Bioinformatics

First semester, 2019/2020

Lab 2

Group 10:
André Miranda, 83811
Diogo Ramalho, 86407
Manuel Madeira, 83836
Rafael Correia, 83828

Group I

Matrix from dynamic programming:

			S2						
			G	G	A	T	C	C
		0	1	2	3	4	5	6
	0	0	-4	-8	-12	-16	-20	-24
S1	G	1	-4	3	-1	-5	-9	-13	-17
	G	2	-8	-1	6	2	-2	-6	-10
	C	3	-12	-5	2	4	0	1	-3
	C	4	-16	-9	-2	0	2	3	4
	G	5	-20	-13	-6	-4	-2	0	1

Alignment	Score
GGATCC GG-CCG	1
GGATCC GGC-CG	

Group 2

				S2			
			I	I	W	P	I
		0	1	2	3	4	5
	0	0	0	0	0	0	0
S1	W 1	0	0	0	15	11	7
	P 2	0	0	0	11	25	21
	I 3	0	5	5	7	21	30
	W 4	0	1	2	20	17	26
	P 5	0	0	0	16	30	26
	C 6	0	0	0	12	26	28

Alignment	Score
WPIWPC ||| IIWPI	30
WPIWPC ||| IIWPI	

Group 3

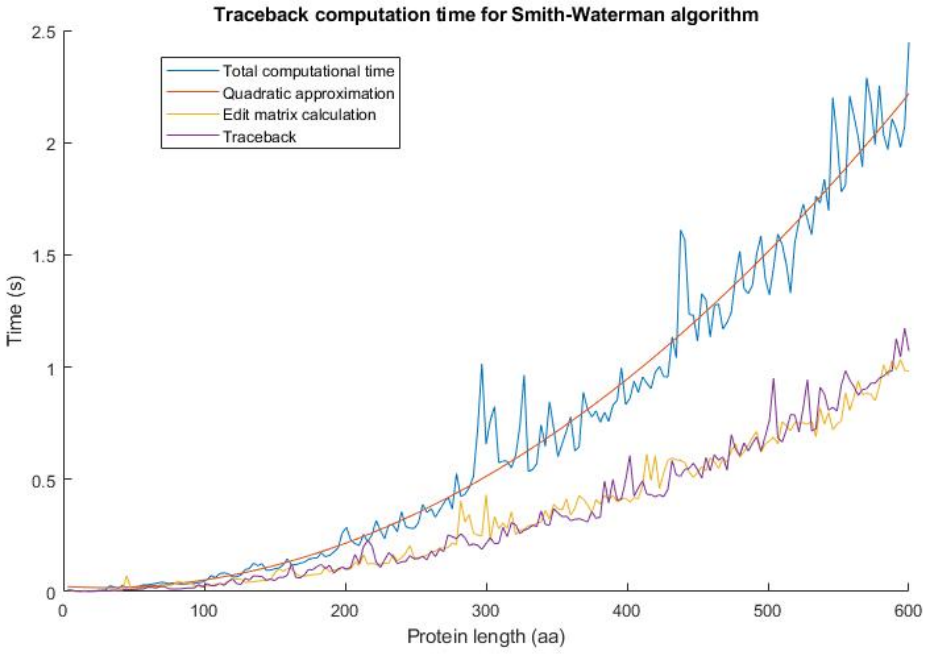
(code attached to .zip)

Two optimal alignments are possible, both with a score of 30 (same result as in **Group 2**):

```
WPIWPC        WPIWPC
|||            |||
IIWPI          IIWPI
```

<u>Performance results</u>

Assuming both strings are the same size (n) we can plot the performance by running the algorithm on different sized proteins. The observed behaviour is quadratic as expected from the algorithm analysis.

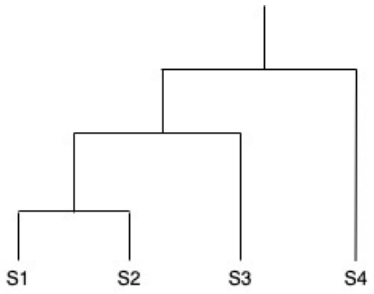


Group 4

All the pairwise global alignments are given in the following table, along with the corresponding score:

Pair		Optimal global alignment(s)	Score
S1-S2	AACGTC AGCGCC	AACGTC AGCGCC	6
S1-S3	AACGTC CCCGT	AACGTC CCCGT-	1
S1-S4	AACGTC ACAT	AACGTC A-CAT- or AACGTC -ACAT-	-1
S2-S3	AGCGCC CCCGT	AGCGCC CCCGT- or AGCGCC CCCG-T	-2
S2-S4	AGCGCC ACAT	AGCGCC A-CAT- or AGCGCC A-CA-T or AGCGCC A-C-AT	-4
S3-S4	CCCGT ACAT	CCCGT ACA-T or CCCGT AC-AT or CCCGT A-CAT or CCCGT -ACAT	-1

Guide Tree



Distance Matrix

	S1	S2	S3	S4
S1		6	1	-1
S2			-2	-4
S3				-1
S4				

The distance matrix diagonal is not filled because it doesn´t make sense to align the sequences with themselves and also because they would have the higher scores in the matrix (S1-S1 = 12; S2-S2 = 12; S3-S3 = 10; S4-S4 = 8).

According to the pairwise alignment, we first add to the alignment matrix the pair that has the greatest score. That is the pair AACGTC | AGCGCC with a score of 6. Hence, we have so far:

AACGTC
AGCGCC

Then, we search for pairwise alignments with sequences that were already added in the alignment matrix that have the next highest score. Now, that is the pair AACGTC | CCCGT with score 1. We add CCCGT to the matrix with the alignment from the table above, after applying the Needleman-Wunsch algorithm to the matrix alignment and the sequence to be added (S3):

			S3				
			C	C	C	G	T
S1	S2	0	-6	-12	-18	-24	-30
A	A	-6	-2	-8	-14	-20	-26
A	G	-12	-8	-4	-10	-13	-19
C	C	-18	-8	-4	0	-6	-12
G	G	-24	-14	-10	-6	4	-2
T	C	-30	-20	-13	-9	-2	5
C	C	-36	-26	-16	-9	-8	-1

AACGTC
AGCGCC
CCCGT-

Next, both AACGTC | ACAT and CCCGT | ACAT pairs have -1 score which is the next highest score. We add the first pair by applying the Needleman-Wunsch algorithm to the matrix alignment (S1,S2,S3) and the sequence to be added (S4) and since the second pair is already present in the matrix the tie problem is resolved. However, if that was not the case, we could choose randomly since this algorithm does not search for the optimal solution but only for a good enough solution of the multiple alignment problem.

				S4			
				A	C	A	T
S1	S2	S3	0	-9	-18	-27	-36
A	A	C	-9	3	-6	-15	-24
A	G	C	-18	-6	3	-6	-15
C	C	C	-27	-15	0	0	-9
G	G	G	-36	-24	-9	-3	-3
T	C	T	-45	-33	-18	-12	0
C	C	-	-51	-42	-27	-21	-9

AACGTC
AGCGCC
CCCGT-
A-CAT-

The final score of this particular solution is given by:

$$score_{SP} = \sum_{1 \le p < q \le k} s(A_p, A_q) = \sum_{1 \le p < q \le k} \sum_{i=1}^{N} s(a_{pi}, a_{qi})$$

Where A_p is the sequence with index p and a_{pi} is the i^{th} character of sequence A_p.

$$s(AACGTC, AGCGCC) = 2 + (-1) + 2 + 2 + (-1) + 2 = 6$$
$$s(AACGTC, CCCGT-) = (-1) + (-1) + 2 + 2 + 2 + (-3) = 1$$
$$s(AACGTC, A-CAT-) = 2 + (-3) + 2 + (-1) + 2 + (-3) = -1$$
$$s(AGCGCC, CCCGT-) = (-1) + (-1) + 2 + 2 + (-1) + (-3) = -2$$
$$s(AGCGCC, A-CAT-) = 2 + (-3) + 2 + (-1) + (-1) + (-3) = -4$$
$$s(CCCGT-, A-CAT-) = (-1) + (-3) + 2 + (-1) + 2 + 0 = -1$$

And so,

$$score_{SP} = 6 + 1 - 1 - 2 - 4 - 1 = -1$$

Bonus:

In order to confirm the reasonability of the heuristic used in the MSA referred in class, we developed a version where the assumptions are less restrictive. Firstly, we calculate a distance matrix as in the MSA class version. However, the results achieved for each sequence individually might be wrong, since the initial distance matrix may not be updated accordingly to the already aligned sequences. Consequently, in order to get along with dynamics, we developed an iterative version, where in each iteration we update the distance matrix by replacing the previously separated sequences with the already aligned ones. The algorithm ends when all sequences are aligned. Nevertheless, we should refer that in our algorithm, we still use the max value of the distance matrix to choose which sequences/alignments to align next, but this rises the possibility of having two optimal alignments for an iteration. And so, in order to solve this, we calculate the distance matrixes between each of the new optimal alignments and the other sequences/alignments and we choose the one with a higher mean of all entries. This criterion was chosen since our objective, in the end, is to maximize the homogeneity among sequences. We also highlight the code we developed for a general implementation Needleman-Wunsch algorithm, which is able to receive as input individual sequences and alignments.