Sublinear Algorithms

LECTURE 9

Last time

- Approximate counting
- Estimation of the 2^{nd} moment
- Linear sketching

Today

- Multipurpose sketches
- Count-min and count-sketch
- Range queries, heavy hitters, quantiles

Project meetings (today) will be on Zoom
Project proposals due Tuesday, Feb 25 at 11am on Gradescope

Multipurpose Sketches: Problems

Input: a stream $\langle a_1, a_2, \dots, a_m \rangle \in [n]^m$

- The frequency vector of the stream is $f = (f_1, \dots, f_n)$, where f_i is the number of times i appears in the stream

Goal: to maintain data structures that can answer the following queries

- Point Query: For $i \in [n]$, estimate f_i
- Range Query: For $i, j \in$ [n], estimate $f_i + f_{i+1} + \dots + f_j$
- Quantile Query: For $\phi \in [0, 1]$, find j with $f_1 + \dots + f_j \approx \phi m$
- Heavy Hitters Query: For $\phi \in [0, 1]$, find all i with $f_i \geq \phi m$.

Desired accuracy: $\pm \varepsilon m$ with error probability δ

Based on Andrew McGregor's slides: https://people.cs.umass.edu/~mcgregor/711S18/vectors-3.pdf

Today's Lecture

Techniques

- Randomized Count-Min sketch
- Deterministic Count-Min sketch
- Binary trees for range and efficient heavy-hitters
- Count-Sketch = Count-Min + AMS

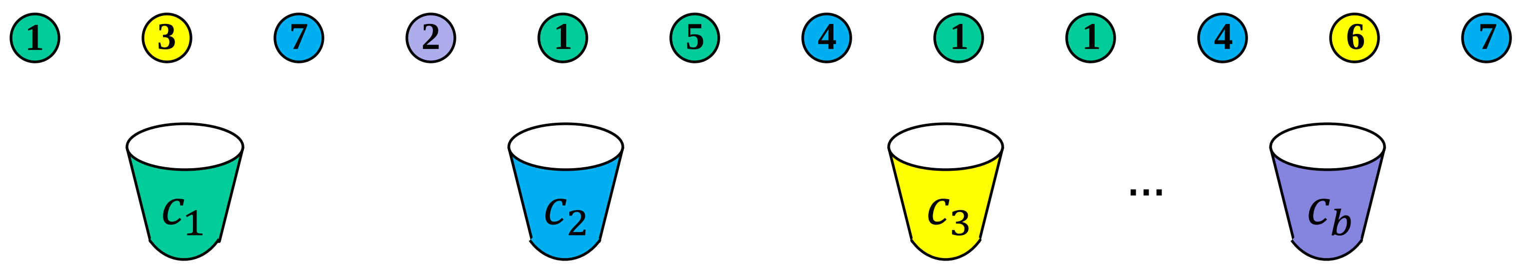
Takeaways

- We can do cool stuff in polylog space
- Good algorithms are versatile
- Can get better analyses using more than stream length

Initial Solution to Point Queries

- We could maintain the whole frequency vector $(f_1, \dots, f_n)$
- Then, on query i, we can output f_i

Idea: Group counts for some numbers together



If i falls into bucket j, then $f_i \leq c_j$.

Point Query Algorithm (initial version)

1. Sample a hash function $h : [n] \to [b]$ from a 2-wise independent family
2. Initialize counters $c_1, \dots, c_b$ to 0
3. For each element a, increment $c_{h(a)}$ by 1.
4. To answer a point query i, return $c_{h(i)}$.

Never underestimate

Initial Solution to Point Queries: Analysis

Point Query Algorithm (initial version)

1. Sample a hash function $h : [n] \to [b]$ from a 2-wise independent family
2. Initialize counters $c_1, \dots, c_b$ to 0
3. For each element a, increment $c_{h(a)}$ by 1.
4. To answer a point query i, return $c_{h(i)}$.

Never underestimate

- Fix $i^* \in [n]$.
- Let $Z = c_{h(i^*)} - f_{i^*}$ be the overestimation error.

by 2-wise independence

- For all $i \neq i^*$, let $X_i = \begin{cases} 1 & \text{if } h(i) = h(i^*) \\ 0 & \text{otherwise} \end{cases}$

$$\mathbb{E}[X_i] = \Pr[h(i) = h(i^*)] = \frac{1}{b}$$

$$Z = \sum_{i \neq i^*} X_i \cdot f_i$$

$$\mathbb{E}[Z] = \sum_{i \neq i^*} \mathbb{E}[X_i] \cdot f_i = \frac{1}{b} \sum_{i \neq i^*} f_i \leq \frac{m}{b}$$

by linearity of expectation

- By Markov's inequality, if $b = 2/\varepsilon$ then

$$\Pr[Z \geq \varepsilon m] \leq \frac{\mathbb{E}[Z]}{\varepsilon m} \leq \frac{1}{\varepsilon b} \leq \frac{1}{2}$$

Count-Min Sketch [Cormode Muthukrishnan 05]

Point Query Algorithm

1. Set $t = \log_2 1/\delta$ and $b = 2/\varepsilon$
2. Sample t hash functions $h_j: [n] \to [b]$ from a 2-wise independent family
3. Initialize tb counters $c_{j,k}$ to 0
4. For each element a and each $j \in [t]$, increment $c_{j,h(a)}$ by 1.
5. To answer a point query i, return $\widetilde{f_i} = \min_{j \in [t]} c_{j,h(i)}$.

Never underestimate

- Correctness: $\Pr\left[f_i \le \widetilde{f_i} \le f_i + \varepsilon m\right]$

$$= 1 - \Pr[\text{all } t \text{ hash functions overestimate by more than } \varepsilon m]$$

$$\ge 1 - \left(\frac{1}{2}\right)^t = 1 - \delta$$

since hash functions are chosen independently

- Space: $O(t\,(\log n + \log b))$ for the hash functions + $O(tb \log m)$ for the counters

Total: $O\left(\left(\log n + \frac{1}{\varepsilon}\log m\right)\log\frac{1}{\delta}\right)$

Review: Is Count-Min a linear sketch?

Recall a **linear sketch**

- is given by a (maybe randomly chosen) $k \times n$ matrix S,
- stores Sf (a vector with k entries) where f is the frequency vector of the stream

True or false? Count-Min is linear.

Is randomization necessary for streaming?

- Count-Min relies heavily on hash functions being random
  - Space $O\left(\left(\log n + \frac{1}{\varepsilon}\log m\right)\log\frac{1}{\delta}\right)$ dominated by hash functions if n is big

- Can we get away without randomness?
  - Not possible for many tasks (e.g.. distinct elements)
  - What about point queries?

- What do we need from the hash functions?
  - In some bucket, i^* should collide with “few” other stream elements
  - But we actually bounded the average over hash functions we use of the number of collisions
  - Suffices for each $i \neq i^*$ to collide for $\epsilon/2$ fraction of the hash functions

CR-Precis: Deterministic Count-Min [Ganguly Majumder 07]

Use deterministic hash functions:

$$h_j(a) = a \bmod p_j, \text{ where } p_j \text{ is the } j\text{-th prime, for } j \in [t]$$

Analysis: Fix $i^* \in [n]$. Define $z_1, \ldots, z_t$ such that $c_{j,h_j(i^*)} = f_{i^*} + z_j$, that is,

$$z_j = \sum_{i \neq i^* : h_j(i) = h_j(i^*)} f_i$$

- Let $B_i = \{j: \quad h_j(i) = h_j(i^*)\}$
- Claim: For each $i \neq i^*$, we have $|B_i| \leq \log n$.

by Chinese Remainder Theorem

- Thus, $\sum_{j \in [t]} z_j = \sum_i \sum_{j: h_j(i) = h_j(i^*)} f_i = \sum_i \sum_{j \in B_i} f_i \leq \sum_i f_i \log n = m \log n$

$$\widetilde{f_{i^*}} = \min_{j \in [t]} c_{j,h(i^*)} = \min_{j \in [t]} (f_{i^*} + z_j) = f_{i^*} + \min_{j \in [t]} z_j \leq f_{i^*} + \frac{m \log n}{t}$$

- We set $t = \frac{\log n}{\varepsilon}$ to get $f_i \leq \widetilde{f_i} \leq f_i + \varepsilon m$
- Requires keeping at most $t \cdot p_t = \tilde{O}\left(\frac{\log^2 n}{\varepsilon^2}\right)$ counters since $p_t = O(t \log t)$

by number theory magic

Exercise: Improve this construction using error-correcting codes.

Today's Lecture

Techniques

- Randomized Count-Min sketch
- Deterministic Count-Min sketch
- Binary trees for range and efficient heavy-hitters
- Count-Sketch = Count-Min + AMS

Takeaways

- We can do cool stuff in polylog space
- Good algorithms are versatile
- Can get better analyses using more than stream length

Multipurpose Sketches: Problems

Input: a stream $\langle a_1, a_2, \dots, a_m \rangle \in [n]^m$

- The frequency vector of the stream is $f = (f_1, \dots, f_n)$,
  where f_i is the number of times i appears in the stream

Goal: to maintain data structures that can answer the following queries

- Point Query: For $i \in [n]$, estimate f_i
- Range Query: For $i, j \in$ [n], estimate $f_i + f_{i+1} + \dots + f_j$

Denote by $f_{[i,j]}$

- Quantile Query: For $\phi \in [0, 1]$, find j with $f_1 + \dots + f_j \approx \phi m$
- Heavy Hitters Query: For $\phi \in [0, 1]$, find all i with $f_i \geq \phi m$.

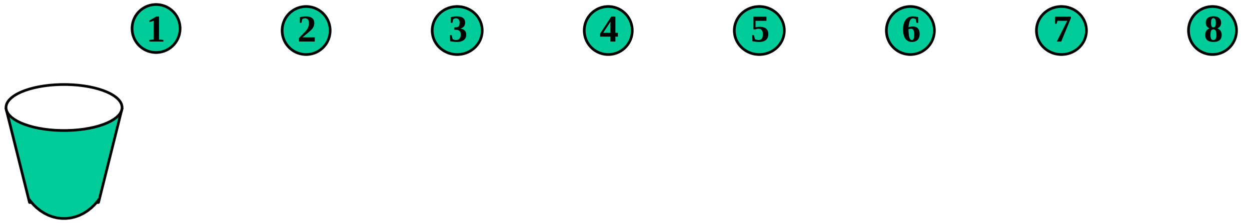
Desired accuracy: $\pm \varepsilon m$ with error probability δ

Based on Andrew McGregor's slides: https://people.cs.umass.edu/~mcgregor/711S18/vectors-3.pdf

Range Queries

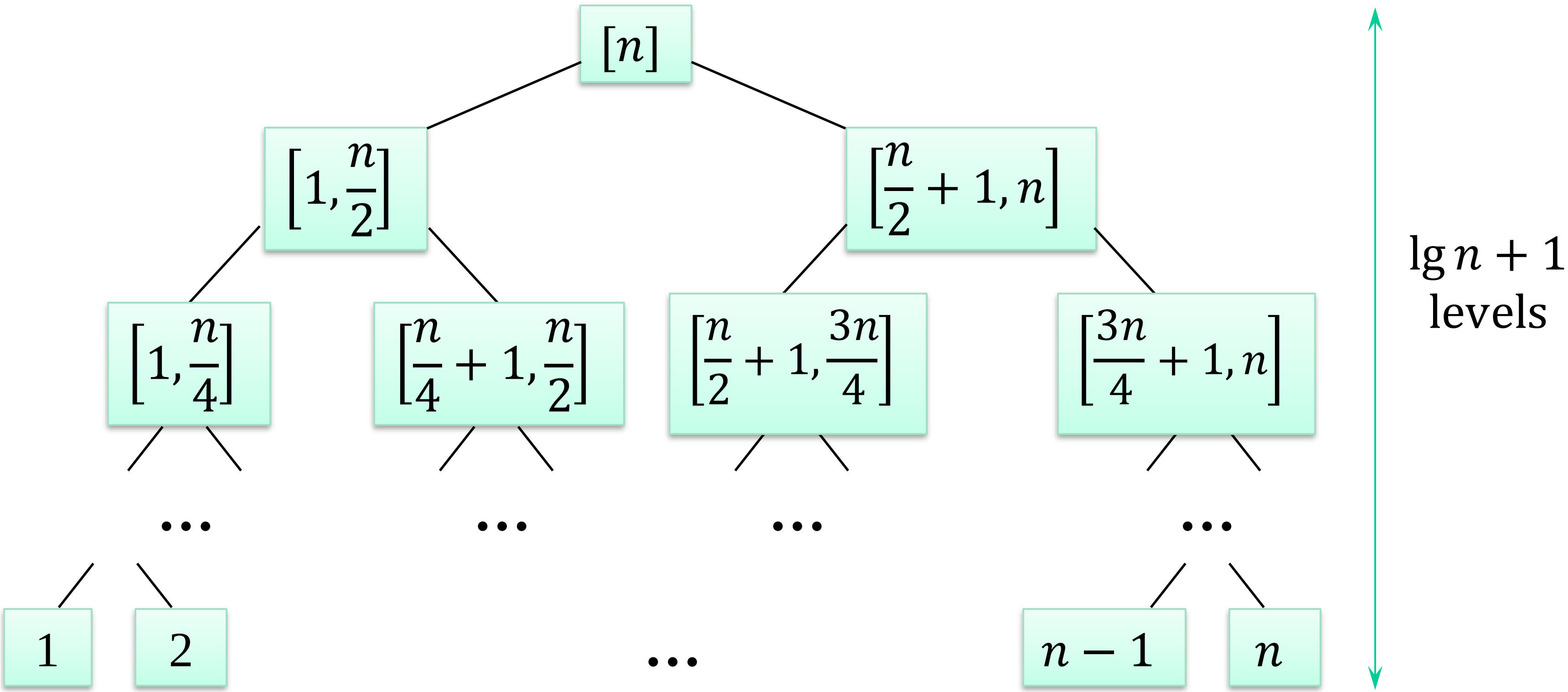
- We could estimate $f_{[i,j]}$ by $\widetilde{f_i} + \tilde{f}_{i+1} + \ldots + \widetilde{f_j}$
  But errors add up: need too much space to keep accurate enough estimates

Idea: We could estimate counts for some intervals directly by grouping $i, \ldots, j$



How many intervals do we need so that each interval is a sum of $O(\log n)$ original intervals?

Dyadic Intervals



- Exercise: Each interval $[i, j]$ is a sum of at most $2 \lg n$ dyadic intervals.
- Such a representation of an interval is its dyadic decomposition.

Count-Min Strikes Back

Range Query Algorithm

1. Construct $\lg n + 1$ Count-Min sketches, one for each level, such that for all intervals I at that level, our estimate $\tilde{f}_I$ for f_I satisfies
$$\Pr\left[f_I \le \tilde{f}_I \le f_I + \varepsilon m\right] \le 1 - \delta$$
2. To answer a range query $[i, j]$, let $I_1, \dots, I_k$ be its dyadic decomposition
Return $\tilde{f}_{[i,j]} = \tilde{f}_{I_1} + \cdots + \tilde{f}_{I_k}$

- Correctness: $\Pr\left[f_{[i,j]} \le \tilde{f}_{[i,j]} \le f_{[i,j]} + \varepsilon m(2 \lg n)\right] \ge 1 - \delta(2 \lg n)$
- Space:

  Multiply the old space complexity by $\log n$ and divide ε and δ by $\log n$:
$$O\left(\log^2 n\,(\log n + \log m)\frac{1}{\varepsilon}\log\frac{\log n}{\delta}\right)$$
- Quantile Query: For $\phi \in [0, 1]$ find j with $f_{[1,j]} \approx \phi m$

Approximate Median: Find j such that $f_{[1,j]} \ge \frac{m}{2} - \varepsilon m$ and $f_{[1,j-1]} \le \frac{m}{2} + \varepsilon m$

We can approximate median via binary search of range queries.

Based on Andrew McGregor's slides: https://people.cs.umass.edu/~mcgregor/711S18/vectors-3.pdf

Count-Min Strikes Back (Part 2)

Heavy Hitters Query: For $\phi \in (\varepsilon, 1-\varepsilon)$, find a set S that

- includes all i with $f_i \geq (\phi + \varepsilon)m$
- excludes all j with $f_j \leq (\phi - \varepsilon)m$

First attempt:

- Use CM sketch to evaluate $\widetilde{f_i}$ for every $i \in [n]$, return $S = \{i : \widetilde{f_i} \geq \phi m\}$
  - If **all $\boldsymbol{n}$ estimates** are accurate $\pm\varepsilon m$, then S is correct.
  - Space usage?
  - Why is this unsatisfactory?

Count-Min Strikes Back (Part 2)

Heavy Hitters Query: For $\phi \in (\varepsilon, 1-\varepsilon)$, find a set S that

- includes all i with $f_i \geq (\phi + \varepsilon)m$
- excludes all j with $f_j \leq (\phi - \varepsilon)m$

Heavy Hitters Algorithm

1. Construct $\lg n + 1$ Count-Min sketches for levels of dyadic tree, as before
2. To answer query ϕ, mark the root. Going level-by-level from the root, mark children I of marked nodes if $\tilde{f}_I \geq \phi m$
3. Return all marked leaves

Correctness: If $f_i \geq \phi m$, then for all ancestors I of the leaf i,

$$\tilde{f}_I \geq f_I \geq \phi m$$

- If we ensure that Pr[point query overestimates by $> \varepsilon m] \leq \delta / n$, then, by union bound, all point queries are correct w.p. $\geq 1 - \delta$
- There are at most $1/\phi$ indices i with $f_i \geq \phi m$

  Thus, $O(\phi^{-1} \log n)$ time suffices for post-processing

Based on Andrew McGregor's slides: https://people.cs.umass.edu/~mcgregor/711S18/vectors-3.pdf

Today's Lecture

Techniques

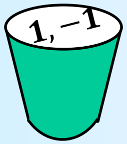
- Randomized Count-Min sketch
- Deterministic Count-Min sketch
- Binary trees for range and efficient heavy-hitters
- Count-Sketch = Count-Min + AMS

Takeaways

- We can do cool stuff in polylog space
- Good algorithms are versatile
- Can get better analyses using more than stream length

Count-Sketch: Count-Min+AMS combined

Count-Sketch



1. In addition to $h_j : [n] \to [b]$, use hash functions $r_j : [n] \to \{-1,1\}$
2. Maintain tb counters $c_{j,k} = \sum_{i:h_j(i)=k} r_j(i) f_i$
3. To answer a point query i, return $\hat{f}_i = \text{median}\left(r_1(i) c_{1,h_1(i)}, \dots, r_t(i) c_{t,h_t(i)}\right)$

Claim. $\mathbb{E}\left[r_j(i) c_{j,h_j(i)}\right] = f_i$ and $\mathrm{Var}\left[r_j(i) c_{j,h_j(i)}\right] \le \frac{F_2}{b} \quad \forall j \in [t]$

- By Chebyshev, for $b = 2/\varepsilon^2$,

$$\Pr\left[\left|f_i - r_j(i) c_{j,h_j(i)}\right| \ge \varepsilon\sqrt{F_2}\right] \le \frac{F_2}{\varepsilon^2 b F_2} = \frac{1}{3}$$

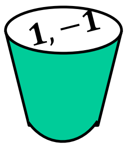
- By Chernoff, for $t = \Theta(\log 1/\delta)$

$$\Pr\left[\left|f_i - \hat{f}_i\right| \ge \varepsilon\sqrt{F_2}\right] \le \delta$$

Recall that $\sqrt{F_2} \le F_1 = m$, so this is a better error guarantee than just "$\pm\varepsilon m$"
(but b now grows as $1/\varepsilon^2$)

Count-Sketch: Proof of Claim

Count-Sketch: $\hat{f}_i = \text{median}\left(r_1(i)c_{1,h_1(i)}, \dots, r_t(i)c_{t,h_t(i)}\right)$



Claim. $\mathbb{E}\left[r_j(i)c_{j,h_j(i)}\right] = f_i$ and $\text{Var}\left[r_t(i)c_{t,h_t(i)}\right] \le \frac{F_2}{b} \quad \forall j \in [t]$

Proof: Fix $i = i^*$ and $j \in [b]$. We omit subscripts j.

- For all $i \neq i^*$, let $X_i = \begin{cases} 1 & \text{if } h(i) = h(i^*) \\ 0 & \text{otherwise} \end{cases}$

by 2-wise independence

- Expectation: $\mathbb{E}\left[r(i^*)\, c_{h(i^*)}\right] = \mathbb{E}\left[f_i^* + \sum_{i \neq i^*} r(i)r(i^*)X_i f_i\right] = f_i^*$

- Variance: $\text{Var}\left[r(i^*)\, c_{h(i^*)}\right] \le \mathbb{E}\left[\left(\sum_{i \neq i^*} r(i)r(i^*)X_i f_i\right)^2\right]$

$$= \mathbb{E}\left[\sum_{i \neq i^*} X_i^2 f_i^2 + \sum_{i \neq k} r(i)r(k)X_i X_k f_i f_k\right] = \frac{F_2}{b}$$

Today's Lecture

Techniques

- Randomized Count-Min sketch
- Deterministic Count-Min sketch
- Binary trees for range and efficient heavy-hitters
- Count-Sketch = Count-Min + AMS

Takeaways

- We can do cool stuff in polylog space
- Good algorithms are versatile
- Can get better analyses using more than stream length