CS 5150/6150 Graduate (Advanced) Algorithms
Fall 2022
Assignment 4: Randomized Algorithms
Total Points: 80

Submission notes

- Due at 11:59 pm on **Thursday, October 27, 2022**.
- Solutions must be typeset (not hand-written or scanned).
- You should strive to write clear, concise solutions to each problem, ideally fitting on a page or less. The easier your work is to read & follow, the easier it is for the TAs to award you points.
- Upload a PDF version of your completed problem set to Gradescope.
- Teaching staff reserve the right to request original source/tex files during the grading process, so please retain these until an assignment has been returned.
- Please remember that for problem sets, *collaboration with other students must be limited to a high-level discussion of solution strategies.* If you do collaborate with other students in this way, you must identify the students and describe the nature of the collaboration at the top of your homework submission. **You are not allowed to create a group solution, and all work that you hand in must be written in your own words. Do not base your solution on any other written solution, regardless of the source.**

Instructions on How to Answer Algorithm Design Questions

Most homework questions in this semester will be algorithm design questions. Please read the following instructions on how to answer them properly (you may lose points if you do not follow the instructions).

1. **Algorithm Description** You are required to clearly describe the main idea of your algorithm.

2. **Pseudocode** Providing pseudocode for your algorithm is optional. However, pseudocode is very helpful for explaining algorithms clearly, and you are *strongly encouraged* to provide pseudocode for your algorithms. Additionally, having pseudocode available often leads to more partial credit for incorrect solutions, as the TAs can more easily identify correct ideas in your solution.

   Note: copying code from an implementation (e.g. python/java/c++) as "pseudocode" does not satisfy this requirement, as grading it is difficult, and key steps may be difficult to identify. In principle, pseudocode should be relatively easy to read while keeping all relevant operations of the algorithms. *You are encouraged to do your best to write good pseudocode, following examples given in class.*

3. **Correctness** You also need to explain/argue why your algorithm is correct, and handles all necessary cases – especially when the correctness of your algorithm is not obvious.

4. **Time Analysis** You are required to analyze the time complexity of your algorithm (i.e., provide a written justification for the claimed asymptotic runtime in terms of the algorithm's operations, not just state the time complexity). You may refer to your pseudocode in this analysis, as it often simplifies the descriptors (e.g. "the `for` loop on line 2").

1. **(20 points)** Consider a swarm of bumblebees looking to collect nectar/pollen from a patch of clover. Each flower has plenty of nectar, and can feed as many bees as land on it. However, because clover stems are not very sturdy, only one bee can collect from each flower at a time. Further, the clover heads are functionally identical to the bees, so they have no preference for one flower over another. If two bees try to land on the same flower simultaneously, they both fly back up into the swarm, and then arbitrarily choose a new target. We want to analyze how long it will take for all the bees in the swarm to successfully collect nectar/pollen. We will model this problem using rounds. In each round, all remaining bees independently and uniformly at random choose a target clover flower to land on for collection. All attempted landings occur simultaneously. If a flower was chosen by exactly one bee, the bee successfully lands, collects the nectar/pollen, and flies away (back to the colony). If a flower was chosen by more than one bee, there is a collision and all bees which selected that flower fail to collect (and rejoin the swarm for the next round). We begin with n bumblebees in round 1, assume there are n clover flowers in the patch, and finish when the entire swarm has collected food and flown away.

   (a) (10 points) If there are b bees in the swarm at the start of round i, what is the expected number of bees remaining at the start of round $i + 1$?

   (b) (10 points) Let x_j be the expected number of bees left in the swarm after round j has completed. One can prove (and you may assume) that $x_{j+1} \leq x_j^2/n$. Use this to argue that if in every round the number of bees remaining in the swarm was exactly x_j, then the process would finish in $O(\log \log n)$ rounds. *[Hint: you will need a base case - consider the outcome after one or two rounds.]*

2. **(20 points)**

(a) [8] Suppose we invite n people to a costume party. At the door, you have a table with stacks of m different masks. As they arrive, each guest is given a name-tag with their arrival number (the first guest gets 1, the second 2, etc). They then spin a giant wheel to determine (uniformly at random) which of the m mask styles they will wear, are handed one according to the outcome, and join the party. Assume that you never run out of any style of mask. What is the expected number of pairs (i, j) with $i < j$ such that the guest with name-tag i and the guest with name-tag j are wearing matching masks? For what value of n (as a function of m) does this number become 1?

(b) [12] This idea has some nice applications in CS, one of which is in estimating the "support" of a distribution. Suppose the neighborhood elementary school ran a fall fundraiser by selling pumpkin carving kits to local families. The company that assembled the kits claims to have a library of over one million different designs, and randomly selects one from their collection (with replacement) to print for each kit made. The fundraiser was very successful, and sold 200 kits to local families. However, while out trick-or-treating, you notice that two of the houses on the next street both received the same template design. Prove that the probability of this having happened (if the template library really has a million designs) is very small – your bound must be less than 0.05 to receive full credit.

Note: This casts doubt on the legitimacy of the company's claim about their library size / kit-making process, and should be considered when planning next year's fundraiser. More generally, this idea has many applications in CS, for estimating the size of sets without actually enumerating them.

Note: This problem is based on the lectures that will be given in the week right after fall break.

3. **(20 points)** Here is a generalized version of the selection problem, called *multiple selection*. Let $A[1 \cdots n]$ be an (unsorted) array of n numbers. Given a sequence of m sorted integers $k_1, k_2, \ldots, k_m$, with $1 \leq k_1 < k_2 < \cdots < k_m \leq n$, the *multiple selection problem* is to find the k_i-th smallest number in A for all $i = 1, 2, \ldots, m$. For simplicity, we assume that no two numbers of A are equal.

   For example, let $A = \{1, 5, 9, 3, 7, 12, 15, 8, 21\}$, and $m = 3$ with $k_1 = 2$, $k_2 = 5$, and $k_3 = 7$. Hence, the goal is to find the 2nd, the 5-th, and the 7-th smallest numbers of A, which are 3, 8, and 12, respectively.

   You are asked to give a randomized algorithm for the problem. Your randomized algorithm should be a Las Vegas one (i.e., it always produces the correct answer) and the expected running time of your algorithm should be $O(n \log m)$. Note that this is better than an $O(n \log n)$ time algorithm when m is much smaller than n (e.g., when $m \leq \log n$).

Note: This problem is based on the lectures that will be given in the week right after fall break.

4. **(20 points)** Let $A[1 \cdots n]$ be an (unsorted) array of n elements. We assume that no two elements of A are equal. We are also given another number M. The problem is to find the *largest* element x in A such that the sum of the elements of A less than x is at most M (i.e., $\sum_{a \in A, a<x} a \leq M$). We assume that the sum of all elements of A is larger than M.

   For example, suppose $A = \{4, 8, 5, 2, 3, 6, 1\}$, and $M = 10$. Then, the largest such element x we are looking for is 5.

   You are asked to give a randomized algorithm for the problem. Your randomized algorithm should be a Las Vegas one (i.e., it always produces the correct answer) and the expected running time of your algorithm should be $O(n)$.