Needleman-Wunsch algorithm for DNA sequence alignment (2 sequences)

Update (May 26th 2014): See the RESTful web service implementation of this algorithm here

Given two DNA sequences, you are asked to align these two sequences where for each non-matching nucleotide pair you will be penalized by 1 point and for each gap (space) you insert into a sequence to shift a portion of the sequence, you will be penalized by 2 points. The requirement is that you will have to do this alignment with the least amount of penalties as possible (thus minimizing the total edit distance between these two sequences).

Here is an example:
Align:
GC
C

First possible alignment:
GC
C-
edit distance: mismatch(G,C): 1 point penalty + gap (C,-): 2 point penalty = 3

Second possible alignment:
GC
-C
edit distance: gap (G,-): 2 point penalty

Since the second alignment gives us the smaller edit distance (least total penalties), the second alignment is optimal.

This problem can be solved for any pair of sequences of arbitrary sizes using a dynamic programming approach that is known as the Needleman-Wunsch algorithm published in 1970.

This algorithm is used for the *global alignment* problem in bioinformatics, where the sequences are of more or less the same size and largely similar. The gist of the algorithm is to divide the problem into smaller sub-problems (align prefixes and build the alignment of the whole sequences based on that). The advantage of dynamic programming is that it computes the solution to a sub-problem only once.

If the sequences are of significantly different sizes and/or mostly dissimilar, *local alignment* algorithms are used to identify the sub-sequences that are similar. See the Smith-Waterman algorithm for local alignment.

These algorithms are so common in bioinformatics--they are part of many online toolkits. Likewise, many implementations in Java, Python and other languages can be found online. My suggestion is that you implement the algorithm from scratch to appreciate its elegance and to convince yourselves that it indeed works.

Here is my implementation of the Needleman-Wunsch algorithm in Java:

Helper classes: SimpleAlignmentParameters and AlignmentResult:

Here is the result for a pair of DNA sequences where the gap penalty is 2 and substitution penalty is 1:

Here is how the algorithm finds the solution by backtracking the steps on the score matrix:

The algorithm constructs the alignment in reverse order, a diagonal move means match (black arrow) or a mismatch (red arrow) and a vertical or horizontal move (green arrow) means insertion of a gap either on the first (vertical) or the second sequence (horizontal). Note here that, the first sequence is placed horizontally (running from left to right) and the second sequence is placed vertically (running from top to bottom) on the score matrix.

Question: do you think this is the only optimal alignment for this given pair?
show answer

Here is the code that creates the result given above:

Test your understanding of the algorithm :
-If all you wanted was the optimal alignment score (edit distance, smallest total penalties) but not the aligned sequences themselves, where in the score matrix would you look to retrieve this score?

-How do you prove that the optimal alignment score computed by this algorithm is indeed optimal (ie. the smallest edit distance between the two sequences) ?

-Can you calculate this optimal alignment score without using O(NxM) space for the score matrix (N,M are lengths of the sequences)? How many rows would you need on the score matrix?

A couple of ideas to explore:
-How do you modify the algorithm to print out all optimal alignments instead of just one?

-How easy is it to parallelize this algorithm? What would be the runtime/space complexity of such an algorithm?

-How to modify the Needleman-Wunsch algorithm to perform local alignment?

-What happens if you modify the gap and substitution penalties? What are biologically plausible ways to extend these penalties for DNA sequence matching in real applications?

-Can you improve the space complexity of the algorithm to be linear instead of quadratic? (Hint: Look up Hirschberg's algorithm that was published in 1975)

-Can you extend this algorithm to align 3 DNA sequences at once? How about N DNA sequences?

Resources:
- Check out http://amrita.vlab.co.in/?sub=3&brch=274&sim=1431&cnt=1 for more information on the biology side as well as the algorithm itself

-Dynamic programming and sequence alignment :
http://www.ibm.com/developerworks/opensource/library/j-seqalign/index.html