# COMPSCI 514: Algorithms for Data Science

Cameron Musco

University of Massachusetts Amherst. Fall 2022.

Lecture 3

# Logistics

- Problem Set 1 has been posted on the course website and is due **Friday 9/23 at 11:59pm**.
- I have to end my office hours at 3pm today. I will add office hours from **11am-12pm next Tuesday 9/20** to compensate.
- We generally don't give extensions on quizzes, since we discuss solutions in class on Tuesday. To make up for this, we drop the lowest quiz grade at the end of the semester.
- On the quiz feedback question, several people asked for more practice questions/examples. Check out the MIT and Khan academy material posted on the Schedule. We will also keep having probability practice questions on the first few quizzes.
- It is common to not catch everything in lecture. I strongly encourage going back to the slides to review/check your understanding after class. Also come to office hours for more in-depth discussion/examples.

# Content Overview

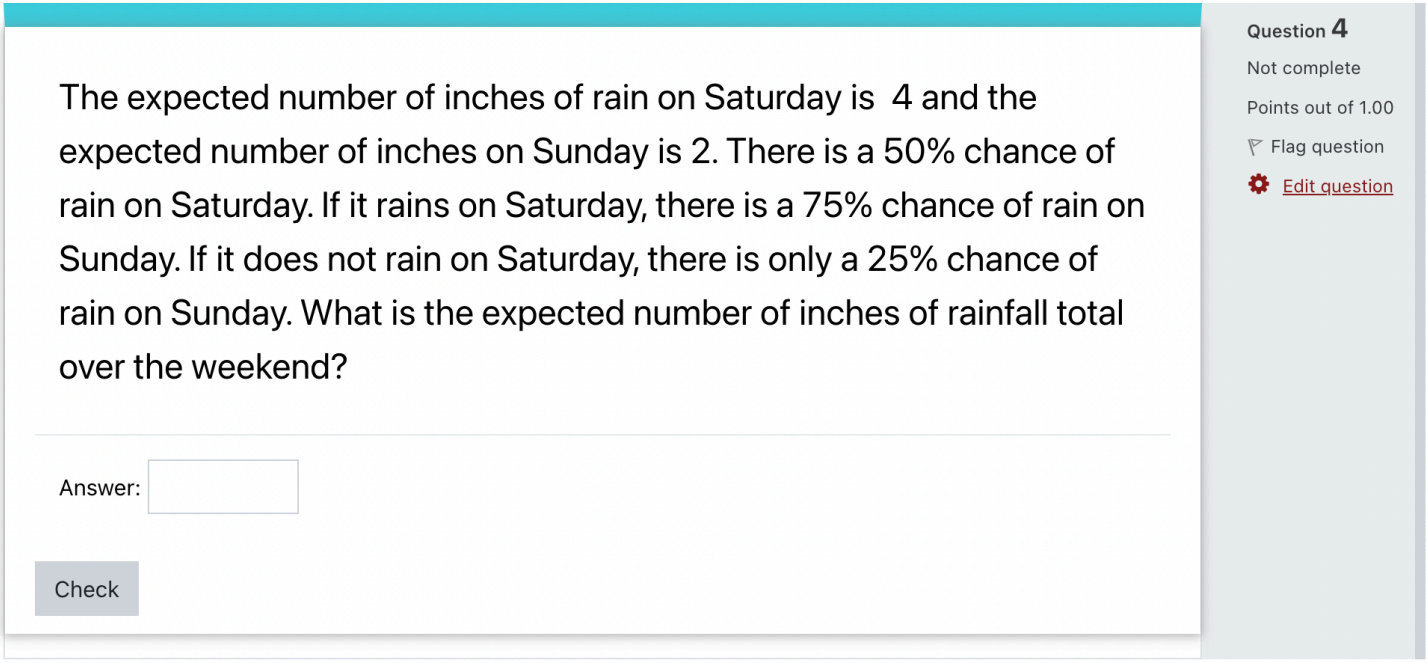
Last Class:

- Linearity of variance.
- Markov's inequality: the most fundamental concentration bound. $\Pr(X \geq t \cdot \mathbb{E}[X]) \leq 1/t$.
- Algorithmic applications of Markov's inequality, linearity of expectation, and indicator random variables:
  - Counting collisions to estimate CAPTCHA database size.
  - Start on analyzing hash tables with random hash functions.

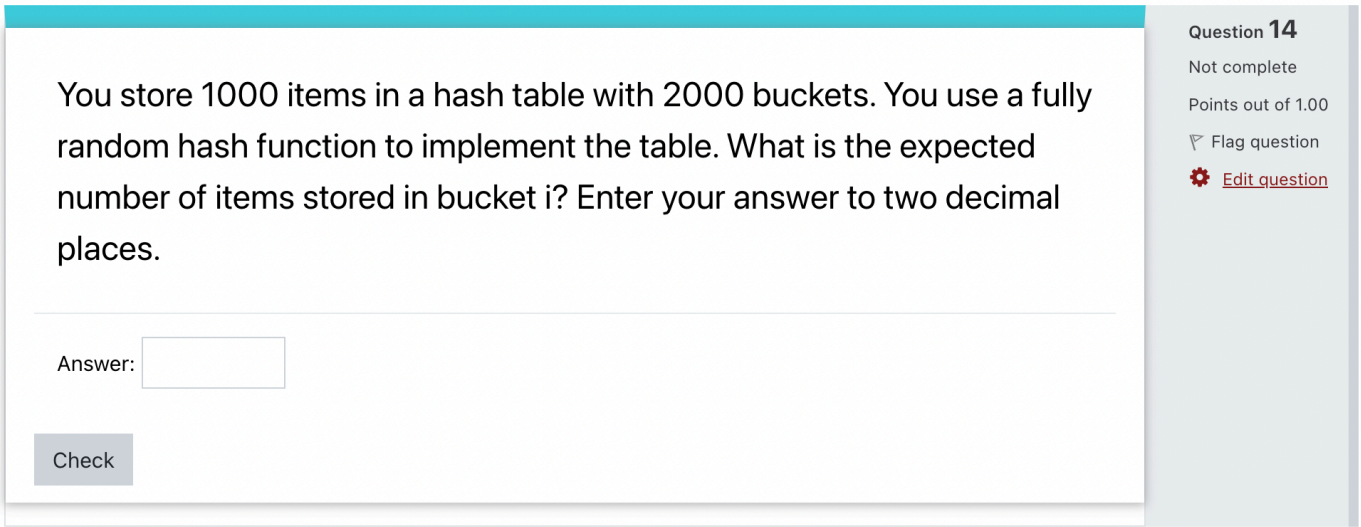
Today:

- Finish up random hash functions and hash tables.
- 2-level hashing, 2-universal and pairwise independent hash functions.

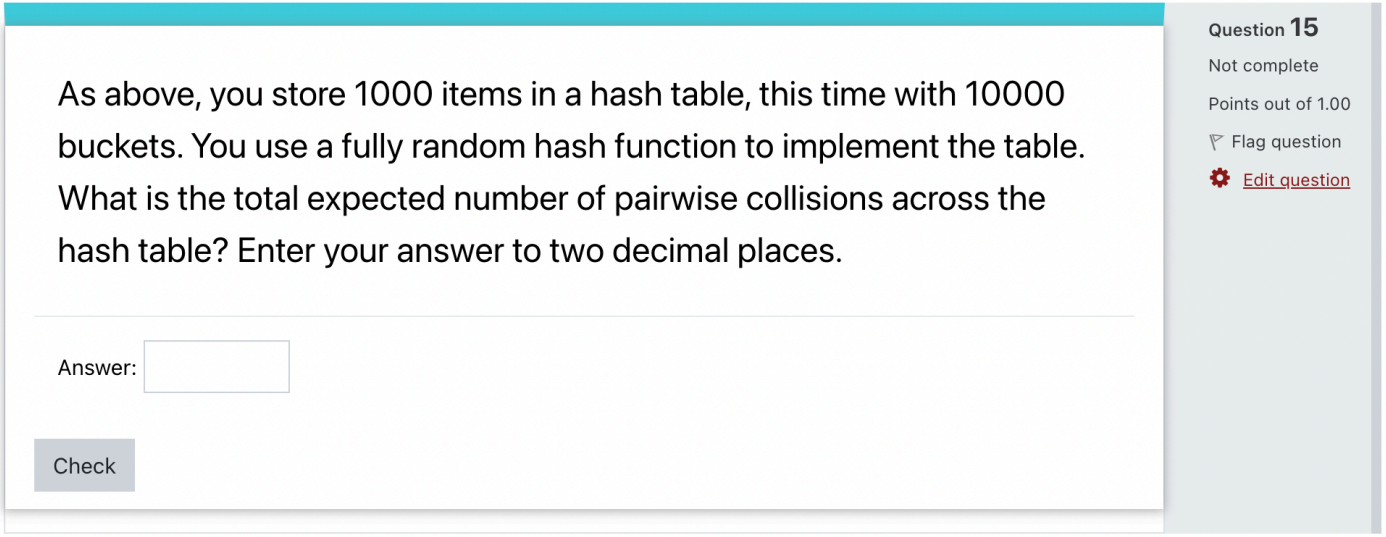
# Quiz Questions



# Quiz Questions

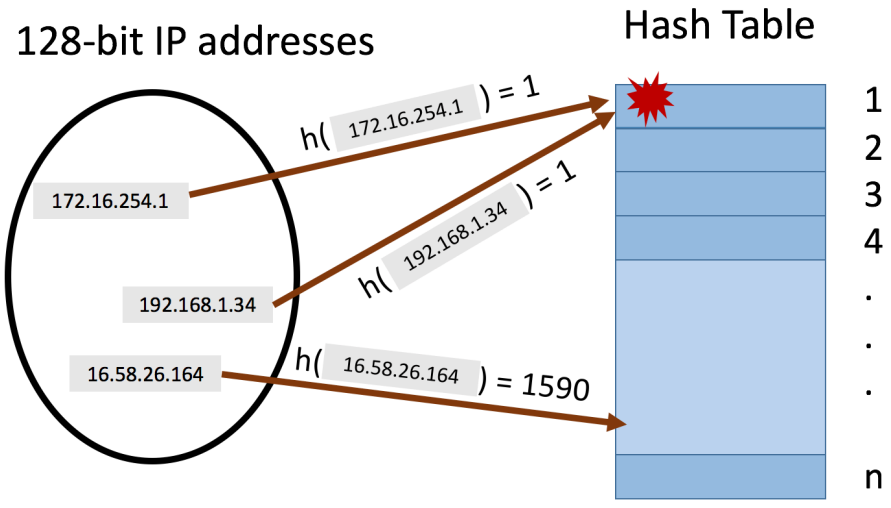


# Quiz Questions



# Hash Tables

We store $m$ items from a large universe in a hash table with $n$ positions.




- Want to show that when $\mathbf{h} : U \to [n]$ is a fully random hash function, query time is $O(1)$ with good probability.
- Equivalently: want to show that there are few collisions between hashed items.

# Linearity of Expectation

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

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

(linearity of expectation)

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

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

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

Identical to the CAPTCHA analysis!

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

# Collision Free Hashing

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

- For $n = 4m^2$ we have: $\mathbb{E}[C] = \frac{m(m-1)}{8m^2} \leq \frac{1}{8}$.
- **Think-Pair-Share:** Give an upper bound on the probability that we have at least one collision, i.e., $\Pr[C \geq 1]$.

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

$$\Pr[C = 0] = 1 - \Pr[C \geq 1] \geq 1 - \frac{1}{8} = \frac{7}{8}$$

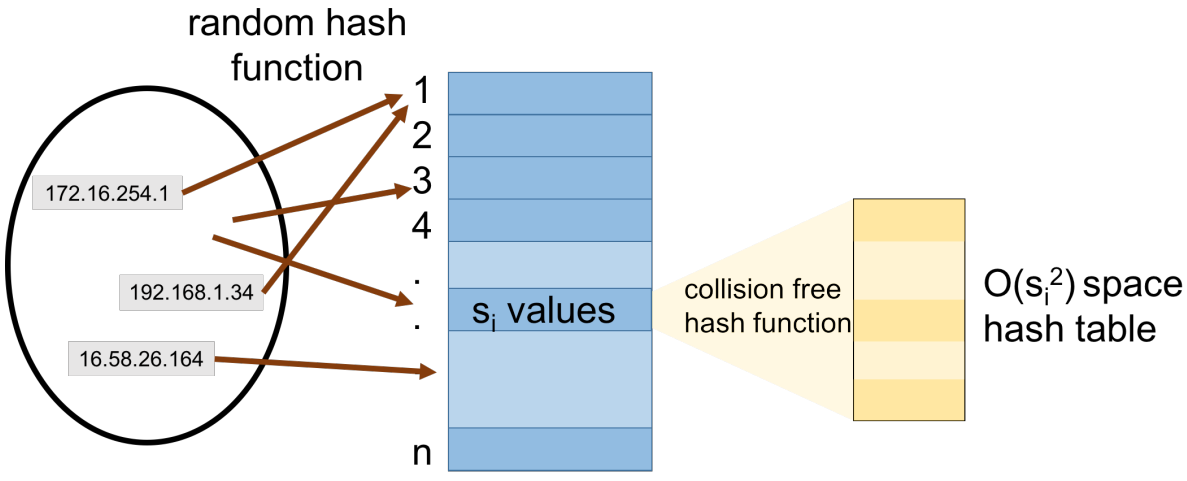
Pretty good...but we are using $O(m^2)$ space to store $m$ items...

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

# Two Level Hashing

Want to preserve $O(1)$ query time while using $O(m)$ space.

## Two-Level Hashing:




- For each bucket with $s_i$ values, pick a collision free hash function mapping $[s_i] \to [s_i^2]$.
- **Just Showed:** A random function is collision free with probability $\geq \frac{7}{8}$ so can just generate a random hash function and check if it is collision free.

# Space Usage

Query time for two level hashing is $O(1)$: requires evaluating two hash functions. What is the expected space usage?

Up to constants, space used is: $\mathbf{S} = n + \sum_{i=1}^{n} \mathbf{s}_i^2 \mathbb{E}[\mathbf{S}] = n + \sum_{i=1}^{n} \mathbb{E}[\mathbf{s}_i^2]$

$$\mathbb{E}[\mathbf{s}_i^2] = \mathbb{E}\left[\left(\sum_{j=1}^{m} \mathbb{I}_{\mathbf{h}(x_j)=i}\right)^2\right]$$

$$= \mathbb{E}\left[\sum_{j,k\in[m]} \mathbb{I}_{\mathbf{h}(x_j)=i} \cdot \mathbb{I}_{\mathbf{h}(x_k)=i}\right] = \sum_{j,k\in[m]} \mathbb{E}\left[\mathbb{I}_{\mathbf{h}(x_j)=i} \cdot \mathbb{I}_{\mathbf{h}(x_k)=i}\right].$$

- For $j = k$,
  $\mathbb{E}\left[\mathbb{I}_{\mathbf{h}(x_j)=i} \cdot \mathbb{I}_{\mathbf{h}(x_k)=i}\right] = \mathbb{E}\left[\left(\mathbb{I}_{\mathbf{h}(x_j)=i}\right)^2\right] = \Pr[\mathbf{h}(x_j) = i] = \frac{1}{n}.$
- For $j \neq k$, $\mathbb{E}\left[\mathbb{I}_{\mathbf{h}(x_j)=i} \cdot \mathbb{I}_{\mathbf{h}(x_k)=i}\right] = \Pr[\mathbf{h}(x_j) = i \cap \mathbf{h}(x_k) = i] = \frac{1}{n^2}.$

Collisions again!

$x_j, x_k$: stored items, $n$: hash table size, $\mathbf{h}$: random hash function, $\mathbf{S}$: space usage of two level hashing, $\mathbf{s}_i$: # items stored in hash table at position $i$.

# Space Usage

$$\mathbb{E}[\mathbf{s}_i^2] = \sum_{j,k\in[m]} \mathbb{E}\left[\mathbb{I}_{\mathbf{h}(x_j)=i} \cdot \mathbb{I}_{\mathbf{h}(x_k)=i}\right]$$

$$= m \cdot \frac{1}{n} + 2 \cdot \binom{m}{2} \cdot \frac{1}{n^2}$$

$$= \frac{m}{n} + \frac{m(m-1)}{n^2} \leq 2 \text{ (If we set } n = m.)$$

- For $j = k$, $\mathbb{E}\left[\mathbb{I}_{\mathbf{h}(x_j)=i} \cdot \mathbb{I}_{\mathbf{h}(x_k)=i}\right] = \frac{1}{n}$.
- For $j \neq k$, $\mathbb{E}\left[\mathbb{I}_{\mathbf{h}(x_j)=i} \cdot \mathbb{I}_{\mathbf{h}(x_k)=i}\right] = \frac{1}{n^2}$.

**Total Expected Space Usage:** (if we set $n = m$)

$$\mathbb{E}[\mathbf{S}] = n + \sum_{i=1}^{n} \mathbb{E}[\mathbf{s}_i^2] \leq n + n \cdot 2 = 3n = 3m.$$

Near optimal space with $O(1)$ query time!

$x_j, x_k$: stored items, $m$: # stored items, $n$: hash table size, $\mathbf{h}$: random hash function, $\mathbf{S}$: space usage of two level hashing, $\mathbf{s}_i$: # items stored at pos $i$.

# Efficiently Computable Hash Function

**So Far:** we have assumed a **fully random hash function** $\mathbf{h}(x)$ with $\Pr[\mathbf{h}(x) = i] = \frac{1}{n}$ for $i \in 1, \ldots, n$ and $\mathbf{h}(x), \mathbf{h}(y)$ independent for $x \neq y$.

- To compute a random hash function we have to store a table of $x$ values and their hash values. Would take at least $O(m)$ space and $O(m)$ query time to look up $\mathbf{h}(x)$ if we hash $m$ values. Making our whole quest for $O(1)$ query time pointless!

| x | h(x) |
|---|---|
| $x_1$ | 45 |
| $x_2$ | 1004 |
| $x_3$ | 10 |
| ⋮ | ⋮ |
| $x_m$ | 12 |

# Efficiently Computable Hash Functions

What properties did we use of the randomly chosen hash function?

**2-Universal Hash Function** (low collision probability). A random hash function from $\mathbf{h} : U \to [n]$ is two universal if:

$$\Pr[\mathbf{h}(x) = \mathbf{h}(y)] \leq \frac{1}{n}.$$

**Exercise:** Rework the two level hashing proof to show that this property is really all that is needed.

When $\mathbf{h}(x)$ and $\mathbf{h}(y)$ are chosen independently at random from $[n]$, $\Pr[\mathbf{h}(x) = \mathbf{h}(y)] = \frac{1}{n}$ (so a fully random hash function is 2-universal)

**Efficient Alternative:** Let $p$ be a prime with $p \geq |U|$. Choose random $\mathbf{a}, \mathbf{b} \in [p]$ with $\mathbf{a} \neq 0$. Represent $x$ an an integer and let

$$\mathbf{h}(x) = (\mathbf{a}x + \mathbf{b} \mod p) \mod n.$$

# Pairwise Independence

Another common requirement for a hash function:

**Pairwise Independent Hash Function.** A random hash function from $\mathbf{h} : U \to [n]$ is pairwise independent if for all $i, j \in [n]$:

$$\Pr[\mathbf{h}(x) = i \cap \mathbf{h}(y) = j] = \frac{1}{n^2}.$$

**Think-Pair-Shair:** Which is a more stringent requirement?
2-universal or pairwise independent?

**2-Universal Hash Function** (low collision probability). A random hash function from $\mathbf{h} : U \to [n]$ is two universal if:

$$\Pr[\mathbf{h}(x) = \mathbf{h}(y)] \leq \frac{1}{n}.$$

Questions?