"Algorithmics and computer programming"

Skill 6: Choose the best data structures and algorithms to minimize the algorithmic complexity

Exercise 1 : Complexity of operations on Python lists

1.1 Download from Moodle the script called `perf_measure_example.py` and save it in a directory of your choice. Examine the source code, make sure you understand what it does and run it.

1.2 Create a new script called `complexity_lists.py` in the directory of your choice. Write there a function called `append_time` which:
- accepts as parameters a list *L* and an integer *k*,
- makes a clone of the list,
- appends *k* times the string "toto" into the clone list and monitors the time it took to perform those *k* insertions,
- returns the average duration of a call to append.

1.3 Test this function with a list of your choice. Make sure that the list is unchanged before and after the call to the function.

1.4 Add to your script a function called `inserthead_time`, which works like the former one, except that it makes the insertions at the beginning of the clone list rather than at the end. Use Python documentation to find which built-in method to use. Test your function, again making sure that the list is unchanged before and after the call.

1.5 Add to your script a function called `access_time`, which makes *k* accesses ([ ] operator) to the list, at randomly chosen positions. Test your function. **Hint**: if you import the module `random`, you can use `random.randrange(0, n)` to draw a random integer uniformly distributed between 0 and n-1 included.

1.6 Add to your script a function called `create_big_list_from_file`, which:
- accepts as parameters a filename and an encoding (typically "`utf-8`"),
- opens the specified file in reading mode with the appropriate encoding,
- reads each line of the file and adds it into a list,
- closes the file,
- and returns the list.

1.7 Download from Moodle the file called `words.txt` and test your function with it.

1.8 Write in the main program a loop which, for *n* = 1000 to 200000 by increments of 1000:
- extracts a sublist of size *n* from the big list (for example the *n* first items),
- measures the average access time, the average time of front insertion and the average time of back insertion in that sublist (using *k* = 200),
- writes *n* and those three measures in a tabulated text file (one line per *n* value).

1.9 Use R to plot the three execution times as a function of *n*.

1.10 Is the obtained graph consistent with the complexities announced on https://wiki.python.org/moin/TimeComplexity ? Explain why each operation behaves that way.

Exercise 2 : Text filtering

In this exercise, one wants to remove from a text all occurrences of certain words. We will optimize the performance of the filtering by profiling the code to detect the slowest operation(s), and by choosing carefully the appropriate container for the words to remove.

2.1 Download from Moodle the file called filter_with_cprofile.py and save it to a directory of your choice. Open this script and read it carefully. It contains a first version of the filtering program, in which the words to remove are stored in a list. Run this script and read the online documentation of the cProfile module to interpret the results. Was append the most costly operation?

2.2 For a more precise analysis (line by line), we can use the module line_profiler. To do so, install the module by typing the command `pip3 install line_profiler --user`. This will install the line_profiler module and the kernprof script in the .local subdirectory of your home directory[1]. You can then launch the code profiling with the command `~/.local/bin/kernprof -l -v filter_with_line_profiler.py`. (download `filter_with_line_profiler.py` from Moodle). Which line is the most costly?

2.3 Using the information on the page https://wiki.python.org/moin/TimeComplexity , suggest a faster version of the filtering program, using a more appropriate container.

[1] Or, if you use a Mac and installed python3 with homebrew, in the directory `/Users/yourlogin/Library/Python/3.7/bin/`