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

[https://classes.engineering.wustl.edu/cse584/ehomework/ehomework-guide.html](https://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

[https://classes.engineering.wustl.edu/cse584/ehomework/composing-tips.html](https://classes.engineering.wustl.edu/cse584/ehomework/composing-tips.html)

Please remember to

- create a separate PDF file (typeset or scanned) for each problem;
- 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%) 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 $m$ and a reference sequence $R$ of length $n$, and we want to find all
occurrences of $Q$ in $R$ with up to $d$ differences (substitutions or indels).

(a) In class, we sketched a proof that an occurrence of $Q$ in $R$ with up to $d$ differences contains a
perfect substring match of length at least $k = \lfloor m/(d+1) \rfloor$. (This was the argument that in the
worst case, the differences between $Q$ and the approximate match in $R$ are equally spaced along
the alignment.)
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 $(m,d)$-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 $(m,d)$-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 $m = 24$ 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 $P = \text{xxxxxxx0xxxxxxx}$ 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 $m$ and $d$ in the limit of very large $n$?

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:

- xx0x0xx00xxxxxxx0xxxxxx
- x0xx00xxxxxxx0xxxx0x0xx
- xxxxxxx0xxxx0xx00xxx
- xxx0xxxx0xx0xx0xxxxxxx
- xxx0x0xx00xxxxxxx0xxx
- xx00xxxxxxx0xxx0xx00x

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
3. (33%) Consider the following all-substrings alignment problem. We are given sequences \(S[1..n]\) and \(T[1..m]\), and we want to compute a matrix \(C_{m+1 \times m+1}\), s.t. \(C_{ij}\) is the score of an optimal (global) alignment between \(S\) and \(T[i..j]\). We assume the usual alignment scoring system of \(\sigma(x, y)\) for matches and mismatches and \(-g\) for gaps.

(a) Prove the following property of \(C\): For any \(i, j > 0\),

\[
C_{i-1,j-1} + C_{i,j} - C_{i,j-1} - C_{i-1,j} \geq 0.
\]

(Hint: draw a picture of a DP matrix for aligning \(S\) and \(T\) and sketch in the four alignment paths corresponding to these four terms.)

(b) Let the density matrix \(D\) be defined by

\[
D_{i,j} = C_{i-1,j-1} + C_{i,j} - C_{i,j-1} - C_{i-1,j}.
\]

Show that given only \(D\), \(C_{0,*}\) and \(C_{*,0}\), we can reconstruct all of \(C\) in time \(O(m^2)\).

(c) Now suppose that, instead of general all-substrings alignment, we are interested only in the longest common subsequence of \(S\) with each substring of \(T\). That is, \(\sigma(x, x) = 1\), \(\sigma(x, y) = 0\) for \(x \neq y\), and \(g = 0\).

Show that for this restricted problem, the density \(D\) for the all-substrings score matrix \(C\) has at most \(m\) non-zero entries, all of which are 1. (Hint: how much can the score \(C_{i,j}\) change, and in what direction, if we make \(i\) or \(j\) one character larger?)

Fun fact: Because the density matrix in all-substrings LCS is so sparse, it is possible to compute its nonzero entries in time \(O(nm)\). By the result of part (b), we can then obtain \(C\) in time \(O(nm + m^2)\), which is much better than the \(O(nm^2)\) time you might expect for naive all-substrings alignment. This result extends to more general scoring functions, leading to efficient “core-sensitive” algorithms for all-substrings alignment.