Goals

Streaming Algorithms

- Distinct Elements
- Heavy Hitters

HW4 due tonight

Practice Exam released today

Office hours 4-6

Mckelvey Hall 2nd floor

Overview

Model: given some high-dim vector $x = \mathbb{R}^n$, and we want to support updates to x, queries about x

goal: minimize space complexity while using stream once

Turnstile model:
a stream of
- updates (i, v) (length m)
causes $x_i \leftarrow x_i + v$

- x starts as $\vec{0} \in \mathbb{R}^n$
- We're going to assume v is positive (insertion only)
- example queries:
  1) what is x_i?
  2) How many distinct x_i's?

- Basic algorithm - remember x or whole stream (space is either n or m)

Goals:
- Only use stream once
- Use sublinear space

Distinct Elements

deterministic

X

One goal: use very little space

query(x) = |support(x)| = F_0

X. all updates (i, v) have v = 1

claim: Any deterministic alg for F_0 requires

space = $\Omega(\min\{n, m\})$

Proof? Whoa to find

(Alon, Matias, Szegedy)

1996

Basic Idea:

Want a compression, that is an injection.

Can't compress n bits to be smaller

Distinct Elements Rand. Approx.

Det. / Exact F_0?	X	Same paper (AMS)
Det. / approx ?	X	
Random / Exact ?	X	
Random / Approx ?	✓	

Goal Develop a random algorithm A s.t. $\forall s$

1/3 can be smaller $\quad P_r(|A(s) - F_0| > \varepsilon \cdot f_0) < \frac{1}{3}$

Idealized algorithm

- we have a totally random hash function $h: \{1, \ldots, n\} \to [0,1]$ (uniform)

Let $Z = \min h(i)$

Output $= \frac{1}{Z} - 1$

set $R = \Theta(\frac{1}{\varepsilon^2})$

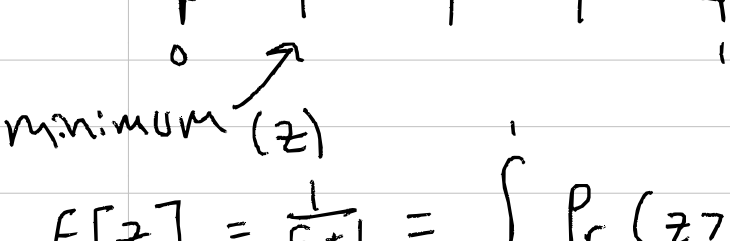
$h_1, \ldots, h_R$

Let $Z_j = \min h_j(i)$ for hash function j

$\overline{Z} = \frac{1}{R} \sum_{j=1}^{R} Z_j$

Output: $\frac{1}{\overline{Z}} - 1$

Proof



$$E[Z] = \frac{1}{F_0+1} = \int_0^1 P_r(Z > x)\,dx = \int_0^1 P_r(\forall i \; h(i) > x) = \int_0^1 (1-x)^{F_0}$$

$$E[Z^2] = \frac{2}{(F_0+1)(F_0+2)}$$

Chebyshevs

$$P_r(|Z - EZ| > \varepsilon EZ) < \frac{Var[Z]}{\varepsilon^2 \cdot (EZ)^2} \leq \frac{E[Z^2]}{\varepsilon^2 E[Z]^2}$$

$E[Z]$ is close to Z

So Z is close to $\frac{1}{F_0+1}$

So output is close to F_0

What's the point of averaging?

Expectation stays the same

We can lower the variance which lowers ε

Variance of R variables will lower variance by ε^2 which will kill the ε^2 in the final equation

Majority

Given that a majority element exists can we find it?

Assume:

$$\exists\, i \text{ s.t. } x_i > \frac{\sum X}{2}$$

Algorithm:

Set $c = 0$ and $\ell = \emptyset$ ← most common value

for update in stream:

```
if update = ℓ
    c = c + 1
else
    c = c - 1

if (c ≤ 0)
    c = 1, ℓ = update
```

Proof sketch:

cross canceling every value that is not the majority removes 1 item that is

K- Heavy Hitters

Can we find all elements that appear ε fraction of the stream?

return $L \subseteq [1,n]$ s.t

if $|x_i| > \varepsilon \|x\|_1$, $i \in L$

else if $|x_i| < \varepsilon/2 \|x\|_1$, $i \notin L$

else $i \in L$ or $i \notin L$

Algorithm (Misra-Gries)

Majority, but with k counters and locations ($k = \frac{1}{\varepsilon}$)

dictionary L

for update in stream

if x_i in L

$L[x_i] = L[x_i] + 1$

else if

size$(L) < k-1$

add x_i

else

for $x_i \in L$

$L[x_i] = L[x_i] - 1$

space $= O(k(\log(m) + \log(n))$

Limitations

We don't know which elements are heavy

We don't know the counts for any element

We could loop again, but that fails a goal from the beginning, so does storing m

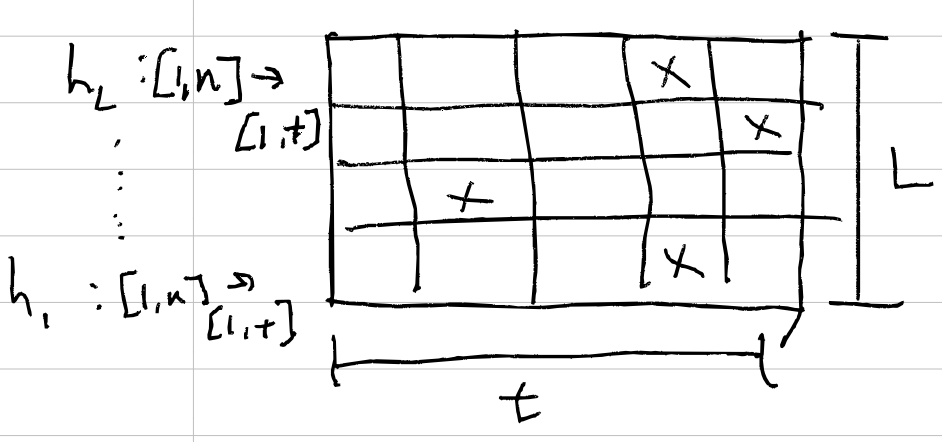
Randomized Point Query

$$Query(i) = x_i \pm \varepsilon \|x\|_1$$
for all $i \in n$

Count min (CM) sketch

say- "A sketch is just an approximate data structure"

- Hashing



- Counters $C_{a,b}$ $a \in [1,L]$, $b \in [1,t]$
- $C_{a,b} = \sum_{\substack{i \in [1,n] \\ h_a(i)=b}} x_i$

Algorithm:

ε-Point query w/ failure prob δ

$$Pr(|A(x_i) - x_i| > \varepsilon \cdot \|x\|_1) \leq \delta$$

$$t = \frac{2}{\varepsilon}, \quad L = \log_2\left(\frac{1}{\delta}\right)$$

Output: $\min_{1 \leq r \leq L} C_{r, h_r(i)}$

What is in the counter?

every count is an overestimate of x_i.

This is because it contains all x_i + collisions

What is the best overestimate?

the smallest one

Claim:

fix i, $r \in [1,L]$ let

$$z_j = \begin{cases} 1 & h_r(i) = h_r(i) \\ 0 & \text{else} \end{cases}$$

$$C_{r, h_r(i)} = x_i + \underbrace{\sum_{j \neq i} x_j \cdot z_i}_{(e)} \quad \leftarrow \text{Error}$$

- $E[e] = \sum_{j \neq i} x_j \cdot E[z_j]$

$E[z_j] = \frac{1}{t}$ by pairwise independence

$$E[e] = \sum_{j \neq i} \frac{x_j}{t} \leq \frac{\varepsilon}{2} \cdot \|x\|_1$$

using markov's

$$Pr(e > \varepsilon \cdot \|x\|_1) \leq \frac{1}{2}$$

{Every counter has to fail for the min to fail}

$$Pr\left(\min_r C_{r, h_r(i)} > x_i + \varepsilon \|x\|_1\right) < \frac{1}{2^L} = \delta$$

α - Heavy hitters w.p. $1-\delta$

Naive - do point query w/ $\varepsilon = \frac{\alpha}{4}$, $\delta = \frac{\delta}{n}$

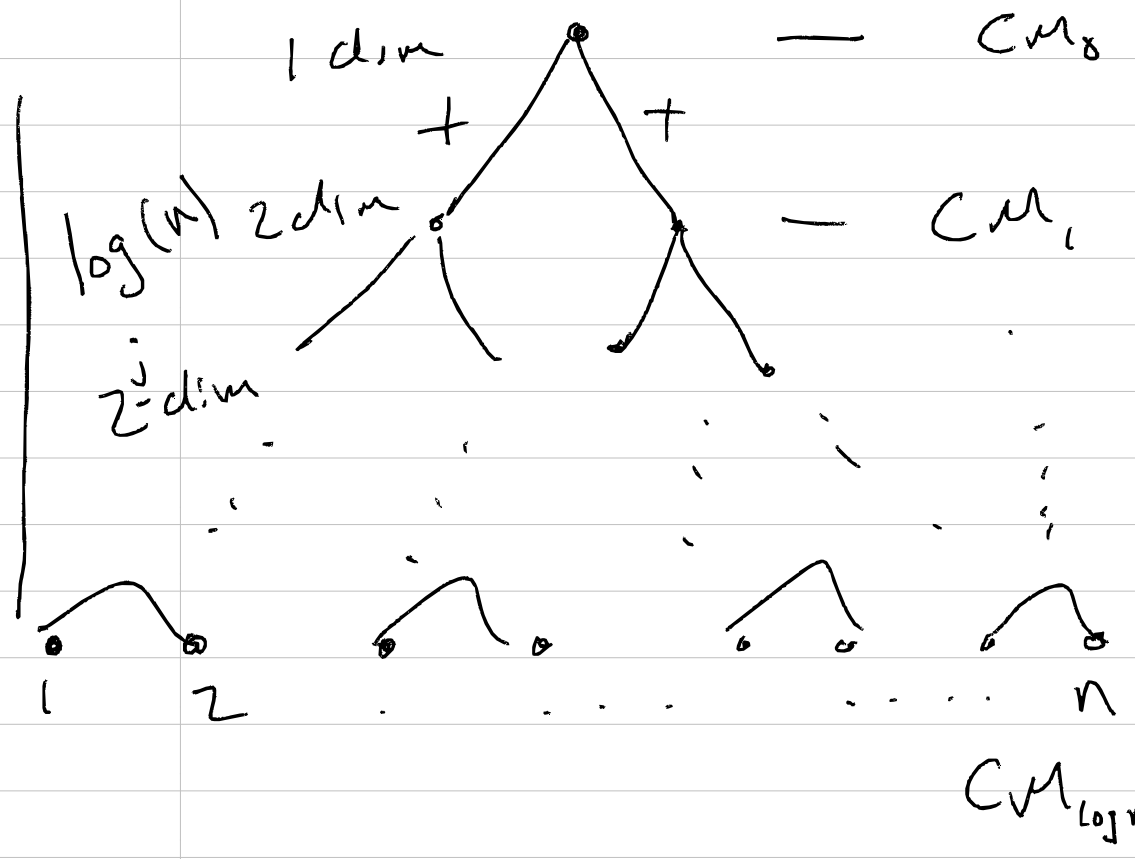
$$\text{Space} = O\left(\frac{1}{\alpha} \cdot \log\left(\frac{n}{\delta}\right)\right),$$

$$\text{Time} = O\left(n \cdot \log\left(\frac{n}{\delta}\right)\right)$$

Can we do better?

ε- guarentees we exclude all $\alpha/2$ elements

faster α-heavy hitters



Words

each node is the sum of its children

if a node at level i is a HH then all parents are HH

- Sum is the same at every level

- if a node contains a heavy hitter amongst the leaves in its sub tree $\Rightarrow$ it is a HH at its level

---

updates are done up the path

queries are done by starting at the root.

root is always a HH

ask which children are HH

Compare to Search (BFS)

How many HH at each level?

- only $\leq \frac{2}{\alpha}$ nodes at any level which are $\frac{\alpha}{2}$-HH at that level
$\Rightarrow$ we only touch $\frac{2}{\alpha} \cdot \log n$ vertices during search

---

What are the ε, δ

CM_j, $\varepsilon = \frac{\alpha}{4}$

$\delta = \frac{1}{\frac{4}{\alpha} \cdot \log(n)}$ ← how many queries done inside

$\text{Space}(CM_j) = O\left(\frac{1}{\alpha} \cdot \log\left(\frac{\log n}{\alpha \delta}\right)\right)$

total space $= O\left(\frac{1}{\alpha} \cdot \log n \cdot \log\left(\frac{\log n}{\alpha \delta}\right)\right)$

runtime $= O\left(\frac{1}{\alpha} \log n\right)$