Solutions: **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

a) (10)

1. Exhaustive algorithms (brute force): examine every possible alterative to find the solution
    a. Partial digest problem
    b. Motif finding problem
2. Branch-and-bound algorithms: omit searching through a large number of alternatives by branch-and-bound or pruning
    a. Partial digest problem
    b. Motif finding problem
3. Greedy algorithms: find the solution by always choosing the currently ”best” alternative
    a. Genome rearrangements
    b. Motif finding
    c. Approximation algorithms
4. Dynamic programming: use the solution of the subproblems of the original problem to construct the solution
    a. Sequence alignment
    b. Gene prediction: exon chaining problem
    c. Hidden Markov models
    d. RNA folding
5. Machine learning: induce models based on previous labeled observations (examples)
    a. Hidden Markov models for modeling multiple alignments
6. Randomized algorithms: finds the solution based on randomized choices
    a. Motif finding problem (Gibbs sampling)

Correction guidlines: 1.66 points for each method. 1 point for correct description, 0.66 for correct application. Round upwards to nearest half point.

b) (10)

The algorithm recursively search through (all) combinations of start positions in a depth first manner. However, when investigating sequence i it checks whether the start positions $s_1, s_2, \ldots, s_i$ can possible produce a better score than the best score so far by assuming that start positions $s_{i+1}, \ldots, s_t$ results in a perfect score. If not, further search for this prefix is not explored.
Correction guidelines: 3 points

The algorithm is a brute force algorithm with pruning, i.e. a Branch-and-bound algorithm.
Correction guidelines: 4 points

The running time is $O(n^t)$ in the worst case, even though the pruning may result in a practical speed-up.
Correction guidelines: 3 points

Task 2

a) (5)

Entries in scoring matrices reflect how often one amino acid is substituted for another amino acid in multiple alignments of related proteins. Here "related" depends on what proteins you want to align using the scoring matrix. When aligning closely related proteins, the scoring matrix is determined from closely related proteins, and visa versa.
Correction guidelines: 3 points (2+1)

Some amino acids are particularly important for protein structure/function and are therefore scored higher than others.
Correction guidelines: 1 point

Some amino acids have similar properties and may therefore substitute each other without greatly affecting structure/function. Thus a mismatch between hydrophobic amino acids is less severely penalized than a mismatch between a hydrophobic and a hydrophilic amino acid.
Correction guidelines: 1 point (0.5+0.5)

b) (5)

Local alignment:

v/w			**B**	**O**	**A**	**S**	**T**
		0	**1**	**2**	**3**	**4**	**5**
	0	0	0	0	0	0	0
M	**1**	0	0	0	0	0	0
O	**2**	0	0	1	0	0	0
A	**3**	0	0	0	2	0	0
T	**4**	0	0	0	0	1	**3**

The score for the optimal alignment is 3 (bold). This gives one alignment:

v: OA-T
w:OAST

Correction guidelines: 5 point (3 for correct matrix + 2 for correct interpretation)

c) (10)

Recurrence:

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

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

Values for shorter subsequences (i.e. sequences between i and j) are filled in first. All subsequences where $j - i < 3$ are initiated to 0.
Correction guidelines: 10 points (5+5)

Task 3

a) (15)

Let P_j be the jth position/column in the profile $\boldsymbol{P}$, e. g. P_3 = (P_3(A), P_3(I), …) = (0.03_A,0.09_I, …) meaning that 3% of the amino acids in position 3 are As, 9% are Is, etc.

Let δ' be a modified scoring matrix that allows the second argument to be the column of a profile:

$$\delta'(v_i, P_j) = \sum_{x \in \text{amino acids}} \delta(v_i, x) P_j(x)$$

The modified recurrence for the dynamic programming that aligns a sequence $\boldsymbol{v}$ to a profile $\boldsymbol{P}$ will be:

$$s_{i,j} = \max \begin{cases} s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta'(-, P_j) \\ s_{i-1,j-1} + \delta'(v_i, P_j) \end{cases}$$

Correction guidelines: 5 points

The algorithm starts by building a profile between the two most similar sequences and then iteratively add the most similar sequence to the profile. Input is a set of t sequences $\boldsymbol{S}$ of lengths $\boldsymbol{n}$. The algorithm hides details on how a profile is built given an alignment. It is assumed that the function GlobalAlign implements the recurrence given above.
Correction guidelines: 3 points

```
GreedyAlign(S, t, n)
1       bestScore ← -∞
2       for i ← 1 to t
3               for j ← i + 1 to t
4                       if GlobalAlign(S_i,S_j,n) > bestScore
5                               bestScore ← GlobalAlign(S_i,S_j, n)
6                               seq1 ← i
7                               seq2 ← j
8       P ← BuildProfile(seq1,seq2, bestScore)
9       Remove seq1 and seq2 from S
10      while (|S| > 0)
11              bestScore ← -∞
12               for i ← 1 to |S|
13                      if GlobalAlign(S_i, P, n) > bestScore
14                              bestScore ← GlobalAlign(S_i, P, n)
15                              seq ← i
16              P ← UpdateProfile(P, seq, bestScore)
17              Remove seq from S
18      return P
```

Correction guidelines: 7 points

b) (5)

A correct algorithm is an algorithm that for each input outputs the correct solution according to some predefined goal. In this case it is natural to assume that correct means an optimal multiple alignment given the scoring matrix. Thus the algorithm in **a)** is not correct, since its greedy nature is unable to correct early mistakes:

Correction guidelines: 2 points (1 + 1)

Input:

1: ACCB
2: BCCA
3: ABCBA

The greedy algorithm would first align sequence 1 and 2, and then add 3:

```
-ACCB-
-BCCA-
ABC-BA
```

However, the optimal alignment is this:

```
A-CCB-
-BCC-A
ABC-BA
```

Correction guidelines: 3 points

c) (5)

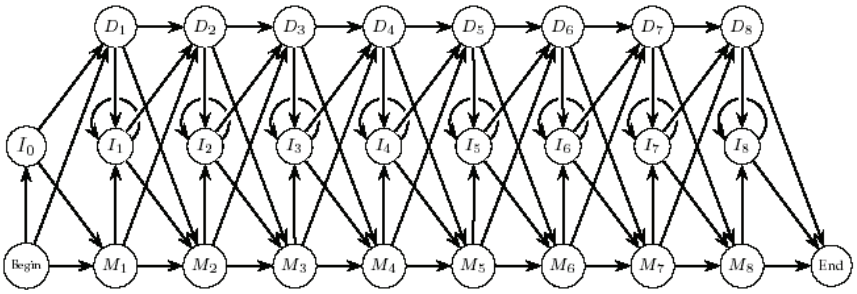
The running time of aligning two sequences or a sequence and a profile is $O(n^2)$ assuming that all sequences are of length n. Hence the time complexity of the algorithm is given by the initiation, where all sequences are align against all: $O(t^2n^2)$

Correction guidelines: 5 points

d) (10)

Distant similarities will not be significant for a pair of sequences. However, they may be significant if they are shared by the entire family. Thus aligning sequences in the database with the "whole family" (i.e. profile) may greatly increase the sensitivity.
Correction guidelines: 3 points

A profile HMM:



Assign each column in the multiple alignment to a Match state. Then add Insertion and Deletion states.
Correction guidelines: 4 points

Estimate the emission probabilities of the match states according to amino acid counts in columns. Insertion states may emit amino acids according to their occurence in all the sequences. Deletion states do not emit.

Estimate the transition probabilities between Match, Deletion and Insertion states using the known paths of the sequences in the multiple alignment.
Correction guidelines: 3 points

e) (5)

Consecutive deletions or insertions are most likely due to the same single evolutionary event. Hence gap initiation should be punished harder than gap extension.
Correction guidelines: 2.5 points

The gap penalty model can be implemented in the HMM in terms of the transition probabilities: $t_{MI} + t_{IM}$ = gap initiation, t_{II} = gap extension.
Correction guidelines: 2.5 points

Task 4

a) (10)

```
NewStartPosition(p, n)
1       sum ← 0
2       for i ← 1 to n
3               sum ← sum + p_i
4       r ← Random(sum)
5       sum ← 0
6       for i ← 1 to n
7               sum ← sum + p_i
8               if r < sum
9                       return i
```

Correction guidelines: 5 points

Save Score(***s***, ***DNA***) after each iteration and terminate the algorithm when the improvement in score stagnate, e.g. the improvement over the last 100 iterations is less than 10%.
Correction guidelines: 3 points

This is a randomized algorithm, it makes randomized choices as a part of it search strategy.
Correction guidelines: 2 points

b) (10)

Approximation algorithms are algorithms that outputs approximate solutions rather than optimal ones. Such algorithms are often associated with an approximation ratio or performance guarantee, i.e. the worst possible solution that the algorithm could output for a given input relative to the optimal solution for the same input.
Correction guidelines: 4 points (2+2)

Many problems have no optimal/correct algorithms that run in polynomial time, and researchers therefore use polynomial time approximation algorithms for these problems.
Correction guidelines: 3 points

The approximation ratio R most be larger than 5/2. The approximation ratio for a minimization algorithm is the maximum of the ratio $A(\pi)/OPT(\pi)$ over all π. 5/2 is the maximum observed value .
Correction guidelines: 3 points