

# COMPSCI 514: Algorithms for Data Science

---

Cameron Musco

University of Massachusetts Amherst. Fall 2026.

Lecture 2

## Reminders:

- Remember to sign up for Piazza.
- Find homework teammates (see Piazza Post) and sign up for Gradescope (code on course website).
- Week 1 Quiz will be available after class and is due **Monday at 8:00pm.**
- Let me know on Piazza if you see any issues with the quiz.
- Echo360 is recording lecture but not capturing the slides. Campus IT is working to resolve.

# Overview

## Last Class:

- Basic probability review. See course site for links to resources to refresh your probability background.
- Linearity of expectation and variance.

## Today:

- Algorithmic applications of linearity of expectation and variance.
- Introduce Markov's inequality a fundamental **concentration bound** that let us prove that a random variable lies close to its expectation with good probability.
- Learn about random hash functions, which are a key tool in randomized methods for data processing. Probabilistic analysis via linearity of expectation.

# Linearity of Expectation and Variance

**Linearity of Expectation:**  $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$  for *any* random variables  $X$  and  $Y$ . Proof via definition of expectation.

**Linearity of Variance:**  $\text{Var}[X + Y] = \text{Var}[X] + \text{Var}[Y]$  when  $X$  and  $Y$  are uncorrelated, and in particular, when they are independent.

Proof using two simple claims:

**Claim 1:**  $\text{Var}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$  (via linearity of expectation).

**Claim 2:**  $\mathbb{E}[XY] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$  (i.e.,  $X$  and  $Y$  are uncorrelated) when  $X, Y$  are independent (via definition of independence).

# An Algorithmic Application

You have contracted with a new company to provide CAPTCHAS for your website.



- They claim that they have a database of 1,000,000 unique CAPTCHAS. A random one is chosen for each security check.
- You want to independently verify this claimed database size.
- You could make test checks until you see 1,000,000 unique CAPTCHAS: would take  $\geq 1,000,000$  checks!

# An Algorithmic Application

**An Idea:** You run some test security checks and see if any **duplicate CAPTCHAS** show up. If you're seeing duplicates after not too many checks, the database size is probably not too big.

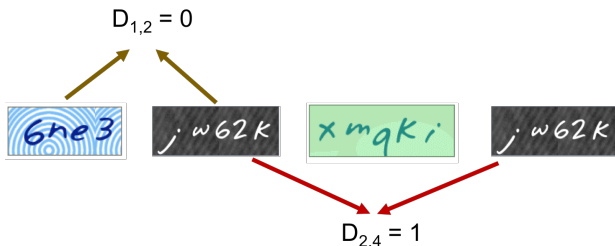


**Think-Pair-Share:** If you run  $m$  security checks, and there are  $n$  unique CAPTCHAS, how many pairwise duplicates do you see in expectation?

If e.g. the same CAPTCHA shows up three times, on your  $i^{\text{th}}$ ,  $j^{\text{th}}$ , and  $k^{\text{th}}$  test, this is three duplicates:  $(i, j)$ ,  $(i, k)$  and  $(j, k)$ .

# Linearity of Expectation

Let  $D_{i,j} = 1$  if tests  $i$  and  $j$  give the same CAPTCHA, and 0 otherwise. An **indicator random variable**.



The number of pairwise duplicates (a random variable) is:

$$D = \sum_{i,j \in [m], i < j} D_{i,j}. \mathbb{E}[D] = \sum_{i,j \in [m], i < j} \mathbb{E}[D_{i,j}].$$

For any pair  $i, j \in [m], i < j$ :  $\mathbb{E}[D_{i,j}] = \Pr[D_{i,j} = 1] = \frac{1}{n}$ .

$$\mathbb{E}[D] = \sum_{i,j \in [m], i < j} 1 \cdot \binom{m}{2} = \frac{m(m-1)}{2}$$

## Connection to the Birthday Paradox



If there are a 60 people in this room, each whose birthday we assume to be a uniformly random day of the 365 days in the year, how many pairwise duplicate birthdays do we expect there are?

$$\mathbb{E}[D] = \frac{m(m-1)}{2n} = \frac{60 \cdot 59}{2 \cdot 365} \approx 5.$$

# Linearity of Expectation

You take  $m = 1000$  samples. If the database size is as claimed ( $n = 1,000,000$ ) then expected number of duplicates is:

$$\mathbb{E}[\mathbf{D}] = \frac{m(m-1)}{2n} = .4995$$

You see **10 pairwise duplicates** and suspect that something is up. But how confident can you be in your test?

**Concentration Inequalities:** Bounds on the probability that a random variable deviates a certain distance from its mean.

- Useful in understanding how statistical tests perform, the behavior of randomized algorithms, the behavior of data drawn from different distributions, etc.

$n$ : number of CAPTCHAS in database,  $m$ : number of random CAPTCHAS drawn to check database size,  $\mathbf{D}$ : number of pairwise duplicates in  $m$  random CAPTCHAS.

# Markov's Inequality

The most fundamental concentration bound: **Markov's inequality**.

For any **non-negative** random variable  $X$  and any  $t > 0$ :

$$\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}.$$

**Proof:**

$$\begin{aligned}\mathbb{E}[X] &= \sum_s \Pr(X = s) \cdot s \geq \sum_{s \geq t} \Pr(X = s) \cdot s \\ &\geq \sum_{s \geq t} \Pr(X = s) \cdot t \\ &= t \cdot \Pr(X \geq t).\end{aligned}$$

Useful form:  $\Pr[X \geq t \cdot \mathbb{E}[X]] \leq \frac{1}{t}$ .

The larger the deviation  $t$ , the smaller the probability.

## Back to Our Application

Expected number of duplicate CAPTCHAS:

$$\mathbb{E}[D] = \frac{m(m-1)}{2n} = .4995.$$

You see  $D = 10$  duplicates.

Applying Markov's inequality, if the real database size is  $n = 1,000,000$  the probability of this happening is:

$$\Pr[D \geq 10] \leq \frac{\mathbb{E}[D]}{10} = \frac{.4995}{10} \approx .05$$

This is pretty small – you feel pretty sure the number of unique CAPTCHAS is much less than 1,000,000. But how can you boost your confidence? **We'll discuss in the next few classes.**

$n$ : number of CAPTCHAS in database ( $n = 1,000,000$  claimed),  $m$ : number of random CAPTCHAS drawn to check database size ( $m = 1000$  in this example),  
 $D$ : number of pairwise duplicates in  $m$  random CAPTCHAS.

# Hash Tables

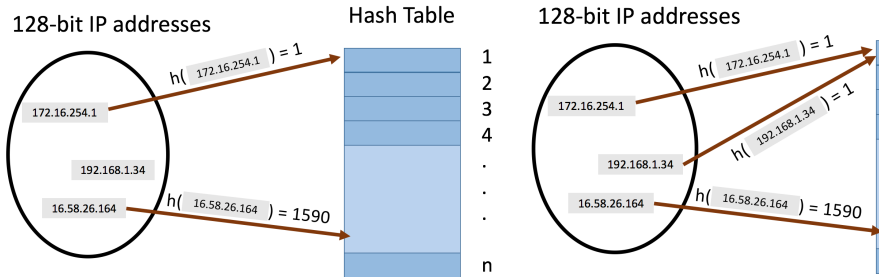
Want to store a set of items from some finite but massive universe  $U$  of items (e.g., images of a certain size, text documents, 128-bit IP addresses).

**Goal:** support  $query(x)$  to check if  $x$  is in the set in  $O(1)$  time.

**Classic Solution:** Hash tables

- *Static hashing* since we won't worry about insertion and deletion today.

# Hash Tables



- **hash function**  $h : U \rightarrow [n]$  maps elements from the universe to indices  $1, \dots, n$  of an array.
- Typically  $|U| \gg n$ . Many elements map to the same index.
- **Collisions:** when we insert  $m$  items into the hash table we may have to store multiple items in the same location (typically as a linked list).

# Collisions

**Query runtime:**  $O(c)$  when the maximum number of collisions in a table entry is  $c$  (i.e., must traverse a linked list of size  $c$ ).



## How Can We Bound $c$ ?

- In the worst case could have  $c = m$  (all items hash to the same location).
- To avoid this, we'll assume the hash function is random, and so this event is very unlikely.

# Random Hash Function

Let  $h : U \rightarrow [n]$  be a **fully random hash function**.

- I.e., for  $x \in U$ ,  $\Pr(h(x) = i) = \frac{1}{n}$  for all  $i = 1, \dots, n$  and  $h(x), h(y)$  are independent for any two items  $x \neq y$ .
- **Caveat 1:** It is *very expensive* to represent and compute such a random function. We will later see how a hash function computable in  $O(1)$  time function can be used instead.
- **Caveat 2:** In practice, often suffices to use hash functions like MD5, SHA-2, etc. that ‘look random enough’.

**Think-Pair-Share:** Assuming we insert  $m$  elements into a hash table of size  $n$  using a fully random hash function, what is the expected total number of pairwise collisions?

# Linearity of Expectation

Let  $C_{i,j} = 1$  if items  $i$  and  $j$  collide ( $h(x_i) = h(x_j)$ ), and 0 otherwise. The number of pairwise duplicates is:

$$C = \sum_{i,j \in [m], i < j} C_{i,j} \cdot \mathbb{E}[C] = \sum_{i,j \in [m], i < j} \mathbb{E}[C_{i,j}].$$

(linearity of expectation)

For any pair  $i, j, i < j$ :

$$\mathbb{E}[C_{i,j}] = \Pr[C_{i,j} = 1] = \Pr[h(x_i) = h(x_j)] = \frac{1}{n}.$$

$$\mathbb{E}[C] = \sum_{i,j \in [m], i < j} \frac{1}{n} = \frac{\binom{m}{2}}{n} = \frac{m(m-1)}{2n}.$$

Identical to the CAPTCHA analysis!

$x_i, x_j$ : pair of stored items,  $m$ : total number of stored items,  $n$ : hash table size,  $C$ : total pairwise collisions in table,  $h$ : random hash function.

# Collision Free Hashing

$$\mathbb{E}[C] = \frac{m(m-1)}{2n}.$$

- Say we have a lot of space. In particular, let  $n = 4m^2$ . Then:  
 $\mathbb{E}[C] = \frac{m(m-1)}{8m^2} \leq \frac{1}{8}$ .
- **Think-Pair-Share:** What is an upper bound on the probability that we have any collisions, i.e.,  $\Pr[C \geq 1]$ ?

Apply Markov's Inequality:  $\Pr[C \geq 1] \leq \frac{\mathbb{E}[C]}{1} = \frac{1}{8}$ .

So with probability at least  $7/8$  we have no collisions and worst-case  $O(1)$  query time.

Pretty good...but we are using  $O(m^2)$  space to store  $m$  items...

$m$ : total number of stored items,  $n$ : hash table size,  $C$ : total pairwise collisions.

Questions?