$\boxed{\text{18 Apr 2022}}$ Streaming Algorithms

Input is a sequence of tokens

$a_1, a_2, \ldots, a_n$

Each token represented by b bits.

$a_i \in \{0,1\}^b$.

Number of potential tokens is $m = 2^b$.

Algorithm has storage space s,
typically assumed $s = O(\text{poly}(\log n, \log m))$.

The algorithm observes the tokens in a
<u>single</u> <u>pass</u>, i.e. its main loop is
allowed to:

- observe a_i
- do some ~~update~~ update to internal state
- move on to a_{i+1}

... but cannot observe a_i again
after moving on.

<u>EXAMPLE</u>. Finding frequent elements

In this example $s = O(k(\log n + \log m))$
for some small k, e.g. $k = O(1)$.

The algorithm outputs a set of $\leq k$ tokens
including every token that appears

more than $\frac{n}{k+1}$ times in the stream.

MISRA-GRIES ALGORITHM

Initialize $\overbrace{(b_1, \ldots, b_k)}^{\text{tokens}} = \overbrace{(\perp, \perp, \ldots, \perp)}^{\text{null value}}$

$(c_1, \ldots, c_k) = (0, 0, \ldots, 0)$

// $\{b_1, \ldots, b_k\}$ includes all tokens that have appeared more than $\frac{i}{k+1}$ times among $a_1, \ldots, a_i$

for $i = 1, \ldots, n$:

observe a_i

if $a_i = b_j$ for some $j \in [k]$

$c_j \leftarrow c_j + 1$ (I)

else if $c_j = 0$ for some $j \in [k]$

$b_j \leftarrow a_i$

$c_j \leftarrow 1$ (II)

else // $a_j \notin \{b_1, \ldots, b_k\}$, $c_j > 0$ $\forall j$

$c_j \leftarrow c_j - 1$ for all $j \in [k]$ (III)

endfor

output $\{b_1, \ldots, b_k\}$.

$\boxed{k=2}$

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
$\emptyset$	1	$\emptyset$	2	1	$\emptyset$	3	2	$\emptyset$

Why correct?

<u>Invariant:</u> At all times, $\{b_1, \ldots, b_k\}$ includes every token with an unerased red mark and c_j = # of unerased red marks on b_j.

To show correctness: argue if b is some token occurring $> \frac{n}{k+1}$ times in the stream, then there is at least one unerased red mark on a copy of b at the end of last loop iteration.

This is true because the # of red marks on b increases (by 1) each time b occurs in the stream, and decreases (by ≤ 1) each time line (III) is executed.

The former event happens $> \frac{n}{k+1}$ times.
The latter event happens $\leq \frac{n}{k+1}$ times.

Counting Distinct Elements/Tokens

Given $a_1, \dots, a_n$ how many distinct tokens does it contain?

If $s < m$ and $n \geq m$, no deterministic algorithm can succeed at this task for all possible inputs.

Consider the internal memory state after processing $a_1, \dots, a_{n-1}$.
Since memory is $s < m$ bits, and there are $2^m - 1$ possible sets of distinct tokens occurring among $a_1, \dots, a_{n-1}$

$$2^m - 1 > 2^s \xrightarrow{\text{pigeonhole}} \exists \text{ sets } S_0 \neq S_1$$

and streams $a_1, \dots, a_{n-1}$

$a'_1, \dots, a'_{n-1}$

s.t. $S_0 = \{\text{tokens occurring in } a_1, \dots, a_{n-1}\}$

$S_1 = \{\text{tokens} \quad - \quad - \quad a'_1, \dots, a'_{n-1}\}$

but memory state ~~of~~ alg is the same after seeing $a_1, \dots, a_{n-1}$ or $a'_1, \dots, a'_{n-1}$.

There is some token $b \in S_0, b \notin S_1$
or $b \notin S_0, b \in S_1$.

If $a_n = a'_n = b$, the algorithm is certain to output wrong answer on at least one of $a_1, \dots, a_n$ or $a'_1, \dots, a'_n$.

Algorithm outputs the same answer on

$a_1, \ldots, a_n$ as $a_1', \ldots, a_n'$

but, by construction, the # of distinct elements is different in the 2 cases.

<u>Conclusion</u>: Streaming algs for counting distinct elements must be randomized and have some (maybe small) probability of failure.