NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

School of Science

Information Technologies in Medicine and Biology

Direction: *Bioinformatics*

Algorithms in Structural Bioinformatics

Postgraduate Student: *Begetis Nikolaos*

Professor: *Emiris Ioannis*

Deadline Date: *26/03/2013*

Assignment 1

1(a).

We are assigned to find the global sequence alignment for the two protein sequences:

x = ASRFALFF, and y = ASIRVVFALF

To do this we will use the Needleman-Wunsch algorithm. Figure 1 shows the filled-in table and the backtrack path for the optimal solution. We set a score of +2 for matches, 0 for mutations and -1 for gap insertion.

Needleman Wunsch	*	A	S	I	R	V	V	F	A	L	F
*	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
A	-1	2	1	0	-1	-2	-3	-4	-5	-6	-7
S	-2	1	4	3	2	1	0	-1	-2	-3	-4
R	-3	0	3	2	5	4	3	2	1	0	-1
F	-4	-1	2	1	4	5	4	5	4	3	2
A	-5	-2	1	0	3	4	5	4	7	6	5
L	-6	-3	0	-1	2	3	4	5	6	9	8
F	-7	-4	-1	-2	1	2	3	6	5	8	11
F	-8	-5	-2	-3	0	1	2	5	6	7	10

Figure 1: Global Sequence Alignment using Needleman-Wunsch algorithm

It is worthy to mention that on the **topleft** corner of every table cell there are set the three possible scores given from the three possible ancestors of each cell, respectively. The best of the scores are highlighted in red color. These red colored possible scores also indicate the

only pointers for the path to be followed when the dynamic algorithm is backtracking to find the global sequence alignment. In the case of equal possible scores, the dynamic algorithm selects to follow the path through the cell that has the biggest given score. Such a case, in our alignment can be found in the last x=F → y=F cell. Finally, as it is obvious from Figure 1, the backtrack path for the optimal solution is highlighted in pale pink color. Given this backtrack path, we can infer that the optimal match including the inserted gaps is as follows:

y = A S I R V V F A L - F
| | | | | | | |
x = A S - R - - F A L F F

and as a result of the optimal match, the score gathered is 2+2-1+2-1-1+2+2+2+2-1 = **10**.

We also implemented a biojava application using the resource packages of the Needleman-Wunsch algorithm. In this way we confirmed the output results. The biojava implementation is given with the deliverable .zip package.

1(b).

In the second part of the first exercise, this time, we are assigned to find the local sequence alignment for the two protein sequences:

x = ASRFALFF, and y = ASIRVVFALF

To do this we will use the Smith-Waterman algorithm. Figure 2 shows the filled-in table and the backtrack path for the optimal solution. We set a score of +2 for matches, -1 for mutations and -1 for gap insertion.

Smith Waterman	*	A	S	I	R	V	V	F	A	L	F
*	0	0	0	0	0	0	0	0	0	0	0
A	0	2	1	0	0	0	0	0	2	1	0
S	0	1	4	3	2	1	0	0	1	0	0
R	0	0	3	3	5	4	3	2	1	0	0
F	0	0	2	2	4	4	3	5	4	3	2
A	0	2	1	1	3	3	3	4	7	6	5
L	0	1	1	0	2	2	2	3	6	9	8
F	0	0	0	0	1	1	1	4	5	8	11
F	0	0	0	0	0	0	0	3	4	7	10

Figure 2: Local Sequence Alignment using Smith-Waterman algorithm

It is worthy to mention that on the **topleft** corner of every table cell there are set the three possible scores given from the three possible ancestors of each cell, respectively. The best of the scores are highlighted in red color. These red colored possible scores also indicate the only pointers for the path to be followed when the dynamic algorithm is backtracking to find the global sequence alignment. In addition to the above, as you can also observe, when all possible scoring scores are negative then a 0-restart is done, unlike the Needleman-Wunsch algorithm. What is more, when the dynamic algorithm fills in all the cells of the table, it starts the backtrack path from the cell of the table that has scored the maximum value, in contrast Needleman-Wunsch that starts backtracking from the bottom right cell. In our case the cell with the biggest score is highlighted in purple.

In the case of equal possible scores, the dynamic algorithm selects to follow the path through the cell that has the biggest given score. In this table, it happens not to exist such an case. Finally, as it is obvious from Figure 2, the backtrack path for the optimal solution is highlighted in pale pink color. Given this backtrack path, we can infer that the optimal match including the inserted gaps is as follows:

```
y = A S I R V V F A L F
    | |   |       | | | |
x = A S - R - - F A L  F  F
```

and as a result of the optimal match, the score gathered is 2+2-1+2-1-1+2+2+2+2 = **11**.

We also implemented a biojava application using the resource packages of the Smith-Waterman algorithm. In this way we confirmed the output results. The biojava implementation is given with the deliverable .zip package.

As a supplementary to all the above, a nice web application with which we also checked our results can be found in this link: http://baba.sourceforge.net/

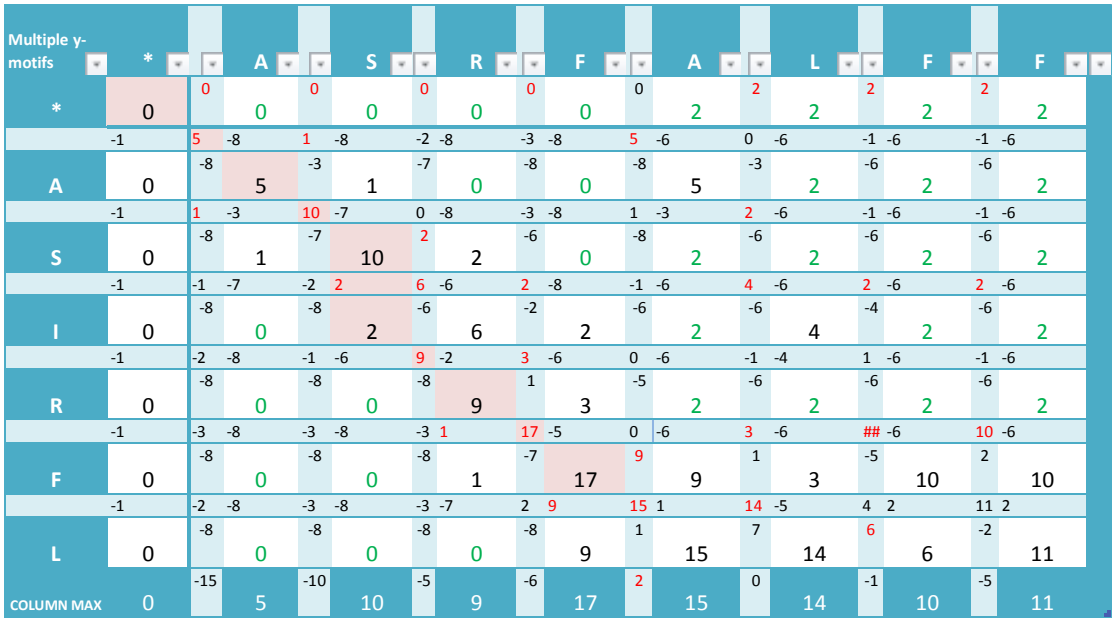
2(a).

We are assigned to find a maximum number of local matches scoring beyond threshold *T*, maximizing the function of multiple *y*-motifs [cf 1.dynpr.pdf, p.17], with BLOSUM50 scores [cf 1.dynpr.pdf, p.6], gap cost of *d*=8 and threshold *T*=15.

The two protein sequences are:

x = ASRFALFF, and y = ASIRFL

To do this we will use the Smith-Waterman algorithm. Figure 3 shows the filled-in table and the backtrack paths of the matches scoring > *T*.



Multiple y-motifs	*		A		S		R		F		A		L		F		F
*	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2
	-1	5	-8	1	-8	-2	-8	-3	-8	5	-6	0	-6	-1	-6	-1	-6
A	0	-8	5	-3	1	-7	0	-8	0	-8	5	-3	2	-6	2	-6	2
	-1	1	-3	10	-7	0	-8	-3	-8	1	-3	2	-6	-1	-6	-1	-6
S	0	-8	1	-7	10	2	2	-6	0	-8	2	-6	2	-6	2	-6	2
	-1	-1	-7	-2	2	6	-6	2	-8	-1	-6	4	-6	2	-6	2	-6
I	0	-8	0	-8	2	-6	6	-2	2	-6	2	-6	4	-4	2	-6	2
	-1	-2	-8	-1	-6	9	-2	3	-6	0	-6	-1	-4	1	-6	-1	-6
R	0	-8	0	-8	0	-8	9	1	3	-5	2	-6	2	-6	2	-6	2
	-1	-3	-8	-3	-8	-3	1	17	-5	0	-6	3	-6	##	-6	10	-6
F	0	-8	0	-8	0	-8	1	-7	17	9	9	1	3	-5	10	2	10
	-1	-2	-8	-3	-8	-3	-7	2	9	15	1	14	-5	4	2	11	2
L	0	-8	0	-8	0	-8	0	-8	9	1	15	7	14	6	6	-2	11
COLUMN MAX	0	-15	5	-10	10	-5	9	-6	17	2	15	0	14	-1	10	-5	11

Figure 3: Local Sequence Alignment using Multiple y-motifs algorithm and BLOSUM50.

It is worthy to mention that multiple *y*-motifs algorithm has a high resemblance to Smith-Waterman's algorithm. On the **topleft** corner of every table cell there are set the three possible scores given from the three possible ancestors of each cell, respectively. The best of the scores are highlighted in red color. These red colored possible scores also indicate the only pointers for the path to be followed when the dynamic algorithm is backtracking to find the global sequence alignment. In addition to the above, as you can also observe, when all possible scoring scores are negative then a 0-restart is done, unlike the Needleman-Wunsch algorithm. Moreover, the multiple *y*-motifs algorithm begins the implementations in a vertical order. Firstly, it fills in with zeros the first column and sets the first line's significance as more important. This is because every other cell in every column (except for the first column and the first cell in each column) depends not only from the 3 ancestor cells but also from the first cell of its column, too. So, we highlight all the cells in the first line too. The cells highlighted in green have taken their values from these first cells' values.

What is more, when the dynamic algorithm fills in all the cells of the table, it starts all the backtrack paths from the cells of the table that have scored a value bigger than the threshold. In our case there exist only one path beginning from a cell whose value is bigger than the threshold.

In the case of equal possible scores, the dynamic algorithm selects to follow the path through the cell that has the biggest given score. In this table, it happens not to exist such an case. Finally, as it is obvious from Figure 3, the backtrack path whose cell value is bigger than the threshold of score=15 is highlighted in pale pink color.

Given this backtrack path, we can infer that the optimal match including the inserted gaps is for the algorithm of multiple *y*-motifs is as follows:

```
y = A S I R F L
    | |  | |
x = A S - R  F A L  F  F
```

2(b).

As we found in the previous step of this exercise there is only one path that matches a score above the threshold of 15. It is obvious, though, from the above table that if we set the threshold to 14, rather than 15, then we are going to have two matching sequence alignments. One beginning from F → F (score = 17) and the other beginning from L → A (score = 15).