

Algorithms in Bioinformatics

close-book midterm exam

November 25, 2003

Instructions

1. Please write down your name and ID on each piece of your answer sheets. Cheating will be most seriously punished. Any dishonest attempt in this exam implies an 'F' as your final grade.
2. Throughout the exam, let S denote the 16-bit binary string obtained from the odd digits of your ID by the following rules.

$0 \rightarrow 0000, 1 \rightarrow 0001, 2 \rightarrow 0010,$
 $3 \rightarrow 0011, 4 \rightarrow 0100, 5 \rightarrow 0101,$
 $6 \rightarrow 0110, 7 \rightarrow 0111, 8 \rightarrow 1000,$
 $9 \rightarrow 1001.$

Let Y be the reverse string of X . For example, if your ID is "B7*5*6*4*", then your X is 0111010101100100 and your Y is 0010011010101110.

3. You may use anything of our lectures as a subroutine to your answers, unless you are explicitly asked to explain something we explained in class.

Problem 1 (20 points)

- (5 points) Write down your strings X and Y . (Do "double check" your answers, since lots of the following problems depend on your answers to this subproblem.)
- (5 points) Give the Z value for each position of your binary string X .
- (10 points) Give the "left-guard" $L(i)$ and "right-guard" $R(i)$ of each position i for your binary string Y .

Problem 2 (20 points)

Write down ALL longest common substrings of your X and Y . That is, if there are k distinct longest common substrings of your X and Y , you have to write down all k substrings.

Problem 3 (20 points)

- (10 points) Write down ONE longest common subsequence of your X and Y .
- (10 points) What is the edit distance between your X and Y ?

Problem 4 (20 points)

You are asked to design a linear-time algorithm for computing a longest common substring for k strings. Specifically, given k binary strings S_1, S_2, \dots, S_k , your algorithm should run in $O(|S_1| + |S_2| + \dots + |S_k|)$ time and output a longest string C that occurs in each S_i with $1 \leq i \leq k$. (You may NOT assume $k = O(1)$. That is, it is important that the running time of your algorithm does not depend on k .) Don't forget to justify the correctness and time complexity of your algorithm.

Problem 5 (20 points)

Explain why the problem of finding a longest common subsequence of two strings can be *reduced* to the problem of finding a longest increasing subsequence. That is, explain how to solve the longest common subsequence problem using an algorithm for longest increasing subsequence as a subroutine. Don't forget to justify your answer.