85301 – Algorithms and Data Structures in Biology (2022/23)

Enrico Malizia and Riccardo Treglia

DISI, University of Bologna, Italy

Lab 3

This lab and the next one will be devoted to the development of a little project. A problem will be described and then you will be asked to design a solution for the problem, carry out experiments on your proposed solution, and write a report in LaTeX. If you will want, you can send to us the report that you will have produced and get some feedback. (This, is *not* compulsory, and will *not* contribute to the final grade for the exam.)

1 The Problem

Suppose you are an 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 which s_i is incompatible with (so L_i is of length at most $n-1$).

Your company asks you to determine the purchase plan, i.e., the suppliers to be selected, which allow the company to maximize the amount of chemical substance to purchase.

1.1 Your tasks

This assignment asks you to:

1. Model the problem described above as a combinatorial optimization problem, abstracting away from unessential details.
2. Provide a *simple* 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.
3. Analyse the algorithm's complexity: provide a relevant upper bound to the worst-case computation time, in big O notation, in function of n (the total number of suppliers).
4. Implement the algorithm you designed in `Python`; please do not use external modules.
5. Design and implement a `Python` testing routine which tests your algorithm on randomly generated inputs in which:
   - n is, successively, 8, 12, 16, 18, 20, 22, 24, and 26.
   - the weights w_i are floating point numbers 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 (around) $\frac{n}{2}$ of numbers in $\{1, \ldots, n\}$, without repetitions and not containing i.

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

6. Finally, use the modules `cProfile` or `Timer` 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`.

The results of your work can be summarized into a report (a PDF file) written in LaTeX. The report should be self-contained, and allow the teachers to provide feedback on what you have done without delving into the details of your `Python` code or running the tests themselves. Provide in the report extensive comments on your algorithm, explaining why and how it solves the problem. Your algorithm can be listed in your report as pseudocode, if you prefer. Do not assume that your code/algorithm will be self-explanatory; your code/aglorithm should be thoroughly commented, to explain the various steps of your code/algorithm. Explain the ideas behind the various solutions that you provide, e.g., for the solution of the problem, for the generation of the random input instances, and so on.