CSCE 658: Randomized Algorithms

Lecture 8

Samson Zhou

Last Time: The Streaming Model

- Input: Elements of an underlying data set S, which arrive sequentially
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size m of the input S

1 0 1 1 1 0 0 1

Last Time: Reservoir Sampling

- Suppose we see a stream of elements from $[n]$. How do we uniformly sample one of the positions of the stream?
- [Vitter 1985]: Initialize $s = \perp$
- On the arrival of element i, replace s with x_i with probability $\frac{1}{i}$

47 72 81 10 14 33 51 29 54 9 36 46 10

Last Time: Reservoir Sampling

- Suppose we see a stream of elements from $[n]$. How do we uniformly sample one of the positions of the stream?
- [Vitter 1985]: Initialize $s = \perp$
- On the arrival of element i, replace s with x_i with probability $\frac{1}{i}$

47 72 81 10 14 33 51 29 54 9 36 46 10

Last Time: Frequent Items

- Goal: Given a set S of m elements from $[n]$ and a parameter k, output the items from $[n]$ that have frequency at least $\frac{m}{k}$

f_1	f_2	f_3	f_4	f_5	f_6	f_7
10	0	1	1	2	0	9

- How many items can be returned? At most k coordinates with frequency at least $\frac{m}{k}$

- For $k = 20$, want items that are at least 5% of the stream

Last Time: Majority

- Goal: Given a set S of m elements from $[n]$ and a parameter $k = 2$, output the items from $[n]$ that have frequency at least $\frac{m}{2}$

- Find the item that forms the majority of the stream

Last Time: Majority

- Initialize item $V = 1$ with count $c = 0$
- For updates $1, \ldots, m$:
  - If $c = 0$, set $V = x_i$ and $c = 1$
  - Else if $V = x_i$, increment counter c by setting $c = c + 1$
  - Else if $V \neq x_i$, decrement counter c by setting $c = c - 1$

- Initialize $V = x_1$ and counter $c = 1$
- If x_1 is not majority, it must be deleted at some time T
- At time T, the stream will have consumed $\frac{T}{2}$ instances of x_1, preserving majority

Misra Gries

- Drawbacks: Misra-Gries may return false positives, i.e., items that are not frequent

- In fact, no algorithm using $o(n)$ space can output ONLY the items with frequency at least $\frac{n}{k}$

- Intuition: Hard to decide whether coordinate has frequency $\frac{n}{k}$ or $\frac{n}{k} - 1$

Misra Gries

- Intuition: Hard to decide whether coordinate has frequency $\frac{n}{k}$ or $\frac{n}{k} - 1$

- $x_1 = 2, x_2 = 5, x_3 = 4, x_4 = 7, x_5 = 1, x_6 = 9, \ldots$

- $\underbrace{x_{n-\frac{n}{k}+1} = \alpha, x_{n-\frac{n}{k}+2} = \alpha, \ldots, x_n = \alpha}_{\frac{n}{k} - 1 \text{ times}}$

(ε, k)-Frequent Items Problem

- Goal: Given a set S of m elements from $[n]$, an accuracy parameter $\varepsilon \in (0,1)$, and a parameter k, output a list that includes:
  - The items from $[n]$ that have frequency at least $\frac{m}{k}$
  - No items with frequency less than $(1-\varepsilon)\frac{m}{k}$

Misra Gries for (ε, k)-Frequent Items Problem

- Initialize k items $V_1, \dots, V_k$ with count $c_1, \dots, c_k = 0$
- For updates $1, \dots, m$:
  - If $V_t = x_i$ for some t, increment counter c_t, i.e., $c_t = c_t + 1$
  - Else if $c_t = 0$ for some t, set $V_t = x_i$
  - Else decrement all counters c_j, i.e., $c_j = c_j - 1$ for all $j \in [k]$

Misra Gries for (ε, k)-Frequent Items Problem

- Set $r = \left\lceil \frac{k}{\varepsilon} \right\rceil$

- Initialize r items $V_1, \ldots, V_r$ with count $c_1, \ldots, c_r = 0$
- For updates $1, \ldots, m$:
  - If $V_t = x_i$ for some t, increment counter c_t, i.e., $c_t = c_t + 1$
  - Else if $c_t = 0$ for some t, set $V_t = x_i$
  - Else decrement all counters c_j, i.e., $c_j = c_j - 1$ for all $j \in [r]$

Misra Gries for (ε, k)-Frequent Items Problem

- Claim: For all estimated frequencies $\widehat{f_i}$ by Misra-Gries, we have

$$f_i - \frac{\varepsilon m}{k} \leq \widehat{f_i} \leq f_i$$

- Intuition: Have a lot of counters, so relatively few decrements

(ε, k)-Frequent Items Problem

- Goal: Given a set S of m elements from $[n]$, an accuracy parameter $\varepsilon \in (0,1)$, and a parameter k, output a list that includes:
  - The items from $[n]$ that have frequency at least $\frac{m}{k}$
  - No items with frequency less than $(1-\varepsilon)\frac{m}{k}$

Misra Gries for (ε, k)-Frequent Items Problem

- Set $r = \left\lceil \frac{2k}{\varepsilon} \right\rceil$ rather than $r = \left\lceil \frac{k}{\varepsilon} \right\rceil$

- Initialize r items $V_1, \ldots, V_r$ with count $c_1, \ldots, c_r = 0$
- For updates $1, \ldots, m$:
  - If $V_t = x_i$ for some t, increment counter c_t, i.e., $c_t = c_t + 1$
  - Else if $c_t = 0$ for some t, set $V_t = x_i$
  - Else decrement all counters c_j, i.e., $c_j = c_j - 1$ for all $j \in [r]$
- Output coordinates V_t with $c_t \geq (1 - \varepsilon) \cdot \frac{m}{k}$

Misra Gries for (ε, k)-Frequent Items Problem

- Claim: For all estimated frequencies $\widehat{f_i}$ by Misra-Gries, we have

$$f_i - \frac{\varepsilon m}{2k} \le \widehat{f_i} \le f_i$$

- If $f_i \ge \frac{m}{k}$, then $\widehat{f_i} \ge f_i - \frac{\varepsilon m}{2k}$ and if $f_i < (1-\varepsilon) \cdot \frac{m}{k}$, then $\widehat{f_i} < f_i - \frac{\varepsilon m}{2k}$
- Returning coordinates V_t with $c_t \ge \left(1 - \frac{\varepsilon}{2}\right) \cdot \frac{m}{k}$ means:
  - i with $f_i \ge \frac{m}{k}$ will be returned
  - NO i with $f_i < (1-\varepsilon) \cdot \frac{m}{k}$ will be returned

Misra Gries for (ε, k)-Frequent Items Problem

- Summary: Misra-Gries can be used to solve the (ε, k)-frequent items problem

- Misra-Gries uses $O\left(\frac{k}{\varepsilon}\log n\right)$ bits of space

- Misra-Gries is a deterministic algorithm

- Misra-Gries *never* overestimates the true frequency

Insertion-Deletion Streams

- Stream of length $m = \Theta(n)$
- Universe of size $[n]$, underlying vector $f \in R^n$
- Each update increases or decreases a coordinate in f

f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	0	0	0	0	0	0

- "Decrease f_6"

f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	0	0	0	0	-1	0

Insertion-Deletion Streams

- Database Management: In database management, insertion-deletion streams are used to track changes made to the database over time

- Transaction logs often utilize this concept to record insertions and deletions to ensure data integrity and support features like rollbacks and recovery

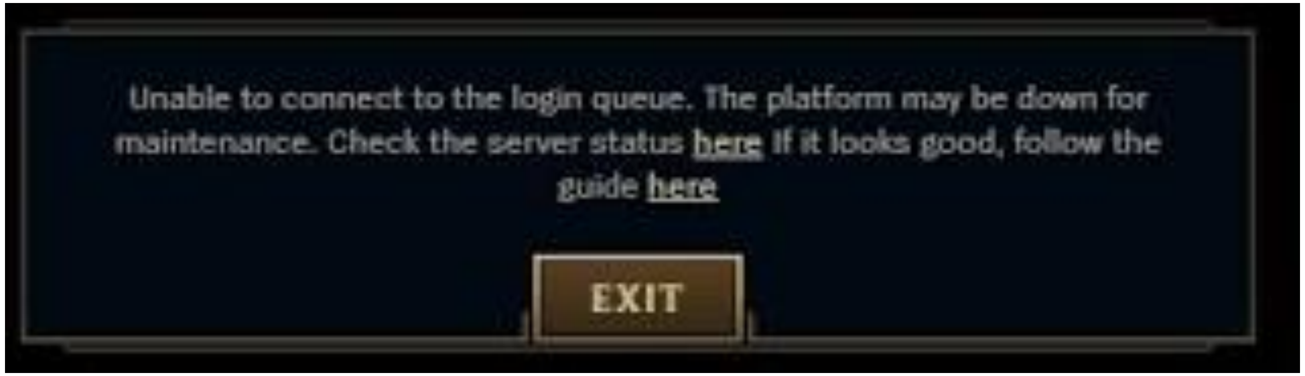
Insertion-Deletion Streams

- Version Control Systems: Insertion-deletion streams track changes made to files, enabling users to see what has been added (inserted) or removed (deleted) in each version

- Crucial for collaboration and managing software development projects, central to version control systems

Insertion-Deletion Streams

- Traffic Flow and Transportation Systems: Insertion-deletion streams are used to analyze traffic patterns and changes in transportation systems
- This helps in optimizing traffic flow, managing congestion, and improving transportation infrastructure



Frequent Items on Insertion-Deletion Streams

- Misra-Gries on Insertion-Deletion Streams
- “Increase f_1”
- “Increase f_3”
- “Increase f_2”
- “Increase f_2”
- “Decrease f_2”
- “Decrease f_2”
- “Decrease f_3”

CountMin

- Another algorithm for the (ε, k)-frequent items problem
- Can be used on insertion-deletion streams
- Can be easily parallelized across multiple servers

CountMin

- Initalization: Create b buckets of counters and use a random hash function $h: [n] \to [b]$
- Algorithm: For each update x_i, increment the counter $h(x_i)$

c_1	c_2	c_3	c_4
0	0	0	0

- At the end of the stream, output the counter $h(x_i)$ as the estimate for x_i

CountMin

f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	0	0	0	0	0	0

$$1$$

c_1	c_2	c_3	c_4
0	0	0	0

CountMin

f_1	f_2	f_3	f_4	f_5	f_6	f_7
1	0	0	0	0	0	0

$h(x) = 3x + 2 \pmod 4$

1

c_1	c_2	c_3	c_4
0	0	0	0

CountMin

f_1	f_2	f_3	f_4	f_5	f_6	f_7
1	0	0	0	0	0	0

$h(x) = 3x + 2 \pmod 4$

1

c_1	c_2	c_3	c_4
0	0	0	0

CountMin

f_1	f_2	f_3	f_4	f_5	f_6	f_7
1	0	0	0	0	0	0

$h(x) = 3x + 2 \pmod 4$

1

c_1	c_2	c_3	c_4
1	0	0	0

CountMin

f_1	f_2	f_3	f_4	f_5	f_6	f_7
1	0	0	0	0	0	0

$$h(x) = 3x + 2 \pmod 4$$

3

c_1	c_2	c_3	c_4
1	0	0	0

CountMin

f_1	f_2	f_3	f_4	f_5	f_6	f_7
1	0	1	0	0	0	0

$h(x) = 3x + 2 \pmod 4$

3

c_1	c_2	c_3	c_4
1	0	1	0

CountMin

f_1	f_2	f_3	f_4	f_5	f_6	f_7
1	0	1	0	0	0	0

$$h(x) = 3x + 2 \pmod 4$$

2

c_1	c_2	c_3	c_4
1	0	1	0

CountMin

f_1	f_2	f_3	f_4	f_5	f_6	f_7
1	1	1	0	0	0	0

$h(x) = 3x + 2 \pmod 4$

2

c_1	c_2	c_3	c_4
1	0	1	1

CountMin

f_1	f_2	f_3	f_4	f_5	f_6	f_7
1	1	1	0	0	0	0

$$h(x) = 3x + 2 \pmod 4$$

$$1$$

c_1	c_2	c_3	c_4
1	0	1	1

CountMin

f_1	f_2	f_3	f_4	f_5	f_6	f_7
2	1	1	0	0	0	0

$h(x) = 3x + 2 \pmod 4$

1

c_1	c_2	c_3	c_4
2	0	1	1

CountMin

f_1	f_2	f_3	f_4	f_5	f_6	f_7
2	1	1	0	0	0	0

$h(x) = 3x + 2 \pmod 4$

5

c_1	c_2	c_3	c_4
2	0	1	1

CountMin

f_1	f_2	f_3	f_4	f_5	f_6	f_7
2	1	1	0	1	0	0

$h(x) = 3x + 2 \pmod 4$

5

c_1	c_2	c_3	c_4
3	0	1	1

CountMin

f_1	f_2	f_3	f_4	f_5	f_6	f_7
2	1	1	0	1	0	0

$$h(x) = 3x + 2 \pmod 4$$

- What is the estimation for f_4?
- What about f_3?
- What about f_5? What about f_1?

c_1	c_2	c_3	c_4
3	0	1	1

CountMin

- Given a set S of m elements from $[n]$, let $\widehat{f_i}$ be the estimated frequency for f_i

- Claim: We always have $\widehat{f_i} \geq f_i$

- Suppose $h(i) = a$ so that $c_a = \widehat{f_i}$
- Note that c_a counts the number f_j of occurrences of any j with $h(j) = a = h(i)$, including f_i itself

CountMin

- Suppose $h(i) = a$ so that $c_a = \widehat{f_i}$
- Note that c_a counts the number f_j of occurrences of any j with $h(j) = a = h(i)$, including f_i itself
- $c_a = \sum_{j:h(j)=a} f_a \geq f_i$ since $h(i) = a$

- $c_a = f_i + \sum_{j \neq i,\text{ with } j:h(j)=a} f_j$

CountMin Error Analysis

- $c_a = f_i + \sum_{j \neq i, \text{ with } j : h(j) = a} f_j$
- What is the expected error for f_i?

CountMin Error Analysis

- $c_a = f_i + \sum_{j \neq i, \text{ with } j : h(j) = a} f_j$
- What is the expected error for f_i?
- $\mathrm{E}\left[\left|\sum_{j \neq i, \text{ with } j : h(j) = a} f_j\right|\right] \leq \sum_{j \neq i} \mathrm{E}\left[\left|f_j\right| \cdot I_{h(j) = h(i)}\right]$

CountMin Error Analysis

- $c_a = f_i + \sum_{j \neq i,\text{ with } j:h(j)=a} f_j$
- What is the expected error for f_i?
- $\mathrm{E}\left[\left|\sum_{j \neq i,\text{ with } j:h(j)=a} f_j\right|\right] \leq \sum_{j \neq i} \mathrm{E}\left[\left|f_j\right| \cdot I_{h(j)=h(i)}\right]$
  $= \sum_{j \neq i} \mathrm{E}\left[I_{h(j)=h(i)}\right] \cdot \left|f_j\right|$

CountMin Error Analysis

- $c_a = f_i + \sum_{j \neq i,\ \text{with}\ j:h(j)=a} f_j$
- What is the expected error for f_i?
- $\mathrm{E}\left[\left|\sum_{j \neq i,\ \text{with}\ j:h(j)=a} f_j\right|\right] \leq \sum_{j \neq i} \mathrm{E}\left[\left|f_j\right| \cdot I_{h(j)=h(i)}\right]$

$$= \sum_{j \neq i} \mathrm{E}\left[I_{h(j)=h(i)}\right] \cdot \left|f_j\right|$$

$$= \sum_{j \neq i} \Pr[h(j) = h(i)] \cdot \left|f_j\right|$$

CountMin Error Analysis

- $c_a = f_i + \sum_{j \neq i,\text{ with } j: h(j) = a} f_j$
- What is the expected error for f_i?
- $\mathrm{E}\left[\left|\sum_{j \neq i,\text{ with } j: h(j) = a} f_j\right|\right] \leq \sum_{j \neq i} \mathrm{E}\left[\left|f_j\right| \cdot I_{h(j) = h(i)}\right]$

$$= \sum_{j \neq i} \mathrm{E}\left[I_{h(j)=h(i)}\right] \cdot \left|f_j\right|$$

$$= \sum_{j \neq i} \Pr[h(j) = h(i)] \cdot \left|f_j\right|$$

$$= \sum_{j \neq i} \frac{1}{b} \cdot \left|f_j\right| \leq \frac{\|f\|_1}{b}$$

CountMin Error Analysis

- $c_a = f_i + \sum_{j \neq i, \text{ with } j: h(j) = a} f_j$
- What is the expected error for f_i?
- $\mathrm{E}\left[\left|\sum_{j \neq i, \text{ with } j: h(j) = a} f_j\right|\right] \leq \sum_{j \neq i} \mathrm{E}\left[\left|f_j\right| \cdot I_{h(j) = h(i)}\right]$

$$= \sum_{j \neq i} \mathrm{E}\left[I_{h(j) = h(i)}\right] \cdot \left|f_j\right|$$

$$= \sum_{j \neq i} \Pr[h(j) = h(i)] \cdot \left|f_j\right|$$

$$= \sum_{j \neq i} \frac{1}{b} \cdot \left|f_j\right| \leq \frac{\|f\|_1}{b}$$

- Set $b = \frac{9k}{\varepsilon}$, then the expected error is at most $\frac{\varepsilon \|f\|_1}{9k}$

CountMin Error Analysis

- Set $b = \frac{9k}{\varepsilon}$, then the expected error is at most $\frac{\varepsilon\|f\|_1}{9k}$

- By Markov's inequality, the error for f_i is at most $\frac{\varepsilon\|f\|_1}{3k}$ with probability at least $\frac{2}{3}$

- How to ensure accuracy for all $i \in [n]$?

CountMin Error Analysis

- By Markov's inequality, the error for f_i is at most $\frac{\varepsilon\|f\|_1}{3k}$ with probability at least $\frac{2}{3}$

- How to ensure accuracy for all $i \in [n]$?

- Repeat $\ell := O(\log n)$ times to get estimates $e_1, \ldots, e_\ell$ for each $i \in [n]$ and set $\widehat{f_i} = \mathrm{median}(\mathrm{e}_1, \ldots, \mathrm{e}_\ell)$ (or min for insertion-only)

CountMin for (ε, k)-Frequent Items Problem

- Claim: For all estimated frequencies $\widehat{f_i}$ by CountMin, we have

$$f_i - \frac{\varepsilon\|f\|_1}{3k} \le \widehat{f_i} \le f_i + \frac{\varepsilon\|f\|_1}{3k}$$

- If $f_i \ge \frac{\|f\|_1}{k}$, then $\widehat{f_i} \ge f_i - \frac{\varepsilon\|f\|_1}{2k}$ and if $f_i < (1-\varepsilon) \cdot \frac{\|f\|_1}{k}$, then $\widehat{f_i} < f_i - \frac{\varepsilon\|f\|_1}{2k}$
- Returning coordinates V_t with $c_t \ge \left(1 - \frac{\varepsilon}{2}\right) \cdot \frac{\|f\|_1}{k}$ means:
  - i with $f_i \ge \frac{\|f\|_1}{k}$ will be returned
  - NO i with $f_i < (1-\varepsilon) \cdot \frac{\|f\|_1}{k}$ will be returned

CountMin for (ε, k)-Frequent Items Problem

- Summary: CountMin can be used to solve the (ε, k)-frequent items problem on an insertion-deletion stream

- CountMin uses $O\left(\frac{k}{\varepsilon}\log^2 n\right)$ bits of space

- CountMin is a randomized algorithm

- CountMin *never* underestimates the true frequency for insertion-only streams