

Hashing and Heavy Hitters

welcome to Randomized Algorithms

[Fundamentalalgorithms.com/randomized](https://fundamentalalgorithms.com/randomized)

Randomized Algorithms, CS588, Fall 2025

Lectures: 4:30 to 5:45 PM, Tuesday and Thursday, in Forney Hall G124.

Staff:	Email (@purdue.edu)	Office hours
Kent Quanrud	krq	Tuesday, 2–3PM, Lawson 1211
Tanmay Devale	tdevale	TBD

Please see the syllabus (Page 391) for more information.

If you are unable to register for the class because it is full, just wait. Historically, slots have always opened up. You can add yourself to gradescope and submit homework in the meantime.

Homework

- Homework 0 due September 4.
- Homework 1 due September 18.
- Homework 2 due October 2.
- Homework 3 due October 16.
- Homework 4 due October 30.
- Homework 5 due November 13.
- Homework 6 due December 4.

Class schedule¹

1. *August 26.* Randomized SAT and quicksort (Chapter 1). Please read Chapter 0 before class.

Please read the syllabus

– will ask for questions next class

Highlights

midterm, final

biweekly homework

drop lowest 20% of HW scores

"self-grading" HW policy

HW in groups of 3

Probability

Events, probabilities,
conditional probabilities,
union bound
Random variables,
expected value,
linearity of expectation
Markov's inequality

Algorithms

Approx. 3-SAT,
Quicksort,
Quickselect (HW)

2 disjoint clauses

① prove silly algo is $7/8$ -APX

1. for $i=1, \dots, n$ The silliest APX algorithm
 a. randomly set $x_i = \begin{cases} \text{TRUE} & \text{w/ prob. } .5 \\ \text{FALSE} & \text{w/ prob. } .5 \end{cases}$

$$\mathcal{S}(x) = (\underbrace{x_1 \vee x_2 \vee \bar{x}_3}_{C_1}) \wedge (\underbrace{x_4 \vee \bar{x}_5 \vee x_6}_{C_2})$$

E_1 = event that C_1 satisfied

E_2 = ~ C_2 ~

$$\Pr[E_1, E_2] = \frac{7}{8} \cdot \frac{7}{8}$$

$$= \Pr[E_1] \cdot \Pr[E_2]$$

2 clauses w/
1 common variable

① prove silly algo is $7/8$ -APX

1. for $i=1, \dots, n$ The silliest APX algorithm
 a. randomly set $x_i = \begin{cases} \text{TRUE} & \text{w/ prob. } .5 \\ \text{FALSE} & \text{w/ prob. } .5 \end{cases}$

$$\mathcal{S}(x) = (\underbrace{x_1 \vee \bar{x}_2 \vee x_3}_{C_1}) \wedge (\underbrace{x_3 \vee x_4 \vee \bar{x}_5}_{C_2})$$

E_1 = event that C_1 satisfied

E_2 = ~ C_2 ~

$$\Pr[E_1, E_2] =$$

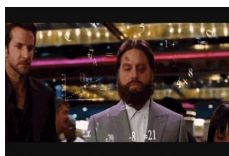
$$\underbrace{\Pr[E_1, E_2 | x_3 = T] \cdot \frac{1}{2}}_1 + \underbrace{\Pr[E_1, E_2 | x_3 = F] \cdot \frac{1}{2}}_{(\frac{3}{4})^2}$$

m clauses w/
common variables

① prove silly algo is $7/8$ -APX

1. for $i=1, \dots, n$ The silliest APX algorithm
 a. randomly set $x_i = \begin{cases} \text{TRUE} & \text{w/ prob. } .5 \\ \text{FALSE} & \text{w/ prob. } .5 \end{cases}$

$$\mathcal{S}(x) = (\underbrace{x_1 \vee \bar{x}_2 \vee x_3}_{C_1}) \wedge (\underbrace{x_3 \vee x_4 \vee \bar{x}_5}_{C_2}) \wedge (\underbrace{x_2 \vee \bar{x}_4 \vee x_6}_{C_3}) \wedge \dots$$



E_1 = event that C_1 satisfied

E_2 = ~ C_2 ~

E_3 = ~ C_3 ~

$\vdots$

general question

how to analyze a randomized mechanism
w/ a combinatorial explosion of possibilities?

Big idea: we only want the average

Linearity of Expectation

$$E[X+Y] = E[X] + E[Y] \quad (\text{ridiculously useful})$$

"average sum = sum of averages"

$$\text{let } Y_i = \begin{cases} 1 & \text{if } i\text{th clause satisfied} \\ 0 & \text{otherwise} \end{cases}$$

($i=1, \dots, m$)

$$E[Y_i] = 7/8 \quad (1 \text{ clause case})$$

$$E[\text{\# clauses satisfied}] = E\left[\sum_{i=1}^m Y_i\right] = \sum_{i=1}^m E[Y_i] = 7/8 m$$

(actually easy!)

let $Y_i = \begin{cases} 1 & \text{if } i\text{th clause satisfied} \\ 0 & \text{otherwise} \end{cases}$

$$Z = \sum_{i=1}^m Y_i = \# \text{ clauses satisfied}$$

$$E[Z] = \frac{1}{2} E[Z | x_1 = \text{true}] + \frac{1}{2} E[Z | x_1 = \text{false}]$$

$\Rightarrow$ either $E[Z | x_1 = \text{true}]$ or $E[Z | x_1 = \text{false}]$ is $\geq E[Z]$!

$E[Z | x_1 = \text{true}]$ or $E[Z | x_1 = \text{false}]$ is $\geq E[Z]$!

how to decide which?

$$E[Z | x_1 = \text{true}] = \sum_{i=1}^m E[Y_i | x_1 = \text{true}]$$

$\uparrow$
easy to calculate

so we can compute $E[Z | x_1 = \text{true}]$, $E[Z | x_2 = \text{false}]$
and decide better

$Y_i = \begin{cases} 1 & \text{if } i\text{th clause satisfied} \\ 0 & \text{otherwise} \end{cases}$

$$Z = \sum_{i=1}^m Y_i = \# \text{ clauses satisfied}$$

$$E[Z] = \frac{1}{2} E[Z | x_1 = \text{true}] + E[Z | x_1 = \text{false}]$$

induction step

$a_1, \dots, a_k \in \{\text{true}, \text{false}\}$ s.t.

$$E[Z | x_1 = a_1, \dots, x_k = a_k] \geq E[Z]$$

for x_{k+1} :

$$E[Z | x_1 = a_1, \dots, x_k = a_k] =$$

$$\frac{1}{2} E[Z | x_1 = a_1, \dots, x_k = a_k, x_{k+1} = \text{true}]$$

$$+ \frac{1}{2} E[Z | x_1 = a_1, \dots, x_k = a_k, x_{k+1} = \text{false}]$$

"method of conditional expectations"

$Y_i = \begin{cases} 1 & \text{if } i\text{th clause satisfied} \\ 0 & \text{otherwise} \end{cases}$

$$Z = \sum_{i=1}^m Y_i = \# \text{ clauses satisfied}$$

$$E[Z] = \frac{1}{2} E[Z | x_1 = \text{true}] + E[Z | x_1 = \text{false}]$$

$$E[Z | x_1 = \text{true}] = \sum_{i=1}^m E[Y_i | x_1 = \text{true}]$$

$\uparrow$
easy to calculate

one of these
must be good

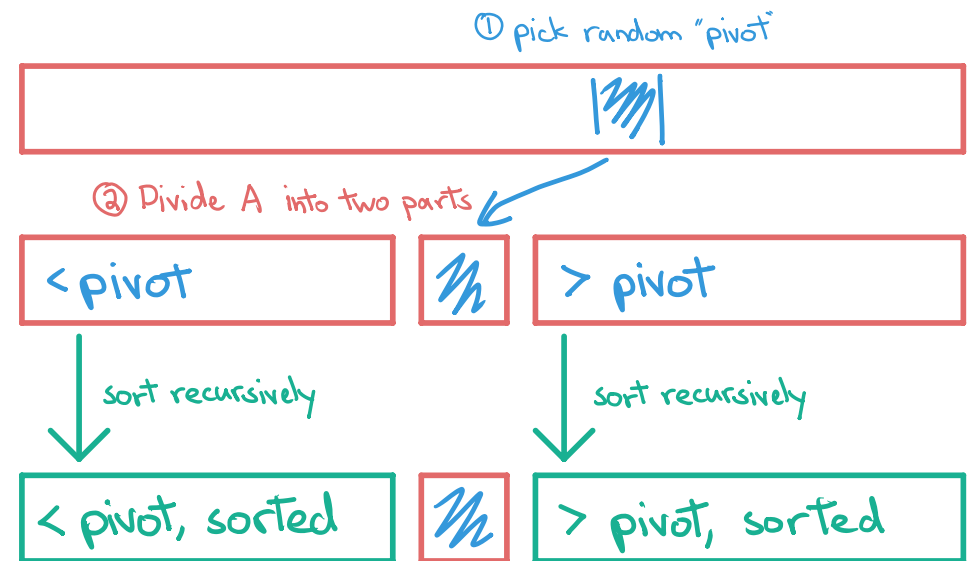
Randomized Sorting

Goal: sort $A[1, n, n]$

Silly algorithm:

1. choose $i \in [n]$ uniformly at random
 2. Recursively sort all #'s $> A[i]$
 3. Recursively sort all #'s $< A[i]$
- ↪ called "Quick-Sort"

Random divide and conquer



we will prove that QuickSort:

① takes $O(n \log n)$ time in expectation

② takes $O(n \log n)$ time w/ high probability

Expected running time of QuickSort

For each $i, j \in [n]$ w/ $i < j$,

$$X_{ij} = \begin{cases} 1 & \text{if } i\text{th smallest element is} \\ & \text{compared to } j\text{th smallest} \\ 0 & \text{otherwise} \end{cases}$$

$\sum_{i < j} X_{ij}$ = total # comparisons made
by the algorithm
 $\approx$ total running time

$E[\sum_{i < j} X_{ij}]$ = average running time

$$E[\sum_{i < j} X_{ij}] = \sum_{i < j} E[X_{ij}]$$

Linearity of Expectation
 $E[X+Y] = E[X] + E[Y]$ (extremely useful)
"average sum = sum of averages"

now analyze $E[X_{ij}]$ in isolation

$$E[X_{ij}] = \Pr[X_{ij} = 1]$$

$$= \Pr \left[\begin{array}{l} i\text{th element is} \\ \text{compared to } j\text{th} \end{array} \right]$$



i th elem is compared w/ j th
 $\Leftrightarrow$

i th or j th elem selected
before any element in between
happens w/ probability $\frac{2}{j-i+1}$

$$E[X_{ij}] = \frac{2}{j-i+1}$$

$$E[\sum_{i < j} X_{ij}] = \sum_{i < j} E[X_{ij}] \quad \text{by linearity of expectation}$$

$$= \sum_{i=1}^n \sum_{j=i+1}^n \frac{2}{j-i+1} \equiv \sum_{j=1}^n \sum_{i=1}^j \frac{2}{j-i+1}$$

$$\leq 2n \left(\frac{1}{1} + \frac{1}{2} + \dots + \frac{1}{n} \right)$$

$$\leq \underline{O(n \log n)} \quad \text{nth harmonic number}$$

Today:

Probability

Hash Functions,
Markov's Inequality

Algorithms

Frequency Estimation,
Heavy Hitters (in streams)

Explore what
United States
is searching for
right now

taylor townsend

Explore

Search interest, past 24 hours

Why is **taylor townsend** trending?

Search it



Why the absence of a common
act of tennis sportsmanship led t...
4 hours ago • CNN



Taylor Townsend snaps back at
Jelena Ostapenko after heated ...
19 hours ago • Yahoo Sports



Taylor Townsend-Jelena
Ostapenko argument: What...
14 hours ago • USA Today

Dive deeper

Explore issues and events in detail. Curated by the Trends Data Team.

Summer Trends

United States

Past 24 hours

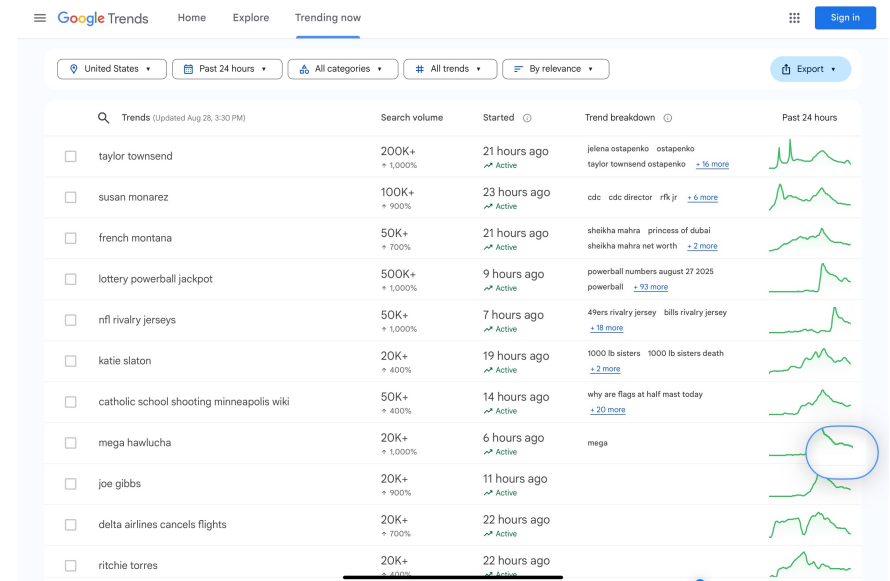
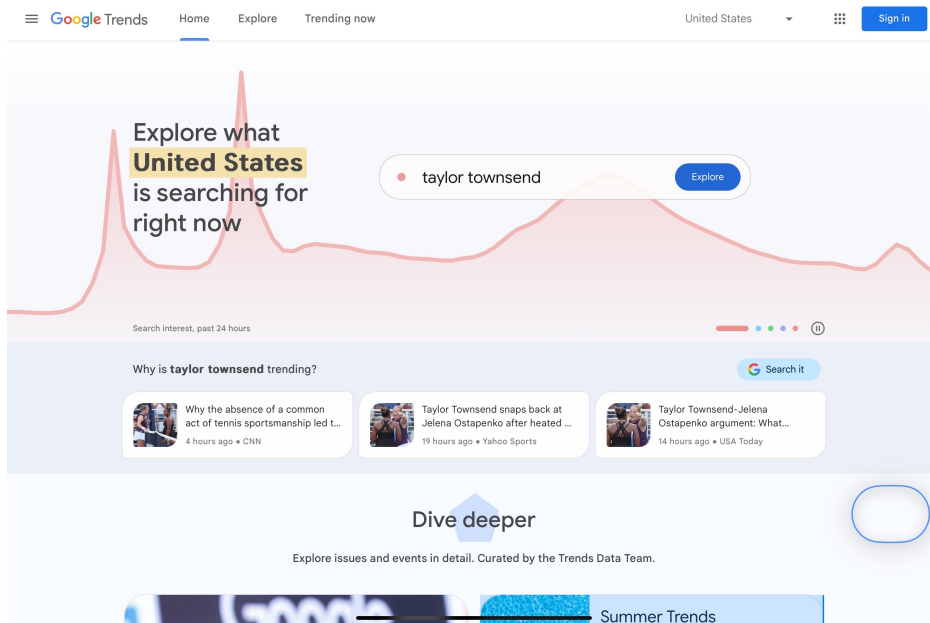
All categories

All trends

By relevance

Export

Trends (Updated Aug 28, 3:30 PM)	Search volume	Started ⓘ	Trend breakdown ⓘ	Past 24 hours
☐ taylor townsend	200K+ ↑ 1,000%	21 hours ago ↗ Active	jelena ostapenko ostapenko taylor townsend ostapenko + 16 more	
☐ susan monarez	100K+ ↑ 900%	23 hours ago ↗ Active	cdc cdc director rfk jr + 6 more	
☐ french montana	50K+ ↑ 700%	21 hours ago ↗ Active	sheikha mahra princess of dubai sheikha mahra net worth + 2 more	
☐ lottery powerball jackpot	500K+ ↑ 1,000%	9 hours ago ↗ Active	powerball numbers august 27 2025 powerball + 93 more	
☐ nfl rivalry jerseys	50K+ ↑ 1,000%	7 hours ago ↗ Active	49ers rivalry jersey bills rivalry jersey + 18 more	
☐ katie slaton	20K+ ↑ 400%	19 hours ago ↗ Active	1000 lb sisters 1000 lb sisters death + 2 more	
☐ catholic school shooting minneapolis wiki	50K+ ↑ 400%	14 hours ago ↗ Active	why are flags at half mast today + 20 more	
☐ mega hawlucha	20K+ ↑ 1,000%	6 hours ago ↗ Active	mega	
☐ joe gibbs	20K+ ↑ 900%	11 hours ago ↗ Active		
☐ delta airlines cancels flights	20K+ ↑ 700%	22 hours ago ↗ Active		
☐ ritchie torres	20K+ ↑ 400%	22 hours ago ↗ Active		



tracking most popular queries is important for
caching

But how?

Way too many different queries to
maintain counters for each

Heavy Hitters In Streams



- Elements one by one online
(can repeat)

a, b, c, b, a, d, a, e, a, b, a, \$

↑
small space

- 5% Heavy Hitters: elements that appear $\geq 5\%$ of the time
(more generally 5% $\rightarrow$ parameter $\epsilon > 0$)

Heavy Hitters In Streams

- Elements one by one online
(can repeat)



a, b, c, b, a, d, a, e, a, b, a, f

- 5% Heavy Hitters: elements that appear $\geq 5\%$ of the time

(more generally 5% $\rightarrow$ parameter $\epsilon > 0$)

Goal: identify all heavy hitters
w/ sublinear space

Saving all counts in a dictionary
takes too much space

e.g. $O(\sqrt{n})$, $O(\log n)$ where all elements are
IDs between 1 and n

- 5% Heavy Hitters: elements that appear $\geq 5\%$ of the time

(more generally $5\% \rightarrow$ parameter $\epsilon > 0$)

Goal: identify all heavy hitters
w/ sublinear space

How is this even possible?

(a) Only 20 (5%)-heavy hitters

(b) Randomization: allow small prob. of failure

(c) Approximation: allow some margin of error

rigorous, worst case bounds



Today:

Probability

Hash Functions,
Markov's Inequality

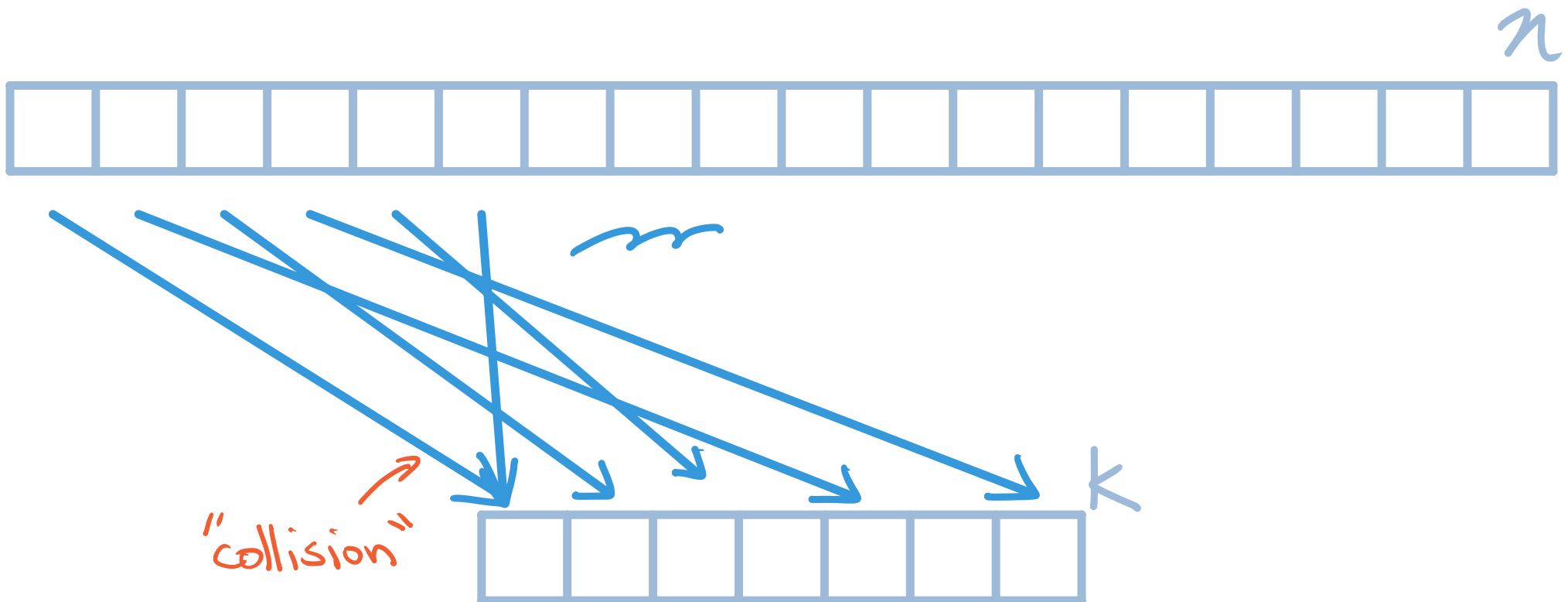
Algorithms

Frequency Estimation,
Heavy Hitters (in streams)

② Hash Functions

Randomly ^(initialized) constructed function

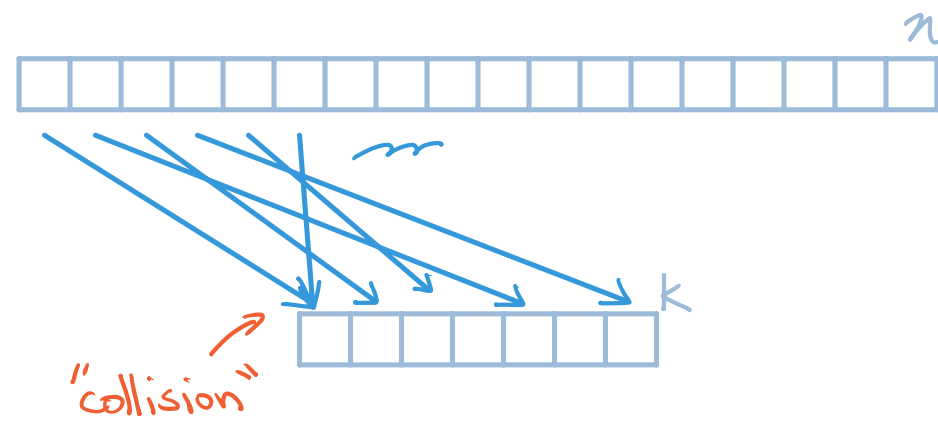
big $h: [n] \rightarrow [k]$ rel. small



② Hash Functions

Randomly constructed function

big $h: [n] \rightarrow [k]$ rel. small



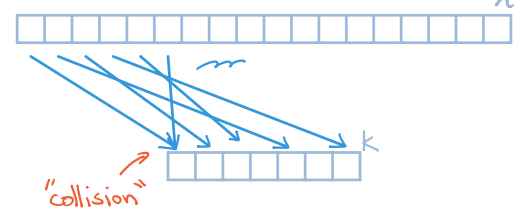
"Ideal Hash Function"

$h: [n] \rightarrow [k]$

w/ each $h(i)$ indep. unif. random in $[k]$

↑
requires $\Omega(\log(k^n)) = \Omega(n \log k)$ space $\ddot{\smile}$

Universal Hash Functions



$h: [n] \rightarrow [k]$ s.t. for all distinct $i, j \in [n]$

$$\Pr[h(i) = h(j)] \leq \frac{1}{k}$$

(much weaker than ideal hash)

e.g. $h(x) = ((ax + b) \bmod p) \bmod k$

where

- p prime, $> k$ ish
- a in $\{1, \dots, p\}$ unif. random
- b in $\{0, \dots, p-1\}$ unif. random

(proofs left as exercise)

Today:

② Randomized algorithms ("average" over random bits)

Probability

Hash Functions,

Universal Hash Functions

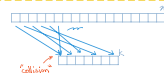
$h: [n] \rightarrow k$ s.t. for all pairs $i, j \in [n]$,
 $\Pr[h(i) = h(j)] \leq \frac{1}{k}$ (much weaker than ideal hash)

e.g. $h(x) = (ax + b) \bmod p \bmod k$

where • p prime, $> k$

• a in $\{1, \dots, p-1\}$ unif. random

• b in $\{0, \dots, p-1\}$ unif. random



Algorithms

Frequency Estimation, Heavy Hitters (in streams)

Markov's Inequality

Markov's Inequality (special case)

Let $X \geq 0$ be nonnegative RV

$$\Pr[X \geq 2E[X]] \leq \frac{1}{2}$$

e.g.

$\leq 50\%$ of US popul. earns $2 \times$ average

$\leq 50\%$ of NBA players score
 $2 \times$ league average points

$\leq 50\%$ students get
 $2 \times$ average midterm score

Markov's Inequality

Let $X \geq 0$ be nonnegative RV

$$\Pr[X \geq 2E[X]] \leq \frac{1}{2}$$

more generally, for $\alpha \geq 1$,

$$\Pr[X \geq \alpha E[X]] \leq \frac{1}{\alpha}$$

Today:

Probability

Hash Functions,

Universal Hash Functions

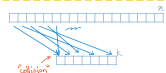
$h: [n] \rightarrow k$ s.t. for all pairs $i, j \in [n]$,
 $\Pr[h(i) = h(j)] \leq \frac{1}{k}$ (much weaker than ideal hash)

e.g. $h(x) = (ax + b) \bmod p \bmod k$

where • p prime, $> k$

• a in $\{1, \dots, p\}$ unis. random

• b in $\{0, \dots, p-1\}$ unis. random



Algorithms

Frequency Estimation, Heavy Hitters (in streams)

Markov's Inequality

Markov's Inequality (special case)

Let $X \geq 0$ be nonnegative RV

$$\Pr[X \geq 2E[X]] \leq \frac{1}{2}$$

Heavy Hitters In Streams



- Elements one by one online
(can repeat)

a, b, c, b, a, d, a, e, a, b, a, \$

↑
small space

- 5% Heavy Hitters: elements that appear $\geq 5\%$ of the time
(more generally 5% $\rightarrow$ parameter $\epsilon > 0$)

Heavy Hitters In Streams



- Elements one by one online
(can repeat)

a, b, c, b, a, d, a, e, a, b, a, f

- 5% Heavy Hitters: elements that appear $\geq 5\%$ of the time

(more generally 5% $\rightarrow$ parameter $\epsilon > 0$)

Goal: identify all heavy hitters
w/ sublinear space

Saving all counts in a dictionary
takes too much space

e.g. $O(\sqrt{n})$, $O(\log n)$ where all elements are
IDs between 1 and n

From now on:

ϵ -heavy hitters for fixed $\epsilon > 0$

Heavy Hitters In Streams

- Elements one by one online
(can repeat)

a, b, c, b, a, d, a, e, a, b, a, f

- 5% Heavy Hitters: elements that appear $\geq 5\%$ of the time
(more generally $5\% \rightarrow$ parameter $\epsilon > 0$)



Related Problem

estimate relative frequency of each element

(up to ϵ additive error)

(absolute freq x) = # times x appears in stream

$$\left(\begin{array}{c} \text{relative} \\ \text{freq. of } x \end{array} \right) = \frac{(\text{abs. freq. } x)}{\text{total length of stream}}$$

ϵ -heavy hitter $\Leftrightarrow$ relative freq $\geq \epsilon$

we will focus on estimating  up to ϵ -error

So to recap:

want to estimate all relative frequencies
up to ϵ -additive error

not enough space for all counters $\frac{\log n}{\epsilon^3}$

lucky for us:

0 is a good estimate for all but $\frac{1}{\epsilon}$ elements

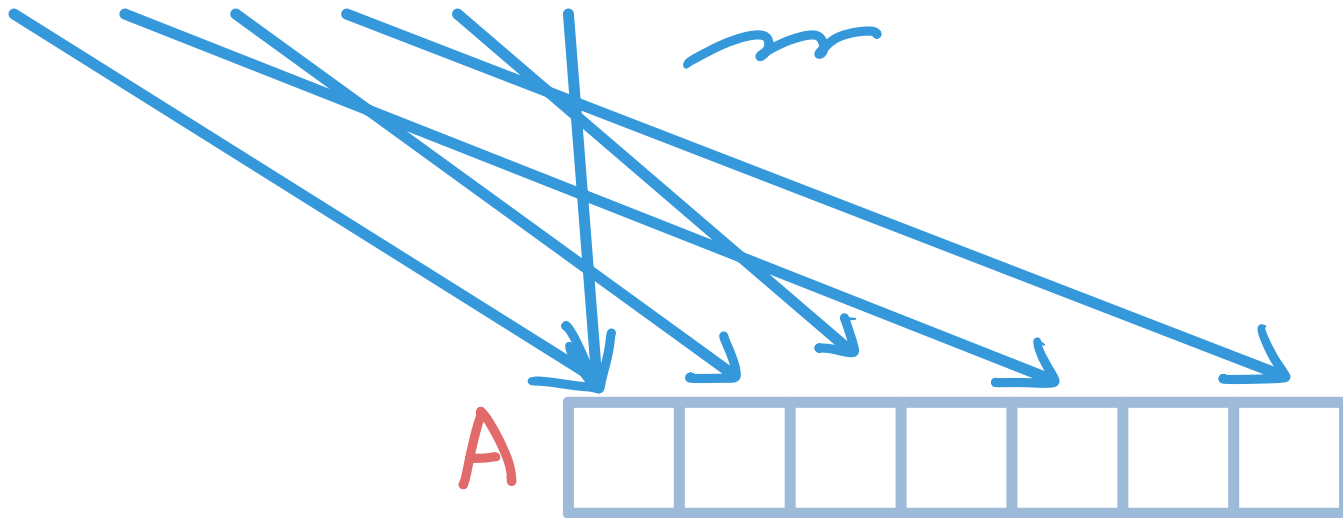
still: don't know which $\frac{1}{\epsilon}$ elements to count

Nashing has entered the chat

Idea: allocate $w = 10/\epsilon$ counters

hash element i to counter $h(i)$

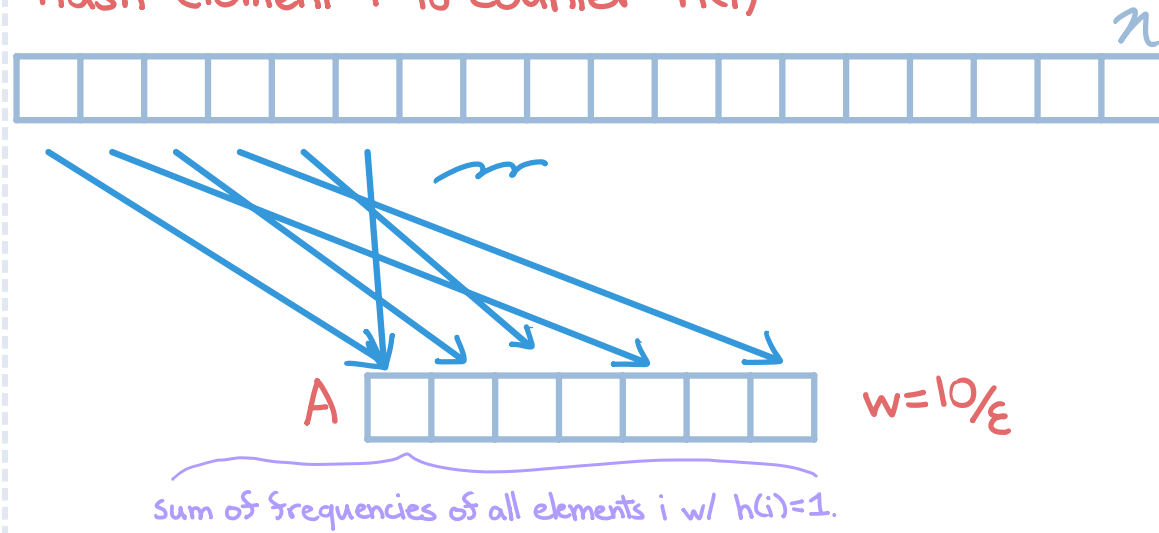
n



$w = 10/\epsilon$

sum of frequencies of all elements i w/ $h(i) = 1$.

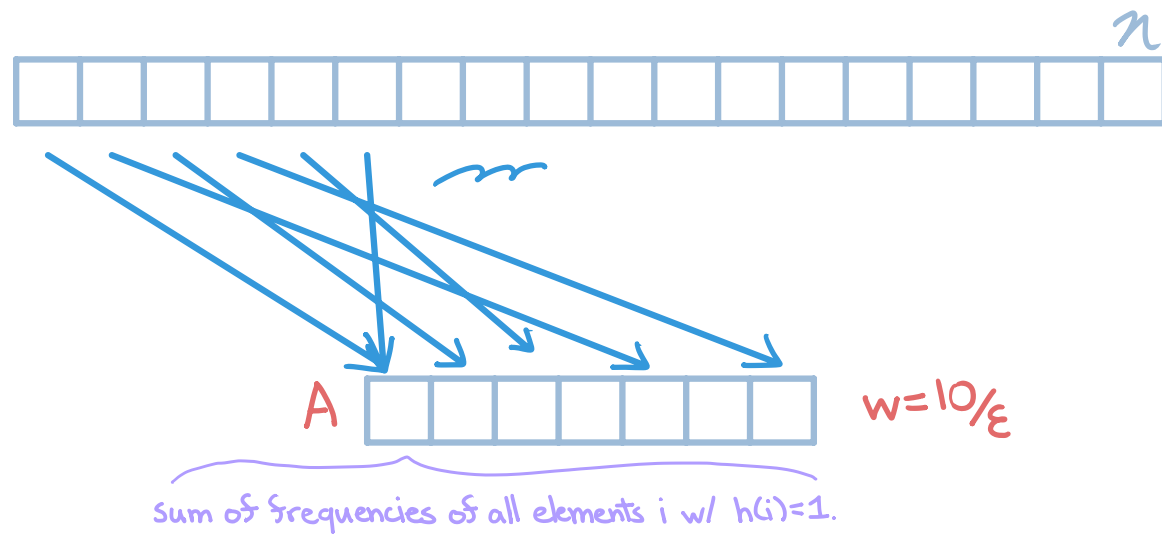
Idea: allocate $w = 10/\epsilon$ counters
hash element i to counter $h(i)$



To estimate frequency f_i of i :

return $h(i)$

(overcounts f_i)



Notation

$A[1, \dots, w]$ = array of counters

f_i = (true) frequency of i

$h: [n] \rightarrow [w]$: universal hash function

m = length of stream

e.g. $A[j] = \sum_{i: h(i)=j} f_i$

Lemma For all i ,
w/ probability $\geq \frac{1}{2}$,

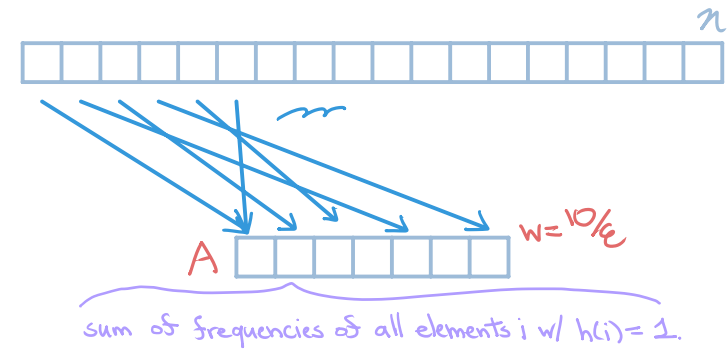
$$\underline{f_i \leq A[h(i)] \leq f_i + \frac{\epsilon}{5} m}$$

$$E[A[h(i)]] = E\left[\sum_{k: h(k)=h(i)} f_k\right]$$

$$\Rightarrow X_k = \begin{cases} f_k & \text{if } h(k)=h(i) \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} E\left[\sum_k X_k\right] &= \sum_k E[X_k] = \sum_k f_k \Pr[h(k)=h(i)] \leq \frac{1}{w} \sum_k f_k \\ &\leq f_i + \sum_{k \neq i} \frac{f_k}{w} \leq f_i + \frac{m}{w} = f_i + \frac{\epsilon m}{10} \end{aligned}$$

Universal Hash Functions
 $h: [n] \rightarrow [k]$ s.t. for distinct $i, j \in [n]$
 $\Pr[h(i)=h(j)] \leq \frac{1}{k}$



$A[1..w]$ = array of counters

f_i = (true) frequency of i

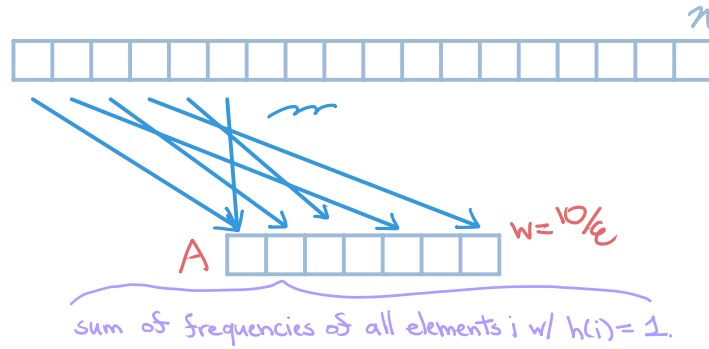
$h: [n] \rightarrow [w]$: universal hash fn.

m = length of stream

$$E[A[h(i)] - g_i] \leq \frac{\epsilon m}{10}$$

$$\text{Markov: } P\left[\sum_{i=1}^m (A[h(i)] - g_i) \geq \frac{\epsilon m}{5} = 2 \cdot E[\sum_{i=1}^m (A[h(i)] - g_i)]\right] \leq \frac{1}{2}$$

Recap:



$A[1..w]$ = array of counters

f_i = (true) frequency of i

$h: [n] \rightarrow [w]$: universal hash fn.

m = length of stream

Lemma For all i , w/ probability $\geq \frac{1}{2}$,
 $f_i \leq A[h(i)] \leq f_i + \frac{\epsilon}{5} m$

we have: for each element, good estimate
of frequency w/ constant probability

we want: estimates of all frequencies w/
high probability

we want to "amplify" our bounds



Repeatedly flip coins:

$$P[\text{first toss is tails}] = \boxed{}$$

$$P[\text{first 2 tosses are tails}] = \boxed{}$$

6

$$P[\text{first 100 tosses are tails}] = \boxed{}$$

make d copies of hashed counters

A_1, h_1

--	--	--	--	--	--	--	--

A_2, h_2

--	--	--	--	--	--	--	--

A_3, h_3

--	--	--	--	--	--	--	--

A_4, h_4

--	--	--	--	--	--	--	--

A_d, h_d

--	--	--	--	--	--	--	--

$$d \rightarrow \frac{d}{2}$$

total space
 $O(wd)$

$$\left(\frac{1}{2d}\right)^{d/d}$$

estimate S_i w/ $\min_{j=1, \dots, d} A_j[h_j(i)]$

bad
estimate

iff ALL
hashed counters
are bad

prob $\frac{1}{2d}$

set $d = 10 \log(n)$

each element is "bad" w/ prob.

$$\leq \frac{1}{2^d} = \frac{1}{n^{10}}$$

goal:

ensure all elements are correct

union bound:

$P[\text{some } e \text{ bad}]$

$$\leq \sum_e \Pr[e \text{ bad}] \leq \sum_e \frac{1}{n^{10}} = \frac{1}{n^9}$$

Heavy Hitters In Streams



- Elements one by one online
(can repeat)

$a, b, c, b, a, d, a, e, a, b, a, \dots$

- 5% Heavy Hitters: elements that appear $\geq 5\%$ of the time
(more generally $5\% \rightarrow$ parameter $\epsilon > 0$)

A_1, h_1

--	--	--	--	--	--	--	--

A_2, h_2

--	--	--	--	--	--	--	--

A_3, h_3

--	--	--	--	--	--	--	--

A_4, h_4

--	--	--	--	--	--	--	--

$\vdots$

A_d, h_d

--	--	--	--	--	--	--	--

estimate f_i

w/ $\min_{j=1, \dots, d} A_j[h_j(i)]$

bad
estimate

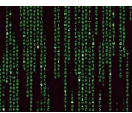
iff ALL
hashed counters
are bad
 $\uparrow$
prob $\frac{1}{2^d}$

$\Rightarrow$ all elements estimated accurately w/ prob. $\geq 1 - \frac{1}{n^9}$

Space:

$$O(wd) = O(\log(n)/\epsilon)$$

Heavy Hitters In Streams



- Elements one by one online (can repeat)

a, b, c, b, a, d, a, e, a, b, a, f

- 5% Heavy Hitters: elements that appear $\geq 5\%$ of the time

(more generally 5% $\rightarrow$ parameter $\epsilon > 0$)

A_1, h_1

A_2, h_2

A_3, h_3

A_4, h_4

$\vdots$

A_d, h_d

estimate S_i

w/ $\min_{j=1, \dots, d} A_j[h_j(i)]$

bad estimate iff ALL hashed counters are bad
 $\uparrow$
 prob $\frac{1}{2^d}$

set $d = 10 \log(n)$

each element is "bad" w/ prob.

$$\leq 2^{-d} = \frac{1}{n^{10}}$$

$$\text{Prob[any element bad]} \leq \sum_{i=1}^n \text{Pr[element } i \text{ bad]} \\ \leq n \cdot \frac{1}{n^{10}} = \frac{1}{n^9}$$

$\Rightarrow$ all elements estimated accurately w/ prob. $\geq 1 - \frac{1}{n^9}$

$\Rightarrow$ all elements estimated accurately w/ prob. $\geq 1 - \frac{1}{n^9}$

Space: $O(wd) = O(\log(n)/\epsilon)$

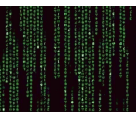
All put together

Estimate all rel. frequencies up to $\pm \epsilon$ error w/

- $O(\log n / \epsilon)$ space

- $\geq 1 - \frac{1}{n^{10}}$ prob. of success

Heavy Hitters In Streams



- Elements one by one online (can repeat)

a, b, c, b, a, d, a, e, a, b, a, f

- 5% Heavy Hitters: elements that appear $\geq 5\%$ of the time

(more generally 5% $\rightarrow$ parameter $\epsilon > 0$)

A_1, h_1

A_2, h_2

A_3, h_3

A_4, h_4

$\vdots$

A_d, h_d

estimate f_i

w/ $\min_{j=1, \dots, d} A_j[h_j(i)]$

$\nearrow$

bad estimate

iff ALL hashed counters are bad

$\nearrow$ prob $\frac{1}{2^d}$

set $d = 10 \log(n)$

each element is "bad" w/ prob.

$\leq 2^{-d} = \frac{1}{n^{10}}$

Prob[any element bad] $\leq \sum_{i=1}^n \Pr[\text{element } i \text{ bad}]$
 $\leq n \cdot \frac{1}{n^{10}} = \frac{1}{n^9}$

$\Rightarrow$ all elements estimated accurately w/ prob. $\geq 1 - \frac{1}{n^9}$

Probability

Hash Functions,

Universal Hash Functions

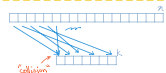
$h: [n] \rightarrow k$ s.t. for all pairs $i, j \in [n]$,
 $\Pr[h(i) = h(j)] \leq \frac{1}{k}$ (much weaker than ideal hash)

e.g. $h(x) = (ax + b) \bmod p \bmod k$

where • p prime, $> k$

• a in $\{1, \dots, p-1\}$ unif. random

• b in $\{0, \dots, p-1\}$ unif. random



Markov's Inequality

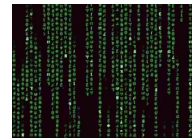
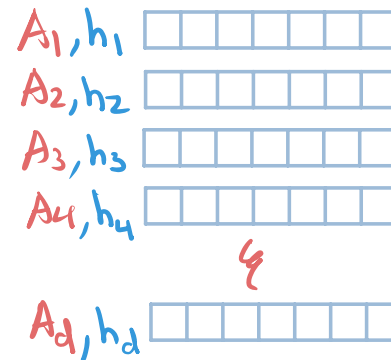
Markov's Inequality (special case)

Let $X \geq 0$ be nonnegative RV

$$\Pr[X \geq 2E[X]] \leq \frac{1}{2}$$

Algorithms

Frequency Estimation, Heavy Hitters (in streams)



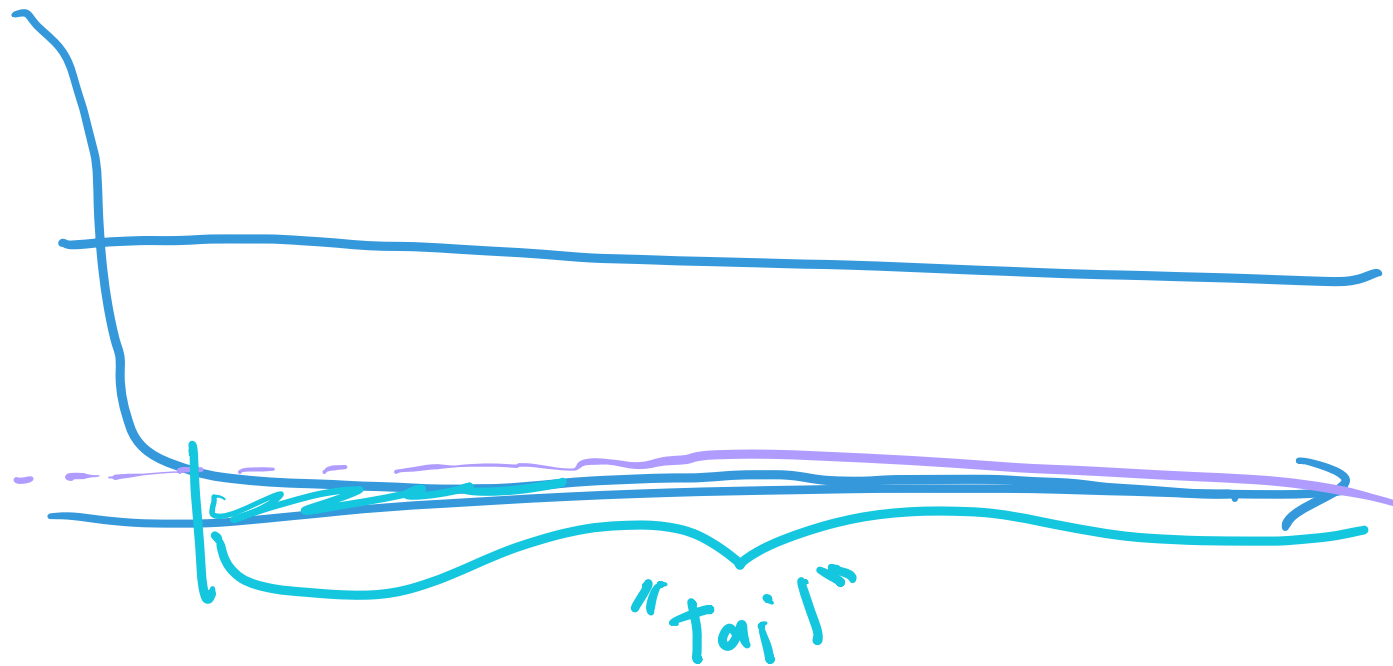
estimate f_i w/
 $\min_{j=1, \dots, d} A_j[h_j(i)]$

Heavy hitters
take first element
on element i_m :

count it $\rightarrow \tilde{f}_{i_m}$
if $\tilde{f}_{i_m} \geq \epsilon m$

goal: $\approx \epsilon$ -heavy hitters
do count-min w/ param $\frac{\epsilon}{2}$

$\Rightarrow$ list of $\frac{\epsilon}{2}$ -heavy hitters
includes all ϵ -heavy hitters



"sketching"

