76558 | Algorithms in Computational Biology | Exercise 1

Eran Eben Chaime 308240597, Guy Lutsker 207029448

Question 1

Let there be the sequences S, T of length n, how many possible sequence alignments are there? Show its at least 3^n.

$$Solution:$$

To assess the number of possible sequence alignments we can think about it as a combinatorical problem in which we can choose from the following pairs: $S_i|T_j, S_i|-, -|T_j, -|-$
(where i, j are indices in the S, T sequences).
And so at each position of an alignment we have 4 possible choices - altogether there are 4^n possible alignments.
This is more than 3^n as required.

Question 2

Generate a recursive function to solve the problem if the cost of the first gap we see is d and every subsequent one is e.

$$Solution:$$

Initialization:
* $V \in \mathbb{R}^{n \times n}$
* $L \in \mathbb{R}^{n \times n}$
* $\eta : \mathbb{R}^3 \to \mathbb{R}$
* σ (We assume its provided) : $\mathbb{R}^2 \to \mathbb{R}$

Initialize matrices :

$$V(0,0) = 0$$

$$\forall i > 0 \; V(i,0) = V(i-1,0) + \eta(i,0)$$

$$\forall j > 0 \; V(0,j) = V(0,j-1) + \eta(0,j)$$

$$L(0,0) = 0$$

$$\forall i > 0 \; L(i,0) = 1$$

$$\forall j > 0 \; L(0,j) = 1$$

Recursion function:

$$\forall i, j \in [1,n] \times [1,n]$$

$$V(i,j) = max \begin{cases} V(i-1,j-1) + \eta(S_i, T_j, 0) \\ V(i-1,j) + \eta(S_i, '-', L(i-1,j)) \\ V(i,j-1) + \eta('-', T_j, L(i,j-1)) \end{cases}$$

$$\eta(S_i, T_j, l) = \begin{cases} d & l = 0 \\ e & l = 1 \text{ or } l = 2 \\ \sigma(S_i, T_j) & else \end{cases}$$

$$LIST = [V(i-1,j-1) + \eta(S_i, T_j, 0), V(i-1,j) + \eta(S_i, '-', L(i-1,j)), V(i,j-1) + \eta('-', T_j, L(i,j-1))]$$

$$L(i,j) = argmax\{LIST\}$$

Explanation:

V is a matrix similar to the one we defined in class, the only difference is that it uses η as a cost function instead of σ.

η is a cost function for adding new letters to the alignment, it takes into consideration the previous letter that was added to the alignment as required in the question - adds d for a single gap and e for every subsequent gap.

To "remember" when adding a gap if its the first gap we add, or one in a sequence of gaps we define a matrix L.

$L(i,j)$ holds the direction of the cell we generated $V(i,j)$ from(0=topleft-letter,1=top-gap,2=left-gap) .

And so we know in η whether to add d/e as a cost.

The extraction of the the optimal path is identical to the one we saw in the lecture - we start from the last cell of the matrix - $V[n,n]$ and the matrix L holds the positions we came from to generate each cell - and so we can simply use L as pointers to which letter/gap to add, as we saw in the lecture.

And so our pseudo code is simply to fill the the tables as we mentioned, and the optimal solution will be in the last cell of the matrix V - $V[n,n]$, the extraction of the path was already mention beforehand.