Hashing and Heavy Hitters

welcome to Randomized Algorithms

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 schedule[1]

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

Last time

Probability

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

Algorithms

Approx. 3-SAT,

Quicksort,

Quickselect (HW)

3-SAT

"conjuctive normal form"

CNF formula

$$\varphi(x_1,\dots,x_n) = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_4 \vee \cdots) \wedge \cdots$$

m clauses, n variables

3 (distinct) variables per clause

The silliest APX algorithm

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

We will:

① prove silly algo is $\frac{7}{8}$-APX

② *derandomize* silly algo $\Rightarrow$ deterministic $\frac{7}{8}$-APX

Even more surprising:

"probabilistically checkable proofs"

PCP Theorem

$(\frac{7}{8}+\varepsilon)$-APX is NP-Hard $\forall\ \varepsilon > 0$

Approximating 3SAT: "Max 3-SAT"

$$\varphi(x_1,\dots,x_n) = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_4 \vee \cdots) \wedge \cdots$$

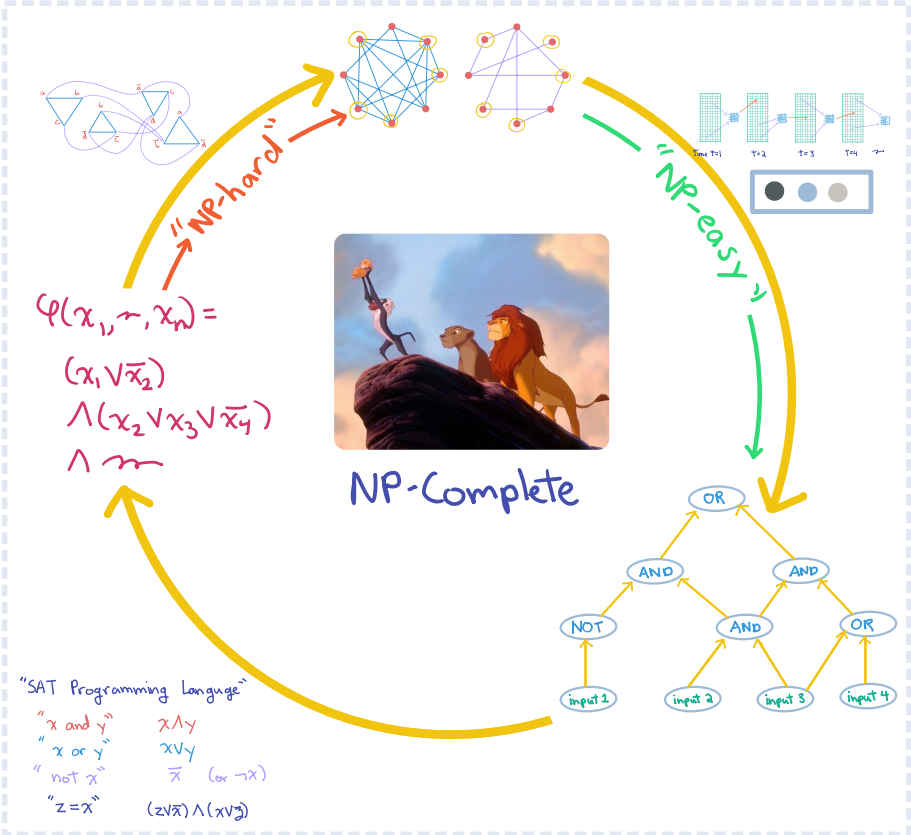
Goal: satisfy Max # clauses

Interesting even if φ is unsatisfiable

"α-Approximation algorithm" $\alpha \in (0,1)$

Algorithm satisfies $\alpha \cdot OPT$ clauses

optimum (max) value



2 disjoint clauses

① prove silly algo is $\frac{7}{8}$-APX

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

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

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 $\frac{7}{8}$-APX

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

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

E_1 = event that C_1 satisfied
E_2 = ~~~ C_2 ~~~

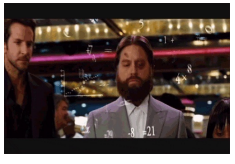
$$\Pr[E_1, E_2] = \underbrace{\Pr[E_1, E_2 \mid x_3 = T]}_{1} \cdot \frac{1}{2} + \underbrace{\Pr[E_1, E_2 \mid x_3 = F]}_{(\frac{3}{4})^2} \cdot \frac{1}{2}$$

m clauses w/ common variables

① prove silly algo is $\frac{7}{8}$-APX

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

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



E_1 = event that C_1 satisfied
E_2 = ~~~ C_2 ~~~
E_3 = ~~~ C_3 ~~~
⋮

general question

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

Big idea: we only want the average

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

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

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

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

(actually easy!)

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

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

$$E[Z] = \frac{1}{2}E[Z|x_1=T] + \frac{1}{2}E[Z|x_1=F]$$

$\Rightarrow$ either $E[Z|x_1=T]$ or $E[Z|x_1=F]$ is $\geq E[Z]$!

$E[Z|x_1=T]$ or $E[Z|x_1=F]$ is $\geq E[Z]$!

how to decide which?

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

easy to calculate

so we can compute $E[Z|x_1=T]$, $E[Z|x_2=F]$

and decide better

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

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

$E[Z] = \frac{1}{2}E[Z|x_1=T] + E[Z|x_1=F]$

induction step

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

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

for x_{k+1}:

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

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

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

one of these must be good

"method of conditional expectations"

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

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

$E[Z] = \frac{1}{2}E[Z|x_1=T] + E[Z|x_1=F]$

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

easy to calculate

Randomized Sorting

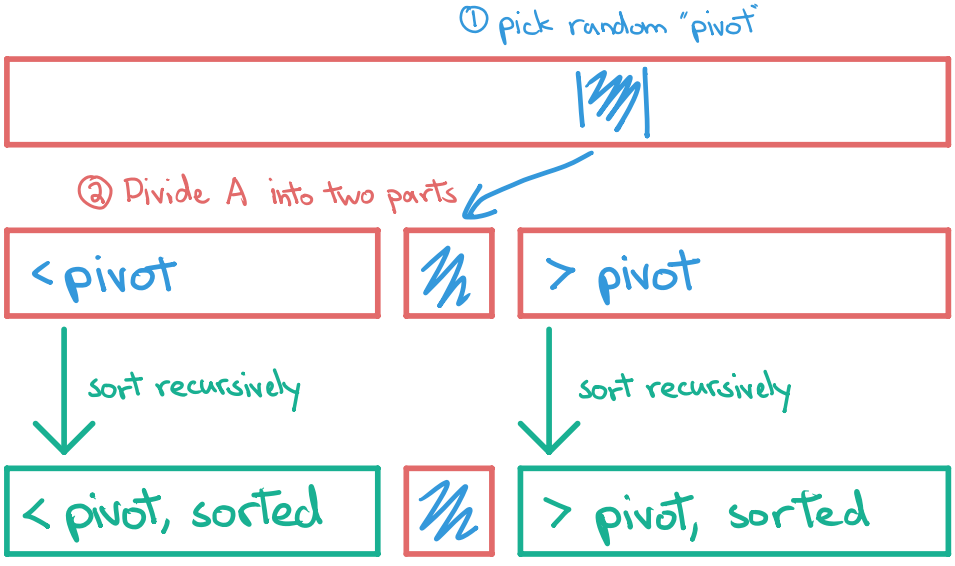
Goal: sort $A[1, \ldots, n]$

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:

ⓐ takes $O(n \log n)$ time *in expectation*

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

Expected running time of QuickSort

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

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

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

$\approx$ total running time

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

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

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

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

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

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

ith elem is compared w/ jth

$\Leftrightarrow$

ith or jth elem selected before any element in between

↑ happens w/ probability $\frac{2}{j-i+1}$

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

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

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

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

$$\leq O(n \log n)$$

nth harmonic number

Today:

Probability	Algorithms
Hash functions, Markov's Inequality	Frequency Estimation, Heavy Hitters (in streams)

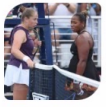
Google Trends Home Explore Trending now United States Sign in

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 ...

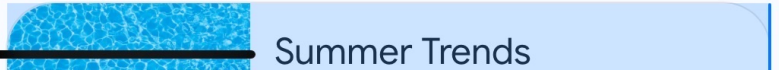
19 hours ago • Yahoo Sports

Taylor Townsend-Jelena Ostapenko argument: What...

14 hours ago • USA Today

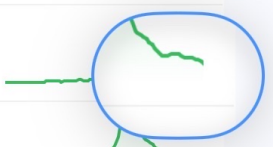
Dive deeper

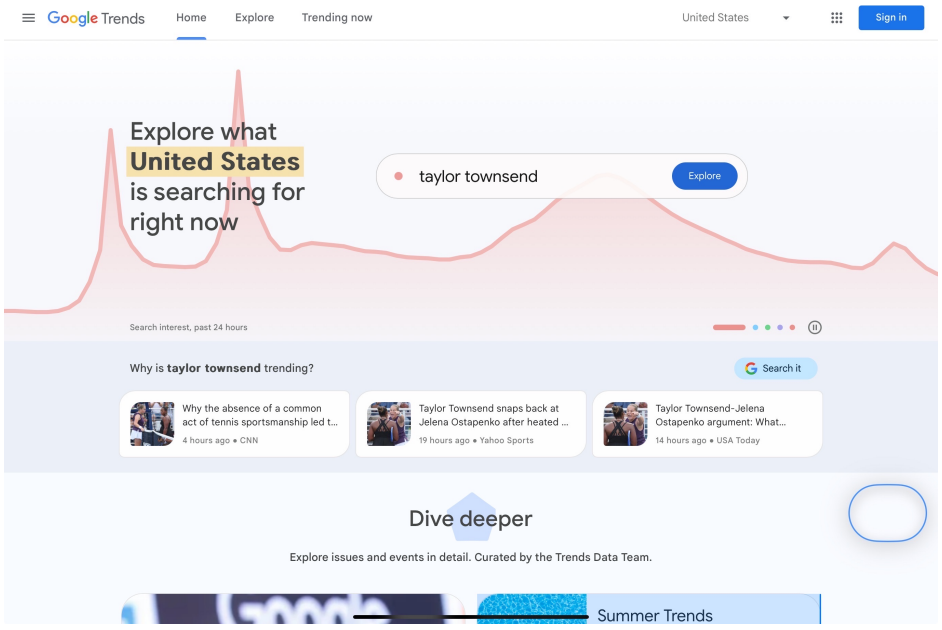
Explore issues and events in detail. Curated by the Trends Data Team.

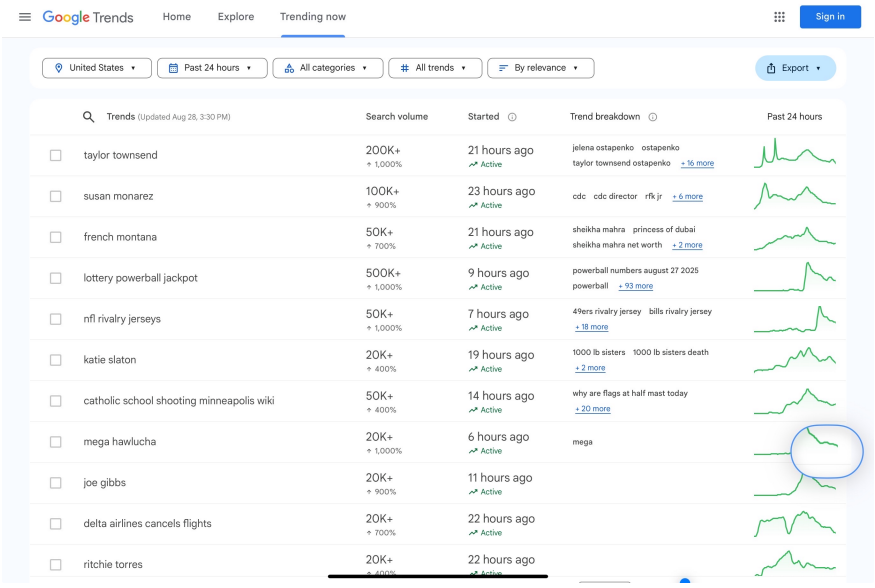


Google Trends Home Explore Trending now Sign in

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, f

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

  (more generally 5% → parameter $\varepsilon > 0$)

Heavy Hitters In Streams

- Elements one by one online (can repeat)

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

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

  (more generally 5% → parameter $\varepsilon > 0$)

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% → parameter $\varepsilon > 0$)

Goal: identify <u>all</u> 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

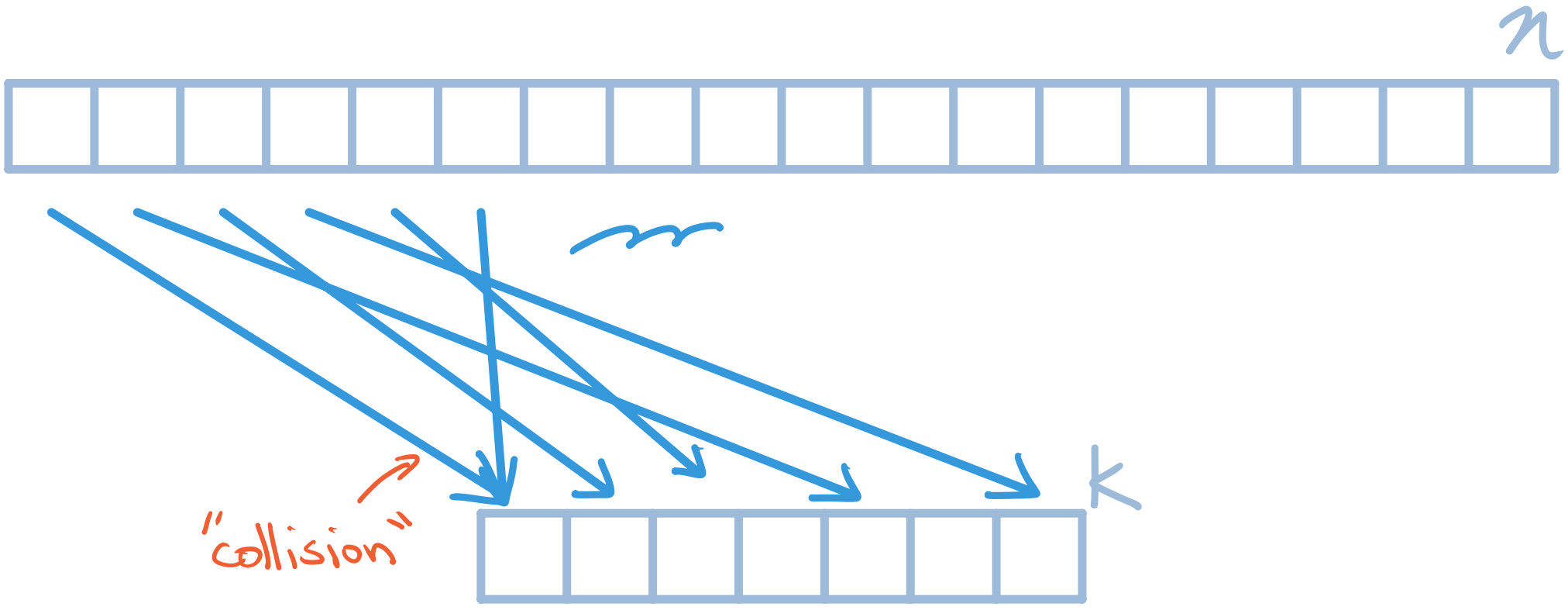
Today:

Probability	Algorithms
Hash functions, Markov's Inequality	Frequency Estimation, Heavy Hitters (in streams)

② Hash Functions

Randomly (initialized) constructed function

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

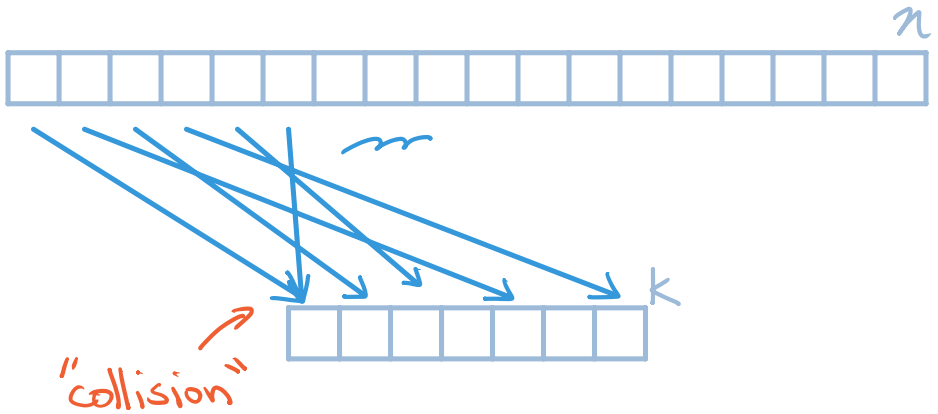


② Hash Functions

Randomly constructed function

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

big n → n; rel. small k → k



"Ideal Hash Function"

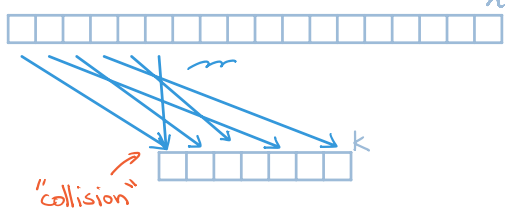
$h:[n] \to [k]$

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

↑

requires $\Omega(\log(k^n)) = \Omega(n \log k)$ space ∴

Universal Hash Functions



$h: [n] \to [k]$ 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, $> k$
- a in $\{1, \ldots, p\}$ unif. random
- b in $\{0, \ldots, p-1\}$ unif. random

(proof left as exercise)

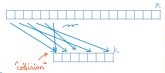
Today:

② Randomized algorithms ("average" over random bits)

Probability

Hash functions,

Universal Hash Functions



$h: [n] \to k$ 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 $\{1, \ldots, p^3\}$ unif. random
- b in $\{0, \ldots, p-1\}$ unif. random

Markov's Inequality

Algorithms

Frequency Estimation,

Heavy Hitters (in streams)

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× average

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

$\leq 50\%$ students get 2× 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 $\alpha \geq 1$,

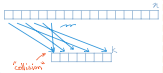
$$\Pr[X \geq \alpha E[X]] \leq \frac{1}{\alpha}$$

Today:

Probability

Hash functions,

Universal Hash Functions



$h: [n] \to k$ 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 $\{1, \ldots, p^3\}$ unif. random
- b in $\{0, \ldots, p-1\}$ 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)

Heavy Hitters In Streams

- Elements one by one online
  (can repeat)

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

small space

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

  (more generally 5% → parameter $\varepsilon > 0$)

Heavy Hitters In Streams

- Elements one by one online
  (can repeat)

  a, b, c, b, a, d, a, e, a, b, a, ...

- 5% Heavy Hitters: elements that appear ≥ 5% of the time
  (more generally 5% → parameter $\varepsilon > 0$)

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 $\varepsilon > 0$

Related Problem

Heavy Hitters In Streams

- Elements one by one online (can repeat)

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

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

  (more generally 5% → parameter $\varepsilon > 0$)

estimate relative frequency of each element

(up to ε additive error)

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

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

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

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

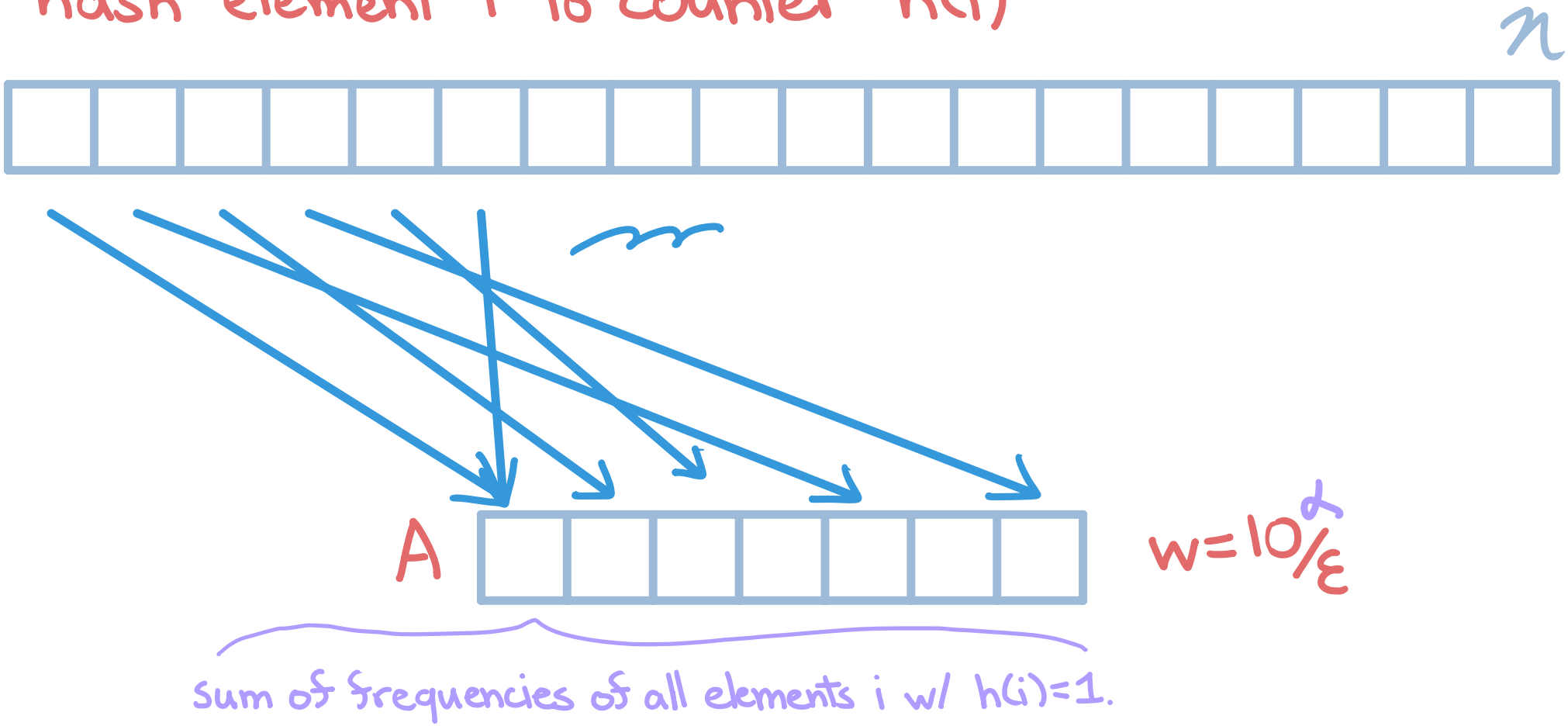
lucky for us:
0 is a good estimate for all but $\frac{1}{\varepsilon}$ elements

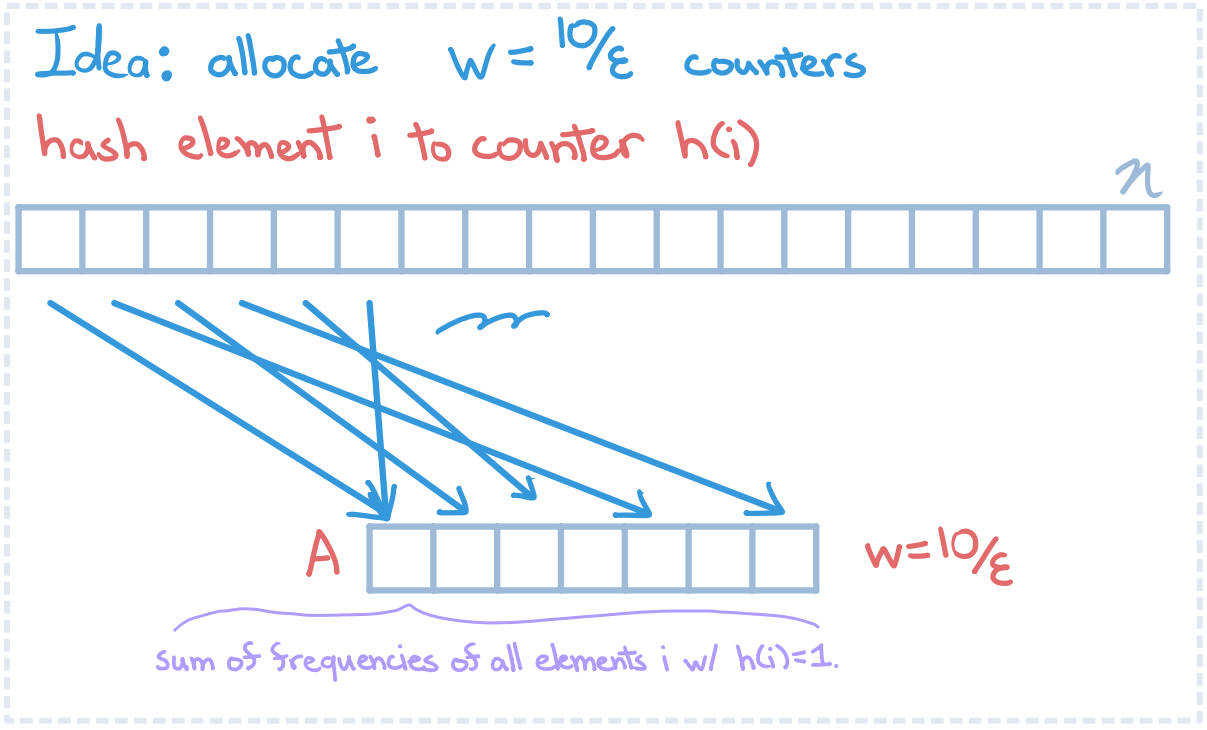
still: don't know which $\frac{1}{\varepsilon}$ elements to count

Hashing has entered the chat

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

hash element i to counter $h(i)$

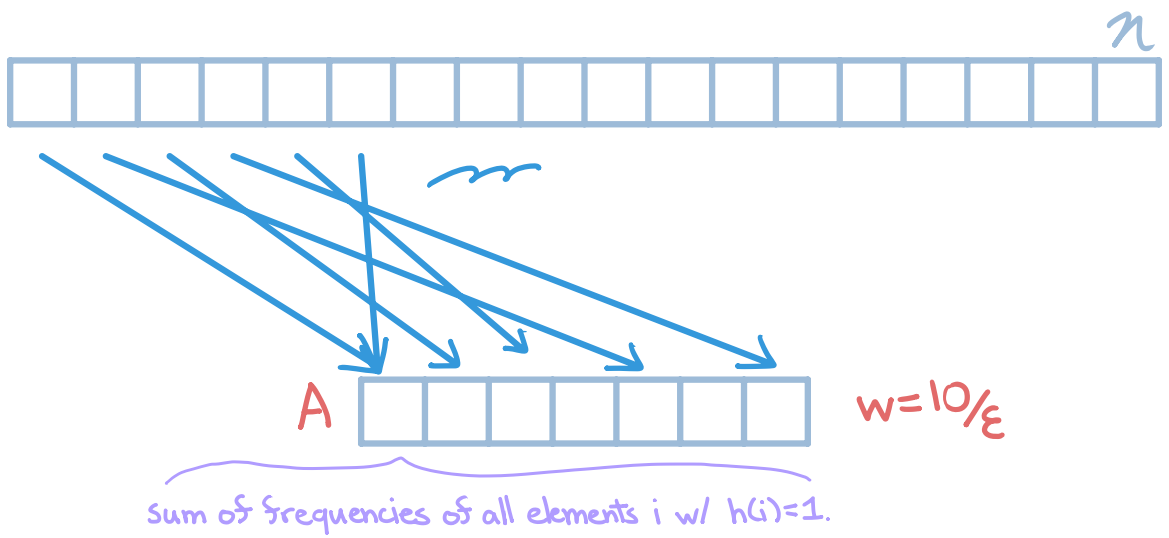




To estimate frequency f_i of i:

return h(i)

(overcounts f_i)



Notation

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

f_i = (true) frequency of i

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

m = length of stream

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

Lemma For all i,

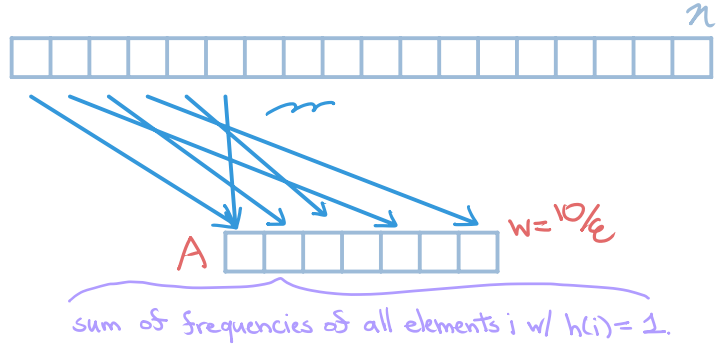
w/ probability $\geq \frac{1}{2}, \leq \frac{1}{2d}$

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

Universal Hash Functions

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

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



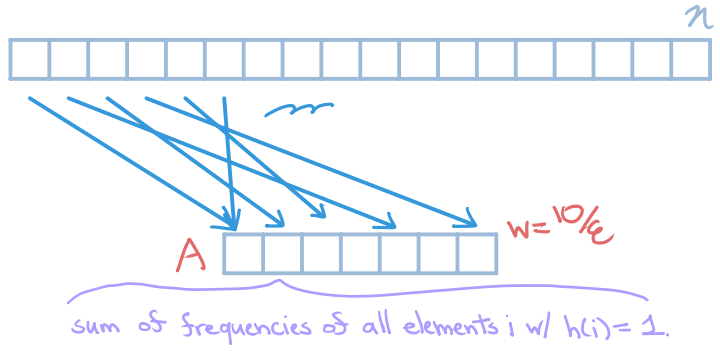
$A[1 \ldots m, w]$ = array of counters

f_i = (true) frequency of i

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

m = length of stream

Recap:



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

f_i = (true) frequency of i

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

m = length of stream

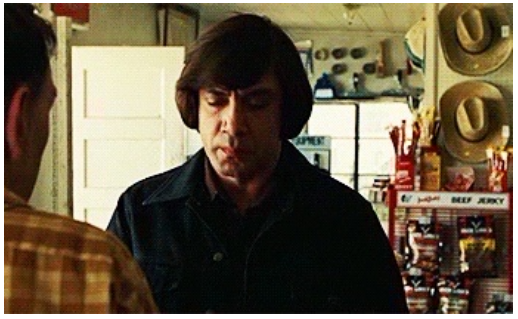
Lemma For all i, w/ probability $\geq \frac{1}{2}$,

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

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:

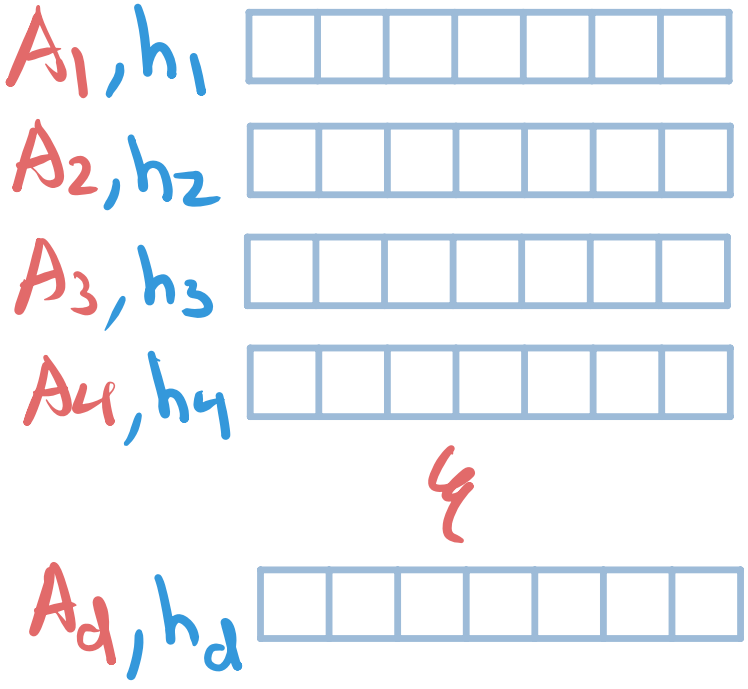
$P[\text{first toss is tails}] = 1/2$

$P[\text{first 2 tosses are tails}] = 1/4$

$\vdots$

$P[\text{first 100 tosses are tails}] = 1/2^{100}$

make d copies of hashed counters



total space
$O(wd)$

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

bad estimate iff ALL hashed counters are bad

prob $1/2^d$

set $d = 10\log(n)$

each element is "bad" w/ prob.

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

goal:

ensure <u>all</u> elements are correct

union bound:

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

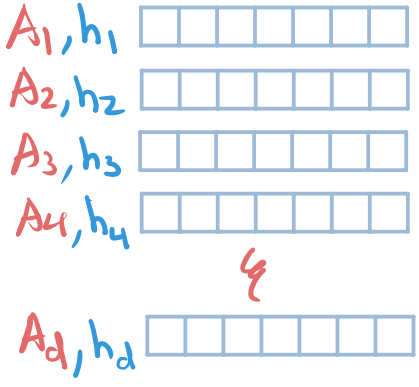
Heavy Hitters In Streams

- Elements one by one online (can repeat)

  a, b, c, b, a, d, a, e, a, b, a, f_i

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

  (more generally 5% → parameter $\varepsilon > 0$)



estimate f_i

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

bad estimate iff ALL hashed counters are bad

prob $1/2^d$

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

Space:

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

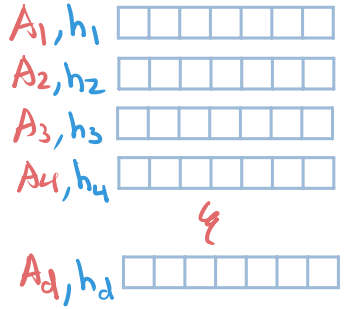
Heavy Hitters In Streams

- Elements one by one online (can repeat)

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

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

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



estimate f_i

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

bad estimate iff ALL hashed counters are bad (prob $1/2^d$)

set $d = 10 \log(n)$

each element is "bad" w/ prob. $\leq 2^d = \frac{1}{n^{10}}$

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

$$\leq n \cdot \frac{1}{n^{10}} = \frac{1}{n^9}$$

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

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

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

All put together

Estimate <u>all</u> rel. frequencies up to $\pm\varepsilon$ error w/

- $O(\log n/\varepsilon)$ space
- $\geq 1 - \frac{1}{n^{O(1)}}$ prob. of success

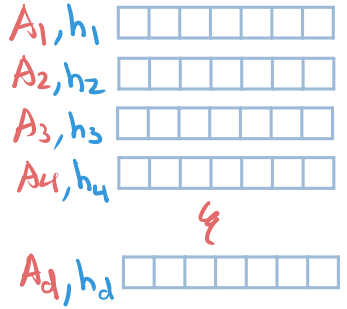
Heavy Hitters In Streams

- Elements one by one online (can repeat)

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

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

(more generally 5% → parameter $\varepsilon > 0$)



estimate f_i

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

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

set $d = 10\log(n)$

each element is "bad" w/ prob. $\leq 2^d = \frac{1}{n^{10}}$

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

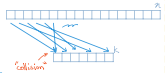
$\leq n \cdot \frac{1}{n^{10}} = \frac{1}{n^9}$

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

Probability

Hash Functions,

Universal Hash Functions



$h: [n] \to k$ 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 $\{1, \sim, p\}$ unif. random
- b in $\{0, \sim, p-1\}$ 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)