Exercise Set VIII, Algorithms II 2024

These exercises are for your own benefit. Feel free to collaborate and share your answers with other students. Solve as many problems as you can and ask for help if you get stuck for too long. Problems marked * are more difficult but also more fun :).

These problems are taken from various sources at EPFL and on the Internet, too numerous to cite individually.

1 In the previous exercise set you showed that if n balls are placed in n bins using a pairwise independent hash function $h : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$, then the maximum bin load is $O(\sqrt{n})$ with high constant probability. Give an example of a pairwise independent hash family (i.e., a distribution over hash functions) such that the expected maximum bin load is $\Omega(\sqrt{n})$ with high probability.

2 In this problem we shall use the following properties of MinHashing (prove them if you feel good today): Let $X_1, \ldots, X_n$ be independent random variables uniformly distributed in $[0, 1]$ and let $Y = \min\{X_1, \ldots, X_n\}$. Then $\mathbb{E}[Y] = \frac{1}{n+1}$ and $\mathrm{Var}(Y) \leq \frac{1}{(n+1)^2}$.

We now use these properties to analyze a different algorithm than the one explained in class for estimating the number of distinct elements in a sequence. Indeed, consider the following algorithm for estimating F_0, the number of distinct elements in a sequence $x_1, \ldots, x_m \in \{0, 1, \ldots, n-1\}$. Let $h : \{0, 1, \ldots, n-1\} \to [0, 1]$ s.t. $h(i)$ is chosen uniformly and independently at random in $[0, 1]$ for each i. We start with $Y = 1$. After reading each element x_i in the sequence we let $Y = \min\{Y, h(x_i)\}$.

2a Show that by the end of the stream $\frac{1}{\mathbb{E}[Y]} - 1$ is equal to F_0.

2b *(*)* Use the above idea to design a streaming algorithm to estimate the number of distinct elements in the sequence with multiplicative error $1 \pm \epsilon$. For the analysis you can assume that you have access to k independent hash functions as described above. Show that $k \leq O(1/\epsilon^2)$ many such hash functions are enough to estimate the number of distinct elements within factor $1 + \epsilon$ with probability at least $9/10$.

Hint: run the k copies of the above algorithm in parallel. Let Y_i be the Y variable of the i:th copy. Then what is the expected value and variance of the random variable $(Y_1 + Y_2 + \ldots Y_k)/k$? Then apply Chebychev's Inequality.

3 Consider the problem where we wish to find a large cardinality matching in a graph in the semi-streaming model. That is, the edges are streamed one-by-one. The graph has n vertices and we assume that your algorithm has access to storage space $O(n \cdot \operatorname{poly} \log n)$ (that is why it is called semi-streaming and not streaming as we have quite a lot of memory compared to the logarithmic memory seen in the lecture). Notice that although you have quite a lot of storage space, you do not have enough memory to store all (potentially $\Omega(n^2)$ many) edges. Devise an algorithm in this setting that returns a matching that has cardinality at least $1/2$ that of a maximum cardinality matching. (We note that improving the factor $1/2$ is considered a major open problem.)

Hint: think greedy.

4 *(Final exam question from 2017)* **Set packing in the semi-streaming model.**

Consider the problem of finding a maximum cardinality set packing in the semi-streaming model. An instance of this problem consists of a known universe U of n elements and sets $S \subseteq U$ are streamed one-by-one. The goal is to select a family $\mathcal{T}$ of pairwise disjoint sets (i.e., $S \cap S' = \emptyset$ for any two distinct sets $S, S' \in \mathcal{T}$) of maximum cardinality while only using $O(n \cdot \operatorname{poly} \log n)$ storage space.

Devise an algorithm in this setting that returns a set packing of cardinality at least $1/k$ times that of a maximum cardinality set packing, assuming that each streamed set S has cardinality at most k, i.e., $|S| \leq k$.

(In this problem you are asked to (i) design the algorithm, (ii) show that it uses $O(n \cdot \operatorname{polylog} n)$ space, and (iii) prove that it returns a solution of cardinality at least $1/k$ times the cardinality of a maximum cardinality set packing. Recall that you are allowed to refer to material covered in the course.)