Algorithms and Data Structures for Biology
23 April 2020 — Lab Session

Ugo Dal Lago Guillaume Geoffroy

This assignment will be the first one to be marked. Please carefully read the instruction below before starting to work on the assignment.

1 Instructions

This assignment must be completed individually, the deadline is April 3rd 2020 at 23.59 CET. To complete the assignment, you need to upload the following files to the course's IOL page (where a specific link will be available):

- One Python file named `FIRSTNAME_LASTNAME.py`, with the code you have developed.
- One PDF file name `FIRSTNAME_LASTNAME.pdf`, preferably produced by way of LaTeX, including a description of the algorithms you designed, the relevant parts of the `Python`code you wrote, and the experimental results.

2 The Problem

Suppose you are the employee of a pharmaceutical company, which has to buy as much of a certain type of chemical substance as possible in a short amount of time. Suppose you have access to offers from n suppliers, which for convenience we indicate with $s_1, \ldots, s_n$. The supplier s_i can offer your company an amount of the chemical substance (within the given period) equal to w_i grams. However, there is a problem: for reasons related to the nature of the products purchased and to your company's policy, the product supplied by certain suppliers is incompatible with that supplied by some of the other suppliers. In other words, the company has, for each supplier s_i, a list L_i of the other suppliers s_j with which s_i is incompatible (so L_i is of length at most $n-1$). Your task, as an employee of the company, is to determine the purchase plan, *i.e.* the suppliers to be selected, which allows the company to maximize the amount of chemical substance to be purchased.

This assignment asks you to:

- Model the problem described above as a combinatorial optimization problem, abstracting away from unessential details.
- Give an exhaustive search algorithm solving the combinatorial problem from the previous point; the key idea here consists in finding an appropriate way to define the search space, in such a way that exploring it turns out to be easy.
- Analyse the algorithm's complexity: give a relevant upper bound to the worst-case computation time, in big O notation, in function of n (the total number of suppliers).
- Implement the algorithm you designed in `Python`. Please do not use external modules.
- Design and implement a `Python` testing routine which tests your algorithm on randomly generated inputs in which

  - n is, successively, $8, 16, 24, 32, 40$,
  - the weights w_i are drawn uniformly and independently between 0 and 1.
  - for each $i \in \{1, \ldots, n\}$, the list L_i is a random (preferably but non-necessarily uniformly distributed) list of length $\frac{n}{2}$ of numbers in $\{1, \ldots, n\}$, without repetitions and not containing i.

  The testing routine should make use of the `random` module, only.

- Finally, use the modules `cProfile` or `timeit` to experimentally test the runtime of your program, and give a graphical presentation of your results. Moreover, try to find the best-fit curve matching the algorithm's complexity by way of `numpy` and `scipy`.

As previously mentioned, the results of your work should be summarized in a PDF file. The report should be self-contained, and allow the teachers to judge what you have done without delving into the details of your `Python` code or running the tests themselves.