Carleton University
Department of Systems and Computer Engineering
SYSC 2100 - Algorithms and Data Structures - Winter 2021

Assignment 3 - Characterizing the Running Times of Sorting Algorithms

Background

Big-Oh notation gives an upper bound on the growth of a function; for example, the running time of an algorithm, One definition of big-Oh is:

> Given functions $f, g : R \rightarrow R$ with g nonnegative, we say
>
> $$f = O(g)$$
>
> iff there exists a constant $c \geq 0$ and an x_o such that for all $x \geq x_o$, $|f(x)| \leq cg(x)$.
>
> This definition is complicated, but the idea is simple: $f(x) = O(g(x))$ means that $f(x)$ is less than or equal to $g(x)$, except that we're willing to ignore a constant factor, namely c, and to allow exceptions for small x, namely, for $x < x_0$.[1]

In this assignment, you'll develop some experiments to verify the Big-Og characterizations of some sorting algorithms.

Exercise 1

Download heapsort.py from the *Lab Materials* section of the main cuLearn course page. Function `heapsort` implements the heapsort algorithm that is presented in a set of lecture slides.

In heapsort.py, modify `heapsort` to keep track of the number of comparisons and swaps performed by the algorithm. The function should return a tuple: the first value will be the number of comparisons and the second value will be the number of swaps.

Exercise 2

Create a new file named Asst_3.py. In this file, write a script (program) that imports your modified bubble_sort module from Lab 12. The script will call function `bubble_sort` to sort three lists:

- A list containing n integers, [1, 2, 3, ... n], sorted in ascending order
- A list containing n integers, [n, n - 1, n - 2, ... 1], "reverse-sorted" in descending order
- A list containing n random integers

The script should perform this experiment for lists of different lengths; for example, it might call `bubble_sort` with three lists of length 10 (n = 10), three lists of length 100 (n = 100), and so

[1] Section 14.7.2 of *Mathematics for Computer Science*, rev. Wednesday, 6th June, 2018, 13:43, Eric Lehman, F Thomson Leighton, Albert R Meyer.

on.

For each list, record the number of comparisons and number of swaps performed by the bubble sort algorithm.You should end up with a table that looks like this:

n	**sorted list**		**reverse sorted list**		**random list**	
	# comparisons	**# swaps**	**# comparisons**	**# swaps**	**# comparisons**	**# swaps**
10						
100						
...						
...						

It's up to you to decide how many values of *n* you should use in order to give you enough data to characterize the running time of bubble sort.

Exercise 3

Repeat Exercise 2 for selection sort and merge sort (Lab 12) and heap sort (Exercise 1). The script in Asst_3.py should now contain the experiments for all four sorting algorithms.

Exercise 4

Plot some graphs that use the data you obtained in Exercises 3 and 4. Write a short report that includes your experimental data and graphs to verify that bubble sort and selection sort are $O(n^2)$ and merge sort and heap sort are $O(n \log n)$.

Wrap Up

Please read *Important Considerations When Submitting Files to cuLearn*, on the last page of the course outline.

The submission deadline for this lab is Wednesday, April 14, 23:55 (Ottawa time), for all lab sections.

To submit your lab work, go to the cuLearn page **for your lab section** (not the main course page). Submit your modified heap.py and Asst_3.py files, along with your report. The report must be a PDF file. Other file formats (e.g, .docx, .odt, etc.) will not be graded.

You are permitted to edit and resubmit the files as many times as you want, up to the deadline. Only the most recent submission is saved by cuLearn.

Solutions that are emailed to your instructor or a TA will not be graded, even if they are emailed before the deadline.

Last edited: April 7, 2021 (initial release)