CS4810 / CS7800: Advanced Algorithms — Fall 2022
Homework 6: Due Friday, December 9, 2022

Mahsa Derakhshan and Jonathan Ullman

Collaboration and Honesty Policy Reminder. Collaboration in the form of discussion is allowed and encouraged. However, all forms of cheating are not allowed and will be penalized harshly. These include, but are not limited to, copying parts of an assignment from a classmate, finding answers to problems on the internet or from anyone not enrolled in the class, and plagiarizing from research papers or old posted solutions. A rule of thumb is that you should be able to walk away from discussing a homework problem with no notes and write your solution on your own.

- You must write up all solutions by yourself, and may not share any written solutions, even if you collaborate with others to solve the problem.
- You must identify all your collaborators. If you did not collaborate, write "no collaborators" or something to that effect. You may have a maximum of two collaborators per assignment, and collaboration is transitive (if you list a collaborator they must list you).
- Asking and answering questions in class forums (lectures, office hours, Piazza) is allowed and encouraged, and you do not need to list these interactions as collaborators.
- Seeking out alternative sources (e.g. classmates, textbooks, the internet) for general concepts you need (e.g. greedy algorithms, probability) is allowed and encouraged.

Assigned Problems

Problem 1 In class we only studied hash tables that store a set of items $S = \{x_1, \ldots, x_n\} \subseteq \mathcal{U}$ so that we can check very quickly if an element $x \in S$ or not. These data structures used close to the minimum amount of space $n \log_2 |\mathcal{U}|$. In this problem you will build a data structure that can save space by storing S *approximately*, meaning we allow for some mistakes when checking if $x \in S$.

Consider the following data structure:

- Choose k *independent, uniformly random* hash functions $h_1, \ldots, h_k : \mathcal{U} \to [m]$.[1]
- Create an array $T[1 : m]$ and initialize $T[j] = 0$ for every j.
- For each $x_i \in S$ and each $\ell = 1, \ldots, k$: set $T[h_\ell(x_i)] = 1$.

Given the array T, we can check to see if $x \in S$ as follows:

- If $T[h_\ell(x)] = 1$ for every $\ell = 1, \ldots, k$, return TRUE, else return FALSE.

Ideal random hash functions would require an unreasonable amount of space, but it makes it possible to analyze this data structure, so we will ignore this issue for the rest of the question and assume that the number of bits required for this data structure is just m.

[1] That is, we choose each h uniformly from the set of all functions from $\mathcal{U}$ to $[m]$.

1.1 Prove that this data structure has no *false negatives*, that is, if $x \in S$, then the data structure will return TRUE for every choice of the random hash functions $h_1, \ldots, h_k$.

1.2 Calculate $\mathbb{P}(T[j] = 0)$ after we have constructed the table T. You should allow S to be any set, and the only thing random in your analysis should be the hash functions $h_1, \ldots, h_k$.

1.3 Prove an upper bound on the probability of a *false positive* in terms of k, m, and n. That is, the probability that the data structure returns TRUE when we look for $x \notin S$.[2]

1.4 Show that for some choice of k and some choice of $m = O(n \log(1/\varepsilon))$ we can make the probability of a false positive at most ε.

Putting these steps together, we can store an approximate version of S using just $O(n \log(1/\varepsilon))$ bits, while ensuring no false negatives and a false positive rate of ε.

Problem 2 In this problem you will build a streaming algorithm for a very simple problem—counting the number of 1's in a stream of bits. You are given a stream of bits $x_1, \ldots, x_n$ and would like to compute $s = \sum_{i=1}^{n} x_i$. Since the sum can be as large as n, computing it exactly requires $\log_2 n$ bits of space. In this problem we will see a simple algorithm that *approximates* the sum with space complexity proportional to $\log \log n$.

Consider the following algorithm that outputs an estimator of s, denoted $\tilde{s}$:

- Let $X = 0$
- For $i = 1, \ldots, n$: if $x_i = 1$ increment X to $X + 1$ with probability 2^{-X}, otherwise do nothing
- Return $\tilde{s} = 2^X - 1$

First we will analyze the space required to compute $\tilde{s}$. Note that the number of bits required to store X at any given time is $\log_2 X$, so we will try to argue that $\log_2 X$ does not get too big.

2.1 Prove that $\mathbb{P}(\log_2 X \geq k) \leq s/2^{2^k}$ and conclude that the expected number of bits required to store the value X is $O(\log \log s)$.[3]

Next we will show that we can use the estimator $\tilde{s}$ to obtain a good estimate of s.

2.2 Prove by induction that $\mathbb{E}(2^X) = s + 1$, so $\tilde{s}$ is an unbiased estimator of s. You may find it helpful to use the notation X_t to refer to the value of X after seeing the value 1 in the stream t times.

2.3 Prove that $Var(2^X) = O(s^2)$ where Var denotes the *variance*.

2.4 Prove that when U, V are *independent* random variables $Var(U + V) = Var(U) + Var(V)$.

2.5 Suppose we run $\tilde{s}$ in parallel k times on the same inputs *independently* to obtain $\tilde{s}_1, \ldots, \tilde{s}s_k$, and then return the average $\bar{s}_k = \frac{1}{k} \sum_{j=1}^{k} \tilde{s}_j$. Compute the mean and variance of $\bar{s}$.

2.6 Using Chebyshev's inequality, conclude that $\mathbb{P}(|\bar{s}_k - s| > s/100) \leq 1/100$ for some constant k that does not depend on s or n.

Putting these steps together, we have proven that we can estimate s to within ±1% error using just $O(\log \log s)$ space in expectation.

[2] **Hint:** You may freely use the approximation $(1 - 1/t)^u \approx e^{-u/t}$.

[3] **Hint:** You may find the following fact useful: For a non-negative integer random variable $\mathbb{E}(X) = \sum_{i=1}^{\infty} \mathbb{P}(X \geq i)$.