Algorithms for Bioinformatics: Project 2
Definition of Python Class for Sequence Alignment

Miguel Sozinho Ramalho, UP201403027

March 6^{th}, 2019

1 Introduction

The current report describes a practical approach for extending the Python3 library developed in Project 1 to be capable of performing alignment on the defined biological sequences, namely local and global alignment, using the NeedlemanWunsch and SmithWaterman algorithms, respectively. It is considered paramount that, in the case of multiple optimal solutions, all of them are discovered. The new version also contains relevant methods that can be used to better understand how the alignment works, such as visualisation for dot plots and other metrics, such as Hamming distance. Moreover, a large effort was made to allow for reusable, tested, documented and efficient code. Contextualising, this project belongs to the course of Algorithms in Bioinformatics.

2 Implementation

The main challenge of this project was the simple and correct implementation of the Dynamic Programming algorithms and also how to refactor the old module to achieve this goal.

From the first project, only one new class was introduced: **Matrix**, this was an isolation of numerical matrices with the functionality required by the remaining algorithms and classes (and others used during the development), namely: sum, max, min, square, last, add_val, mul_val, apply, display, set_col, __len__, __setitem__, __getitem__, __iter__, __str__ (this last one was particulary challenging as it was required for a smart pretty printing functionality). The precise goal of each function is described in the documentation of the class.

In order to guarantee a good quality of the tool, it was required to write unit tests for all the developed methods (see Figure 1 for the 100% coverage) and to document every implemented feature.

Coverage report: 100%

Module ↓	*statements*	*missing*	*excluded*	*branches*	*partial*	*coverage*
bioseq\bioseq.py	131	0	3	66	0	100%
bioseq\dnaseq.py	17	0	0	2	0	100%
bioseq\matrix.py	60	0	11	24	0	100%
bioseq\proteinseq.py	11	0	0	0	0	100%
bioseq\rnaseq.py	29	0	0	16	0	100%
bioseq\utils.py	25	0	0	10	0	100%
Total	**273**	**0**	**14**	**118**	**0**	**100%**

Figure 1: Automatically generated test coverage report for the developed Python module

3 Results

All of the required functionality was implemented and also some others that were inspired by the work done in the practical lectures, namely:

- hamming distance calculator (hamming_distance)
- create a dotplot between two sequences (dot_plot)
- display the dot_plot in a plot if matplotlib is installed (display_dot_plot)
- score of aligning two sequences, given a gap and substitution matrix (score_seq)
- affine gap score, using the gap penalty and the keep gap penalty (score_affine_gap)
- NeedlemanWunsch (global_align_multiple_solutions)
- given two sequences and their global Score and Traceback matrices, return all the optimal alignments (recover_global_align_multiple_solutions)
- SmithWaterman (local_align_multiple_solutions)
- given two sequences and their local Score and Traceback matrices, return all the local alignments (recover_local_align_multiple_solutions)
- for every combination return Score and Traceback matrices, global alignment (compare_pairwise_global_align)
- for every combination return Score and Traceback matrices, local alignment (compare_pairwise_local_align)

Some other interesting points:

- optimisation of the calculation of symmetric matrices
- the recovery algorithm is also very scalable as it is not recursive (as the original implementation) and takes a total of three very eloquent lines
- tests, documentation, code structure, Continuous Integration with `https://travis-ci.com/` and push notifications of pipeline success with `https://www.pushbullet.com/` (see the *pushbullet_notify.py*)

The documentation can be seen by opening the file *docs/_build/html/index.html* in an external browser or by typing **help(bioseq.moduleName)** on the Python console for the desired module.

4 Conclusions

Looking back at all the progress made with this project, the author concludes there was a lot learned and also that most concepts became clearer when they had to be organised so meticulously, as in the current tool. It is his belief that the work produced is of sufficient quality to be used in future projects and also to be extended as more relevant bioinformatics and dynamic programming techniques are learned.

5 References

(No paper references required and all other mentions like Python, FASTA, DNA, RNA, Aminoacid, OOP, NeedlemanWunsch, SmithWaterman, Dynamic Programming and others are considered base knowledge prior to reading this document)