CS1200: Intro. to Algorithms and their Limitations Anshu & Vadhan

Problem Set 4

Harvard SEAS - Fall 2024 *Due: Wed Oct. 9, 2024 (11:59pm)*

Your name:
Collaborators:
No. of late days used on previous psets:
No. of late days used after including this pset:

1. (Randomized Algorithms in Practice)

   (a) Implement Randomized QuickSelect, filling in the template we have given you in the Github repository.

   (b) In the repository, we have given you datasets x_n of key-value pairs of varying sizes to experiment with: dataset x_n is of size n. For each dataset x_n and any given number k, we will consider how to efficiently answer the k selection queries `select`$(x_n, 0)$, `select`$(x_n, [n/k])$, `select`$(x_n, [2n/k])$, ..., `select`$(x_n, [(k-1)n/k])$ on x_n, where $[\cdot]$ denotes rounding to the nearest integer. For example, if $k = 4$, then we release the minimum, the 25th percentile, and the 75th percentile of the dataset. You will compare the following two approaches to answering the queries:

      i. Running (randomized) `QuickSelect()` k times.
      ii. Running `MergeSort()` (provided in the repository) once and using the sorted array to answer the k queries.

      Specifically, you will compare the *distribution* of runtimes of the two approaches for a given pair (n, k) by running each approach many times and creating density plots of the runtimes. The runtimes will vary because `QuickSelect()` is randomized, and because of variance in the execution environment (e.g. other processes that are running on your computer during each execution).

      We have provided you with the code for plotting. Before plotting, you will need to implement `MergeSortSelect()`, which extends `MergeSort()` to answer k queries. Your goal is to use these experiments and the resulting density plots to propose a value for k, denoted $k^*(n)$, at which you should switch over from `QuickSelect()` to `MergeSortSelect()` for each given value of n (you can choose any reasonable statistical feature to propose $k^*(n)$, such as the peak runtime of the distribution or the mean runtime, etc). Do this by experimenting with the parameters for k (code is included to generate the appropriate queries once the k's are provided) and generate a plot for each experiment. Explain the rationale behind your choices, and submit a few density plots for each value of n to support your reasoning. (There is not one right answer, and it may depend on your particular implementation of `QuickSelect()`.)

   (c) Extrapolate to come up with a simple functional form for $k^*(n)$, e.g. something like $k^*(n) = 3\sqrt{n} + 6$ or $k^*(n) = 10 \log^2 n$. (Again there is not one right answer.) Briefly discuss how your extrapolation aligns with theoretical values. That is, what kind of functional form for $k^*(n)$ (in asymptotic notation) would be predicted by the asymptotic

runtimes of `QuickSelect()` and `MergeSortSelect()` for answering k selection queries on a dataset of size n?

(d) (*optional) One way to improve `QuickSelect()` is to choose a pivot more carefully than by picking a uniformly random element from the array. A possible approach is to use the ***median-of-3*** method: choose the pivot as the median of a set of 3 elements randomly selected from the array. Add `Median-of-3 QuickSelect()` to the experimental comparisons you performed above and interpret the results. That is, in what way (if any) does `Median-of-3 QuickSelect()` offer benefits over `QuickSelect()`?

2. (Dictionaries and Hash Tables) Consider the following computational problem:

Input	**:** An array $(a_0, \ldots, a_{n-1})$ of natural numbers (each fitting in one word).
Output	**:** A duplicate element; that is, a number a such that there exist $i \neq j$ such that $a_i = a_j = a$.

Computational Problem DuplicateSearch

(a) Show that DuplicateSearch can be solved by a Las Vegas algorithm with expected runtime $O(n)$ using a dictionary data structure. (You should prove correctness and analyze runtime quoting the expected runtimes stated for the Dictionary data structure in Lecture 9, but you do not need to do a formal probability calculation using expectations.)

(b) DuplicateSearch can be solved by a deterministic algorithm in runtime $O(n \log n)$. Briefly describe this algorithm in 2-3 sentences (you do not need to write a pseudocode and do not need to provide a proof of correctness).

3. (Choosing Algorithms and Data Structures) Suppose the US Census Bureau was going to develop a new database to keep track of the exact ages of the entire US population, and publish statistics on it. The data it has on each person is an exact birthdate `bday` (year, month, and date) and a unique identifier `id` (e.g. social security number — pretend that these are assigned at birth).

For each of the three scenarios below,

(a) Select the best algorithm or data structure for the Census Bureau to use from among the following:

- sorting and storing the sorted dataset
- storing in a binary search tree (balanced and possibly augmented)
- storing in a hash table
- running randomized QuickSelect.

(b) Explain how you would use the algorithm or data structure (including any necessary augmentations) to solve the stated problem, e.g. what would you take as keys and values, what updates and queries (in case you use a dynamic data structure) would you issue, and how you would read off the results to obtain the desired statistics.

(c) State what the runtime would be as a function of all of the relevant parameters: the size n of the US population being surveyed, the number u of updates issued at the specified time intervals, and/or the number s of statistics released at the specified time intervals.

(These parameters are not all freely varying in the parts below, e.g. s may be a fixed constant or a function of n; state any such constraints in your answers.)

In each scenario, you should assume (unrealistically) that the described queries or statistics are the *only* way in which the data is going to be used, so there is no need to support anything else.

(a) (Reporting Age Rankings) Every ten years as part of the Decennial Census, the Bureau collects a fresh list of (`id`, `bday`) pairs from the entire US population. (It does not reuse data from the previous Decennial Census, so everyone is re-surveyed.) In order to incentivize participation, the Bureau promises to tell every respondent their age-ranking in the population after the survey is done (e.g. "you are the 796,421'th oldest person among those who responded to the Census").

(b) (Daily Quartiles) After each day, the Bureau obtains a list of (`id`, `bday`) pairs to add to or remove from its database due to births, deaths, and immigration, and publishes an updated 25th, 50th, and 75th percentile of the population ages.

(c) (Age Lookups) For privacy reasons, the Bureau decides to not publish any statistics on the population ages, but just wants to maintain a database where the age of any member of the population can be looked up quickly, and the database can be quickly updated daily according to births, deaths, and immigration.

4. (Reflection) Skim the course material from the beginning of the course through Lecture 9 (i.e. the scope of the upcoming class midterm). Identify one concept or skill that you would like to study or practice in greater depth, and discuss why. It can be because you feel that you haven't fully understood or internalized it, or because you found it interesting and are curious to learn more, or any other motivation you have.

   Note: As with the previous psets, you may include your answer in your PDF submission, but the answer should ultimately go into a separate Gradescope submission form.

   Quick note on grading: Good responses are usually about a paragraph, with something like 7 or 8 sentences. Most importantly, please make sure your answer is specific to this class and your experiences in it. If your answer could have been edited lightly to apply to another class at Harvard, points will be taken off.

5. Once you're done with this problem set, please fill out this survey so that we can gather students' thoughts on the problem set, and the class in general. It's not required, but we really appreciate all responses!