COMPSCI 514: Algorithms for Data Science

Cameron Musco

University of Massachusetts Amherst. Fall 2023.

Lecture 3

Logistics

- Problem Set 1 has been posted on the course website and is due **Friday 9/22 at 11:59pm**.
- On the quiz feedback question, several people mentioned concerns about linear algebra background. The good news is you have some time – we will do essentially no linear algebra before the midterm. See resources on Lecture 15 for review material to get started on.
- For probability/problem solving practice beyond the quizzes/pset I highly recommend looking at the exercises in *Foundations of Data Science* and *Probability and Computing*. Feel free to ask for solutions to these on Piazza.
- 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.
  - Collisions free hashing using a table with $O(m^2)$ slots to store m items.

Content Overview

Today:

- Finish up random hash functions and hash tables.
- 2-level hashing, 2-universal and pairwise independent hash functions.
- Application of random hashing to distributed load balancing.
- Through this application learn about Chebyshev's inequality, which strengthens Markov's inequality (maybe not until next class).

Quiz Questions

The expected number of inches of rain on Saturday is 4 and the expected number of inches on Sunday is 2. There is a 50% chance of rain on Saturday. If it rains on Saturday, there is a 75% chance of rain on Sunday. If it does not rain on Saturday, there is only a 25% chance of rain on Sunday. What is the expected number of inches of rainfall total over the weekend?

Answer:

Check

Question 4

Not complete

Points out of 1.00

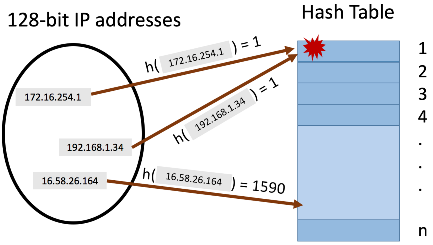
Flag question

Edit question

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.

Collision Free Hashing

Let $\mathbf{C} = \sum_{i,j \in [m], i<j} \mathbf{C}_{i,j}$ be the number of pairwise collisions between items.

$$\mathbb{E}[\mathbf{C}] = \frac{m(m-1)}{2n} \quad \text{(via the Captcha analysis)}$$

- For $n = 4m^2$ we have: $\mathbb{E}[\mathbf{C}] = \frac{m(m-1)}{8m^2} \leq \frac{1}{8}$.

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

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

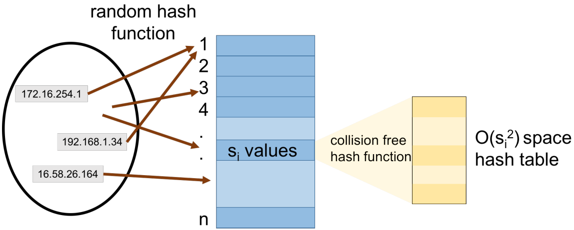
I.e., with probability at least 7/8 we have no collisions and thus $O(1)$ query time. But we are using $O(m^2)$ space to store m items...

m: total number of stored items, n: hash table size, $\mathbf{C}$: 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 $j = k$,

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

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

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** $h(x)$ with $\Pr[h(x) = i] = \frac{1}{n}$ for $i \in 1, \ldots, n$ and $h(x), h(y)$ 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 $h(x)$ 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?

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

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

Exercise 1: Check the two-level hashing proof to confirm that this property is all that was needed.

When $\mathbf{h}(x)$ and $\mathbf{h}(y)$ are chosen independently at random from $[n]$, $\Pr[\mathbf{h}(x) = i \cap \mathbf{h}(y) = j] = \frac{1}{n^2}$ (so a fully random hash function is pairwise independent).

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

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

Universal Hashing

Another common requirement for a hash function:

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

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

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

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

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