Exercise 8 (Fri 2.3, 10-12, CK108, Veli Mäkinen)

Solving any 5 from below gives the maximum mark. In this last exercise, you can also return max 5 extra solutions, including any assignments during the course you have not yet marked.

1. Database search I.

BLAST is a heuristic aligner that makes local alignment feasible in large sequence databases. The following describes the main principles of BLAST with some simplifications. The database is indexed using a \( k \)-mer index, where each substring \( W \) of length \( k \) is associated with a list of pointers to the occurrence of \( W \) in the database. The lists of pointers of \( k \)-mers that are within Hamming distance 1 from some substring of a query sequence give the candidate occurrences. Dynamic programming is applied to extend candidate occurrences and to join nearby candidates, to form the final alignment results.

Show that BLAST is a lossy filter, meaning that it might miss some optimal local alignments.

2. Database search II.

Implement the \( k \)-mer index, e.g. by modifying first week’s code for \( k \)-th order Markov chain. Basic version is enough without considering how to compress the lists of occurrences.

3. Database search III.

Implement BLAST-like search on top of the \( k \)-mer index of the previous assignment to report candidate occurrences. You can ignore the dynamic programming part.

4. Insight on alignments.

Let \( A = a^{n/2}b^{n/2} \) and \( B = b^{n/2}a^{n/2} \). Show that there are at least \( 2^{n/2} \) optimal alignments for \( A \) and \( B \) with unit cost edit distance \( n \).

5. Eukaryote gene prediction by HMMs.

The flexibility of choosing the states, transitions, emissions, and their probabilities makes HMMs a powerful modeling device. So far we have used a zeroth-order Markov model for emission probabilities (probabilities only depended on the state, not on the sequence context). We could just as well use first-order Markov chains or, more generally, \( k \)-th order Markov chains, in which the probability depends on the state and on the last \( k \) symbols preceding the current one:

\[
P(s_i | s_{i-k} \ldots s_{i-1}) = P(s_i | s_1 \ldots s_{i-1}).
\]

Notice that the states of the HMM are independent, in the sense that each state can choose a Markov chain of a different order from that of the Markov chain for its emission probabilities. In addition to the use of different order Markov chains, we could adjust how many symbols are emitted in each state. Use these considerations to design a realistic HMM for eukaryote gene prediction. Try to
take into account intron/exon boundary di-nucleotides, codon adaptation, and other known features of eukaryote genes. Consider also how you can train the HMM.

6. **Profile HMMs.**

*Profile HMMs* are an extension of HMMs to the problem of aligning a sequence with an existing multiple alignment (profile). Consider for example a multiple alignment of a protein family:

```
AVLSLSKTTTNNVSPA
AV-SLSK-TANVSPA
A-LSLSK-TANV-PA
A-LSLSK-TNNV-PA
AS-SSSK-TNNV-PA
AVLSLSKTTANV-PA
```

We considered the problem of aligning a sequence \( A \) against a profile in the context of progressive multiple alignment in Section 6.6.4, and the idea was to consider the multiple alignment as a sequence of columns and apply normal pairwise alignment with proper extensions of substitution and indel scores. Consider \( A = AVTLSLSTAANVSPA \) aligned to our example profile above, for example, as follows:

```
AVTLSLS--TAANVSPA
AV-LSLSKTTN-NVSPA
AV--SLSK-TA-NVSPA
A--LSLSK-TA-NV-PA
A--LSLSK-TN-NV-PA
AS--SSSK-TN-NV-PA
AV-LSLSKTTA-NV-PA
```

Here we have added two gaps to the sequence and two gap columns to the profile following the “once a gap, always a gap” principle.

Profile HMMs are created using *inhomogeneous* Markov chains, such that each of the columns will form separate match, insertion, and deletion states, and transitions go from left to right, as illustrated in Figure 1. Match and deletion states emit the columns of the profile, so they do not contain self-loops. Insertion states emit symbols from the input sequence, so they contain self-loops to allow any number of symbols emitted between states that emit also columns of the profile.

Since the resulting HMM is reading only one sequence, the Viterbi, forward, and backward algorithms are almost identical to the ones we have studied so far. The only difference is that deletion states are *silent* with respect to the input string, since they do not emit any symbol.

a) Modify the Viterbi recurrences to handle both emitting and silent states.
Figure 1: Profile HMM illustration without showing the transition and emission probabilities.

b) Derive the Viterbi recurrences specific to profile HMMs.

7. Local alignment version of profile HMMs.

Derive a local alignment version of profile HMMs: Add states and transitions that emit any prefix and suffix of the input sequence with probability 1, and transitions inside the profile HMM and from the profile HMM that enable skipping any number of first columns and any number of last columns.

8. Pair HMMs.

Pair HMMs are a variant of HMMs emitting two sequences, such that a path through the HMM can be interpreted as an alignment of the input sequences. Such pair HMMs have a match state emitting a symbol from both sequences simultaneously, and symmetric insertion and deletion states to emit only from one input sequence.

a) Fix a definition for pair HMMs and derive the corresponding Viterbi, forward, and backward recurrences. Hint. The result should look very similar to Gotoh’s algorithm for global alignment with affine gap costs from Section 6.4.4.

b) Apply a derivation for pair HMMs similar to the ones we used in obtaining relation (7.13), in order to define the probability of $a_i$ aligning to $b_j$ over all alignments of $A = a_1 \cdots a_m$ and $B = b_1 \cdots b_n$.

c) Let $p_{ij}$ denote the probability derived above to align $a_i$ to $b_j$. We say that the most robust alignment of $A$ and $B$ is the alignment maximizing the sum of values $p_{ij}$ over $i,j$ such that the $a_i \to b_j$ substitution is part of the alignment. Derive a dynamic programming algorithm to compute this most robust alignment.

9. Bidirectional BWT index applications I.

Give the pseudocode for the $k$-mismatches search using case analysis pruning on the bidirectional BWT (see last paragraph of Sect. 10.3.2 on page 209). You may
assume that a partitioning of the pattern is given together with the number of errors allowed in each piece. Start the search from a piece allowed to contain the fewest errors.

10. **Bidirectional BWT index applications II.**

   Give an algorithm to construct the indicator bitvector $I$ of Algorithm 11.2 in $O((m + n) \log \sigma)$ time and $O((m + n) \log \sigma)$ bits of space.

11. **Bidirectional BWT index applications III.**

   Modify Algorithm 11.2 for maximal unique matches on two strings to use two bidirectional indexes instead of the indicator bitvector; see the solution for maximal exact matches for an example.

12. **Bidirectional BWT index applications IV.** Recall the algorithm for computing maximal overlaps in Section 13.2.3. Fill in the details for constructing in linear time the tree of nested intervals from the suffix tree associated with intervals. Hint. First prove that the locus of $R^i[k..|R^i|]$ for any $k$ is either a leaf or an internal node of depth $|R^i| - k + 1$.

13. **Haplotype analysis and positional BWT.**

   Read the paper


   What is the difference between PBWT and normal BWT? How is the input to the algorithm produced?