**Analysis of sequence alignments II**

Of course, the lengths may vary, but they still have upper bounds. Consider two sequences $A$ and $B$ of the same length -- $n$. In the alignment of the extended-sequences that are also maximally long, every amino acid of $A$ or $B$ is aligned against a gap -- $(a/-)$ and $(-/b)$. An increase of the length beyond $2n$ will necessarily include an alignment of an indel with respect to another indel, i.e. $(-/-)$. Such an alignment does not make sense from a scientific point of view. We have no way to determine the number of “double gaps” or their locations. Moreover, also from a technical view point there is a problem if the $(-/-)$ pair generates a favorable plausible score. Consider the alignment

$$a_1 - ... - a_{n-1} - a_n$$

$$- b_1 - ... - b_{n-1} b_n$$

Let the total score be $T_{AB}$. Extending the above alignment by one more pair of indels, we have

$$a_1 - ... - a_{n-1} - a_n$$

$$- b_1 - ... - b_{n-1} b_n$$

The new score is $T_{AB} + S_{-/-}$, where $S_{-/-}$ is the element of the substitution matrix replacing an indel by an indel. If the new score is better than $T_{AB}$, it is trivial to construct even a better alignment (and score) by adding yet another pair of indels with yet a better score of $T_{AB} + 2 \cdot S_{-/-}$. The favorable extension with indels can proceed to infinite and is unbound.

We therefore eliminate in our sequence-to-sequence alignments the possibility of $(-/-)$. Note however, that if we wish to compare more than two sequences simultaneously (multiple sequence alignment), then the possibility of matching a gap against a gap exists.
For example, two gaps may appear in the element comparison \((-/-/a)\) when matching three sequences.

Ok. It is settled then, the maximum length of \(\overline{A}\) is \(2n\). How many possible alignments (with gaps) do we have? In a naïve approach, this number may be related to the number of scores we need to compute before deciding on the optimal alignment.

### 1.2 Counting alignments

To count the number of possible alignments, and as a starting point of the discussion on optimal alignments, we consider the dynamic matrix. The dynamic matrix is a table used for the alignment of two sequences. Below we provide one example for two sequences with the same length \((n = 5)\). The rows are associated with the \(A\) sequence and the columns with the \(B\). The numbers at different matrix entries will be explained below.

<table>
<thead>
<tr>
<th></th>
<th>(-)</th>
<th>(a_1)</th>
<th>(a_2)</th>
<th>(a_3)</th>
<th>(a_4)</th>
<th>(a_5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(b_1)</td>
<td>1</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>(b_2)</td>
<td>1</td>
<td>5</td>
<td>13</td>
<td>25</td>
<td>41</td>
<td>61</td>
</tr>
<tr>
<td>(b_3)</td>
<td>1</td>
<td>7</td>
<td>25</td>
<td>63</td>
<td>129</td>
<td>231</td>
</tr>
<tr>
<td>(b_4)</td>
<td>1</td>
<td>9</td>
<td>41</td>
<td>129</td>
<td>321</td>
<td>681</td>
</tr>
<tr>
<td>(b_5)</td>
<td>1</td>
<td>11</td>
<td>61</td>
<td>231</td>
<td>681</td>
<td>1683</td>
</tr>
</tbody>
</table>
The hairy picture with the numerous arrows is actually telling. Paths in this table, which start at the upper left corner and end at the right lower corner, present all the possible alignments of the whole two sequences. From each entry in the dynamic matrix, there are three alternative moves: Going down along the diagonal, going straight down, or moving to the right. A step in the matrix, which is a part of a legitimate alignment, never proceeds to the left or up. For example, the thick line in the above matrix corresponds to the alignment:

\[
\begin{align*}
a_1 & \quad - \quad a_2 \quad a_3 \quad a_4 \quad a_5 \\
b_1 & \quad b_2 \quad b_3 \quad - \quad b_4 \quad b_5
\end{align*}
\]

A move along a diagonal aligns an amino acid against another amino acid. A vertical step in the matrix aligns a “b” amino acid against an indel and a horizontal step puts a gap against an “a” amino acid. Note that we use “−” for the gap “residue” (or an indel).

The numbers at the different entries of the table denote the number of paths (alignments) that can reach this point. For example, there are 5 possible alignments of \( a_1a_2 \) against \( b_1 \) (check the table element at the cross between \( a_2 \) and \( b_1 \)). They are:

\[
\begin{align*}
(1) & \quad a_1 \quad - \quad a_2 \\
(2) & \quad - \quad a_1 \quad a_2 \\
(3) & \quad a_1 \quad - \quad a_2 \\
(4) & \quad a_1 \quad a_2 \quad - \\
(5) & \quad - \quad - \quad a_1 \quad a_2 \quad -
\end{align*}
\]

The last three alignments include the same elements for comparison and their scores are therefore degenerate. It is therefore not possible to decide on a “best” alignment from the group of three. There is more than one optimal alignment. At present, we consider all paths, including the degenerate ones. In the Appendix a clever counting protocol (by Dr.
Jaroslaw Meller) is outlined that estimates the number of non-degenerate paths. Interestingly, the exact number of non-degenerate paths is \( \binom{n+m}{m} \). It has the same asymptotic behavior as the approximate lower-bound expression we sketched below for the number of all paths.

Another interesting property of this table is a summation rule and the possibility of constructing a recursion formula for the number of alignments. There are three ways (“sources”) to extend a shorter alignment to obtain one of the five longer alignments listed above. (a) Extend an earlier alignment by the pair \((a_2/b_1)\), a diagonal move in the above table. Alternatively, (b) the pairs \((a_2/-)\), or (c) \((-/b_1)\) (horizontal or perpendicular moves in the above table) can be used to extend a shorter alignment and to obtain a member of the above group.

Each of the three “sources” \((a)-(c)\) has a corresponding position in the matrix. The number at the entry to the matrix is the number of paths aligning a segment of \(A\) against a segment of \(B\). For example, before adding the pair \((a_2/b_1)\) we were at a table entry aligning \(a_i\) against "-". We find that there is only one-way of aligning \(a_i\) against a gap and “1” is indeed the corresponding entry.

Another example of extending the alignment is to add a gap against \(a_2\), which means that our earlier position in the table was the alignment of \(a_i\) and \(b_i\). The last alignment can be
done in three different ways and therefore the table entry is “3”

\[
\begin{array}{cc}
ad_{1} & a_{1} \\
b_{1} & b_{1} \\
\end{array}
\]

If we add the number of paths starting at the previous three “sources”, the number of alignments of \(a_{1}a_{2}\) with respect to \(b_{1}\) summed up to \(1+1+3 = 5\), exactly the number of alignments of our target.

To summarize the above empirical observations more precisely:

The number of possible alignments of \(n\ a\)-s against \(m\ b\)-s is defined as \(N(n,m)\), this number can be determined using the recursive formula\n
\[
N(n,m) = N(n-1,m-1) + N(n-1,m) + N(n,m-1),
\]

and the initial conditions \(N(0,0) = N(0,m) = N(m,0) = 1\). Note that the definition of \(N(0,0)\) was done for computational convenience and it does not imply that “nothing” against “nothing” can be aligned in exactly one way.

While this formula can be used directly, it is useful to have a quick, order-of-magnitude estimate of the number of alignments. This estimate is especially useful if we are planning a computation that will enumerate all of the alignments. If a calculation is not feasible with existing computer resources it is better knowing that it is not feasible in advance, and not after a few weeks of futile attempts to execute the desired computation.

A simple lower bound for \(N(n,m)\) can be obtained quickly. \(N(n,m)\) is a positive number, so we can write \(N(n,m) > N(n-1,m) + N(n,m-1)\) (we “forgot” for
convenience the term \( N(n-1,m-1) \) in the original equality). So, the alternative recursion formula \( N'(n,m) = N'(n-1,m) + N'(n,m-1) \) (with the same initial conditions as for \( N(n,m) \)) always yields lower numbers than \( N(n,m) \). For \( N'(n,m) \) we have a close expression: \( N'(n,m) = \frac{(n+m)!}{m!n!} \). This formula is easy to verify by direct substitution of the closed expression to the recursion. Note that this is the same as the (exact) result derived by Meller for non-degenerate paths (Appendix).

We now make use of the Stirling formula: \( \log[n!] \approx n \log(n) - n - [x] \), valid for large \( n \)-s.

The logarithm of the number of alignments \( N'(n,n) \) is estimated as

\[
\log \left[ \frac{(2n)!}{(n!)^2} \right] = \log[(2n)!] - 2 \log[n!] \approx 2n \cdot \log(2n) - 2n - 2n \log(n) + 2n = 2n \log(2)
\]

And the lower bound for the number of alignments is \( N'(n,n) \approx 2^{2n} \). For a short protein \( n \approx 50 \); \( N \approx 1.27 \cdot 10^{30} \) the number is substantial. Even if the computation of a single alignment requires a nanosecond \( (10^{-9} \text{ second}) \), which is unrealistically fast, it still necessary to use \( 10^{21} \text{ sec} \approx 10^{13} \text{ years} \) to examine all possible alignments.

If this is not impressive enough, remember that this is a lower bound and the precise counting of all paths will yield a number significantly larger than this one. For example, for \( n = 5 \) we have \( 2^{10} = 1024 \) already a number significantly lower than the exact number in the table (1683). Hence, this estimate underlines the claim that it is impossible to examine all alignments one by one in order to find the alignment with the highest score.
Readers with a background in structural biology may recall the Levinthal paradox in protein folding. The paradox puts in contrast the huge number of plausible protein conformations and the efficiency in which proteins fold in nature. We cannot be sure (until the next section) that there is a solution for the sequence alignment problem. However, nature solves the protein folding problem. So at least the existence of the solution for the protein folding problem is confirmed. This is (again) in contrast to sequence alignment for which an efficient solution exists. As we see below a large space to search does not necessarily mean that optimization in that space is difficult. It is not obvious that the optimization must be performed at a cost proportional to the volume of that space. In fact it can be profoundly cheaper.