CS 473 ✧ Spring 2016

Homework 5

Due Tuesday, March 1, 2016, at 8pm

Unless a problem specifically states otherwise, you may assume a function RANDOM that takes a positive integer k as input and returns an integer chosen uniformly and independently at random from $\{1, 2, \ldots, k\}$ in $O(1)$ time. For example, to flip a fair coin, you could call RANDOM(2).

---

1. ***Reservoir sampling*** is a method for choosing an item uniformly at random from an arbitrarily long stream of data.

```
GETONESAMPLE(stream S):
  ℓ ← 0
  while S is not done
      x ← next item in S
      ℓ ← ℓ + 1
      if RANDOM(ℓ) = 1
          sample ← x        (⋆)
  return sample
```

At the end of the algorithm, the variable ℓ stores the length of the input stream S; this number is *not* known to the algorithm in advance. If S is empty, the output of the algorithm is (correctly!) undefined. In the following, consider an arbitrary non-empty input stream S, and let n denote the (unknown) length of S.

(a) Prove that the item returned by GETONESAMPLE(S) is chosen uniformly at random from S.

(b) Describe and analyze an algorithm that returns a subset of k distinct items chosen uniformly at random from a data stream of length at least k. The integer k is given as part of the input to your algorithm. Prove that your algorithm is correct.

For example, if $k = 2$ and the stream contains the sequence $\langle$♠, ♥, ♦, ♣$\rangle$, the algorithm should return the subset $\{$♦, ♠$\}$ with probability $1/6$.

2. In this problem, we will derive a streaming algorithm that computes an accurate estimate $\widetilde{n}$ of the number of distinct items in a data stream S. Suppose S contains n unique items (but possible several copies of each item); the algorithm does *not* know n in advance. Given an accuracy parameter $0 < \varepsilon < 1$ and a confidence parameter $0 < \delta < 1$ as part of the input, our final algorithm will guarantee that $\Pr[|\widetilde{n} - n| > \varepsilon n] < \delta$.

   As a first step, fix a positive integer m that is large enough that we don't have to worry about round-off errors in the analysis. Our first algorithm chooses a hash function $h\colon \mathcal{U} \to [m]$ at random from a ***2-uniform*** family, computes the minimum hash value $\hbar = \min\{h(x) \mid x \in S\}$, and finally returns the estimate $\widetilde{n} = m/\hbar$.

   (a) Prove that $\Pr[\widetilde{n} > (1+\varepsilon)n] \leq 1/(1+\varepsilon)$. *[Hint: Markov's inequality]*

   (b) Prove that $\Pr[\widetilde{n} < (1-\varepsilon)n] \leq 1-\varepsilon$. *[Hint: Chebyshev's inequality]*

   (c) We can improve this estimator by maintaining the k smallest hash values, for some integer $k > 1$. Let $\widetilde{n}_k = k \cdot m/\hbar_k$, where $\hbar_k$ is the kth smallest element of $\{h(x) \mid x \in S\}$.

   Estimate the smallest value of k (as a function of the accuracy parameter ε) such that $\Pr[|\widetilde{n}_k - n| > \varepsilon n] \leq 1/4$.

   (d) Now suppose we run d copies of the previous estimator in parallel to generate d independent estimates $\widetilde{n}_{k,1}, \widetilde{n}_{k,2}, \ldots, \widetilde{n}_{k,d}$, for some integer $d > 1$. Each copy uses its own independently chosen hash function, but they all use the same value of k that you derived in part (c). Let $\widetilde{N}$ be the *median* of these d estimates.

   Estimate the smallest value of d (as a function of the confidence parameter δ) such that $\Pr[|\widetilde{N} - n| > \varepsilon n] \leq \delta$.