Approximating Frequency Counts over Streams.

We are in the same situation as last Thursday — we are given a sequence of elements \( x = (x_1, \ldots, x_n) \) where each \( x_i \in \Sigma = \{1, 2, \ldots, n\} \).

We are also given a \( \Theta \) such that \( 0 < \Theta \leq 1 \). We wish to keep track of

\[
I(x, \Theta) = \sum_{a \in \Sigma} \min(a, m(a, x, \Theta)),
\]

where \( m(a, x, \Theta) \) is the number of times \( a \) occurs in the stream \( x \).

Recall also that \( x \) is a data stream — we get to make only one pass over it, and we wish to optimize space.
The solution that we gave last time maintained a set $K$ of at most $\frac{1}{\epsilon^2}$ elements that is guaranteed to contain $\mathbb{I}(x, 0)$. But $K$ may contain other, very infrequently occurring elements as well.

This is the issue we'd like to address today. Suppose we are given also given an $\epsilon$ such that $0<\epsilon<1$. We'll describe an algorithm that maintains a set $D$ of at most $\frac{1}{\epsilon}$ log $\epsilon$ $N$ elements and:
1. Outputs all elements with frequency \( \geq 0.1N \).

2. Outputs no elements with frequency less than \((0 - \epsilon)N\).

3. For all elements in \( D \), estimated frequencies. \( \epsilon \)

Suppose we set \( \epsilon = 0.001 \). One natural choice for \( \epsilon = 0.0001 \).

Then all elements with frequency at least 0.001 \( N \) are output. No element with frequency less than 0.0009 is output.
The algorithm: let \( w = \lceil \frac{1}{e} \rceil \). Conceptually divide stream into buckets of length \( w \). First \( w \) elements are in bucket 1, next \( w \) elements are in bucket 2, and so on. Let \( \text{bccreen} \) denote current bucket – note \( \text{bccreen} = \lceil \frac{N}{w} \rceil \).

Example: \( e = 0.2 \), \( w = 5 \)

\[
\begin{array}{cccccc}
\text{bucket 1} & & & & & \\
12 & 17 & 12 & 11 & 10 & \text{bucket 2} \\
& 1 & 7 & 11 & 18 & 18 \\
\text{bucket 3} & & & & &
\end{array}
\]

\[
\begin{array}{cccccc}
23 & 11 & 10 & 18 & 5 & \\
& & & & & \\
\end{array}
\]

Each element \( e \) stored in \( D \) is stored as a triple \((e, f, \Delta)\), where \( f \) is estimated frequency of \( e \) in stream thus far, \( \Delta \) is maximum possible error in \( f \).
When a new element \( e \) arrives, we check if \( e \) is present in \( D \). If so, we change triple \((e, f, \Delta)\) to \((e, f+1, \Delta)\) — that is, we increment \( f \) by 1. If \( e \) is not present in \( D \), we insert the triple \((e, 1, \text{count} - 1)\).

**Cleanup:** At end of each bucket, we go through each triple \((e, f, \Delta)\) in \( D \). If \( f + \Delta \leq \text{count} \), we delete this triple from \( D \).

---

Run Algorithm on Example.
As for the output, the algorithm has an estimated frequency for each element it stores. It outputs all elements whose estimated frequency is at least $(\Theta - \epsilon)N$.

Let us begin understanding why the algorithm works by showing that if an element gets deleted, its current frequency is quite low. Let $f_e$ denote the frequency of $e$ in stream thus far.

**Lemma 1**: Whenever an entry $(e, f, \Delta)$ gets deleted, $f_e \leq b \text{current}$.

**Proof**: By induction on $b \text{current}$. Base case $b \text{current} = 1$ is easily checked.
Now suppose \((e, f, \Delta)\) gets deleted when \(\text{bucket} > 1\). This entry was inserted in bucket \(\Delta + 1 \leq \text{bucket}\). Using induction hypothesis (and some additional reasoning), we can conclude that actual frequency of \(e\) in first \(\Delta\) buckets is \(\leq \Delta\). \(f\) is the true frequency of \(e\) from bucket \(\Delta + 1\) onwards.

Thus, \(f_e \leq \Delta + f\).

Now \((e, f, \Delta)\) is deleted because \(\Delta + f \leq \text{bucket}\).

We conclude that \(f_e \leq \text{bucket}\).
Lemma 2: If $e$ does not appear in $D$, then $f_e \leq eN$.

Proof: Clearly lemma is true if $e$ does not occur in stream at all. Otherwise, consider last time it was deleted.

At that time, $f_e \leq \text{current } N = \frac{N}{w}$

$$\frac{N}{T_e^t} \leq eN.$$ 

So $f_e \leq eN$ must hold now, since $f_e$ has not increased since that time and $N$ has not decreased.

Lemma 3: If $(e, f, \Delta) \in D$, then $f \leq f_e \leq f + eN$. 
Proof: First inequality is obvious.

For second, we can reason as in Lemma 1 to claim that

\[ f_e \leq f + \Delta. \]

Since \( \Delta \leq \text{bound} - 1 \),

we have \( f_e \leq f + \text{bound} - 1 \)

\[ \leq f + cN. \]

Correctness: By Lemma 2, any \( e \) with \( f_e \geq cN \) is in D. Since \( 0 > c \), any \( e \) with \( f_e \geq 0 \) is in D. We output every \( e \) in D such that \( f_e \geq (0 - e)N \). By Lemma 3, \( f \geq (0 - e)N \) for every \( e \in D \) such that \( f_e \geq 0 \) \( N \). So we output elements we want.
On the other hand, by Lemma 3 again, \( f \geq (\Omega - \epsilon)N \) for every \( \epsilon > 0 \) that we output.

Finally, we come to the space used by data structure, that is, the no. of elements in \( D \).

Lemma \# of entries in \( D \) is at most \( \lceil \frac{1}{\epsilon} \rceil \log (eN) + \lceil \frac{1}{\epsilon} \rceil \)

Proof: Let's prove a bound of \( \lceil \frac{1}{\epsilon} \rceil \log (eN) \) when we are at the end of a bucket. The additive \( \lceil \frac{1}{\epsilon} \rceil \) takes care of the incomplete bucket.
Let $B$ denote bucket. For each $i \in \{1, 2, \ldots, B\}$ let $d_i$ denote no. of entries now in $D$ that got inserted in bucket $B_{i+1}$.

Each of the $d_i$ entries must occur at least $i$ times in buckets $B_{i+1}$ to $B$. So we have

Otherwise, such an entry wouldn't be in $D$. So

But buckets $B_{i+1} \ldots B$ have at most $iw$ elements in them.

So we have:

$$d_i \leq w$$

$$2d_2 + d_1 \leq 2w$$

$$3d_3 + 2d_2 + d_1 \leq 3w$$
\[ B d_B + (B-1) d_{B-1} + \ldots + 2 d_2 + d_1 \leq B w. \]

The maximum value of

\[ d_1 + d_2 + \ldots + d_B \]

subject to these constraints is

\[ \omega \left( 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{B} \right). \]

This can be attained by setting

\[ d_1 = \omega \]
\[ d_2 = \frac{\omega}{2} \]
\[ d_3 = \frac{\omega}{3} \]
\[ \vdots \]
\[ d_B = \frac{\omega}{B}. \]
The proof that maximum value of \( d_1 + \ldots + d_B \) is \( \omega \left( 1 + \frac{1}{2} + \ldots + \frac{1}{B} \right) \) is omitted. If you are interested, read the paper (Thm 4.2).

So \( |D| = d_1 + \ldots + d_B \)

\[ \leq \omega \left( 1 + \frac{1}{2} + \ldots + \frac{1}{B} \right) \]

\[ \leq \frac{1}{e} \log B \]

\[ \leq \frac{1}{e} \log e \in N. \]