Insertions-only streams

Piotr Indyk
MIT
The “great streaming divide”

- Insertions and deletions
  - Linear mappings
  - Randomized algorithms (typically)
  - Covered so far
- Insertions-only
  - Divide and conquer, etc.
  - Deterministic algorithms (typically)
  - Will cover in the next few lectures
Today

• L1 heavy hitters’ using $O(1/\varphi)$ words of space
  – Compare to $O(1/\varphi \log m)$ if deletions allowed

$n$ is the length of the stream
Heavy Hitters

• Define (today)

\[ \text{HH}_\varphi(x) = \{ i: |x_i| > \varphi \ |x|_1 = \varphi n \} \]

• Heavy Hitters’ Problem:
  – Parameter: \( \varphi \)
  – Goal: return a set \( S \) of coordinates s.t.
    • \( S \) contains \( \text{HH}_\varphi(x) \)
    • \( S \) has size \( O(1/\varphi) \)

• The basic idea by [Misra-Gries’82]
  – Presentation from [Demaine-Munro-OrtizLopez’02]
Warmup: $\varphi = \frac{1}{2}$

- If there exists a “majority” element, we want to find it.

- One-counter algorithm:
  - Set $c=0$
  - For each stream element $a$
    - If $c=0$ then $e=a$
    - If $e=a$ then $c=c+1$ else $c=c-1$
  - Report $e$
Correctness

• Assume majority element (MAJ) exists
• Algorithm recap:
  – Set $c=0$
  – For each stream element $a$
    • (invariant)
    • If $c=0$ then $e=a$
    • If $e=a$ then $c=c+1$ else $c=c-1$
  – Report $e$
• Invariant: if $c=0$ then MAJ is the majority element in the remainder of the stream (including $a$)
  – Between the times when $c=0$, the stream contains exactly 50% of $e$’s
General $\varphi = 1/(L+1)$

- Algorithm:
  - Set $S = \emptyset$; $c: S \rightarrow \{1 \ldots n\}$
  - For each stream element $a$
    - If $a \notin S$ and $|S| < L$ then add $a$ to $S$ and set $c(a) = 0$
    - If $a \in S$ then $c(a) = c(a) + 1$ else $c(a') = c(a') - 1$ for all $a' \in S$
    - For all $a' \in S$, if $c(a') = 0$ then remove $a'$ from $S$
  - Report $S$

- Main proof idea: each time we decrement all counters in $S$ (the “decrement event”) can be charged to $L+1$ unique stream elements
Correctness

• Iteration:
  – If \( a \notin S \) and \(|S| < L\) then add \( a \) to \( S \) and set \( c(a) = 0 \)
  – If \( a \in S \) then \( c(a) = c(a) + 1 \) else \( c(a') = c(a') - 1 \) for all \( a' \in S \)
  – For all \( a' \in S \), if \( c(a') = 0 \) then remove \( a' \) from \( S \)

• Proof:
  – Consider any element \( a \) which occurs \( > n/(L+1) \) times
  – Denote:
    • \( t_f \) = number of decrement events while reading \( a \notin S \)
    • \( t_d \) = number of decrement events while \( a \in S \)
    • \( t_i \) = number of increments while reading \( a \in S \)
    • \( t = t_f + t_i \) = total number of occurrences of \( a \)
  – By the previous slide observation, we have \( (L+1)(t_f + t_d) \leq n \)
  – If the final count of \( a \) is zero, then \( t_d = t_i \), and therefore
    \[ \frac{n}{L+1} < t = t_f + t_i = t_f + t_d \leq \frac{n}{L+1} \]
    which is a contradiction
Stronger guarantees

- One can show that
  \[ |c(a) - x_a| \leq n/(L+1) \]
- In fact, even stronger guarantees