Exam:	**1MB304: Discrete structures for bioinformatics II**
Date:	**2007.10.19**
Time:	**9-14**
Place:	**B10:1, BMC**
Contact:	**Torgeir R. Hvidsten (tel. 6687)**

Pocket calculator and dictionary allowed. No other aids (books, notes, etc.). Answers can be given in English or Swedish.

Task 1 (20%)

a) (10%) Name the most common algorithm design techniques. For each technique, briefly describe its characteristics (in one or two sentences) and mention at least one typical application in bioinformatics.

b) (10%) The following motif finding algorithm takes as input a $t \times n$ table ***DNA*** containing t sequences of length n, and outputs a set of start positions (one for each sequence) that defines a set of motifs of length l. To evaluate the set of motifs, the algorithm uses Score(***s***, ***DNA***) that returns the number of matches between the motifs given by the start positions in ***s*** and the corresponding consensus motif. Score(***s***, i, ***DNA***) returns the partial score for the i first sequences.

```
YetAnotherMotifSearch(DNA, t, n, l)
bestMotif ← (1,1, …, 1)
bestScore ← 0
for start ← 1 to n – l + 1
    s ← (start,1, …, 1)
    for i ← 2 to t
        bestScoreStart ← 0
        for s_i ← 1 to n – l + 1
            if Score(s, i, DNA) > bestScoreStart
                bestScoreStart ← Score(s, i, DNA)
                bestPosition ← s_i
        s_i ← bestPosition
    if bestScoreStart > bestScore
        bestScore ← bestScoreStart
        bestMotif ← s
return bestMotif
```

What is the general idea behind this algorithm (explain shortly what the algorithm does)?

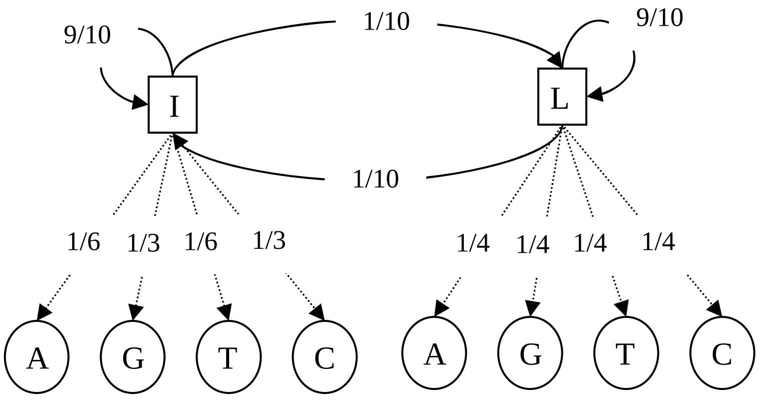
To which of the algorithm design techniques in **a)** would you classify YetAnotherMotifSearch? Why?

What can you say about the running time of YetAnotherMotifSearch? Explain.

Task 2 (30%)

a) (10%) The goal of secondary structure prediction is to assign a label H(elix), S(trand) or C(oil) to each amino acid in a protein sequence. The protein sequence is given, while the secondary structure is unknown. Sketch the architecture of a hidden Markov model that can be used for secondary structure prediction. Identify the hidden states, the emission symbols and the parameters. Note: You do not have to assign values to the parameters.

b) (10%) A hidden Markov model for detecting CG-islands in a genomic sequence could have two states, CG-island (I) and not CG-island (L), where state I is more likely to emit nucleotides C/G than A/T and state L has an equal probability of emitting each nucleotide.



What is the probability of observing the sequence CGCG given this HMM? Assume that the there is an equal probability for starting in states I and L. Show the computations.

HINT: The probability $f_{k,i}$ of emitting the sequence $x_1 \ldots x_i$ and reaching the state $\pi = k$ can recursively be computed using the recurrence $f_{k,i} = e_k(x_i) \sum_{l \in Q} f_{l,i-1} a_{lk}$, where $e_k(x_i)$ is the probability of emitting the symbol x_i in hidden state k, a_{lk} is the probability of moving from state l to state k and Q is the set of all hidden states.

What is the name of the algorithm you used? Explain in *one* sentence what this algorithm computes and how this is achieved. What type of algorithm is this (from Task1a)? What is the time complexity of this algorithm?

Is this a good HMM architecture for finding CG-islands? Explain.

c) (10%) Assume that you did not know the transition and emission probabilities in b). The Viterbi training algorithm uses a set of training sequences $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^n$ to estimate the parameters:

```
1    Guess a set of transition probabilities $a_{kl}$ (probability of moving from state $k$ to state
     $l$) and emission probabilities $e_k(b)$ (probability of emitting symbol $b$ while in state $k$)
2    while (no change in paths)
3        Compute the most probable path $\boldsymbol{\pi}^i$ for each $\boldsymbol{x}^i$ using the current parameters
4        Calculate new parameters $a_{kl}$ and $e_k(b)$ using the computed paths $\boldsymbol{\pi}^1, \boldsymbol{\pi}^2, \ldots, \boldsymbol{\pi}^n$
```

Describe the algorithm used in line 3? What probability does this algorithm compute?

Describe how you would calculate new parameters a_{kl} and $e_k(b)$ in line 4 using the paths from line 3.

Task 3 (15%)

Let $s = s_1 s_2 \ldots s_n$ denote an RNA sequence of length n. The secondary structure of the RNA can be described by its set of base pairs, $\boldsymbol{B}$. In a very simplified formulation of the RNA folding problem, one tries to find a maximum set of complementary base pairs such that

- If $(i,j) \in \boldsymbol{B}$, then $j - i > 3$.
- A nucleotide can be involved in at most *one* base pair.
- For any two base pairs $(i,j) \in \boldsymbol{B}$ and $(k,l) \in \boldsymbol{B}$, if $i < k < j$ then $i < k < l < j$ must be true (i.e. no pseudoknots).
- The complementary nucleotides are: A-U, G-U and G-C.

Given the above constraints, the following recurrence gives the maximum number of base pairs for the sequence $s_i s_{i+1} s_{i+2} \ldots s_{j-1} s_j$:

$$N_{i,j} = \max \begin{cases} c(i,j) + N_{i+1,j-1} \\ N_{i,j-1} \\ \max_{i<k<j} \{ c(k,j) + N_{i,k-1} + N_{k+1,j-1} \} \end{cases}$$

where c(i,j) = 1 if nucleotides i and j are complementary and 0 otherwise.

Given the sequence

AGUGUCGU,

calculate the maximum number of base pairs N$_{1,8}$. Show your calculations.

Draw the resulting structure(s) and clearly show how you deduced it/them from your calculations.

Task 4 (15%)

a) (5%) Name the three major approaches to computational protein structure prediction that use a database of known structures. Describe their characteristics in one or two sentences and their major application area in one sentence.

b) (10%) Protein structure prediction often relies on finding sequence similarity between the protein sequence of interest (the target) and proteins that have known structure.

Let's assume you have a family of proteins with similar structure and you want to find new members of this family in a database of protein sequences with unknown structure. You have two alternative strategies:

1. Do pair-wise sequence alignments between each protein in the family and each protein in the sequence database. Align a sequence from the sequence database to the family if a high enough score is obtained for at least one family member.
2. Build a model from the entire protein family and use that model to search for new members in the sequence database. Align all sequence with a high enough score.

Name and briefly describe two different approaches to "building a model from the entire protein family" in strategy 2.

What is the advantage of strategy 2 over strategy 1? Explain.

Task 5 (20%)

Evolution is often manifested as the divergence in gene order rather than the divergence of gene sequence. An important computational problem is to find the smallest number of reversals that transform one genome into another.

An equivalent formulation of this problem is: Given a permutation π, find a shortest series of reversals that transforms it into the identity permutation (1 2 … n)

For example, given the permutation π = 1 2 5 4 3 6 7 8, the reversal $\varrho(3,5)$ flips the fragment starting at position 3 and ending at position 5 to obtain the identity permutation: $\varrho(3,5) \cdot \pi$ = 1 2 3 4 5 6 7 8.

One approach to this problem is to identify all breakpoints in the permutation, and try to find reversals that reduce the number of breakpoints b(π). For example, the permutation π above has two breakpoints: π = **0** 1 2 | 5 4 3 | 6 7 8 **9**. Note that 0 and 9 are added to each end of the permutation. Furthermore, the fragment 0 1 2 is called an increasing strip, while 5 4 3 is an example of a decreasing strip. Single-element strips are defined to be decreasing, with the exception of the strips including 0 and 9.

The following algorithm is a greedy approach to the reversal problem:

```
GreedyBreakpointReversalSort($\pi$)
1   while b($\pi$) > 0
2       if $\pi$ has a decreasing strip
3           Among all possible reversals, choose reversal $\varrho$
            that minimizes b($\pi \cdot \varrho$)
4       else
5           Choose a reversal $\varrho$ that flips an increasing strip in $\pi$
6       $\pi \leftarrow \pi \cdot \varrho$
7       output $\pi$
8   return
```

a) (15%) GreedyBreakpointReversalSort is an approximation algorithm. Explain what that means.

Write an exhaustive search algorithm ExhaustiveReversalSort that solves the reversal problem. For simplicity, it is enough that your pseudo-code performs the search. It does not have to return or print out the reversals in the solution. Your pseudo-code should not be more than 10 lines long! What is the time complexity of this algorithm?

Let's say you run the two algorithms on four different permutations and get solutions with the following number of reversals:

	π_1	π_2	π_3	π_4
GreedyBreakpointReversalSort	5	8	3	5
ExhaustiveReversalSort	2	2	3	4

From this data, what can you say about the performance guarantee of GreedyBreakpointReversalSort? Explain.

b) (5%)

Rewrite line 3 in GreedyBreakpointReversalSort so that the algorithm becomes a randomized algorithm.

Could you see any potentially advantage of the randomized algorithm over the greedy version? Explain.

Is your algorithm a Monte Carlo algorithm or a Las Vegas algorithm? Explain.