CSE 584 Algorithms for Biosequence Comparison Spring Semester 2016

Homework 4

Assigned: 4/7/2016 *Due Date: 4/25/2016*

This homework must be completed and submitted electronically. Formatting standards, submission procedures, and (optional) document templates for homeworks may be found at

`http://classes.engineering.wustl.edu/cse584/ehomework/ehomework-guide.html`

Advice on how to compose homeworks electronically, with links to relevant documentation for several different composition tools, may be found at

`http://classes.engineering.wustl.edu/cse584/ehomework/composing-tips.html`

When the homework is graded, you will receive your marked-up solutions in your SVN repository at

`https://svn.seas.wustl.edu/repositories/yourid/cse584a_sp16/hwk4`

where "yourid" is replaced with your WUSTL Key ID.
Please remember to

- **create a separate PDF file (typeset or scanned) for each problem;**
- **include a header with your name, WUSTL key, and the homework number at the top of each page of each solution;**
- **include any figures (typeset or hand-drawn) inline or as floats;**
- **upload and submit your PDFs to Blackboard before class time on the due date.**

Always show your work.

1. (**34%**) Extend Hirschberg's algorithm for finding an optimal global alignment of two strings in linear space to work with affine gap penalties. Your solution should still run in time $\Theta(nm)$ for sequences of lengths n and m. (*Hint*: is the score of an optimal alignment passing through cell (i, j) still the sum of scores for optimal alignments reaching (i, j) from the two corners of the matrix?)

2. (**33%**) Sketch pseudocode for performing dynamic programming alignment to locate all alignments with at most k differences (single-character substitutions or indels) of a pattern P against the virtual suffix tree of a text T. The alignment should be global with respect to P but local with respect to T. Your pseudocode should print a list of all tree positions where some sufficiently good alignment to P ends.

   Because the suffix tree is only virtual, we represent tree positions by maximal intervals of the suffix array $\tilde{A}$ of T^R. You should implement your traversal in terms of a primitive EXTEND(I) that takes an interval and returns a list (c_j, I_j) of its non-empty child sub-intervals I_j obtained by extending I's label with character c_j.

   Be sure to cut off alignment along a given path when every possible extension would incur $> k$ differences. (You need not implement any clever admissible heuristics.) Also, use an explicit stack and organize your traversal so as to ensure that the stack depth is $\Theta(\log |T|)$.

3. (**33%**) The following problem concerns generate-and-filter strategies that are guaranteed to find all alignments with sufficiently few differences. The setting for the problem is as follows. We are given a query sequence Q of length n and a reference R of length m, and we want to find all occurrences of Q in R with up to d differences (substitutions or indels).

   (a) In class, we showed that an occurrence of Q in R with up to d differences contains a perfect substring match of length at least $k = \lfloor n/(d+1) \rfloor$.

   What is the false-positive rate of this heuristic, i.e. the expected number of chance occurrences of a perfect match of length k between Q and R? Assume that Q and R are unrelated, i.i.d. random sequences with equal base frequencies.

   (b) Baeza-Yates and Perleberg gave the following method to reduce the false-positive rate in the above comparison problem without sacrificing the guarantee of finding all (n, d)-approximate matches. Rather than seek matches to *any* length-k substring of Q in R, seek only matches to substrings $Q[jk + 1..(j + 1)k]$, for integers $j \geq 0$.

   Prove that this revised method still finds every (n, d)-approximate match to Q in R, and compute its false positive rate in the model of part (a). How much of an improvement is the BYP method over naively looking for all substring matches of length k?

   (c) There are many ways to improve on the false-positive rate of BYP. To give just a taste of the possibilities, suppose $n = 24$ and $d = 1$, and suppose we want to find approximate matches of Q in R that differ only by *substitutions*, not indels.

   Consider the pattern $P =$ `xxxxxxx0xxxxxxxx` of length 16. Two 16-mers are said to *match under pattern* P if they agree at every position marked by an x. Positions marked by 0 are "don't-cares," since we don't care whether the strings match at that position or not.

   Prove that every occurrence of the 24-mer Q in R with at most 1 mismatch contains a pair of 16-mers that match under pattern P. If we check for pattern matches starting at each possible position in Q, what is the false-positive rate of this method in the model of part (a)? How does it compare to the rate for the BYP method for this n and d in the limit of very large m?

Fun fact: It can be shown (!) that every occurrence of a 25-mer Q in R with up to two substitutions must contain a pair of 23-mers that match under at least one of the following patterns:

```
xx0x0xx00xxxxxxx0xxxx0x
x0xx00xxxxxxx0xxxx0x0xx
xxxxxxx0xxxx0x0xx00xxx
xxx0xxxx0x0xx00xxxxxxx
xxxx0x0xx00xxxxxxx0xxx
xx00xxxxxxx0xxx0x0xx00x
```

This pattern set is one example of a large class of combinatorial designs described by Kucherov, Noé, and Roytberg in "Multiseed lossless filtration," *IEEE Transactions on Computational Biology and Bioinformatics* 2(1):51-61 (2005).