

Computer Science 549 – Computational Biology

Prof. Steven Skiena

Fall 2011

Homework 1

Due Thursday, October 6, 2011

September 21, 2011

Each of the problems should be solved on a separate sheet of paper to facilitate grading. Limit the solution of each problem to one sheet of paper unless otherwise stated.

Many of these problems are deliberately open-ended and vague; do the best you can on them but don't make yourself crazy. Please don't wait until the last minute to look at the problems.

You must do this assignment in groups of 2 to 3 people. Mixed groups of computational and life scientists are best, if possible.

1. Spend some time playing with the Genbank/Entrez database. Write a one page paper about your experiences. Below are representative topics for investigation:
 - (a) Hunt for the sequence of the gene which produces Myoglobin. Why are there so many results – how do they differ?
 - (b) Can you find the complete sequence of some human chromosomes? Where are they?
 - (c) How many organisms have had their genomes completely sequenced? How many of these are mammals, plants, fungi, bacteria, and viruses, respectively.
2. Spend some time playing with the PubMed, the database of all biological/medical literature available at www.ncbi.nlm.nih.gov/PubMed/.

Write a one page paper about your experiences. Below are representative topics for investigation:

- (a) Find an obscure disease, like Stevens-Johnson Syndrome or Dupuytren contracture. Look up this disease on PubMed to see what that most up-to-date treatments are. What is known about the cause of this disease?
- (b) Find an obscure organism (such as the Echidna or Slow Loris) and see what is known about it. You might need to find Latin names for such a beast and search on that.
- (c) How many of Skiena's papers are listed on Pubmed? Are any of these any good? How many of these do not appear in standard computer science bibliographic sources such as Google Scholar, (DBLP) <http://www.informatik.uni-trier.de/~ley/db>, the ACM Digital Library, or the Web of Science database. The later two should be available through the Stony Brook Library webpage.

3. Implement the greedy heuristic for the shortest common superstring in your favorite programming language, and do some experiments with it to get a better understanding of sequence assembly. Your program must take as input a collection of strings (one per line) and return a short superstring of them.

Your program can use the naive algorithm to test pairs of strings for the longest overlap, although feel free to do better if you are brave and have enough time on your hands. Do not run any experiment which will take more than a few minutes.

You should start by writing a program which takes an input sequence of length n and generates simulated fragments with the following parameters: m random fragments, each of length l . There should be no base errors – all fragments should be substrings of the input.

Write a 2-3 page paper answering the following questions. Attach printouts of your programs. Note that it might be more useful to conduct repeated experiments on scaled-down problem sizes than fewer experiments on larger instances to minimize computation time.

- (a) How large a project does it take (as a function of n and m) before reconstruction times starts to be a problem?
 - (b) What coverage do you need before the reconstructed superstring tends to be correct (the same as the input is generated from)?
 - (c) How is the accuracy of reconstruction affected by coverage and the fragment length?
 - (d) How does the reconstruction accuracy differ over random sequences, DNA from Genbank, and English text?
4. Give an algorithm that take in two strings A and B (of lengths n and m) and find the longest suffix of A that exactly matches the prefix of B . This algorithm should run on $O(n + m)$ time.
 5. Explain how to adapt the algorithm for computing the longest palindrome within a string to the notion of “biological palindromes”, where we seek the longest substring of a DNA sequence which is its own reverse complement.
 6. Consider the problem of partitioning a string into a sequence of palindromes (pieces that read the same forward as backwards). Since each single character is a palindrome, this can always be done – but we seek the partitioning into the minimum number of pieces possible. Thus “AMMADAM” can be partitioned into three pieces, “A-M-MADAM” and “ABBABABA” can be partitioned into three pieces as “ABBA-B-ABA”
 - Does the greedy heuristic (find the largest palindrome) always work? Give a proof or a counterexample?
 - Does repeatedly finding the largest left-most remaining palindrome always work? Give a proof or a counterexample?
 - Give an efficient algorithm for finding the optimal partitioning.