

# Pairwise Alignment

## The dynamic programming approach

1. Inspired by examples of alignments of biological sequences (with the basic operations match, insert and delete), define a suitable computational notion of pairwise string alignment.
2. Design a statistics set-up that is suitable to further define the notion of optimal alignment (score) of two sequences.

*Hints:*

- a. Define two statistics models: a *random* model (R), and a *match* model (M).
  - b. State certain independence assumptions you can make in order to conveniently compute the probability of a pairwise alignment  $(x, y)$  in each of the two models.
  - c. Given an alignment  $(x, y)$ , compare  $P_R(x, y)$  to  $P_M(x, y)$ .
3. Given two sequences  $x = x_1x_2 \dots x_n$  and  $y = y_1y_2 \dots y_m$ , how large is the number of all their possible alignments?  
What implications arise in what concerns the (further) design of alignment algorithms?
  4. Would dynamic programming (DP) be suitable for computing pairwise alignments? Yes/No? Why?
  5. Define the initialization and the recurrence relations for a DP algorithm for global alignment of two sequences.  
What is the biological/computational significance of the  $(i, j)$  element in the DP matrix?
  6. Propose other versions of pairwise alignment: local, local repeat, overlap, overlap repeat, suboptimal alignments.
  7. What is the complexity of the DP algorithms for pairwise alignment? Could we perform (e.g. global) pairwise alignment using linear space? What about sub-quadratic time?
  8. Could we define a FSA (finite state automaton) alternative to the DP algorithm for global alignment?
  9. What about designing heuristic algorithms for fast (not necessarily optimal) pairwise alignments?

◦ **Recommended readings:**

Richard Durbin et al. "Biological sequence analysis", 1998, Ch. 2  
Serafim Batzoglou, "The many faces of sequence alignment", 2005

## For your laboratory portfolio

1. Implement in a single program C all the four variants of the DP alignment algorithms presented in [Durbin et al, 1998] chapter 2.3: global, local, repeated matches, overlap matches.

Use command line arguments to specify:

- t*: the type of the alignment algorithm
- d* (and eventually -*e*) : the gap penalties
- s*: the name of a file containing the score/substitution matrix.  
Alternative: when general match and mismatch scores are used, provide them via the  $-M$  and respectively  $-m$  arguments.
- H*: display the usage of all command line arguments.

The two sequences will be introduced from the standard input.  
(Use input redirection (<) in the command line if you want the two sequences to be read from a file.)

Output both the optimal alignment score and the optimal alignment(s).

2. Implement in C the FSA for global alignment with affine scores, as given in [Durbin et al, 1998], figure 2.9, with the recurrence relations (2.16). Use command line arguments as above.
3. \* Implement in C the pair alignment DP algorithm with linear space, as described in [Durbin et al, 1998] chapter 2.6. Compute also the optimal alignment path.
4. \*\* Implement in C the block alignment algorithm presented in [Jones and Pevzner, 2004], chapter 7.3, including the *Four Russians* optimization.
5. \*\* Implement in C the  $O(n^2/\log n)$  algorithm for solving the LCS (Longest Common Substring) problem, as explained in [Jones and Pevzner, 2004], chapter 7.4 and 6.5.
6. \*\*\* Try to re-compute BLOSUM50 starting from the BLOCKS database.
7. Play with Sean Eddy's C program for computing the  $\lambda$  parameter used for finding a probabilistic interpretation arbitrary score matrices. (You will find on Sean Eddy's home page a link to the nice short paper describing BLOSUM-62, and the program program we addressed here. See also the comment on  $\lambda$  found in [Durbin et al, 1998] at page 85.)