CS/SE 2XB3 — Lab 2

Due Friday, January 29th, 11:59pm

Submit the lab on Avenue. Note, Avenue must receive your lab by the due date. **Do not leave your submission to the last minute! Late submissions, even by seconds, will not be graded.** You have been warned.

This lab is worth (59/7)% of your final grade in the course. Read this document carefully and completely. **Whenever I ask you to discuss something, I implicitly mean to include that discussion in your lab report.**

Purpose

The goals of this lab are as follows:

1. Analyze timing data.
2. Identify potential pitfalls/errors which data can contain
3. Reverse engineering implementations based off timing data

Submission

You will submit your lab via Avenue. You will submit your lab as a .zip. Within your .zip there will be at least two documents:

- code.py
- report.pdf (or docx, etc.)

You may include other .py files in the .zip file if you wish. Only one member of your lab group will submit to Avenue – see the section below for more details on that. Your report .pdf will contain all information regarding your group members, i.e. name, students number, McMaster email, and enrolled lab section. This will be given on the title page of the report. For the remainder of this document, read carefully to gauge what other material is required in the report. Your report should be professional and free from egregious grammar, spelling, and formatting errors. Moreover, all graphs/figures in your report should be professional and clear. You may lose grades if this is not the case. Organize the report in a such a way to make things easy for the TA to grade. You are not doing yourself any favors by making your report difficult to mark!

Timing Data [30%]

On Avenue you'll find a corresponding file with this document named `lab2_data.xls`. There are three data sets, one for each function $f(n)$, $g(n)$, and $h(n)$. Give your best determination of how each of these functions is growing in n. In your report, include discussion and graphs for each function. You must include appropriate analysis to verify your claims.

Python Lists [70%]

How well do Python lists perform? What methods of the list would you want to perform well? How are Python lists implemented? This portion of the lab will have you analyze Python lists and answer these questions with the data to back up your claims. You will be analyzing three methods specifically:

- copy()
- append()
- lookups, i.e. `L[i]`

Copy [10%]

Using the timeit library perform experiments on copy() to determine its complexity. In your report include the following:

1. A description of how you chose to test the copy method.
2. Observations and conclusions made from your experiments.
3. Evidence to back up your observations and conclusions (you should include at least one graph here).
4. An explanation as to why copy() performs in the manner it does.

Lookups [20%]

Create a list containing 1,000,000 arbitrary values. Run a timing experiment where you measure the time to perform a single lookup on each of the one million indices. However, before you run the experiment, document in your report any predictions you have regarding the data. Does your data match your prediction? Plot the data via a scatter plot and include it in your report. There are 1 million data-points... I do not suggest copy and pasting from the console. Discuss what potential problems are present in your experiment. How would you fix these issues? Redesign and rerun your experiment. Again, include a scatter plot in your report alongside any conclusions you have regarding the runtime of lookups for Python lists.

Append [40%]

Create a timing experiment which builds a list with one million values by appending a single value to it one step at a time. Measure the time it takes to complete each of these appends. Similar to your Lookups experiment, document in your report any predictions you have regarding the data. Include the scatter plot of your experiment. Discuss the data compared to your predictions. As before with the Lookups what potential problems are there in your experiment which in turn cause *potential* anomalies in your data?

From here I want you to proceed on your own, exploring the data and performance of appends. Conduct whatever experiments you wish, but include your reasoning, predictions, graphs, etc., in your report. For any claim you make, back it up with empirical evidence (you can't just say you think things to be so).

I searched Python list method complexity, this was one of the first links which came up:

`https://blog.finxter.com/runtime-complexity-of-python-list-methods-easy-table-lookup/`

Do your experiments agree with the claims made there? Be precise!