ITX 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 4 (20 points)

Hand out: Thursday, May 18
Hand in due: Thursday, May 25, 18:00
Direct inquiries via the ILIAS forum or to your respective tutor at:
caroline.jachmann@uni-tuebingen.de
meret.haeusler@student.uni-tuebingen.de
simon.hackl@uni-tuebingen.de
carolin.schwitalla@qbic.uni-tuebingen.de

Theoretical Assignments

1. **MSA: How small is small?** (3P)

   Astrophysicists predict that the sun will run out of hydrogen and then turn into a red giant in about 5 billion years ($5 \cdot 10^9$). Assume as of today, you start computing an MSA of r sequences, each of length $L = 500$. Assume your computer needs 10^0 seconds per pairwise alignment, and 10^6 seconds for four sequences. Using the simplified time complexity of $O(2^r L^r)$ of the dynamic programming algorithm based on the sum of pairs score, the computer computes an MSA of these sequences. Compute how many sequences r could be aligned until the sun has died? Imagine you also access to a supercomputer to compute such an MSA. How much faster does the computer have to be in order to achieve an MSA with double as many aligned sequences as when using your own computer?

2. **Unlabeled branching pattern** (1P)

   For unrooted binary trees, how many leaves do there have to be at least to obtain more than one unlabeled branching pattern? Provide sketches of the topologies to support your answer.

   An example for labeled and respective unlabeled tree for $n = 4$ sequences is shown in the following figure.

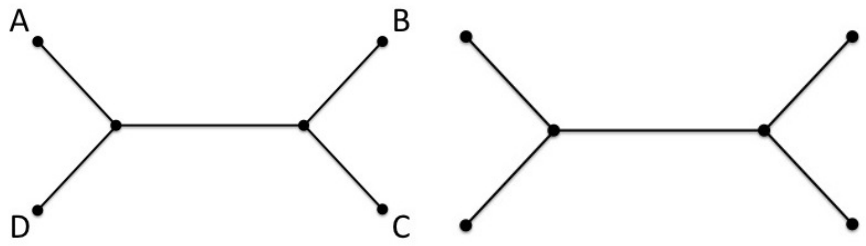

Figure 1: Example for a labeled (left) and unlabeled (right), unrooted tree on four leaves.

3. **Marker genes for the prokaryotic tree of life** (2P)

In the lecture, two universal marker genes (16S rRNA and aminoacyl-tRNA synthetase) were discussed as they are used in phylogenetic reconstruction of prokaryotes. Try to find general criteria for marker genes (in terms of phylogenetic reconstruction) in prokaryotes. Write a maximum of 100 words and provide literature references in your answer.

Practical Assignments

4. **Progressive alignment** (14P)

In this task we ask you to implement a program that can be used to compute MSAs based on the progressive alignment approach as discussed in the lecture. In particular, you are asked to implement the pseudocode of progressive alignment (see p. 68 in lecture notes). You may want to use your program from the previous assignment (A3, task No. 3) as a basis for the implementation.

(a) Implement a class that reads and stores scoring matrices from `.tsv` files. The accepted format should correspond to the formatting of the file `/resources/blosum62.tsv`. The class should also be able to return the respective score of two characters stored in a read scoring matrix. Catch cases in which the score of two characters is requested, but not stored in the scoring matrix.

(b) Adapt your implementation of the Needleman-Wunsch algorithm (from the last assignment) to allow the use of a general scoring matrix (as implemented in (a)) instead of a constant match and mismatch score.

(c) Implement the pseudocode from Chapter 5 (p.68) for progressive alignment. Use your adaption of the Needleman-Wunsch algorithm to align sequences and combine alignment profiles. For convenience, for the pair-guided step, always use the first sequence of the two respective profiles (sub-alignments) when combining profiles. Find an appropriate data structure that stores the order in which sub-alignments are chosen and supply this data structure as a parameter to your method. The final MSA should be written to the console.

(d) Finally, apply your program to the sequences provided in `resources/sequences_to_align.fasta`, the provided scoring matrix and a gap penalty of $d = 5$. The ordering for the selection of sub-alignments is shown in Figure 2. For this assignment we allow you to hard code the tree structure to use for the data structure you have implemented in (c). Your program should be executable with the following command, i.e., parse all required arguments from the command line:

```
java -jar NAME1_NAME2_Assignment_4.jar –scoring-matrix FILE.tsv –gap-penalty NUMBER –sequences FILE.fasta
```

(d′) Alternatively, feel free to implement a method that allows a general tree structure to be used as a guide tree. We leave it up to you to specify the tree data structure, but you may use the Newick format (`https://evolution.genetics.washington.edu/phylip/newicktree.html`) as a role model. You may additionally catch scenarios in which sequences are to be aligned but that are not represented in the guide tree or vice versa. Your program should be executable with the following command:

```
java -jar NAME1_NAME2_Assignment_4.jar –scoring-matrix FILE.tsv –gap-penalty NUMBER –sequences FILE.fasta
–guide-tree FILE.tree.
```

The Newick format of the depicted guide tree to use for the sequence input is stored in the file `/resources/guide_tree.tree`.

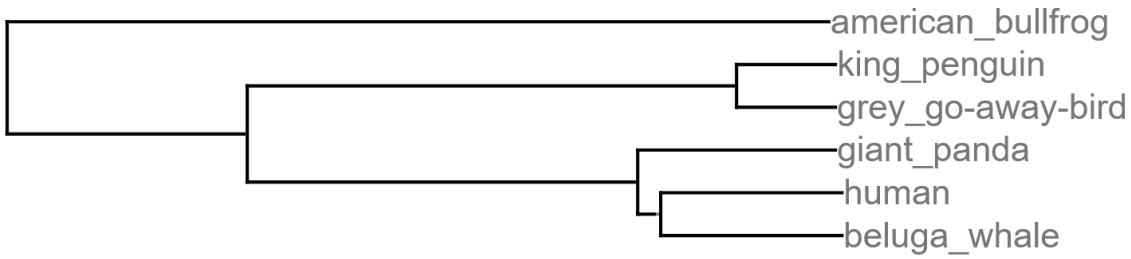

Figure 2: Guide tree to use for task four.

In case you did not manage to implement the subtasks (a) or (b) we provide you with the `ProfileAlignment.jar` file that includes the following methods:

- `ProfileAlignment(String filepath)`: Constructs a new `ProfileAlignment` object. The parameter is used to parse a `.tsv` file into a `ScoringMatrix` object, which is stored and used internally for the `alnProfiles` method.
- `alnProfiles(ArrayList<String> p1, ArrayList<String> p2)` → `ArrayList<String>` combines two alignment profiles into one by aligning the first entry of each list and adding the respective gap symbols.

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.