

BCB 567/CprE 548 Bioinformatics I
Fall 2007
Exam 1 Solutions

General Grading Principles: If for any question on the exam, you wrote down a statement that is *not true* for a problem, you were docked half a point for that problem. Equivalent solutions have been given equivalent number of points.

1. (a) The best local alignment is:

CC-GA
CCAGA

- (b) **Correct Answers:** There are many valid ways to answer this question. The easiest answer is to decrease the gap score to a very low value. This causes an optimal alignment to be of length 2. Another easy answer is to choose $\gamma > \alpha$. This causes an optimal alignment with only gaps.

Incorrect Answers: It was incorrect to try to change the parameters to cause no mismatches. There are no mismatches in the optimal alignment. However, you were give partial credit based upon your reasoning.

2. **Correct Answers:** Two interpretations of the problem were given full credit. In one interpretation, you must have a *proper* suffix–prefix alignment. In this interpretation, you are not allowed to use the entirety of either string. The second correct interpretation allows the use of the entirety of a string (e.g. a prefix of \mathcal{A} can be \mathcal{A} itself). This second interpretation still requires using a suffix of \mathcal{A} and a prefix of \mathcal{B} .

The difference between these two interpretations is mostly in the initialization of the table. For the interpretation requiring proper suffixes and prefixes, the first row of the table is initialized to $-\infty$, while for a interpretation allowing all of \mathcal{A} to be used, we initialize the first row as we would for global alignment. In both cases, the first column is initialized with zeros.

To find a suffix-prefix alignment, you must find the maximum in the last row only, and then trace back to any zero in the first column.

Incorrect Answers: You were docked points if you did not discuss how initialization was different, if you did not discuss where to look for the maximum, if you tried to use local alignment. If you included a correct picture, you received a minimum of 1.5 points.

3. **Correct Answers:** The correct answer to this problem involved inventing a function:

$$\omega(x) = \begin{cases} 0 & x = \text{'T'} \\ \gamma & \text{otherwise} \end{cases}$$

The initialization of the table becomes:

$$\begin{aligned}S[0, 0] &= 0 \\S[i, 0] &= S[i - 1, 0] + \omega(\mathcal{A}[i - 1]) \\S[0, j] &= S[0, j - 1] + \omega(\mathcal{B}[j - 1])\end{aligned}$$

The recursion becomes:

$$S[i, j] = \max \begin{cases} S[i - 1, j - 1] + \delta(i - 1, j - 1) \\ S[i - 1, j] + \omega(\mathcal{A}[i - 1]) \\ S[i, j - 1] + \omega(\mathcal{B}[j - 1]) \end{cases}$$

Credit was given if instead of having the ω function, you wrote out all cases. Credit was given if you did not correctly index the strings (for example using $\mathcal{A}[i]$ instead of $\mathcal{A}[i - 1]$). Credit was given if you correctly described the ω function in words instead of using mathematical notation. If you correctly described the recursions, any reasonable description of traceback resulted in full credit.

Incorrect Answers: Points were docked if you did not recognize the need to do special initialization. Points were docked if you created an ω function that required input from both strings. All other people losing points on this problem did not see the solution. In all cases, some partial credit was given.

4. (a) **Correct Answers:** The key to getting full credit on this problem was the following. First, you must recognize that this is a special application of semi-global alignment. Second, you must recognize that we cannot allow gaps and mismatches in the alignment. The easiest way to prevent gaps and mismatches is through by choosing $\beta = -\infty$, $\gamma = -\infty$, and $\alpha > 0$. I will discuss the solution using these parameters, although there were other correct solutions that did not follow this outline exactly.

To construct the shortest super-string, we must first fill out the table using semi-global alignment, as normal. Then we must look for the maximum value in the last row or column as usual. If the two strings have an exact suffix-prefix or containment overlap, that value will be positive, otherwise it will be zero.

Using the found maximum, we can create a path through the table that starts at the lower right cell, travels to the found maximum, (possibly) moves diagonally to the other side of the table, and then continues to the upper left corner.

This path can be used to directly write out the superstring in the following way. When tracing the path, we write out a character from \mathcal{B} for any horizontal move, a character from \mathcal{A} for any vertical move, and arbitrarily choose to write out a character from \mathcal{A} or \mathcal{B} for diagonal moves.

Credit was given for any correct description of how the path corresponds to the super-string using diagrams and/or words. In general, any correct solution similar to what was described here was given full credit.

Incorrect Answers: You lost points if you did not correctly identify the need to disallow gaps and mismatches in the alignment. You lost points if you tried to use local alignment. You lost points if you did not describe how the table corresponds to a superstring.

(b) **Correct Answers:** The shortest common superstring is not unique. Consider the simple case $\mathcal{A} = x, \mathcal{B} = y$ then $\mathcal{C} = xy$ or $\mathcal{C} = yx$ are both answers.

Incorrect Answers: This question was all or nothing. No partial credit was given.

5. **Correct Answers:** There are two ways in which to answer this problem correctly.

The first is to create two S tables. For S_1 , we run global alignment using the given α, β , and γ . For S_2 , we set $\beta = -\infty$ (or if you wish, $\beta < 2\gamma$). In this way, we will never choose a mismatch to get the optimal score. Now we use the following assignment:

$$Q[i, j] = \begin{cases} 1 & \text{if } S_1[i, j] = S_2[i, j] \\ 0 & \text{if } S_1[i, j] \neq S_2[i, j] \end{cases}$$

The second correct answer is to build $Q[i, j]$ directly using dynamic programming.

$$Q[0, 0] = 1$$

$$Q[0, j] = 1$$

$$Q[i, 0] = 1$$

$$Q[i, j] = \begin{cases} 1 & \text{if } Q[i-1, j-1] = 1 \text{ AND } S[i-1, j-1] + \alpha = S[i, j] \\ 1 & \text{if } Q[i-1, j] = 1 \text{ AND } S[i-1, j] + \gamma = S[i, j] \\ 1 & \text{if } Q[i, j-1] = 1 \text{ AND } S[i, j-1] + \gamma = S[i, j] \\ 0 & \text{otherwise} \end{cases}$$

The exact form of the recursive equation can be slightly different and still result in full credit, although this is the simplest form. If you correctly gave a recursive formulation without giving the base case, you lost half a point. There was a third answer that involved a complicated traceback function that was also generally correct.

Incorrect Answers: The main error that people made if they were close to a correct answer was to mishandle the fact that we want $Q[i, j]$ to be 1 if *any* optimal path exists that does not contain a mismatch. Some people instead answered the question: $Q[i, j] = 0$ if and only if there exists a path that contains a mismatch, which, while similar to what was asked, was not the correct answer. People who made this mistake still received a high score.

If you answered the question: $Q[i, j]$ is 1 if and only if there exists an optimal alignment such that the last move is a not a mismatch, you received a score of 1.5.