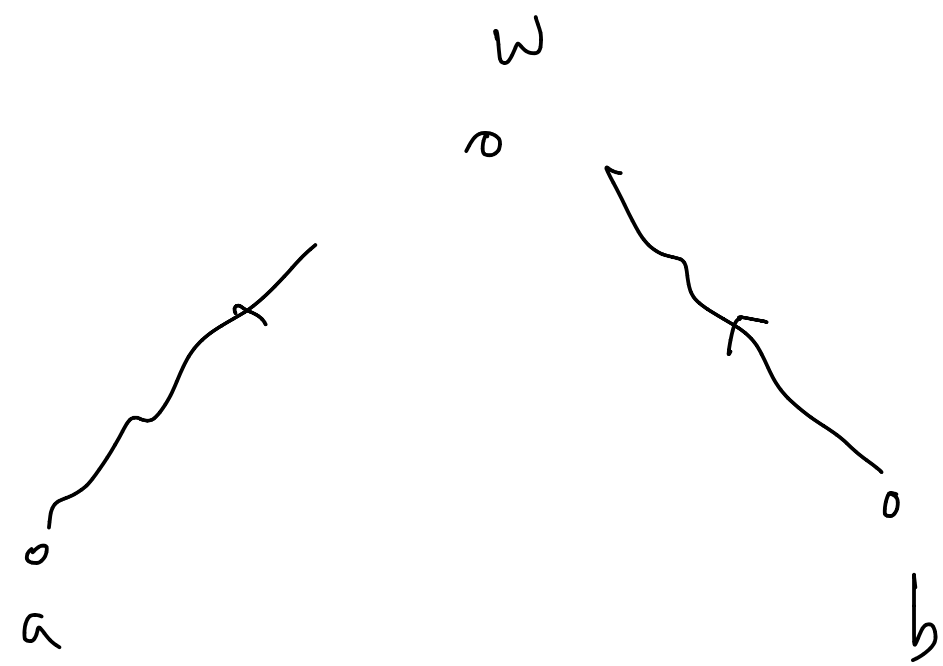
Advanced Algorithms

Lecture 19: Sampling (contd.), Optimization

Announcements

- **HW 4 due tomorrow!**

Problem 4



minimize $\max\{d(a,w),\ d(b,w)\}$

Sampling

"Confidence interval":

X_i : value of the i'th sampled element.

Problem: let A be an array with n elements, each in interval [0,1]. Find the average of all elements.

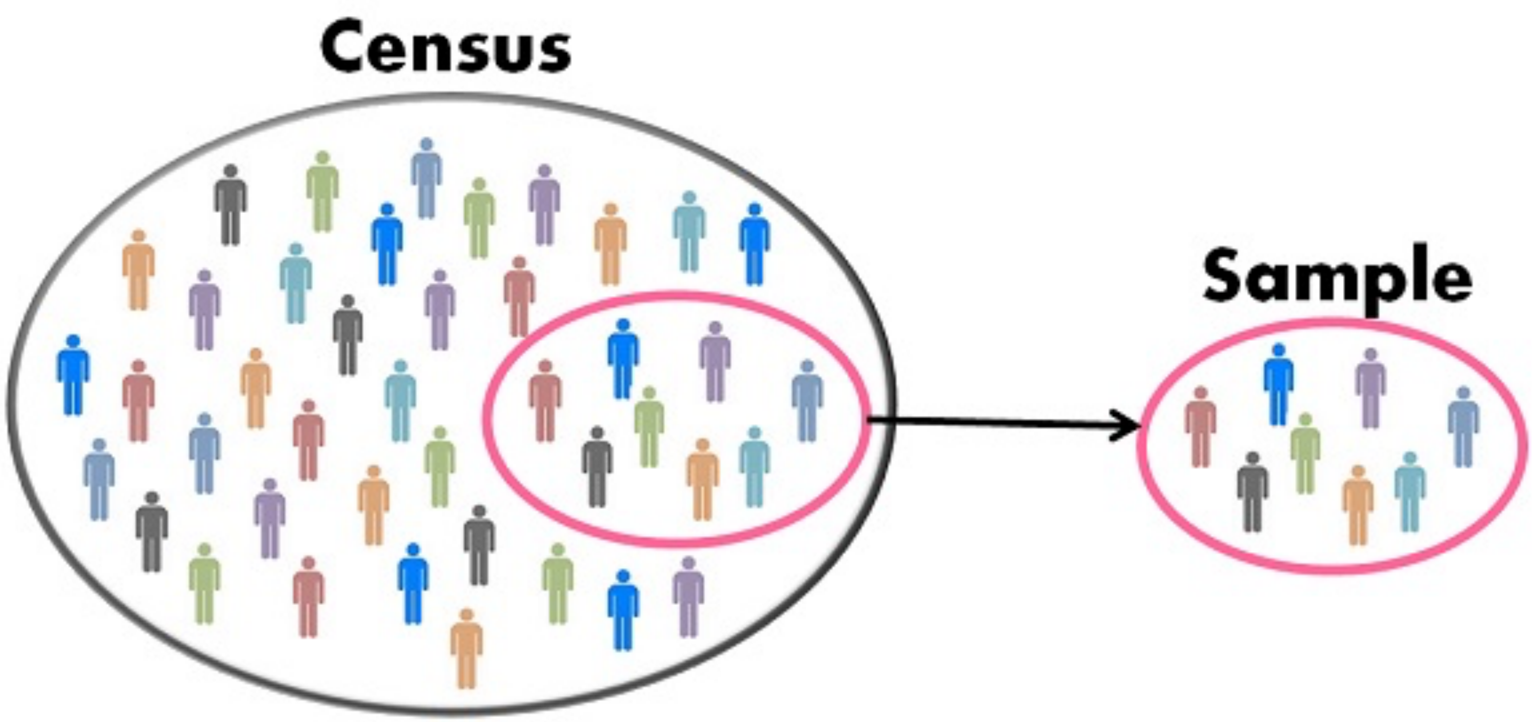
- Algorithm: take k samples and return sample mean
- Chebychev's inequality: prob [error > t/\sqrt k] <= 1/t² $\frac{t}{\sqrt{k}}$ for any $t > 1$
- Exercise: test that the error in estimation *truly* around 1/\sqrt k
- Central limit theorem

(with decent confidence)

Moral: getting error z requires roughly 1/z² samples

(0.01) 10^4

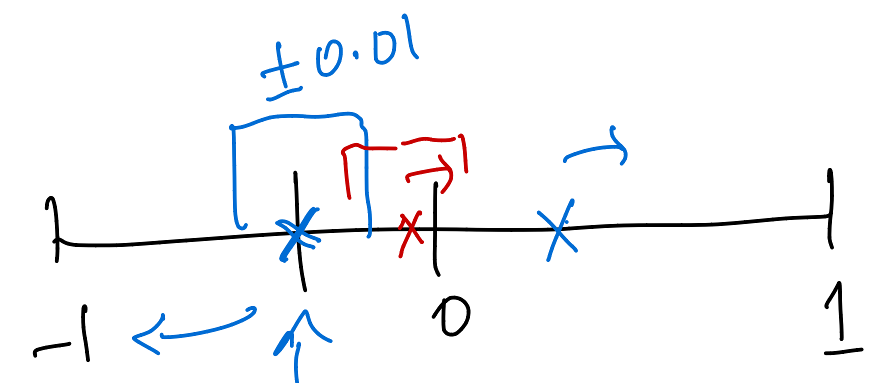
Applications of sampling



Problem: predicting an election; say everyone votes *R* or *B* and majority wins

"Reduction" to avg-finding

Associate R: −1; B: +1



p_1 p_2 - - - - p_N

→ 1 −1 1 1 −1 . - -

R is majority ⟺ avg. value < 0

B " " > 0.

average = $\frac{\#B - \#R}{N} = \frac{\sum_{i=1}^{N} p_i}{N}$

of the elements in array.

Result from last lecture:

To get result with confidence $\frac{3}{4}$, & error ε, we need $\sim \frac{4}{\varepsilon^2}$ samples.

↓

When is this good enough?

$(\varepsilon = 0.01)$. ↳ only if the "true" average $\frac{\#B - \#R}{N}$ is NOT in $(-\varepsilon, \varepsilon)$

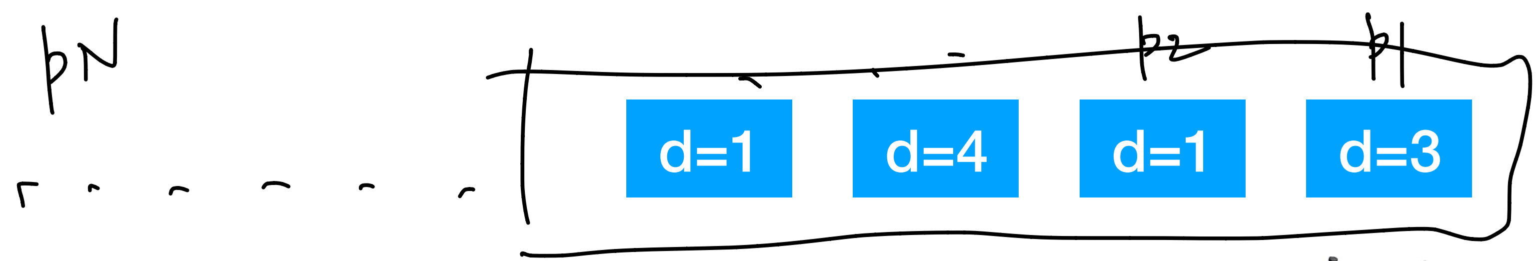
Trade-offs

1, 3, 4, 1, 7, 2

- Number of samples (k)
- Error in result (+/- “true average”)
- Confidence in result (error bound holds w.p. …)
- How close is the margin in the **true population**?

Streaming algorithms

Suppose we have data arriving one-element-at-a-time, and our goal is to find number of "distinct elements"



- Suppose destinations range from 1, …, 2^{32}
- We are OK with multiplicative error (factor 2, say)

true ans: m

$\left(\frac{m}{2}, 2m\right)$

Store all destinations in a hash table.

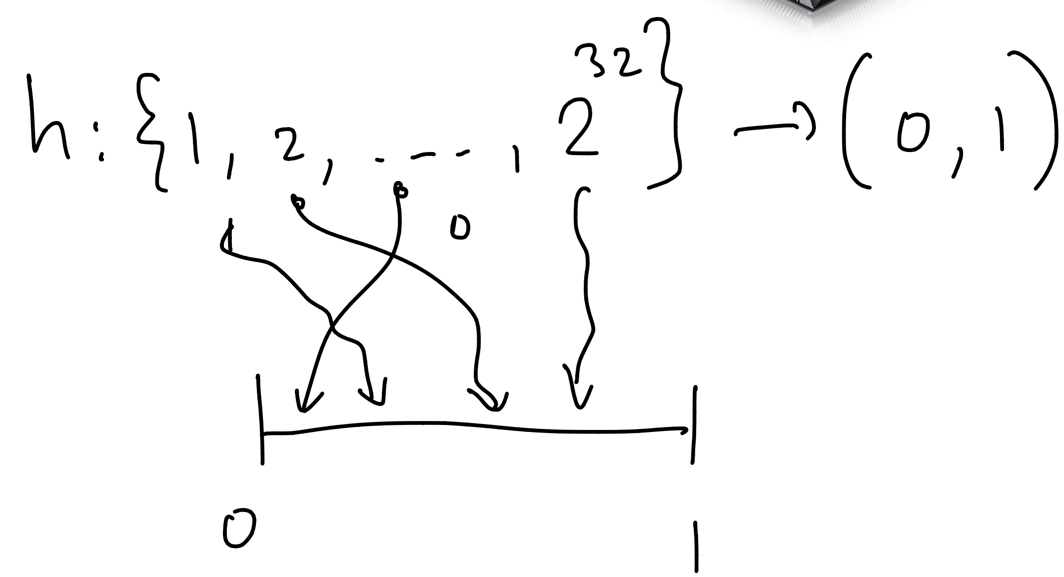
amount of memory needed ≈ # distinct destinations.

Streaming algorithms

- Hash function "h" from $\{1, \ldots, 2^{32}\}$ to (0,1)
- **Algorithm:**

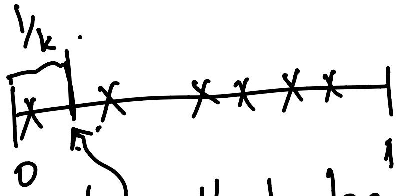
  - whenever p_i arrives: compute $h(p_i)$
  - denote by x the smallest value of $h(p_i)$ so far.

  in the end,
  - return $1/x$.



h: maps each j to a random x in (0, 1).

Qn: Suppose we have k random real #s in the interval $(0,1)$.



What do you expect the <u>smallest</u> # to be?

Obs: $h(j)$ is basically a random real # in the interval $(0,1)$

$p_1, p_2, \ldots, p_N \rightsquigarrow m$ distinct ones.

$h(p_1), h(p_2), \ldots, h(p_N) \longrightarrow m$ random #s in $(0,1)$.

X in the alg $\approx \frac{1}{m}$.

Expected value

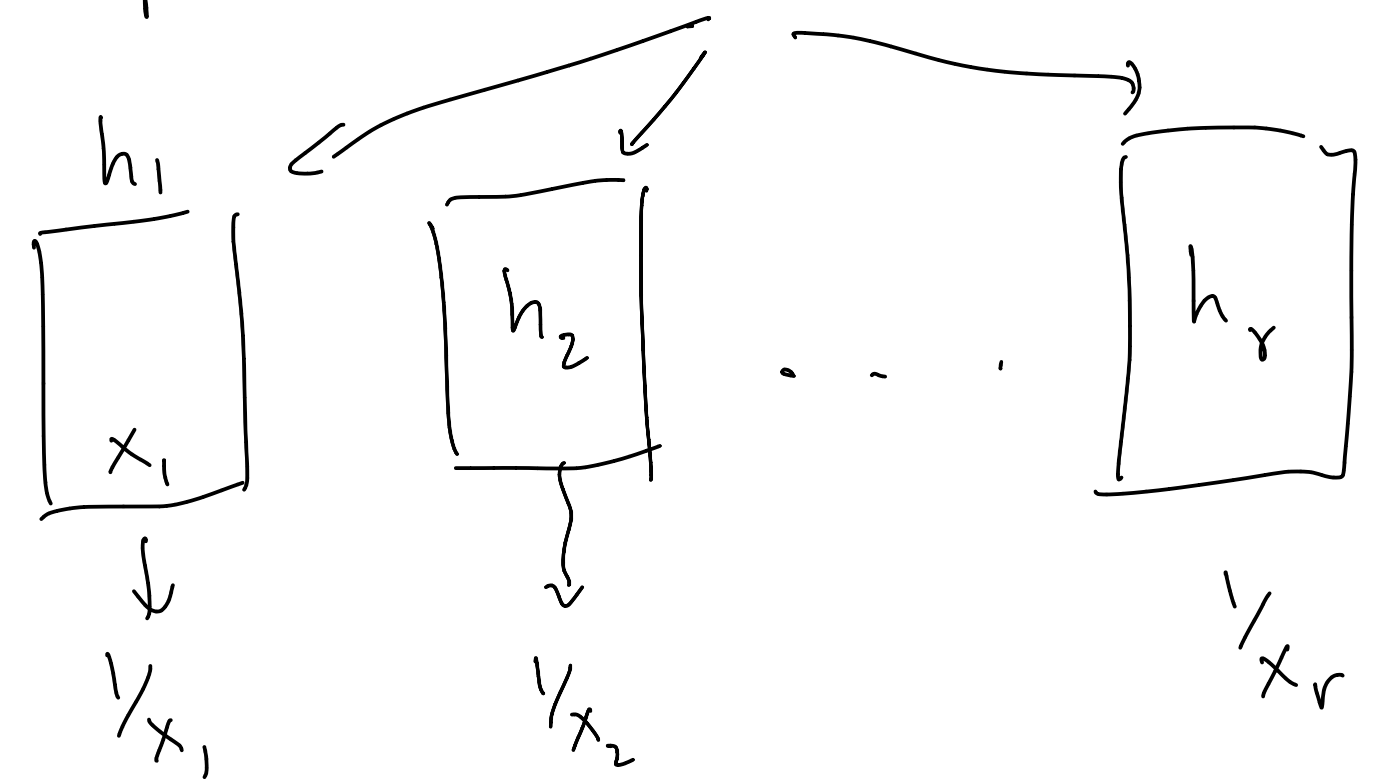
- Output $\frac{1}{X}$. $\mathbb{E}[X] = \frac{1}{m}$ → true answer.

$\mathbb{E}\left[\frac{1}{X}\right] = m$ ✗.

$\Pr\left[\min < \frac{1}{10k}\right] \approx \frac{1}{10}$.

Boosting probability

- Keep track of r hash functions.



$r/10$ of them will be $\approx 10m$

Median of these answers is a **very good estimate**

“Power of randomness”

.14152 . . .

- Randomness often helps under “resource constraints”

- Sub-linear algorithms (not looking at or being able to store full input) — still obtain good estimates

- Big caveat: not clear how to generate random numbers! can often take a lot of time

- Complexity question: don’t know if randomness helps solve problems “significantly faster”

$$P \stackrel{?}{=} RP$$

Optimization formulations

Optimization?

- Variables in a domain
- Objective
- Constraints

Classic examples

- Linear programming

- Convex optimization

Optimization for “discrete” problems

- Variables in a domain
- Objective
- Constraints

Phrasing problems as opt

- Matching?
- Shortest path

Motivations, plan

- Why useful?
- Complexity issues