Bioinformatics I
WS 15/16
Tutor: Alexander Seitz

1	2	3	$\sum$

Jonas Ditz
& Benjamin Schroeder

Blatt 7

(Abgabe am 26. October 2015)

Exercise 1 - *Hirschberg algorithm for alignment in linear space*

With two given Sequences (X= TGGA and Y=TTGAGA), the scoring of s(a,a) = 4, s(a,b)=-3 and a gap penalty g=-3, the Hirschberg algorithm was started. First the sequence X was split into two substrings, one was the prefix (-green): TG and one the suffix sequence GGA (- yellow). The prefix was used in a NW alignment and the suffix was also used in a NW alignment, but this time reversed. Both NW Matrices were calculated with the space reduced algorithm technique. The result is an array with the maxima the matrix saved in an array (-red columns). Both arrays are cellwise added up and the maximum is used to determine the two positions of both sequences (-red cell), which should be aligned. In step 1 this was the G(X2) to the G(Y3). These position were finally saved in a final array.

First, the Sequence X was split into two parts and The second steps starts with cutting the

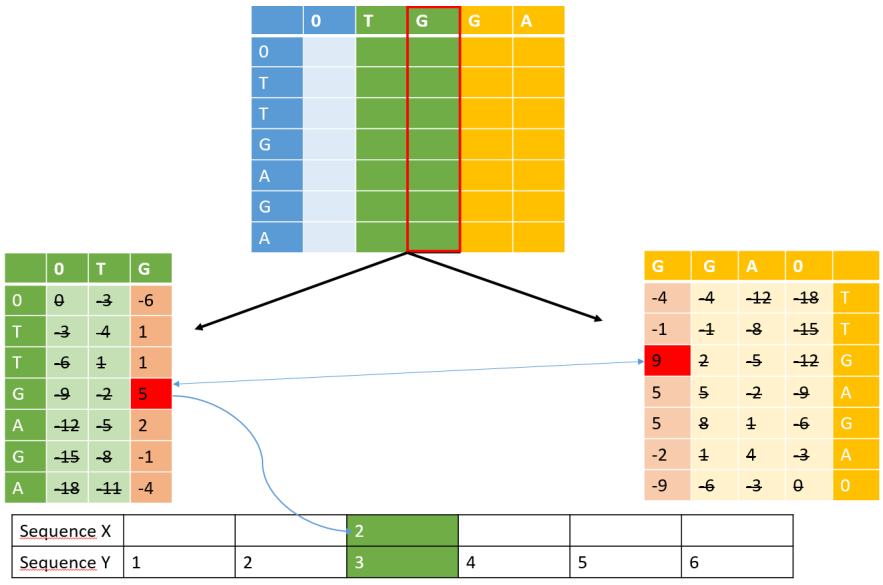

Figure 1: In the first step the sequence X was split into two parts and for both a NW global alignment was calculated. It's important to state, that the NW of the suffix of the sequence splitting was calculated reverse. The Maximum which was found after the substraction of both maxima was used to identify two letters, whrch are to be aligned.

matrix in the Sequence Y Dimension. Definition of how to start the cutting is not very clearly described, and therefore we cutted at position y+1(- yellow and gold area) and y-1 (- green area) of the last maximum. The yellow and gold part was processed like in the first step. The green one

was interpreted as the trivial case, were one sequence has the length one. Which means we simply took the maximum of the table as the two aligning position and saved it in the result array.

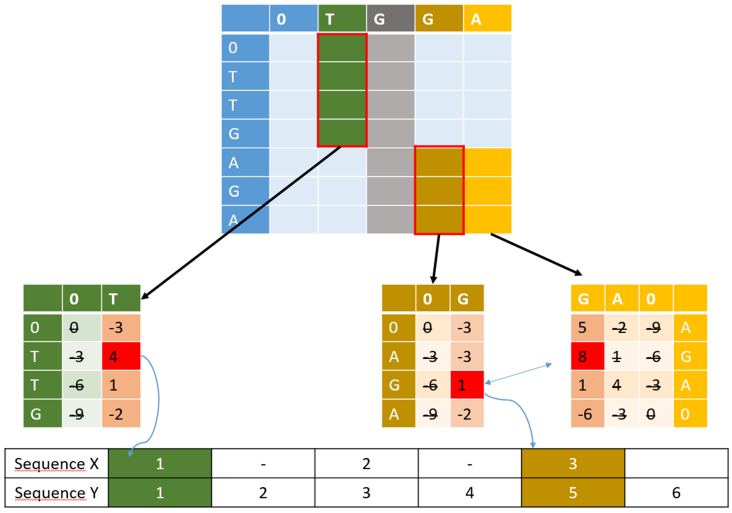

Figure 2: Step 2 - calculating the needleman wunsch alignments for the sub sequences.

In step 3 the trivial case was now in the yellow area (-yellow area). And the last position for aligning was filled into the result matrix.

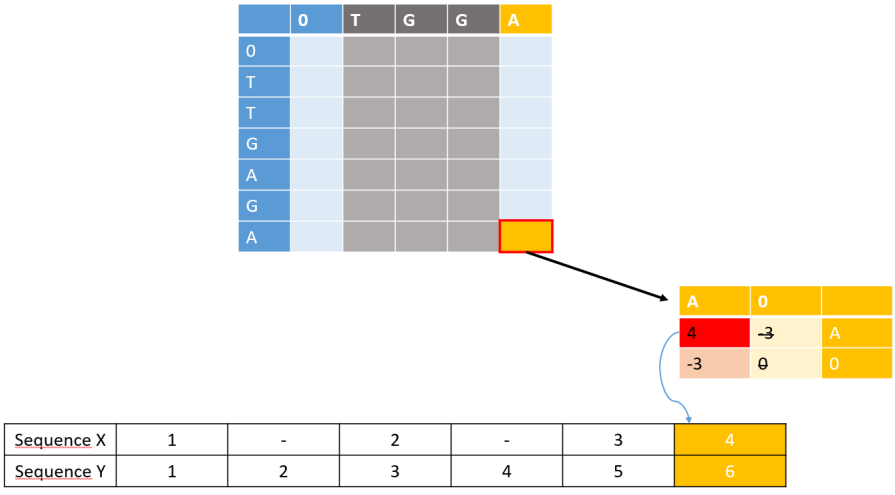

Figure 3: step 3 - final trivial case

In step 4 the interpretation of the result array was processed. This means, that if a position was in the result array, both letters were aligned according to the positions. If there was no position in the array, a gap was filled in ().

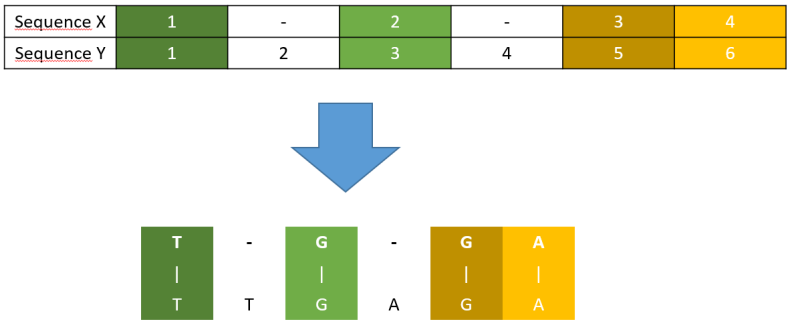

Figure 4: Interpretation of the Result array.

Exercise 2 - *Do scoring matrices have expected score smaller than zero?*

The substitution matrices were evaluated according to the expectancy value formula given in the lecture, the substitution matrices were evaluated with the assumption of uniform distributed amino acids in two sequences.

$$\sum_{a,b\in\Sigma} p_a p_b s(a,b) <=> (\sum_{a,b\in\Sigma} s(a,b)) p_a p_b$$

For the determination of one expectance value, p_a and p_b were each substituted with $1/20$, for the probabilistic occurrence in sequences with uniform distributed amino acids. The sum was calculated via R and the total summation of the substitution matrix .The result for each matrix can be seen in table .

Substitution Matrix	Expectance Value
BLSOUM50	-1.155
BLOSUM52	-1.065
BLOSUM80	-2.3275
PAM250	-1.14
PAMN	0.155

Table 1: Result of the calculation with each given substitution matrix according to equation

As a conclusion we want to state, that all substitution matrices are possible feasible matrices. Except of the PAMN matrix, which has a positive expectancy value, which could lead to wrong conclusion. One Aspect we also want to state, is that we did not calculate the values for the matrices with the unknown sign x or X. and the B or Z sign, which stands for two amino acids. Although it would be possbile to use the unknown sign, if they are also uniform distributed in a sequence. But as a result of the wanted comparison, between matrices which didn't contain these signs, we did not use these columns and rows in our calculations.

Exercise 3 - *Needleman-Wunsch algorithm for affine gap scores*

We extended our GapPenalty class from the previous assignment. Now it is possible to get the initialization cost. One just has to call the function getInitCost() with an integer, which defines the current cell index i or j, respectively. This leads us to a very convenient way to initialize our matrix.