Biol 591 Introduction to Bioinformatics (Fall 2002): Problem Set 3

This problem set is smaller than usual (bearing in mind that it combines questions concerning bioinformatics, molecular biology, and programming) because of the heft of the study questions. Please understand that study questions are to be given the same consideration as questions on problem sets.

P3.1. What can you use to determine whether a string of characters in a Perl program is:

a. a comment **c.** a subroutine **e.** a regular expression
b. a variable **d.** a call to a subroutine **f.** an array

P3.2. Modify BlastN so that it no longer prints out a complete match but prints out instead only each initial exact match of a word.

P3.3. Examine BlastN and determine the values used for the following quantities:

a. Match reward **c.** Gap open penalty **e.** Word size
b. Mismatch penalty **d.** Gap extension penalty

P3.4. Modify BlastN so that it prints out for each hit both the raw score and the score in bits. To do this you may need to find values for lambda and K. Do this by running ANY pairwise sequence comparison at the NCBI site, using the same parameters you use in local BlastN, and noting the values of lambda and K at the end of the output.

P3.5 Estimate how much more efficient BlastN is than a full Smith-Waterman algorithm. Proceed as follows.

A. Presume that the total time spent by each program is proportional to the number of cells in scoring tables each has to calculate (so your job is reduced to figuring out how many cells that is in each case).

B. Consider a specific case of a comparison of a 100-nucleotide query sequence with the E. coli genome. How big would the Smith-Waterman scoring matrix be?

Don't know how big the E. coli genome is? You have a program that can tell you! Recall that SequenceSearch *reads in the genome of Nostoc in order to search for putative NtcA binding sites. Well, perhaps you can change where it reads in the Nostoc sequence and have it read in the E. coli sequence. Once the sequence is in a variable you can add the line*

print length(variable);

and you have it! (you'll have to put in the right name in place of variable).

C. OK, you got half the job done. Now you need to find out how many cells Blast would need to calculate. First of all, how many word matches would you expect Blast to find? Consider two cases: a word-size of 11 and a word-size of 7.

How do you find how many exact word matches there will be? Again, you have a program for the job – in fact the same program! Modify

SequenceSearch *to read in the E. coli sequence (you may have already done this in Step 2). Make up some 11-bp sequences and have the program search for them. Count how many it finds. Make up some 7-bp sequences and have the program search for them. Count how many it finds. If counting is too difficult, then notice that* SequenceSearch *puts all exact matches in an array. How can you find out how many elements are contained in that array?*

D. For each word match found by Blast, how many cells does it have to calculate while attempting to extend the match forwards and backwards? This is difficult to estimate, but take as typical the last figure in the notes for Monday, September 22.

P3.6. How do you explain the fact that *BlastN* cannot find the evident similarity between DG47 and the *lef* gene?

P3.7. Consider the program Distant_pairs.pl (if you don't already have a copy of the program, you can get it from a link in the 1D array notes for Wednesday, 25 September).

A. Note that one column of the form to the right is labeled $j and the other is labeled $count. Use the form to follow changes in the values of $j and $count in the following manner:

a. Whenever the value of $j changes, write down the new value in the first free cell in the "$j" column. Carry over the current value of "$count" to cell next to the $j cell you just filled in.

b. Whenever the value of $count changes, write down the new value in the first free cell in the "$count" column. Carry over the current value of "$j" to the cell next to the $count cell you just filled in

c. When you've completed all the rows, stop.

B. Pretend you're a computer and go through the execution of the program Distant_pairs.pl from the beginning. I gave you the first three lines of the form for free (a hyphen means that the variable has not yet been initialized).

$j	**$count**
--	--
--	0
1	0