

CS329E

Spring 2007

HW5

Assigned: 3/19/06

Due:

Administrative Task Due: 3/21/07

Homework problems due: 3/25/07

Warning: you will have a small platform related assignment due 3/27/07

Administrative Task: The programming project is best done in teams of two (unless you already are comfortable as an applications developer). Find yourself a partner and email the pairing to the T.A. **Include your names and your majors.**

Final pairings are subject to Professor Miranker's approval. My goals (class wide constraints) for the teams include, biology majors should not work in pairs. Nonbiology related students should not work in pairs. While it will not be a hard constraint, people who are clearly accomplished programmers should not work in pairs.

You may do this homework in teams of two. If you choose to do this work as a team additional instructions apply. Those instructions are detailed last in this document, but will impact your work from the beginning (i.e. be sure to read the whole assignment prior to deciding if you want to form a team). As usual, you are welcome to discuss the homework in larger study groups. You may not share any on-line representation of answers.

Part I: Modeling Local Alignments

In anticipation of the Rosetta Analysis and what is otherwise a very common biological data model problem, consider a set of sequences, their annotation, (i.e. genbank feature table) and recording in a database the result of a local alignment.

Assume sequences are gene (nucleotide) sequences. Each sequence will have, an accession number (unique-id assigned by genbank), the length of the sequence, the organism, a name, a functional description. Sequences have features. A feature has a name and an identifying interval in the original sequence, beginning offset, ending offset. Local alignments for a pair of sequences must record a total score, a list of matches, (the matching subsequences from each sequence).

In the style of the previous homework, develop a data model and forward engineer the DDL.

Part II: Programming Alignments

On the class web site you will find Java code for computing edit distance. You are welcome to use and modify this code.

A. Implement a computer program that computes simple edit distance. Use the recurrence and cost function in lecture. Turn in your code and an example run computing the edit distance between

- a) peter and pepper
- b) AAACCCG and AAGCGC
- c) their and their
- d) their and there

B. Similarly, implement a global-alignment algorithm, (both score and alignment), per the traditional bioinformatics definition (similarity). Use the cost function defined below. One alignment is all that is required.

This programming assignment comprises three parts beyond question A

1. Changing the filling out of the dynamic programming matrix (score)
 - a. Change the boundary condition (0's in the first row and first column)
 - b. A more detailed cost function. Hint: Make the cost function and explicit method **cost(char,char)**.
2. Introduce backtrace and output an alignment (alignment)
 - a. An implementation of the backtrace comprises instantiating a matrix of the same size as the dynamic programming matrix, call it `thewinner[][]`. Record in `thewinner[i][j]`, if the winning case for the dynamic programming matrix cell, `distance[I][j]`, was the up, left, or diagonal.
 - b. For the alignment, it will be easiest to build the alignment and the output it. (recall, we have to build the alignment from right to left. Unless you do some very fancy I/O, print statements output from left to right). A way to implement this is to define two character arrays, one for each sequence. Then you can write a loop that traverses the `thewinner` matrix. Starting from the highest scoring positing, at each step, up, left or diagonal, you can write into the character arrays the character that should appear in the output.

Extra Credit: Emit all alignments with the best score.

Partial Credit: Output the score of the global alignment, but not the alignments themselves. (i.e. if you can't get backtrace to work, still turn in something)

Cost function:

- matching characters are rewarded 1,
- mismatching characters -0.8,
- indels -0.9.

i.e. $c(v_i, w_j) =$

$$\begin{aligned}
&1 \text{ if } v_i = w_j \\
&-0.8 \text{ if } v_i \neq w_j \\
c(_, w_j) = c(v_i, _) = -0.9
\end{aligned}$$

The recurrence is

$$S(0, 0) = 0$$

$$S(i, j) = \max(S(i-1, j-1) + c(v_i, w_j), S(i, j-1) + c(_, w_j), S(i-1, j) + c(v_i, _))$$

The answer for “hello” and “yellow” is 2.3 with the alignment

hello_
yellow

The following is provided for your convenience. Do not hand in the computed matrices, (nor is it required that your program print them, but it might help you debug.),

		H	E	L	L	O	
	0.0	0.0	0.0	0.0	0.0	0.0	
Y	0.0	-0.8	-0.8	-0.8	-0.8	-0.8	
E	0.0	-0.8	0.2	-0.7	-1.6	-1.6	
L	0.0	-0.8	-0.7	1.2	0.3	-0.6	
L	0.0	-0.8	-1.6	0.3	2.2	1.3	
O	0.0	-0.8	-1.6	-0.6	1.3	3.2	
W	0.0	-0.8	-1.6	-1.5	0.4	2.3	

Test your code on a-d above. Turn in the results of running your code on,

- e) peter and piper
- f) peter and pepper
- g) betty and botter
- h) botter and butter
- i) bettybotter and bitterbutter