CS 466/666: Algorithm Design and Analysis **University of Waterloo: Fall 2024**

Homework 1

Due: Thursday, September 26, 2024

Problem 1. Suppose we pick a permutation π from $[n]$ to $[n]$ uniformly at random.

(a) Let X be the random variable for the number of elements in π such that $\pi(i) = i$.
Compute $\mathbb{E}[X]$ and $\mathrm{Var}[X]$ exactly. **(5 points)**

(b) Let Y be the random variable for the number of indices i such that if $\pi(i) = j$, then i is the largest number among entries 1 to j in the image of π. For instance, for the permutation π of [5] below, all such elements are marked and $Y = 2$:

$$[\underline{\mathbf{3}}, 2, 1, \underline{\mathbf{5}}, 4].$$

Prove that $\mathbb{E}[Y] \leqslant \ln(n)$. **(5 points)**

(c) Let Z be the random variable for the length of the longest increasing subsequence (LIS) in π. For instance, for the permutation π of [5] below, the elements of one LIS are marked and we have $Z = 3$:

$$[\underline{\mathbf{2}}, \underline{\mathbf{3}}, 1, \underline{\mathbf{5}}, 4].$$

Prove that $\mathbb{E}[Z] = O(\sqrt{n})$. **(10 points)**

Hint: First show that for every (non-negative) integer valued random variable X, $\mathbb{E}[X] = \sum_{i=1}^{\infty} \Pr(X \geqslant i)$. Then, for any $i \in [n]$, find an upper bound on $\Pr(Z \geqslant i)$ using Stirling approximation.

Problem 2. A family of functions $\mathcal{H} = \{h \mid h : [n] \mapsto [m]\}$ is called a **pairwise independent hash family** iff for all $x \neq y \in [n]$ and $a, b \in [m]$, for h chosen uniformly at random from $\mathcal{H}$, we have,

$$\Pr(h(x) = a \wedge h(y) = b) = \frac{1}{m^2}.$$

In this problem, we investigate one method to obtain such a hash family (although this is *not* the most efficient method).

(a) Let $X_1, X_2, \ldots, X_k$ be independent and uniform binary random variables, i.e.,

$$\Pr(X_i = 0) = \Pr(X_i = 1) = 1/2.$$

Given $S \subseteq \{1, \ldots, k\}$, $S \neq \emptyset$, define a random variable $Y_S = \oplus_{i \in S} X_i$, i.e., the XOR of X_i's for $i \in S$. Prove that the set

$$\{Y_S \mid S \subseteq \{1, \ldots, k\}, S \neq \emptyset\}$$

is a collection of **pairwise independent random variables**, meaning that for all non empty sets $S \neq T \subseteq \{1, \ldots, k\}$ and pairs $a, b \in \{0, 1\}$:

$$\Pr[Y_S = a \wedge Y_T = b] = \frac{1}{4}.$$

(12.5 points)

(b) Use the construction in part (a) to design a family of pairwise independent hash functions

$$\mathcal{H} = \{h \mid h : [n] \mapsto [m]\}$$

constructed using $O(\log n \log m)$ random bits and $\text{poly}(\log n, \log m)$ time to compute. You may assume that both n and m are powers of two. **(12.5 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 $n = 3^h$ nodes. 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 $n^{0.9}$. **(15 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 $1 - 1/\text{poly}(n)$.

(15 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 $t = \lceil 24\varepsilon^{-2} \log m \rceil$. 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 $1 - 1/\text{poly}(n)$. **(15 points)**

Problem 5 (Extra credit). Consider Problem 1 again. What is the best asymptotic upper bound on $\text{Var}[Y]$ for the random variable Y in part (b) that you can establish? **(+5 points)**

What about the random variable Z? **(+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)**