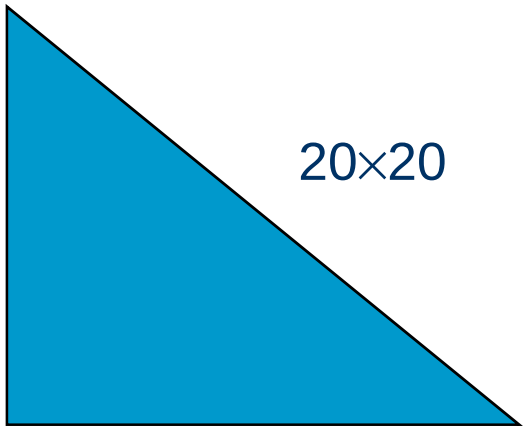

# 1-month Practical Course Genome Analysis (Integrative Bioinformatics & Genomics)

## Lecture 4: Pair-wise (2) and Multiple sequence alignment

Centre for Integrative Bioinformatics VU (IBIVU)
Vrije Universiteit Amsterdam
The Netherlands
ibivu.nl heringa@cs.vu.nl

# Alignment input parameters

## Scoring alignments




*Amino Acid Exchange Matrix*

A number of different schemes have been developed to compile residue exchange matrices

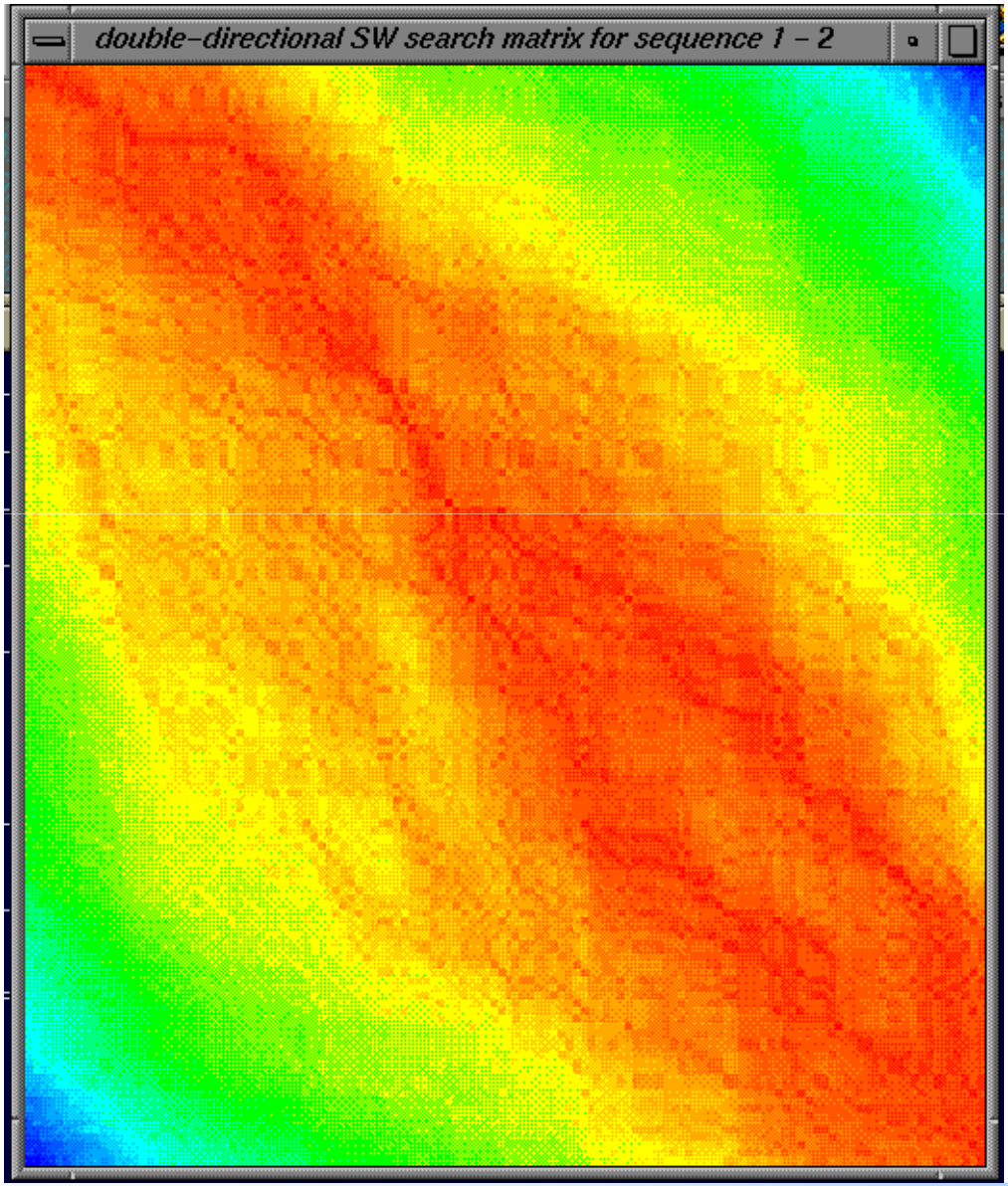
However, there are no formal concepts to calculate corresponding gap penalties

10 1

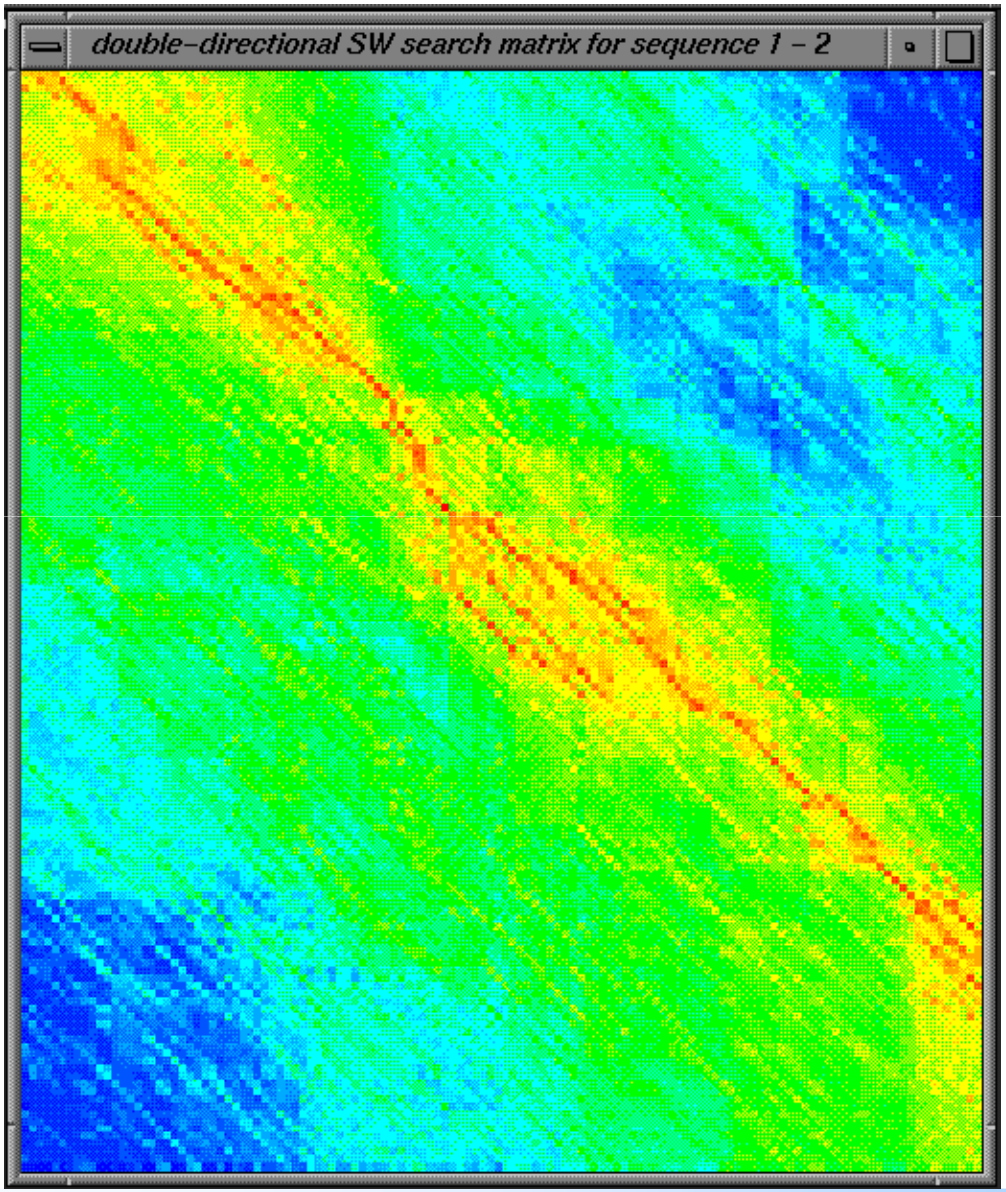
*Gap penalties (open, extension)*

Emperically determined values are recommended for PAM250, BLOSUM62, etc.

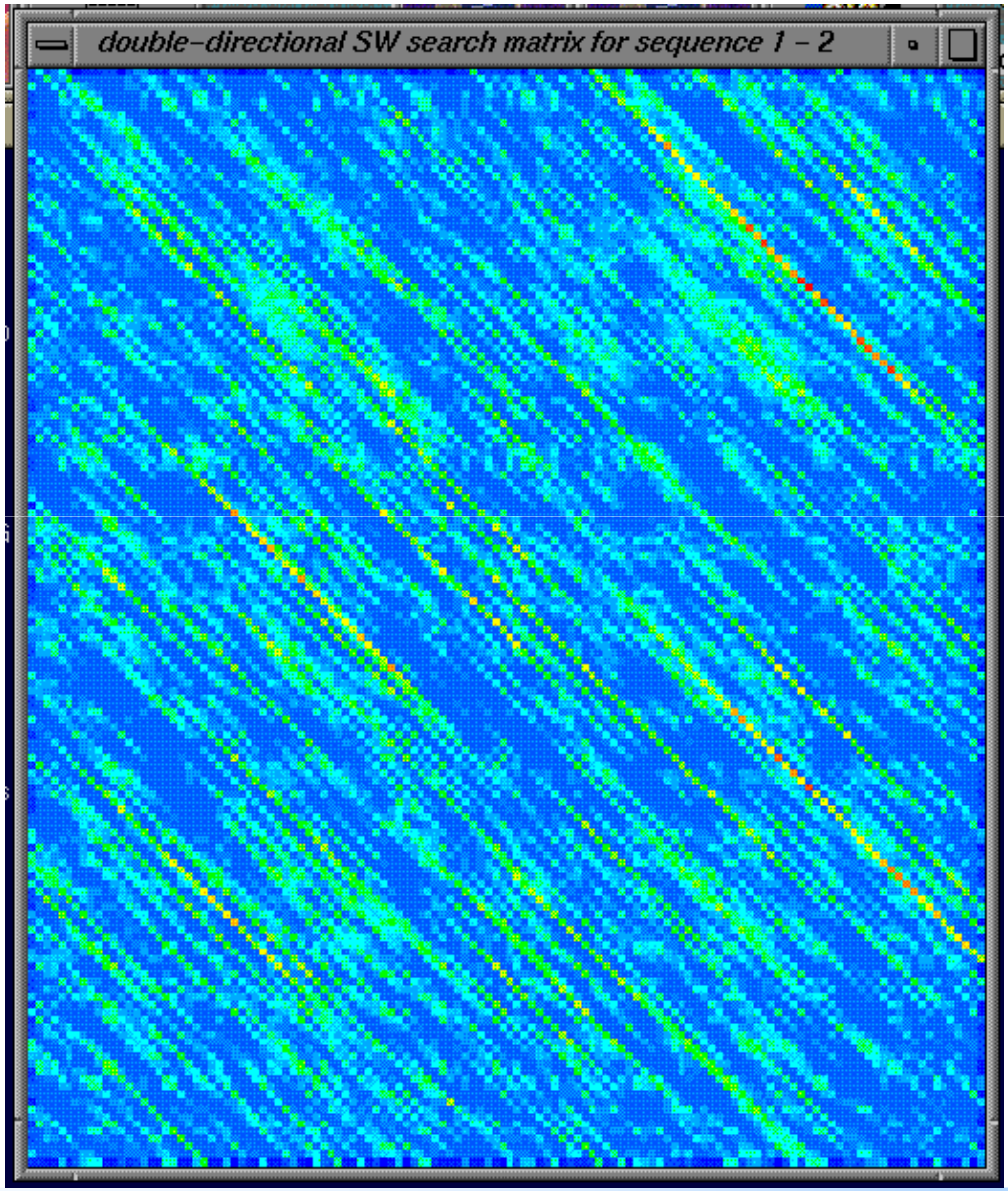
# M = BLOSUM62, $P_o$= 0, $P_e$= 0



# M = BLOSUM62, $P_o$= 12, $P_e$= 1



# M = BLOSUM62, $P_o$= 60, $P_e$= 5



# There are three kinds of alignments

- Global alignment (preceding slides)
- **Semi-global alignment**
- Local alignment

# Variation on global alignment

- *Global* alignment: previous algorithm is called *global* alignment because it uses all letters from both sequences.

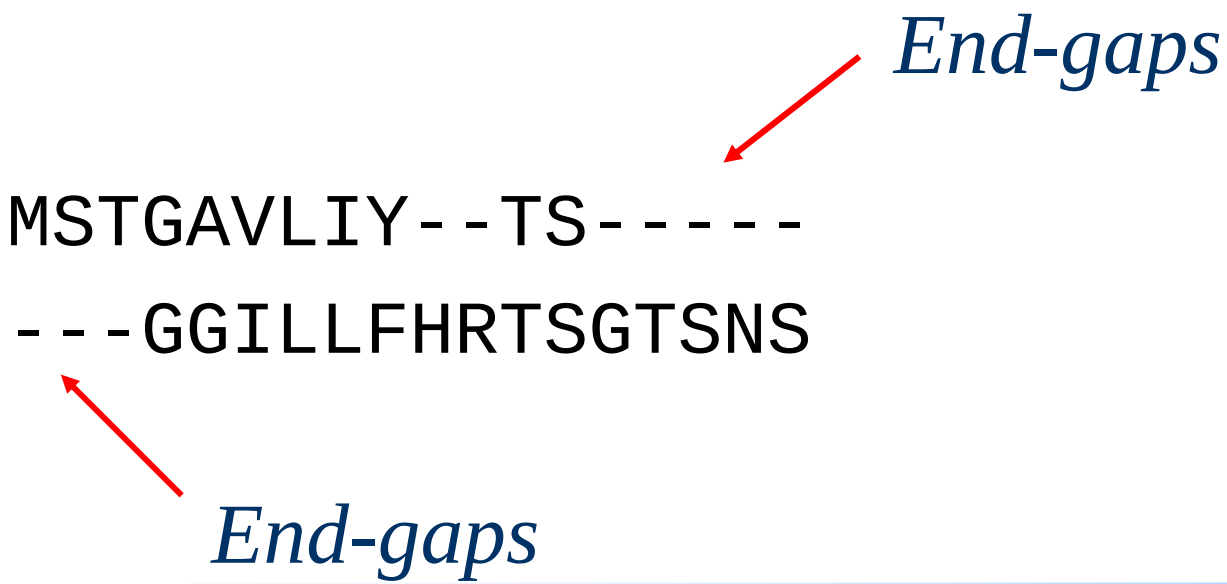
```
CAGCACTTGGATTCTCGG
CAGC-----G-T----GG
```

- *Semi-global* alignment: uses all letters but does not penalize for end gaps

```
CAGCA-CTTGGATTCTCGG
---CAGCGTGG---------
```

# Semi-global alignment

- Global alignment: all gaps are penalised
- Semi-global alignment: N- and C-terminal (5’ and 3’) gaps (end-gaps) are not penalised



# Semi-global alignment

Applications of *semi-global*:

– Finding a gene in genome

– Placing marker onto a chromosome

– One sequence much longer than the other

*Risk:* if gap penalties high -- really bad alignments for divergent sequences

Protein sequences have N- and C-terminal amino acids that are often small and hydrophilic

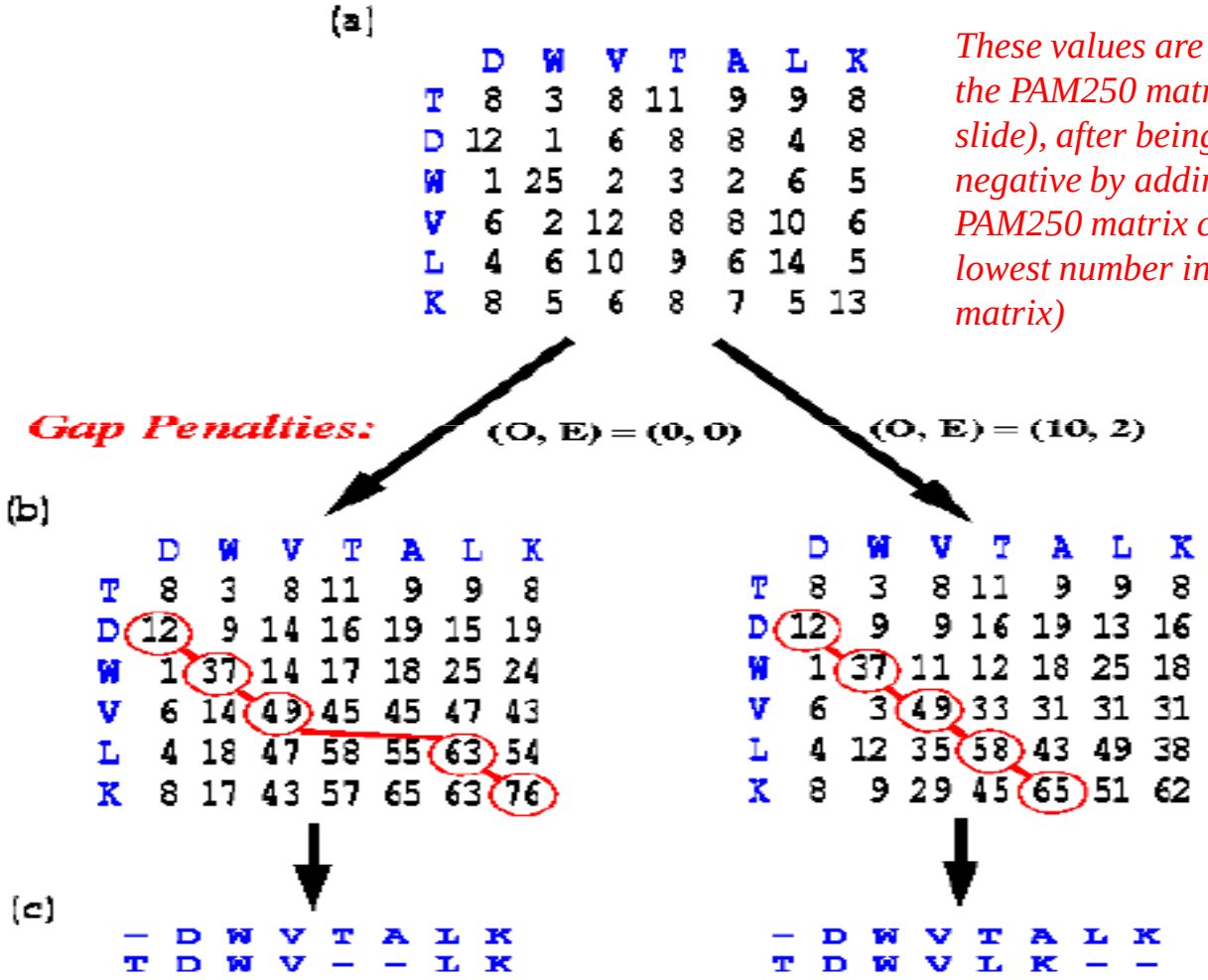
# Semi-global alignment

- Ignore 5’ or N-terminal end gaps
  - First row/column set to 0
- Ignore C-terminal or 3’ end gaps
  - Read the result from last row/column (select the highest scoring cell)

| | - | G | A | G | T | G |
|---|---|---|---|---|---|---|
| - | 0 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | -1 | 1 | -1 | -1 | -1 |
| G | 0 | 1 | -2 | 2 | 0 | -2 |
| T | 0 | -1 | 0 | 0 | 3 | 1 |

# Semi-global dynamic programming

## - two examples with different gap penalties -



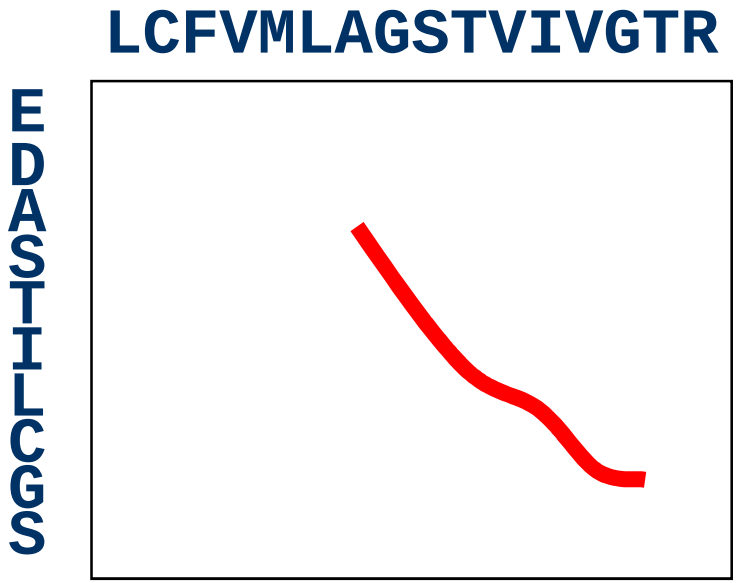

*These values are copied from the PAM250 matrix (see earlier slide), after being made non-negative by adding 8 to each PAM250 matrix cell (-8 is the lowest number in the PAM250 matrix)*

# There are three kinds of alignments

- Global alignment
- Semi-global alignment
- **Local alignment**

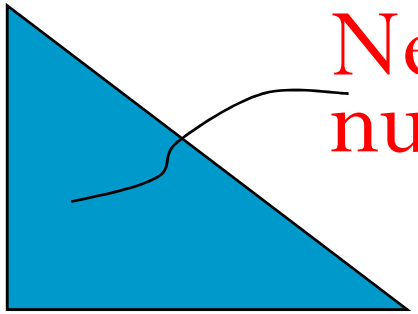
# Local dynamic programming

(Smith & Waterman, 1981)




*Search matrix*

**AGSTVIVG**
**A-STILCG**

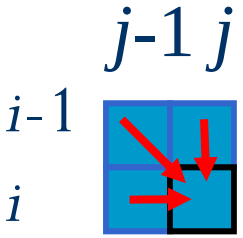



*Amino Acid Exchange Matrix*

*Gap penalties (open, extension)*

# Local dynamic programming (Smith and Waterman, 1981)

## basic algorithm




$$H(i,j) = Max \begin{cases} H(i\text{-}1,j\text{-}1) + S(i,j) & \text{diagonal} \\ H(i\text{-}1,j) - g & \text{vertical} \\ H(i,j\text{-}1) - g & \text{horizontal} \\ \mathbf{0} \end{cases}$$

# Example: *local* alignment of two sequences

- Align two DNA sequences:
  - GAGTGA
  - GAGGCGA (note the length difference)
- Parameters of the algorithm:
  - Match: score(A,A) = 1
  - Mismatch: score(A,T) = -1
  - Gap: g = -2

$$M[i, j] = \max \begin{cases} M[i\text{-}1, j\text{-}1] \pm 1 \\ M[i, j\text{-}1] - 2 \\ M[i\text{-}1, j] - 2 \\ 0 \end{cases}$$

# The algorithm. Step 1: init

- Create the matrix
- Initiation
  - No beginning row/column
  - Just apply the equation…

$$M[i, j] = \max \begin{cases} M[i-1, j-1] \pm 1 \\ M[i, j-1] - 2 \\ M[i-1, j] - 2 \\ 0 \end{cases}$$

| | j→ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| i↓ | | G | A | G | T | G | A |
| 1 | G | | | | | | |
| 2 | A | | | | | | |
| 3 | G | | | | | | |
| 4 | G | | | | | | |
| 5 | C | | | | | | |
| 6 | G | | | | | | |
| 7 | A | | | | | | |

# The algorithm. Step 2: fill in

- Perform the forward step…

$$M[i, j] = \max \begin{cases} M[i\text{-}1, j\text{-}1] \pm 1 \\ M[i, j\text{-}1] - 2 \\ M[i\text{-}1, j] - 2 \\ 0 \end{cases}$$

| | j→ | *1* | *2* | *3* | *4* | *5* | *6* |
|---|---|---|---|---|---|---|---|
| i↓ | | G | A | G | T | G | A |
| *1* | G | 1 | 0 | 1 | 0 | 1 | 0 |
| *2* | A | 0 | 2 | 0 | 0 | **0** | 2 |
| *3* | G | 1 | 0 | 3 | 1 | 1 | 0 |
| *4* | G | 1 | 0 | 1 | 2 | | |
| *5* | C | | | | | | |
| *6* | G | | | | | | |
| *7* | A | | | | | | |

# The algorithm. Step 2: fill in

- Perform the forward step…

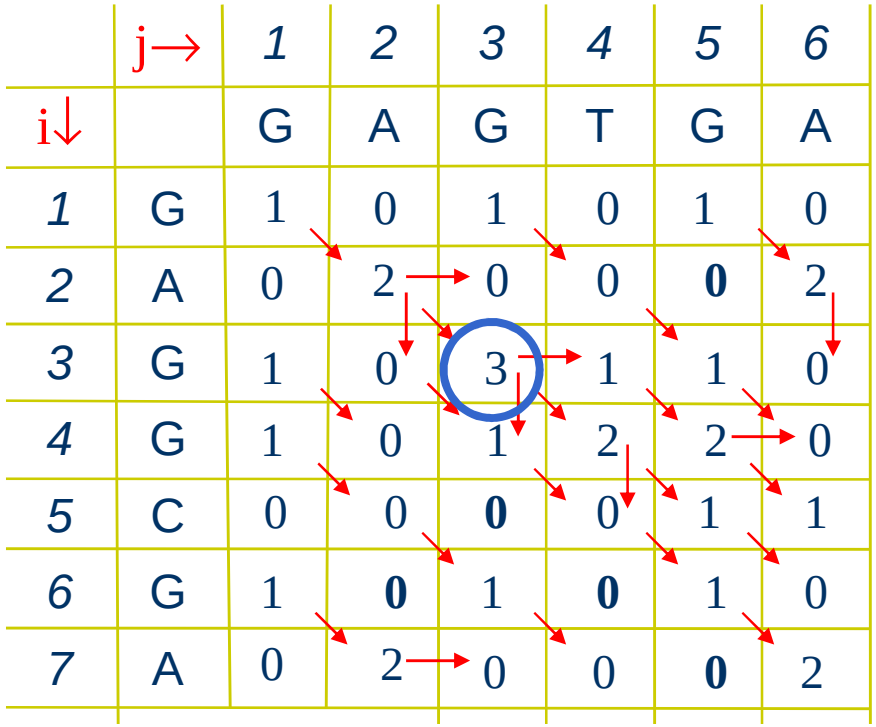
$$M[i, j] = \max \begin{cases} M[i\text{-}1, j\text{-}1] \pm 1 \\ M[i, j\text{-}1] - 2 \\ M[i\text{-}1, j] - 2 \\ 0 \end{cases}$$

| | j→ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| i↓ | | G | A | G | T | G | A |
| 1 | G | 1 | 0 | 1 | 0 | 1 | 0 |
| 2 | A | 0 | 2 | 0 | 0 | **0** | 2 |
| 3 | G | 1 | 0 | 3 | 1 | 1 | 0 |
| 4 | G | 1 | 0 | 1 | 2 | 2 | 0 |
| 5 | C | 0 | 0 | **0** | 0 | 1 | 1 |
| 6 | G | 1 | **0** | 1 | **0** | 1 | 0 |
| 7 | A | 0 | 2 | 0 | 0 | **0** | 2 |

# The algorithm. Step 2: fill in

$$M[i, j] = \max \begin{cases} M[i\text{-}1, j\text{-}1] \pm 1 \\ M[i, j\text{-}1] - 2 \\ M[i\text{-}1, j] - 2 \\ 0 \end{cases}$$

- We're done
- Find the highest cell anywhere in the matrix



| | j→ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| i↓ | | G | A | G | T | G | A |
| 1 | G | 1 | 0 | 1 | 0 | 1 | 0 |
| 2 | A | 0 | 2 | 0 | 0 | **0** | 2 |
| 3 | G | 1 | 0 | 3 | 1 | 1 | 0 |
| 4 | G | 1 | 0 | 1 | 2 | 2 | 0 |
| 5 | C | 0 | 0 | **0** | 0 | 1 | 1 |
| 6 | G | 1 | **0** | 1 | **0** | 1 | 0 |
| 7 | A | 0 | 2 | 0 | 0 | **0** | 2 |

# The algorithm. Step 3: trace back

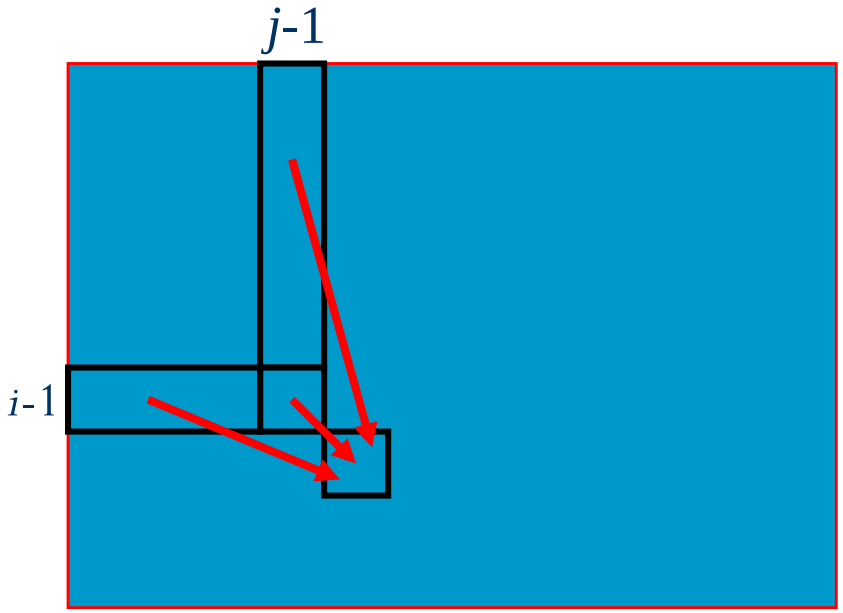
- Reconstruct path leading to highest scoring cell
- Trace back until zero or start of sequence: alignment path can begin and terminate anywhere in matrix
- Alignment: GAG
  GAG

$$M[i, j] = \max \begin{cases} M[i\text{-}1, j\text{-}1] \pm 1 \\ M[i, j\text{-}1] - 2 \\ M[i\text{-}1, j] - 2 \\ 0 \end{cases}$$

| | j→ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| i↓ | | G | A | G | T | G | A |
| 1 | G | 1 | 0 | 1 | 0 | 1 | 0 |
| 2 | A | 0 | 2 | 0 | 0 | **0** | 2 |
| 3 | G | 1 | 0 | 3 | 1 | 1 | 0 |
| 4 | G | 1 | 0 | 1 | 2 | 2 | 0 |
| 5 | C | 0 | 0 | **0** | 0 | 1 | 1 |
| 6 | G | 1 | **0** | 1 | **0** | 1 | 0 |
| 7 | A | 0 | 2 | 0 | 0 | **0** | 2 |

# Local dynamic programming

## (Smith & Waterman, 1981; Gotoh, 1984)




This is the general DP algorithm, which is suitable for linear, affine and concave penalties, although for the example here affine penalties are used

*Gap opening penalty*

$$S_{i,j} = Max \begin{cases} S_{i,j} + Max\{S_{0<x<i-1,j-1} - Pi - (i-x-1)Px\} \\ S_{i,j} + S_{i-1,j-1} \\ S_{i,j} + Max\{S_{i-1,0<y<j-1} - Pi - (j-y-1)Px\} \\ \mathbf{0} \end{cases}$$

*Gap extension penalty*

# Measuring Similarity

- Sequence identity (number of identical exchanges per unit length)
- Raw alignment score
- Sequence similarity (alignment score normalised to a maximum possible)
- Alignment score normalised to a randomly expected situation (database/homology searching)

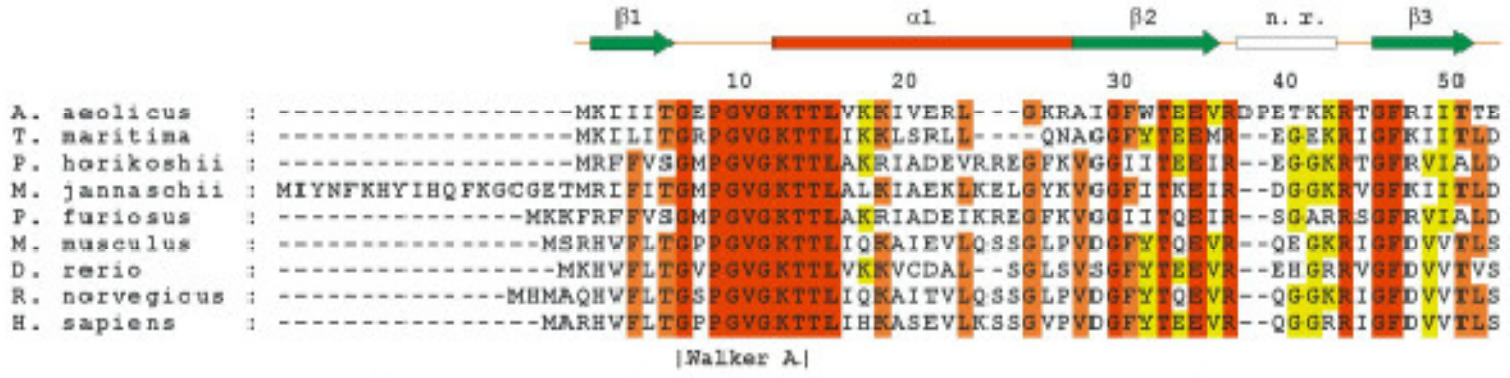
# Pairwise alignment

- Now we know how to do it:
- How do we get a multiple alignment (three or more sequences)?

- Multiple alignment: much greater combinatorial explosion than with pairwise alignment…..

# Multiple alignment idea

- Take three or more related sequences and align them such that the greatest number of similar characters are aligned in the same column of the alignment.

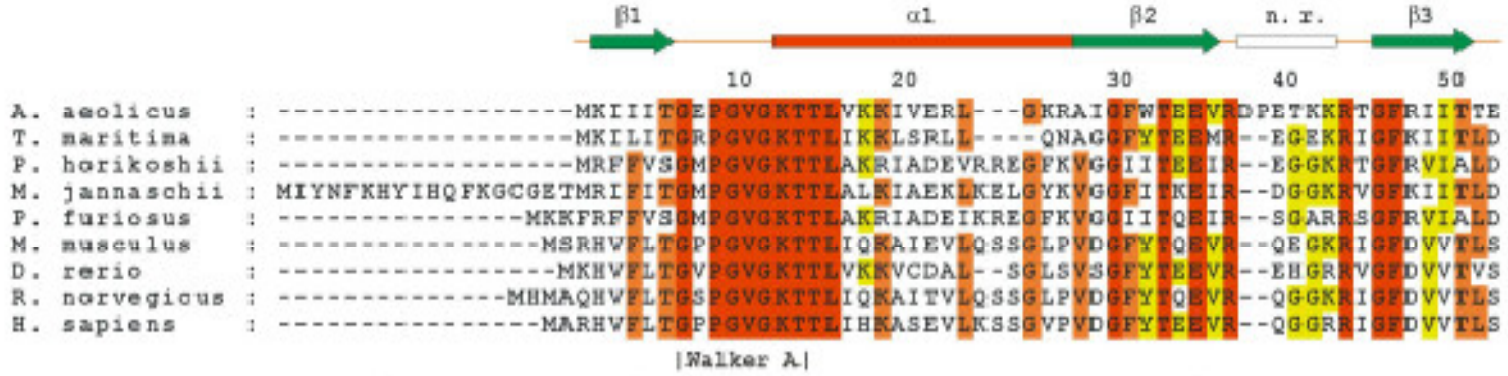



*Ideally, the sequences are orthologous, but often include paralogues.*

# Scoring a multiple alignment

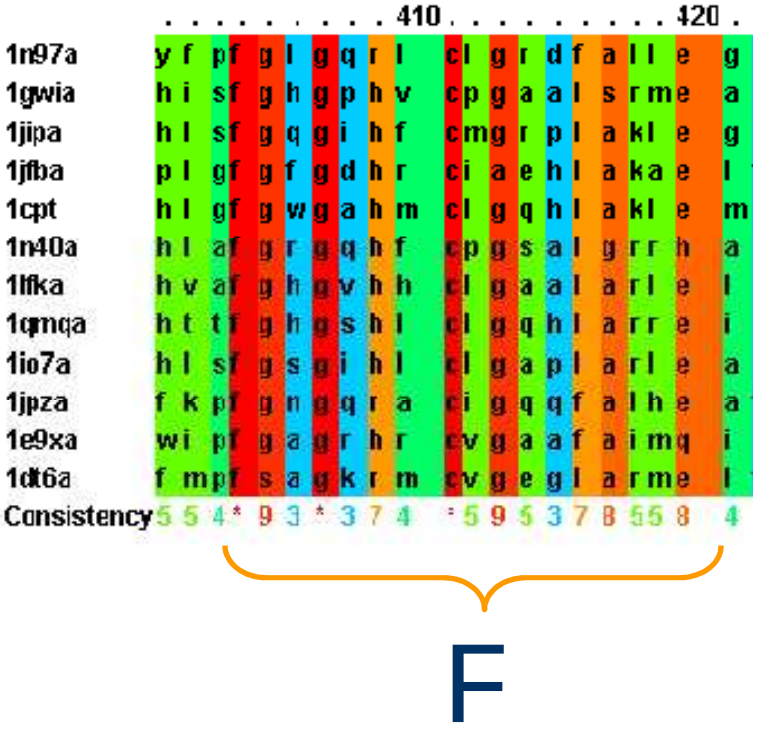
- You can score a multiple alignment by taking all the pairs of aligned sequences and adding up the pairwise scores:

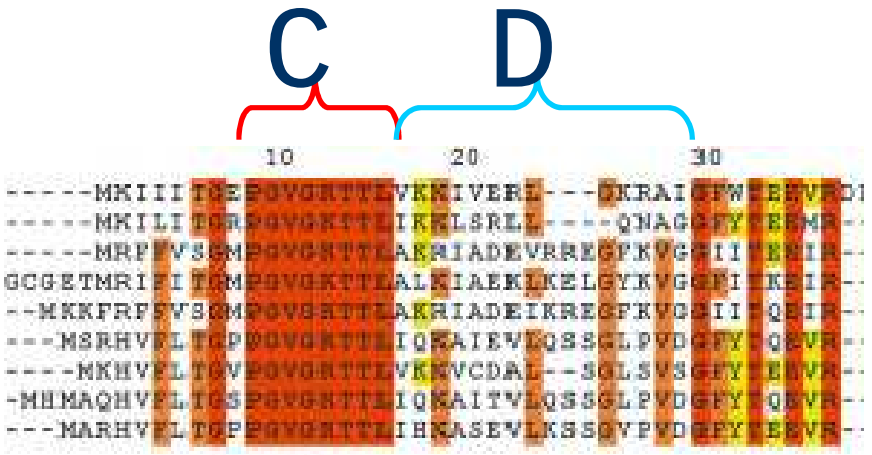
$$S_{a,b} = \sum_l s(a_i, b_j) - \sum_k N_k \bullet gp(k)$$




- This is referred to as the Sum-of-Pairs score

# Information content of a multiple alignment



# What to ask yourself

- What program to use to get a multiple alignment?
  (three or more sequences)

- What is our aim?
  - Do we go for max accuracy?
  - Least computational time?
  - Or the best compromise?

- What do we want to achieve each time?

# Multiple alignment methods

- **Multi-dimensional dynamic programming**
  > extension of pairwise sequence alignment.

- Progressive alignment
  > incorporates phylogenetic information to guide the alignment process

- Iterative alignment
  > correct for problems with progressive alignment by repeatedly realigning subgroups of sequence

# Exhaustive & Heuristic algorithms

- Exhaustive approaches
  - Examine all possible aligned positions simultaneously
  - Look for the optimal solution by (multi-dimensional) DP
  - Very (very) slow
- Heuristic approaches
  - Strategy to find a near-optimal solution (by using rules of thumb)
  - Shortcuts are taken by reducing the search space according to certain criteria
  - Much faster

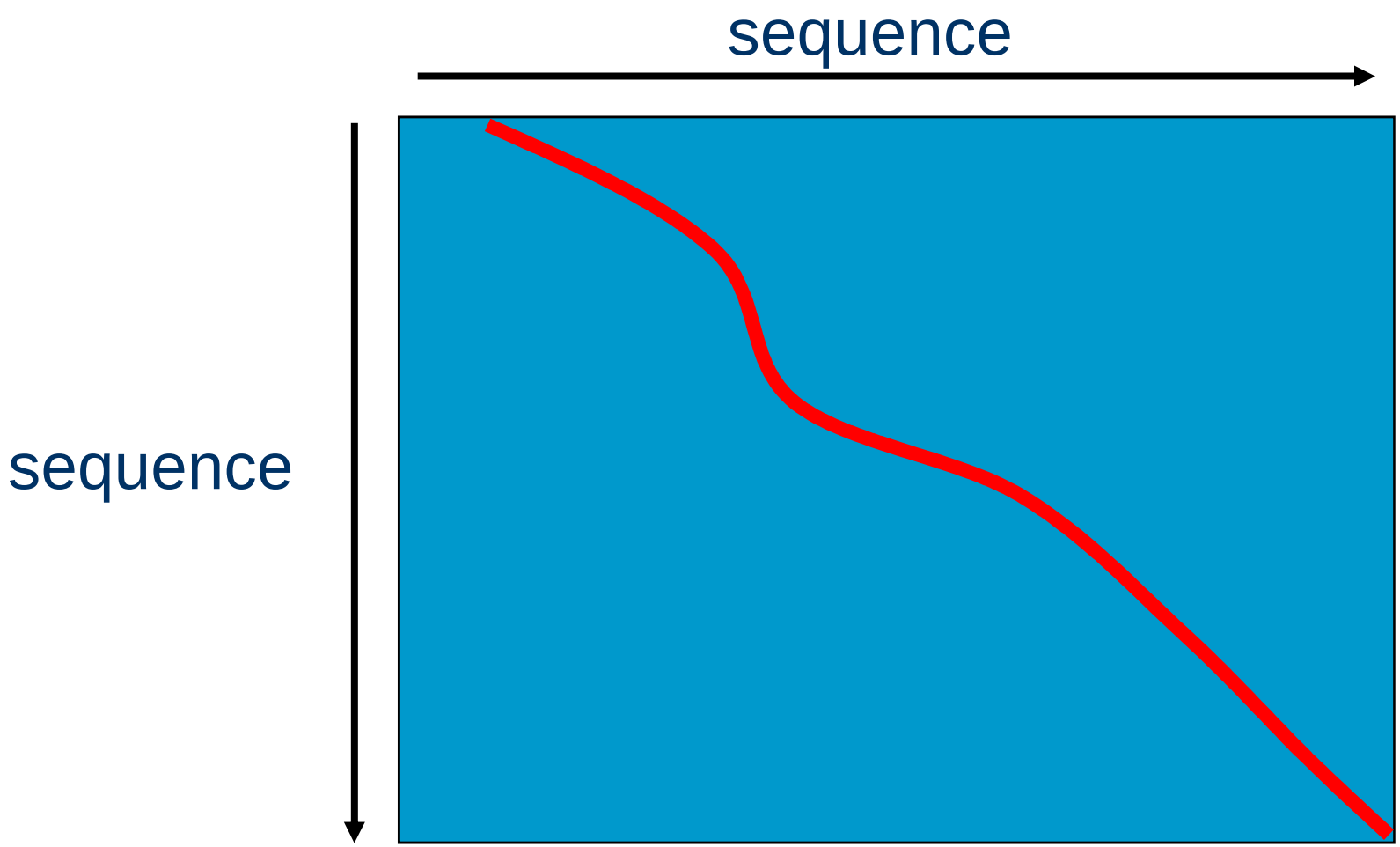
# Simultaneous multiple alignment

## Multi-dimensional dynamic programming

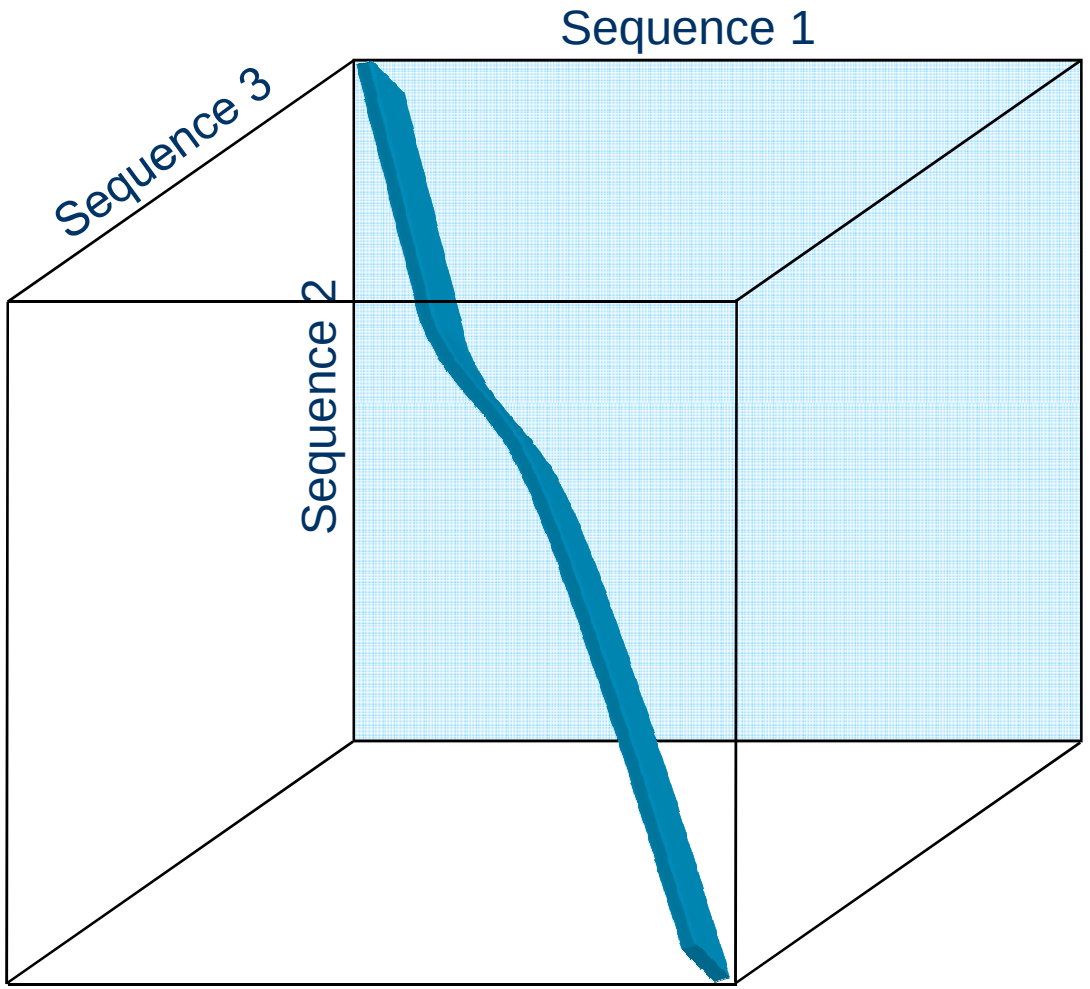
*Combinatorial explosion*

- DP using two sequences of length n
  - $n^2$ comparisons
- Number of comparisons increases exponentially
  - i.e. $n^N$ where n is the length of the sequences, and N is the number of sequences
- Impractical even for small numbers of short sequences

# Sequence-sequence alignment by Dynamic Programming



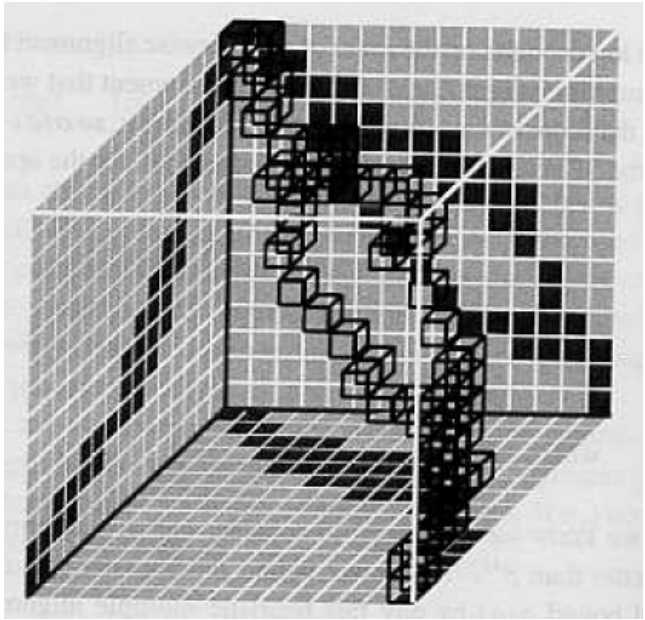
# Multi-dimensional dynamic programming (Murata et al., 1985)



# The MSA approach

Lipman et al. 1989

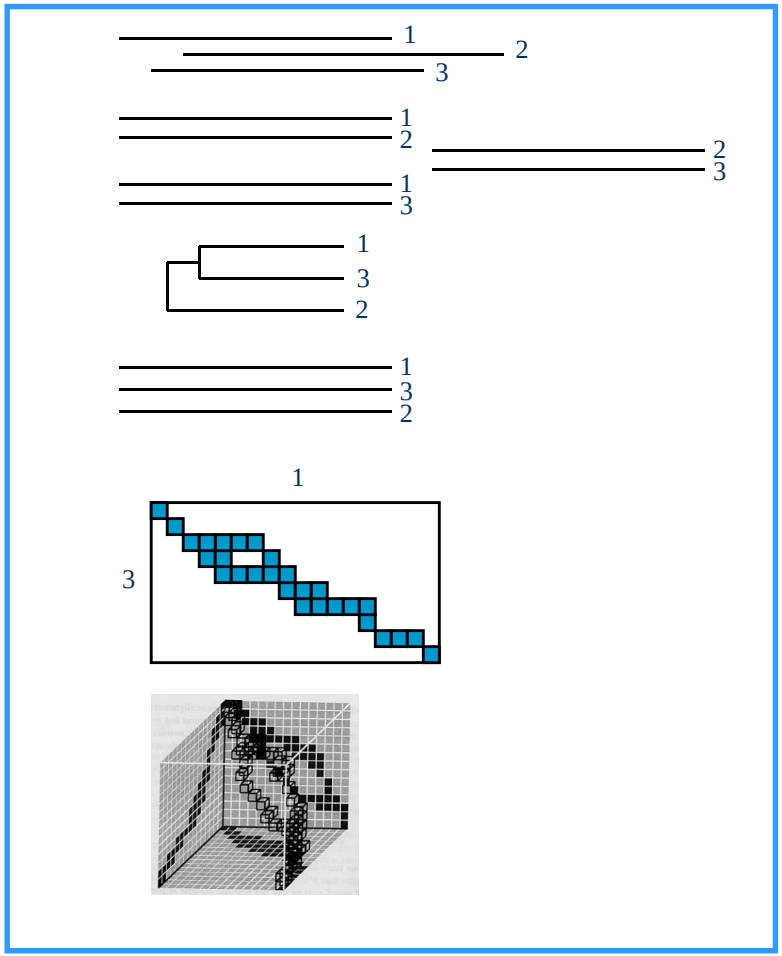
- Key idea: restrict the computational costs by determining a minimal region within the n-dimensional matrix that contains the optimal path



# The MSA method *in detail*

1. Let's consider 3 sequences
2. Calculate all pair-wise alignment scores by Dynamic programming
3. Use the scores to predict a tree
4. Produce a heuristic multiple align. based on the tree (quick & dirty)
5. Calculate maximum cost for each sequence pair from multiple alignment (upper bound) & determine paths with < costs.
6. Determine spatial positions that must be calculated to obtain the optimal alignment (intersecting areas or 'hypersausage' around matrix diagonal)
7. Perform multi-dimensional DP

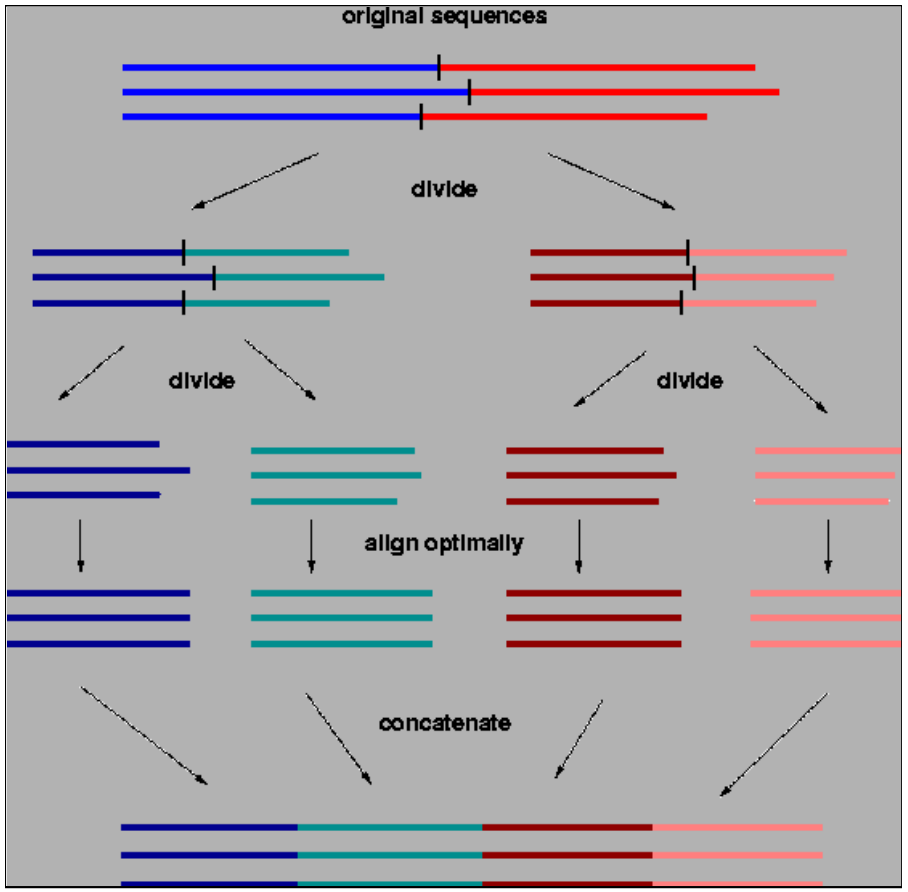
Note *Redundancy caused by highly correlated sequences is avoided*



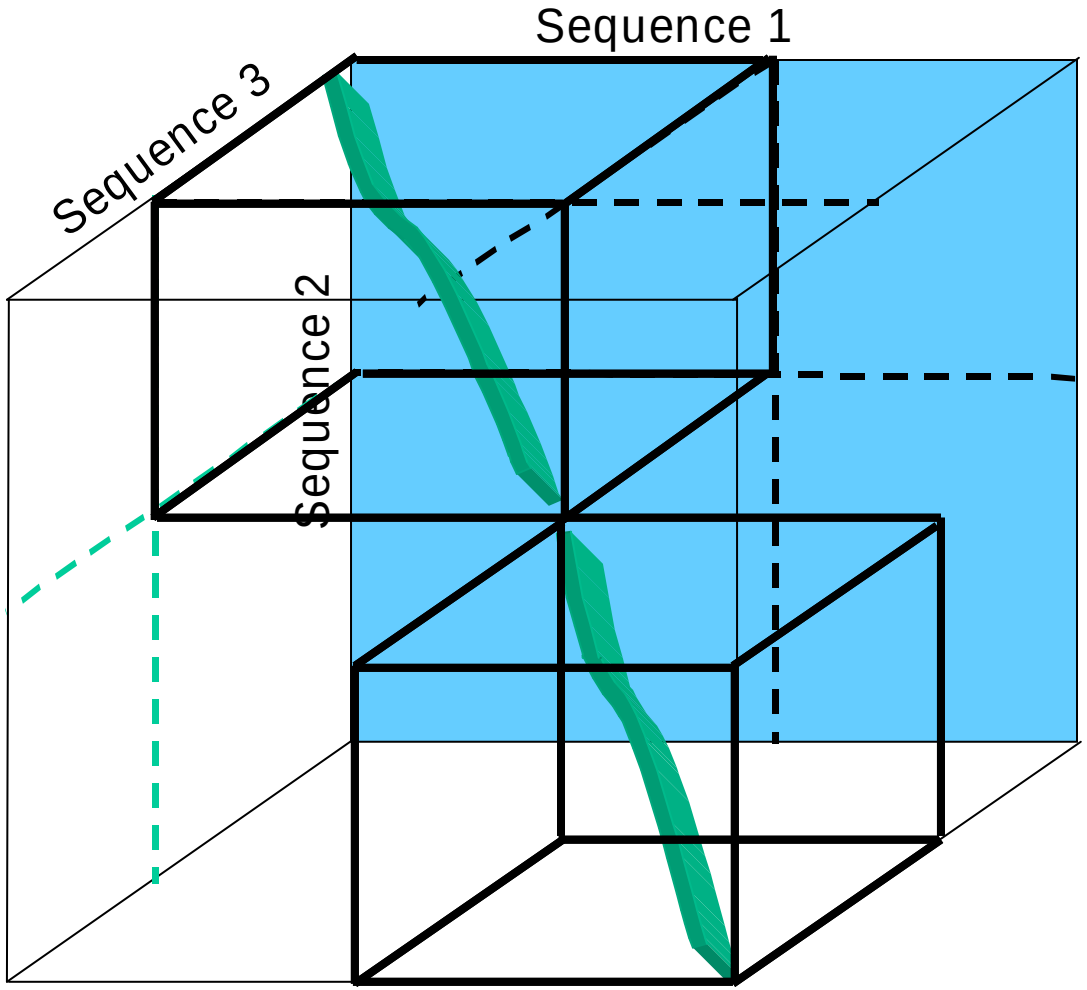
# The DCA (Divide-and-Conquer) approach

## Stoye et al. 1997

- Each sequence is cut in two behind a suitable cut position somewhere close to its midpoint.
- This way, the problem of aligning one family of (long) sequences is divided into the two problems of aligning two families of (shorter) sequences.
- This procedure is re-iterated until the sequences are sufficiently short.
- Optimal alignment by MSA.
- Finally, the resulting short alignments are concatenated.



# So in effect …



# Multiple alignment methods

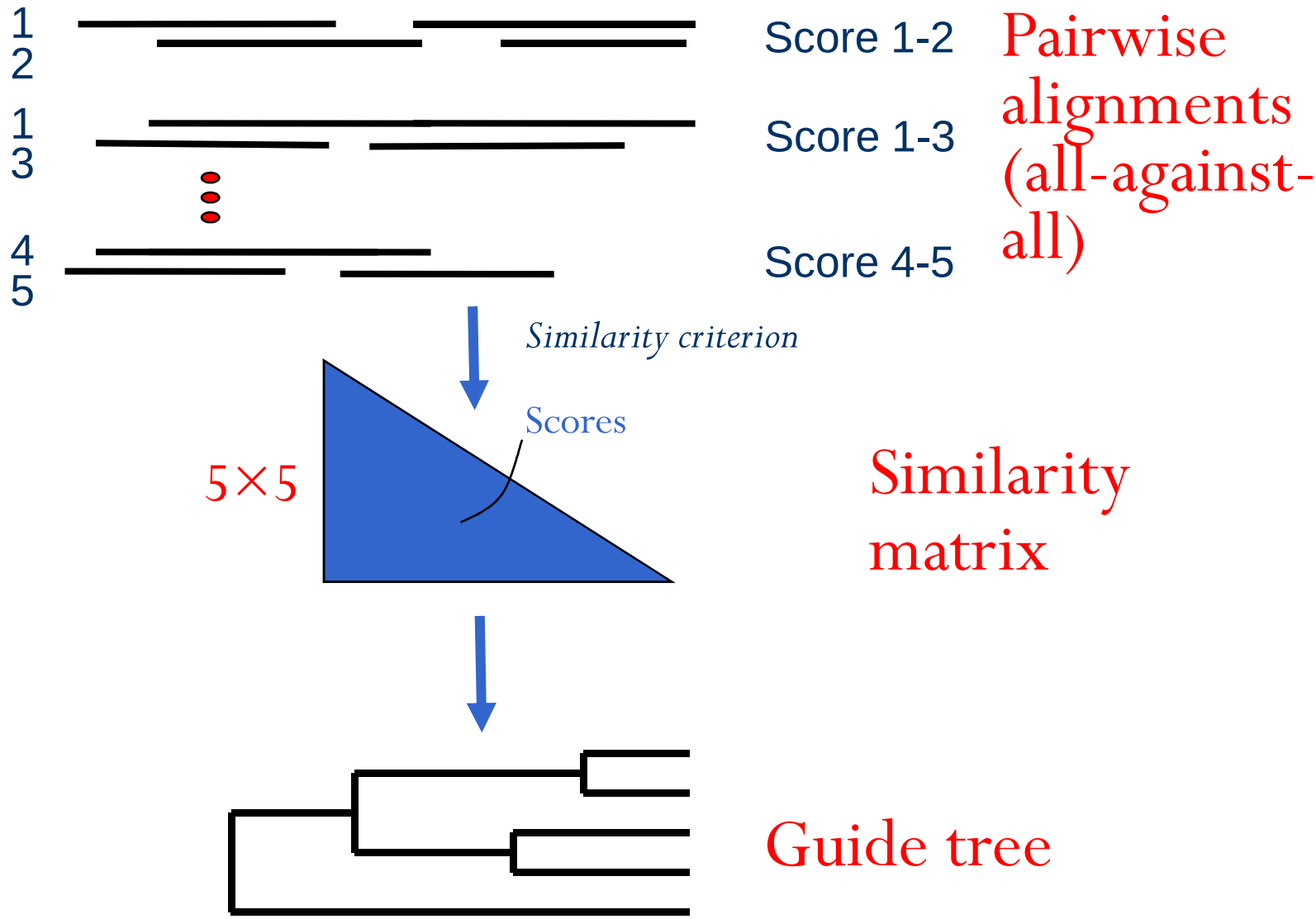
- Progressive alignment
  > incorporates phylogenetic information to guide the alignment process

- Iterative alignment
  > correct for problems with progressive alignment by repeatedly realigning subgroups of sequence

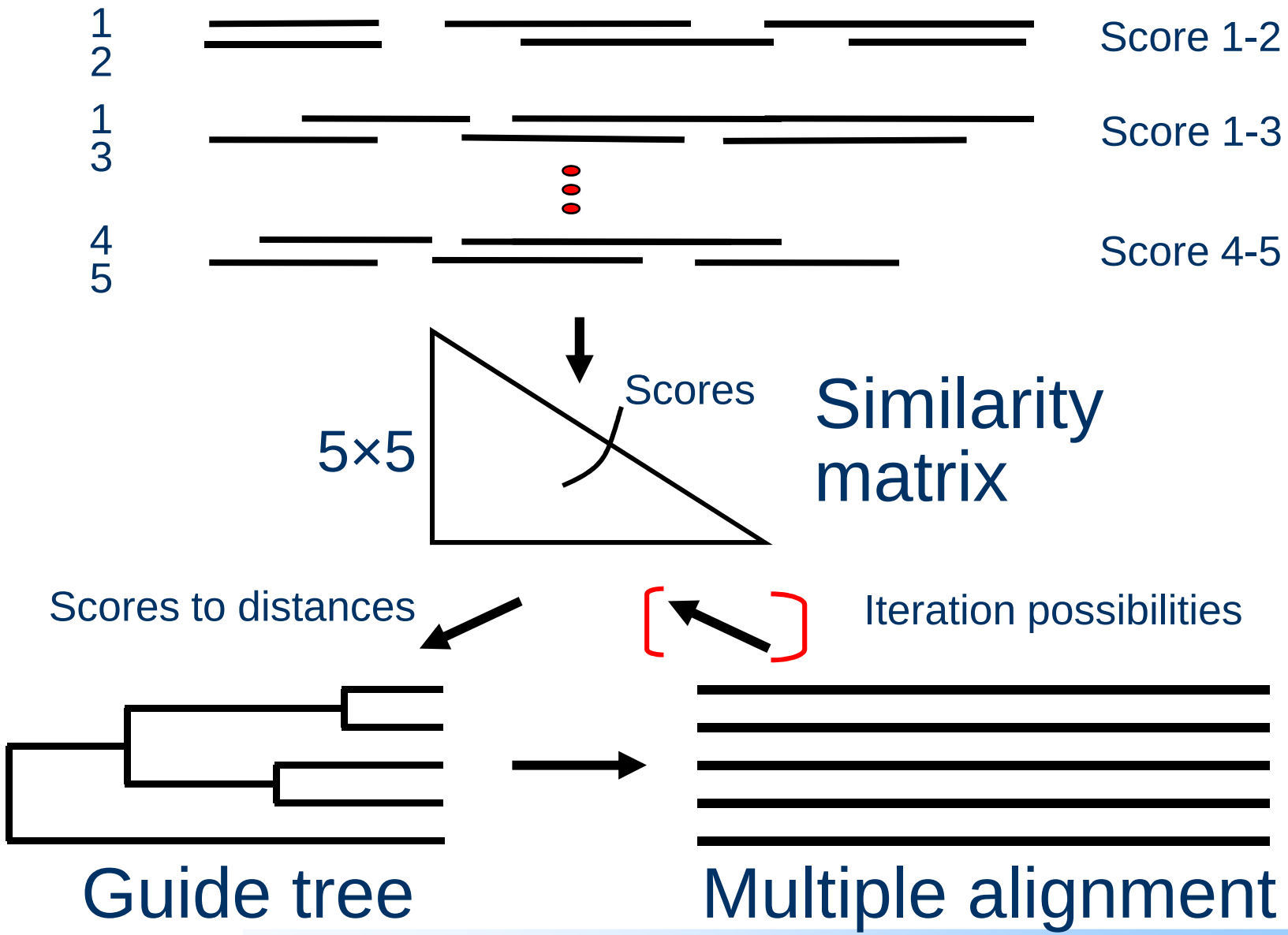
# The progressive alignment method

- Underlying idea: we are interested in aligning families of sequences that are evolutionary related
- Principle: construct an approximate phylogenetic tree for the sequences to be aligned (‘guide tree’) and then build up the alignment by progressively adding sequences in the order specified by the tree.

# Making a guide tree



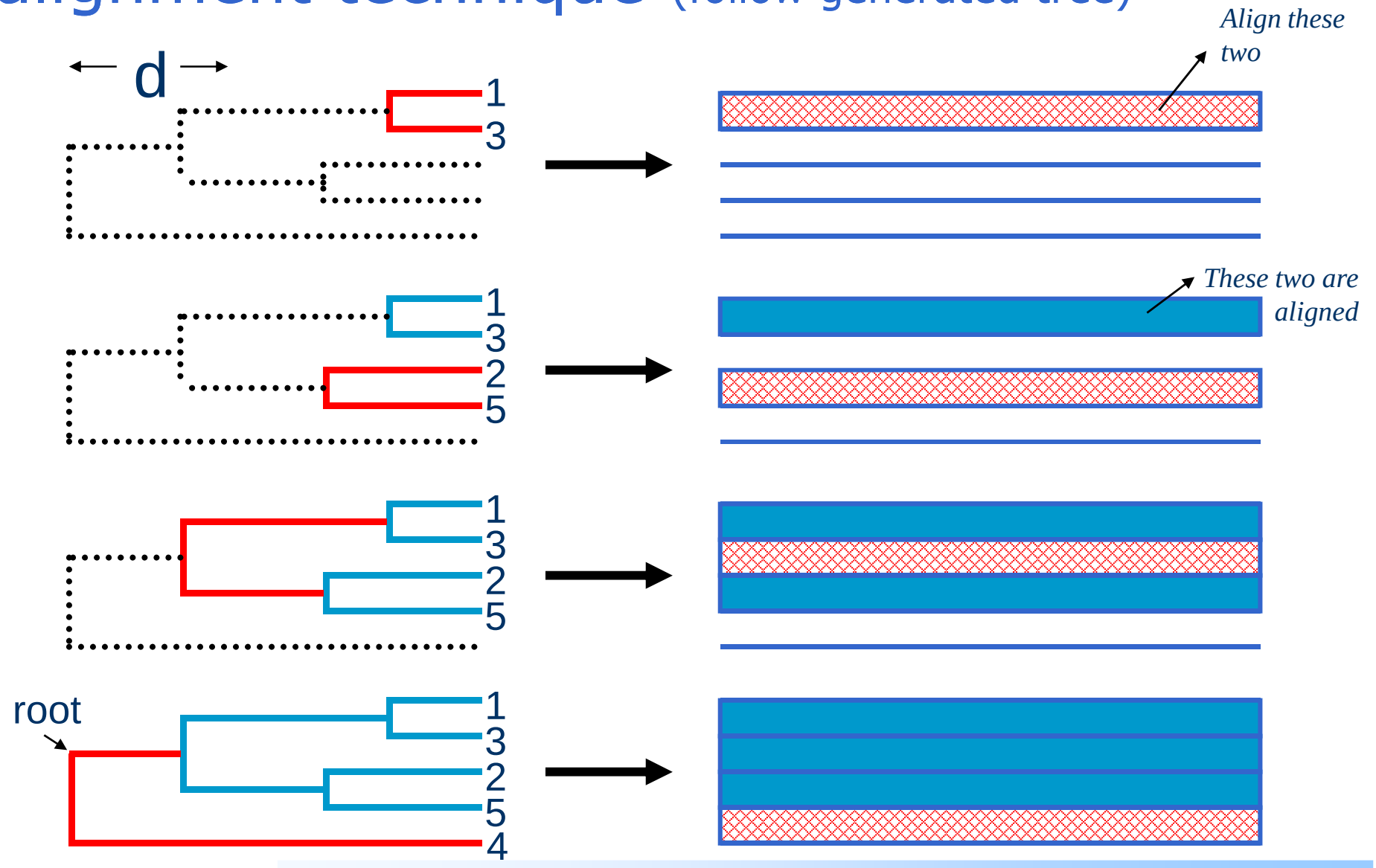
# Progressive multiple alignment



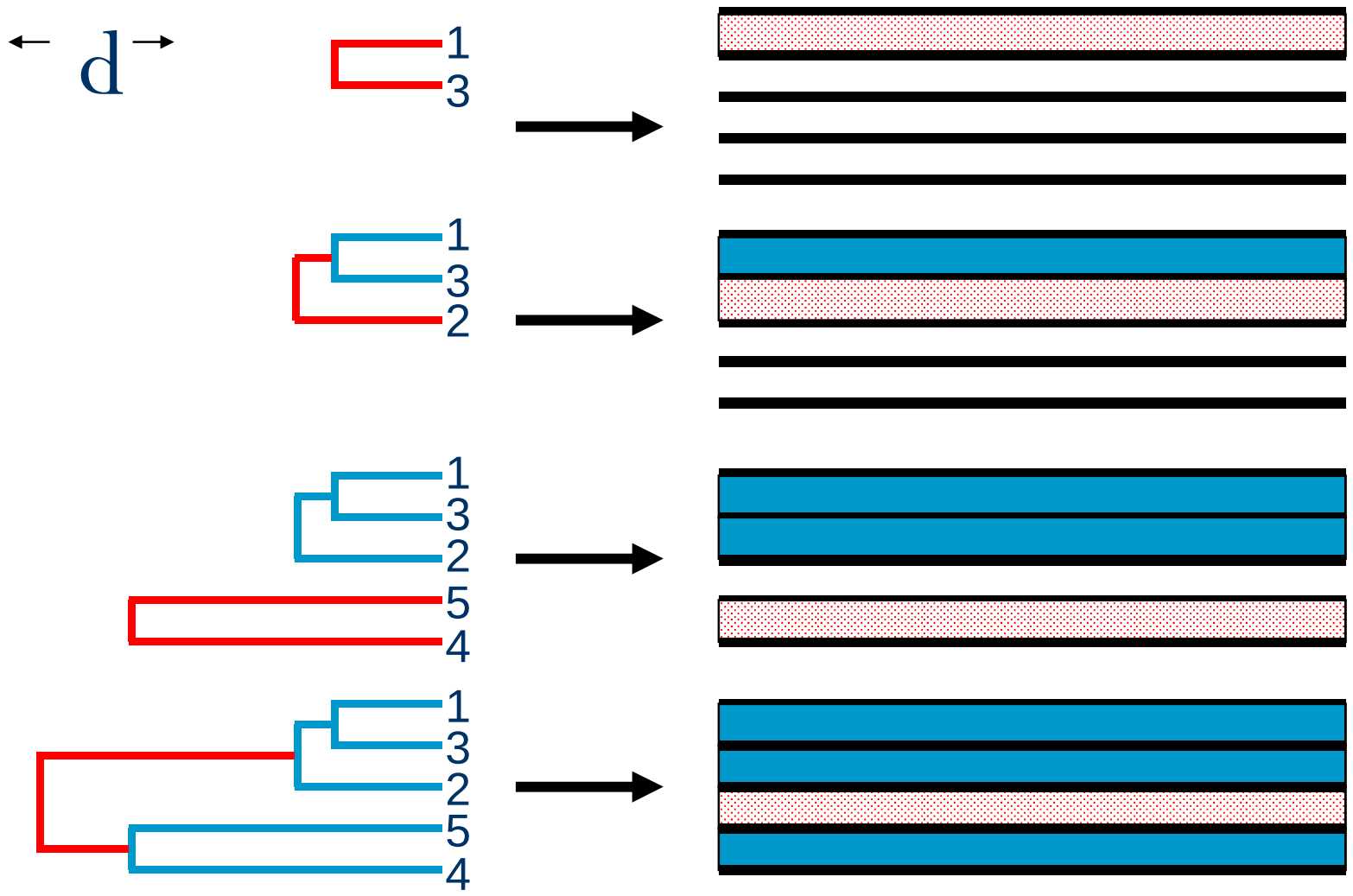
# Progressive alignment strategy

1. Perform pair-wise alignments of all of the sequences (all against all; e.g. make N(N-1)/2 alignments)
   - Most methods use semi-global alignment
2. Use the alignment scores to make a similarity (or distance) matrix
3. Use that matrix to produce a guide tree
4. Align the sequences successively, guided by the order and relationships indicated by the tree (N-1 alignment steps)

# General progressive multiple alignment technique (follow generated tree)

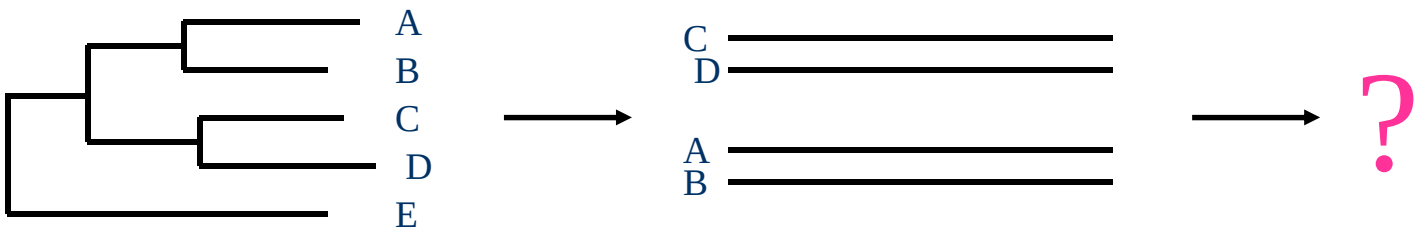


# PRALINE progressive strategy




*At each step, Praline checks which of the pair-wise alignments (sequence-sequence, sequence-profile, profile-profile) has the highest score – this one gets selected*

# But how can we align blocks of sequences ?




- The dynamic programming algorithm performs well for pairwise alignment (two axes).
- So we should try to treat the blocks as a “single” sequence …

# How to represent a block of sequences

- **Historically**: **consensus sequence**
  *single sequence that best represents the amino acids observed at each alignment position.*

- **Modern methods**: **alignment profile**
  *representation that retains the information about frequencies of amino acids observed at each alignment position.*

# Consensus sequence

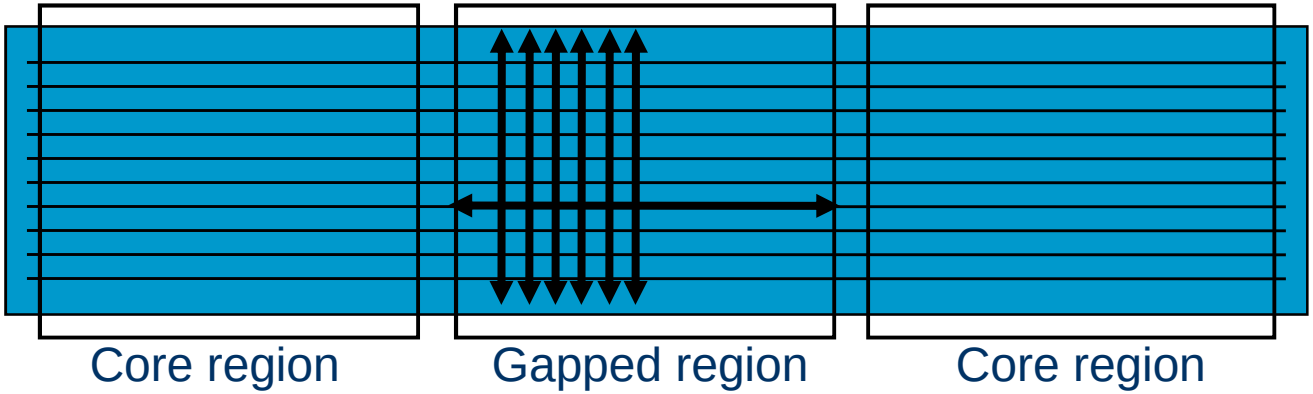
| | |
|---|---|
| Sequence 1 | F A T N M G T S D P P T H T R L R K L V S Q |
| Sequence 2 | F V T N M N N S D G P T H T K L R K L V S T |
| Consensus | F * T N M * * S D * P T H T * L R K L V S * |

- **Problem: loss of information**
- **For larger blocks of sequences it "punishes" more distant members**

# Alignment profiles

- **Advantage: full representation of the sequence alignment (more information retained)**
- **Not only used in alignment methods, but also in sequence-database searching (to detect distant homologues)**
- **Also called PSSM in BLAST (*Position-specific scoring matrix*)**

# Multiple alignment profiles



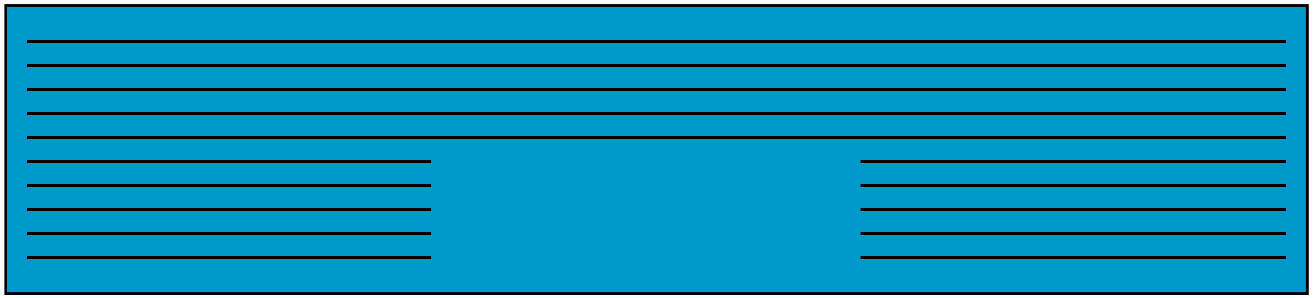

| | *i* | | |
|---|---|---|---|
| A | fA.. | fA.. | fA.. |
| C | fC.. | fC.. | fC.. |
| D | fD.. | fD.. | fD.. |
| • | • | • | • |
| • | • | • | • |
| • | • | • | • |
| W | fW.. | fW.. | fW.. |
| Y | fY.. | fY.. | fY.. |
| - | Gapo, gapx | Gapo, gapx | Gapo, gapx |

frequencies

*Position-dependent gap penalties*

# Profile building

- Example: each aa is represented as a frequency and gap penalties as weights.

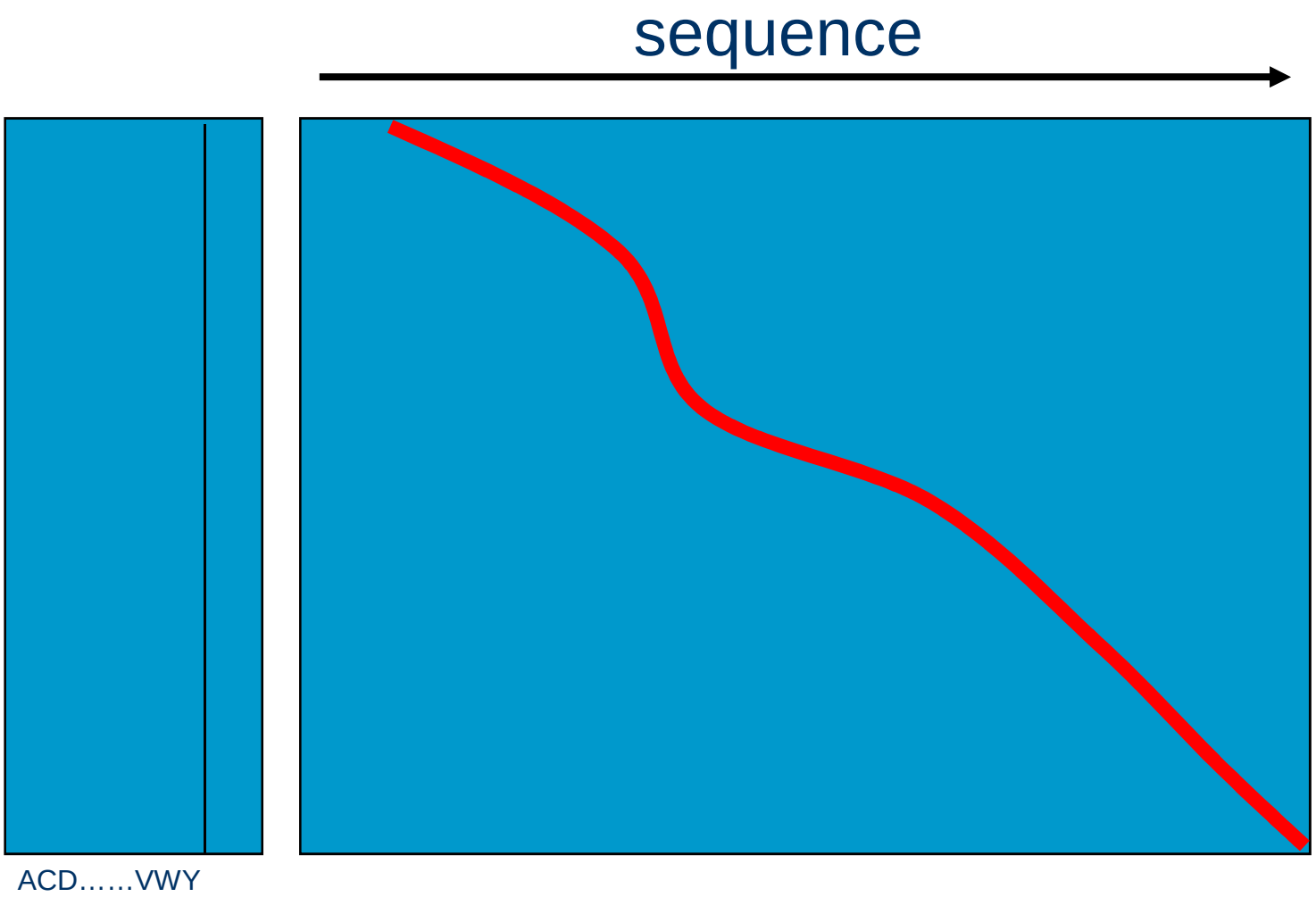


*i*

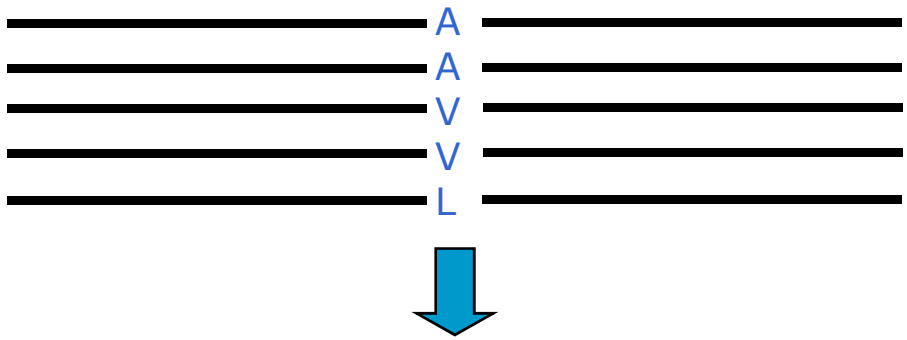
| | | | |
|---|---|---|---|
| A | 0.5 | 0.3 | 0 |
| C | 0 | 0.1 | 0.5 |
| D | 0 | 0 | 0.2 |
| • | • | • | • |
| • | • | • | • |
| • | • | • | • |
| W | 0 | 0.3 | 0.1 |
| Y | 0.5 | 0.3 | 0.2 |
| Gap penalties | 1.0 | 0.5 | 1.0 |

Position dependent gap penalties

# Profile-sequence alignment



# Sequence to profile alignment




0.4 A

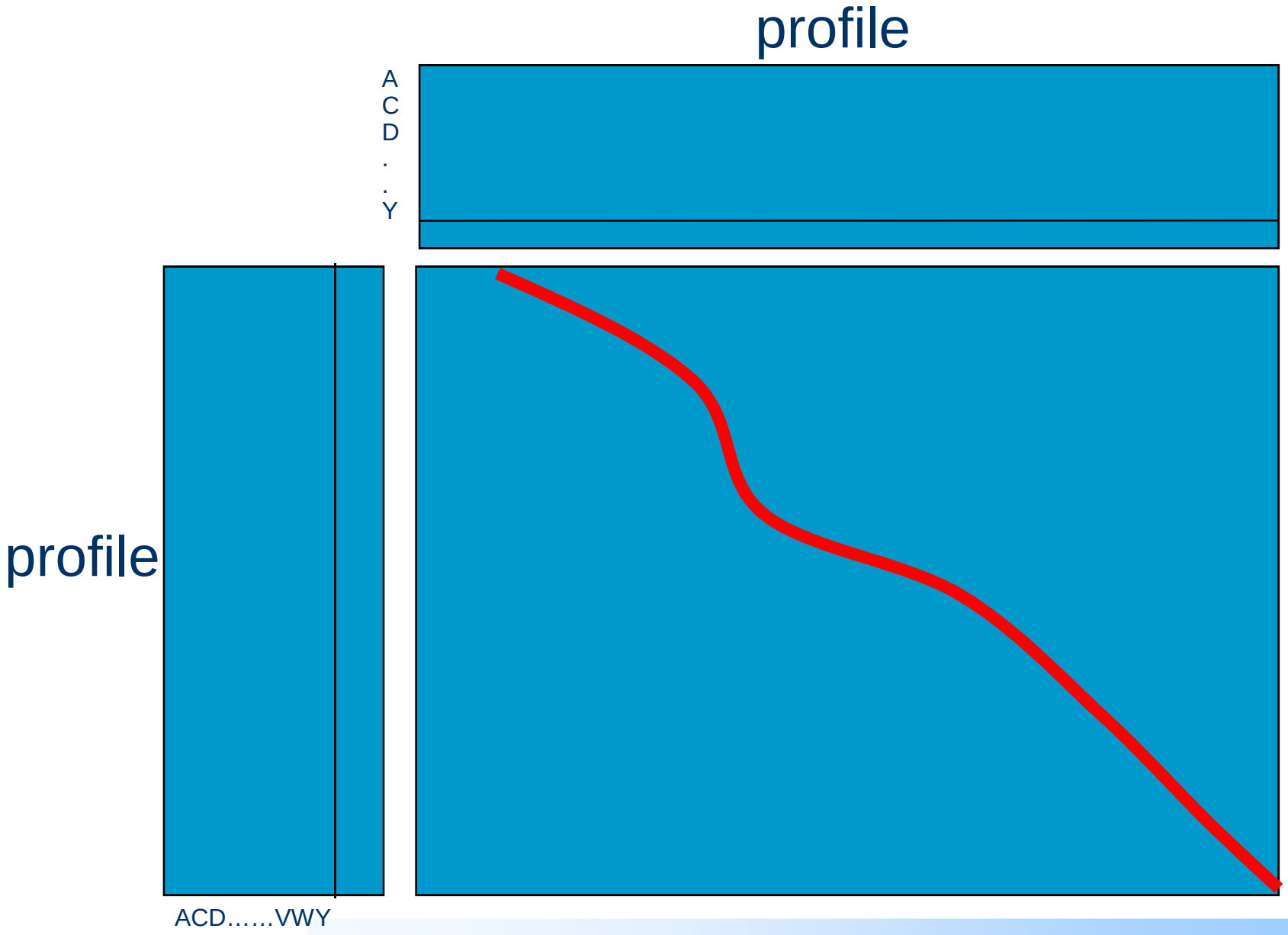
0.2 L

0.4 V

Score of amino acid L in a sequence that is aligned against this profile position:

$$\text{Score} = 0.4 * s(L, A) + 0.2 * s(L, L) + 0.4 * s(L, V)$$

# Profile-profile alignment



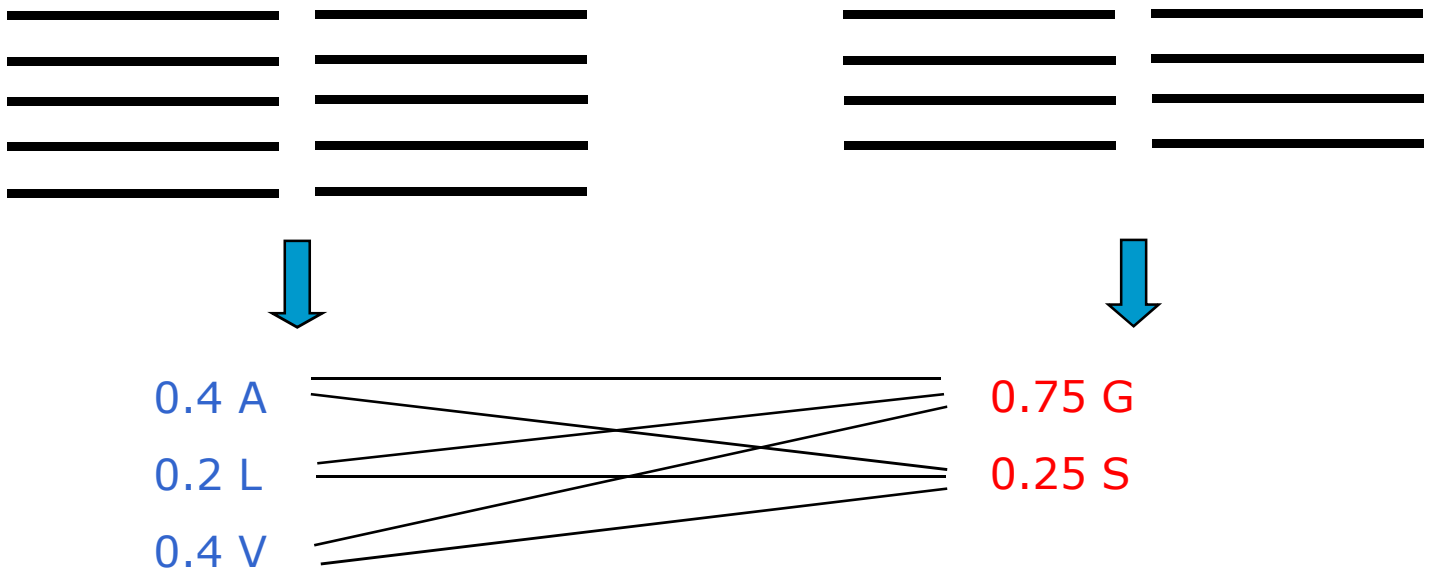
# General function for profile-profile scoring


- At each position (column) we have different residue frequencies for each amino acid (rows)

- Instead of saying $S=s(aa_1, aa_2)$ for pairwise alignment
- For comparing two profile positions we take:

$$S = \sum_{i}^{20} \sum_{j}^{20} faa_i \times faa_j \times s(aa_i, aa_j)$$

# Profile to profile alignment




Match score of these two alignment columns using the a.a frequencies at the corresponding profile positions:

Score = 0.4*0.75*s(A,G) + 0.2*0.75*s(L,G) + 0.4*0.75*s(V,G) +

+ 0.4*0.25*s(A,S) + 0.2*0.25*s(L,S) + 0.4*0.25*s(V,S)

s(x,y) is value in amino acid exchange matrix (*e.g.* PAM250, Blosum62) for amino acid pair (x,y)

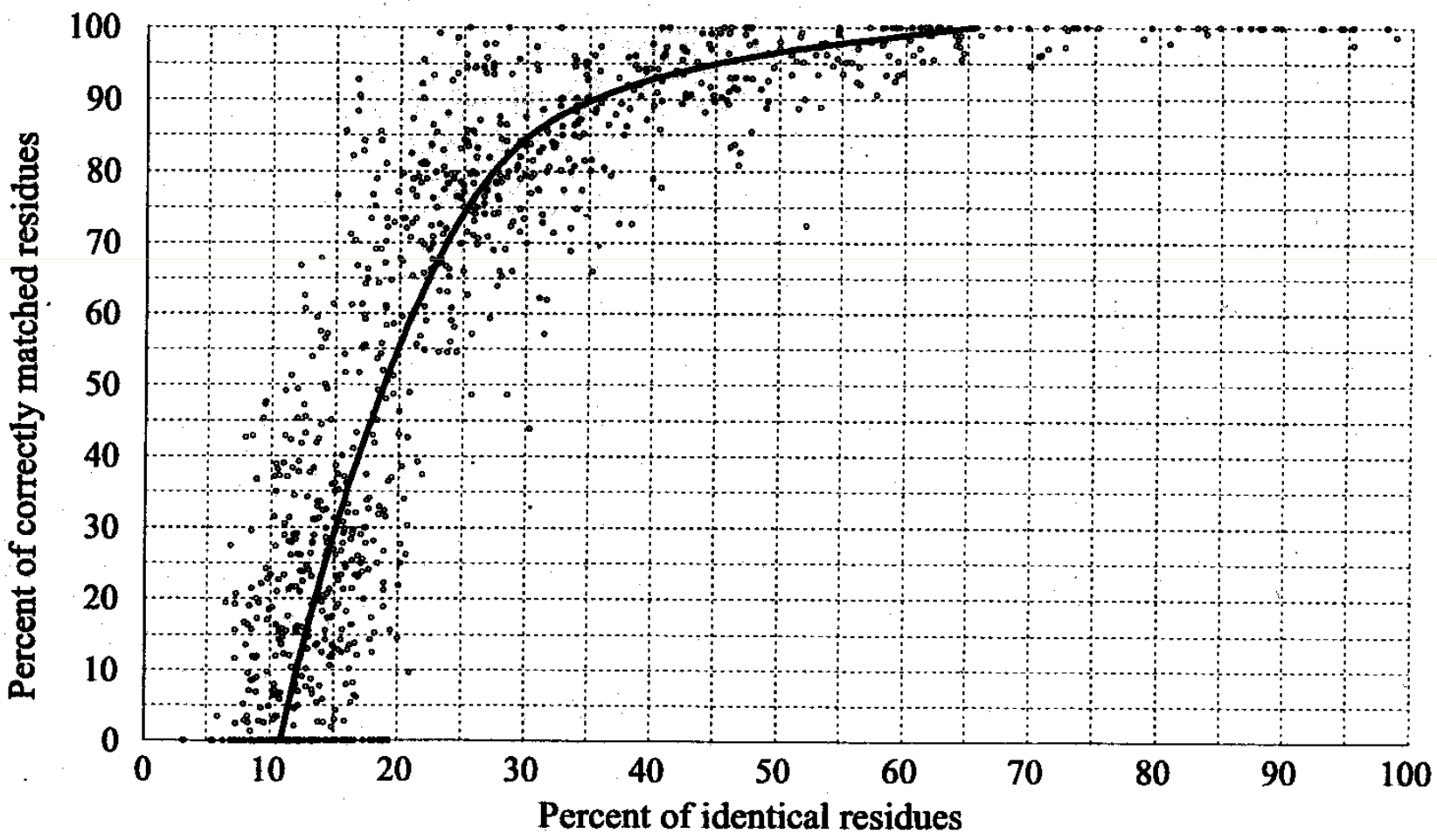
# Progressive alignment strategy

*Methods:*

- Biopat (Hogeweg and Hesper 1984 -- first integrated method ever)
- MULTAL (Taylor 1987)
- DIALIGN (1&2, Morgenstern 1996) – local MSA
- PRRP (Gotoh 1996)
- **ClustalW (Thompson et al 1994)**
- **PRALINE (Heringa 1999)**
- **T-Coffee (Notredame 2000)**
- **POA (Lee 2002)**
- **MUSCLE (Edgar 2004)**
- **PROBSCONS (Do, 2005)**
- **MAFFT**

# Pair-wise alignment quality *versus* sequence identity

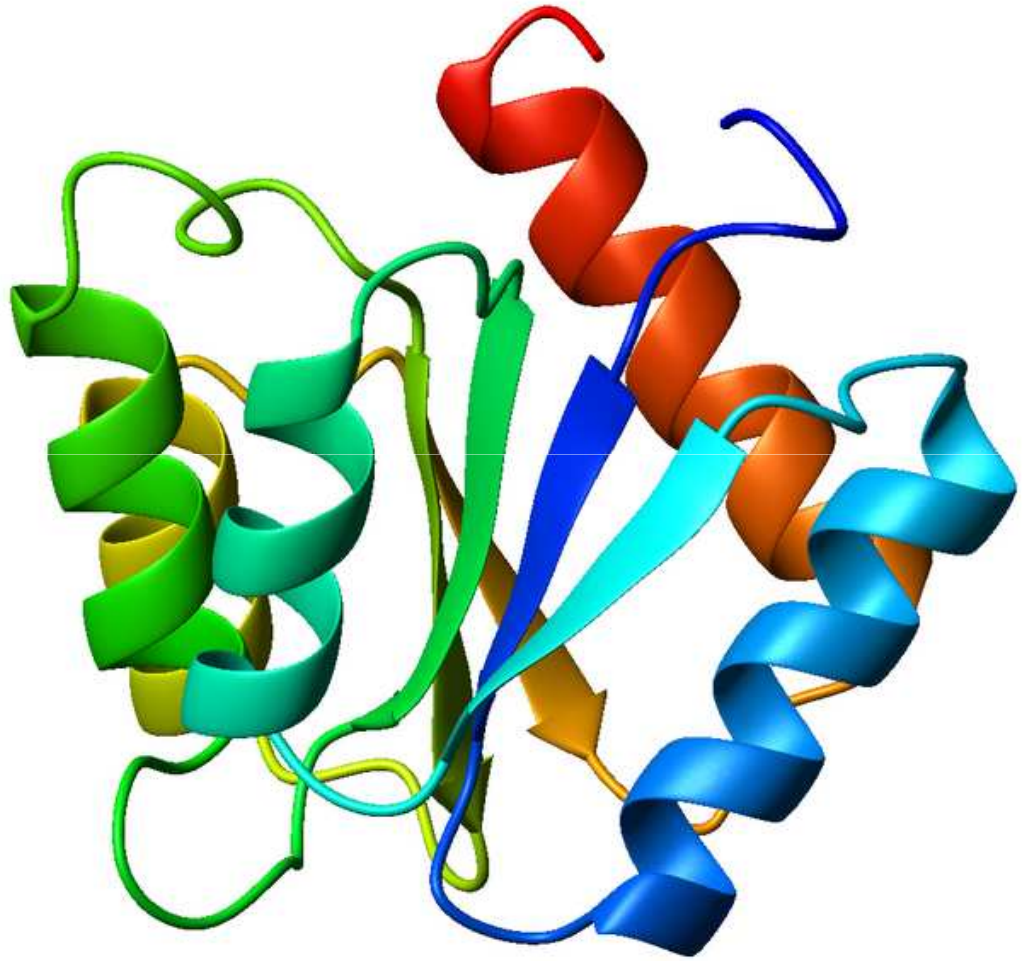
(Vogt et al., JMB 249, 816-831,1995)



# Clustal, ClustalW, ClustalX

- CLUSTAL W/X (Thompson *et al*., 1994) uses Neighbour Joining (NJ) algorithm (Saitou and Nei, 1984), widely used in phylogenetic analysis, to construct a guide tree (see lecture on phylogenetic methods).
- Sequence blocks are represented by profile, in which the individual sequences are additionally weighted according to the branch lengths in the NJ tree.
- Further carefully crafted heuristics include:
  - (i) local gap penalties
  - (ii) automatic selection of the amino acid substitution matrix, (iii) automatic gap penalty adjustment
  - (iv) mechanism to delay alignment of sequences that appear to be distant at the time they are considered.
- CLUSTAL (W/X) does not allow iteration (Hogeweg and Hesper, 1984; Corpet, 1988, Gotoh, 1996; Heringa, 1999, 2002)

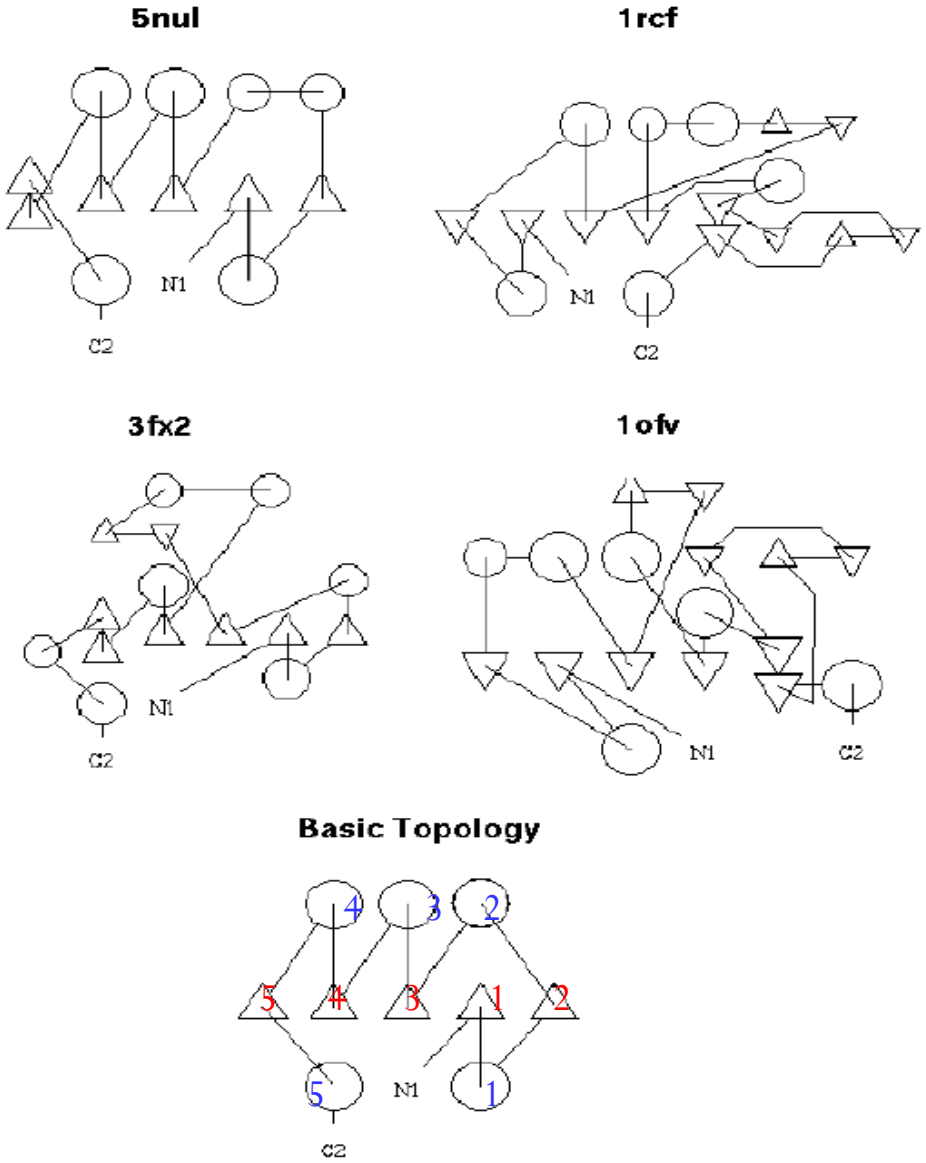
# Aligning 13 Flavodoxins + cheY



Flavodoxin fold: doubly wound βαβ structure

5(βα)

# Flavodoxin family - TOPS diagrams

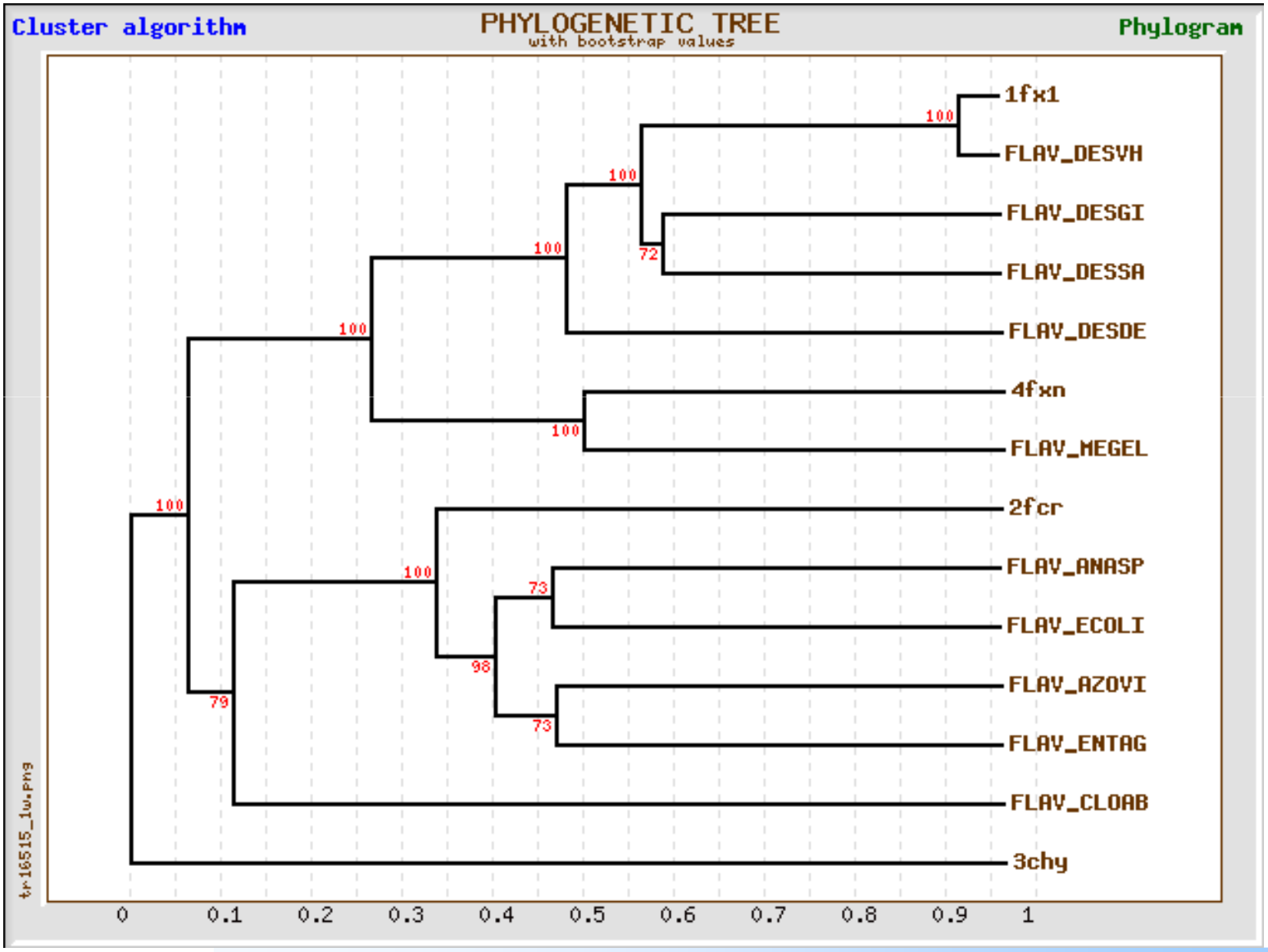



The basic topology of the flavodoxin fold is given below, the other four TOPS diagrams show flavodoxin folds with local insertions of secondary structure elements (David Gilbert)

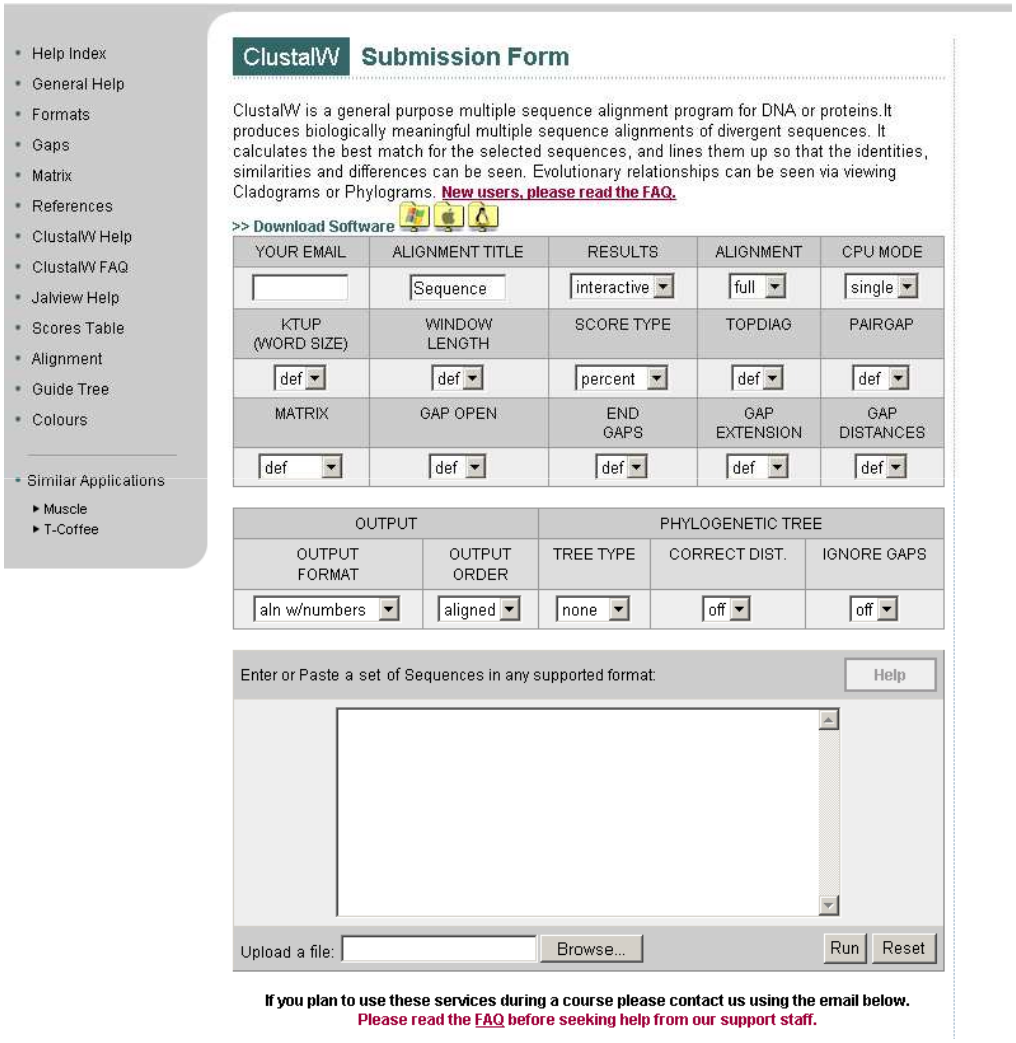
○ α-helix

△ β-strand

# Flavodoxin-cheY NJ tree



# ClustalW web-interface



# CLUSTAL X (1.64b) multiple sequence alignment Flavodoxin-cheY

```
1fx1          -PKALIVYGSTTGNTEYTAETIARQLANAG-Y-EVDSRDAASVEAGGLFEGFDLVLLGCSTWGDDSIE------LQDDFIPLFD-SLEETGAQGRK
FLAV_DESVH    MPKALIVYGSTTGNTEYTAETIARELADAG-Y-EVDSRDAASVEAGGLFEGFDLVLLGCSTWGDDSIE------LQDDFIPLFD-SLEETGAQGRK
FLAV_DESGI    MPKALIVYGSTTGNTEGVAEAIAKTLNSEG-M-ETTVVNVADVTAPGLAEGYDVVLLGCSTWGDDEIE------LQEDFVPLYE-DLDRAGLKDKK
FLAV_DESSA    MSKSLIVYGSTTGNTETAAEYVAEAFENKE-I-DVELKNVTDVSVADLGNGYDIVLFGCSTWGEEEIE------LQDDFIPLYD-SLENADLKGKK
FLAV_DESDE    MSKVLIVFGSSTGNTESIAQKLEELIAAGG-H-EVTLLNAADASAENLADGYDAVLFGCSAWGMEDLE------MQDDFLSLFE-EFNRFGLAGRK
FLAV_CLOAB    -MKISILYSSKTGKTERVAKLIEEGVKRSGNI-EVKTMNLDAVDKKFLQE-SEGIIFGTPTYYAN---------ISWEMKKWID-ESSEFNLEGKL
FLAV_MEGEL    --MVEIVYWSGTGNTEAMANEIEAAVKAAG-A-DVESVRFEDTNVDDVAS-KDVILLGCPAMGSE--E------LEDSVVEPFF-TDLAPKLKGKK
4fxn          ---MKIVYWSGTGNTEKMAELIAKGIIESG-K-DVNTINVSDVNIDELLN-EDILILGCSAMGDE--V------LEESEFEPFI-EEISTKISGKK
FLAV_ANASP    SKKIGLFYGTQTGKTESVAEIIRDEFGNDVVT----LHDVSQAEVTDLND-YQYLIIGCPTWNIGELQ---SD-----WEGLYS-ELDDVDFNGKL
FLAV_AZOVI    -AKIGLFFGSNTGKTRKVAKSIKKRFDDETMSD---ALNVNRVSAEDFAQ-YQFLILGTPTLGEGELPGLSSDCENESWEEFLP-KIEGLDFSGKT
2fcr          --KIGIFFSTSTGNTTEVADFIGKTLGAKADAP---IDVDDVTDPQALKD-YDLLFLGAPTWNTGADTERSGT----SWDEFLYDKLPEVDMKDLP
FLAV_ENTAG    MATIGIFFGSDTGQTRKVAKLIHQKLDGIADAP---LDVRRATREQFLS--YPVLLLGTPTLGDGELPGVEAGSQYDSWQEFTN-TLSEADLTGKT
FLAV_ECOLI    -AITGIFFGSDTGNTENIAKMIQKQLGKDVAD----VHDIAKSSKEDLEA-YDILLLGIPTWYYGEAQ-CD-------WDDFFP-TLEEIDFNGKL
3chy          --ADKELKFLVVDDFSTMRRIVRNLLKELG----FNNVEEAEDGVDALN------KLQAGGYGFV--I------SDWNMPNMDG-LELLKTIR---
                    .    ...       :   .                    .        :

1fx1          VACFGCGDSSYEYF--CGAVDAIEEKLKNLGAEIVQDG---------------LRIDGDPRAARDDIVGWAHDVRGAI--------------
FLAV_DESVH    VACFGCGDSSYEYF--CGAVDAIEEKLKNLGAEIVQDG---------------LRIDGDPRAARDDIVGWAHDVRGAI--------------
FLAV_DESGI    VGVFGCGDSSYTYF--CGAVDVIEKKAEELGATLVASS---------------LKIDGEPDSAE--VLDWAREVLARV--------------
FLAV_DESSA    VSVFGCGDSDYTYF--CGAVDAIEEKLEKMGAVVIGDS---------------LKIDGDPERDE--IVSWGSGIADKI--------------
FLAV_DESDE    VAAFASGDQEYEHF--CGAVPAIEERAKELGATIIAEG---------------LKMEGDASNDPEAVASFAEDVLKQL--------------
FLAV_CLOAB    GAAFSTANSIAGGS--DIALLTILNHLMVKGMLVYSGGVA----FGKPKTHLGYVHINEIQENEDENARIFGERIANKVKQIF----------
FLAV_MEGEL    VGLFGSYGWGSGE-----WMDAWKQRTEDTGATVIGTA---------------IVN-EMPDNAPECKE-LGEAAAKA--------------
4fxn          VALFGSYGWGDGK-----WMRDFEERMNGYGCVVVETP---------------LIVQNEPDEAEQDCIEFGKKIANI--------------
FLAV_ANASP    VAYFGTGDQIGYADNFQDAIGILEEKISQRGGKTVGYWSTDGYDFNDSKALR-NGKFVGLALDEDNQSDLTDDRIKSWVAQLKSEFGL-----
FLAV_AZOVI    VALFGLGDQVGYPENYLDALGELYSFFKDRGAKIVGSWSTDGYEFESSEAVV-DGKFVGLALDLDNQSGKTDERVAAWLAQIAPEFGLSL----
2fcr          VAIFGLGDAEGYPDNFCDAIEEIHDCFAKQGAKPVGFSNPDDYDYEESKSVR-DGKFLGLPLDMVNDQIPMEKRVAGWVEAVVSETGV-----
FLAV_ENTAG    VALFGLGDQLNYSKNFVSAMRILYDLVIARGACVVGNWPREGYKFSFSAALLENNEFVGLPLDQENQYDLTEERIDSWLEKLKPAVL-------
FLAV_ECOLI    VALFGCGDQEDYAEYFCDALGTIRDIIEPRGATIVGHWPTAGYHFEASKGLADDDHFVGLAIDEDRQPELTAERVEKWVKQISEELHLDEILNA
3chy          AD--GAMSALPVL-----MVTAEAKKENIIAAAQAGAS---------------GYV-VKPFTAATLEEKLNKIFEKLGM-------------
                 .  .          :     .     .
```

The secondary structures of 4 sequences are known and can be used to asses the alignment (red is β-strand, blue is α-helix)

# There are problems …

## Accuracy is very important !!!!

- Progressive multiple alignment is a greedy strategy: Alignment errors during the construction of the MSA **cannot be repaired anymore and** these errors are propagated into later progressive steps.
- Comparisons of sequences at early steps during progressive alignment cannot make use of information from other sequences.
- It is only later during the alignment progression that more information from other sequences (e.g. through profile representation) becomes employed in the alignment steps.

# Progressive multiple alignment

*"Once a gap, always a gap"*

*Feng & Doolittle, 1987*

# Additional strategies for multiple sequence alignment

- **Matrix extension (T-coffee)**
- Profile pre-processing (Praline)
- Secondary structure-induced alignment

Objective: try to avoid (early) errors

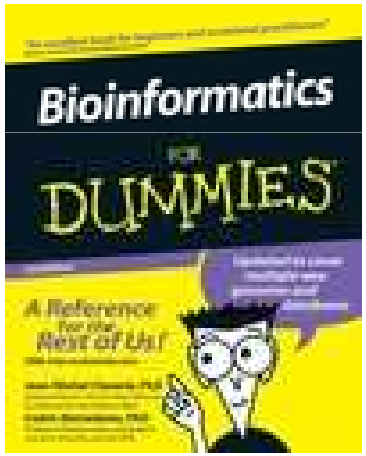
# Integrating alignment methods and alignment information with T-Coffee

- Integrating different pair-wise alignment techniques (NW, SW, ..)
- Combining different multiple alignment methods (consensus multiple alignment)
- Combining sequence alignment methods with structural alignment techniques
- Plug in user knowledge

# Matrix extension

## T-Coffee

*Tree-based Consistency Objective Function*
*For alignmEnt Evaluation*

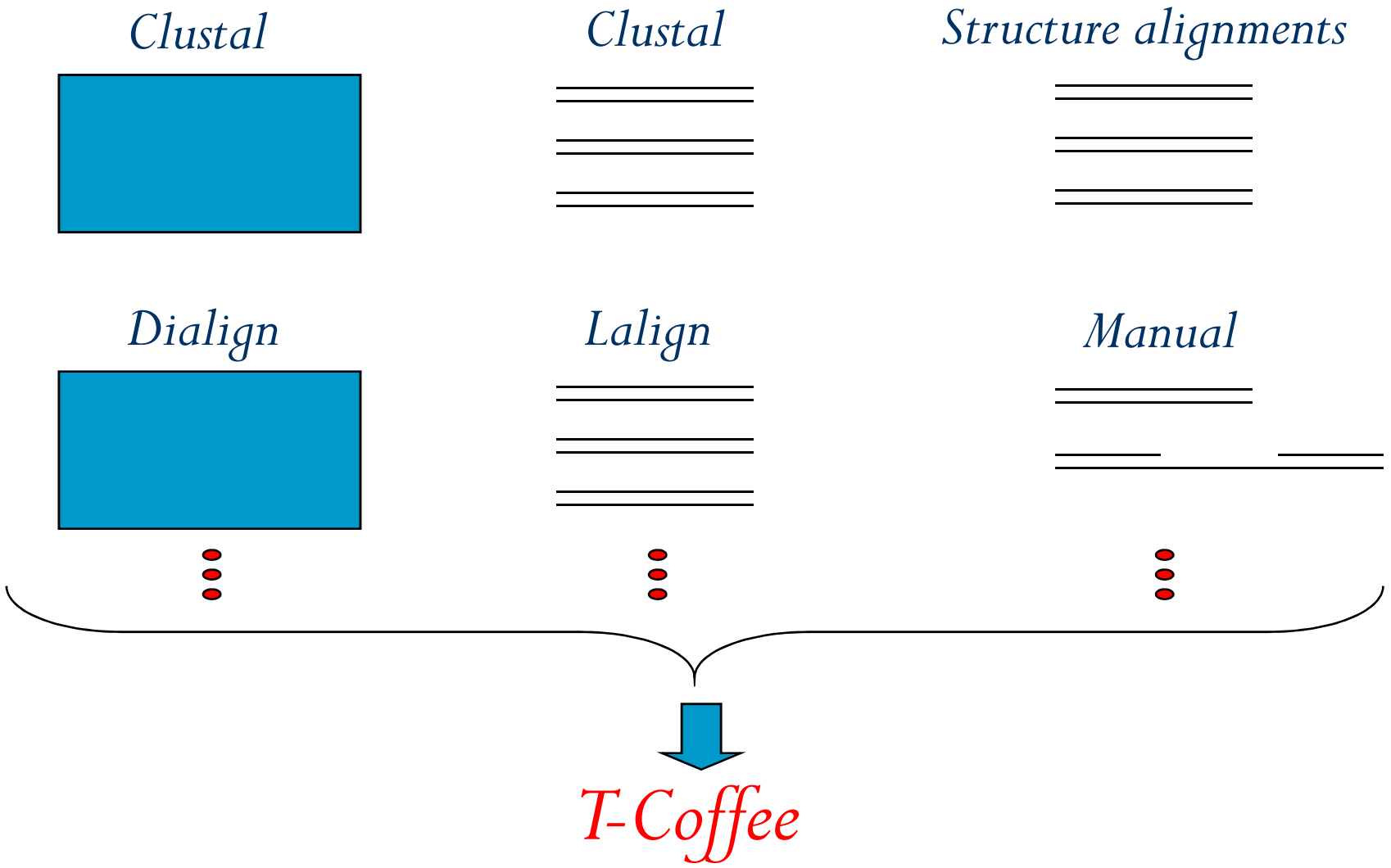



Cedric Notredame ("*Bioinformatics for dummies*")

Des Higgins

Jaap Heringa *J. Mol. Biol.*, 302, 205-217;2000

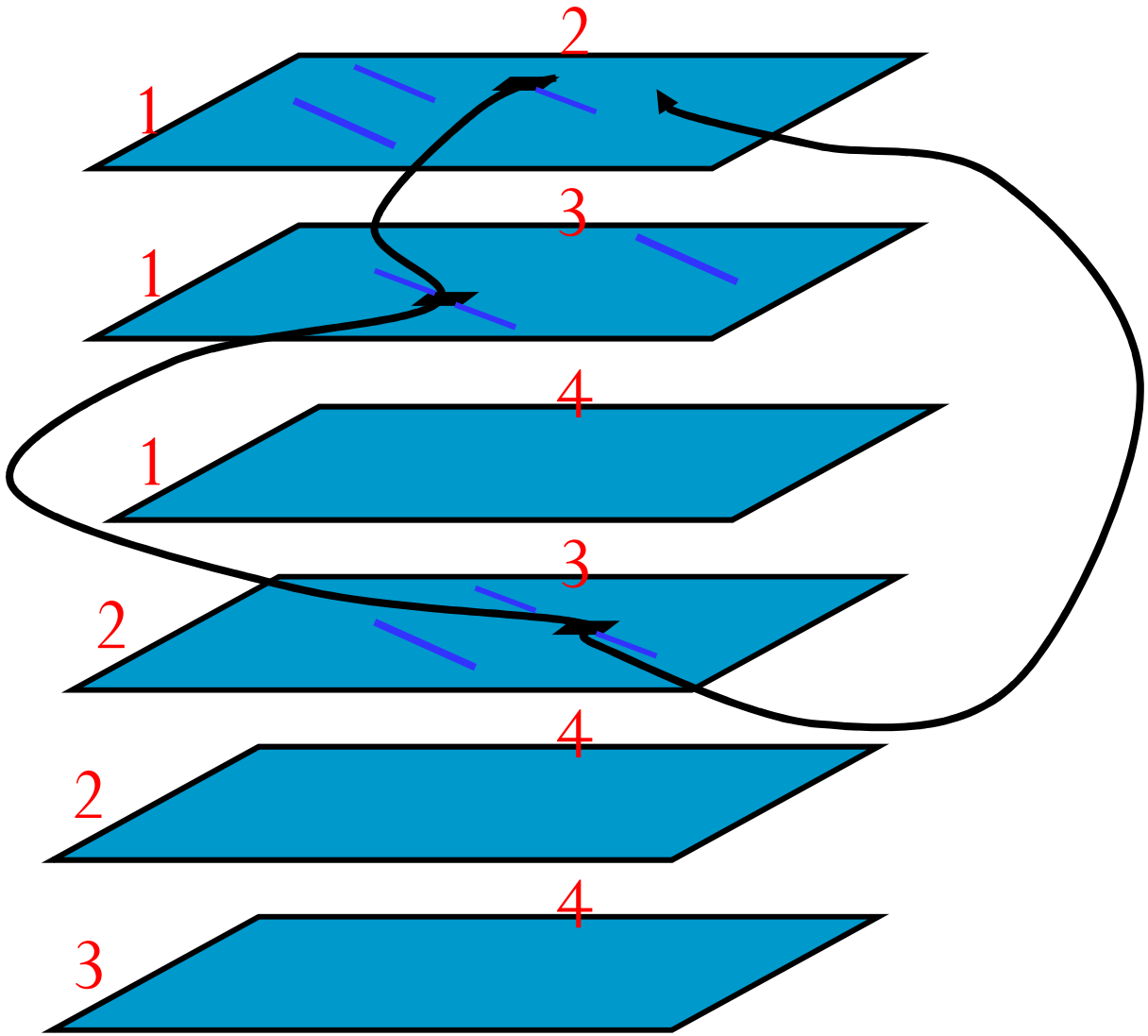
# Using different sources of alignment information



# T-Coffee library system

| *Seq1* | *AA1* | *Seq2* | *AA2* | *Weight* |
|---|---|---|---|---|
| | | ⋮ | | |
| *3* | *V31* | *5* | *L33* | *10* |
| *3* | *V31* | *6* | *L34* | *14* |
| | | ⋮ | | |
| *5* | *L33* | *6* | *R35* | *21* |
| *5* | *l33* | *6* | *I36* | *35* |
| | | ⋮ | | |

# *Matrix extension*



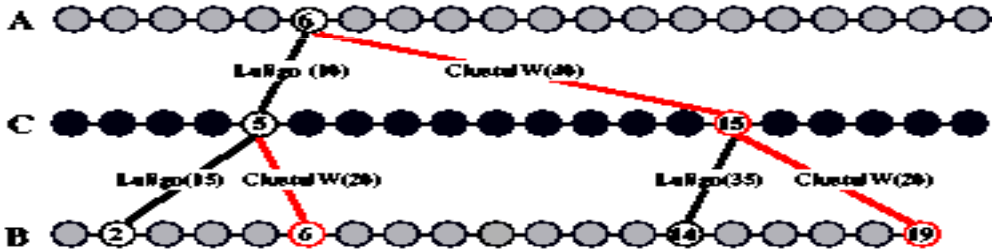
# Search matrix extension – alignment transitivity

**a)Signal Addition: Computation of the primary weights.**

| Primary Library | | |
|---|---|---|
| Res. 1 | Res. 2 | Weight |
| A(6) | B(3) | 30 |
| A(6) | B(14) | 25 |

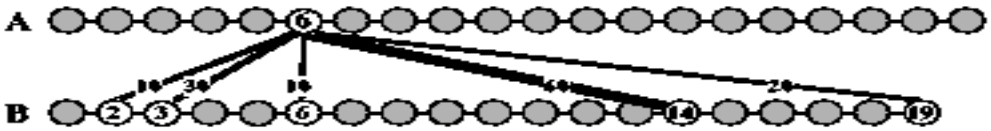

**b)Extension: Alignment of A and B through sequence C.**

| Res. 1 | Res. 2 | Weight |
|---|---|---|
| A(6) | B(2) | 10=min(10,15) |
| A(6) | B(6) | 10=min(10,20) |
| A(6) | B(14) | 35=min(40,35) |
| A(6) | B(19) | 20=min(40,20) |

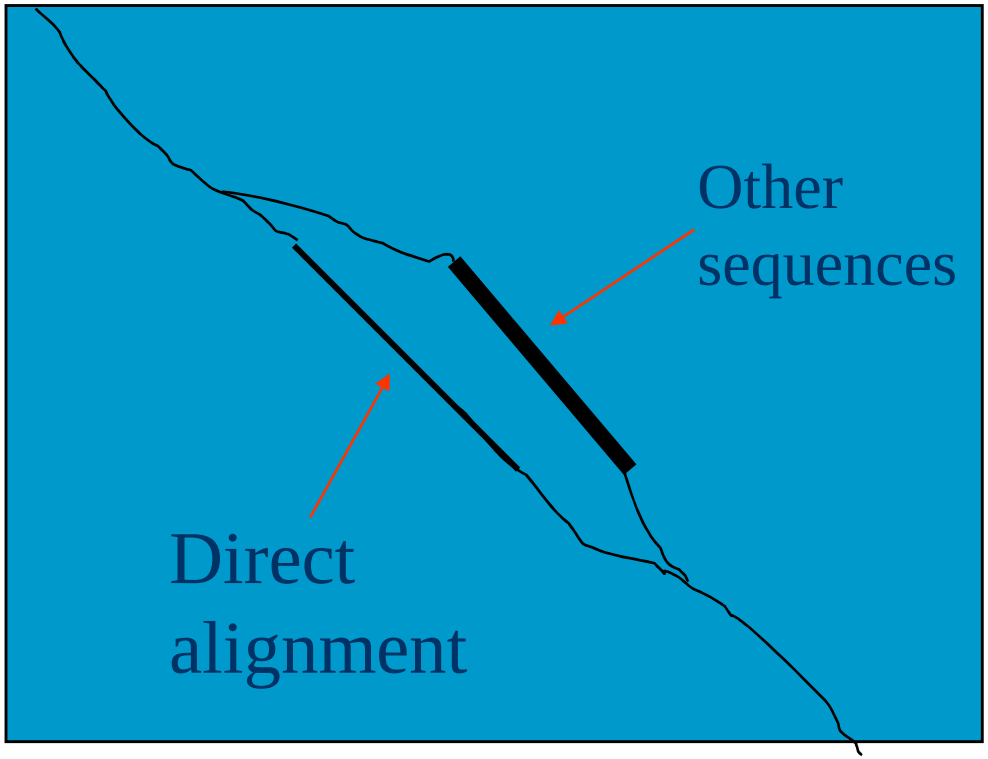



**c) Combination: Position Specific Scoring Scheme for A and B using information from sequence A, B and C.**

| Extended Library | | |
|---|---|---|
| Res. 1 | Res. 2 | Weight |
| A(6) | B(2) | 10 |
| A(6) | B(3) | 30 |
| A(6) | B(6) | 10 |
| A(6) | B(14) | 60=25+35 |
| A(6) | B(19) | 20 |

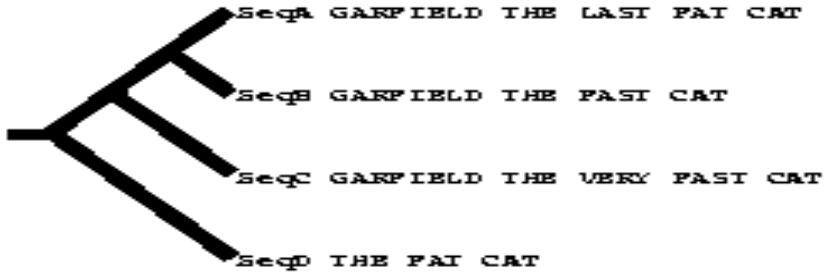


# T-Coffee



# Search matrix extension

a)Regular Progressive Alignment Strategy

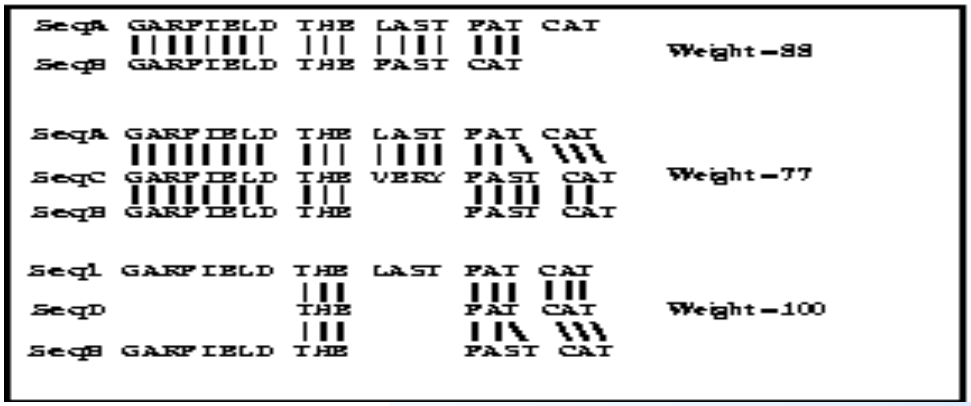


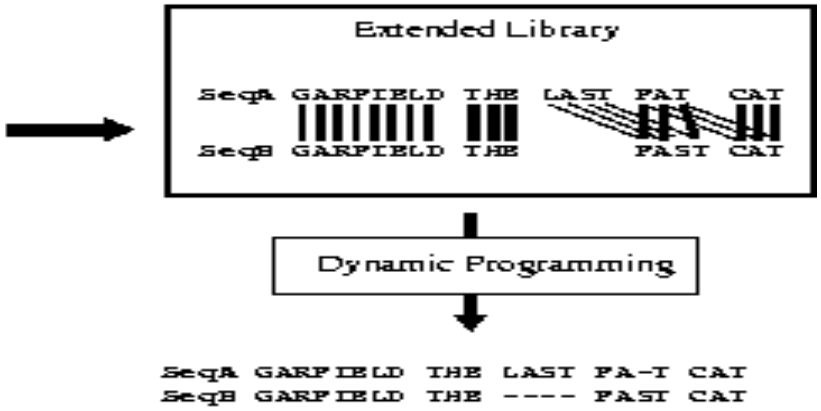

```
SeqA GARFIELD THE LAST FA-T CAT
SeqB GARFIELD THE FAST CA-T ---
SeqC GARFIELD THE VERY FAST CAT
SeqD -------  THE ---- FA-T CAT
```

b)Primary Library

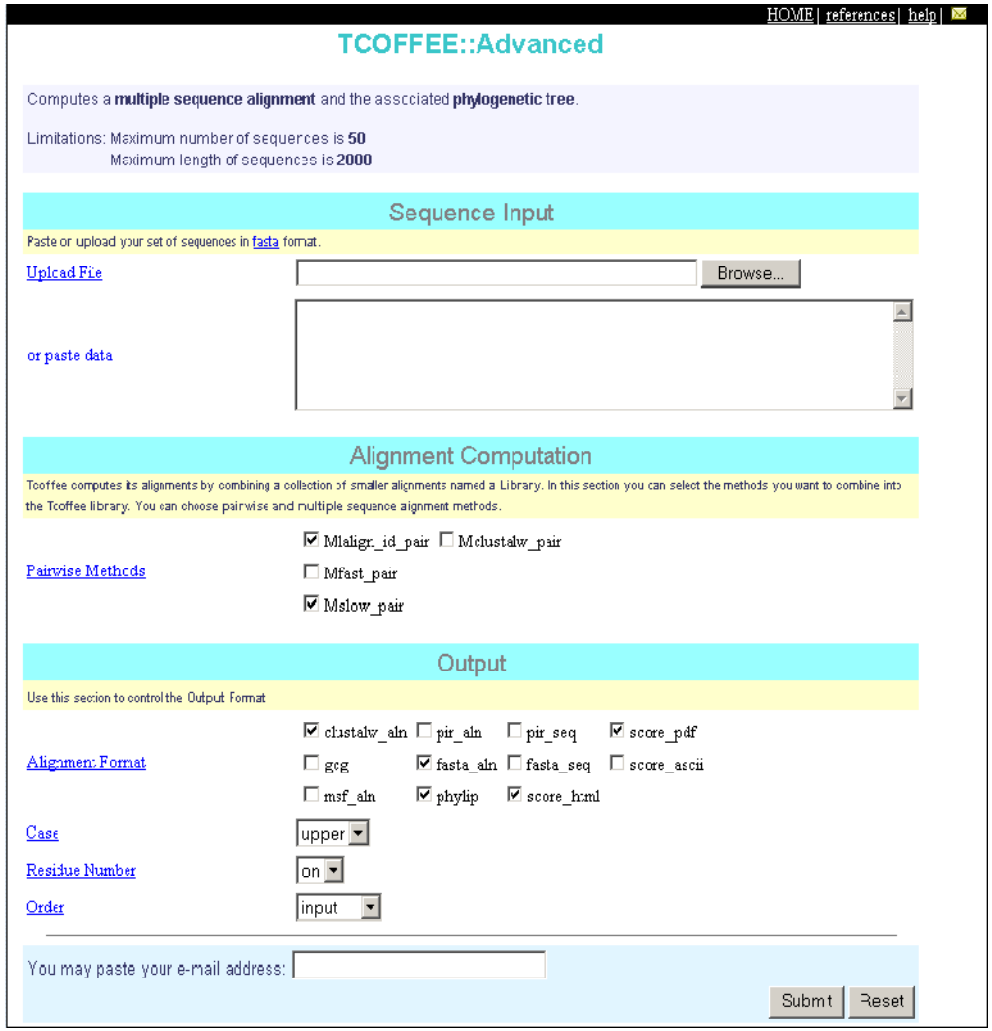
```
SeqA GARFIELD THE LAST FAT CAT   Prim_Weight=88
SeqB GARFIELD THE FAST CAT ---

SeqA GARFIELD THE LAST FA-T CAT  Prim_Weight=77
SeqC GARFIELD THE VERY FAST CAT

SeqA GARFIELD THE LAST FAT CAT   Prim_Weight=100
SeqD -------  THE ---- FAT CAT

SeqB GARFIELD THE ---- FAST CAT  Prim_Weight=100
SeqC GARFIELD THE VERY FAST CAT

SeqB GARFIELD THE FAST CAT       Prim_Weight=100
SeqD -------  THE FA-T CAT

SeqC GARFIELD THE VERY FAST CAT  Prim_Weight=100
SeqD -------  THE ---- FA-T CAT
```

c)Extended Library for seq1 and seq2



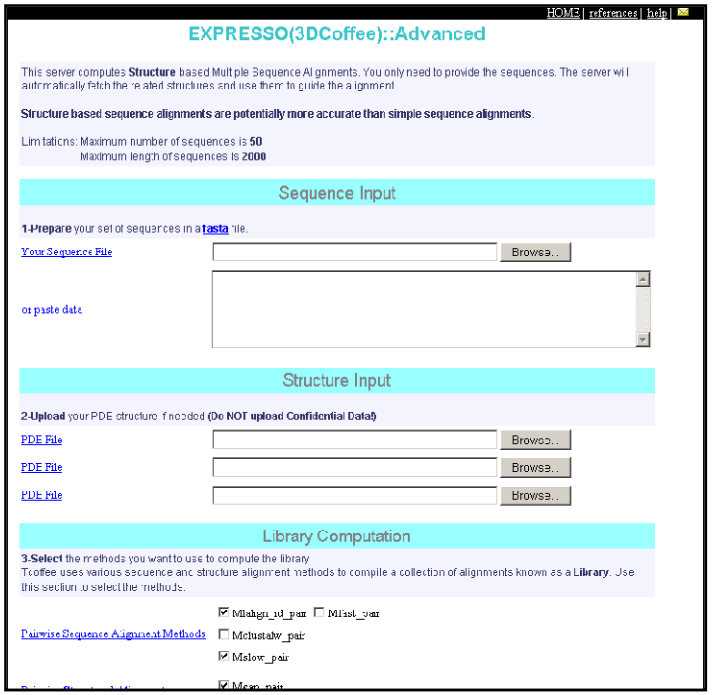



# T-COFFEE web-interface



# 3D-COFFEE

- **Computes structural alignments**
- **Structures associated with the sequences are retrieved and the information is used to optimise the MSA**
- **More accurate … but for many proteins we do not have a structure**



# but.....

## T-COFFEE (V1.23) multiple sequence alignment

## Flavodoxin-cheY

```
1fx1          ----PKALIVYGSTTGNTEYTAETIARQLANAG-YEVDSRDAASVE-AGGLFEGFDLVLLGCSTWGDDSIE------LQDDFIPL-FDSLEETGAQGRK-----
FLAV_DESVH    ---MPKALIVYGSTTGNTEYTAETIARELADAG-YEVDSRDAASVE-AGGLFEGFDLVLLGCSTWGDDSIE------LQDDFIPL-FDSLEETGAQGRK-----
FLAV_DESGI    ---MPKALIVYGSTTGNTEGVAEAIAKTLNSEG-METTVVNVADVT-APGLAEGYDVVLLGCSTWGDDEIE------LQEDFVPL-YEDLDRAGLKDKK-----
FLAV_DESSA    ---MSKSLIVYGSTTGNTETAAEYVAEAFENKE-IDVELKNVTDVS-VADLGNGYDIVLFGCSTWGEEEIE------LQDDFIPL-YDSLENADLKGKK-----
FLAV_DESDE    ---MSKVLIVFGSSTGNTESIAQKLEELIAAGG-HEVTLLNAADAS-AENLADGYDAVLFGCSAWGMEDLE------MQDDFLSL-FEEFNRFGLAGRK-----
4fxn          ------MKIVYWSGTGNTEKMAELIAKGIIESG-KDVNTINVSDVN-IDELL-NEDILILGCSAMGDEVLE-------ESEFEPF-IEEIS-TKISGKK-----
FLAV_MEGEL    -----MVEIVYWSGTGNTEAMANEIEAAVKAAG-ADVESVRFEDTN-VDDVA-SKDVILLGCPAMGSEELE-------DSVVEPF-FTDLA-PKLKGKK-----
FLAV_CLOAB    ----MKISILYSSKTGKTERVAKLIEEGVKRSGNIEVKTMNLDAVD-KKFLQ-ESEGIIFGTPTYYAN---------ISWEMKKW-IDESSEFNLEGKL-----
2fcr          -----KIGIFFSTSTGNTTEVADFIGKTLGAKA---DAPIDVDDVTDPQAL-KDYDLLFLGAPTWNTGA----DTERSGTSWDEFLYDKLPEVDMKDLP-----
FLAV_ENTAG    ---MATIGIFFGSDTGQTRKVAKLIHQKLDGIA---DAPLDVRRAT-REQF-LSYPVLLLGTPTLGDGELPGVEAGSQYDSWQEF-TNTLSEADLTGKT-----
FLAV_ANASP    ---SKKIGLFYGTQTGKTESVAEIIRDEFGNDV---VTLHDVSQAE-VTDL-NDYQYLIIGCPTWNIGEL--------QSDWEGL-YSELDDVDFNGKL-----
FLAV_AZOVI    ----AKIGLFFGSNTGKTRKVAKSIKKRFDDET-M-SDALNVNRVS-AEDF-AQYQFLILGTPTLGEGELPGLSSDCENESWEEF-LPKIEGLDFSGKT-----
FLAV_ECOLI    ----AITGIFFGSDTGNTENIAKMIQKQLGKDV---ADVHDIAKSS-KEDL-EAYDILLLGIPTWYYGEA--------QCDWDDF-FPTLEEIDFNGKL-----
3chy          ADKELKFLVVD--DFSTMRRIVRNLLKELGFN-NVE-EAEDGVDALNKLQ-AGGYGFVISDWNMPNMDGLE-------------LLKTIRADGAMSALPVLMV
                      :.    .    .  :   .                          ::

1fx1          ---------VACFGCGDSS--YEYFCGA-VDAIEEKLKNLGAEIVQDG--------------------LRIDGDPRAA--RDDIVGWAHDVRGAI--------
FLAV_DESVH    ---------VACFGCGDSS--YEYFCGA-VDAIEEKLKNLGAEIVQDG--------------------LRIDGDPRAA--RDDIVGWAHDVRGAI--------
FLAV_DESGI    ---------VGVFGCGDSS--YTYFCGA-VDVIEKKAEELGATLVASS--------------------LKIDGEPDSA----EVLDWAREVLARV--------
FLAV_DESSA    ---------VSVFGCGDSD--YTYFCGA-VDAIEEKLEKMGAVVIGDS--------------------LKIDGDPE----RDEIVSWGSGIADKI--------
FLAV_DESDE    ---------VAAFASGDQE--YEHFCGA-VPAIEERAKELGATIIAEG--------------------LKMEGDASND--PEAVASFAEDVLKQL--------
4fxn          ---------VALFGS------YGWGDGKWMRDFEERMNGYGCVVVETP--------------------LIVQNEPD--EAEQDCIEFGKKIANI---------
FLAV_MEGEL    ---------VGLFGS------YGWGSGEWMDAWKQRTEDTGATVIGTA--------------------IV--NEMP--DNAPECKELGEAAAKA---------
FLAV_CLOAB    ---------GAAFSTANSI--AGGSDIA-LLTILNHLMVKGMLVY----SGGVAFGKPKTHLGYVHINEIQENEDENARIFGERIANKVKQIF-----------
2fcr          ---------VAIFGLGDAEGYPDNFCDA-IEEIHDCFAKQGAKPVGFSNPDDYDYEESKSVRDG-KFLGLPLDMVNDQIPMEKRVAGWVEAVVSETGV------
FLAV_ENTAG    ---------VALFGLGDQLNYSKNFVSA-MRILYDLVIARGACVVGNWPREGYKFSFSAALLENNEFVGLPLDQENQYDLTEERIDSWLEKLKPAVL-------
FLAV_ANASP    ---------VAYFGTGDQIGYADNFQDA-IGILEEKISQRGGKTVGYWSTDGYDFNDSKALRNG-KFVGLALDEDNQSDLTDDRIKSWVAQLKSEFGL------
FLAV_AZOVI    ---------VALFGLGDQVGYPENYLDA-LGELYSFFKDRGAKIVGSWSTDGYEFESSEAVVDG-KFVGLALDLDNQSGKTDERVAAWLAQIAPEFGLSL----
FLAV_ECOLI    ---------VALFGCGDQEDYAEYFCDA-LGTIRDIIEPRGATIVGHWPTAGYHFEASKGLADDDHFVGLAIDEDRQPELTAERVEKWVKQISEELHLDEILNA
3chy          TAEAKKENIIAAAQAGASGYVVKPFT---AATLEEKLNKIFEKLGM-----------------------------------------------------------
                            .
```