CSC3310 — Program #2 — Divide and Conquer

Objectives

- Design a divide-and-conquer algorithm for a given computational problem
- Justify the correctness of an algorithm
- Perform asymptotic complexity analysis of the run time
- Design and execute benchmarks for an algorithm

Overview

For this assignment, you will apply the divide-and-conquer strategy to create an algorithm that solves the selection problem.

The Problem

The problem that you will be solving is called the SELECTION problem.

> SELECTION
> **Input** A sequence of $n \geq 2$ numbers and an integer k, $1 \leq k \leq n$
>
> **Output** The k^{th} smallest number in the sequence.

For example, if we are given the sequence $[5, 1, 6, 7, 3, 4, 8]$ and $k = 3$, we would return 4, because 4 is the 3^{rd} smallest number, after 1 and 3.

The simplest way to solve this problem is to sort the sequence (or at least the first k elements) and take the element at index $k - 1$. This will take $\mathrm{O}(n \log n)$ time. The time can be reduced to $\mathrm{O}(kn)$ or $\mathrm{O}(k \log n)$ if we only partially sort.

Your goal is to create an even faster algorithm using a divide-and-conquer strategy. **Hint:** If you select a pivot value, you can partition the array into the j numbers smaller than the pivot and the $n - j$ numbers that are larger than the pivot.

Deliverables

Complete a lab report containing the following:

- A paragraph describing the approach you will use to solve the problem. Provide at least 2 illustrations that explain the approach.
- High-level pseudocode for an algorithm that uses that rule to solve the computational problem for any input
- An explanation and justification for why your algorithm is correct (1-3 paragraphs)
- A table of your test cases, the answers you expect, and the answers returned by running your implementation of the algorithm.
- The derivation of a recurrence relation describing the run time in terms of the number of points n. (Show your work!) You may assume that the random pivot divides the elements in half each time.
- A solution to the recurrence relation (show your work). Ideally, you will get a run time in terms of $\mathrm{O}(n)$ in asymptotic notation.
- A table and graph from benchmarking different lists with different sizes and values of k. Benchmark your implementation versus an approach that sorts the numbers and picks the element at index $k - 1$.

  If the benchmarks do not support your theoretically-derived run time and/or do not provide evidence that the run time of your algorithm grows more slowly than the sorting approach, this may indicate a flaw in your implementation.
- Include code and report together in a pdf of a Jupyter notebook.

  (or else as an appendix containing all of your source code and test cases.)

  Submit your report to Canvas in PDF format (make sure source code is included).

Rubric

		Full Credit	Partial Credit	No Credit
In class participation	5%			
Lab report writing and presentation quality	10%			
Solution Approach Description	10%	Approach is well-described, with all necessary details to implement and/or to explain to someone else	Approach is decently described with most details necessary to implement. Explanation may be unclear in places	The description is insufficient to implement or explain to others.
Algorithm, as described in Pseudocode	10%	Algorithm is correct for all allowed inputs	Algorithm is correct for most inputs	Algorithm is incorrect for many inputs
Justification of Correctness	10%	Uses techniques described in class to provide a solid and convincing argument that the algorithm is correct	Provides a somewhat convincing argument that the algorithm is correct	Argument contains one or more serious flaws
Asymptotic run-time analysis	10%	Analysis is correct for the provided pseudocode	Analysis contains minor flaws	Analysis is significantly flawed
Algorithm Implementation	10%	Implementation is both faithful to the pseudocode and correct	Implementation is mostly faithful to the pseudocode and/or correct for most inputs	Implementation is not faithful to the pseudocode or not correct for common inputs
Test Cases	10%	Test cases consider a range of problem sizes, complexities, classes, and edge cases	Limited number of test cases only testing obvious or simple cases	Only the examples
Benchmarking	10%	Benchmark experiments were set up and implemented correctly.	Benchmark experiments, implementations, or results are mostly correct.	Benchmark experiments, implementations, or results are flawed.
Algorithmic Runtime	10%	Runtime is asymptotically *faster than* $\mathrm{O}(k \log n)$, as determined by both empirical results and theoretical analysis	Runtime is asymptotically *faster than* $\mathrm{O}(n \log n)$	Runtime is asymptotically equal-to or slower-than $\mathrm{O}(n \log n)$ brute-force search
Submitted correctly	5%			