CMPS 2200 Assignment 4

In this assignment we'll look at randomness in computation, both in theory and in practice.

To make grading easier, please place all written solutions directly in `answers.md`, rather than scanning in handwritten work or editing this file.

All coding portions should go in `main.py` as usual.

Part 1: Selection

Suppose that you had an algorithm $\mathcal{A}$ for finding the median of an unsorted list that requires $O(n)$ work. Design an algorithm that, given an unsorted list L and a rank k, finds the element of rank k using $O(n)$ work.

Part 2: Deviation from Expectations

We learned in lecture that Quicksort takes $O(n \log n)$ expected work. A fair question is how tight that expectation is. Luckily we have some bounds that allow us to look at this question. For a random variable X, Markov's inequality states that:

$$\mathbf{P}[X \geq \alpha] \leq \frac{\mathbf{E}[X]}{\alpha}$$

2a) What is the probability that Quicksort does $\Omega(n^2)$ comparisons? .

.

.

.

2b) What is the probability that Quicksort does $10^c n \ln n$ comparisons, for a given $c > 0$? What does this say about the "concentration" of the expected work for Quicksort? .

.

.

.

Part 3: From "Maybe" to "Definitely"

At your new job designing algorithms for really hard problems, you're put to work solving problem X. Your predecessor has left you with an algorithm $\mathcal{A}$ for problem X that has a deterministic worst-case work, but only produces the correct output with a certain probability of success. Moreover, we can also check whether the correct result was produced with $O(W(n))$ work in the worst case.

Let $\mathcal{A}(\mathcal{I})$ denote the output of an algorithm $\mathcal{A}$ on input $\mathcal{I}$. So $\mathcal{A}(\mathcal{I})$ has a probability of ϵ of being correct and a failure probability of $1 - \epsilon$. Furthermore let $\mathcal{C}(\mathcal{A}(\mathcal{I}))$ denote the output of (deterministically) checking $\mathcal{A}$'s solution.

3a) You find that ϵ is too small to be reliable. You want to be able to have *any* guaranteed success probability δ, for $\epsilon < \delta < 1$. Use $\mathcal{A}$ to construct an algorithm $\mathcal{A}'$, where $\mathcal{A}'(\mathcal{I}, \delta)$ is the correct output with probability δ. It is sufficient to give a high level description of $\mathcal{A}'$. What is the work of $\mathcal{A}'$ in terms of n, δ, and ϵ? (**Hint**: Each run of $\mathcal{A}$ is independent and does not depend on previous runs.)

.

.

.

.

3b) Your boss and co-workers are impressed, but you want to do even better. Show how to convert $\mathcal{A}$ into an algorithm that always produces the correct result, but has an expected runtime that depends on $W(n)$ and the success probability ϵ. .

.
.
.

Part 3: Determinism versus Randomization in Quicksort

In lecture we saw that adding a random choice of pivot reduced the probability of worst-case behavior for any given input in selection. Let's look at how pivot choices affect Quicksort. For this question, refer to the code in `main.py`

3a)

Complete the implementation of the stub `qsort` implementation. Feel free to take from code given in the lectures to help you perform list partitioning. You'll note that implementation is such that an input argument controls whether the pivot selection is done deterministically or at random. .

.
.
.

3b)

Compare running times using `compare-qsort`. What differences in performance do you observe when comparing a random choice of pivot with a deterministic one. How do the running times compare to the asymptotic trend of $O(n \log n)$?