Streaming and Sketching

What kind of useful computation can be done in small space

Model :

sequence of 'tokens'

$a_1, a_2, \dots, a_n$ (stream)

Each $a_i \in \{0,1\}^b$ — b bit string

Trivial space requirement : $n \cdot b$

Space allowed for 'algorithm' : $S \sim o(n)$, linear dependence on b

e.g., $\sqrt{n}$, $2^{\sqrt{\log n}}$, $\text{poly}(\log n)$, $\log n$

Objectives:
1. Most frequent element ✓
2. # distinct elements
3. Frequency moments

→ Goal : Find frequent elements in data stream

Misra-Gries Algorithm

- Fix parameter K. Output all elements that appear at least $\frac{n}{K+1}$ times (possibly with

elements that are less frequent)

Algorithm:

(1) k tokens: $b_1, b_2, \dots, b_k$, all initialized to $\perp$

k numbers: $c_1, c_2, \dots, c_k$, all initialized to $\perp$.

(2) When a_i appears:

- if one of the b_j's is equal to a_i: increase c_j by 1
- if some b_j is $\perp$, set $b_j = a_i$ and $c_j = 1$.
- none of b_j's are a_i (or $\perp$): decrease c_j by 1 for all $j = 1, \dots, k$.

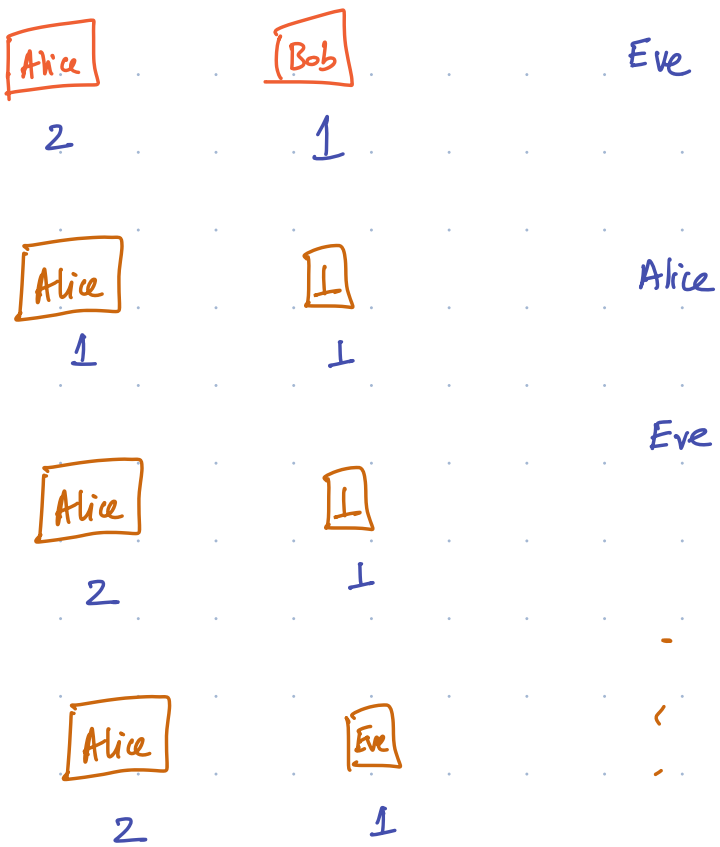
(if $c_j = 0$, then set $b_j = \perp$)

(3) Output: $b_1, \dots, b_k$.

- Space requirement: $S = \underbrace{k \cdot b}_{b_1, \dots, b_k} + \underbrace{k \lceil \log n \rceil}$

Tokens | Counters

Example:
$K = 2$



Proof of Correctness:

On seeing a_i, mark it as 'red'.

a_i → is one of stored b_j's
→ or There b_j which is $\perp$
} mark on a_i remains

→ a_i is not one of b_j's,
all b_i's are diff. from $\perp$
}

(1) remove mark from a_i

(2) remove " " each b_j : from the first marked copy of b_j.

Invariant: at any point, there are exactly c_j marked copies of b_j ($\forall j = 1, \dots, K$)

Proof is by induction on i. $[i = 0, \dots, n]$

$i = 0$ ✓.

Follows at step i directly from algorithm.

Now, note that at any 'removal step', exactly $K+1$ marks are removed.

$\Rightarrow$ at most $\frac{n}{K+1}$ 'removal steps'.

$\Rightarrow$ if a token appears $> \frac{n}{k+1}$ times, then some marked copy remains.

□

Data Sketching

streaming + data structures

tokens: $a_1, a_2, \ldots, a_n$, each $a_i \in \{0,1\}^b$

space: S

↳ a data structure to answer queries.

→ Query: frequency of x? f_x

Trivial bounds: → $S = b \cdot n$ → store stream

→ $S = 2^b \cdot \log n$ → store frequency vector.

$\vec{f}$: 2^b coordinates indexed by $x \in \{0,1\}^b$.

x^{th} coordinate: f_x.

$S \sim \text{poly}(\log n, b)$.

Algorithm: that w.p $1-\delta$ outputs $\tilde{f}_x$ s.t $|\tilde{f}_x - f_x| \leq \varepsilon \cdot n$.

Idea: 'store hash of frequency vector'

→ Sample uniformly a 2-universal hash function $h: \{0,1\}^b \to [B]$. (B to be chosen later)

→ Let C be an array of length B, all entries initialized to 0.

→ for $i = 1$ to n:

on seeing a_i:

let $k = h(a_i)$.

$C[k] \leftarrow C[k] + 1$;

end for.

→ On query x, output $C[h(x)] = \tilde{f}_x$

Observation $\tilde{f}_x \geq f_x$ [could be more due to collisions]

Space: $B \cdot \log n$ + [space for hash f^n] ← $O(b \cdot \log B)$