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 single pass, i.e., its main loop is

- observe \( a_i \);
- do some update to internal state
- move on to \( a_{i+1} \)

... but cannot observe \( a_i \) again after moving on.

Example: 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 \( k \) tokens including every token that appears
more than $\frac{n}{k+1}$ times in the stream.

**MISRA-GRIES ALGORITHM**

Initialize $(b_1, \ldots, b_k) = (\bot, \bot, \ldots, \bot)$

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

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

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

observe $a_i$

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

$c_j <\!\!< c_j + 1$ (I)

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

$b_j \leftarrow a_i$  
$c_j \leftarrow 1$  (II)

else // $a_j \notin \{b_1, \ldots, b_k\}$, $c_j > 0$ for all $j \in [k]$

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

endfor

output $\{b_1, \ldots, b_k\}$. 
Why correct?

**Invariant:** At all times, \( \{ b_1, \ldots, b_k \} \)
includes every token with an unerased red mark and

\[ c_j = \# \text{ of unerased red marks on } b_j. \]

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 \( < \frac{n}{k+1} \) times.
**Counting Distinct Elements/ Tokens**

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

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

Consider the internal memory state after processing $a_1, \ldots, a_{n-1}$.

Since memory is $s < m$ bits, and there are $2^m - 1$ possible sets of distinct tokens occurring among $a_1, \ldots, a_{n-1}$,

$$2^m - 1 > 2^s \implies \exists \text{sets } S_0, \ldots, S_r,$$

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

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

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

$S_1$ = tokens $\ldots, a'_1, \ldots, a'_{n-1}$

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

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 $a_1, \ldots, a_n$ or $a'_1, \ldots, a'_n$. 
Algorithm outputs the same answer on
\( a_1, \ldots, a_n \) as \( a_1', \ldots, a_n' \)
but, by construction, the number of distinct elements is different in the 2 cases.

\underline{Conclusion}: Streaming algs for counting distinct elements must be randomized and have some (maybe small) probability of failure.