Smith waterman algorithm

***Aim***: The best alignment over the conserved domain of two sequences.

***Features of smith waterman algorithm:***

* The **smith –waterman algorithm** performs a ***global alignment*** on two sequences
* It is an example of ***dynamic programming***, and was the first application of dynamic programming to biological sequence comparison
* Suitable when the two sequences are of similar length, with a significant degree of similarity throughout
* Aim: The best alignment over the entire length of two sequences .
* The main difference to the ***Needleman–Wunsch algorithm*** is that negative scoring matrix cells are set to zero, which renders the (thus positively scoring) local alignments visible. Backtracking starts at the highest scoring matrix cell and proceeds until a cell with score zero is encountered, yielding the highest scoring local alignment.

***Steps:***

* Initialization
* Scoring
* Trace back (Alignment)
* Consider the two sequences to be globally aligned are:

ATCG (x=4, length of sequence 1)

TCG (y=3, length of sequence 2)

* Create a matrix with X +1 Rows and Y +1 Columns.
* The 1st row and the 1st column of the score matrix are filled with 0s

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| - | - | **T** | **C** | **G** |
| - | 0 | 0 | 0 | 0 |
| **A** | 0 |  |  |  |
| **T** | 0 |  |  |  |
| **C** | 0 |  |  |  |
| **G** | 0 |  |  |  |

***Scoring:***

* The score of any cell C(i, j) is the maximum of:

scorediag = C(i-1, j-1) + S(I, j)

scoreup = C(i-1, j) + g

scoreleft = C(i, j-1) + g

And

0

here S(I, j) is the substitution score for letters i and j, and g is the gap

penalty.

***Scoring:***

* Example:

The calculation for the cell C(2, 2):

scorediag = C(i-1, j-1) + S(I, j) = 0 + -1 = -1

scoreup = C(i-1, j) + g = 0 + -1 = -1

scoreleft = C(i, j-1) + g = 0 + -1 = -1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| - | - | **T** | **C** | **G** |
| - | 0 | 0 | 0 | 0 |
| **A** | 0 | 0 |  |  |
| **T** | 0 |  |  |  |
| **C** | 0 |  |  |  |
| **G** |  |  |  |  |

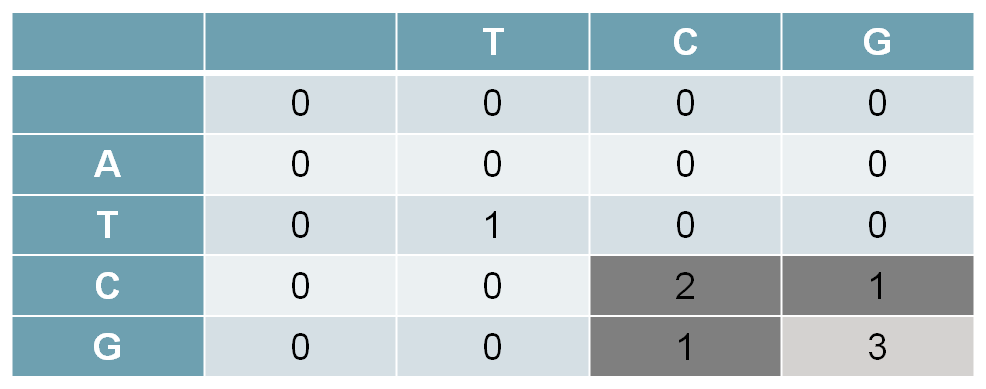
***Final Scoring Matrix***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **T** | **C** | **G** |
|  | 0 | 0 | 0 | 0 |
| **A** | 0 | 0 | 0 | 0 |
| **T** | 0 | 1 | 0 | 0 |
| **C** | 0 | 0 | 2 | 1 |
| **G** | 0 | 0 | 1 | 3 |

**Note: It is not mandatory that the last cell has the maximum alignment score!**

***Trace back:***

* The trace back step determines the actual alignment(s) that result in the maximum score
* There are likely to be multiple maximal alignments
* Trace back starts from the cell with maximum value in the matrix
* Gives alignment in reverse order
* There are three possible moves: diagonally (toward the top-left corner of the matrix), up, or left
* Trace back takes the current cell and looks to the neighbor cells that could be direct predecessors. This means it looks to the neighbor to the left (gap in sequence #2), the diagonal neighbor (match/mismatch), and the neighbor above it (gap in sequence #1). The algorithm for trace back chooses as the next cell in the sequence one of the possible predecessors. This continues till cell with value 0 is reached.



* The only possible predecessor is the diagonal match/mismatch neighbor. If more than one possible predecessor exists, any can be chosen. This gives us a current alignment of

Seq 1: G

|

Seq 2: G

