Write your answers on these pages and show your work. You may use the back sides of pages as necessary. Make sure your exam has every page (numbered 1 through 7).

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
<th>Max Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>50</td>
</tr>
</tbody>
</table>
1. (10 pts) For each of the following graphs, determine if the graph is a valid sequencing-by-hybridization (SBH) graph \((k = 3)\) for some string. If a graph is valid, give one string that corresponds to it. If a graph is invalid, explain why it is not valid.

(a) (2.5 pts)

Valid

Six possible strings:
- ATCGTCAT
- TCGTCATC
- CGTCATCG
- GTCATCGT
- TCATCGTC
- CATCGTCA

(b) (2.5 pts)

Not valid

A SBH graph must be connected. This graph has two connected components.
An edge \((u,v)\) is allowed only if the last character of \(u\) matches the first character of \(v\).
2. (10 pts) Professor X really enjoys dynamic programming and decides to apply this algorithmic technique to the shortest superstring problem. Professor X’s key idea is that every superstring, \( s \), corresponds to a sequence of indices \( i_1, i_2, \ldots, i_m \), which indicates a sequence of strings, \( s_{i_1}, s_{i_2}, \ldots, s_{i_m} \) that are merged (in order) to produce \( s \). For example, if \( s_1 = \text{TACT} \), \( s_2 = \text{AGTA} \), and \( s_3 = \text{GTAC} \), then the superstring \( s = \text{AGTACT} \) corresponds to the sequence 2, 3, 1.

(a) (5 pts) Professor X comes up with a subproblem definition for a dynamic programming algorithm. Let \( M_{j,k} \) be the length of the shortest string that results from merging a sequence of \( j \) strings, such that the last string in the sequence is \( s_k \). Clarifying note: a sequence of strings may contain the same string more than once. Complete the equation for the recurrence of \( M_{j,k} \):

\[
M_{j,k} = \begin{cases} 
\text{length}(k), & \text{if } j = 1 \\
\min_i \left\{ M_{j-1,i} + \frac{\text{length}(k) - \text{overlap}(i,k)}{\text{length}(k)} \right\}, & \text{if } j > 1
\end{cases}
\]

\( \text{length}(k) \) = length of \( s_k \)

\( \text{overlap}(i,k) \) = maximum length of overlap between a suffix of \( s_i \) and a prefix of \( s_k \)

(b) (5 pts) Professor X claims that, given \( m \) strings (none of which are substrings of each other), the length of the shortest superstring containing all of the strings can be found by computing:

\[
\min_i M_{m,i}
\]

Explain why Professor X’s claim is incorrect (if this were correct, then a NP-hard problem would have a polynomial-time algorithm!).

While Professor X’s dynamic programming algorithm is correct as defined, it does not solve the shortest substring problem. Although a valid superstring can be represented by a length \( m \) sequence of input string indices, it is not the case that any length \( m \) sequence of input string indices corresponds to a valid superstring. In particular, because a sequence of string indices may contain the same index multiple times, there is no guarantee that all \( m \) input strings are included, which is a requirement of the shortest superstring problem. In fact, Professor X’s algorithm will only find a sequence of indices that consists of the index of the shortest input string repeated \( m \) times (merging such a sequence of strings results in just the shortest input string itself!). Only sequences of indices that are permutations of \( \{1, 2, \ldots, m\} \) represent valid superstrings.
3. (10 pts) In this problem, you will find optimal global alignments of the sequences CATG and CTGCAT. We will use a linear gap penalty with parameters \( \text{match} = 2 \), \( \text{mismatch} = -1 \), and \( \text{space} = -1 \).

(a) (5 pts) Fill in the values and traceback pointers for the empty cells in the dynamic programming matrix below.

(b) (5 pts) Give both the highroad and lowroad optimal global alignments for these sequences.

Highroad alignment:
---CATG
CTGCAT--

Lowroad alignment:
CATG---
C-TGCAT
4. (10 pts) We have learned about 1st-order Markov chains, $k$th-order Markov chains, inhomogeneous Markov chains, and Hidden Markov models. For each of the following types of sequences, specify the model type that is most appropriate and very briefly explain your reasoning.

(a) (2 pts) DNA sequences where the probability of observing a $C$ at position $i$ depends on the characters at positions $i - 1$ and $i - 3$.

3rd order Markov chain. A 3rd order Markov chain allows us to model dependencies between a character and the three characters preceding it.

(b) (2 pts) DNA sequences where the probability of observing a $C$ at position $i$ depends on the character at position $i - 1$ and whether $i$ is odd or even.

1st order inhomogeneous Markov chain. An inhomogeneous Markov allows us to model dependencies between a character and its position (in this case, whether the position is odd or even).

(c) (2 pts) DNA sequences made up of two sequence classes, $X$, and $Y$, and where the probability of observing a $C$ at position $i$ depends on the sequence class of position $i$.

HMM. An HMM allows us to model the dependencies of a character on the “hidden” class of the sequence at its position.

(d) (2 pts) DNA sequences where the probability of observing a $C$ at position $i$ depends on the character at position $i - 1$.

1st order Markov chain. A 1st order Markov chain allows us to model dependencies between a character and the character preceding it.

(e) (2 pts) DNA sequences that code for protein sequences, and for which we do not know the reading frame.

HMM. For coding DNA, we would like to model the different frequencies of characters at the three codon positions. In this case, because we do not know the reading frame, the codon position (1, 2, or 3) of each of the characters is “hidden”. Thus, we must use an HMM with states representing the different codon positions.
5. (10 pts) Provide short answers to the following questions:

(a) (2 pts) For which alignment problem is BLAST a heuristic solution?

Local alignment

(b) (2 pts) For what biological reason might we prefer performing alignment with an affine gap penalty instead of a linear gap penalty?

Insertion and deletion events in biological sequences often involve more than one character at a time. Thus, we might like the penalty of an insertion of length \( L \) to be less than the sum of the penalties of \( L \) insertions of length one.

(c) (2 pts) Suppose the entry for amino acid pair \((X, Z)\) is negative for some substitution matrix, \( S \). What does this tell you about the relation between \( p_{XZ} \) (the frequency of observing \( X \) and \( Z \) as a homologous pair) and \( q_X \) and \( q_Z \) (the frequencies of \( X \) and \( Z \) in known protein sequences).

\[
S(X, Z) = \log \frac{p_{XZ}}{q_X q_Z}.
\]
Therefore, if \( S(X, Z) < 0 \), then \( p_{XZ} < q_X q_Z \).

(d) (2 pts) Suppose we have Markov chain models for two classes of sequences, \( A \) and \( B \), and for some sequence \( x \), we have \( P(x|A) > P(x|B) \). For what probabilistic reason might we still classify sequence \( x \) as belonging to class \( B \)?

If we wish to classify according to \( P(B|x) \), we need to take into account the prior probabilities of the two classes: \( P(A) \) and \( P(B) \). If \( P(B) > P(A) \), then it is possible that \( P(B|x) = \frac{P(x|B)P(B)}{P(x)} > 0.5 \) even if \( P(x|A) > P(x|B) \).

(e) (2 pts) What is the purpose if using pseudocounts (e.g., for Laplace estimates or \( m \)-estimates) while estimating the parameters of a Markov chain model or HMM?

If we don’t have a lot of training data, the use of pseudocounts will help prevent overfitting to our small training set and allow the model to better generalize to data we see in the future.