Distinct Elements in Vanilla Streaming Model

Fact: Let $F_0 = 1 \forall j: f_j > 0 \exists I$. Given $(\varepsilon, \delta)$-approximation algorithm for $F_0$.

Preliminaries: Pairwise independent family of hash functions.

A family $H$ of hash functions is said to be pairwise independent if for any $x, y \in [n]$ and $v, w \in \mathbb{C}$,

$$\Pr_{h \in H}[h(x) = v \land h(y) = w] = \prod_{i=1}^{d} \frac{1}{\mathbb{C}} = \frac{1}{\mathbb{C}^d}$$

Note: implies $h(x)$ uniformly distributed in $\mathbb{C}$, for any $x$. Example: Assume $p$ is prime. For any $a, b \in \mathbb{F}$, let $h_{a,b}(x) = ax + b \mod p$.

Claim: $H$ is a pairwise independent hash family.

Proof: Fix $x, y \in [n], v, w \in \mathbb{F}$. What is $\Pr_{h \in H}[h(x) = v \land h(y) = w]$?

It is $\left| \{ (a, b) : h_{a,b}(x) = v \land h_{a,b}(y) = w \} \right| = \frac{1}{|\mathbb{F}|^2}$

$ax + b = v \mod p$ two linearly independent equations in 2 unknownt two linearly independent equations in 2 unknowns have unique solution.

Note: $h$ has only $O(\log n)$ bits to store, and it can be evaluated with $O(1)$ multiplications and additions, mod $p$.

Fun fact: if $p = 2^{61} - 1$ or $p = 2^{127} - 1$, then reducing mod $p$ is just a bit-wise shift and $\&$ operation, and multiplication mod $p$ can be done with $O(1)$ machine multiplications.

Outline for rest of lecture

- Recall template for designing $(\varepsilon, \delta)$-multiplicative approximation algorithm for
  - First, develop "basic estimator" that is unbiased but might have high variance (say variance $v$)
    - Take the average of many copies of the basic estimator to decrease variance to $O(\varepsilon)$
  - Apply Chernoff to conclude the above is a $(1+\varepsilon)$-approx. with $p \geq \frac{1}{2}$.
  - Take median of $O(\log(4))$ calls of the above to get a $(1+\varepsilon)$-approx. with high probability.
E.g., last time the basis estimator for $F_\kappa$ had variance at most $\nu \leq \kappa \cdot n^{1-\frac{1}{k}} \cdot F_\kappa^2$. So, we took $t = \frac{\nu}{\epsilon^2 F_\kappa} = \frac{\kappa \cdot n^{1-\frac{1}{k}}}{\epsilon^2}$ mutually independent copies of the basis estimator and looked at their average. We then took the median of $O(\log(\frac{1}{\delta}))$ copies of the above, for total space usage $O\left(\frac{\kappa \cdot n^{1-\frac{1}{k}} \cdot \log(\frac{1}{\delta}) \cdot \log(n \cdot m)}{\epsilon^2}\right)$. 
Today, we will give a basic estimator that is *not* unbiased.

We will not bound its variance so we won’t get a *tail-mult.

approx.

We will use Markov’s inequality to show it *does* satisfy

\[
\frac{F_0}{10} \leq \tilde{F}_0 \leq 10F_0 \quad \text{with probability at least } \frac{3}{4}
\]

Taking the median of \( \Omega(\log(\frac{1}{\delta})) \) copies then gives an estimator \( \tilde{F}_0 \) s.t.

\[
\frac{F_0}{10} \leq \tilde{F}_0 \leq 10F_0 \quad \text{with probability at least } 1-\delta.
\]

Tomorrow we will see the *actual* \( (k,\delta) \)-mult. approx.

an actual \((k,\delta)\)-mult. approx. alg.

Some for \( F_0 \) following the template

\[
\text{sort of}
\]
A Simple (Suboptimal) Distinct Element Algorithm.

Given a number \( n \), let \( \text{zeros}(n) \) denote the number of 'trailing zeros' in the binary representation of \( n \). This is the same as the largest \( i \) such that \( 2^i \) divides \( n \).

Assume \( 2^n \) is a power of two.

Algorithm: * Choose a random \( h \) from a pairwise independent hash family \( H \) of hash functions mapping \( [n] \) to \( [r] \).

* \( Z \leftarrow 0 \)

* For each stream update \( a_i \):

  - If \( \text{zeros}(h(a_i)) > Z \), then \( Z \leftarrow \text{zeros}(h(a_i)) \)

Output \( Z \).

i.e. the algorithm simply outputs the largest number of trailing zeros ever seen in a hash value.

Note: Space usage is \( O(\log_2 n) \) bits for storing \( Z \), and \( O(\log_2 n) \) bits for storing \( h \).

Note also that the algorithm implicitly ignores duplicates and is independent of the order of the stream.

Intuition: If there are \( d \) distinct items, then we expect about \( \frac{d}{2^i} \) of them to have at least \( i \) trailing zeros. When \( i = \log_2 d \), the number \( \frac{d}{2^i} \) is very close to 0, so \( Z \) should be fairly close to \( 2^{\log_2 d} = \sqrt{d} \).
Claim: With probability at least $1 - 2^{-n/8}$, the algorithm will output a number $\hat{d}$ such that $\frac{d}{4} \leq \hat{d} \leq 4d$.

Proof: For each $j \in [n]$ and $t \in \mathbb{Z}_+$, let $X_{t,j} = \sum_{0 \leq t \leq d | h(j)} 2^t$. Define $Y_t = \sum_{j : f_j > 0} X_{t,j}$. Let $\hat{Z}$ be the final value of $Z$ computed by the algorithm. Note:

$$Y_t > 0 \iff \hat{Z} > 0 \iff 2^t > 0 \iff \hat{Z} < 0 \iff 2^t = 0$$

Fact 1: $\mathbb{E}[Y_t] = \sum_{j : f_j > 0} \mathbb{E}[X_{t,j}] = \sum_{j : f_j > 0} \mathbb{P}[2^t \text{ divides } h(j)] = \sum_{j : f_j > 0} \frac{1}{d_{2^t}} \frac{1}{d_{2^t}}$

Fact 2: $\mathbb{V}[Y_t] \leq \frac{d}{2t},$ in particular, independent for $t < f_j$.

Proof: $\mathbb{V}[Y_t] = \mathbb{V}\left[\sum_{j : f_j > 0} X_{t,j}\right] = \sum_{j : f_j > 0} \mathbb{V}[X_{t,j}]

\leq \sum_{j : f_j > 0} \mathbb{E}[X_{t,j}^2] \leq \sum_{j : f_j > 0} \mathbb{E}[X_{t,j}] = \frac{d}{2t}$. 

Let \( a = \log_2(8d) \).

\[
\Pr[Y_a = 1] \leq \frac{E[Y_a]}{d} = \frac{d}{e^d} \leq \frac{1}{2^n}
\]

Let \( b = \lceil \log_2 \left( \frac{d}{8} \right) \rceil \).

\[
\Pr[Y_b = 0] \leq \Pr[|Y_b - E[Y_b]| \geq \frac{d}{2b}]
\]

Claim:

\[
\frac{1}{2^n} \leq \frac{d}{2b} \leq \frac{d}{e^d} \leq \frac{1}{2^n}
\]

To amplify the probability of a bad estimate from \( \frac{3}{8} \) to \( \frac{1}{2} \), repeat \( O(\log \frac{1}{\delta}) \) times and take the median.

Next time, rather than an estimate \( \hat{d} \), let \( \frac{d}{8} \leq \hat{d} \leq 8d \) with probability \( 1-\delta \), we'll see an actual \( \hat{d} \) approximation algorithm using space about \( O\left(2^n + \frac{1}{\epsilon^2}\right) \) optimal or we'll wish for

The rough idea will be to run about \( \frac{1}{\epsilon^2} \) copies of the simple algorithm in parallel, and aggregate their estimates into a single, smarter estimate in a clever way.