CS 466/666: Algorithm Design and Analysis **University of Waterloo: Fall 2025**

Homework 1

Due: Thursday, September 25, 2025

Homework Policy

- If you leave a question completely blank, you will receive 25% of the grade for that question. This however does not apply to the extra credit questions. These problems are more challenging than the standard problems you see in this course.
- Refer to your course outline for the policies regarding extensions, submission format, groups, etc.

Problem 1. Suppose we throw a uniform six-sided die until we see the number six for the first time and let X be the random variable for this number.

(*a*) Compute $\mathbb{E}[X]$ and $\mathrm{Var}[X]$ exactly. **(5 points)**

(*b*) Suppose we are additionally told that all throws have landed on even numbers. Compute the conditional expected value of X in this case. More precisely, let E be the event that all throws of the die before we stop have been even numbers. What is $\mathbb{E}[X \mid E]$? **(15 points)**

Hint: This is a tricky question, so do <u>not</u> trust what your intuition tells you without a solid proof ...

Problem 2. Recall the streaming distinct elements problem from Lecture 3: we have a stream of n numbers $e_1, \ldots, e_n$ from a universe of $[m]$. The stream can have repeated numbers and the goal is to estimate the number of distinct elements denoted by DE. Specifically, given a parameter $\varepsilon \in (0, 1/4)$, our goal is to output an estimate $\widetilde{\mathsf{DE}}$ such that

$$\Pr\left((1-\varepsilon)\cdot \mathsf{DE} \leqslant \widetilde{\mathsf{DE}} \leqslant (1+\varepsilon)\cdot \mathsf{DE}\right) \geqslant 2/3,$$

using a small space algorithm.

Our goal in this question is to analyze another algorithm for this problem that the one presented in the class. In particular, this algorithm directly solves the problem without the need for the threshold testing part first and then running the test for different thresholds.

Algorithm 1. Another streaming algorithm for the distinct elements problem.

1. Pick a pairwise independent hash function $h : [m] \to [m]$.
2. Store the $t = 100/\varepsilon^2$ *distinct* numbers in the stream with the *smallest* hash values[a].
3. Let X be the random variable for the *largest* hash value stored in the previous step.
4. Return $\widetilde{\mathsf{DE}} = m \cdot t/X$.

[a] The constant 100 is chosen as a very large constant here to make the calculations in the proof easier.

(*a*) Prove that if $\widetilde{\mathsf{DE}} > (1+\varepsilon)\cdot\mathsf{DE}$, then we should have $X < \frac{m\cdot t}{(1+\varepsilon)\cdot\mathsf{DE}}$. **(5 points)**

(*b*) Let Y be the random variable for the number of distinct elements in the input that are hashed to the numbers in $[\frac{mt}{(1+\varepsilon)\cdot\mathsf{DE}}]$. Prove that if $X < \frac{m\cdot t}{(1+\varepsilon)\cdot\mathsf{DE}}$, then we have $Y \geqslant t$. **(5 points)**

(*c*) Prove that $\mathbb{E}[Y] = t/(1+\varepsilon)$ and $\mathrm{Var}[Y] \leqslant \mathbb{E}[Y]$. **(5 points)**

(*d*) Use the above parts plus the Chebyshev's inequality to prove that **(10 points)**

$$\Pr\left(\widetilde{\mathsf{DE}} > (1+\varepsilon)\cdot\mathsf{DE}\right) \leqslant 1/6.$$

(*e*) Repeat the above steps for the case when $\widetilde{\mathsf{DE}} < (1-\varepsilon)\cdot\mathsf{DE}$ and prove that **(10 points)**

$$\Pr\left(\widetilde{\mathsf{DE}} < (1-\varepsilon)\cdot\mathsf{DE}\right) \leqslant 1/6.$$

(*f*) Combine the above algorithm and also analyze the space complexity of the algorithm to prove the algorithm solves the distinct elements problem using $O(\log m/\varepsilon^2)$ bits of space. **(5 points)**

Problem 3. Consider a complete tree of height h, wherein the root, as well as any internal node has exactly 3 child-nodes; thus, the tree has $n = 3^h$ leaves. Suppose each leaf of the tree is assigned a Boolean value. We define the value of each internal node as the *majority* of the value of its child-nodes. The goal in this problem is to determine the value of the root.

An algorithm for this problem is provided with the structure of the tree (not the valuation of the leaves) and at each step it can *query* a leaf and read its value.

(*a*) Show that for any deterministic algorithm, there is an instance (a set of Boolean values for the leaves) that forces the algorithm to query all the $n = 3^h$ leaves.

(10 points)

(*b*) Consider the recursive randomized algorithm that evaluates two subtrees of the root chosen at random. If the values returned disagree, it proceeds to evaluate the third subtree. Show that the expected number of the leaves queried by the algorithm on any instance is at most $n^{0.9}$. **(10 points)**

Problem 4. Given a set of numbers S and a number $x \in S$, the **rank** of x is defined to be the number of elements in S that have value at most x:

$$rank(x, S) = |\{y \in S : y \leqslant x\}|$$

Given a parameter $\varepsilon \in (0, 1/2]$, we say that an element $x \in S$ is an **ε-approximate element of rank** r if

$$(1-\varepsilon)\cdot r \leqslant rank(x, S) \leqslant (1+\varepsilon)\cdot r$$

Recall the streaming model of computation discussed in the class. Suppose we are given a stream of numbers $S = (s_1, s_2, \ldots, s_n)$, where $s_i \in [m]$ for $i \in [n]$, and assume that all s_i's are distinct. Our goal is to design an $O(\varepsilon^{-2}\log m \log n)$ space streaming algorithm for retrieving an ε-approximate element for any given rank value.

(*a*) Recall that the median of a set S of n (distinct) elements is the element of rank $r = \lfloor n/2 \rfloor$ in S.

Consider this algorithm for computing an ε-approximate median: sample $O(\varepsilon^{-2} \log n)$ numbers from the stream uniformly at random (with repetition) and then return the median of the sampled numbers. Prove that this algorithm returns an ε-approximate median with probability at least $1 - 1/\text{poly}(n)$.

(10 points)

(*b*) We now extend the previous algorithm to compute an ε-approximate element of rank r for any $r \in [n]$.

Consider this algorithm: Let $t = \lceil 24\varepsilon^{-2} \log n \rceil$. If $r \leqslant t$, then simply maintain a list T of r smallest elements seen in the stream, and output the largest element in T at the end of the stream. Otherwise, choose each element in the stream with probability t/r, and maintain the t smallest sampled values in a list T. At the end of the stream, output the largest number in T. Prove that this algorithm outputs an ε-approximate element of rank r with probability at least $1 - 1/\text{poly}(n)$. **(10 points)**

Problem 5 (**Extra credit**)**.** Consider Problem 1 again. What is the variance of the random variable X conditioned on the event E in part (*b*), i.e., what is $\text{Var}[X \mid E]$? **(+10 points)**

Problem 6 (**Extra credit**)**.** In the $(\deg +1)$ coloring problem, we are given an undirected graph $G = (V, E)$ and the goal is to find a coloring of vertices of G such that (*i*) no edge is monochromatic, and (*ii*) every vertex $v \in V$ receives a color from the set $\{1, 2, \ldots, \deg(v) + 1\}$ where $\deg(v)$ is the degree of v in G. The difference of this problem with the $(\Delta + 1)$ coloring problem we saw in Lecture 1 is that vertices that have a lower degree here can only receive a color from a smaller range of colors as well (as opposed to all vertices having access to the same $(\Delta + 1)$ colors).

Suppose we are given access to both adjacency list and adjacency matrix of G (and we can read degree of each vertex from its adjacency list in $O(1)$ time). Modify the $(\Delta + 1)$ coloring algorithm of Lecture 1 to solve the $(\deg +1)$ coloring problem in $O(n\sqrt{n \log n})$ expected time. **(+15 points)**