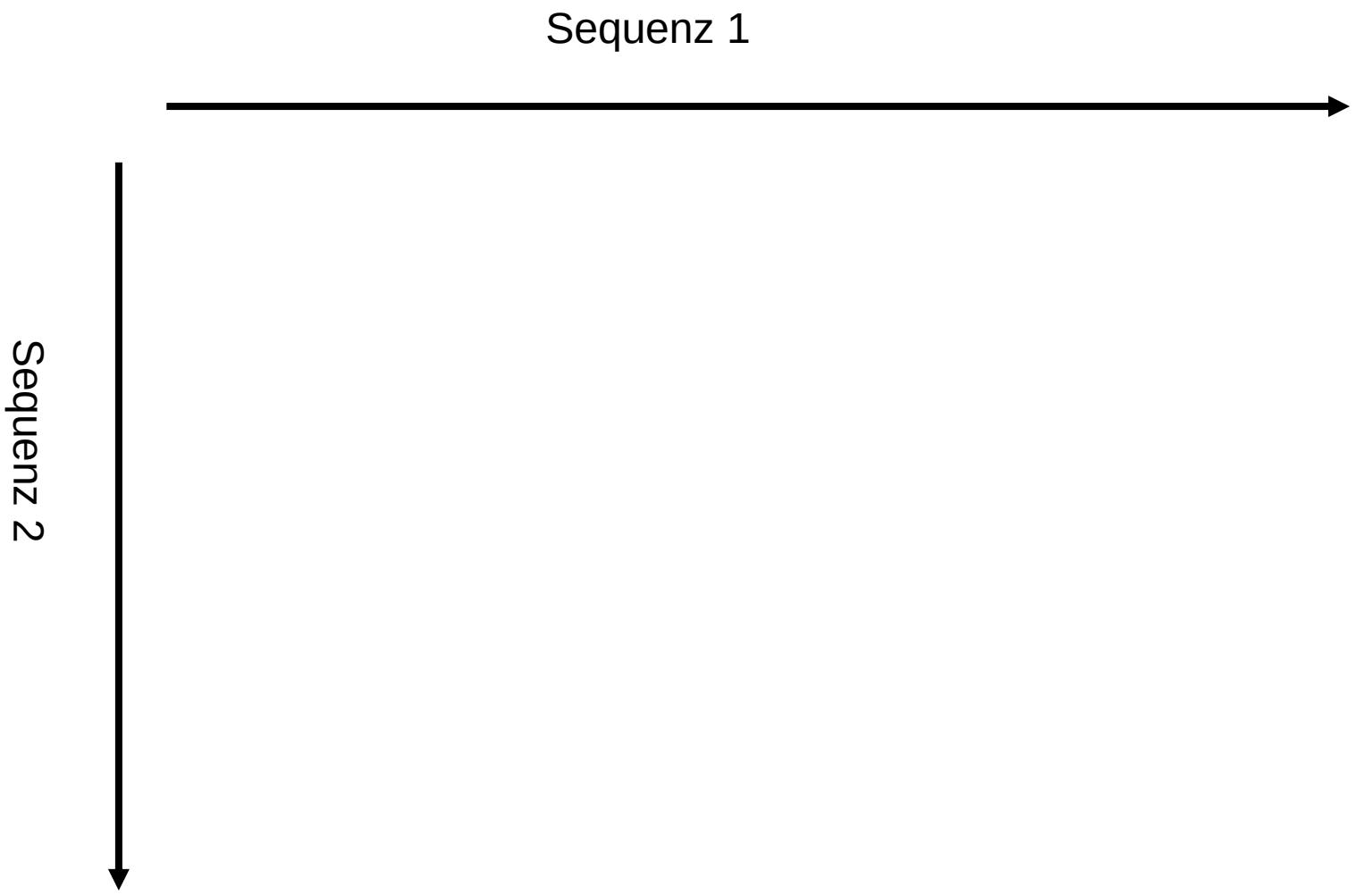
Dynamic Programming

- Needleman - Wunsch Algorithm
  - **global** alignment

- Smith - Waterman Algorithm
  - **local** alignment

Dynamic Programming

Define alignment matrix:



Dynamic Programming

Define alignment matrix:

	_	H	E	A	G	A	W	G	H	E	E
_	0										
P											
A											
W											
H											
E											
A											
E											

leave blank
1. row and
1. column

Start: 0

Calculate Alignment Matrix

		H	E	A	G	A	W	G	H	E	E
	0	-8	-16	-24	-32	-40	-48	-56	-64	-72	-80
P	-8										
A	-16										
W	-24										
H	-32										
E	-40										
A	-48										
E	-56										

Boundary conditions

$F(i, 0) = -i\,d$

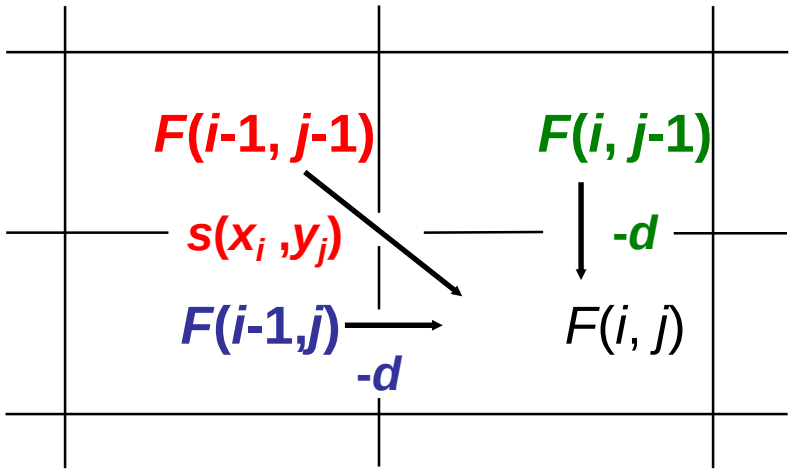
$F(j, 0) = -j\,d$

d=8 (linear gap costs)

The borders correspond to an alignment of a sequence only with gaps => adding up the gap costs at the edges

Calculate Alignment Matrix

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



Calculate Alignment 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								
A	-16	-10	-3								
W	-24										
H	-32										
E	-40										
A	-48										
E	-56										

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

P−H=−2

E−P=−1

H−A=−2

E−A=−1

$$F(1,1) = \max \begin{cases} F(0,0) + s(x_i, y_j) = 0 - 2 = -2 \\ F(0,1) - d = -8 - 8 = -16 \\ F(1,0) - d = -8 - 8 = -16 \end{cases} = -2$$

$$F(2,1) = \max \begin{cases} F(1,0) + s(x_i, y_j) = -8 - 1 = -9 \\ F(1,1) - d = -2 - 8 = -10 \\ F(2,0) - d = -16 - 8 = -24 \end{cases} = -9$$

$$F(1,2) = \max \begin{cases} -8 - 2 = -10 \\ -16 - 8 = -24 = \\ -2 - 8 = -10 \end{cases} -10$$

$$F(2,2) = \max \begin{cases} -2 - 1 = -3 \\ -10 - 8 = -18 \\ -9 - 8 = -17 \end{cases} = -3$$

Complete Alignment 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

Backtracking

End

		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

Start

Optimal global alignment:

```
HEAGAWGHE-E
--P-AW-HEAE
```

Dynamic Programming

- Needleman - Wunsch Algorithm
  - **global** alignment

- Smith - Waterman Algorithm
  - **local** alignment

Smith – Waterman (Lokal Alignment)

Two simple differences:

1. If the value you would set in the matrix is negative, write 0 instead

2. An alginment can start and end anywhere in the matrix (=> look for the largest number in the backtracking matrix as a starting point)

$$F(i, j) = \max \begin{cases} 0 \\ F(i, j) = F(i\text{-}1, j\text{-}1) + s(x_i, y_j) \\ F(i, j) = F(i\text{-}1, j) - d \\ F(i, j) = F(i, j\text{-}1) - d \end{cases}$$

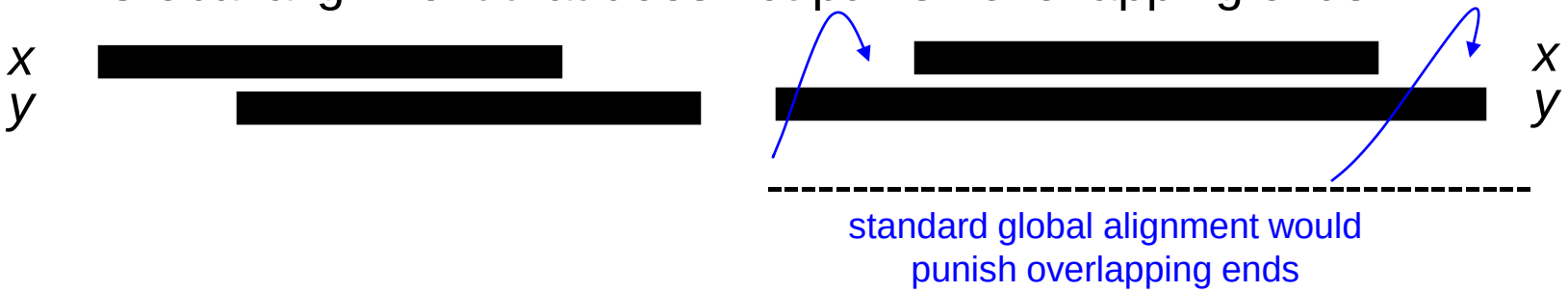
Smith-Waterman Alignment

		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	**5**	0	0	0	0	0
W	0	0	0	0	2	0	**20**	**12**	4	0	0
H	0	10	2	0	0	0	12	18	**22**	14	6
E	0	2	16	8	0	0	4	10	18	$\boxed{\mathbf{28}}$	20
A	0	0	8	21	13	5	0	4	10	20	27
E	0	0	6	13	18	12	4	0	4	16	26

Optimal local alignment: **AWGHE**
AW-HE

Overlap Matches

Global alignment that does not punish overlapping ends



Initialisierung

$$F(0,0) = 0,$$
$$F(i,0) = 0,\ 1 \le i \le n,$$
$$F(0,j) = 0,\ 1 \le j \le m,$$

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

Overlap Matches

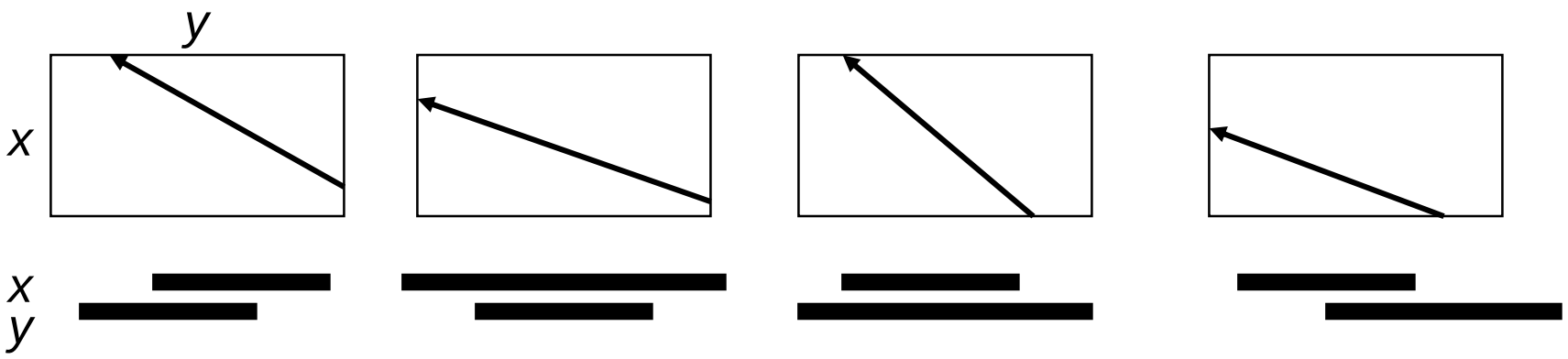
similar to global alignment, but different initialization and traceback

Traceback:

- *start* from maximum in *n-th* row and *m-th* column:

  max{F(*0,n*), F(*1,n*),….,F(*m,n*), F(*m,n-1*), F(*m,n-2*),…, F(*m,0*)}

- *end* when row 0 or column 0 is reached
  - row 0: end of sequence *x*
  - col 0: end of sequence *y*



Overlap Matches

Gap Penalty d = -8

		C	C	A	G	T	C	T
	0	0	0	0	0	0	0	0
A	0	-7	-7	2	-5	-7	-7	-7
G	0	-7	-14	-6	4	-4	-12	-14
C	0	2	-5	-13	-4	-1	-2	-10
C	0	2	4	-4	-12	-9	1	-7
A	0	-6	-4	6	-2	-10	-7	-6
T	0	-5	-11	-2	-1	0	-8	-5

f	A	G	T	C
A	2			
G	-5	2		
T	-7	-7	2	
C	-7	-7	-5	2

Substitution function

score of best
overlap alignment: 0

- - CCAGTCT
AGCCA -T - -

Alignments with Affine Gap Costs

1. Initialization: $\mathrm{F}(k,0) = I_x(0,k) = I_y(k,0) = F(0,k) = -d-(k-1)e;$

2. $$\mathrm{F}(i,j) = \max\begin{cases} F(i-1,j-1) + s(x_i,y_j) \\ I_x(i,j) \\ I_y(i,j) \end{cases}$$

3. $$I_x(i,j) = \max\begin{cases} F(i-1,j)-d \\ I_x(i-1,j)-e \end{cases}$$

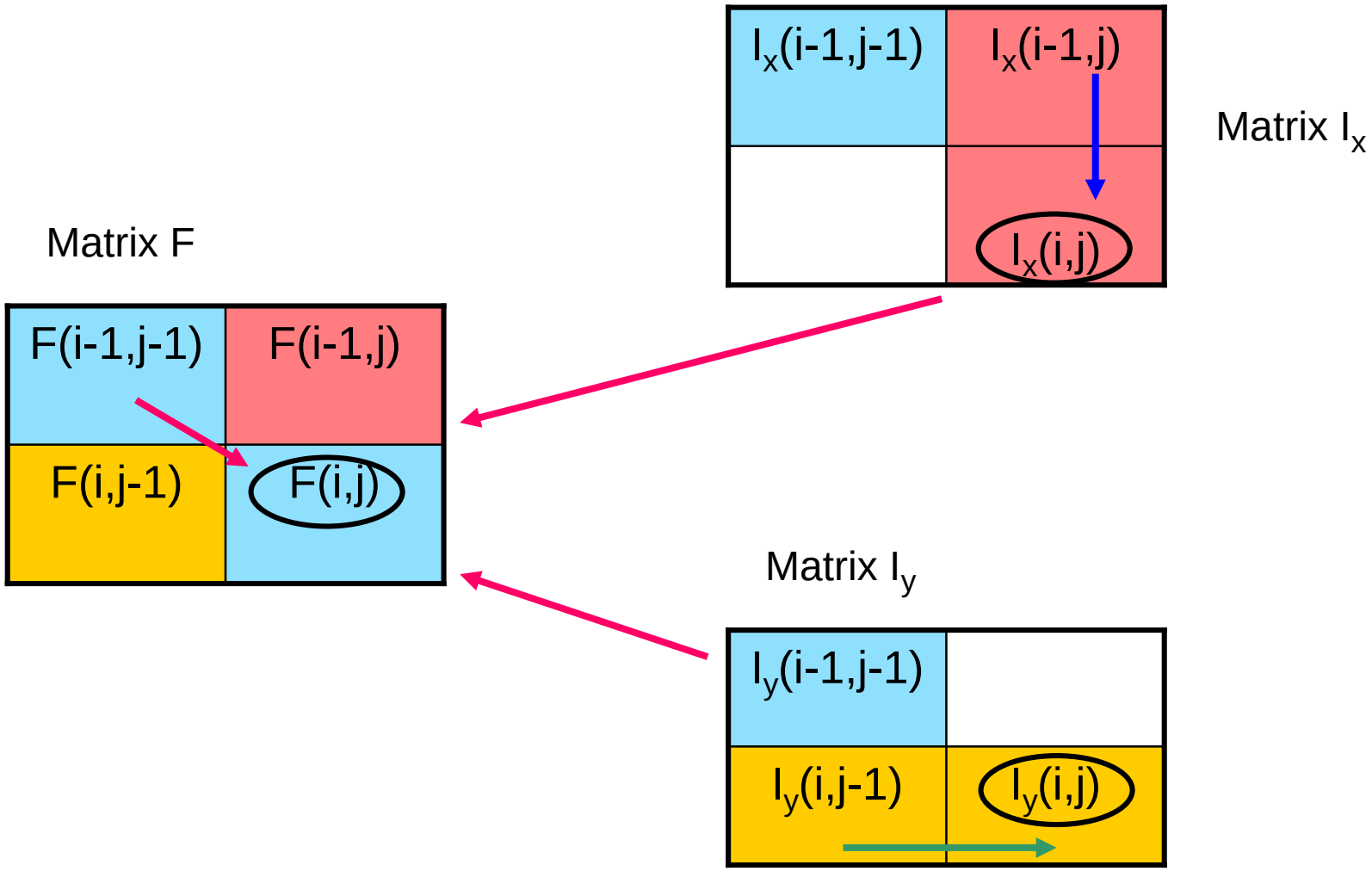
Start a new gap in *x*
Continue an existing gap in *x*

4. $$I_y(i,j) = \max\begin{cases} F(i,j-1)-d \\ I_y(i,j-1)-e \end{cases}$$

Start a new gap in *y*
Continue an existing gap in *y*

Algorithmus verwendet 3 Matrizen !

Alignments with Affine Gap Costs



Alignments with Affine Gap Costs

O. Gotoh: *An improved algorithm for matching biological sequences.* In: *J. Mol. Biol.*. 162, 1982, S. 705-708

	Δ	A	A	A	T	T
Δ	0	12	22	32	42	52
A	12	0	12	22	42	52
		1	*3*	*3*	*3*	*3*
A	22	12	0	12	32	42
		1	*1*	*3*	*3*	*2*
A	32	22	12	0	22	32
		1	*1*	*1*	*3*	*2*
G	42	42	32	22	10	32
		1	*1*	*5*	*1*	*3*
G	52	52	42	32	32	20
		1	*4*	*4*	*5*	*1*
T	62	62	52	42	32	32
		1	*4*	*4*	*1*	*1*
T	72	72	62	52	42	32
		1	*4*	*4*	*1*	*1*

(a)

	Δ	A	A	A	T	T
Δ	0	12	22	32	42	52
A	12	34	44	54	64	74
		1	*3*	*3*	*3*	*3*
A	22	22	34	44	64	74
		5	*1*	*3*	*3*	*2*
A	32	32	22	34	54	64
		5	*5*	*1*	*3*	*2*
G	42	42	32	22	44	54
		5	*5*	*5*	*1*	*3*
G	52	52	42	32	32	54
		5	*4*	*4*	*5*	*1*
T	62	62	52	42	42	44
		5	*4*	*4*	*1*	*1*
T	72	72	62	52	52	64
		5	*4*	*4*	*1*	*1*

(b)

FIG. 1. An example of operation of the algorithm. (a) $D_{m,n}$ (Arabic), and $e_{m,n}$ (Italic) obtained after the first run. (b) $Q_{m,n}$ (Arabic), and the completed $e_{m,n}$ (Italic). The underlined $e_{m,n}$ values are altered by the second run. The arrows indicate the paths of backtracking. To avoid going the wrong way, such as in the way shown by broken arrows, we always go straight ahead, if possible, at each branch point. The weight values used are $d(a_m, b_n) = 0$ if $a_m = b_n$, $d(a_m, b_n) = 10$ if $a_m \neq b_n$, and $w_k = 10k + 12$.

Alignments with Affine Gap Costs

s	A	G	T	C
A	2			
G	-5	2		
T	-7	-7	2	
C	-7	-7	-5	2

Substitution function s

d = 10

e = 2

F		A	A	A	G	T
	0	-10	-12	-14	-16	-18
A	-10	2	-8	-10	-12	-14
T	-12	-8	-5	-15	-17	-10

```
Alignment: AAAGT
           A---T
Score:     -10
```

Ix		A	A	A	G	T
	0	-10	-12	-14	-16	-18
A	-10	-12	-8	-10	-12	-14
T	-12	-14	-16	-15	-17	-19

Iy		A	A	A	G	T
	0	-10	-12	-14	-16	-18
A	-10	-12	-14	-16	-18	-20
T	-12	-8	-16	-18	-20	-22

Runtime Considerations

Complexity of dynamic programmig: O(nm)

Let protein database	→	100 million residues
Sequence length 1000	→	10^{11} matrix cells must be evaluated
Let 10 million matrix cells	→	1 sec
Full search	→	3 hours

Solution:

- Search the smallest fraction as possible of cells in the dynamic programming matrix.
- For very similar subsequences use of exact maching algorithms.

Heuristic alignment algorithms - **BLAST, FASTA**