

Algorithms in Bioinformatics II, SS2003

Assignment sheet # 8

Daniel Huson

June 16, 2003

1 Finding MUMs (2 points)

Given two sequences $xseq$ and $yseq$. A *maximal unique match (MUM)* is a word m that occurs precisely once in both $xseq$ and in $yseq$ and is not contained in any longer word with this property. Consider the suffix tree T belonging to the concatenation $xseq\%yseq$ using a unique separator symbol $\%$.

Describe how to identify a MUM in $xseq$ and $yseq$ using T . Develop an algorithm that systematically enumerates all MUMs in $xseq$ and $yseq$ of length greater or equal to a given parameter k .

2 Fast comparison of two sequences (5 points)

The `WOTD` class can be used to produce a useful program for quickly comparing two different sequences at different levels of detail.

The program reads two sequences $xseq$ and $yseq$ as input. Additionally, a threshold parameter k is also given. Both sequences are concatenated using a unique separator symbol $\%$ and a suffix tree is built for the text $xseq\%yseq$. A recursive algorithm `findAllMUMs` is then run on the suffix tree that finds all MUMs in $xseq$ and $yseq$ that are not shorter than k .

Each detected MUM $xseq(i, i + k - 1)$ to $yseq(j, j + k - 1)$ is drawn as a line from (i, j) to $(j, j + k - 1)$ in a two-dimensional *dot-plot*.

Please implement an algorithm that systematically computes all MUMs in $xseq$ and $yseq$.

To solve the task, please add a new `findAllMUMs` method to your `WOTD` class. If possible, construct the tree lazily. Use the provided `Compare.java` program to test your algorithm on the supplied input files.

3 Application: human vs. mouse (1 point)

Consider the following three pairs of files: `human1.seq` and `mouse1.seq`, `human2.seq` and `mouse2.seq` and `human3.seq` and `mouse3.seq`. Each pair contains DNA sequences of related regions of the human and mouse genomes.

Use the program `Compare` to compare each pair of sequences by finding all MUMs of length $\geq k$. Explore the use of the parameter k . Answer the following questions: How does k influence what the dot-plot looks like? Does k influence the run-time of the program?

4 Ukkonen's algorithm (2 points)

Run Ukkonen's algorithm "by hand" on the text `yummy`, displaying the table of suffixes for each extension, indicating the range of relevant suffixes, the active suffixes $\alpha(xa)$ and $\alpha(x)a$, and showing the current implicit

suffix tree after each extension.

Please transform the implicit suffix tree into a proper suffix tree at the end.

Download

The java file `Compare.java` and all sequence files are available from:

www-ab.informatik.uni-tuebingen.de/teaching/ss03/abi2/java/assign08.zip

Assignments due: **Monday, June 23, 10am**