Programming project

From ADN to formation of proteins : how to align sequences ?

Marie Albenque: *albenque@lix.polytechnique.fr*

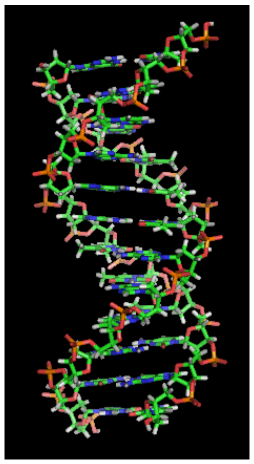
Java is the only programming language accepted for this project.

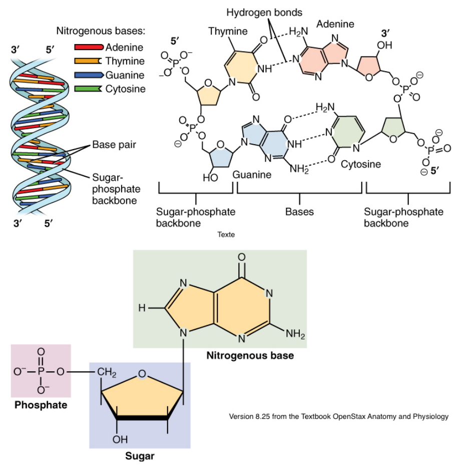
1 Biological context

This section is a very basic introduction to the structure of *Deoxyribonucleic Acid* or *DNA* and how proteins are formed from it. It is by no means exhaustive and you are more than welcome to look up for more information.

All the genetic information of living cells or organisms is stored in the DNA molecules. Watson and Crick established in 1953 (and earned the Nobel prize for it) that most DNA molecules consist of two biopolymer strands that form a double helix, see Figure 1(a).

The two DNA strands are formed by smaller units: the *nucleotides*. Each nucleotide is composed of a sugar (called deoxyribose, giving the "D" of DNA), a phosphate group and a nitrogen base, see Figure 1(b). Each nucleotide can be of 4 different types, that differ one from another only through their nitrogen base, which can either be of type A (for Adenine), T (for Thymine), G (for Guanine) or C for (Cytosine). All the genetic diversity is hence fully described by some (very long) novels using only those four letters. The two strands of DNA are mirror-images by the correspondence, where the image of A is T and the image of G is C.


(a) Double-helix structure of DNA



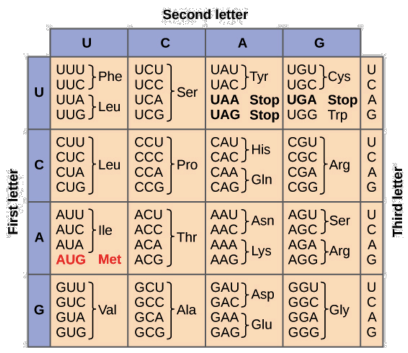
(b) Structure of nucleotides

Figure 1: Structure of DNA

There has been a huge effort in the recent years to sequence genomes and completely sequenced genomes include now humans, chimpanzees, mice, drosophiles, ... And scientists now need to know what to do with this amout of information. To illustrate the size of these data, the human genome is constituted of $3.4 * 10^9$ bases.

DNA of different species evolved from a common ancestor through mutations. Determining *homologous genes* between two species, that is portions of DNA that have the same ancestor sheds some light on the function of these genes. The number of mutations between homologous genes also allows to measure the distance between two species via the number of mutations between their DNA. Phylogenetic trees can then be built by relying on these distances.

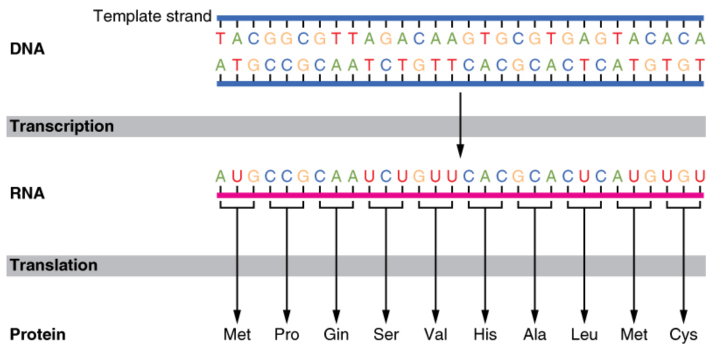
DNA expresses the information it contains through the formation of proteins. More precisely, the DNA is first *transcripted* (link to wikipedia page about Transcription: click !), so as to produce some RNA and the RNA is then *translated* (link to wikipedia page about Translation: click !) to form proteins. The structure of RNA is similar to the one of DNA, except that it is only stranded and instead of having A, T, G and C as possible bases, it has A, U, G and C. To form proteins, bases are read 3 by 3 (called *codon*) and each triple of bases correspond to an amino acid, according to the following table and mechanism.



Second letter

First letter	U	C	A	G	Third letter
U	UUU Phe	UCU Ser	UAU Tyr	UGU Cys	U
U	UUC Phe	UCC Ser	UAC Tyr	UGC Cys	C
U	UUA Leu	UCA Ser	**UAA Stop**	**UGA Stop**	A
U	UUG Leu	UCG Ser	**UAG Stop**	UGG Trp	G
C	CUU Leu	CCU Pro	CAU His	CGU Arg	U
C	CUC Leu	CCC Pro	CAC His	CGC Arg	C
C	CUA Leu	CCA Pro	CAA Gln	CGA Arg	A
C	CUG Leu	CCG Pro	CAG Gln	CGG Arg	G
A	AUU Ile	ACU Thr	AAU Asn	AGU Ser	U
A	AUC Ile	ACC Thr	AAC Asn	AGC Ser	C
A	AUA Ile	ACA Thr	AAA Lys	AGA Arg	A
A	AUG Met	ACG Thr	AAG Lys	AGG Arg	G
G	GUU Val	GCU Ala	GAU Asp	GGU Gly	U
G	GUC Val	GCC Ala	GAC Asp	GGC Gly	C
G	GUA Val	GCA Ala	GAA Glu	GGA Gly	A
G	GUG Val	GCG Ala	GAG Glu	GGG Gly	G

(a) Genetic Code


Template strand
DNA
TACGGCGTTAGACAAGTGCGTGAGTACACA
ATGCCGCAATCTGTTCACGCACTCATGTGT
Transcription
RNA
AUGCCGCAAUCUGUUCACGCACUCAUGUGU
Translation
Protein
Met Pro Gln Ser Val His Ala Leu Met Cys

(b) Traduction and translation

Instead of focusing solely on DNA, it makes in fact more sense to look at the proteins they encode, in particular for the following reasons:

- Only 3% of the DNA actually encode some proteins,
- A mutation that replaces for instance UUU by UUC has no influence of the protein synthesized by this part of the DNA and should not taken into account when measuring the distances between two species,
- Because there are 22 different amino acids and only 4 bases, the probability for two random amino acids to be equal is much lower that the probabilty for two random nucleotide bases.

The main objective of this project is to implement some algorithms, which measure the evolutionnary distance between two sequences of DNA or amino acids (since there are 22 different amino acids, they can be encoded each by one different letter, see here for the usual encoding).

Unless mentioned otherwise, the input of the algorithms is a text file with a first sequence in the first line and a second sequence in the second line. Your code should be able to deal with such input files.

2 Longest common subsequences and edition distance

2.1 Length of the longest common subsequence

Let $s = s_1 \dots s_n$ and $t = t_1 \dots t_m$ be two sequences of letters (representing either DNA, proteins or whatever you like really). The sequence $u = u_1 \dots u_k$ is a common subsequence of s and t if there exist $1 \leq i_1 <$

$\ldots < i_k \leq n$ and $1 \leq j_1 < \ldots < j_k \leq m$ such that for all $\ell \in \{1, \ldots k\}$, $u_\ell = s_{i_\ell} = t_{j_\ell}$. The longest common subsequence problem takes two sequences as input and returns the length of one longest common subsequence between the two initial sequences.

Example: One longest common subsequence of CACTAAGCATCAGC and ATAGGGCAATCT is ATAGCAAC and has length 8.

Task 1. (theoretical) Find a naive algorithm to compute one longest common subsequence, compute its complexity and see how to improve the complexity by using dynamic programming.

Task 2. Write an algorithm based on dynamic programming which computes one longest common subsequence between two sequences given in an input file. Be careful of keeping the heap size under control !

You can find a couple of test files here (click !).

2.2 Distance edition and alignment of sequences

Let s and t be as above. The editing distance between s and t is the minimal number of operations to change s into t, where the possible operation are : inserting a letter ('I'), deleting a letter ('D') or transforming a letter ('T').

For instance to go from CACTAAGCATCAGC to ATAGGGCAATCT, we start by deleting the two first C, transform the first A into G, then insert a G between G and C, ...

An equivalent formulation is to say that we insert hyphens in s and t in such that the number of hyphens plus the numbers of letters for which s and t differ is minimal. Describe this equivalence in your report.

Example: The editing distance between CACTAAGCATCAGC and ATAGGGCAATCT is 8 and the corresponding optimal alignment is given by:

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

Task 3. Write an algorithm based on dynamic programming which computes and displays (nicely) one optimal alignment between two sequences given in an input file.

3 Refinements on alignment

3.1 Substitution matrices

In fact, we can try to give more importance to the biological data. Some mutations are indeed more likely to occur than some others and some amino acids are also more frequent than others. Some substitution matrices have been computed, these substitution matrices give a score to each couple of amino acids that can appear in an alignment (it includes in particular couples with one hyphen and one amino acid and couples with the same amino acids to take into account that if two rare amino acids happen to coincide then the score of this alignment should be high). The total score of the alignment is the sum of the score of each position. The difference between the substitution matrices come from the precise biological setting. We will focus on the "Blosum 50" matrix in this project.

The previous section is a particular case of this setting where the substitution matrix is the identity and the score corresponds to the length of the longest common subsequence.

You can find here (click !) a java file that gives the value of the Blosum50 matrix for each pair of amino acid (where each amino acid is encoded by a letter as indicated above). More precisely this class features a method `public static float getScore(char c, char d)`, which returns the score of the alignment of `c` and `d`.

Task 4. Write an algorithm based the previous one which computes and displays (nicely) one optimal alignment between two sequences of amino acids using the Blosum50 matrix.

3.2 Affine penalty

Insertions or deletions are usually not single events and it is much more likely to insert or delete two letters at the same spot than to insert or delete two letters at two different spots. We will introduce a opening gap penalty for each new insertion and deletion event and an increasing gap penalty for each following insertion or deletion that happens at the same spot.

Similarly, the sequences do not have to start and to end at the same spots so that inserting gaps at the beginning or at the end have no cost.

Task 5. Adapt your previous algorithm so that it takes two additional parameters for opening gap penalty and the increasing gap penalty and computes one optimal alignment in this new setting.

You can fix the opening gap penalty at 10 and the increasing gap penalty at 1 in your tests.

3.3 Local Alignment

One of the main difficulty when working with sequences of amino acids is to determine which part corresponds for a protein. Instead of trying to align fully some sequences, it make therefore sometimes more sense to try to align only some parts of the sequences.

More precisely, for two strings s and t, we denote $\mathrm{Opt}(s,t)$ the score of the maximal alignment between s and t computed in Task 5. We define then $\mathrm{OptLoc}(s,t)$ to be the maximal score of a local alignment that is:

$$\mathrm{OptLoc}(s,t) := \max_{i,i',j,j'} \{\mathrm{Opt}(s[i,j], t[i',j'])\},$$

where $s[i,j]$ denote the substring of s located between the i-th and j-th letter.

Task 6. Write an algorithm that computes one optimal local alignment based on the Blosum50 matrix and with affine gap penalty.

4 Basic Local Alignment Search Tool

The alignment algorithms we described above are exact in the sense that they return the alignment (or the local alignment) with the best score. Unfortunately, this is at the cost of a large complexity (explain it in your report !). To lower the complexity, some other approaches based on heuristics (and not exact) have been developed. The BLAST (Basic Local Alignment Search Tool) is one of them.

Biologically speaking, if a new gene is identified in the drosophila, we may wonder if a similar gene exists in the human genome. So we are going to scroll through the human genome (very long) to try to find (relatively short) sequences close to the one identified in the drosophila.

The basic idea is the following (you can find much more detailed information online, for instance here click !). Let g be the string representing the new gene (e.g. identified in the drosophila) and let t be the string representing the string in which you hope to identify some similar sequences (e.g. the human genome).

Fix k (typically k is 11 for amino acids) and consider the set $\mathcal{W}_g$ all the subwords of g of length k. For instance if $g = ADCRGHC$ and $k = 4$, then $\mathcal{W}_g = \{ADCR, DCRG, CRGH, RGHC\}$. For each word w of $\mathcal{W}_g$, denote s_w the score of its alignment with itself; for instance $s_{\mathrm{ADCR}} = 5 + 8 + 13 + 7 = 33$. Now fix $th > 0$ (think of $th = 0.9$), and for each word $w \in \mathcal{W}_g$, consider all the words of length k, whose alignment with w has a score greater than $th \cdot s_w$. Denote $\mathcal{S}_g$ (for "Seeds"), the set of these words.

Task 7. Write a program that takes g, t and th as parameters and returns all the indices that correspond to beginning of perfect matches between an element of $\mathcal{S}_g$ and a subword of t.

For instance if $\mathcal{S}_g = \{ADCR, DCRG, CRGH, RGHC\}$ and $t = BDADCRGNRADACRGHC\}$, your programm should return $2, 3, 13, 14$.

Try to optimize the complexity by choosing in a appropriate manner the way you store elements of $\mathcal{S}_g$. Keyword trees is a good option here. Describe your choice in your report.

Now, for each perfect match between $\mathcal{S}_g$ and a subword of t, we extend the matching in both directions as long as the corresponding score does not decrease. Returning to the previous example, consider the index 3, which corresponds to the following local alignment:

B	D	A	D	C	R	G	N	R	A	D	A	C	R	G	H	C
		A	D	C	R	G	H	C								

Since the letter at position 2 in t is A and the letter before D in g is also A, we can extend the matching to the left. To the right, the letter at position 7 in t is N and the letter after G is H, the score of aligning N and H in the Blosum50 matrix is 1, we can then extend the alignment to the right. But the score of aligning R and C is -4 so we stop the alignment here, and get the following local alignment of 6 consecutive amino acids :

B	D	A	D	C	R	G	N	R	A	D	A	C	R	G	H	C
		A	D	C	R	G	H	C								

A second threshold th_ℓ is fixed (think of $th_\ell = 0.1$ (even if it will be significantly smaller in applications)) and if the score of the local alignment is more than $th_\ell \cdot$ score of aligning g with itself, we store this aligment (by storing the index of its first letter, its length and score for instance), otherwise we discard it.

Task 8. Write a program that takes g, t, th and th_ℓ as parameters and returns all the local alignments with sufficiently high scores.

5 Possible extension: Protein Folding in the Hydrophobic-Hydrophilic Model

If you want to become an expert in bioinformatics or like a challenge, you can try the following possible extension !

Protein Folding (i.e. the way that proteins arrange in 3d) is one of the main challenge today. The hydrophobic-polar protein folding model is a toy-model to determine how proteins fold in 2-dimension. It originates in [2] in 1985 and takes only into account that all amino acid types can be classified as either hydrophobic (H) or polar (P). A protein is represented by a word on the two letters H or P. A folding is a self-avoiding walk in the square lattice and the score of a folding is the number of pairs H-H that are adjacent in the grid but not in the protein, see example below.

Task 9. Write a program that takes a string on $\{P, H\}$ as input and returns one optimal folding.

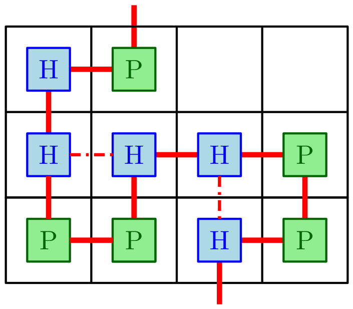


Figure 2: A possible folding for the protein PHHPPHHPPH, with score 2.

References

[1] Aho, Alfred V., and Margaret J. Corasick. *Efficient string matching: an aid to bibliographic search.* Communications of the ACM 18.6 (1975): 333-340.

[2] Dill, Ken A. *Theory for the folding and stability of globular proteins.* Biochemistry 24.6 (1985): 1501-1509.