CS 170

Hashing and Streaming

Hashing

Refresher

- **Hash function:**

$$h : U \to \{0, ..., m-1\}$$

- **Universal hash family $\mathcal{H}$:**

$$\forall\ k_1 \neq k_2 \in U, \quad \Pr_{h \in \mathcal{H}}[h(k) = h(k')] \leq \frac{1}{m}$$

- **Perfect hashing:**
  use universal hashing in two layer scheme to achieve zero collisions

Hashing 1

Suppose a hash function $h : \{0, 1, \ldots, m-1\} \to \{0, 1, \ldots, m-1\}$ is chosen from a universal hash family. Then

$$\Pr[h(2) = 2 \cdot h(1) \mod m] = \ ?$$

Hashing 1 Solution

Suppose a hash function $h : \{0, 1, \ldots, m-1\} \to \{0, 1, \ldots, m-1\}$ is chosen from a universal hash family. Then

$$\Pr[h(2) = 2 \cdot h(1) \mod m] = \frac{1}{m}$$

Hashing 2

Let $\mathcal{H}$ be the set of all functions
$h : \{0, 1, \ldots, m-1\} \to \{0, 1, \ldots, m-1\}$.

(a) Is $\mathcal{H}$ universal?

(b) How many random bits are needed to sample a function from $\mathcal{H}$?

Hashing 2 Solution

Let $\mathcal{H}$ be the set of all functions
$h : \{0, 1, \dots, m-1\} \to \{0, 1, \dots, m-1\}$.

(a) Is $\mathcal{H}$ universal?
Since $\Pr[h(x) = h(y)] = \sum_{i=0}^{m-1} \frac{1}{m^2} = \frac{1}{m}$ (for $x \neq y$), $\mathcal{H}$ is universal.

(b) How many random bits are needed to sample a function from $\mathcal{H}$?
The size of $\mathcal{H}$ is m^m, so we need $m \log m$ bits to sample a function.

Hashing 3

Given a prime p and $a, b \in \{0, \ldots, p-1\}$, define the function $h_{a,b}(x) = ax + b \bmod p$ where $x \in \{0, \ldots, p-1\}$. **Show that $\mathcal{H} = \{h_{a,b}\}_{a,b \in \{0,\ldots,p-1\}}$ is a pairwise independent hash function family**, i.e. show that for every $x \neq y$ and $c, d \in \{0, \ldots, p-1\}$,

$$\Pr_{h_{a,b} \leftarrow \mathcal{H}} \Big[h_{a,b}(x) = c \wedge h_{a,b}(y) = d \Big] = \frac{1}{p^2} \ .$$

The notation $h_{a,b} \leftarrow \mathcal{H}$ means that $h_{a,b}$ is chosen uniformly at random from $\mathcal{H}$ (in other words, a and b are chosen independently uniformly at random from $\{0, \ldots, p-1\}$).

Hashing 3 Solution

All equations are mod p.

$h(x) = c$ and $h(y) = d$ iff $ax + b = c$ and $ay + b = d$. This is true iff a, b solve the system of equations

$$\begin{aligned} ax + b &= c \\ ay + b &= d \end{aligned}$$

Solving this, we have

$$\begin{aligned} a &= (c - d)(x - y)^{-1} \\ b &= (cy - dx)(y - x)^{-1} \end{aligned}$$

We are guaranteed that the multiplicative inverses exist because p is prime, and we know $x \neq y$. Thus there is only one value of a and one value of b that satisfy these equations. Since a and b are chosen independently at random, the probability of this occurring is $1/p^2$.

Streaming

Refresher

- Have data stream $S = \{x_1, \ldots, x_m\}$ of unknown length
- **Streaming algorithms** process streams $\rightarrow$ give useful information
    - Should be single-pass and use a small amount of space
    - Three components: *initialization*, *processing*, and *output*

Stream Sampling

Given a stream of integers of unknown length, how do you pick one at random while using no more than two integers' worth of storage?

Every data point should have a $1/N$ chance of being selected, where N is the total length of the stream.

Stream Sampling Solution

Given a stream of integers of unknown length, how do you pick one at random while using no more than two integers' worth of storage?

Every data point should have a $1/N$ chance of being selected, where N is the total length of the stream.

Store two integers, x (the result) and n (the length of the stream so far). For each new data point x_i, set x to x_i with probability $1/n$. At the end of the day, return x.

The probability of item i surviving through the nth time step (for $i \leq n$) is

$$\left(\frac{1}{i}\right) \cdot \left(\frac{i}{i+1}\right) \cdot \left(\frac{i+1}{i+2}\right) \cdots \left(\frac{n-2}{n-1}\right) \cdot \left(\frac{n-1}{n}\right) = \frac{1}{n} \ .$$

Document Comparison

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 $\frac{|I|}{|U|}$ between the words in the documents, where I is the set of words that occur in both A and B, and U is the set of words that occur in either A or B.

Clearly explain your algorithm and briefly justify its correctness and memory usage (at most $O(\log(|A| + |B|))$. How accurate is its output?

Document Comparison Solution

Simply use the F_0 streaming algorithm on A, B, and $C := A \cup B$ (for this third stream, process words in A and words in B). Let $|A|, |B|$, and $|C|$ be the output of the algorithm on the corresponding set (our estimate for the number of distinct words in it). Then our estimate for $|I|/|U|$ is $(|A| + |B| - |C|)/|C|$.

Memory usage is logarithmic in the length of the documents, as we're using a constant number (three) copies of the streaming algorithm shown in class, which used logarithmic memory. Correctness flows from the correctness of the streaming algorithm for F_0, combined with the set theory axiom $|A \cap B| = |A| + |B| - |A \cup B|$.

The accuracy of the underlying F_0 algorithm (and by extension our algorithm) can be boosted to any level we desire.