Exam for *Algorithms for Genomes 2005, 13-18-05, group A.*

1. *(10 points)* What is the Central Dogma of Molecular Biology? Describe the information flow in an organism: how DNA is passed from one organism to another and how it is processed within one organism. Draw the graph, name processes and substances involved.

   - *(3 bonus points)* Name violations of the dogma.

2. *(5 points)* What is the genetic code? Name the chemicals involved, include stop signals and degenerative nature of the code.

3. *(5 points)* Define synonymous and nonsynonymous mutations. Describe the Ka/Ks ratio. What do we need to be able to calculate it? What does the value mean?

4. *(10 points)* What is log-odds ratio? How is it calculated? What does it mean when the value is less than 0? More than 0? Where is it used - give a few examples. Why are we using logarithms in the first place?

5. *(10 points)* Define the Shortest Superstring Problem. Where is it used? What is the input, output. Describe how to transform the SSP input into input for the Traveling Salesman Problem.

6. *(15 points)* Describe in detail how the ungapped BLAST algorithm works. For what kind of queries will BLAST fail to detect the orthologous sequence?

7. *(15 points)* Write the algorithm for a *sample-driven exhaustive motif search.* Describe the input, output and the time complexity of the algorithm.

8. *(15 points)* Given a signed permutation $\pi = \pi_1, \pi_2, ..., \pi_n$ (a permutation of numbers from *1* to *n* with a sign), we define an inversion $\rho(i, j)$, $1 \le i < j \le n$, as

$$\pi \cdot \rho(i, j) = \pi_1, \pi_2, ..., \pi_{i-1}, -\pi_j, -\pi_{j-1}, ..., -\pi_i, \pi_{j+1}, \pi_{j+2}, ..., \pi_n$$

   Write an algorithm which finds as small as possible number of inversions which sort a signed permutation π (*i.e.* transform it into the sequence *1, 2, ..., n*). Give the approximation ratio for your algorithm. Do not forget to give the necessary definitions and define initialization step(s).

9. *(15 points)* A new genome sequencing method has been developed at our university. VUGA (Vrije Universiteit Genome Assembler) is an error-prone process. Mistakes are plenty: improper recognition, omitting and inserting happen for approximately a few percent of nucleotides. VUGA produces pairs of reads, each ~200bp long. Even though we know that the two reads are overlapping, we don't know the order.

   For given X and Y sequences, write an algorithm *VUGA_order(*X, Y*)* which returns *true* iff it is more likely that X precedes Y (with an overlap) on the genome. Example: *VUGA_order(*TTGCACG, TGCAGTCT*)* returns *true*, because

```
TTGCACG
 TGCA-GTCT
```

   is more likely to represent the real overlap (despite the omitted/inserted nucleotide) than, for example,

```
      TTGCACG
TGCAGTCT
```

   If your algorithm needs additional parameters, leave them as undefined constants for the user to decide.

10. * *(10 bonus points)* Define a Hidden Markov Model for which there exists a hidden state q and a value i, where the most probable state at step i is q, but q is not the i-th element of the most probable sequence of hidden states. Draw the HMM, give the most probable sequence of hidden states and calculate the probabilities of each state at step i. *Hint:* emitted letters does not have to play a role.