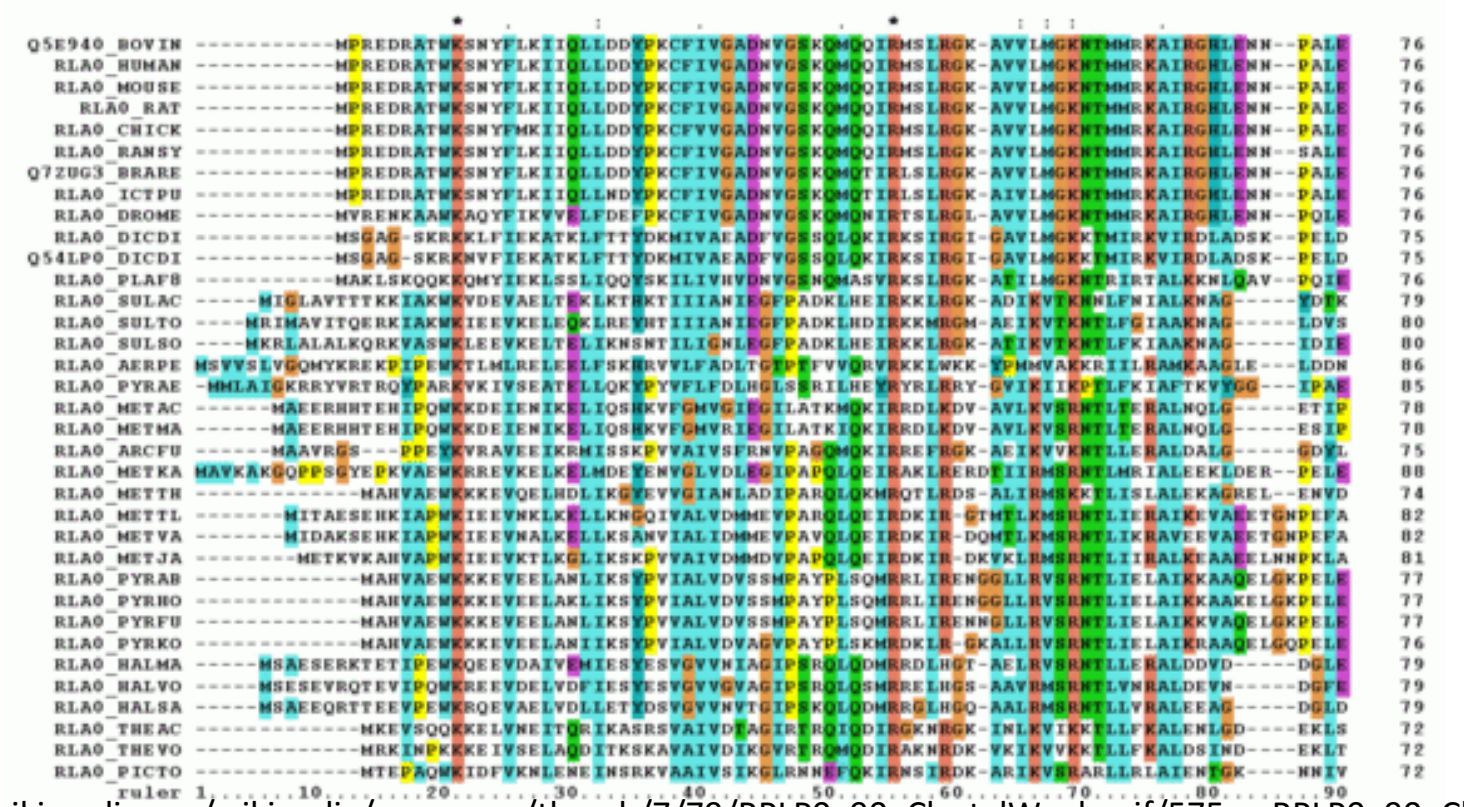
EECS730: Introduction to Bioinformatics

Lecture 06: Multiple Sequence Alignment



https://upload.wikimedia.org/wikipedia/commons/thumb/7/79/RPLP0_90_ClustalW_aln.gif/575px-RPLP0_90_ClustalW_aln.gif

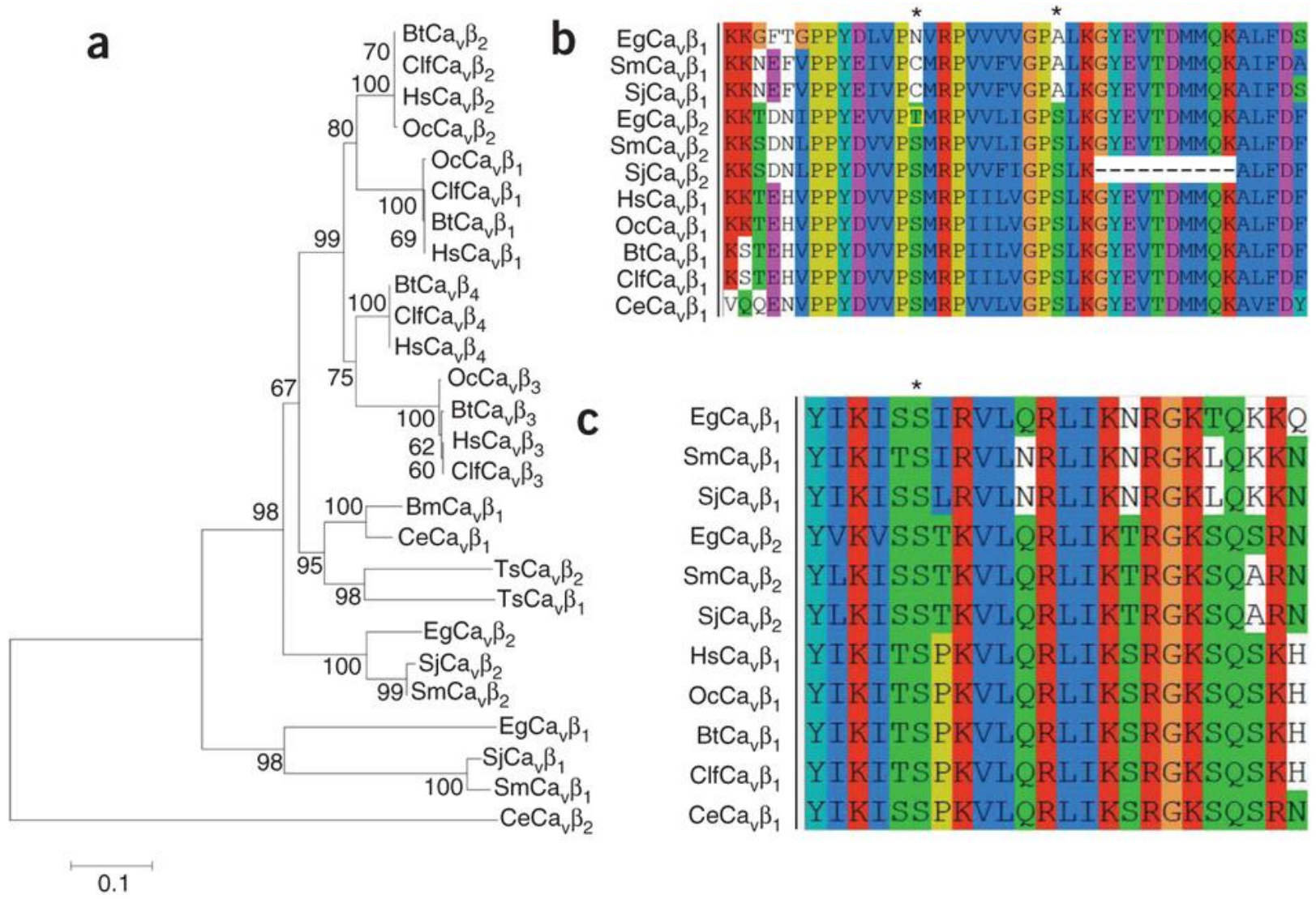
Slides adapted from Dr. Shaojie Zhang (University of Central Florida)

Multiple alignments

- Reveal evolutionary history (speciation-related mutations)
- Prediction of protein structure and protein function
- Determine consensus sequence for sequence assembly

- Generalization of the pairwise alignment algorithm

Multiple alignments



https://s9.postimg.org/z7797xw2n/ng_2757_F3.jpg

Object function

- To maximize the conservation of the alignment columns
- The more conserved the columns, the better the alignment

- Three scoring functions to characterize the conservation of the columns
  - Multiple Longest Common Sequence
  - Entropy
  - Sum-of-pair scores

Multiple Longest Common Subsequence

- A column is a “match” if all the letters in the column are the same

AAA
AAA
AAT
ATC

- Similar idea to the LCS problem formulation for pairwise alignment
- Only good for very similar sequences

Entropy

- Define frequencies for the occurrence of each letter in each column of multiple alignment (gap may be included into the alphabet)
  - $p_A = 1$, $p_T=p_G=p_C=0$ (1st column)
  - $p_A = 0.75$, $p_T = 0.25$, $p_G=p_C=0$ (2nd column)
  - $p_A = 0.50$, $p_T = 0.25$, $p_C=0.25$ $p_G=0$ (3rd column)
- Compute entropy of each column

$$-\sum_{X=A,T,G,C} p_X \log p_X$$

AAA
AAA
AAT
ATC

Entropy cont.

- Best case

$$entropy\begin{pmatrix} A \\ A \\ A \\ A \end{pmatrix} = 0$$

- Worst case

$$entropy\begin{pmatrix} A \\ T \\ G \\ C \end{pmatrix} = -\sum \frac{1}{4}\log\frac{1}{4} = -4(\frac{1}{4} * -2) = 2$$

- Entropy for a multiple alignment is the sum of entropies of its columns

Sum-of-pair score

Every multiple alignment induces pairwise alignments

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

Induces:

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

Sum-of-pair score cont.

- The alignment score for the multiple alignment is the sum of the alignment scores of all of its induced pairwise alignments

- Consider pairwise alignment of sequences

  a_i and a_j

  imposed by a multiple alignment of k sequences
- Denote the score of this suboptimal (not necessarily optimal) pairwise alignment as

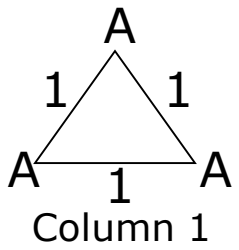
  $s^*(a_i, a_j)$
- Sum up the pairwise scores for a multiple alignment:

  $s(a_1,\ldots,a_k) = \Sigma_{i,j}\, s^*(a_i, a_j)$

Sum-of-pair score cont.

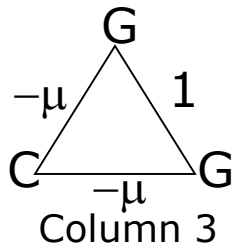
- It can also be computed column-wise
- This is useful for dynamic programming algorithm that breaks the problem into smaller sub-problems

a_1 ATG–C–AAT
· A–G–CATAT
a_k ATCCCATTT



Column 1

Score=3



Column 3

Score = $1 - 2\mu$

How to compute optimal multiple alignment

- Extending the dynamic programming algorithm for pairwise alignment
- Recall what does the score mean for each entry in the 2D dynamic programming table for pairwise alignments
- What is the dimension of the multiple sequence alignment dynamic programming table and what should we store there?

How to compute optimal multiple alignment

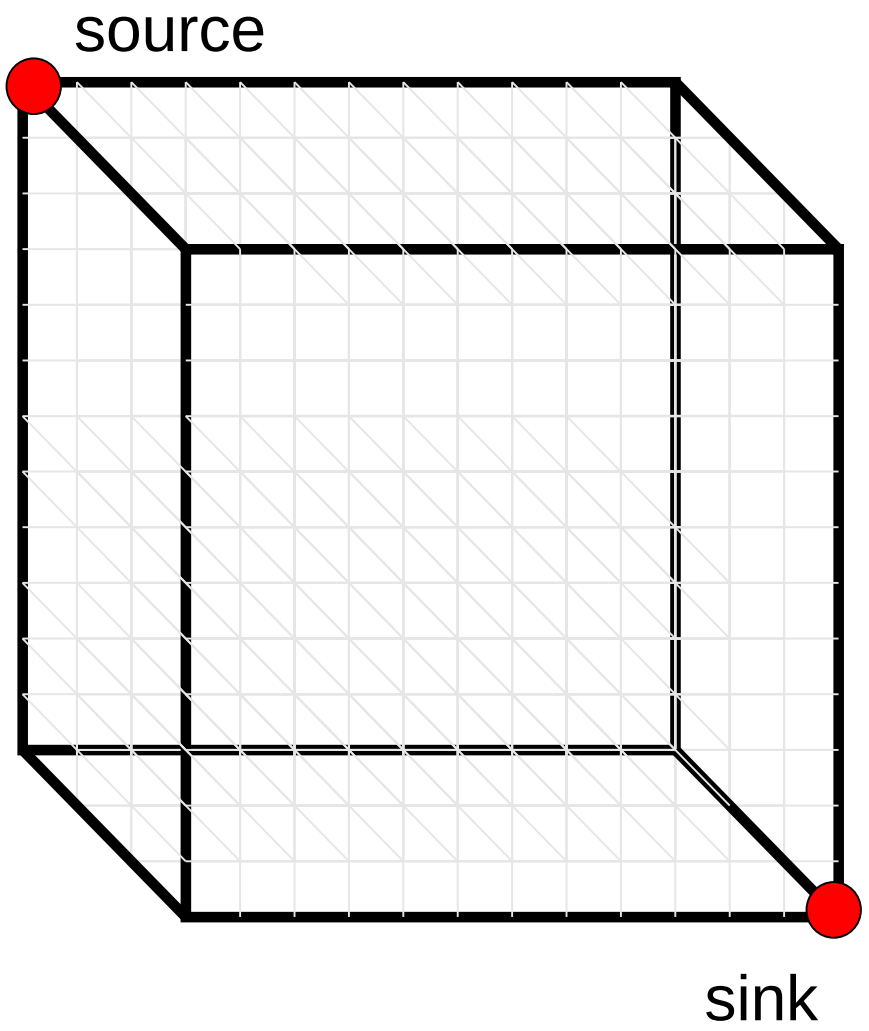
- Each entry in the 2D DP table stores the best score for aligning the prefixes of the two sequences:
  - The entry (i, j) stores alignment score between S1(0, i) and S2(0, j), where S1 and S2 are the two sequences being aligned.
- This can also be extended to multiple alignment case
- How many different combinations of prefixes alignment for n sequences?
  - l1 * l2 * … * ln, where l is the length of a given sequence
  - So the DP table for multiple alignment is an n-dimensional table
  - It degenerates to 2D table for pairwise alignment

How to compute optimal multiple alignment

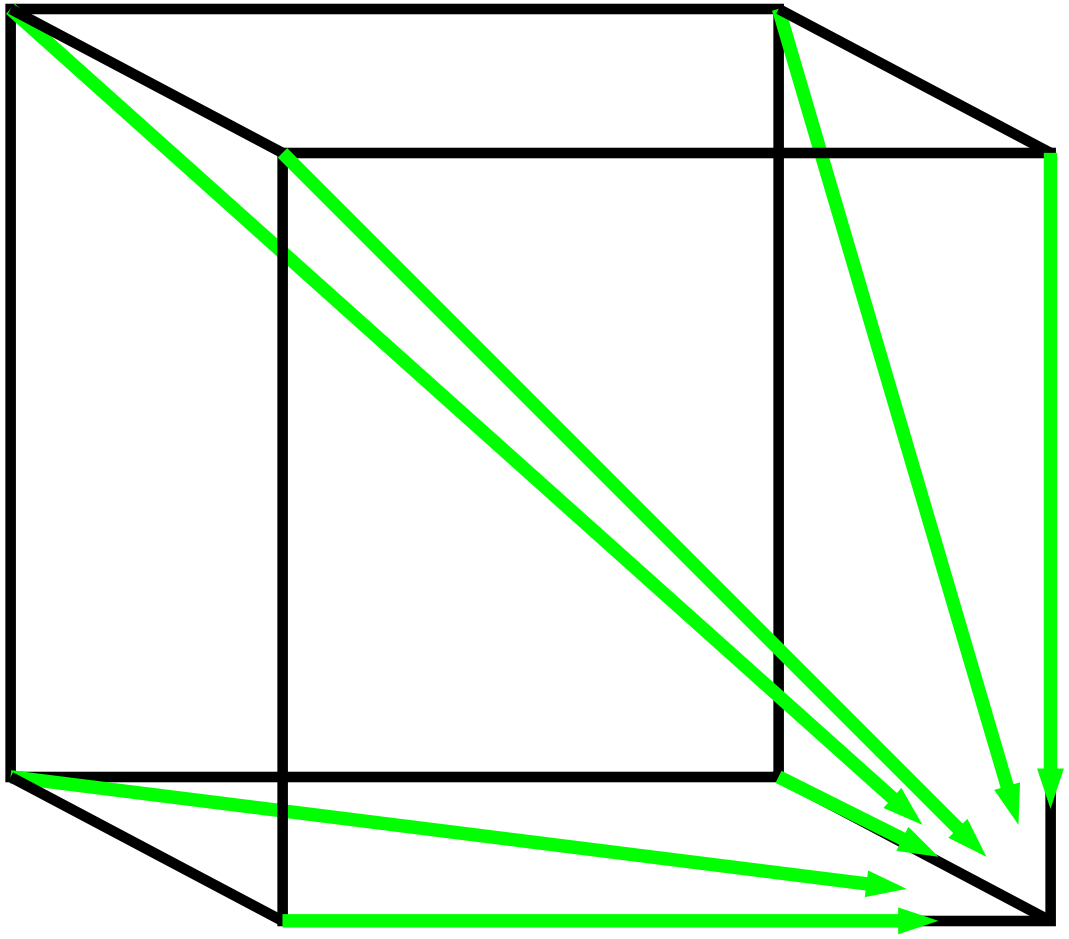
- Now, how many entries do we need to refer to in order to compute the score of an entry?
- Recall that each stage of the DP algorithm append one or zero character of each sequence to the existing alignment
- There are two choices (one or zero character), and there are n sequences in total, so there are 2^n entries to refer in total
- More precisely, we do not allow a column of all gaps, which means the combination of all zeros is invalid, and it reduces the number of entries to refer to as 2^n – 1
- For pairwise alignments, we need to refer to 2^2-1 = 3 entries in the DP table (left, up, and upper-left)

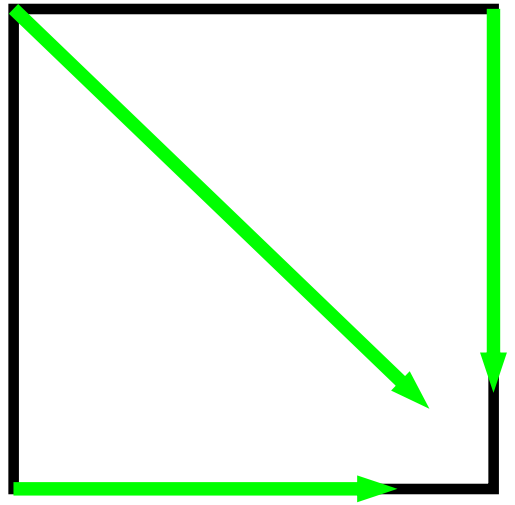
An example for aligning three sequences

- A three-dimensional Manhattan Tourist Problem
- The DP matrix is 3D
- We aims at finding the path that corresponds to the best alignment



Filling the DP matrix

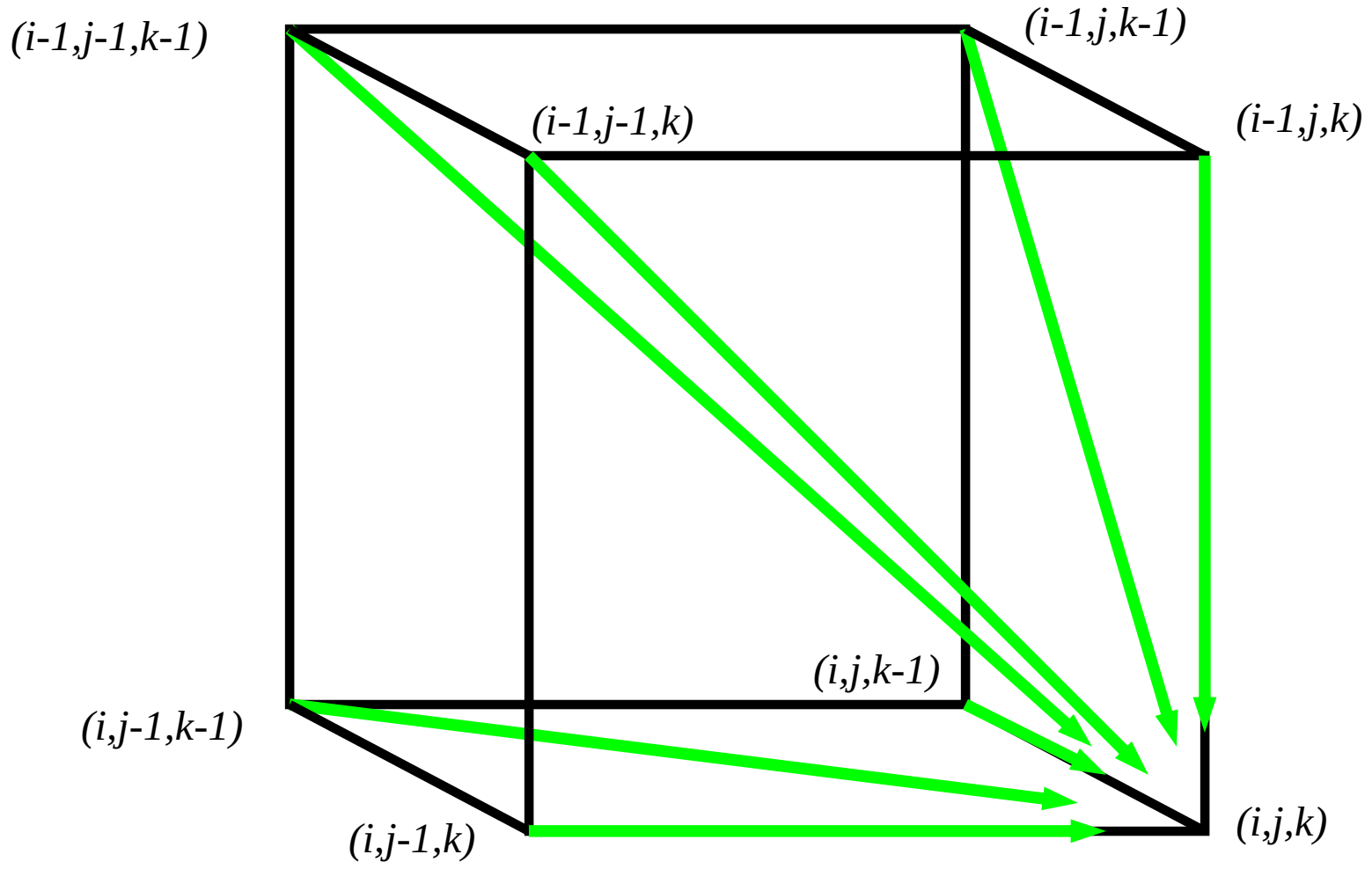




In **2-D**, 3 edges in each unit square

In **3-D**, 7 edges in each unit cube

Architecture of the 3D alignment cell



Recursive function for MSA

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

  cube diagonal: no indels

  face diagonal: one indel

  edge diagonal: two indels

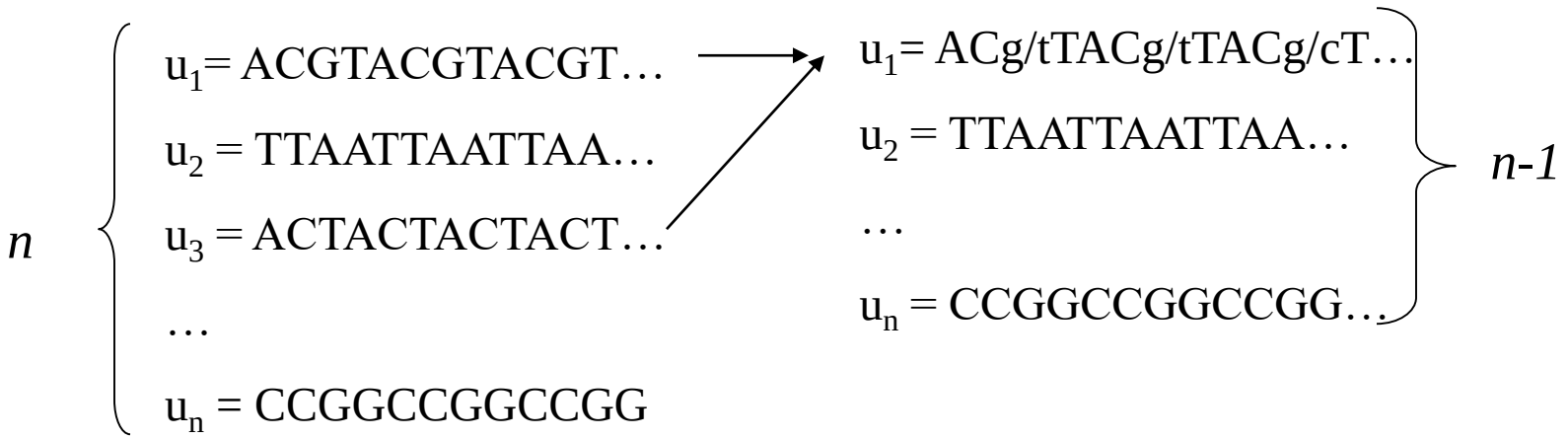
- $S(x, y, z)$ is an entry in the 3-D scoring matrix
- $\delta(x, y, z)$ can be computed as the sum-of-pair score

What is the time complexity?

- We have l^n entries to fill, filling each entry takes 2^n time
- The overall complexity is O(l^n * 2^n)
- Conclusion: dynamic programming approach for alignment between two sequences is easily extended to ***n*** sequences but it is impractical due to exponential running time.

Progressive multiple sequence alignment

- Perform all-against-all pairwise alignments for the *n* sequences
- Choose most similar pair of strings and combine into a profile , thereby reducing alignment of *n* sequences to an alignment of *n-1* sequences/profiles. **Repeat until 1 sequence/profile remains**
- This is a heuristic greedy method



Completing the iteration

- We need to find the most similar pair of strings at each iteration
- Therefore we need to redefine the similarity between the newly summarized profile and the other strings/profiles
- Using the Neighbor Joining Algorithm

$$d(u,k) = \frac{1}{2}[d(f,k) + d(g,k) - d(f,g)]$$

https://en.wikipedia.org/wiki/Neighbor_joining

- Here, *u* is the new profile, and *f* and *g* are the two strings/profiles that form *u*, and *k* is an arbitrary string/profile remains

Completing the iteration

- Profile representation

```
  -  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

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
```

- A single sequence can be viewed as a special case profile

Aligning two profiles

- Two profiles represented using frequencies can be aligned using slightly modified pairwise sequence alignment algorithm
- The score for matching two columns can be computed as the sum-of-pair scores of the two columns
- Affine gap penalty can be easily incorporated

Example

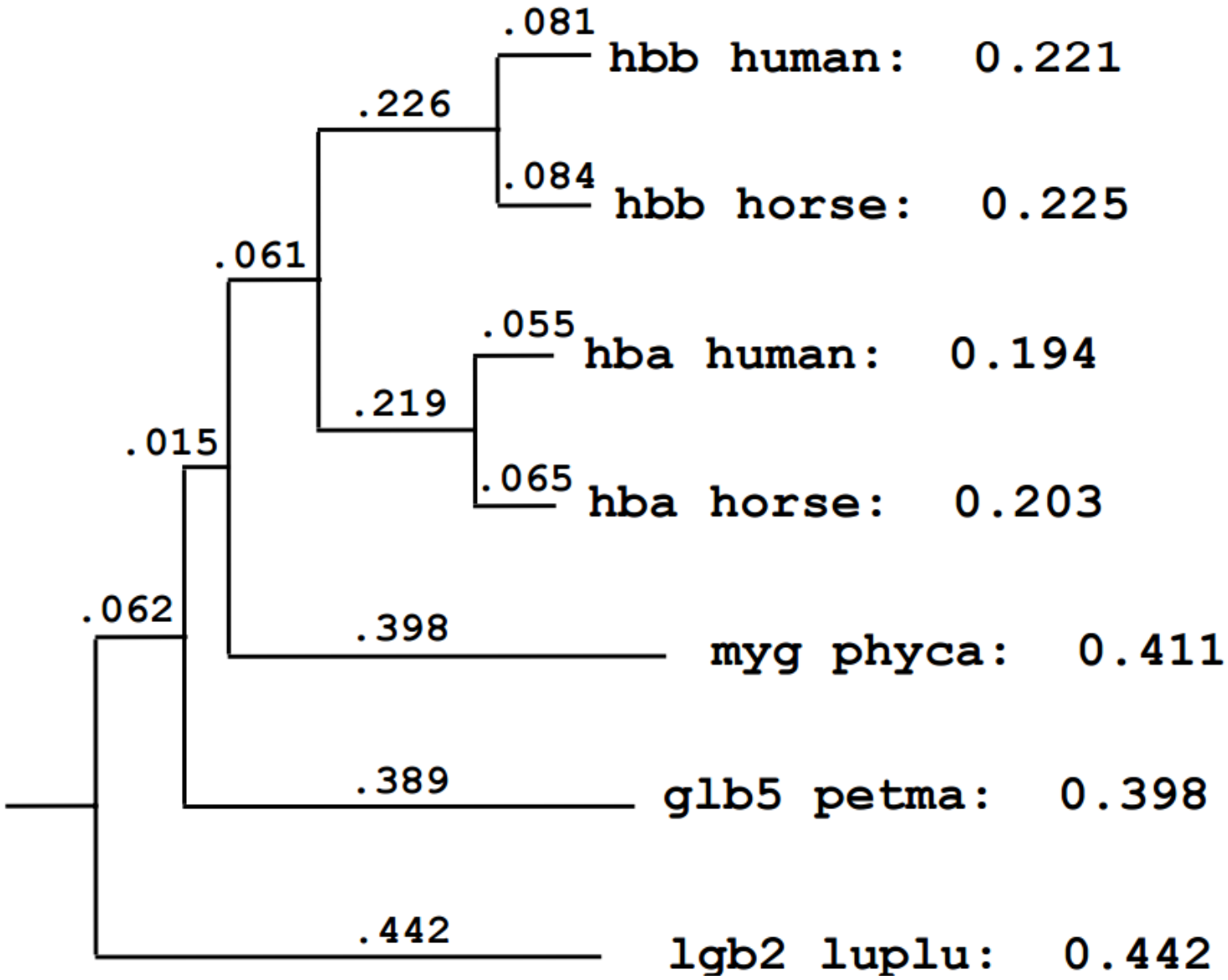
```
 DISTANCES between protein sequences:

 Calculated over: 1 to 167
 Correction method: Simple distance (no corrections)
 Distances are: observed number of substitutions per 100 amino acids
Symmatrix version 1
Number of matrices: 1

//
Matrix 1, dimension: 7

Key for column and row indices:

  1 hba_human
  2 hba_horse
  3 hbb_human
  4 hbb_horse
  5 glb5_petma
  6 myg_phyca
  7 lgb2_luplu

 Matrix 1: Part 1
```

	1	2	3	4	5	6	7
1	0.00	12.06	54.68	55.40	64.12	71.74	83.57
2		0.00	55.40	53.96	64.89	72.46	82.86
3			0.00	16.44	74.26	73.94	82.52
4				0.00	75.74	73.94	81.12
5					0.00	75.91	82.61
6						0.00	80.95
7							0.00

http://statweb.stanford.edu/~nzhang/345_web/sequence_slides3.pdf

The guide tree

For computation of the distances see https://en.wikipedia.org/wiki/Neighbor_joining



http://statweb.stanford.edu/~nzhang/345_web/sequence_slides3.pdf

Progressive alignment based on the guide tree

In our globin example, we align in the following order:

a) human and horse beta-globin;
b) human and horse alpha-globin;
c) the two beta-globins and the two alpha-globins;
d) myoglobin and the haemoglobins;
e) cyanohaemoglobin and the combined haemoglobin, myoglobin group;
f) leghaemoglobin and the rest.

http://statweb.stanford.edu/~nzhang/345_web/sequence_slides3.pdf

Time complexity for progressive multiple alignment

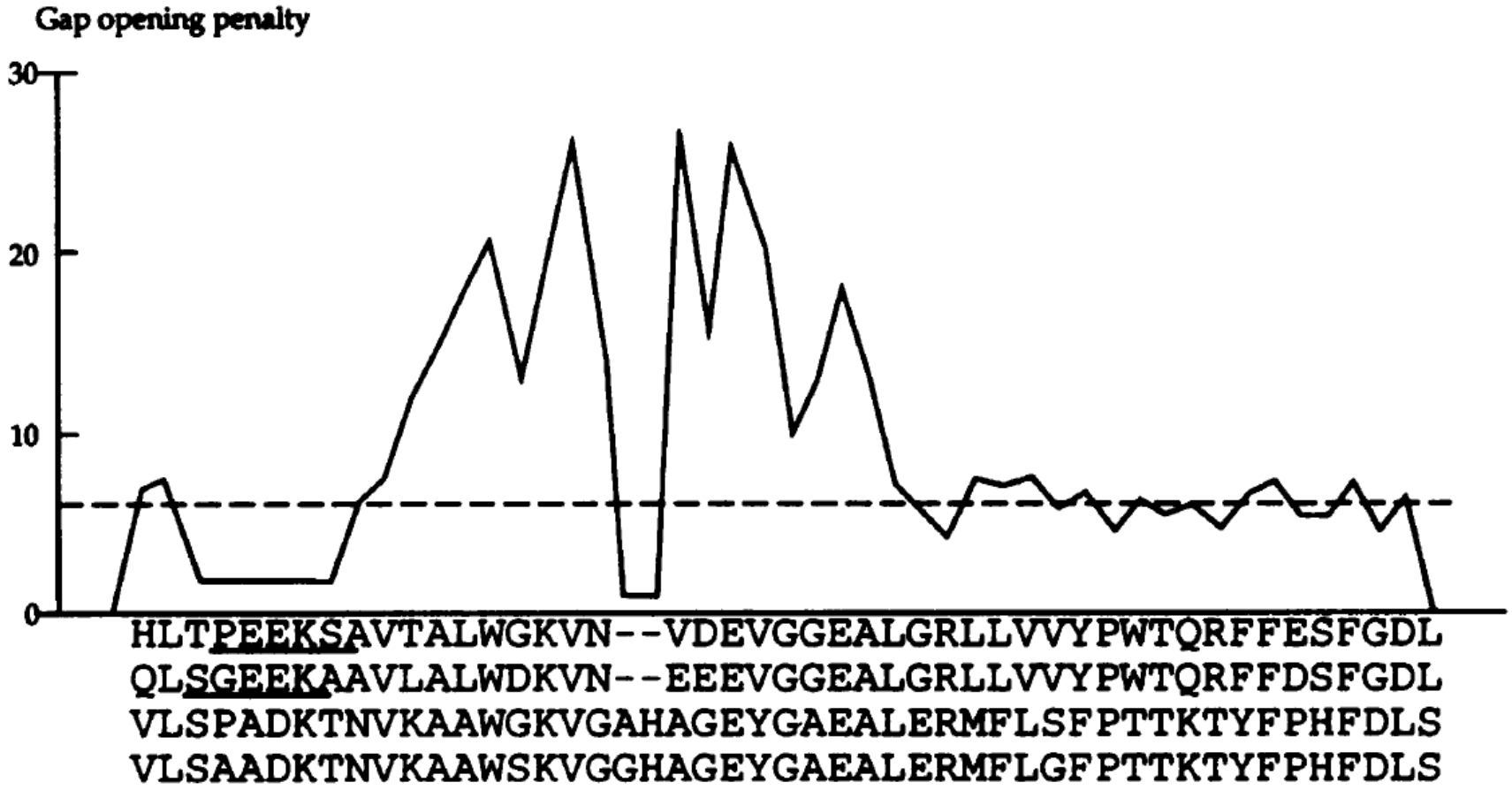
- N^3
- At each stage, we need to re-compute the distance between the newly formed profile and the other N (in the worst case) sequences/profiles (linear)
- At each stage, we also need to perform pairwise alignment (square)
- Taken together, each stage requires square time
- We have N stages because each stage we reduce the size of set of sequences/profiles by 1
- So N^3 in total

ClustalW: more sophisticated scoring function

- Firstly, individual weights are assigned to each sequence in a partial alignment in order to downweight near-duplicate sequences and up-weight the most divergent ones.
- Secondly, amino acid substitution matrices are varied at different alignment stages according to the divergence of the sequences to be aligned.
- Thirdly, residue-specific gap penalties and locally reduced gap penalties in hydrophilic regions encourage new gaps in potential loop regions rather than regular secondary structure.
- Fourthly, positions in early alignments where gaps have been opened receive locally reduced gap penalties to encourage the opening up of new gaps at these positions.

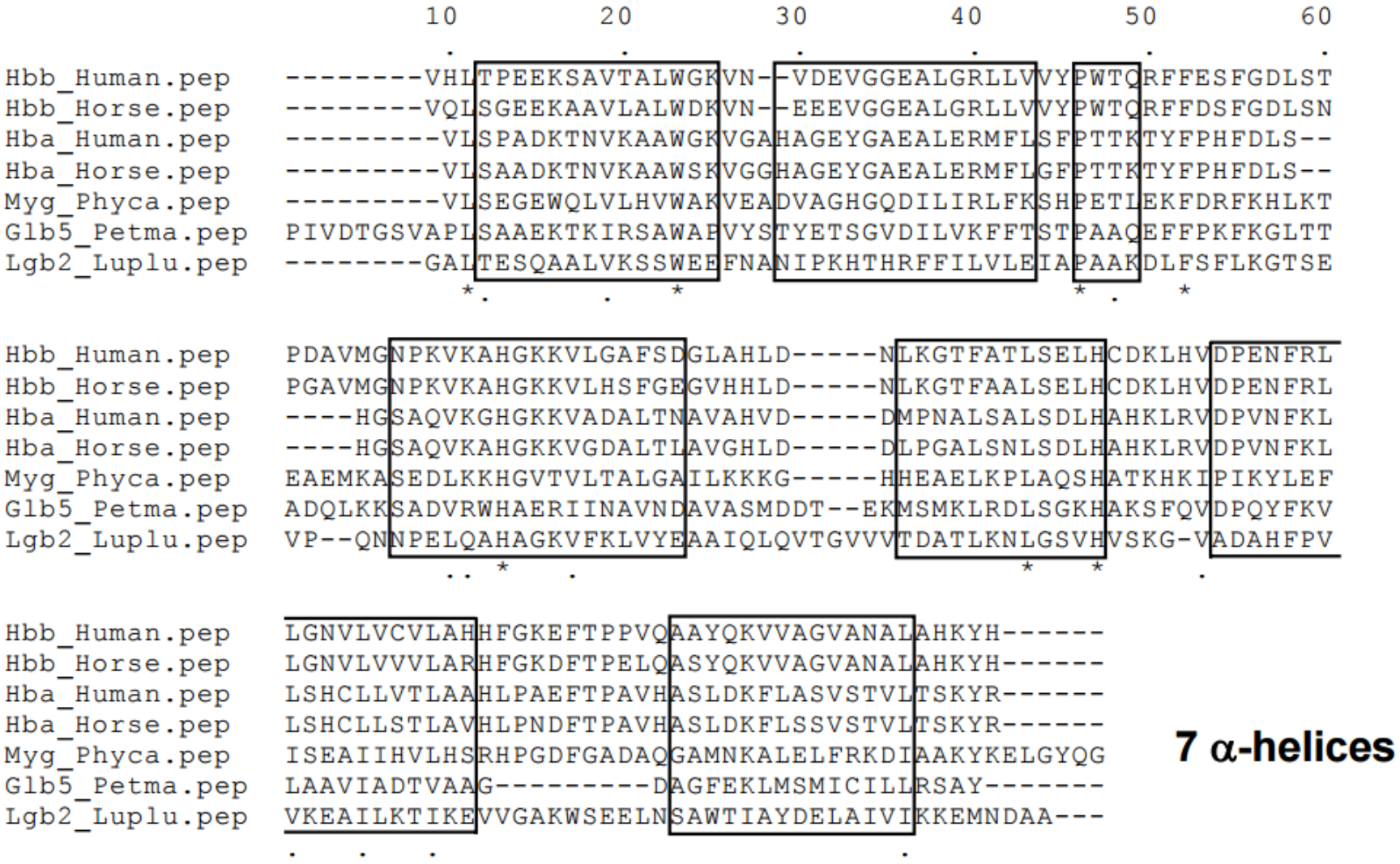
Thompson et al. 1994 *Nucleic Acids Research*

ClustalW: residue-dependent gap penalty



Thompson et al. 1994 *Nucleic Acids Research*

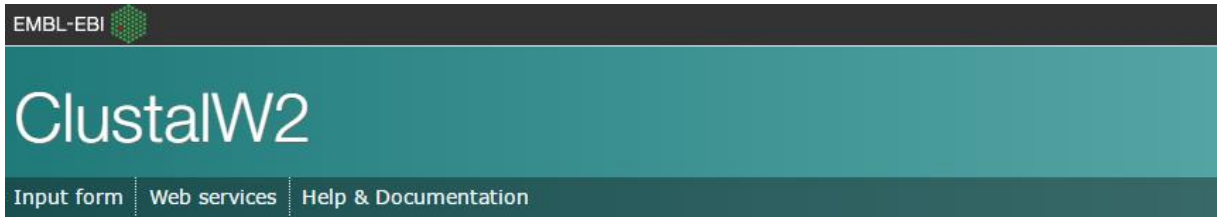
ClustalW: output



http://statweb.stanford.edu/~nzhang/345_web/sequence_slides3.pdf

ClustalW: multiple sequence alignment

- http://www.ebi.ac.uk/Tools/msa/clustalw2/



- http://www.genome.jp/tools/clustalw/