Sequence alignment is a tool to test for similarities between a sequence of an unknown target protein, and a single (or a family of) well-characterized protein(s). It is assumed that sequence similarity implies similarity in function.

Figure 1: A cartoon plot illustrating the use of a database of annotated proteins to identify a novel sequence

We denote the sequence of the unknown protein by $A$. It is an abbreviation of the explicit sequence $A = a_1, a_2, ..., a_n$, where $a_i$ is an amino acid placed at the $i^{th}$ position along the sequence. The sequences of the known proteins, that make the comparison database, are called $\{B_q\}_{q=1}^Q$ where $B_q = b_1^q, b_2^q, ..., b_m^q$, and $Q$ is the total number of proteins in the database. For sequences only, the database may include hundreds of thousands of entries [http://www.expasy.ch/srs5/]. Databases that include both sequences and structures are
limited to about ten thousand entries [http://www.rcsb.org/pdb/]. For brevity we omit the index from $B_q$ and use $B$ to represent a member of the comparison set.

Table 1: A sample of protein sequences. Typical lengths of protein sequences vary from a few tens to a few hundreds. The single letter notation (one character for an amino acid) is used.

<table>
<thead>
<tr>
<th>Protein Type</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serine Protease inhibitor</td>
<td>EDICSLPPEV GFCRAGFLKF AYYSELNKCK LFTYGGCQGN ENNFETLQAC XQA</td>
</tr>
<tr>
<td>Amicyanin alpha</td>
<td>MRALAAALAAAPSATAALAAGALEAVQEAPAGSTEVKIAMKFQOTPEVR</td>
</tr>
<tr>
<td></td>
<td>IKAGSAYTWNTEALPHNVHFSGFPGVEKDVPSMLRNSQGYSVKFNAPG TYDICTHPFMKGVVVE</td>
</tr>
<tr>
<td>Major Histocompatibility Complex (class I)</td>
<td>MAVMAPRTLVL LLSSGALALT QTWAGSHMR YFSTSVSRPG RGEPRFIAVG YVDDTFQVRF</td>
</tr>
<tr>
<td></td>
<td>DSDAASQRME PRAPWIEQEG PEYWDRNTRN VKAHSSQTDVR DLGTLRGYYN QSEDGSHTIQ</td>
</tr>
<tr>
<td></td>
<td>RMYGCDVGS QGRFLGYYQD AYDGKDYIAL NEDLRSWTAA DMAEITKRK WEAHFAEQL</td>
</tr>
<tr>
<td></td>
<td>RAYLEGTCVE WLRRLHENGK ETLQRTDAPK THMTHHAVSD HEAILRCWAL SFYPAEITLT</td>
</tr>
<tr>
<td></td>
<td>WQRDGEDQTQ DTELVETRPA GDGTFQKWAA VVVPSQEOQR YTCVQHEGL PEEPLLERWEP</td>
</tr>
<tr>
<td></td>
<td>SSQPTIPIVG IIAGLVLFGA VIAGAVYAVV RWRKSSDRK GGSYQQAASS DSAQGSDVSL</td>
</tr>
<tr>
<td></td>
<td>TACKV</td>
</tr>
</tbody>
</table>

b.2 An introduction to the alignment score

A more precise definition of similar is needed. The simplest definition (one) is the fraction of amino acids in $B$ that are identical to $A$. In proteins, high sequence identity
(roughly above 35 percent) immediately implies close relationship. A more subtle definition of similarity (two) uses amino acids with similar physical or biochemical properties. For example, replacing a hydrophobic amino acid like leucine (L) by valine (V) is (usually) an acceptable mutation. The effect on the hydrophobic core of the protein is expected to be small. However, changing (L) to a charged residue like arginine (R) is likely to be damaging and significantly affect protein stability.

Hence, there is a gray area between an exact matching (two identical amino acids) and a complete mismatch. This gray area is better described by a scoring function with a wide spectrum. The scoring function gives the highest score to an identical pair and the lowest possible score (to) substitutions that do not make chemical, physical or biological sense.

To make sequence comparison compact and easily extendable to a wide range of sequences, we simplify the process. We assume that the total score \( S(A,B) \) of comparing sequences \( A \) and \( B \) will be given by a sum of terms, each term comparing two elements, an element from \( A \) and an element from \( B \). An element is either an amino acid or a “space” (gap). For example, consider the case in which \( A \) is significantly longer than \( B \). In (this) case it is obvious that some amino acids of \( A \) will be aligned against “spaces” (gaps). The choice of the pairs of elements for the similarity test is called an alignment and is the focus of the present section.

There are two different problems that we need to address: The selection of the alignment and the determination of the scoring function. Let us start with the score. Since
we simplified the calculation of the total score, making it a sum of element comparisons, we need to consider only comparison of individual amino acids and comparisons of an amino acid against a gap. However, we do not need to consider separately scores of pairs of amino acids since $S(a_1a_2, b_1b_2) = S(a_1, b_1) + S(a_2, b_2)$.

The scores of pairs of amino acids, which measure the similarity of the $a$-s and the $b$-s, form the so-called substitution matrix $S(a, b)$. The matrix is symmetric, and of size $20 \times 20$ (the number of amino acid types). Note that the term “substitution” may be misleading since it suggests a direction (substituting $a$ by $b$ implies that $a$ was there first). A symmetric matrix does not have a sense of time, and the order of substitutions does not change the score, or the degree of similarity.

Nevertheless, a non-symmetric variation of the sequence as a function of time is of considerable interest. A direction in sequence-similarity-scores is potentially informative, providing molecular fingerprints for evolutionary time scales. However, it is hard to estimate it without a specific model in mind. At present, for the purpose of finding protein relatives, we shall use the simplest way out and ignore the time arrow of evolution. A widely used (symmetric) substitution matrix is given below
The BLOSUM 50 matrix [x]

|     | A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V |
| A   | 5 | -2| -1| -2| -1| -1| 0 | -2| -1| -2| -1| -3| -1| 1 | 0 | -3| -2| 0 |
| R   | -2| 7 | -1| -2| -4| 1 | 0 | -3| 0 | -4| -3| 3 | -2| -3| -3| -1| -1| -3| -1| -3| 0 |
| N   | -1| -1| 7 | 2 | -2| 0 | 0 | 0 | 1 | -3| -4| 0 | -2| -4| -2| 1 | 0 | -4| -2| -3| 0 |
| D   | -2| -2| 2 | 8 | -4| 0 | 2 | -1| -1| -4| -4| -1| -4| -5| -1| 0 | -1| -5| -3| -4| 0 |
| C   | -1| -4| -2| -4| 13| -3| -3| -3| -3| -2| -2| -3| -2| -2| -4| -1| -1| -5| -3| -1| 0 |
| Q   | -1| 1 | 0 | 0 | -3| 7 | 2 | -2| 1 | -3| -2| 2 | 0 | -4| -1| 0 | -1| -1| -1| -3| 0 |
| E   | -1| 0 | 0 | 2 | -3| 2 | 6 | 3 | 0 | -4| -3| 1 | -2| -3| -1| -1| -1| -3| -2| -3| 0 |
| G   | 0 | -3| 0 | 1 | -3| -2| -3| 8 | -2| -4| -4| -4| -2| -3| -4| -2| 0 | -2| -3| -3| 0 |
| H   | -2| 0 | 1 | -1| -3| 1 | 0 | 2 | 0 | -4| -3| 0 | -1| -1| -2| -1| -2| -3| 2 | -4| 0 |
| I   | -1| -4| -3| -4| -2| -3| -4| -4| -4| 5 | 2 | -3| 2 | 0 | -3| -3| -1| -3| -1| -4| 0 |
| L   | -2| -3| -4| -4| -2| -2| -3| -4| -3| 2 | 5 | -3| 3 | 1 | -4| -3| -1| -2| -1| 1 | 0 |
| K   | -1| 3 | 0 | -1| -3| 2 | 1 | -2| 0 | -3| -3| 6 | -2| -4| -1| 0 | -1| -3| -2| -3| 0 |
| M   | -1| -2| -2| -4| -2| 0 | -2| -3| -1| 2 | 3 | -2| 7 | 0 | -3| -2| -1| -1| 0 | 1 | 0 |
| F   | -3| -3| -4| -5| -2| -4| -3| -4| -1| 0 | 1 | -4| 0 | 8 | -4| -3| -2| 1 | 4 | -1| 0 |
| P   | -1| -3| -2| -1| -4| -1| -1| -2| -2| -3| -4| -1| -3| -4| 10| -1| -1| -4| -3| -3| 0 |
| S   | 1 | -1| 1 | 0 | -1| 0 | -1| -3| -3| 0 | -2| -3| -1| 5 | 2 | -4| -2| -2| 2 | 1 | 0 |
| T   | 0 | -1| 0 | -1| -1| -1| -1| -2| -2| -1| -1| -1| -1| -2| -1| 2 | 5 | 3 | -2| 0 | 0 |
| W   | -3| -3| -4| -5| -5| -1| -3| -3| -3| -3| -2| -3| -1| 1 | -4| -4| -3| 15| 2 | -3| 0 |
| Y   | -2| -1| -2| -3| -3| -1| -2| -3| 2 | -1| -1| -2| 0 | 4 | -3| -2| -2| 2 | 8 | -1| 0 |
| V   | 0 | -3| -3| -4| -1| -3| -3| -4| -4| 4 | 1 | -3| 1 | -1| -3| -2| 0 | -3| -1| 5 | 0 |

* * A R N D C Q E G H I L K M F P S T W Y V *
Note the high scores for a substitution of an amino acid to self (the diagonal elements of the matrix). Note also that certain amino acids have higher tendency for self-preservation (e.g. Cysteine) while a small and polar amino acid like threonine can be substituted to other amino acids more easily.

**sequence alignment: definitions**

The choice of pairs of elements (amino acids or gaps) from $A$ and $B$ is not completely free. The order of the amino acids in the sequence (the primary “structure” of the protein) counts, and our comparison should maintain that order. This is a significant restriction on the comparisons that we may make and limits our choices considerably. We are seeking the alignment of the two sequences that will give us the optimal (highest) score. Adding further to the complexity is the observation that not all proteins are of the same length; we need to pay attention to the variation in the lengths of the proteins and for the fact that some amino acids will not have corresponding amino acids to match. Two closely related proteins may be different at a specific site in which an amino acid was added to one protein but not to the second. To describe such cases we introduce a “gap” residue. A gap residue is denoted by “-“ and is used to indicate an empty space along the sequence. For example, the comparison of two proteins below (PDB codes 1rsy and 1a25_A, the “_A” denotes the first chain in the data entry) yields the following optimal arrangement of the two sequences with gaps:
TABLE: Optimal alignment of two sequences (top is 1rsy, Synaptotagmin I (First C2 Domain), lower 1a25, Protein Kinase C (β); Chain: A). The percentage of sequence identity is 33 percent.

| EKLGKLQYSLYDFQNNQLLVGIIQ-AAEL-PALDMGGTSDPYKVFLPPD-K-KKKFE |
| ERRGRIYIQAID-R--EVLIVVVRDAKNLVP-MDPNGLSDPYVKLKLIPDPKSESQKQ |
| TKVHRKTLNPVFNEQFTFKVPYSELGGKTLMAYVFDRFSKHDIIIGEFKVPMTVD-F |
| TKTIKCSLNPEWNETFRFQLKESKD-K--RRLSVEWWDLSRDTSRNDFMGSLSFQISELQKA |

A gap is placed in one sequence (say A) to match an amino acid in the second sequence (say B). It indicates a missing amino acid in the A sequence when comparing it to the B sequence. Note that it is not possible for us to decide if an amino acid was deleted from A or added to B, without a detailed evolutionary mechanism linking the two proteins to a common ancestor. At the level of alignment of a pair of sequences the two mechanisms are equivalent. It is therefore common to use the term indel to describe a gap (indel = an INsertion or a DELetion), a name that remains undecided about the mechanism of gap formation.

Our goal is to find the best match between two sequences including the possibilities of gaps. A score of a match is given (as argued above) by a sum of scores of matching elements.
What is the score of the gap, i.e. the alignment of an indel against another amino acid? It is philosophically useful to think of a gap as an ordinary amino acid and ask what is the score for substituting an amino acid type “a” with an amino acid type “-“. Such an approach is good in theory and there are some studies following this line of investigation. Unfortunately so far, determining gap scores (also called gap penalties) is still an open question. Formulation of statistical theories of gaps is difficult since the number of gaps in aligned sequences is not known to begin with. This difficulty in determining scores for gaps is in contrast to the substitution matrices of amino acids that are pretty well established.

A common practice in alignment programs is to leave the gap energy as a parameter to be determined by the user, setting a default value of (for example) zero. This is not to say that the score of the gap is insignificant. It clearly influences the alignment. If it is set highly negative, then it is difficult to match related proteins of considerable variation in lengths (remote homologous proteins). If it is set too high, two vastly different proteins may get fragmented to many small overlapping pieces and a large number of gaps. The fragmented alignment of two unrelated proteins (with high score for gaps) may still maintain a good overall score. This is not what we expect, since proteins are related only if a substantial continuous fraction of the two sequences is found to be similar.

High-scoring short-sequence segments can be analyzed statistically to determine their significance. In fact such an analysis is behind the popular BLAST algorithm [x]. BLAST does not consider gaps at all. A statistical test determines if short compatible segments of
aligned sequences are significant. In some sense, BLAST solves the gap problem by avoiding it all together. The BLAST algorithm will be discussed in the section: **Approximate alignments.**

A typical practical estimate of the indel/gap penalty is to give it a single negative value, say $g$, regardless of the pairing amino. Hence, the gap penalty is set to be independent of the type of the amino acid it is aligned to (e.g. the pair $W/-$ score the same as the pair $G/-$). This choice is non intuitive since $W$ (tryptophan) is much larger than $G$ (glycine). The space left for the “-“ residue by the removal of $G$ or $W$ will be therefore different and the cost is likely to be dissimilar. The less detailed treatment of gaps, (compared to usual amino acids), may lead to poor alignments. Therefore a few suggestions of extending the simple model of gap penalty were proposed.

One popular model is to differentiate between opening and extension of gaps. It is based on the argument that gaps should aggregate together. Claims were made that insertions and deletions are likely to appear at certain structural domains of proteins (mostly loops). The concentration of gaps at a few structural sites makes the indels appear together, and aggregate. To maximize the size of groups of gap (and minimize the number of gap clusters), two gap penalties are assigned: One penalty is for initiating a gap group. A second (lower penalty) is for growing an existing cluster of gaps. For example, (we construct our alignment from left to right), extending an alignment from $a/-$ to $aa/-$ is less favorable than the extension of the pair $a/-$ to $aa/-$. This is regardless of the fact that in both cases the same pair $(a/-)$ was added to an existing alignment.
This model improves the overall appearance of the alignments. However, the present author
is not enthusiastic about it. This model is highly asymmetric with respect to other “amino
acids” (remember, we would like to consider a gap to be an amino acid), and it is not
obvious that the asymmetry is indeed required. It is also making the identification of optimal
alignment messier. Alternative modeling of gaps (so far less popular) will be discussed later
and include structural dependent gap. For the moment we shall concentrate on the simplest
model of gaps (one value does it all), and after solving that problem, the effect of the
extensions will be examined.

We wish to place the gaps in such a way that the alignment or the comparison of the two
sequences will be “optimal”, that is that the two sequences will be as similar to each other as
possible. Early studies of sequence comparisons found optimal alignments manually. Gaps
were inserted “by hand” into positions that made biochemical sense and increased the
number of good-looking pairs [x]. For an implementation on the computer we consider an
optimal alignment to be an arrangement of the two sequences with respect to each other in
such a way that the total score of the alignment is as high as possible.

Let us examine in more details the problem of optimal alignments and possible
arrangements of sequences. There are many ways of aligning a sequence \(A\) against a
sequence \(B\) once gaps are introduced (even if the original order of the amino acids is
maintained). Gaps can enter anywhere and in wide range of numbers in the alignment,
creating many alternative arrangements. We denote the extended sequences of \(A\) and \(B\)
(with gaps) as $\overline{A}$ and $\overline{B}$. For example, $\overline{A}$ might be $\overline{A} = a_1 a_2 a_3 \cdots a_n$. The introduction of the extended sequences raises another intriguing question, what is the maximum length of the sequences $\overline{A}$ (or $\overline{B}$)?

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

\[
a_1 \quad - \quad \ldots \quad a_{n-1} \quad - \quad a_n \\
- \quad b_1 \quad \ldots \quad - \quad b_{n-1} \quad b_n
\]

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

\[
a_1 \quad - \quad \ldots \quad a_{n-1} \quad - \quad a_n \\
- \quad b_1 \quad \ldots \quad - \quad b_{n-1} \quad b_n
\]

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.
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:

\[
\begin{array}{cccccc}
  a_1 & a_2 & a_3 & a_4 & a_5 \\
  b_1 & b_2 & b_3 & - & b_4 & b_5 \\
\end{array}
\]

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:

\[
\begin{align*}
&\begin{array}{c}
12 \\
\end{array} & \begin{array}{c}
12 \\
\end{array} & \begin{array}{c}
12 \\
\end{array} & \begin{array}{c}
a \\
\end{array} & \begin{array}{c}
a \\
\end{array} & \begin{array}{c}
a \\
\end{array} \\
&\begin{array}{c}
12 \\
\end{array} & \begin{array}{c}
11 \\
\end{array} & \begin{array}{c}
1 \\
\end{array} & \begin{array}{c}
1 \\
\end{array} & \begin{array}{c}
1 \\
\end{array} \\
\end{align*}
\]

(1) $a_1 a_2 - b_1$ (2) $a_1 b_1 - a_2$ (3) $a_1 - a_2 b_1$ (4) $a_1 - b_1 - a_2$ (5) $a_1 - a_2 - b_1$

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 \(b_1\). The last alignment can be done in three different ways and therefore the table entry is “3”

\[
\begin{array}{ccc}
a_i & - & a_i \\
- & b_1 & b_1 & - & b_1
\end{array}
\]

If we add the number of paths starting at the previous three “sources”, the number of alignments of \(a_2a_2\) 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

\[
N(n,m) = N(n-1,m-1) + N(n-1,m) + N(n,m-1),
\]

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:

\[
N'(n,m) = \binom{n+m}{m} = \frac{(n+m)!}{n!m!}.
\]

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: \( \log \lfloor n! \rfloor = n \log n - n \lfloor x \rfloor \), valid for large \( n \)-s.

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

\[
\log \left[ \frac{(2n)!}{n!^2} \right] = \log \left[ (2n)! \right] - 2 \log [n!] = 2n \log (2n) - 2n - 2n \log (n) + 2n = 2n \log (2)
\]
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 ($10^{-9}$ second), which is unrealistically fast, it still necessary to use $10^{31} \text{ sec} \approx 10^{13} \text{ years}$ 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.

1.3 Dynamic programming and optimal alignments

After this long detour, it is about the time to return to the basic question: What is the optimal alignment of $\bar{A}$ and $\bar{B}$? The score matrix and the gap penalty are provided and we need is
to fish out the alignment(s) with the highest overall score(s). This is a point in which algorithms developed by computer scientists can be extremely useful. The fact that the number of possible alignments is exponentially large in the sequence length does not mean that the search for the optimal alignment needs to be done in exponential time. Sequence alignments can be done with dynamic programming, an algorithm that requires only order of \( n^2 \) operations to find the alignment with the best score, a remarkable saving compared to \( 2^{2n} \) - the number of all possible alignments.

The efficient search for the optimal alignments consists of two steps. In the first step a dynamic matrix is constructed (with different entries than what we have seen before) and in the second step an optimal path is found in the newly constructed table. The first step is similar to the counting of possible alignments and the recursive expression we use to compute the total number of alignments. We denote the optimal score of the alignment of a sequence length \( n \) against a sequence with length \( m \) by \( T(n,m) \), and consider the following question:

Assuming that a very kind fellow gave us the optimal scores for the following alignment: \( T(n-1,m-1) \), \( T(n-1,m) \), \( T(n,m-1) \), can we construct the score \( T(n,m) \) ?

The answer is yes. Since the total score is given by a sum of the scores of the individual aligned pairs, we construct the score \( T(n,m) \) from the three alignments leading to it. We consider three possibilities to obtain an alignment of \( n \) against \( m \) amino acids.
Option a: Align $n-1$ against $m-1$ amino acids (this alignment has the known optimal score of $T(n-1,m-1)$). Extend it by the alignment of $a_n/b_m$ with a score of $S(a_n,b_m)$ which is the substitution score according to the types of the amino acids at $a_n$ and $b_m$ (e.g., the BLOSUM matrix). Hence the first suggestion for an optimal score is $T(n-1,m-1)+S(a_n,b_m)$.

Option b: Align $n$ amino acids against $m-1$ amino acids. This alignment also has a known score, which is $T(n,m-1)$. Extend this alignment by $-b_m$ with a corresponding score $g$ for a gap. The second possibility for an optimal alignment is therefore $T(n,m-1)+g$.

Option c: Align $n-1$ amino acids against $m$ amino acids with the known (optimal) score of $T(n-1,m)$. Extend the alignment, by adding $a_n$ with a corresponding score of $g$. The third suggestion is therefore $T(n-1,m)+g$.

Note that we use the simple model of gaps in which only a single score is associated with an indel, regardless of the amino acid it is aligned against.

Our final task is to select the highest score from one of the three alternatives:

$$T(n-1,m-1)+S(a_n,b_m), \ T(n,m-1)+g, \ T(n-1,m)+g,$$

which is the optimal score of $T(n,m)$.

More compactly, we write: $T(n,m) = \max \begin{bmatrix}
T(n-1,m-1)+S(a_n,b_m) \\
T(n,m-1)+g \\
T(n-1,m)+g
\end{bmatrix}$
The above recursion can be used to fill with optimal scores the complete dynamic matrix. We start with the condition $T(1,-) = T(-,1) = g$ and use the initial values to grow the matrix. For example, $T(a_i, b_i) = \max \left[ \begin{array}{c} T(a_i, -) + g \\ T(-, b_i) + g \\ T(0,0) + S(a_i, b_i) \end{array} \right] = \max \left[ \begin{array}{c} 2g \\ 2g \\ S(a_i, b_i) \end{array} \right]$

Where $T(0,0)$ denotes alignment of nothing that (not surprisingly) scores zero. Another simple example is of $T(n,n(-)) = n \cdot g$. Hence, there is only one alignment against “all gaps” arrangement, providing us with immediate optimal scores for the first row and column of the dynamic matrix. It is also clear that similarly to the direct counting of the number of paths we can also construct all the optimal scores.

\[
\begin{array}{cccccc}
- & a_1 & a_2 & a_3 & a_4 & a_5 \\
0 & g & \rightarrow & 2g & \rightarrow & 3g & \rightarrow & 4g & \rightarrow & 5g \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
b_1 & g & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
b_2 & 2g & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
b_3 & 3g & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
b_4 & 4g & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
b_5 & 5g & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow & \rightarrow \\
\end{array}
\]

A simple pseudo code to create the dynamic matrix is given below
/* fill the first (zero) column and the first (zero) row */

T(0,0) = 0

Do I=1:n
    T(I,0) = I*g
End do

Do I=1:m
    T(0,I) = I*g
End do

/* Now fill the rest of the matrix picking the maximum value from the three possibilities */

Do I = 1:n
    Do J = 1:m
        T(I,J) = max[ T(I-1,J)+g,
                      T(I,J-1)+g,
                      T(I-1,J-1)+S(a(i),b(j))]  
    End do
End do
Actually, if we are primarily interested in the score of the alignment of \( A = a_1...a_n \) with respect to \( B = b_1...b_m \) only a single element of the dynamic matrix, \( T(n,m) \), is of interest. Of course, in order to get at it we need to compute first the whole matrix. However this score, which is recorded at the lower right side of the dynamic matrix, is only part of the story.

Besides the score, in many cases, it is important to know the alignment itself. The path in the dynamic matrix that corresponds to the optimal alignment can be found by a trace-back procedure. Starting from \( T(n,m) \) we ask “which of the three possible steps could have generated the final optimal score? That is, we examine the three possibilities

\[
\begin{align*}
T(n,m) &= T(n-1,m-1) + S(a_n, b_m) \\
T(n,m) &= T(n-1,m) + g \\
T(n,m) &= T(n,m-1) + g
\end{align*}
\]

For at least one of the tests above the equality will hold. It is possible that more than one test is correct and in that case the optimal alignment is degenerate. Hence, there is more than one alignment that provides an optimal score. Usually we consider only one path. For example, if the first test is correct the alignment ends with the pair: \( a_n \) and \( b_m \). We repeat the three tests to find the next path segment, this time starting from \( T(n-1,m-1) \). The process is repeated until it reaches the upper left corner and provides the desired optimal alignment. The maximum number of times that the process is repeated is \( n+m \) (all the steps are horizontal or vertical) and the smallest number of steps is either \( n \) or \( m \), the larger number of the two.
We are also ready for a rough estimate of the computational effort. As we discussed earlier a lower bound on the number of possible alignments is $2^{n+m}$. Examining all possible alignments will require computational effort that is growing exponentially with $n+m$. The computations that we just described, using dynamics programming, are much cheaper. Let us consider the two steps separately. In the first step we create the dynamic matrix $T(n,m)$. To generate a single element we need (about) four operations: Evaluating three expressions, and a decision which of the three is the largest. The calculation of a single element is therefore independent of $n$ or $m$, and the cost associated with the computation of the matrix is proportional to $n\cdot m$ (the number of matrix elements). To trace the path of alignment (once the matrix is known) takes a maximum of $n+m$ operations.