CS 170 HW 14 (Optional)

Due 2019-12-11, at 10:00 pm

1 Study Group

List the names and SIDs of the members in your study group.

2 One-to-One Functions

Suppose that \( H \) is a pairwise independent hash family that maps the elements \( U = \{1, \ldots, n\} \) to \( 1, \ldots, n^3 \). Show that the probability that a randomly chosen hash function \( h \) from \( H \) is one-to-one is at least \( 1 - \frac{1}{n} \).

3 Universal Hashing

Let \( [m] \) denote the set \( \{0, 1, \ldots, m - 1\} \). For each of the following families of hash functions, determine whether or not it is pairwise-independent. If it is universal, determine how many random bits are needed to choose a function from the family.

(a) \( H = \{h_{a_1, a_2} : a_1, a_2 \in [m]\} \), where \( m \) is a fixed prime and
\[
h_{a_1, a_2}(x_1, x_2) = a_1 x_1 + a_2 x_2 \mod m
\]
Notice that each of these functions has signature \( h_{a_1, a_2} : [m]^2 \to [m] \), that is, it maps a pair of integers in \( [m] \) to a single integer in \( [m] \).

(b) \( H \) is as before, except that now \( m = 2^k \) is some fixed power of 2.

(c) \( H \) is the set of all functions \( f : [m] \to [m - 1] \).

4 Count-Median-Sketch

Consider the Count-Min-Sketch algorithm from class for determining heavy hitters in a stream. Suppose that we change the algorithm to consider median_{i=1,...,l}M[i, h_i(x)] instead of the min function.

(a) Give an argument that the estimator for heavy hitters is still correct.

\textit{Note: } We are not looking for a rigorous proof here. Just an argument is fine.

(b) If we allow the stream to both insert and delete elements, show that the median based algorithm works to find heavy hitters. Each item \( x \) in the stream now comes with a an element \( \Delta \in \{+1, -1\} \) indicating whether it is an addition operation or deletion. Note that the number of deletions could be more than the number of additions.

\textit{Note: } We are not looking for a rigorous proof here. Just an argument is fine.

(c) Would Count-Min-Sketch work for the setup above?
5 Document Comparison with Streams

You are given a document $A$ and then a document $B$, both as streams of words. Find a streaming algorithm that returns the degree of similarity between the words in the documents, given by $\frac{|I|}{|U|}$, where $I$ is the set of words that occur in both $A$ and $B$, and $U$ is the set of words that occur in at least one of $A$ and $B$.

Clearly explain your algorithm and briefly justify its correctness and memory usage (at most $\log(|A| + |B|)$). Can we achieve accuracy to an arbitrary degree of precision? That is, given any $\epsilon > 0$ can we guarantee that the solution will always be within a factor of $1 \pm \epsilon$ with high probability?

6 Streaming Integers

In this problem, we assume we are given an infinite stream of integers $x_1, x_2, \ldots$, and have to perform some computation after each new integer is given. Since we may see many integers, we want to limit the amount of memory we have to use in total. For all of the parts below, give a brief description of your algorithm and a brief justification of its correctness.

(a) Show that using only a single bit of memory, we can compute whether the sum of all integers seen so far is even or odd.

(b) Show that we can compute whether the sum of all integers seen so far is divisible by some fixed integer $N$ using $O(\log N)$ bits of memory.

(c) Assume $N$ is prime. Give an algorithm to check if $N$ divides the product of all integers seen so far, using as few bits of memory as possible.

(d) Now let $N$ be an arbitrary integer, and suppose we are given its prime factorization: $N = p_1^{k_1} p_2^{k_2} \ldots p_r^{k_r}$. Give an algorithm to check whether $N$ divides the product of all integers seen so far, using as few bits of memory as possible. Write down the number of bits your algorithm uses in terms of $k_1, \ldots, k_r$. 