COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Andrew McGregor
Lecture 3
Last Class We Covered:

- Markov’s inequality: the most fundamental concentration bound.
- Algorithmic applications of Markov’s inequality, linearity of expectation, and indicator random variables:
  - Counting collisions to estimate CAPTCHA database size.
  - Counting collisions to understand the runtime of hash tables with random hash functions.
Last Class We Covered:

• Markov’s inequality: the most fundamental concentration bound.
• Algorithmic applications of Markov’s inequality, linearity of expectation, and indicator random variables:
  • Counting collisions to estimate CAPTCHA database size.
  • Counting collisions to understand the runtime of hash tables with random hash functions.
• Collision counting is closely related to the birthday paradox.
Today:

• Finish up random hash functions and hash tables.
• See an application of random hashing to load balancing in distributed systems.
• Through this application learn about:
  • Chebyshev’s inequality, which strengthens Markov’s inequality.
  • The union bound, for understanding the probabilities of correlated random events.
We store $m$ items from a large universe in a hash table with $n$ positions.

• Want to show that when $h : U \rightarrow [n]$ is a random hash function, query time is $O(1)$ with good probability.
• Equivalently: want to show that there are few collisions between hashed items.
When storing $m$ items in a table of size $n$, the expected number of pairwise collisions (two items stored in the same slots) is:

$$\mathbb{E}[C] = \frac{m(m - 1)}{2n}.$$
When storing $m$ items in a table of size $n$, the expected number of pairwise collisions (two items stored in the same slots) is:

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

- For $n = 4m^2$ we have: $\mathbb{E}[C] = \frac{m(m-1)}{8m^2} \leq \frac{1}{8}$.
- By Markov’s inequality there no collisions with probability at least $\frac{7}{8}$.

$m$: total number of stored items, $n$: hash table size, $C$: total pairwise collisions in table.
When storing $m$ items in a table of size $n$, the expected number of pairwise collisions (two items stored in the same slots) is:

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

- For $n = 4m^2$ we have: $\mathbb{E}[C] = \frac{m(m-1)}{8m^2} \leq \frac{1}{8}$.
- By Markov’s inequality there no collisions with probability at least $\frac{7}{8}$.

$O(1)$ query time, but we are using $O(m^2)$ space to store $m$ items...

$m$: total number of stored items, $n$: hash table size, $C$: total pairwise collisions in table.
Want to preserve $O(1)$ query time while using $O(m)$ space.
Want to preserve $O(1)$ query time while using $O(m)$ space.

**Two-Level Hashing:**

Random hash function:

- For each bucket with $s_i$ values, pick a collision free hash function mapping $[s_i] \rightarrow [s_i^2]$.

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.
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 $[s_i] \rightarrow [s_i^2]$.
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 $[s_i] \rightarrow [s_i^2]$.
- **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.
Query time for two level hashing is $O(1)$: requires evaluating two hash functions.

$x_j, x_k$: stored items, $n$: hash table size, $h$: random hash function, $S$: space usage of two level hashing, $s_i$: # items stored in hash table at position $i$. 
Query time for two level hashing is $O(1)$: requires evaluating two hash functions. What is the expected space usage?

$x_j, x_k$: stored items, $n$: hash table size, $h$: random hash function, $S$: space usage of two level hashing, $s_i$: # items stored in hash table at position $i$. 
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: $S = n + \sum_{i=1}^{n} s_i^2$
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: $\mathbb{E}[S] = n + \sum_{i=1}^{n} \mathbb{E}[s_i^2]$
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:

$$\mathbb{E}[S] = n + \sum_{i=1}^{n} \mathbb{E}[s_i^2]$$

$x_j, x_k$: stored items, $n$: hash table size, $h$: random hash function, $S$: space usage of two level hashing, $s_i$: # items stored in hash table at position $i$. 
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: $\mathbb{E}[S] = n + \sum_{i=1}^{n} \mathbb{E}[s_i^2]$

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

$x_j, x_k$: stored items, $n$: hash table size, $h$: random hash function, $S$: space usage of two level hashing, $s_i$: # items stored in hash table at position $i$. 
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: $\mathbb{E}[S] = n + \sum_{i=1}^{n} \mathbb{E}[s_i^2]$

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

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

Collisions again!

$x_j, x_k$: stored items, $n$: hash table size, $h$: random hash function, $S$: space usage of two level hashing, $s_i$: # items stored in hash table at position $i$. 
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: $\mathbb{E}[S] = n + \sum_{i=1}^{n} \mathbb{E}[s_i^2]$ 

\[
\mathbb{E}[s_i^2] = \mathbb{E} \left[ \left( \sum_{j=1}^{m} I_{h(x_j) = i} \right)^2 \right] 
= \mathbb{E} \left[ \sum_{j,k \in [m]} I_{h(x_j) = i} \cdot I_{h(x_k) = i} \right] = \sum_{j,k \in [m]} \mathbb{E} \left[ I_{h(x_j) = i} \cdot I_{h(x_k) = i} \right].
\]

$x_j, x_k$: stored items, $n$: hash table size, $h$: random hash function, $S$: space usage of two level hashing, $s_i$: # items stored in hash table at position $i$. 
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: $\mathbb{E}[S] = n + \sum_{i=1}^{n} \mathbb{E}[s_i^2]$

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

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

- For $j = k$,

$x_j, x_k$: stored items, $n$: hash table size, $h$: random hash function, $S$: space usage of two level hashing, $s_i$: # items stored in hash table at position $i$. 
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: $\mathbb{E}[S] = n + \sum_{i=1}^{n} \mathbb{E}[s_i^2]$

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

$$= \mathbb{E}\left[ \sum_{j,k \in [m]} \mathbb{1}_{h(x_j) = i} \cdot \mathbb{1}_{h(x_k) = i} \right] = \sum_{j,k \in [m]} \mathbb{E}\left[ \mathbb{1}_{h(x_j) = i} \cdot \mathbb{1}_{h(x_k) = i} \right].$$

- For $j = k$, $\mathbb{E}\left[ \mathbb{1}_{h(x_j) = i} \cdot \mathbb{1}_{h(x_k) = i} \right] = \mathbb{E}\left[ (\mathbb{1}_{h(x_j) = i})^2 \right]$
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: $\mathbb{E}[S] = n + \sum_{i=1}^{n} \mathbb{E}[s_i^2]$}

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

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

• For $j = k$, $\mathbb{E} \left[ \mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i} \right] = \mathbb{E} \left[ \left( \mathbb{I}_{h(x_j)=i} \right)^2 \right] = \Pr[h(x_j) = i]

\begin{align*}
x_j, x_k: & \text{ stored items, } n: \text{ hash table size, } h: \text{ random hash function, } S: \text{ space usage of two level hashing, } s_i: \# \text{ items stored in hash table at position } i.
\end{align*}
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: $\mathbb{E}[S] = n + \sum_{i=1}^{n} \mathbb{E}[s_i^2]$

$$\mathbb{E}[s_i^2] = \mathbb{E} \left[ \left( \sum_{j=1}^{m} \mathbb{1}_{h(x_j) = i} \right)^2 \right] = \mathbb{E} \left[ \sum_{j,k \in [m]} \mathbb{1}_{h(x_j) = i} \cdot \mathbb{1}_{h(x_k) = i} \right] = \sum_{j,k \in [m]} \mathbb{E} \left[ \mathbb{1}_{h(x_j) = i} \cdot \mathbb{1}_{h(x_k) = i} \right].$$

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

$x_j, x_k$: stored items, $n$: hash table size, $h$: random hash function, $S$: space usage of two level hashing, $s_i$: # items stored in hash table at position $i$. 
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: $E[S] = n + \sum_{i=1}^{n} E[s_i^2]$

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

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

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

- For $j \neq k$,

$x_j, x_k$: stored items, $n$: hash table size, $h$: random hash function, $S$: space usage of two level hashing, $s_i$: # items stored in hash table at position $i$. 
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: $E[S] = n + \sum_{i=1}^{n} E[s_i^2]$

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

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

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

$x_j, x_k$: stored items, $n$: hash table size, $h$: random hash function, $S$: space usage of two level hashing, $s_i$: # items stored in hash table at position $i$. 
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: $\mathbb{E}[S] = n + \sum_{i=1}^{n} \mathbb{E}[s_i^2]$

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

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

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

$x_j, x_k$: stored items, $n$: hash table size, $h$: random hash function, $S$: space usage of two level hashing, $s_i$: # items stored in hash table at position $i$. 
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: $\mathbb{E}[S] = n + \sum_{i=1}^{n} \mathbb{E}[s_i^2]$

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

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

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

$x_j, x_k$: stored items, $n$: hash table size, $h$: random hash function, $S$: space usage of two level hashing, $s_i$: # items stored in hash table at position $i$. 
\[ \mathbb{E}[S_i^2] = \sum_{j,k \in [m]} \mathbb{E} \left[ \mathbb{I}_{h(x_j)=i} \cdot \mathbb{I}_{h(x_k)=i} \right] \]

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

\( x_j, x_k \): stored items, \( m \): \# stored items, \( n \): hash table size, \( h \): random hash function, \( S \): space usage of two level hashing, \( s_i \): \# items stored at pos \( i \).
**SPACE USAGE**

$$\mathbb{E}[s_i^2] = \sum_{j,k \in [m]} \mathbb{E}[\mathbb{1}_{h(x_j)=i} \cdot \mathbb{1}_{h(x_k)=i}]$$

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

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

$x_j, x_k$: stored items, $m$: # stored items, $n$: hash table size, $h$: random hash function, $S$: space usage of two level hashing, $s_i$: # items stored at pos $i$. 
\[ \mathbb{E}[s_i^2] = \sum_{j,k \in [m]} \mathbb{E} \left[ \mathbb{I}_h(x_j) = i \cdot \mathbb{I}_h(x_k) = i \right] \]

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

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

\( x_j, x_k \): stored items, \( m \): \# stored items, \( n \): hash table size, \( h \): random hash function, \( S \): space usage of two level hashing, \( s_i \): \# items stored at pos \( i \).
SPACE USAGE

\[ E[s_i^2] = \sum_{j, k \in [m]} E[\mathbb{I}(h(x_j) = i) \cdot \mathbb{I}(h(x_k) = i)] \]
\[ = m \cdot \frac{1}{n} + 2 \cdot \left( \binom{m}{2} \right) \cdot \frac{1}{n^2} \]

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

\( x_j, x_k \): stored items, \( m \): \# stored items, \( n \): hash table size, \( h \): random hash function, \( S \): space usage of two level hashing, \( s_i \): \# items stored at pos \( i \).
SPAC E  USAG E

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

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

\[
= \frac{m}{n} + \frac{m(m-1)}{n^2}
\]

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

\( x_j, x_k \): stored items, \( m \): \# stored items, \( n \): hash table size, \( h \): random hash function, \( S \): space usage of two level hashing, \( s_i \): \# items stored at pos \( i \).
\[ \mathbb{E}[s_i^2] = \sum_{j,k \in [m]} \mathbb{E} \left[ \mathbb{I}_{h(x_j) = i} \cdot \mathbb{I}_{h(x_k) = i} \right] \]
\[ = m \cdot \frac{1}{n} + 2 \cdot \binom{m}{2} \cdot \frac{1}{n^2} \]
\[ = \frac{m}{n} + \frac{m(m-1)}{n^2} \leq 2 \text{ (If we set } n = m.) \]

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

\( x_j, x_k \): stored items, \( m \): \# stored items, \( n \): hash table size, \( h \): random hash function, \( S \): space usage of two level hashing, \( s_i \): \# items stored at pos \( i \).
SPACE USAGE

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

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

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

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

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

\[ \mathbb{E}[S] = n + \sum_{i=1}^{n} \mathbb{E}[S_i^2] \]

\( x_j, x_k \): stored items, \( m \): \# stored items, \( n \): hash table size, \( h \): random hash function, \( S \): space usage of two level hashing, \( s_i \): \# items stored at pos \( i \).
\[ \mathbb{E}[s_i^2] = \sum_{j, k \in [m]} \mathbb{E} \left[ \mathbb{I}_{h(x_j) = i} \cdot \mathbb{I}_{h(x_k) = i} \right] \]

\[ = m \cdot \frac{1}{n} + 2 \cdot \left( \frac{m}{2} \right) \cdot \frac{1}{n^2} \]

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

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

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

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

\( x_j, x_k \): stored items, \( m \): \# stored items, \( n \): hash table size, \( h \): random hash function, \( S \): space usage of two level hashing, \( s_i \): \# items stored at pos \( i \).
\[ \mathbb{E}[s_i^2] = \sum_{j, k \in [m]} \mathbb{E} \left[ \mathbb{I}_{h(x_j) = i} \cdot \mathbb{I}_{h(x_k) = i} \right] \]
\[ = m \cdot \frac{1}{n} + 2 \cdot \binom{m}{2} \cdot \frac{1}{n^2} \]
\[ = \frac{m}{n} + \frac{m(m-1)}{n^2} \leq 2 \text{ (If we set } n = m). \]

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

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

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

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

\( x_j, x_k \): stored items, \( m \): \# stored items, \( n \): hash table size, \( h \): random hash function, \( S \): space usage of two level hashing, \( s_i \): \# items stored at pos \( i \).
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 $x \neq y$. 

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 if we hash $m$ values. Making our whole quest for $O(1)$ query time pointless!

<table>
<thead>
<tr>
<th>$x$</th>
<th>$h(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>45</td>
</tr>
<tr>
<td>$x_2$</td>
<td>1004</td>
</tr>
<tr>
<td>$x_3$</td>
<td>10</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>$x_m$</td>
<td>12</td>
</tr>
</tbody>
</table>
What properties did we use of the randomly chosen hash function?

2-Universal Hash Function

A random hash function from $h : U \rightarrow \mathbb{N}$ is two universal if:

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

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

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

Efficient Alternative:

Let $p$ be a prime with $p \geq |U|$. Choose random $a, b \in \mathbb{N}$ with $a \neq 0$. Let:

$$h(x) = (ax + b \mod p) \mod n.$$
What properties did we use of the randomly chosen hash function?

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

$$\Pr[h(x) = h(y)] \leq \frac{1}{n}.$$
What properties did we use of the randomly chosen hash function?

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

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

**Exercise:** Rework the two level hashing proof to show that this property is really all that is needed.
What properties did we use of the randomly chosen hash function?

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

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

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

When \( h(x) \) and \( h(y) \) are chosen independently at random from \([n]\), \( \Pr[h(x) = h(y)] = \frac{1}{n} \) (so a fully random hash function is 2-universal)
What properties did we use of the randomly chosen hash function?

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

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

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

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

**Efficient Alternative:** Let \( p \) be a prime with \( p \geq |U| \). Choose random \( a, b \in [p] \) with \( a \neq 0 \). Let:

\[
h(x) = (ax + b \mod p) \mod n.
\]
Another common requirement for a hash function:

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

Breakout: Which is a more stringent requirement? 2-universal or pairwise independent?

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

A closely related \((ax + b) \mod p\) construction gives pairwise independence on top of 2-universality.

Remember: A fully random hash function is both 2-universal and pairwise independent. But it is not efficiently implementable.
Another common requirement for a hash function:

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

$$\Pr[h(x) = i \cap h(y) = j] = \frac{1}{n^2}.$$
Another common requirement for a hash function:

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

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

**Breakout:** Which is a more stringent requirement? 2-universal or pairwise independent?
Another common requirement for a hash function:

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

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

**Breakout:** Which is a more stringent requirement? 2-universal or pairwise independent?

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

A closely related $(ax + b) \mod p$ construction gives pairwise independence on top of 2-universality. Remember: A fully random hash function is both 2-universal and pairwise independent. But it is not efficiently implementable.
Another common requirement for a hash function:

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

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

**Breakout:** Which is a more stringent requirement? 2-universal or pairwise independent?

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

A closely related $(ax + b) \mod p$ construction gives pairwise independence on top of 2-universality.
Another common requirement for a hash function:

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

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

**Breakout:** Which is a more stringent requirement? 2-universal or pairwise independent?

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

A closely related $(ax + b) \ mod \ p$ construction gives pairwise independence on top of 2-universality.

**Remember:** A fully random hash function is both 2-universal and pairwise independent. But it is not efficiently implementable.
1. We’ll consider an application where our toolkit of linearity of expectation + Markov’s inequality doesn’t give much.
1. We’ll consider an application where our toolkit of linearity of expectation + Markov’s inequality doesn’t give much.

2. Then we’ll show how a simple twist on Markov’s can give a much stronger result.
Randomized Load Balancing:

**Simple Model:**

$n$ requests randomly assigned to $k$ servers. How many requests must each server handle?
Randomized Load Balancing:

**Simple Model:** \( n \) requests randomly assigned to \( k \) servers. How many requests must each server handle?
Randomized Load Balancing:

Simple Model: $n$ requests randomly assigned to $k$ servers. How many requests must each server handle?
WEAKNESS OF MARKOV’S

Expected Number of requests assigned to server $i$:

$$\mathbb{E}[R_i] = \sum_{j=1}^{n} \mathbb{E}[\mathbb{1}_{\text{request } j \text{ assigned to } i}] = \sum_{j=1}^{n} \Pr [j \text{ assigned to } i] = \frac{n}{k}.$$ 

$n$: total number of requests, $k$: number of servers randomly assigned requests, $R_i$: number of requests assigned to server $i$. 

If we provision each server be able to handle twice the expected load, what is the probability that a server is overloaded? Applying Markov’s Inequality

$$\Pr [R_i \geq 2\mathbb{E}[R_i]] \leq \frac{\mathbb{E}[R_i]}{2\mathbb{E}[R_i]} = \frac{1}{2}.$$ 

Not great...half the servers may be overloaded.
Expected Number of requests assigned to server $i$: \[
\mathbb{E}[R_i] = \sum_{j=1}^{n} \mathbb{E}[\mathbb{I}_{\text{request } j \text{ assigned to } i}] = \sum_{j=1}^{n} \Pr[j \text{ assigned to } i] = \frac{n}{k}.
\]

If we provision each server be able to handle twice the expected load, what is the probability that a server is overloaded?

$n$: total number of requests, $k$: number of servers randomly assigned requests, $R_i$: number of requests assigned to server $i$. 

WEAKNESS OF MARKOV’S

Expected Number of requests assigned to server $i$:

$$\mathbb{E}[R_i] = \sum_{j=1}^{n} \mathbb{E}[\mathbb{1}_{\text{request } j \text{ assigned to } i}] = \sum_{j=1}^{n} \Pr[j \text{ assigned to } i] = \frac{n}{k}.$$ 

If we provision each server be able to handle twice the expected load, what is the probability that a server is overloaded?

Applying Markov’s Inequality

$$\Pr[R_i \geq 2\mathbb{E}[R_i]] \leq \frac{\mathbb{E}[R_i]}{2\mathbb{E}[R_i]} = \frac{1}{2}.$$ 

$n$: total number of requests, $k$: number of servers randomly assigned requests, $R_i$: number of requests assigned to server $i$. 
Expected Number of requests assigned to server $i$:

$$E[R_i] = \sum_{j=1}^{n} E[I_{\text{request } j \text{ assigned to } i}] = \sum_{j=1}^{n} Pr[j \text{ assigned to } i] = \frac{n}{k}.$$ 

If we provision each server be able to handle twice the expected load, what is the probability that a server is overloaded?

Applying Markov’s Inequality

$$Pr[R_i \geq 2E[R_i]] \leq \frac{E[R_i]}{2E[R_i]} = \frac{1}{2}.$$ 

Not great...half the servers may be overloaded.

$n$: total number of requests, $k$: number of servers randomly assigned requests, $R_i$: number of requests assigned to server $i$. 
With a very simple twist Markov’s Inequality can be made much more powerful.
With a very simple twist Markov’s Inequality can be made much more powerful.

For any random variable $X$ and any value $t > 0$:

$$\Pr(|X| \geq t) = \Pr(X^2 \geq t^2).$$
CHEBYSHEV’S INEQUALITY

With a very simple twist Markov’s Inequality can be made much more powerful.

For any random variable $X$ and any value $t > 0$:

$$\Pr(|X| \geq t) = \Pr(X^2 \geq t^2).$$

$X^2$ is a nonnegative random variable. So can apply Markov’s inequality:
With a very simple twist Markov’s Inequality can be made much more powerful.

For any random variable $X$ and any value $t > 0$:

$$\Pr(|X| \geq t) = \Pr(X^2 \geq t^2).$$

$X^2$ is a nonnegative random variable. So can apply Markov’s inequality:

$$\Pr(X^2 \geq t^2) \leq \frac{\mathbb{E}[X^2]}{t^2}.$$
With a very simple twist Markov’s Inequality can be made much more powerful.

For any random variable $X$ and any value $t > 0$:

$$\Pr(|X| \geq t) = \Pr(X^2 \geq t^2).$$

$X^2$ is a nonnegative random variable. So can apply Markov’s inequality:

$$\Pr(|X| \geq t) = \Pr(X^2 \geq t^2) \leq \frac{\mathbb{E}[X^2]}{t^2}.$$
With a very simple twist Markov’s Inequality can be made much more powerful.

For any random variable $X$ and any value $t > 0$:

$$\Pr(|X| \geq t) = \Pr(X^2 \geq t^2).$$

$X^2$ is a nonnegative random variable. So can apply Markov’s inequality:

**Chebyshev’s inequality:**

$$\Pr(|X| \geq t) = \Pr(X^2 \geq t^2) \leq \frac{\mathbb{E}[X^2]}{t^2}.$$
With a very simple twist Markov’s Inequality can be made much more powerful.

For any random variable $X$ and any value $t > 0$:

$$\Pr(|X| \geq t) = \Pr(X^2 \geq t^2).$$

$X^2$ is a nonnegative random variable. So can apply Markov’s inequality:

**Chebyshev’s inequality:**

$$\Pr(|X - \mathbb{E}[X]| \geq t) \leq \frac{\text{Var}[X]}{t^2}.$$

(by plugging in the random variable $X - \mathbb{E}[X]$)
CHEBYSHEV’S INEQUALITY

\[ \Pr(|X - \mathbb{E}[X]| \geq t) \leq \frac{\text{Var}[X]}{t^2} \]

\( X \): any random variable, \( t, s \): any fixed numbers.
Chebyshev’s Inequality

\[ \Pr(|X - \mathbb{E}[X]| \geq t) \leq \frac{\text{Var}[X]}{t^2} \]

What is the probability that \( X \) falls \( s \) standard deviations from its mean?

\[ \text{X: any random variable, } t, s: \text{ any fixed numbers.} \]
**CHEBYSHEV’S INEQUALITY**

\[
\Pr(\left| X - \mathbb{E}[X] \right| \geq t) \leq \frac{\text{Var}[X]}{t^2}
\]

What is the probability that \( X \) falls \( s \) standard deviations from its mean?

\[
\Pr(\left| X - \mathbb{E}[X] \right| \geq s \cdot \sqrt{\text{Var}[X]}) \leq \frac{\text{Var}[X]}{s^2 \cdot \text{Var}[X]} = \frac{1}{s^2}.
\]

\( \mathbf{X} \): any random variable, \( t, s \): any fixed numbers.
**Chebyshev’s Inequality**

\[
\Pr(|X - \mathbb{E}[X]| \geq t) \leq \frac{\text{Var}[X]}{t^2}
\]

What is the probability that \(X\) falls \(s\) standard deviations from its mean?

\[
\Pr(|X - \mathbb{E}[X]| \geq s \cdot \sqrt{\text{Var}[X]}) \leq \frac{\text{Var}[X]}{s^2 \cdot \text{Var}[X]} = \frac{1}{s^2}.
\]

Why is this so powerful?

\(X\): any random variable, \(t, s\): any fixed numbers.