Instructions

• There are five problems in total.

• Submit a single PDF document (final.pdf) containing all of your solutions.

• You must type up your solutions. You may include scans of hand-drawn figures and illustrations.

• To hand in, log into a department linux machine. Put the file that you want the TAs to see into a directory. Then navigate to that directory and type cs1810_handin final.

• This exam is strictly non-collaborative. You may not work with anyone else and you are not allowed to consult any sources other than the textbook, the lecture notes, your own notes, and past assignments from this semester.

• The course staff will only answer questions related to clarifications, explanations, or typos. Hints will not be given. If you ask a question on Piazza, please post privately.

• Do as much as you can to convey your solution clearly. Clean solutions will receive more credit than solutions that are difficult to understand.
1 Pattern Matching (15 points)

(a) The Beatles have been learning about the Burrows-Wheeler Transform and think that they can write an interesting song based on this beautiful technique for data compression. Thus, they decide to write a song that describes the process of doing a BWT of the word YESTERDAY. However, they don’t have time to waste on writing this song as they are busy musicians, so they ask you for help! Can you write this song so that the Beatles can sing it?

Translation: Compute the Burrows-Wheeler Transform of the string YESTERDAY and show your work.

(b) Consider using the Knuth-Morris-Pratt algorithm to find a match of the pattern ATCATG in the text ATGATCATATCATGAT. Assume that the algorithm continues to the end of the text even after a match is found. How many character comparisons are made? How many times is the pattern shifted (we are asking for the number of shifts, not their sizes)? Show your work.

2 Suffix Trees (20 points)

Consider the following problem.

<table>
<thead>
<tr>
<th>LONGEST SHARED SUBSTRING PROBLEM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Strings $u$ and $v$.</td>
</tr>
<tr>
<td><strong>Output:</strong> The longest substring that occurs in both $u$ and $v$.</td>
</tr>
</tbody>
</table>

In class, it was briefly mentioned that we could solve this problem by concatenating the two strings and constructing a joint suffix tree for the resulting string. In this problem, we expand upon this idea.

We begin by appending two different end-of-string characters to $u$ and $v$, giving us $u\#$ and $v\$. We then build the suffix tree for the concatenation of these two strings, $u\#v\$. We shall color the leaves of this tree according to the following rule: if a leaf is labeled by the starting position of a suffix starting in $u$, color it blue; if a leaf is labeled by the starting position of a suffix starting in $v$, color it red. Note that all the leaves in the tree will be colored either red or blue according to this procedure.

Next we color the internal nodes of the tree according to the following rules:

- A node is colored blue (resp. red) if all the leaves in the subtree rooted at that node are blue (resp. red).
- A node is colored purple if the subtree rooted at that node contains both blue and red leaves.

At this point, you are strongly encouraged to run through this procedure on a pair of simple example strings, so that you can better understand the logic and purpose of this coloring scheme. You will not be
graded on whether or not you create an example, so feel free to share your example with your classmates and TAs on Piazza.

(a) Task: Prove that every path ending in a purple node in the suffix tree of $u \# v$ spells out a substring shared by $u$ and $v$.

Given these two facts, it should be clear how we find the longest shared substring of $u$ and $v$: We need only examine the strings spelled by paths that lead to purple nodes; the longest such string is precisely the solution to the Longest Shared Substring Problem.

The above approach makes use of a single suffix tree to solve the Longest Shared Substring Problem. It is also possible to solve this problem using two suffix trees, one for each of $u$ and $v$.

(b) Task: Describe an algorithm that solves the Longest Shared Substring Problem using the two suffix trees $T_u$ and $T_v$.

(c) Task: Compare (in as much detail as possible) the worst case space- and time-efficiency of the algorithm that uses a single suffix tree for $u \# v$ with the worst case space- and time-efficiency of the algorithm you designed in (c) above.

3 Sequence Logos (25 points)

In class we presented the concept of phylogenetic footprinting, which allows us to identify transcription factor binding sites (non-coding regions of DNA that have varying levels of conservation between related species). An alternative way to represent consensus sequences makes use of sequence logos. Just like phylogenetic footprinting, sequence logos are derived using information theoretic concepts and ultimately compute the information content of each position along a finite conserved region. Sequence logos have the added benefit of lending themselves well to neat, colorful graphical representations. The most commonly used method of sequence logo construction was proposed by Schneider et al. in 1990. The method we will focus on is an earlier construction, also by by Schneider et al. (1986). This is what a sequence logo looks like:

![Sequence Logo Diagram]

The height of each letter is related (but not proportional) to its frequency at the indicated position.

Although we urge you peruse the papers above to take in the sublime beauty of sequence logo construction in its entirety, the following definitions are all that are required to complete this problem.
Suppose we have a multiple alignment of \( n \) sequences each of length \( m \). The uncertainty about which base pair occurs at position \( i \in \{1, \ldots, m\} \) can be measured by the entropy \( H_i \), defined by

\[
H_i = - \sum_{a \in \mathcal{X}} f_{a,i} \log_2 f_{a,i},
\]

where \( \mathcal{X} \) is the set of possible characters (for DNA, \( \mathcal{X} = \{A, C, G, T\} \)) and \( f_{a,i} \in [0, 1] \) is the frequency of character \( a \) at position \( i \). Note that for all \( i \),

\[
\sum_{a \in \mathcal{X}} f_{a,i} = 1.
\]

The information content (which is plotted on the vertical axis of the sequence logo above) of a position \( i \) is given by

\[
R_i = \log_2 |\mathcal{X}| - (H_i + \epsilon_n),
\]

where \( \epsilon_n \) is a small-sample correction factor defined by

\[
\epsilon_n = \frac{1}{\ln 2} \cdot |\mathcal{X}| - 1 - \frac{1}{2n}.
\]

To construct the sequence logo, we set the height of character \( a \) at position \( i \) to be \( f_{a,i}R_i \).

Consider the following multiple alignment of 6 DNA sequences, each of length 7:

- AACGGTC
- CGCTACG
- GCCAAGT
- TGCTAAA
- GTATATC
- TGCTAGA

(a) Use the formulas above to compute the information content of each position. In other words, for \( i = 1, 2, \ldots, 7 \), compute \( R_i \). Additionally, for \( i = 3 \) and \( i = 6 \), compute the height of each of the four nucleotides \( A, T, C, \) and \( G \) at position \( i \).

(b) Compare and comment on the relative values of \( R_3 \) and \( R_6 \).

(c) Describe in words what the height of each letter in the graphic representation means. In particular, think about how a character can have two different heights at positions \( i \) and \( j \) in even though its relative frequency at \( i \) is the same as its relative frequency at \( j \). For example, the height of \( A \) at position \( i = 2 \) is different from its height at position \( i = 3 \), even though \( f_{A,2} = f_{A,3} = 1/6 \). Why is this seemingly unnatural property actually desirable in sequence logos?

4 A Hidden Markov Model (25 points)

Chromatin is a complex of a DNA and protein molecules. It is responsible for the structure of chromosomes as seen under the microscope. Chromatin comes in two forms: euchromatin and heterochromatin.
In this problem we will restrict ourselves to euchromatin. In euchromatin, certain segments of the DNA are wrapped around proteins called histones. The resulting structure resembles a series of beads (DNA-bound histones) on a string (intervening linker DNA).

Histone proteins play a role in many DNA-dependent processes, such as transcription, replication, repair, and recombination, so it is important to study them and attempt to infer where they might be located. We can consider the DNA in euchromatin to have two states: one in which it is bound to histones and one in which it is not.

One experimental technique that can be used to locate where proteins bind to DNA is called Chromatin Immunoprecipitation and Sequencing (ChIP-seq). A ChIP-seq experiment has the following steps: (1) proteins are chemically linked to the DNA to which they are bound; (2) the DNA is fragmented into small pieces; (3) fragments containing bound protein are selectively extracted; (4) the extracted fragments are sequenced. Note that the extraction process is imperfect and thus DNA sequence reads are produced from both bound and unbound DNA sequences, albeit with an enrichment for bound sequences.

After aligning these reads to the source genome of length $n$, the output of a ChIP-seq experiment can be represented as a vector of read counts $r = (r_1, \ldots, r_n)$, where $r_i$ is the number of reads whose alignment to the genome contains position $i$. For example, if $r_{22390} = 5$, then this means that 5 reads cover location 22390 in the genome. Note that $0 \leq r_i < \infty$ for all $i \in \{1, \ldots, n\}$. The results of a ChIP-seq experiment can be used to locate histones, since we expect higher observed read counts to be correlated with the presence of a histone at a given position in the genome.

In this problem your task is to come up with a hidden Markov model for determining which genomic regions are bound by histones. Present your model by answering the following questions. The notation below is consistent with the notation in the lecture notes.

(a) What is the sequence of observation symbols $\sigma = \sigma_1\sigma_2\ldots\sigma_T$? What is the meaning of each $\sigma_t$? What is the value of $T$?

(b) How many hidden states should the model contain (what is the value of $N$)? What does each hidden state $s_i$ represent? Please be as specific as possible.

(c) Using your answers and notation from parts (a) and (b), define the probabilities and/or probability distributions needed to completely specify the HMM.

(d) Discuss what you expect the (relative) magnitudes of these probabilities to be.
(e) Give an equation for the most likely euchromatin structure for a DNA sequence, given the data $\sigma$. What algorithm would you use to compute the structure itself (i.e. not the probability)?

5 Assembly (15 points)

The $k$-mer decomposition of a string $s$ is defined to be the collection of all length $k$ substrings of $s$ (including repeats). Symbolically, the $k$-mer decomposition of $s$ is the collection

$$(s_1s_2\ldots s_k, s_2s_3\ldots s_{k+1}, \ldots, s_{n-k+1}s_{n-k+2}\ldots s_n),$$

where $s_i$ denotes the $i$th character in $s$.

(a) What is the 3-mer decomposition of CTTATTGTTC?

(b) What is the 4-mer decomposition of CTTATTGTTC?

(c) Recall that in genome assembly, the only information we have is the collection of short reads. Imagine trying to reconstruct the string CTTATTGTTC from its 3- and 4-mer decompositions. In the context of this particular string, what information does the 4-mer decomposition contatin that the 3-mer decomposition does not?