Andreas, Christian, Lærke

Introduction

For this project, we ended up completing the assigned tasks/programs of sp_exact_3 and sp_approx. Both programs work as expected, and there does not seem to be any issues with either of our programs. Both programs were able to solve the assigned questions to a satisfactory level in an expected amount of time for the inputs given. In general, the project has gone ahead smoothly and without too many hiccups. The biggest problems we ran into were problems with text encoding, but these were solved without too many problems.

Methods

Our programs do not differ significantly from what we talked about in class. Our programs are based on the pseudo-code provided and discussed in the lectures, with small tweaks to make them comply with the assignment.

sp_exact_3

For this program we decided to use one system argument (sys.arg in python), which must be a fasta file containing the sequences to be used. The program will basically take the first 3 sequences in the fasta file. If you want to run over sequences other than the first three in the fasta file, this can be changed in the program itself. Likewise, the gap cost and the score matrix can be changed in the program if desired.
Which sequences to run over can be chosen at the bottom of the sp_exact_3.py file. This is done by changing the index values of “x” in the “matrix” variable.
The gap cost can be changed at the bottom of the file as well by changing the value of the “gap” variable. The score matrix can be found at the top of the document in the “cost” function under the variable name “score”.

Example of program call with the length 10 fasta file:
python sp_exact_3.py testseqs_10_3.fasta

sp_approx

The sp_approx program is called with 3 options, score, file and gap cost, using system arguments (sys.arg in python). sys arg 1 expects a .txt file with a cost matrix as used in project 2 (phylib like format), which is included in the zip file (“score.txt”). Sys arg 2 expects a fasta format file of the sequences to align (any number of sequences more than one). Sys arg 3 expects an integer corresponding to the gap cost to use for the alignment. the output is a multiple sequence alignment in fasta format, and is written to std out. To get the score of the multiple alignment we saved the output of sp_approx to a .txt file, and ran the provided “msa_sp_score_3k.py” program to get the score of the alignment (we also included this in the zip file).

Example of program call with the length 10 fasta file:
python sp_approx score.txt testseqs_10_3.fasta 5 > res.txt

Andreas, Christian, Lærke

python msa_sp_score_3k.py res.txt

Dependencies:

We figured the easiest way to get our pairwise alignments was to use the function we made for project 2, this was implemented by importing the function from the old project file. as such that file is also included in the zip archive.
To import fasta files, we used the code provided (“fasta.py”) which is also included in the zip archive.
Furthermore we use the numpy, cmath and sys imports for our code.

Testing/validation

To verify the correctness of our programs, we used the provided testdata_short.txt & testdata_long.txt sequences. For the sp_exact_3 both scores calculated were exact matches, and for the sp_approx the results were fairly close, at short: 212 and long: 1553 compared to the given scores short: 198, long: 1482. Furthermore the result from experiment 3 also shows that our approximation algorithm follows closely the exact.

Experiments

What is the score of an optimal alignment of the first 3 sequences in brca1-testseqs.fasta (i.e. brca1_bos_taurus, brca1_canis_lupus and brca1_gallus_gallus) as computed by your program sp_exact_3?

Using sp_exact_3 on the three first sequences in “brca1-testseqs.fasta”, we got a score of 790 for an optimal alignment.

How does an optimal alignment look like?

The optimal alignment looks like this:
>brca1_bos_taurus
'ATGGATTTATCTGCGGATCATGTTGAAGAAGTACAAAATGTCCTCAATGCTATGCA-GAAA
ATCTTAG--AGTGTCCAAT-ATGTCTGGAGTTGATCAAAGAG-CCT-GTCTCTACAAAGTGTG
A-CCA-CA-TATTTTGCAAATTTTG-TATGCTGAA-AC-TTCTCAACCA-GAAGAAAGGGCCTT
CACAATGTCC--TTTGTGTAAGAATGA-',
>brca1_canis_lupus
'ATGGATTTATCTGCGGATCGTGTTGAAGAAGTACAAAATGTTCTTAATGCTATGCA-GAAA
ATCTTAG--AGTGTCCAAT-ATGTCTGGAGTTGATCAAAGAG-CCT-GTTTCTACAAAGTGTG
A-TCA-CA-TATTTTGCAAATTTTG-TATGCTGAA-AC-TTCTCAACCA-GAGGAAGGGGCCTT
CACAGTGTCC--TTTGTGTAAGAACGA-',
>brca1_gallus_gallus
'GCGAA---ATGTA-ACA-CG-GTAGAGGTGAT-CGGGGTG-CGTT-ATAC-GTGCGTGGTGAC
CTCGGTCGGTGT-TGACGGTGCCTGGGGTTCCTCAGAGTGTTTTGGGGTCTGAAGGAT
G-GACTTGTCAGTG-ATTGCCATTGGAGACGTGCAAAATGTGCTTTCAGCCATGCAGAA-G
AA-CTT-GGAGTGTCCAGTCTGTTTAGATGTGAT'

Andreas, Christian, Lærke

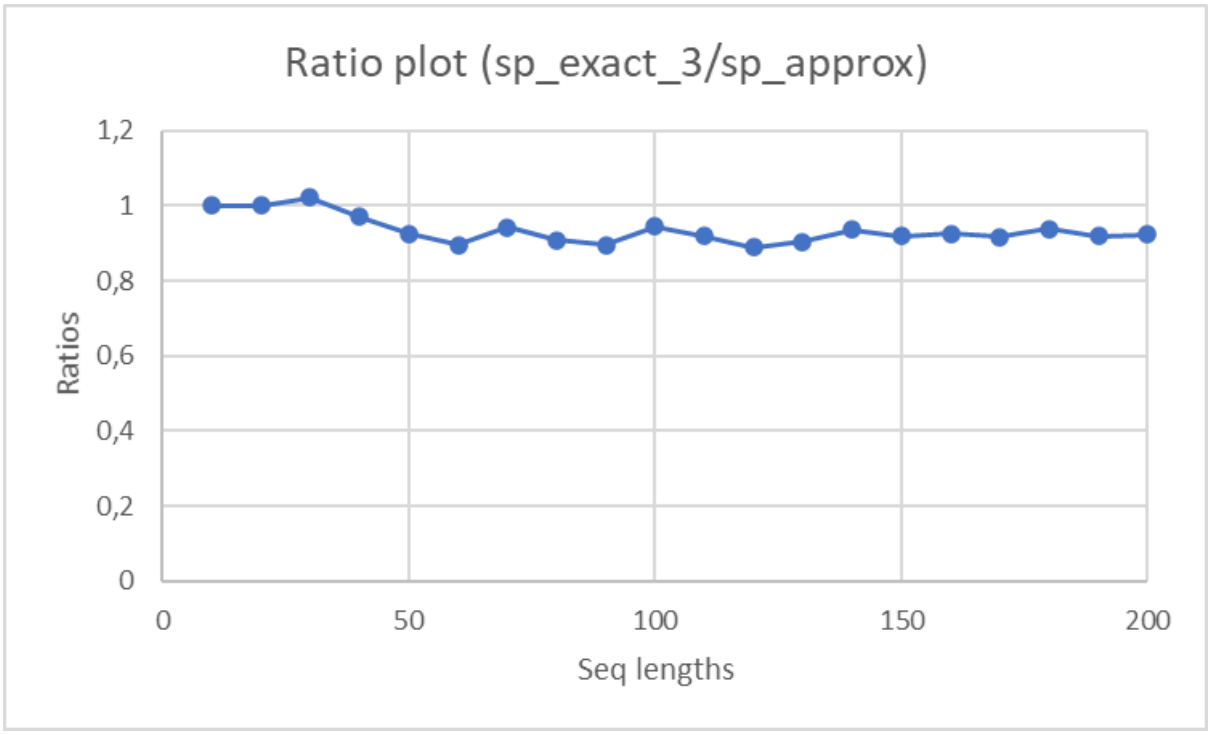
What is the score of the alignment of the first 5 sequences in brca1-testseqs.fasta (i.e. brca1_bos_taurus, brca1_canis_lupus, brca1_gallus_gallus, brca1_homo_sapiens, and brca1_macaca_mulatta) as computed by your program sp_approx?

Using sp_approx on the five first sequences in “brca1-testseqs.fasta”, we got a score of 3273 for an optimal alignment.

Which of the 5 sequences is chosen as the 'center string'?

The first sequence, “brca1_bos_taurus”, is chosen as the ‘center string’ by our program sp_approx.

Make an experiment comparing the scores of the alignments computed by sp_exact_3 and sp_approx that validates that the approximation ratio of sp_approx is 2(k-1)/k for k sequences. i.e 4/3 for three sequences. You should use the testdata in testseqs.zip that contains 20 fasta files (testseqs_10_3.fasta, testseqs_20_3.fasta, ..., testseqs_200_3.fasta) each containing 3 sequences of lengths 10, 20, ..., 200. For each triplet of sequences (i.e. each fasta file), you should compute the optimal score of an MSA using sp_exact_3 and the score of the alignment produced by sp_approx. Make a graph in which you plot the ratio of the computed scores for each sequence length. Comment on what you observe.



As expected, the sp_exact_3 algorithm performs better than the sp_approx algorithm in all cases except for sequence length 30, which, for some weird reason, has sp_aprrox getting the best score, though admittedly by a very small margin, so this might just be down to a small miscalculation or randomness.