Scientific Computation
Autumn 2023
Project 1

Due: November 2nd, 1pm

In addition to this project description, there are 2 files for this assignment:

- project1.py: file which you will complete and submit on Blackboard (see below for details)
- report1.tex: a latex template file for the short report which you will submit. The discussion and figure(s) described below should be placed in this report.

Part 1

In this question, you will analyze the function *part1* provided in *project1.py* that sorts a length-N list of integers in non-decreasing order.

1. (5 points) Explain the strategy that *part1* uses, and analyze the worst-case computational cost of the function and the dependence of this cost on N and *istar*. As part of your analysis, you should examine the asymptotic time complexity and its dependence on N and *istar*. You are not expected to present a line-by-line operation count, however your discussion should provide some explanation of how you constructed your estimates of the cost. You should also discuss what your analysis tells you about how *istar* should be set. When explaining the strategy used by the function, your discussion should focus on the problem-solving approach used rather than the purpose of individual lines of code (though it is fine to refer to particularly important portions of code).

2. (5 points) Design a timing test to analyze how the wall time required by *part1* depends on N and *istar*. Your code should generate up to 4 plots that illustrate what you consider to be important trends. In your report, explain the design of your test, discuss the trend(s) shown in your plot(s), and also discuss the degree to which the trends agree with what you expect based on your analysis of the computational cost. You should include results for *istar*$= 0$, *istar*$= N - 1$, and at least one other value of *istar*. You do not need to consider values of N larger than 1000 (though you are welcome to do so), and your discussion should not be limited to trends for the largest values of N considered. Place the code used to generate the figure(s) in the function, *part1_time*. Place your discussion and plots in your report.

   Notes: You will not be assessed on how well your timing test results match your theoretical analysis, however there should be at least some correspondence. Assessment will instead focus on the reasoning used in the design of your code and in the discussion of its results. A figure with x subplots will be counted as x plots.

Part 2

You will now develop code to search for patterns in a length-n gene sequence, S. The patterns are the length-m sequences in the length-l sequence, T. S, T, and m are all provided as input to the function, *part2*. For each sequence of m consecutive 'letters' in T, you should find all locations where the sequence can be found in S. For example, if $S =$ GTCATGGCTGCATAG, $T =$ TCATG, and $m = 3$, the search should identify the locations 1, 2, 10, and 3. Specifically, the function should return a list of lists, L, where L[i] is a list containing all locations in S where the ith length-m sequence in T can be found (for the example above, L = [[1],[2,10],[3]]). If the ith sequence in T is not found anywhere in S, you should have, L[i] = [].

1. (6 pts) Complete the function *part2*, so that it efficiently constructs L as described above. You should design your code for input with $m \gg 1$, $l \gg 1$, and $n \gg m$ though it may be helpful to consider smaller problem sizes when developing and testing your code. You should also assume that $m < l < n - m$. Your algorithm and its implementation should be efficient with regards to time complexity, and for two methods with comparable time complexities, the method with lower memory usage should be selected. You should also assume that the cost of Python integer arithmetic is independent of the length of the integer. See the function documentation for further details on the function input/output.

2. (4 pts) Add a brief description of your code along with a clear and concise analysis of its time complexity to your report. Include an explanation of why your code should be considered to be 'efficient'. You do not need to run timing tests or discuss the wall time required by your code, and you are not being asked to optimize your code for wall time.

Further guidance

- You should submit both your completed python file (*project*1*.py*) and a pdf containing your discussion and figure(s). You are not required to use the provided latex template, any well-organized pdf is fine. To submit your assignment, go to the module Blackboard page and click on"Project 1". There will be an option to attach your files to your submission. (these should be named *project*1*.py* and *report*1*.pdf*). After attaching the files, submit your assignment.

- Please do not modify the input/output of the provided functions without permission. For part 1, you may import and use any modules that have been used or mentioned during the term so far. Otherwise, please do not import any modules without permission. You may create additional functions as needed, and you may use any code that I have provided during the term.

- Marking will be based on the correctness of your work and the degree to which your submission reflects a good understanding of the material covered up through the slides of lecture 5 and corresponding labs. You should aim to keep the pdf version of your report to less than 2 pages of text.

- Open-ended questions require sensible time-management on your part. Do not spend so much time on this assignment that it interferes substantially with your other modules. If you are concerned that your approach to the assignment may require an excessive amount of time, please get in touch with the instructor.

- Questions on the assignment should be asked in private settings. This can be a "private" question on Ed (which is distinct from "anonymous") or by arrangement with the instructor.
- Please regularly backup your work. For example, you could keep an updated copy of your files on OneDrive.
- In order to assign partial credit, we need to understand what your code is doing, so please add comments to the code to help us.
- You have been asked to submit code in Python functions, but it may be helpful to initially develop code outside of functions so that you can easily check the values of variables in a Python terminal.
- The weightings for assignments can be found in the lecture 1 slides.
- Your submission should be your own work, you should not collaborate with other students, and you should clearly and completely cite external sources which are used in your work (detailed citations of slides/labs from the module are not needed).