In the last lecture, we looked at an algorithm for approximating the number of elements in a stream.

**Algorithm:**
Let \( a \) be a stream of \( d \) elements.
For any integer \( p \geq 0 \), let \( \text{zeroes}(p) \) be the maximum element in the set \( \{ i | 2^i \text{ divides } p \} \)

1: Pick a random hash function \( h : [n] \to [n] \) from a 2-universal family.
2: \( Z \leftarrow 0 \)
3: for each \( a_i \) in the stream do
4: if \( \text{zeroes}(h(a_i)) > Z \) then
5: \( Z \leftarrow \text{zeroes}(h(a_i)) \)
6: return \( 2^{Z+1/2} \)

**Analysis:**
For the analysis we’ll need to introduce two types of random variables, \( X_{r,j} \) and \( Y_r \).

\[
X_{r,j} = 1 \text{ if } \text{zeroes}(h(j)) \geq r \\
X_{r,j} = 0 \text{ otherwise}
\]

The only randomness for \( X_{r,j} \) comes from the choice of hash function \( h : [n] \to [n] \)
\( x_{r,j} \) is a random variable with respect to that space.

\[
Y_r = \sum_{j : f_j > 0} X_{r,j}
\]

where \( f_j \) is the frequency of \( j \).

**Example:**
Let our stream be \( a = \{17, 2, 3, 17, 2, 5, 7, 5\} \)

<table>
<thead>
<tr>
<th>( j )</th>
<th>( \text{zeroes}(h(j)) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
</tr>
</tbody>
</table>
So $Y_0 = 5$ because $\text{zeroes}(h(j)) \geq 0$ for all 5 elements, similarly

\[
\begin{align*}
Y_1 &= 4 \\
Y_2 &= 2 \\
Y_3 &= 1 \\
Y_4 &= 0 \\
Y_5 &= 0 \\
&\ldots \\
Y_{r>3} &= 0
\end{align*}
\]

**Claim**

Let $t$ denote the value of $Z$ at the end of the execution of the algorithm.

\[
Y_r > 0 \iff t \geq r \\
Y_r = 0 \iff t \leq r - 1
\]

We want to find $E[Y_r]$ for some fixed $r$. 

\[
E[Y_r] = \sum_{j : f_j > 0} E[X_{r,j}]
\]

\[
= \sum_j Pr[X_{r,j} = 1]
\]

\[
= \sum_j Pr[2^r \text{ divides } h(j)]
\]

\[
= \sum_j \frac{1}{2r}
\]

\[
= \frac{d}{2r}
\]

One way we can think of this is that every element will contribute to $Y_0$, an element will contribute to $Y_1$ with a probability of $\frac{1}{2}$, an element will contribute to $Y_2$ with a probability of $\frac{1}{4}$, etc. That is,

\[
Pr[X_{0,j} = 1] = 1
\]

\[
Pr[X_{1,j} = 1] = \frac{1}{2}
\]

\[
Pr[X_{2,j} = 1] = \frac{1}{4}
\]

etc.
Because of this $2^r \cdot Y_r$ is a good estimator for $d$.
Assuming any two variables are independent,

$$Var[Y_r] = \sum_j Var[X_{r,j}]$$
$$\leq \sum_j E[(X_{r,j})^2] \quad (Because \ Var(z) = E(z^2) - E(z)^2)$$
$$= \sum_j E[X_{r,j}] \quad (Because \ X_{r,j} \ is \ a \ 01 \ random \ variable.)$$
$$= \frac{d}{2^r}$$

$$Pr[Y_r > 0] = Pr[Y_r \geq 1]$$
$$\leq E[Y_r]$$
$$= \frac{d}{2^r}$$

$$Pr[Y_r = 0] \leq Pr[|Y_r - E[Y_r]| \geq \frac{d}{2^r}]$$
$$\leq \frac{Var[Y_r]}{(d/2^r)^2} \quad (By \ Chebyshev's \ inequality)$$
$$\leq \frac{2^r}{d}$$

So the transition from $Y_r$ going from 0 to nonzero happens around $r = \log(d)$

We now want to show why we output $2^{t+1/2}$ instead of $2^t$

Let $\hat{d} = 2^{t+1/2}$ (estimate of $d$ output by algorithm)

Let $a$ be the smallest integer such that $2^{a+1/2} \geq 3d$

$$Pr[\hat{d} \geq 3d] = Pr[t \geq a]$$
$$= Pr[Y_a > 0]$$
$$\leq \frac{d}{2^r}$$
$$\leq \frac{\sqrt{2}}{3}$$

Let $b$ be the largest integer such that $2^{b+1/2} \leq \frac{d}{3}$
\[
Pr[\hat{d} \leq \frac{d}{3}] = Pr[t \leq b] = Pr[Y_{b+1} = 0] \leq \frac{2^{b+1}}{d} = \frac{2^{b+1/2}}{d} \cdot \sqrt{2} \leq \frac{\sqrt{2}}{3}
\]

So returning \(2^{t+1/2}\) instead of \(2^t\) allows us to get a slightly tighter bound. (3d rather than somewhere around 4d-5d)

When running the algorithm we’ll get an estimate within the bounds \(\frac{d}{3} \leq \hat{d} \leq 3d\) with strictly more than 50% probability. To increase this probability to 1-\(\delta\) we must run \(\log(\frac{1}{\delta})\) independent instances of the algorithm and return the median of the estimates.

**Definition of 2-Universal**

Let \(X\) and \(Y\) be finite sets.

Let \(Y^X\) be the set of all functions from \(X\) to \(Y\).

\(\mathcal{H} \subseteq Y^X\) is said to be 2-universal if for all \(x, x' \in X (x \neq x')\) and \(y, y' \in Y\)

\[
Pr[h(x) = y \land h(x') = y'] = \frac{1}{|Y|^2}
\]

\[
Pr[h(x) = y] = \frac{1}{|Y|}
\]

\[
Pr[h(x') = y'] = \frac{1}{|Y|}
\]

**Choosing a Hash Function**

Now we’ll look at how we can pick the random hash function \(h : [n] \to [n]\).

Each \(j \in [n]\) can be represented as a length \(t\) 0-1 vector. So if \(t = 4\), \(j\) might be

\[
\begin{bmatrix}
1 \\
0 \\
1 \\
1 \\
\end{bmatrix}
\]

One choice of hash function might be a \(h(x) = Ax + b\) where \(A\) is a \(t \times t\) matrix and \(b\) is a length \(t\) vector.
The hash function $h$ is fixed if you know $A$ and $b$. You can randomly select $A$ and $b$ by randomly selecting each element of $A$ and $b$ to be 0 or 1 with equal probability.

It takes $log^2 n$ bits to remember this hash function.

A family of hash functions can be created by taking every possible combination of $A$ and $b$. We can then select one function from this family at random for our algorithm.

**Homework Problem:** (Source: Problem 2-1, Lecture 2, Amit Chakrabarh)

Treat the elements of $X$ and $Y$ as column vectors with 0/1 entries. For a matrix $A \in \{0,1\}^{k \times n}$ and vector $b \in \{0,1\}^k$, define the function $h_{A,b} : X \rightarrow Y$ by $h_{A,b}(x) = Ax + b$, where all additions and multiplications are performed mod2.

Prove that the family of functions $\mathcal{H} = \{h_{A,b} : A \in \{0,1\}^{k \times n}, b \in \{0,1\}^k\}$ is 2-universal.

**Another Streaming Problem: Finding Frequent Elements**

Let the stream be $\sigma = <a_1, a_2, ..., a_m>$ where each $a_i \in [n]$.

In practice stream elements can be any type of object. We assume that we can hash any of these objects to an integer for the purposes of our algorithm.

We define $f = (f_0, f_1, ..., f_{n-1})$ where $f_i$ is the frequency of $i$ in the stream for some $i$.

Given $\epsilon > 0$, we want to identify all $j$ such that $f_j \geq \epsilon \cdot m$.

**The Misra-Gries Algorithm**

First we’ll give a deterministic algorithm for finding an estimate $\hat{f}_a$ of the frequency $f_a$ for some $a$.

We’ll maintain a dictionary $A$ where the keys of $A = [n]$.

For a key $j$, $A[j]$ is an estimate for $f_j$.

We don’t want to maintain a dictionary with all $n$ keys so we’ll restrict ourselves to some $k$ keys.
1: Initialize empty dictionary $A$
2: Pick $k$
3: if $a_i \in \text{keys}(A)$ then
   4: $A[a_i] \leftarrow A[a_i] + 1$
5: else if $|\text{keys}(A)| < k - 1$ then
   6: $A[a_i] \leftarrow 1$
7: else
   8: for each $\ell \in \text{keys}(A)$ do
      9: $A[\ell] \leftarrow A[\ell] - 1$
   10: if $A[\ell] = 0$ then
      11: Remove $\ell$ from $A$
12: return On query $a$ if $a \in \text{keys}(A)$ report $\hat{f}_a = A[a]$ else $\hat{f}_a = 0$

Claim: For each $j \in [n]$

$$f_j - \frac{m}{k} \leq \hat{f}_j \leq f_j$$

where $d$ is the number of unique elements in the stream.

Let $\alpha$ be the number of times we subtract 1 from the estimated frequency of $j$. Each time we subtract 1 from the estimated frequency of $j$ we subtract 1 from the estimate of $k - 1$ other elements. Thus

$$\alpha \cdot k \leq m$$

As a consequence of this,

If $k = \frac{2}{\epsilon}$ then

$$f_j - \frac{\epsilon \cdot m}{2} \leq \hat{f}_j \leq f_j$$

If $f_j \geq \epsilon \cdot m$ then

$$\hat{f}_j \geq \frac{f_j}{2} \geq \frac{\epsilon \cdot m}{2}$$
**Turnstile Model**

Let $\sigma = \langle a_1, a_2, ..., a_m \rangle$ be our stream.

Each $a_i$ is a pair $(j, c)$ where $j \in [n]$ and $c$ is an integer. (positive or negative)

An element $f_i$ of the frequency vector $f$ is the sum of all $c$’s in each pair $(j, c)$ in $\sigma$ for which $j = i$.

This "turnstile model" is a generalized version of the previous model. In the previous model $c$ is always 1.

We want to find the highest $f_i$ in $f$. For now we’ll assume that all elements of the frequency vector $f$ will always be non-negative.

---

1: $C[1..k] \leftarrow [0, 0, ..., 0]$
2: Choose a random hash function $h : [n] \rightarrow [k]$
3: Choose a random hash function $g : [n] \rightarrow \{-1, +1\}$
4: for each $a_i = (j, c) \in \sigma$ do
5: \quad $C[h(j)] \leftarrow C[h(j)] + c \cdot g(j)$
6: \textbf{return} On query $a$ report $\hat{f}_a = g(a) \cdot C[h(a)]$

In the analysis of this algorithm we’ll want to show $E[\hat{f}_a] = f_a$