BLG 336E Analysis of Algorithms II, Spring 2016
Project #2

April 5, 2016

Problem Definition

Sequence Alignment
A sequence alignment is a scheme of writing one sequence on top of another where the residues in one position are deemed to have a common evolutionary origin. If the same letter occurs in both sequences then this position has been conserved in evolution. If the letters differ it is assumed that the two derive from an ancestral letter (which could be one of the two or neither). Homologous sequences may have different length, though, which is generally explained through insertions or deletions in sequences. Thus, a letter or a stretch of letters may be paired up with dashes in the other sequence to signify such an insertion or deletion. Since an insertion in one sequence can always be seen as a deletion in the other one frequently uses the term "indel". For an example sequence alignment of letters see the given below.

BANANA-
- ANANAS

In such a simple evolutionarily motivated scheme, an alignment mediates the definition of a distance for two sequences. One generally assigns 0 to a match, some negative number to a mismatch and a larger negative number to an indel. By adding these values along an alignment one obtains a score for this alignment. A distance function for two sequences can be defined by looking for the alignment which yields the minimum score. Luckily, using dynamic programming this minimization can be effected without explicitly enumerating all possible alignment of two sequences.

Dynamic programming algorithms are recursive algorithms modified to store intermediate results, which improves efficiency for certain problems. The Needleman-Wunsch algorithm uses a dynamic programming approach to find the optimal global alignment of two sequences — a and b. The alignment algorithm is based on finding the elements of a matrix S where the element $S_{i,j}$ is the optimal score for aligning the sequence $(a_1, a_2, ..., a_i)$ with $(b_1, b_2, ..., b_j)$. Two similar amino acids (e.g. arginine and lysine) receive a high score, two dissimilar amino acids (e.g. arginine and glycine) receive a low score. The higher the score of a path through the matrix, the better the alignment. The matrix S is found by progressively finding the matrix elements, starting at $S_{1,1}$ and proceeding in the directions of increasing i and j. Each element is set according to:

$$S_{i,j} = max \begin{cases} S_{i-1,j-1} + s_{i,j} \\ S_{i-1,j} - d \\ S_{i,j-1} - d \end{cases} \tag{1}$$

where $s_{i,j}$ is the similarity score of comparing amino acid a_i to amino acid b_j and d is the penalty for a single gap (see also the Figure 1). In order to understand how to fill the matrix to compute the dynamic programming algorithm you can see the step-by-step example given **here**.

You need to implement Needleman-Wunsch algorithm for global alignment of DNA sequences for this project.

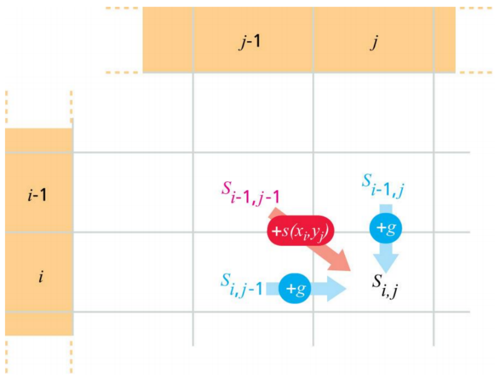

Figure 1: How to calculate $S_{i,j}$ using subsolutions

Report and Code

(a) [**30 points**] Given the DNA sequences **ISALIGNED** and **THISLINE**, find the optimal global alignment and its score. Matches, mismatches, and gaps are scored as 4, -1, and -4, respectively. In your report, you should also draw the matrix to obtain the global alignment and show the traceback step on this matrix. There is no need for coding for this part of the project, show them just on paper.

(b) You are given two input files: **human-hemoglobin-sequence.fasta** and **mouse-hemoglobin-sequence.fasta**. These include the DNA sequences of two homolog genes, human hemoglobin and mouse hemoglobin. In an ordinary fasta file, the first line gives the information about the sequence and the following lines give the sequence itself.

[**40 points**] Your program should read the fasta files and calculate the global alignment of two sequences. Please keep in your mind that there is no difference between alignment of DNA sequences and amino acid sequences. You should use the substitution matrix BLOSUM62 which is given along with other project files for match, mismatch and gap values instead of using the same values for all (e.g. $\{$*match between C and C*$\} \neq \{$*match between G and G*$\}$ according to BLOSUM62).

[**30 points**] (1) Write your pseudo code for the algorithm. (2) Find the complexity of your algorithm and discuss the difference between assessing all possible alignments one-by-one and applying this dynamic programming approach. (3) Print your sequence alignment result. Give the summary of your work in a template given below. (*Length*: length of your final alignment, *Similarity*: similarity between sequences if you only count the exact matches, *Identity*: identity between sequences if you count positively scored mismatches besides exact matches, *Gaps*: number of gaps and *Score*: final global alignment score)

```
########################################################################
#
# Length: 145 (with gaps)
# Identity:      85/145 (58.6%) ("|")
# Similarity:   104/145 (71.7%) ("|" and ":")
# Gaps:           1/145 ( 0.7%)
# Score: 433.0
#
#
#=======================================================================

human              1 MLDQKTRDIIKSTVPALKTHGLEITTTFYKNMFANNPEVKPLFNMDKQDS     50
                     |||.|.:.|||..||.|:..|:|:|..||..||.|||||:.||:.|:|.|
mouse              1 MLDYKVKCIIKDCVPVLRDRGVELTENFYNLMFTNNPEVRSLFDDDRQKS     50

human             51 GEQPKALAMTILAAAQNIDNLPAILPVVKKIGEVHCNKEIQAEHYPIVGK    100
                     |||.||||.::||.|||||||..|||.|||||..|....::.||||||||
mouse             51 GEQAKALASSLLAVAQNIDNLEKILPTVKKIGMSHVKANVKPEHYPIVGK    100

human            101 NLLLTIKEVLGDAATDEVMEAWEKAYQVIADIFISVEKEIYETRK    145
                     ||||.|||.|||.||||::.|:.:.|:.|:.|||.:|||:|.|.
mouse            101 NLLLAIKETLGDTATDELINAFSEVYKYISKIFIDIEKELYNTI-    144

########################################################################
```

Submission

You should be aware that the Ninova system clock may not be synchronized with your computer, watch, or cell phone. Do not e-mail the teaching assistant or the instructors your submission after the Ninova site submission has closed. If you have submitted to Ninova once and want to make any changes to your report, you should do it before the Ninova submission system closes. Your changes will not be accepted by e-mail. Connectivity problems to the Internet or to Ninova in the last few minutes are not valid excuses for being unable to submit. You should not risk leaving your submission to the last few minutes. After uploading to Ninova, check to make sure that your project appears there.

Policy: You may discuss the problem addressed by the project at an abstract level with your classmates, but you should not share or copy code from your classmates or from the Internet. **You should submit your own, individual project**. Plagiarism and any other forms of cheating will have serious consequences, including failing the course.

Submission Instructions: Please submit your homework through Ninova. Please zip and upload all your files using filename Project2_studentID.rar. In the archived file, you must include your completed Report_studentID file and all your program and header files.

All your code must be written in C++, and we must be able to compile and run on it on ITU's Linux Server (you can access it through SSH) using g++. You should supply one studentID.cpp file that calls necessary routines for all questions (Multiple files are acceptable, as long as you state the compilation instructions in your report).

When you write your code, try to follow an object-oriented methodology with well-chosen variable, method, and class names and comments where necessary. **Your code must compile without any errors; otherwise, you may get a grade of zero on the assignment.**

If a question is not clear, please let the teaching assistant know by email (yildirimez@itu.edu.tr)