Integrative Transcriptomics
Prof. K. Nieselt,
Institute for Bioinformatics and Medical Informatics Tübingen
Prof. S. Nahnsen,
Institute for Bioinformatics and Medical Informatics Tübingen

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Lecture: Grundlagen der Bioinformatik SoSe 2023

Assignment 1 (20 points)

Hand out: Thursday, April 27
Hand in due: Thursday, May 5, 18:00
Direct inquiries via the ILIAS forum or to your respective tutor at:
mathias-alexander.witte-paz@uni-tuebingen.de
meret.haeusler@student.uni-tuebingen.de
simon.hackl@uni-tuebingen.de
carolin.schwitalla@qbic.uni-tuebingen.de

Theoretical Assignments

1. **Scoring matrices** (6P)

In the lecture (p.37) the following equation was introduced in the context of scaling likelihood-based values of scoring matrices:

$$s'(a,b) = \frac{1}{\lambda}\log_2(\frac{p_{ab}}{p_a p_b})$$

Set up an equation to solve for the unknown scaling factor λ. (Hint: by definition the sum of all $p_{ab} = 1$, since they are all probabilities). Then try to solve (numerically) for λ for the following scoring matrix on $\Sigma = \{A, G, C, T\}$:

$$\left(\begin{array}{ccccc} & A & C & G & T \\ A & 4 & -4 & -4 & -4 \\ C & -4 & 4 & -4 & -4 \\ G & -4 & -4 & 4 & -4 \\ T & -4 & -4 & -4 & 4 \end{array}\right)$$

assuming $p_a = 0.25$ for all $a \in \Sigma$.

Practical Assignments

For the practical assignments you should keep a good structure in your code, e.g. implement functions that solve the sub-tasks presented.

For this task, you **must** use the provided template. File paths **must not** be hard-coded, but it **must** be possible to provide arbitrary files as input. Use the separate classes `FastaReader.java` and `EditDistance.java` for task 2 and 3 but all functions should be called within the function `main()` for the program to run without the need of any further modification.

All code **must** be well documented. Points will be deducted for insufficient comments.

Your program must run with:

```
Main.java file.fasta
```

If we can't run your program, it will not be graded.

2. **Reading and Writing Sequences in FASTA Format** (6P)

   Implement a Java program that:

   - Reads sequences from a given FASTA file that may contain one or many sequences (use this also in No. 3),
   - Outputs the length of the read sequences to the console after reading and the number of all sequences,
   - Writes out the base frequency to the terminal
   - Converts the given fasta sequence(s) to its corresponding RY-sequence(s) and writes it back to a file *ry_inputfilename.fasta*. (where R stands for the purine-based and Y for the pyrimidine-based nucleobases)

   Your program should handle the sequence data in a way that allows for subsequent processing and saves also other information, e.g. the header of the sequence. For this task, you are not allowed to use the FASTA-reader from any established library.

   Use the data (`single-sequence.fasta` and `multi-sequence.fasta`) and code skeleton provided in the `material-A1.zip` file.

3. **Calculation of edit distance using DP** (8P)

   Write a program that computes the edit distance between two strings using Dynamic Programming as discussed in the lecture. In more detail, your program should expect a FASTA file containing only two sequences as input and report their edit distance as output.

   Apply your program to the provided input file called `sequences-edit.fasta` from the file `material-A1.zip`. The result (edit distance) should be printed to the console.

   Note: If you did not manage to solve task No. 2 you can use the `SimpleFastaReader.jar` provided in the `material-A1.zip` file to read in sequences: `SimpleFastaReader.jar` implements the static parseFastaFile method, that expects a single File object as parameter and returns an ArrayList of Fastas (cf. the code skeleton). You can add a dependency to a JAR via *File > Project Structure > Modules > Dependenceis (Tab) > + > 1 JARs or Directories...*. Ensure that the JAR is located inside the uploaded ZIP Archive!

Please read the questions carefully. If there are any questions, you may ask them during the tutorial session or in the forum of ILIAS. You will usually get an answer in time, but late e-mails (e.g. the evening of the hand-in) might not be answered in time. Please upload all your solutions to ILIAS. Don't forget to put your names on every sheet **and** in your source code files. Please pack both your source code as well as the theoretical part into one single archive file and give it a name using this scheme: `<name1>_<name2>_<Assignment>_<#>.zip`. The program should run without any modification needed.