HW 7

Due on **Wednesday, October 18, 2023 at 9pm**

CS 473: Algorithms, Fall 2023 Version: **1.21**

Submission instructions: As in previous homework.

19 (100 PTS.) Computing the maximum.

You are given a sequence of n random variables $X_1, \ldots, X_n$ drawn independently from the same distribution (say the values are picked uniformly at random from $[0, 1]$). Let $Y_i = \max_{\ell=1}^{i} X_i$. We are interested in computing $Y_n = \max_i X_i$.

19.A. (50 PTS.) You are given the following theorem (seen in class):

Theorem 7.1. *The QuickSort algorithm has recursion depth $c \log n$, with probability $\geq 1 - 1/n^{10}$, where c is some absolute constant.*

Using this theorem (and only this theorem), prove that with high probability, the set

$$\{Y_1, Y_2, \ldots, Y_n\}$$

has $O(\log n)$ distinct values.

19.B. (50 PTS.) Assume you are given $X_1, X_2, \ldots, X_n$ in a stream – that is, at time i, you are given the value of X_i, but you do not have the space to store the whole sequence. Importantly, you know that all the variables $X_1, \ldots, X_n$ are sampled independently from the same distribution, but you do not know what the distribution is. Computing the maximum is still easy – you just maintain the maximum seen so far in a variable, say z. To make it more interesting, let us change the game – conceptually, the variable z can be assigned a value only once.

Specifically, you are given the value X_i at time i. Either, you decide at this time i that X_i is your candidate to be the (global) maximum, and you output it (and you can do it only once), or alternatively, you decide to wait for later values. If you do not output a value, it is lost, and you can not output it later on.

Assume that you know n in advance. Describe an algorithm, that with constant probability outputs the maximum. Prove a lower bound (naturally, as large as possible) on the probability that your algorithm outputs the maximum.

20 (100 PTS.) Hash tree.

You are given a set $S \subseteq \{0, 1, \ldots, p-1\}$ of n values, where p is a prime. We store the values of S in a ***hash tree*** constructed recursively. The root of the tree is going to be a hash table of size $\lceil \sqrt{n} \rceil$, where the hash function used is sampled from a 2-universal hash family (assume you have access to such family, as described in class). All the values that are mapped to the ith bucket of this hash table, say S_i, are going to be stored in their own hash tree, which is constructed recursively on the set S_i (i.e., in the recursive construction for S_i, we will have $n = |S_i|$). Describe an algorithm to construct a hash tree, so that the height of the hash tree is as small as possible. Prove an upper bound on the height of the hash tree, and bound the expected running time and space used by your algorithm.

Naturally, in the bottom of the recursion (i.e., $n = O(1)$), the algorithm uses (say) a list to store the items.

21 (100 PTS.) How many unique values?

You are given a stream of non-negative integer numbers, $x_1, \ldots, x_n$, in a stream (they are not random). For simplicity, you can assume all the numbers in the stream are smaller than some integer M that is provided in advance. While n is very large, the number of unique values $\tau = |\{x_1, \ldots, x_n\}|$ is small (but not known to you in advance).

21.A. (50 PTS.) You are given a guess k to the value of τ [you can assume $k = O(\sqrt{n})$]. Describe an algorithm that uses 2-universal hash family (note, that you can not use hashing as a black box here), and polynomial space in k (the space used should be as small as possible), and with a single pass on the stream, outputs the number τ (assuming that $\tau \leq k$). If $\tau > k$ then the algorithm would fail. The algorithm has to succeed with constant probability if $\tau \leq k$. The algorithm needs to have running time $O(n + k^{O(1)})$.

21.B. (50 PTS.) Describe an algorithm, using the algorithm from the previous part as a subroutine, that in expectation, performs $O(\log \tau)$ passes over the steam, and computes τ. The algorithm is allowed to use space that is polynomial in τ (say in expectation).