Advanced algorithms **Master tracks CPS2/DSC/MLDM**

Project - 2020-2021

---

1 Description

The objective of this project is to implement some algorithms for computing the *Edit distance* (ED) (an adaptation of the LCS problem) and to provide an experimental study of their running time and the quality of the solution given.

2 Work to do

You must program different algorithms for the ED problem, here a non-exhaustive list of recommandations:

- The classic algorithm based on dynamic programming. The algorithm is similar to the LCS problem, your first job will be to look for informations about this algorithm.
- The version combining dynamic programming and divide and conquer approaches allowing one to solve the problem with a linear space complexity (similar version as the one seen for LCS).
- The pure recursive version (having an exponential time complexity).
- A branch-and-bound version of the recursive approach (see lecture slides).
- An approximated version of the classical dynamic programming approach where one fills only a stripe of size k around the diagonal of the matrix. (different band values must be evaluated).
- An approximated version based on a greedy approach (to be defined).
- Other algorithms that you can propose by yourself or that you have found elsewhere during a search on the internet.

Note that all the algorithms must be able to output the distance of a solution and a possible alignment corresponding to an (approximated) ED.

A graphical user interface could be proposed to illustrate the behavior of the algorithm(s), but **this is not required** and must be done after the implementation of ED algorithms.

The running time of the algorithms and the quality of the solutions given (in terms of optimality) have to be evaluated on problems of different sizes (ie the length of the input strings). A random generator of problems could be used for example. You have to provide a full and rigorous experimental study to illustrate the different strong and weak points of each algorithm.

The algorithms must be also evaluated on a protein database available here:
`http://forge.info.univ-angers.fr/~gh/Shspdb/index.php?action=0&mode=0`
(experimental setup to be defined).

The project can be done in groups of 4 students (exceptionally 3), and each student must program at least one algorithm.

3 What to send

All the groups will receive an evaluation based on a report, an oral defense and the code provided.

First, each group must upload before **Wednesday November 4, 11:59 (noon)** a text containing the members of the group on claroline, in the `Group_ Description` resource. This text must also contain a provisional planning giving the milestones of the project with the tasks to achieve, the repartition between the group members and the time deserved to each task, if possible the group can mention the procedures used for testing and evaluating the programs. The ability to respect the schedule will not be used for evaluating the grade, but during the defense the students must present the real planning and discuss the reasons why the project has run late. The quality of the presentation (report, oral, ...) and the quality of the source code will be taken into account.

The final version of the projects must be uploaded on claroline, in the `Project_archive` resource, before **Wednesday December 9, 11:59 (noon)**, in a `.zip` or `.tar.gz` archive. The archive must be **well-organized** and must include the source code of the programs and a report on the work in pdf format. The project is done in groups of 4 students (exceptionally 3). A project defense will be scheduled on **Friday December 11** (mainly in morning and potentially early afternoon, the exact schedule will be given later), be careful to come to the defense with a working implementation.

Grade scaling:

- At least 7/20: recursive and classic dynamic programming version must work correctly, and be evaluated in a rigorous manner on artificial data, report and source code must be presented neatly.
- At least 10/20: each member of the group must program a different approach, experimental evaluations on artificial data must be rigorous, source code and report must be presented neatly and during the defense the answers to the questions must globally be correct.
- At least 12/20: in addition to the previous point, evaluate the algorithms on a part of the protein database and implement another method (the number of methods must be higher than the size of the group).
- At least 14/20: in addition to the previous point, implement an additional method, and process a significant part of the protein database, source code and report must be extremely neat and clear.
- At least 16/20: program all the algorithms, process a large part of the protein database and try to cluster some proteins in groups of similar strings, optionally: propose a graphical user interface.

Note that for a given grade, the absence of one element can be compensated by the addition of an item associated to a higher grade. For people targeting at least 10/20, a correct implementation of the dynamic programming/divide and conquer combined approach can be subject to a bonus. **Be careful, the grading scale is given for information purpose.**