Exam: **1MB304: Discrete structures for bioinformatics II**
Date: **2006.10.18**
Time: **9-14**
Place: **Polacksbacken, Skrivsal**
Contact: **Torgeir R. Hvidsten (tel. 6687)**

No aids (books, notes, etc.) are allowed. 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 ~~mis~~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.

```
AnotherMotifSearch(DNA, t, n, l)
1       s ← (1,1,…,1)
2       bestMotif ← Find(DNA, s, 1, t, n, l)
3       return bestMotif

Find(DNA, s, currentSeq, t, n, l)
1       i ← currentSeq
2       bestScore ← 0
3       for j ← 1 to n – l + 1
4               s_i ← j
5               bestPossibleScore ← Score(s, i, DNA) + (t – i) · l
6               if bestPossibleScore > bestScore
7                       if currentSeq ≠ t
8                               s ← Find(DNA, s, currentSeq + 1, t, n, l)
9                       if Score(s, DNA) > bestScore
10                              bestScore ← Score(s, DNA)
11                              bestMotif ← s
12      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 AnotherMotifSearch? Why?

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

Task 2 (20%)

a) (5%) Scoring matrices are often used to score matches/mismatches in alignments of amino acid sequences. Describe one commonly used strategy for constructing a scoring matrix. How is this strategy changed when the matrix is used to align closely related sequences compared to when it is used to align evolutionary distant sequences?

Why do match scores (i.e. scores on the diagonal) often differ from amino acid to amino acid in scoring matrices?

Why do mismatch scores (i.e. scores outside the diagonal) often differ from amino acid pair to amino acid pair in scoring matrices? Give an example.

b (5%) Consider the following scoring matrix δ for sequences in a six letter alphabet.

	A	**B**	**M**	**O**	**S**	**T**	**-**
A	1	-1	-1	-2	-2	-3	-1
B		1	-1	-1	-2	-2	-1
M			2	-1	-1	-2	-1
O				1	-1	-1	-1
S					1	-1	-1
T						2	-1
-							-1

Also, consider the two sequences:

v **= MOAT**
w **= BOAST**

Fill out the dynamic programming table for a *local* alignment between v and w under the scoring matrix δ. Draw arrows in the cells to store the backtrack information. What is the score for the optimal alignment and what alignment(s) does this score correspond to?

c) (10%) 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, 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 $(i,j) \in B$, then $j - i > 3$.
- A nucleotide can be involved in at most *one* base pair.
- For any two base pairs $(i,j) \in B$ and $(k,l) \in B$, if $i < k < j$ then $i < k < l < j$ must be true (i.e. no pseudoknots).

Develop a dynamic programming algorithm for finding the largest number of complementary base pairs under these assumptions. It is sufficient to give the recurrence for filling in the dynamic programming table and to describe the order in which the table should be filled in.

Task 3 (40%)

a) (15%) Given a set of k amino acid sequences of lengths $\boldsymbol{n} = (n_1, n_2, \ldots, n_k)$, write pseudo code for a greedy algorithm that builds a multiple alignment of these sequences. The algorithm should use dynamic programming to iteratively align one sequence to a profile (i.e. a position specific scoring matrix (PSSM)) and then update the profile with that sequence. Note that in the first iteration the algorithm will be aligning sequences.

Make sure that the greedy strategy and the recurrence for filling in the dynamic programming tables between a sequence and a profile is specified in detail. You do *not* have to specify in detail how a new profile is constructed based on the dynamic programming table and the old profile. Furthermore, it is enough that your algorithm returns the profile and not the actual multiple alignment.

Assume that you already have a 21 × 21 similarity matrix δ for scoring amino acids (plus gaps).

Explain the various choices you are making when designing the algorithm. If you are unable to write pseudo code, you can still answer **b)** and **c)** based on the description above and your design.

b) (5%) What do we mean by a *correct algorithm*? Is the multiple alignment algorithm you designed in **a)** correct according to this definition?

If not, construct an input of three sequences for which your algorithm will not arrive at the optimal multiple alignment. For simplicity, assume that the similarity matrix δ returns +1 for matches and -1 for mismatches and gaps. Show both the optimal alignment and the alignment given by the greedy algorithm.

c) (5%) What can you say about the running time of the algorithm you designed in **a)**?

d) (10%) Given a family of biologically related protein sequences, the algorithm in **a)** can be used to build a multiple alignment, and the corresponding profile can be used to search for new family members in a sequence database. What is the advantage of this strategy over searching for new members of the family by using pairwise alignments between individual family members and sequences in the database?

An alternative strategy would be to represent the multiple alignment of sequences from the same family using a hidden Markov model (HMM). The HMM could then be used to search for new family members. What would this HMM look like (draw a graph)? Identify the parameters of the HMM, and briefly describe how they can be estimated.

e) (5%) When aligning sequences, gaps are often modeled so that the initiation of a gap is associated with a relatively high penalty (gap initiation penalty), while the consecutive gaps are associated with a lower penalty (gap extension penalty). Why is this gap penalty model considered more biologically realistic than a model that penalizes all gaps equally?

How can this gap penalty model be implemented in the HMM described in **d)**?

Task 4 (20%)

a) (10%) Given a profile $\boldsymbol{P}$ and an arbitrary l-mer $\mathbf{a} = a_1a_2\ldots a_l$, we can compute the probability that $\mathbf{a}$ was generated by $\boldsymbol{P}$. This gives the following Gibbs sampling approach to motif finding (see Task 1 for a definition of the parameters):

1. Randomly select starting positions $\boldsymbol{s} = (s_1, s_2, \ldots, s_t)$ in $\boldsymbol{DNA}$.

2. Randomly choose one sequence out of the t sequences in $\boldsymbol{DNA}$.

3. Create a profile $\boldsymbol{P}$ from the l-mers given by $\boldsymbol{s}$ in the remaining $t - 1$ sequences.

4. For each position $1 \leq i \leq n - l + 1$ in the chosen sequence, calculate the probability p_i that the l-mer starting at this position is generated by profile $\boldsymbol{P}$.

5. Choose a new starting position in the chosen sequence randomly proportional to the distribution $\boldsymbol{p} = (p_1, p_2, \ldots, p_{n-l+1})$. Update $\boldsymbol{s}$ with this new position.

6. Repeat steps 2 to 5 until convergence.

Assume that you have already implemented a function Random(x) that returns a random number between 0 and x (x being a real number). Write detailed pseudo code for a function that implements step 5.

Describe how you would determine when the algorithm has converged (i.e. step 6). *Hint:* Use the function Score($\boldsymbol{s}$, $\boldsymbol{DNA}$) from Task 1.

To which of the algorithm design techniques in Task 1 **a)** would you classify this algorithm? Why?

b) (10%) What is an approximation algorithm? How do we define the performance guarantee of an approximation algorithm? When is approximation algorithms used?

Assume that you have written a minimization algorithm A that outputs 15 for some input $\boldsymbol{\pi}_1$ (i.e. $A(\boldsymbol{\pi}_1) = 15$) and 5 for another input $\boldsymbol{\pi}_2$ (i.e. $A(\boldsymbol{\pi}_2) = 5$). Furthermore, assume that you have run an exhaustive search for these two inputs and thus know the optimal output: OPT ($\boldsymbol{\pi}_1$) = 10 and OPT ($\boldsymbol{\pi}_2$) = 2. What can you say about the performance guarantee of algorithm A? Explain.