Lecture 3
Static Perfect Hashing
A dynamic dictionary stores \((key, value)\)-pairs and supports:

- \(\text{add}(key, value)\), \(\text{lookup}(key)\) (which returns \(value\)) and \(\text{delete}(key)\)

Universe \(U\) of \(u\) keys.

Hash table \(T\) of size \(m \geq n\).

Collisions are fixed by chaining.

A hash function maps a key \(x\) to position \(h(x)\) - i.e \(T[h(x)] = (key, value)\).

\(n\) arbitrary operations arrive online, one at a time.
Dictionaries and Hashing recap

A dynamic dictionary stores \((key, value)\)-pairs and supports:

- \(\text{add}(key, value)\), \(\text{lookup}(key)\) (which returns \(value\)) and \(\text{delete}(key)\)

Universe \(U\) of \(u\) keys.

Hash table \(T\) of size \(m \geq n\).

Collisions are fixed by chaining

A hash function maps a key \(x\) to position \(h(x)\) - i.e \(T[h(x)] = (key, value)\).

\(n\) arbitrary operations arrive online, one at a time.

A set \(H\) of hash functions is weakly universal if for any two keys \(x, y \in U\) (with \(x \neq y\)),

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

(h is picked uniformly at random from \(H\))
Dictionaries and Hashing recap

- A **dynamic dictionary** stores \((key, value)\)-pairs and supports:
  - \texttt{add(key, value)}, \texttt{lookup(key)} (which returns \texttt{value}) and \texttt{delete(key)}

Universe \(U\) of \(u\) keys.

Hash table \(T\) of size \(m \geq n\).

Collisions are fixed by chaining.

A **hash function** maps a key \(x\) to position \(h(x)\) - i.e \(T[h(x)] = (key, value)\).

\(n\) arbitrary operations arrive online, one at a time.

A set \(H\) of hash functions is **weakly universal** if for any two keys \(x, y \in U\) (with \(x \neq y\)),

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

\((h\) is picked uniformly at random from \(H\))

Using weakly universal hashing:

For any \(n\) operations, the expected run-time is \(O(1)\) per operation.
A **dynamic dictionary** stores *(key, value)*-pairs and supports: `add(key, value)`, `lookup(key)` (which returns `value`) and `delete(key)`.

Universe $U$ of $u$ keys. Hash table $T$ of size $m \geq n$.

Collisions are fixed by chaining.

A **hash function** maps a key $x$ to position $h(x)$ - i.e $T[h(x)] = (key, value)$.

$n$ arbitrary operations arrive online, one at a time.

A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U$ (with $x \neq y$),

$$\Pr (h(x) = h(y)) \leq \frac{1}{m}$$

($h$ is picked uniformly at random from $H$)

Using weakly universal hashing:

For any $n$ operations, the expected run-time is $O(1)$ per operation.

But this doesn’t tell us much about the worst-case behaviour.
A static dictionary stores \((key, value)\)-pairs and supports:

- lookup\((key)\) (which returns value) - no inserts or deletes are allowed

Universe \(U\) of \(u\) keys.

Hash table \(T\) of size \(m \geq n\).

Collisions are fixed by chaining.

A hash function maps a key \(x\) to position \(h(x)\) - i.e \(T[h(x)] = (key, value)\).

we are given \(n\) different \((key, value)\)-pairs and want to pick a good \(h\).
A static dictionary stores \((key, value)\)-pairs and supports:

\[ \text{lookup}(key) \] (which returns \text{value}) - no inserts or deletes are allowed

Universe \(U\) of \(u\) keys.

Hash table \(T\) of size \(m \geq n\).

Collisions are fixed by chaining.

A hash function maps a key \(x\) to position \(h(x)\) - i.e \(T[h(x)] = \text{(key, value)}\).

we are given \(n\) different \((key, value)\)-pairs and want to pick a good \(h\)

**Theorem**

The FKS hashing scheme:

- Has no collisions
- Every lookup takes \(O(1)\) worst-case time,
- Uses \(O(n)\) space,
- Can be built in \(O(n)\) expected time.
A **static dictionary** stores \((key, value)\)-pairs and supports:

- **lookup**\((key)\) (which returns **value**) - no inserts or deletes are allowed

We are given \(n\) different \((key, value)\)-pairs and want to pick a **good** \(h\)

**Theorem**

The FKS hashing scheme:

- Has no collisions
- Every **lookup** takes \(O(1)\) worst-case time,
- Uses \(O(n)\) space,
- Can be built in \(O(n)\) expected time.

The rest of this lecture is devoted to the FKS scheme
A static dictionary stores \((key, value)\)-pairs and supports:

\[ \text{lookup}(key) \] (which returns \text{value}) - no inserts or deletes are allowed

Universe \( U \) of \( u \) keys.

Hash table \( T \) of size \( m \geq n \).

Collisions are fixed by chaining.

A hash function maps a key \( x \) to position \( h(x) \) - i.e \( T[h(x)] = (key, value) \).

we are given \( n \) different \((key, value)\)-pairs and want to pick a good \( h \)

**Theorem**

The FKS hashing scheme:

- Has no collisions
- Every lookup takes \( O(1) \) worst-case time,
- Uses \( O(n) \) space,
- Can be built in \( O(n) \) expected time.

The rest of this lecture is devoted to the FKS scheme

The construction is based on weak universal hashing
Static Dictionaries and Perfect hashing

A static dictionary stores \((key, value)\)-pairs and supports:

\text{lookup}(key) \ (\text{which returns value}) - no inserts or deletes are allowed

Universe \(U\) of \(u\) keys.

Hash table \(T\) of size \(m \geq n\).

Collisions are fixed by chaining

A hash function maps a key \(x\) to position \(h(x)\) - i.e \(T[h(x)] = (key, value)\).

we are given \(n\) different \((key, value)\)-pairs and want to pick a good \(h\)

\textbf{Theorem}

The FKS hashing scheme:
- Has no collisions
- Every lookup takes \(O(1)\) worst-case time,
- Uses \(O(n)\) space,
- Can be built in \(O(n)\) expected time.

The rest of this lecture is devoted to the FKS scheme

The construction is based on weak universal hashing

(with an \(O(1)\) time hash function)
A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr (h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$. 

**Perfect hashing - a first attempt**
A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U \ (x \neq y)$,

$$\Pr (h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$

**Step 1:** Insert everything into a hash table of size $n$ using a weakly universal hash function
Perfect hashing - a first attempt

A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U \ (x \neq y)$,

$$\Pr (h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$

**Step 1:** Insert everything into a hash table of size $n$
using a weakly universal hash function

*(where any $h(x)$ can be computed in $O(1)$ time)*
Perfect hashing - a first attempt

A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U (x \neq y)$,

$$\Pr (h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$

**Step 1:** Insert everything into a hash table of size $n$
using a weakly universal hash function
Perfect hashing - a first attempt

A set \( H \) of hash functions is **weakly universal** if for any two keys \( x, y \in U \) \((x \neq y)\),

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

where \( h \) is picked uniformly at random from \( H \)

**Step 1:** Insert everything into a hash table of size \( n \) using a weakly universal hash function

**Step 2:** Check for collisions
A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr (h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$.

**Step 1:** Insert everything into a hash table of size $n$ using a weakly universal hash function.

**Step 2:** Check for collisions.

**Step 3:** Profit!
A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr(h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$.

**Step 1:** Insert everything into a hash table of size $n$ using a weakly universal hash function.

**Step 2:** Check for collisions.

**Step 3:** *Repeat if necessary.*
Perfect hashing - a first attempt

A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U \ (x \neq y)$,

$$\Pr(h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$

**Step 1:** Insert everything into a hash table of size $n$ using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** Repeat if necessary

*How many collisions do we get on average?*
Perfect hashing - a first attempt

A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U \ (x \neq y)$,

$$\Pr (h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$

---

**Step 1:** Insert everything into a hash table of size $n$
using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** *Repeat if necessary*

**How many collisions do we get on average?**

$$\mathbb{E}(C) = \mathbb{E}(\sum_{x,y \in T, x < y} I_{x,y})$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$. 

---
A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U \ (x \neq y)$,

$$\Pr(h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$

**Step 1:** Insert everything into a hash table of size $n$ using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** Repeat if necessary

**How many collisions do we get on average?**

$$\mathbb{E}(C) = \mathbb{E}(\sum_{x,y \in T, x < y} I_{x,y}) = \sum_{x,y \in T, x < y} \mathbb{E}(I_{x,y})$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$. 
Perfect hashing - a first attempt

A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U \ (x \neq y)$,

$$\Pr (h(x) = h(y)) \leq \frac{1}{m} \quad \text{where } h \text{ is picked uniformly at random from } H$$

**Linearity of Expectation**

Let $Y_1, Y_2, \ldots, Y_k$ be $k$ random variables. Then

$$\mathbb{E} \left( \sum_{i=1}^{k} Y_i \right) = \sum_{i=1}^{k} \mathbb{E}(Y_i)$$

**Number of collisions**

The number of collisions is

$$\mathbb{E}(C) = \mathbb{E} \left( \sum_{x,y \in T, x<y} I_{x,y} \right) = \sum_{x,y \in T, x<y} \mathbb{E}(I_{x,y})$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$. 

Hash table of size $n$ 

Weakly universal hash function

on average?
A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U$ $(x \neq y)$,

$$\Pr(h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$

**Step 1:** Insert everything into a hash table of size $n$ using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** Repeat if necessary

**How many collisions do we get on average?**

$$E(C) = E\left(\sum_{x,y \in T, x < y} I_{x,y}\right) = \sum_{x,y \in T, x < y} E(I_{x,y})$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.
Perfect hashing - a first attempt

A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U \ (x \neq y)$,

$$\Pr (h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$

---

**Step 1:** Insert everything into a hash table of size $n$ using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** *Repeat if necessary*

---

How many collisions do we get on average?

Let $I_{x,y}$ be the indicator random variable such that $I_{x,y} = 1$ iff $h(x) = h(y)$.

$$E(C) = E(\sum_{x,y \in T, x < y} I_{x,y}) = \sum_{x,y \in T, x < y} E(I_{x,y}) \leq \sum_{x,y \in T, x < y} \frac{1}{m}$$

where $E(C)$ is the expected number of collisions.
Perfect hashing - a first attempt

A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U \ (x \neq y)$,

$$\Pr(h(x) = h(y)) \leq \frac{1}{m}$$  

where $h$ is picked uniformly at random from $H$.

---

**Step 1:** Insert everything into a hash table of size $n$ using a weakly universal hash function.

**Step 2:** Check for collisions.

By the definition of expectation...

$$\mathbb{E}(I_{x,y}) = 1 \cdot \Pr(I_{x,y} = 1) + 0 \cdot \Pr(I_{x,y} = 0) \leq \frac{1}{m}$$

number of collisions  

linearity of expectation

$$\mathbb{E}(C) = \mathbb{E}\left( \sum_{x,y \in T, x < y} I_{x,y} \right) = \sum_{x,y \in T, x < y} \mathbb{E}(I_{x,y}) \leq \sum_{x,y \in T, x < y} \frac{1}{m}$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$. 

Perfect hashing - a first attempt

A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr (h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$.

**Step 1:** Insert everything into a hash table of size $n$ using a weakly universal hash function.

**Step 2:** Check for collisions.

**Step 3:** Repeat if necessary.

How many collisions do we get on average?

The number of collisions is given by:

$$\mathbb{E}(C) = \mathbb{E}(\sum_{x,y \in T, x < y} I_{x,y}) = \sum_{x,y \in T, x < y} \mathbb{E}(I_{x,y}) \leq \sum_{x,y \in T, x < y} \frac{1}{m}$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$. 

Diagram:

- A set of hash functions $H$.
- A hash table of size $n$.
- Insertion into the hash table.
- Checking for collisions.
- Repeat if necessary.
A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr (h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$.

**Step 1:** Insert everything into a hash table of size $n$ using a weakly universal hash function.

**Step 2:** Check for collisions.

**Step 3:** Repeat if necessary.

**How many collisions do we get on average?**

The expected number of collisions can be calculated as follows:

$$E(C) = E\left( \sum_{x,y \in T, x<y} I_{x,y} \right) = \sum_{x,y \in T, x<y} E(I_{x,y}) \leq \sum_{x,y \in T, x<y} \frac{1}{m} = \binom{n}{2} \cdot \frac{1}{m}$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$. 
Perfect hashing - a first attempt

A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr (h(x) = h(y)) \leq \frac{1}{m} \quad \text{where } h \text{ is picked uniformly at random from } H$$

Step 1: Insert everything into a hash table of size $n$ using a weakly universal hash function

Step 2: Check for collisions

Step 3: *Repeat if necessary*

How many collisions do we get on average?

\[
E(C) = E\left( \sum_{x,y \in T, x < y} I_{x,y} \right) = \sum_{x,y \in T, x < y} E(I_{x,y}) \leq \sum_{x,y \in T, x < y} \frac{1}{m} = \binom{n}{2} \cdot \frac{1}{m}
\]

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$. 

\[
\binom{n}{2} = \frac{n(n - 1)}{2}
\]
A set $H$ of hash functions is \textbf{weakly universal} if for any two keys $x, y \in U \ (x \neq y)$, 
\[ \Pr (h(x) = h(y)) \leq \frac{1}{m} \] 
where $h$ is picked uniformly at random from $H$

\begin{itemize}
  \item \textbf{Step 1}: Insert everything into a hash table of size $n$ using a weakly universal hash function
  \item \textbf{Step 2}: Check for collisions
  \item \textbf{Step 3}: Repeat if necessary
\end{itemize}

How many collisions do we get on average?

\[ \mathbb{E}(C) = \mathbb{E} \left( \sum_{x,y \in T, x < y} I_{x,y} \right) = \sum_{x,y \in T, x < y} \mathbb{E}(I_{x,y}) \leq \sum_{x,y \in T, x < y} \frac{1}{m} = \binom{n}{2} \cdot \frac{1}{m} \]

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$. 

\[ \binom{n}{2} = \frac{n(n - 1)}{2} \]
Perfect hashing - a first attempt

A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr (h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$

**Step 1:** Insert everything into a hash table of size $n$
using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** *Repeat if necessary*

**How many collisions do we get on average?**

$$\mathbb{E}(C) = \mathbb{E}\left( \sum_{x,y \in T, x < y} I_{x,y} \right) = \sum_{x,y \in T, x < y} \mathbb{E}(I_{x,y}) \leq \sum_{x,y \in T, x < y} \frac{1}{m} = \binom{n}{2} \cdot \frac{1}{m}$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$. 

$$\leq \frac{n^2}{2}$$
Perfect hashing - a first attempt

A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$ \Pr (h(x) = h(y)) \leq \frac{1}{m} $$

where $h$ is picked uniformly at random from $H$

---

**Step 1:** Insert everything into a hash table of size $n$ using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** *Repeat if necessary*

---

**How many collisions do we get on average?**

$$ E(C) = E( \sum_{x,y \in T, x<y} I_{x,y} ) = \sum_{x,y \in T, x<y} E(I_{x,y}) \leq \sum_{x,y \in T, x<y} \frac{1}{m} = \binom{n}{2} \cdot \frac{1}{m} \leq \frac{n^2}{2m} $$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$. 

A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr (h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$.

**Step 1:** Insert everything into a hash table of size $n$ using a weakly universal hash function.

**Step 2:** Check for collisions.

**Step 3:** Repeat if necessary.

**How many collisions do we get on average?**

$$E(C) = E\left( \sum_{x,y \in T, x < y} I_{x,y} \right) = \sum_{x,y \in T, x < y} E(I_{x,y}) \leq \sum_{x,y \in T, x < y} \frac{1}{m} = \left( \binom{n}{2} \right) \cdot \frac{1}{m} \leq \frac{n^2}{2m} \leq \frac{n}{2}.$$  

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$. 
A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr(h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$.

---

**Step 1:** Insert everything into a hash table of size $n^2$ using a weakly universal hash function.

**Step 2:** Check for collisions.

**Step 3:** *Repeat if necessary.*
A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U \ (x \neq y)$,

$$\Pr (h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$.

---

**Step 1:** Insert everything into a hash table of size $n^2$ using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** *Repeat if necessary*
Perfect hashing - a second attempt

A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U \ (x \neq y)$,

$$\Pr(h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$

---

**Step 1:** Insert everything into a hash table of size $n^2$

**Step 2:** Check for collisions

**Step 3:** *Repeat if necessary*

*How many collisions do we get on average?*
Perfect hashing - a second attempt

A set $H$ of hash functions is \textbf{weakly universal} if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr (h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$.

**Step 1:** Insert everything into a hash table of size $n^2$ using a weakly universal hash function.

**Step 2:** Check for collisions.

**Step 3:** Repeat if necessary.

\textit{How many collisions do we get on average?}

$$E(C) = E\left( \sum_{x,y \in T, x < y} I_{x,y} \right) = \sum_{x,y \in T, x < y} E(I_{x,y}) \leq \sum_{x,y \in T, x < y} \frac{1}{m} = \left( \binom{n}{2} \right) \cdot \frac{1}{m} \leq \frac{n^2}{2m}$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$. 
Perfect hashing - a second attempt

A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U \ (x \neq y)$,

$$\Pr(h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$

**Step 1:** Insert everything into a hash table of size $n^2$ using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** Repeat if necessary

**How many collisions do we get on average?**

$$E(C) = E\left(\sum_{x,y \in T, x < y} I_{x,y}\right) = \sum_{x,y \in T, x < y} E(I_{x,y}) \leq \sum_{x,y \in T, x < y} \frac{1}{m} = \left(\binom{n}{2}\right) \cdot \frac{1}{m} \leq \frac{n^2}{2m} \leq \frac{1}{2}$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.
A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U$ $(x \neq y)$,

$$\Pr (h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$.

---

**Step 1:** Insert everything into a hash table of size $n^2$ using a weakly universal hash function.

**Step 2:** Check for collisions.

**Step 3:** Repeat if necessary.

---

**How many collisions do we get on average?**

- Number of collisions:

  \[ E(C) = E\left( \sum_{x,y \in T, x < y} I_{x,y} \right) \]

- Linearity of expectation:

  \[ \sum_{x,y \in T, x < y} E(I_{x,y}) \leq \sum_{x,y \in T, x < y} \frac{1}{m} = \binom{n}{2} \cdot \frac{1}{m} \leq \frac{n^2}{2m} \leq \frac{1}{2} \]

- Definition of expectation:

  where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

much better!
A set $H$ of hash functions is **weakly universal** if for any two keys $x, y \in U$ ($x \neq y$),

$$\Pr(h(x) = h(y)) \leq \frac{1}{m}$$

where $h$ is picked uniformly at random from $H$.

**Step 1:** Insert everything into a hash table of size $n^2$ using a weakly universal hash function.

**Step 2:** Check for collisions.

**Step 3:** Repeat if necessary.

How many collisions do we get on average?

$$E(C) = E\left(\sum_{x,y \in T, x < y} I_{x,y}\right) = \sum_{x,y \in T, x < y} E(I_{x,y}) \leq \sum_{x,y \in T, x < y} \frac{1}{m} = \binom{n}{2} \cdot \frac{1}{m} \leq \frac{n^2}{2m} \leq \frac{1}{2}$$

where indicator random variable $I_{x,y} = 1$ iff $h(x) = h(y)$.

**much better!**
## Expected construction time

<table>
<thead>
<tr>
<th>Step 1:</th>
<th>Insert everything into a hash table of size $m = n^2$ using a weakly universal hash function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 2:</td>
<td>Check for collisions</td>
</tr>
<tr>
<td>Step 3:</td>
<td><em>Repeat if there was a collision</em></td>
</tr>
</tbody>
</table>
Expected construction time

| Step 1: Insert everything into a hash table of size $m = n^2$ using a weakly universal hash function |
| Step 2: Check for collisions |
| Step 3: Repeat if there was a collision |

*How many times do we repeat on average?*
Expected construction time

**Step 1:** Insert everything into a hash table of size $m = n^2$ using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** *Repeat if there was a collision*

*How many times do we repeat on average?*

The expected number of collisions: $\mathbb{E}(C) \leq \frac{1}{2}$

The probability of at least one collision: $\Pr(C \geq 1) \leq \frac{1}{2}$
Expected construction time

Step 1: Insert everything into a hash table of size \( m = n^2 \) using a weakly universal hash function.

Step 2: Check for collisions.

Step 3: Repeat if there was a collision.

How many times do we repeat on average?

The expected number of collisions: \( \mathbb{E}(C) \leq \frac{1}{2} \)

The probability of at least one collision: \( \Pr(C \geq 1) \leq \frac{1}{2} \)

Markov’s inequality

If \( X \) is a non-negative r.v., then for all \( a > 0 \),

\[
\Pr(X \geq a) \leq \frac{\mathbb{E}(X)}{a}.
\]
Expected construction time

**Step 1:** Insert everything into a hash table of size $m = n^2$ using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** *Repeat if there was a collision*

---

*How many times do we repeat on average?*

The expected number of collisions: $\mathbb{E}(C') \leq \frac{1}{2}$

The probability of at least one collision: $\Pr(C \geq 1) \leq \frac{1}{2}$
Expected construction time

**Step 1:** Insert everything into a hash table of size \( m = n^2 \) using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** *Repeat if there was a collision*

How many times do we repeat on average?

The expected number of collisions: \( \mathbb{E}(C) \leq \frac{1}{2} \)

The probability of at least one collision: \( \Pr(C \geq 1) \leq \frac{1}{2} \)

The probability of zero collisions is at least \( \frac{1}{2} \)
Expected construction time

**Step 1:** Insert everything into a hash table of size $m = n^2$ using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** *Repeat if there was a collision*

**How many times do we repeat on average?**

The expected number of collisions: $\mathbb{E}(C) \leq \frac{1}{2}$

The probability of at least one collision: $\Pr(C \geq 1) \leq \frac{1}{2}$

The probability of zero collisions is at least $\frac{1}{2}$

i.e. at least as good as tossing a heads on a fair coin
Expected construction time

**Step 1:** Insert everything into a hash table of size $m = n^2$ using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** *Repeat if there was a collision*

**How many times do we repeat on average?**

The expected number of collisions: $\mathbb{E}(C) \leq \frac{1}{2}

The probability of at least one collision: $\Pr(C \geq 1) \leq \frac{1}{2}$

The probability of zero collisions is at least $\frac{1}{2}$, i.e. at least as good as tossing a heads on a fair coin

$\mathbb{E}(\text{runs}) \leq \mathbb{E}(\text{coin tosses to get a heads}) = 2$
Expected construction time

**Step 1:** Insert everything into a hash table of size $m = n^2$ using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** *Repeat if there was a collision*

*How many times do we repeat on average?*

The expected number of collisions: $\mathbb{E}(C) \leq \frac{1}{2}$  

Markov’s inequality

The probability of at least one collision: $\Pr(C \geq 1) \leq \frac{1}{2}$

The probability of zero collisions is at least $\frac{1}{2}$

*i.e. at least as good as tossing a heads on a fair coin*

$$\mathbb{E}(\text{runs}) \leq \mathbb{E}(\text{coin tosses to get a heads}) = 2$$

$$\mathbb{E}(\text{construction time}) = O(m) \cdot \mathbb{E}(\text{runs}) = O(m) = O(n^2)$$
Expected construction time

**Step 1:** Insert everything into a hash table of size $m = n^2$ using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** Repeat if there was a collision

**How many times do we repeat on average?**

The expected number of collisions: $\mathbb{E}(C) \leq \frac{1}{2}$  

The probability of at least one collision: $\Pr(C \geq 1) \leq \frac{1}{2}$

The probability of zero collisions is at least $\frac{1}{2}$  

*i.e. at least as good as tossing a heads on a fair coin*

\[ \mathbb{E}({\text{runs}}) \leq \mathbb{E}({\text{coin tosses to get a heads}}) = 2 \]

\[ \mathbb{E}({\text{construction time}}) = O(m) \cdot \mathbb{E}({\text{runs}}) = O(m) = O(n^2) \]

\[ \ldots \text{and then the look-up time is always } O(1) \]
Expected construction time

**Step 1:** Insert everything into a hash table of size $m = n^2$ using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** *Repeat if there was a collision*

---

**How many times do we repeat on average?**

The expected number of collisions: $\mathbb{E}(C) \leq \frac{1}{2}$

The probability of at least one collision: $\Pr(C \geq 1) \leq \frac{1}{2}$

The probability of zero collisions is at least $\frac{1}{2}$

*i.e. at least as good as tossing a heads on a fair coin*

$\mathbb{E}(\text{runs}) \leq \mathbb{E}(\text{coin tosses to get a heads}) = 2$

$\mathbb{E}(\text{construction time}) = O(m) \cdot \mathbb{E}(\text{runs}) = O(m) = O(n^2)$

... and then the look-up time is always $O(1)$

*(because any $h(x)$ can be computed in $O(1)$ time)*
Expected construction time

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>Insert everything into a hash table of size ( m = n ) using a weakly universal hash function</td>
</tr>
<tr>
<td>Step 2</td>
<td>Check for collisions</td>
</tr>
<tr>
<td>Step 3</td>
<td>\textit{Repeat if there are more than } ( n ) \textit{ collisions}</td>
</tr>
</tbody>
</table>
Expected construction time

### Step 1:
Insert everything into a hash table of size $m = n$
using a weakly universal hash function.

### Step 2:
Check for collisions.

### Step 3:
*Repeat if there are more than $n$ collisions*

This looks rubbish but it will be useful in a bit!
Expected construction time

**Step 1:** Insert everything into a hash table of size $m = n$ using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** *Repeat if there are more than $n$ collisions*

---

This looks rubbish but it will be useful in a bit!

---

*How many times do we repeat on average?*

The expected number of collisions: $\mathbb{E}(C) \leq \frac{n}{2}$

The probability of at least $n$ collisions: $\Pr(C \geq n) \leq \frac{1}{2}$
Expected construction time

**Step 1:** Insert everything into a hash table of size $m = n$

**Markov’s inequality**

If $X$ is a non-negative r.v., then for all $a > 0$,

$$\Pr(X \geq a) \leq \frac{\mathbb{E}(X)}{a}.$$  

The expected number of collisions: $\mathbb{E}(C) \leq \frac{n}{2}$

The probability of at least $n$ collisions: $\Pr(C \geq n) \leq \frac{1}{2}$ \ (where $a = n$)

This looks rubbish but it will be useful in a bit!
Expected construction time

**Step 1:** Insert everything into a hash table of size $m = n$ using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** *Repeat if there are more than $n$ collisions*

This looks rubbish but it will be useful in a bit!

How many times do we repeat on average?

The expected number of collisions: $\mathbb{E}(C) \leq \frac{n}{2}$

The probability of at least $n$ collisions: $\Pr(C \geq n) \leq \frac{1}{2}$

The probability of at most $n$ collisions is at least $\frac{1}{2}$

i.e. at least as good as tossing a heads on a fair coin

$\mathbb{E}(\text{runs}) \leq \mathbb{E}(\text{coin tosses to get a heads}) = 2$

$\mathbb{E}(\text{construction time}) = O(m) \cdot \mathbb{E}(\text{runs}) = O(m) = O(n)$
Expected construction time

**Step 1:** Insert everything into a hash table of size $m = n$ using a weakly universal hash function

**Step 2:** Check for collisions

**Step 3:** *Repeat if there are more than* $n$ *collisions*

This looks rubbish but it will be useful in a bit!

*How many times do we repeat on average?*

The expected number of collisions: $\mathbb{E}(C) \leq \frac{n}{2}$

The probability of at least $n$ collisions: $\Pr(C \geq n) \leq \frac{1}{2}$

The probability of at most $n$ collisions is at least $\frac{1}{2}$

*i.e. at least as good as tossing a heads on a fair coin*

$\mathbb{E}(\text{runs}) \leq \mathbb{E}(\text{coin tosses to get a heads}) = 2$

$\mathbb{E}(\text{construction time}) = O(m) \cdot \mathbb{E}(\text{runs}) = O(m) = O(n)$

...but the look-up time could be rubbish (lots of collisions)
Perfect hashing - attempt three

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal hash function, $h$. 
Perfect hashing - attempt three

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal hash function, $h$

Let $n_i$ be the number of items in $T[i]$
Perfect hashing - attempt three

Step 1: Insert everything into a hash table, \( T \), of size \( n \) using a weakly universal hash function, \( h \)

Let \( n_i \) be the number of items in \( T[i] \)

\[
\begin{align*}
n_1 &= 2 \\
n_5 &= 2 \\
n_8 &= 3
\end{align*}
\]
Perfect hashing - attempt three

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal hash function, $h$ … but don’t use chaining

Let $n_i$ be the number of items in $T[i]$

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$

*using another weakly universal hash function denoted $h_i$ (there is one for each $i$)*
Perfect hashing - attempt three

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal hash function, $h$ … but don’t use chaining

$T$

Let $n_i$ be the number of items in $T[i]$

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using another weakly universal hash function denoted $h_i$ (there is one for each $i$)
Perfect hashing - attempt three

**Step 1:** Insert everything into a hash table, $T$, of size $n$
using a weakly universal hash function, $h$

... but don’t use chaining

$$h$$

Let $n_i$ be the number of items in $T[i]$.

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$

using another weakly universal hash function denoted $h_i$ (there is one for each $i$)

**Step 3** Immediately repeat a step if either

a) $T$ has more than $n$ collisions

b) some $T_i$ has a collision
Perfect hashing - attempt three

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal hash function, $h$ 

...but don't use chaining

Let $n_i$ be the number of items in $T[i]$  

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using another weakly universal hash function denoted $h_i$ (there is one for each $i$)

**(Step 3)** Immediately repeat a step if either  

a) $T$ has more than $n$ collisions  

b) some $T_i$ has a collision  

i.e. check (and if necessary rebuild) each table immediately after building it
Perfect hashing - attempt three

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal hash function, $h$ … but don’t use chaining

Let $n_i$ be the number of items in $T[i]$

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using another weakly universal hash function denoted $h_i$ (there is one for each $i$)

**(Step 3)** Immediately repeat a step if either
a) $T$ has more than $n$ collisions
b) some $T_i$ has a collision
Perfect hashing - attempt three

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal hash function, $h$ … but don’t use chaining

Let $n_i$ be the number of items in $T[i]$

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using another weakly universal hash function denoted $h_i$ (there is one for each $i$)

(Step 3) Immediately repeat a step if either
a) $T$ has more than $n$ collisions
b) some $T_i$ has a collision

The look-up time is always $O(1)$

1. Compute $i = h(x)$ ($x$ is the key)
2. Compute $j = h_i(x)$
3. The item is in $T_i[j]$
Perfect hashing - attempt three

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal hash function, $h$.

... but don’t use chaining

Let $n_i$ be the number of items in $T[i]$

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using another weakly universal hash function denoted $h_i$ (there is one for each $i$)

(Step 3) Immediately repeat a step if either

a) $T$ has more than $n$ collisions

b) some $T_i$ has a collision

The look-up time is always $O(1)$

1. Compute $i = h(x)$ ($x$ is the key)
2. Compute $j = h_i(x)$
3. The item is in $T_i[j]$

Two questions remain:

What is the expected construction time?

What is the space usage?
Perfect Hashing - Space usage

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$.

(Step 3) *Immediately repeat if either*
  a) $T$ has more than $n$ collisions
  b) some $T_i$ has a collision

*How much space does this use?*
Perfect Hashing - Space usage

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$

(Step 3) *Immediately repeat if either*
    a) $T$ has more than $n$ collisions
    b) some $T_i$ has a collision

*How much space does this use?*

The size of $T$ is $O(n)$
Perfect Hashing - Space usage

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$.

(Step 3) Immediately repeat if either
   a) $T$ has more than $n$ collisions
   b) some $T_i$ has a collision

How much space does this use?

The size of $T$ is $O(n)$

The size of $T_i$ is $O(n_i^2)$
Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$.

(Step 3) Immediately repeat if either
a) $T$ has more than $n$ collisions
b) some $T_i$ has a collision

How much space does this use?

The size of $T$ is $O(n)$

The size of $T_i$ is $O(n_i^2)$

Storing $h_i$ uses $O(1)$ space)
Perfect Hashing - Space usage

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$.

(Step 3) Immediately repeat if either
   a) $T$ has more than $n$ collisions
   b) some $T_i$ has a collision

How much space does this use?

The size of $T$ is $O(n)$

The size of $T_i$ is $O(n_i^2)$

Storing $h_i$ uses $O(1)$ space

So the total space is...
Perfect Hashing - Space usage

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$

**(Step 3)** Immediately repeat if either
  a) $T$ has more than $n$ collisions
  b) some $T_i$ has a collision

*How much space does this use?*

The size of $T$ is $O(n)$

The size of $T_i$ is $O(n_i^2)$

Storing $h_i$ uses $O(1)$ space)

*So the total space is...*

$$O(n) + \sum_i O(n_i^2)$$
Perfect Hashing - Space usage

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$

(Step 3) Immediately repeat if either
a) $T$ has more than $n$ collisions
b) some $T_i$ has a collision

How much space does this use?

The size of $T$ is $O(n)$
The size of $T_i$ is $O(n_i^2)$
Storing $h_i$ uses $O(1)$ space

So the total space is...

$$O(n) + \sum_i O(n_i^2) = O(n) + O\left(\sum_i n_i^2\right)$$
Perfect Hashing - Space usage

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$.

**Step 3** *Immediately repeat if either*

a) $T$ has more than $n$ collisions
b) some $T_i$ has a collision

**How much space does this use?**

The size of $T$ is $O(n)$

The size of $T_i$ is $O(n_i^2)$

Storing $h_i$ uses $O(1)$ space)

So the total space is...

$$O(n) + \sum_i O(n_i^2) = O(n) + O \left( \sum_i n_i^2 \right)$$
Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$.

How much space does this use?

(Step 3) Immediately repeat if either
a) $T$ has more than $n$ collisions
b) some $T_i$ has a collision

The size of $T$ is $O(n)$
The size of $T_i$ is $O(n_i^2)$

So the total space is...

$$O(n) + \sum_i O(n_i^2) = O(n) + O\left(\sum_i n_i^2\right)$$

Storing $h_i$ uses $O(1)$ space.

How big is this? How big is $\sum_i n_i^2$?
Perfect Hashing - Space usage

Step 1: Insert everything into a hash table, \( T \), of size \( n \) using a weakly universal (w.u.) hash function, \( h \).

Step 2: The \( n_i \) items in \( T[i] \) are inserted into another hash table \( T_i \) of size \( n_i^2 \) using w.u hash function \( h_i \).

How much space does this use?

(Step 3) Immediately repeat if either

a) \( T \) has more than \( n \) collisions

b) some \( T_i \) has a collision

The size of \( T \) is \( O(n) \)

The size of \( T_i \) is \( O(n_i^2) \)

So the total space is...

\[
O(n) + \sum_i O(n_i^2) = O(n) + O \left( \sum_i n_i^2 \right)
\]

How big is \( \sum_i n_i^2 \)?

There are \( \binom{n_i}{2} \) collisions in \( T[i] \).
Perfect Hashing - Space usage

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

How much space does this use?

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$.

How big is $\sum_i n_i^2$?

There are $\binom{n_i}{2}$ collisions in $T[i]$ so there are $\sum_i \binom{n_i}{2}$ collisions in $T$.

How many collisions?

The size of $T$ is $O(n)$.

The size of $T_i$ is $O(n_i^2)$.

Storing $h_i$ uses $O(1)$ space.

So the total space is...

$$O(n) + \sum_i O(n_i^2) = O(n) + O \left( \sum_i n_i^2 \right)$$
Perfect Hashing - Space usage

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_{2i}$ using w.u hash function $h_i$

How much space does this use?

(Step 3)

Immediately repeat if either
a) $T$ has more than $n$ collisions
b) some $T_i$ has a collision

The size of $T$ is $O(n)$
The size of $T_i$ is $O(n_{2i})$

So the total space is...

$O(n) + \sum_i O(n_i^2) = O(n) + O \left( \sum_i n_i^2 \right)$

How big is $\sum_i n_i^2$?

There are $\binom{n_i}{2}$ collisions in $T[i]$ so there are $\sum_i \binom{n_i}{2}$ collisions in $T$

Storing $h_i$ uses $O(1)$ space)
Perfect Hashing - Space usage

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u. hash function $h_i$.

How much space does this use?

(Step 3) Immediately repeat if either

a) $T$ has more than $n$ collisions

b) some $T_i$ has a collision

The size of $T$ is $O(n)$

The size of $T_i$ is $O(n_i^2)$

So the total space is... $O(n) + \sum_i O(n_i^2) = O(n) + O \left( \sum_i n_i^2 \right)$

Storing $h_i$ uses $O(1)$ space.

How big is $\sum_i n_i^2$?

There are $\binom{n_i}{2}$ collisions in $T[i]$ so there are $\sum_i \binom{n_i}{2}$ collisions in $T$.

but we know that there are at most $n$ collisions in $T$ ...
Perfect Hashing - Space usage

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_{2i}$ using w.u hash function $h_i$

How much space does this use?

(Step 3)

Immediately repeat if either

a) $T$ has more than $n$ collisions
b) some $T_i$ has a collision

The size of $T$ is $O(n)$

The size of $T_i$ is $O(n_{2i})$

So the total space is...

$$O(n) + \sum_i O(n_i^2) = O(n) + O\left(\sum_i n_i^2\right)$$
Perfect Hashing - Space usage

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$.

How much space does this use?

(Step 3)

Immediately repeat if either

a) $T$ has more than $n$ collisions
b) some $T_i$ has a collision

The size of $T$ is $O(n)$

The size of $T_i$ is $O(n_i^2)$

So the total space is...

$$O(n) + \sum_i O(n_i^2) = O(n) + O\left(\sum_i n_i^2\right)$$

Storing $h_i$ uses $O(1)$ space

How big is $\sum_i n_i^2$?

There are $\binom{n_i}{2}$ collisions in $T[i]$ so there are $\sum_i \binom{n_i}{2}$ collisions in $T$...

but we know that there are at most $n$ collisions in $T$...

$$\sum_i \frac{n_i^2}{4} \leq \sum_i \binom{n_i}{2} \leq n$$

how big is this?
Perfect Hashing - Space usage

**Step 1:** Insert everything into a hash table, \( T \), of size \( n \) using a weakly universal (w.u.) hash function, \( h \).

**How much space does this use?**

(Step 3) Immediately repeat if either

a) \( T \) has more than \( n \) collisions

b) some \( T[i] \) has a collision

The size of \( T \) is \( O(n) \)

The size of \( T[i] \) is \( O(n^2) \)

So the total space is...

\[
O(n) + \sum_i O(n^2) = O(n) + O\left(\sum_i n^2\right)
\]
Perfect Hashing - Space usage

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$.

How much space does this use?

(Step 3) Immediately repeat if either

a) $T$ has more than $n$ collisions
b) some $T_i$ has a collision

The size of $T$ is $O(n)$

The size of $T_i$ is $O(n_i^2)$

So the total space is...

$O(n) + \sum_i O(n_i^2) = O(n) + O\left(\sum_i n_i^2\right)$

Storing $h_i$ uses $O(1)$ space

How big is $\sum_i n_i^2$?

There are $\binom{n_i}{2}$ collisions in $T[i]$ so there are $\sum_i \binom{n_i}{2}$ collisions in $T$.

but we know that there are at most $n$ collisions in $T$ . . .

$\sum_i \frac{n_i^2}{4} \leq \sum_i \binom{n_i}{2} \leq n$

how big is this?
Perfect Hashing - Space usage

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_{i^2}$ using w.u hash function $h_i$.

How much space does this use?

(Step 3)

- Immediately repeat if either
  a) $T$ has more than $n$ collisions
  b) some $T_i$ has a collision

The size of $T$ is $O(n)$

The size of $T_i$ is $O(n_{i^2})$

So the total space is...

$$O(n) + \sum_i O(n_i^2) = O(n) + O\left(\sum_i n_i^2\right)$$

Storing $h_i$ uses $O(1)$ space

How big is $\sum_i n_i^2$?

There are $\binom{n_i}{2}$ collisions in $T[i]$ so there are $\sum_i \binom{n_i}{2}$ collisions in $T$

but we know that there are at most $n$ collisions in $T$ . . .

$$\sum_i \frac{n_i^2}{4} \leq \sum_i \binom{n_i}{2} \leq n \quad \text{or} \quad \sum_i n_i^2 \leq 4n$$

How big is $\sum_i n_i^2$?

There are $\binom{n_i}{2}$ collisions in $T[i]$ so there are $\sum_i \binom{n_i}{2}$ collisions in $T$

but we know that there are at most $n$ collisions in $T$ . . .

$$\sum_i \frac{n_i^2}{4} \leq \sum_i \binom{n_i}{2} \leq n \quad \text{or} \quad \sum_i n_i^2 \leq 4n$$

So the total space is...

$$O(n) + \sum_i O(n_i^2) = O(n) + O\left(\sum_i n_i^2\right)$$

Storing $h_i$ uses $O(1)$ space

how big is this?
Perfect Hashing - Space usage

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

How big is $\sum_i n_i^2$?

There are $\binom{n_i}{2}$ collisions in $T[i]$ so there are $\sum_i \binom{n_i}{2}$ collisions in $T$.

but we know that there are at most $n$ collisions in $T$.

\[ \sum_i \frac{n_i^2}{4} \leq \sum_i \binom{n_i}{2} \leq n \]

or $\sum_i n_i^2 \leq 4n$

How much space does this use?

Step 3: Immediately repeat if either

a) $T$ has more than $n$ collisions
b) some $T[i]$ has a collision

The size of $T$ is $O(n)$

The size of $T[i]$ is $O(n_i^2)$

so the total space is...

$O(n) + \sum_i O(n_i^2) = O(n) + O \left( \sum_i n_i^2 \right) = O(n)$

Storing $h_i$ uses $O(1)$ space.

How big is this?
**Perfect Hashing - Space usage**

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$

**(Step 3) Immediately repeat if either**
- a) $T$ has more than $n$ collisions
- b) some $T_i$ has a collision

*How much space does this use?*

The size of $T$ is $O(n)$

The size of $T_i$ is $O(n_i^2)$

Storing $h_i$ uses $O(1)$ space)

So the total space is...

$$O(n) + \sum_i O(n_i^2) = O(n) + O \left( \sum_i n_i^2 \right) = O(n)$$
Perfect Hashing - Expected construction time

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$

(Step 3) *Immediately repeat if either*

a) $T$ has more than $n$ collisions
b) some $T_i$ has a collision
Perfect Hashing - Expected construction time

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$.

**Step 3** *(Immediately repeat if either)*

- a) $T$ has more than $n$ collisions
- b) some $T_i$ has a collision

The expected construction time for $T$ is $O(n)$

*(we considered this on a previous slide)*
Perfect Hashing - Expected construction time

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$

(Step 3) *Immediately repeat if either*
   a) $T$ has more than $n$ collisions
   b) some $T_i$ has a collision

The expected construction time for $T$ is $O(n)$

*(we considered this on a previous slide)*

The expected construction time for each $T_i$ is $O(n_i^2)$
Perfect Hashing - Expected construction time

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$.

(Step 3) Immediately repeat if either
   a) $T$ has more than $n$ collisions
   b) some $T_i$ has a collision

The expected construction time for $T$ is $O(n)$

(we considered this on a previous slide)

The expected construction time for each $T_i$ is $O(n_i^2)$

- we insert $n_i$ items into a table of size $m = n_i^2$
- then repeat if there was a collision
Perfect Hashing - Expected construction time

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$

(Step 3) Immediately repeat if either
   a) $T$ has more than $n$ collisions
   b) some $T_i$ has a collision

The expected construction time for $T$ is $O(n)$
   (we considered this on a previous slide)

The expected construction time for each $T_i$ is $O(n_i^2)$
   - we insert $n_i$ items into a table of size $m = n_i^2$
   - then repeat if there was a collision
     (we also considered this on a previous slide)
Perfect Hashing - Expected construction time

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$.

(Step 3) *Immediately repeat if either*

a) $T$ has more than $n$ collisions
b) some $T_i$ has a collision

The expected construction time for $T$ is $O(n)$
*(we considered this on a previous slide)*

The expected construction time for each $T_i$ is $O(n_i^2)$

- we insert $n_i$ items into a table of size $m = n_i^2$
- then repeat if there was a collision
  *(we also considered this on a previous slide)*

The overall expected construction time is therefore:

$$
\mathbb{E}(\text{construction time}) = \mathbb{E}
\left(\text{construction time of } T + \sum_i \text{construction time of } T_i\right)
$$
Perfect Hashing - Expected construction time

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$.

(Step 3) *Immediately repeat if either*

a) $T$ has more than $n$ collisions

b) some $T_i$ has a collision

The expected construction time for $T$ is $O(n)$.
The expected construction time for each $T_i$ is $O(n_i^2)$.
The overall expected construction time is therefore:

$$\mathbb{E}(\text{construction time}) = \mathbb{E}(\text{construction time of } T + \sum_i \text{construction time of } T_i)$$
Perfect Hashing - Expected construction time

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$.

(Step 3) *Immediately repeat if either*
  
  a) $T$ has more than $n$ collisions  
  b) some $T_i$ has a collision

The expected construction time for $T$ is $O(n)$.

The expected construction time for each $T_i$ is $O(n_i^2)$.

The overall expected construction time is therefore:

$$
\mathbb{E}(\text{construction time}) = \mathbb{E}\left( \text{construction time of } T + \sum_i \text{construction time of } T_i \right)
$$

$$
= \mathbb{E}(\text{construction time of } T) + \sum_i \mathbb{E}(\text{construction time of } T_i)
$$
Perfect Hashing - Expected construction time

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$.

**Step 3** *Immediately repeat if either*

a) $T$ has more than $n$ collisions

b) some $T_i$ has a collision

The expected construction time for $T$ is $O(n)$.

The expected construction time for each $T_i$ is $O(n_i^2)$.

The overall expected construction time is therefore:

$$
\mathbb{E}(\text{construction time}) = \mathbb{E} \left( \text{construction time of } T + \sum_i \text{construction time of } T_i \right)
$$

$$
= \mathbb{E}(\text{construction time of } T) + \sum_i \mathbb{E}(\text{construction time of } T_i)
$$

$$
= O(n) + \sum_i O(n_i^2) = O(n) + O \left( \sum_i n_i^2 \right)
$$
Perfect Hashing - Expected construction time

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$.

**Step 3** *Immediately repeat if either*
- a) $T$ has more than $n$ collisions
- b) some $T_i$ has a collision

The expected construction time for $T$ is $O(n)$.
The expected construction time for each $T_i$ is $O(n_i^2)$.
The overall expected construction time is therefore:

$$
E(\text{construction time}) = E \left( \text{construction time of } T + \sum_i \text{construction time of } T_i \right) \\
= E(\text{construction time of } T) + \sum_i E(\text{construction time of } T_i) \\
= O(n) + O\left(\sum_i n_i^2\right) = O(n) + O\left(\sum_i n_i^2\right) = O(n)
$$
Perfect Hashing - Expected construction time

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$.

(Step 3) Immediately repeat if either
   a) $T$ has more than $n$ collisions
   b) some $T_i$ has a collision

The expected construction time for $T$ is $O(n)$.
The expected construction time for each $T_i$ is $O(n_i^2)$.
The overall expected construction time is therefore:

$$
\mathbb{E}(\text{construction time}) = \mathbb{E}\left(\text{construction time of } T + \sum_i n_i^2 \leq 4n \text{ construction time of } T_i\right)
$$

$$
= \mathbb{E}(\text{construction time of } T) + \sum_i \mathbb{E}(\text{construction time of } T_i)
$$

$$
= O(n) + \sum_i O(n_i^2) = O(n) + O\left(\sum_i n_i^2\right) = O(n)
$$
Perfect Hashing - Expected construction time

Step 1: Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$

Step 2: The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$

(Step 3) Immediately repeat if either
a) $T$ has more than $n$ collisions
b) some $T_i$ has a collision

The expected construction time for $T$ is $O(n)$
The expected construction time for each $T_i$ is $O(n_i^2)$
The overall expected construction time is therefore:

$$
\mathbb{E}(\text{construction time}) = \mathbb{E}(\text{construction time of } T + \sum_i \text{construction time of } T_i)
$$

$$
= \mathbb{E}(\text{construction time of } T) + \sum_i \mathbb{E}(\text{construction time of } T_i)
$$

$$
= O(n) + \sum_i O(n_i^2) = O(n) + O \left( \sum_i n_i^2 \right) = O(n)
$$
Perfect Hashing - Summary

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$

**Step 3** *Immediately repeat if either*

a) $T$ has more than $n$ collisions

b) some $T_i$ has a collision

**Theorem**

The FKS hashing scheme:

- Has no collisions
- Every lookup takes $O(1)$ worst-case time,
- Uses $O(n)$ space,
- Can be built in $O(n)$ expected time.

*The look-up time is always $O(1)$*

1. Compute $i = h(x)$ ($x$ is the key)
2. Compute $j = h_i(x)$
3. The item is in $T_i[j]$
Perfect Hashing - Summary

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$.

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$.

**Step 3** *Immediately repeat if either*

a) $T$ has more than $n$ collisions

b) some $T_i$ has a collision

---

**Theorem**

The FKS hashing scheme:

- Has no collisions
- Every lookup takes $O(1)$ worst-case time,
- Uses $O(n)$ space,
- Can be built in $O(n)$ expected time.

The look-up time is always $O(1)$

1. Compute $i = h(x)$ ($x$ is the key)
2. Compute $j = h_i(x)$
3. The item is in $T_i[j]$

In fact this scheme can be made dynamic with $O(1)$ expected time inserts and deletes.
Perfect Hashing - Summary

**Step 1:** Insert everything into a hash table, $T$, of size $n$ using a weakly universal (w.u.) hash function, $h$

**Step 2:** The $n_i$ items in $T[i]$ are inserted into another hash table $T_i$ of size $n_i^2$ using w.u hash function $h_i$

**(Step 3) Immediately repeat if either**

a) $T$ has more than $n$ collisions
b) some $T_i$ has a collision

---

**THEOREM**

The FKS hashing scheme:

- Has no collisions
- Every lookup takes $O(1)$ worst-case time,
- Uses $O(n)$ space,
- Can be built in $O(n)$ expected time.

**The look-up time is always $O(1)$**

1. Compute $i = h(x)$ ($x$ is the key)
2. Compute $j = h_i(x)$
3. The item is in $T_i[j]$

In fact this scheme can be made dynamic with $O(1)$ expected time inserts and deletes but occasionally the inserts take $\Theta(n)$ time.
Longest chain – true randomness (proof omitted from last time)

**Lemma**

If $h$ is selected uniformly at random from all functions $U \to [m]$ then, over $m$ fixed inputs,

$$\Pr (\text{any chain has length } \geq 3 \log m ) \leq \frac{1}{m}.$$
If $h$ is selected uniformly at random from all functions $U \rightarrow [m]$ then, over $m$ fixed inputs,

$$\Pr(\text{any chain has length } \geq 3 \log m) \leq \frac{1}{m}.$$
If \( h \) is selected uniformly at random from all functions \( U \to [m] \) then, over \( m \) fixed inputs,

\[
\Pr(\text{any chain has length } \geq 3 \log m) \leq \frac{1}{m}.
\]

**Observe**

In this lemma we insert \( m \) keys, i.e. \( n = m \).

**Proof**

The problem is equivalent to showing that if we randomly throw \( m \) balls into \( m \) bins, the probability of having a bin with at least \( 3 \log m \) balls is at most \( \frac{1}{m} \).
PROOF

continued…

Let $X_1$ be the number of balls in the first bin.

On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.
Let $X_1$ be the number of balls in the first bin.

On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.

For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$. 
Let $X_1$ be the number of balls in the first bin.

On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.

For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$.

So, the union bound gives

$$\Pr(X_1 \geq k) \leq \binom{m}{k} \cdot \frac{1}{m^k} \leq \frac{1}{k!}.$$
Let $X_1$ be the number of balls in the first bin.

On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.

For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$.

So, the union bound gives

$$\Pr(X_1 \geq k) \leq \binom{m}{k} \cdot \frac{1}{m^k} \leq \frac{1}{k!}.$$
Let $X_1$ be the number of balls in the first bin.

On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.

For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$.

So, the union bound gives

$$\Pr(X_1 \geq k) \leq \binom{m}{k} \cdot \frac{1}{m^k} \leq \frac{1}{k!}.$$
Proof (continued)...

- Let $X_1$ be the number of balls in the first bin.
- On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.
- For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$.
- So, the union bound gives

$$
\Pr(X_1 \geq k) \leq \binom{m}{k} \cdot \frac{1}{m^k} \leq \frac{1}{k!}.
$$
Longest chain – true randomness

\textbf{Proof (continued...)}

- Let $X_1$ be the number of balls in the first bin.
- On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.
- For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$.
- So, the union bound gives

$$
\Pr(X_1 \geq k) \leq \binom{m}{k} \cdot \frac{1}{m^k} \leq \frac{1}{k!}.
$$
Let $X_1$ be the number of balls in the first bin.

On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.

For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$.

So, the union bound gives

$$\Pr(X_1 \geq k) \leq \binom{m}{k} \cdot \frac{1}{m^k} \leq \frac{1}{k!}.$$

\[
\binom{m}{k} = \frac{m!}{k!(m-k)!} = \frac{1}{k!} \cdot \frac{m \times (m-1) \times (m-2) \ldots \times 1}{(m-k) \times (m-k-1) \times (m-k-2) \ldots \times 1} = \frac{1}{k!} \cdot m \times (m-1) \times \ldots \times (m-k+1) \leq \frac{m^k}{k!}
\]
Let $X_1$ be the number of balls in the first bin. On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly. For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$. So, the union bound gives

$$\Pr(X_1 \geq k) \leq \binom{m}{k} \cdot \frac{1}{m^k} \leq \frac{1}{k!}.$$
Let $X_1$ be the number of balls in the first bin.

On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.

For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$.

So, the union bound gives

$$\Pr(X_1 \geq k) \leq \binom{m}{k} \cdot \frac{1}{m^k} \leq \frac{1}{k!}.$$

Using the union bound again, we have

$$\Pr(\text{at least one bin receives at least } k \text{ balls}) \leq m \cdot \Pr(X_1 \geq k) \leq \frac{m}{k!}.$$
Let $X_1$ be the number of balls in the first bin.

On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.

For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$.

So, the union bound gives

$$\Pr(X_1 \geq k) \leq \binom{m}{k} \cdot \frac{1}{m^k} \leq \frac{1}{k!}.$$ 

Using the union bound again, we have

$$\Pr(\text{at least one bin receives at least } k \text{ balls}) \leq m \cdot \Pr(X_1 \geq k) \leq \frac{m}{k!}.$$ 

**Union Bound**

Let $V_1, \ldots, V_q$ be $q$ events. Then

$$\Pr\left( \bigcup_{i=1}^{q} V_i \right) \leq \sum_{i=1}^{q} \Pr(V_i).$$

now $V_i$ is the event that at least $k$ balls go into the $i$-th bin
Let $X_1$ be the number of balls in the first bin.

On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.

For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$.

So, the union bound gives

$$
\Pr(X_1 \geq k) \leq {m \choose k} \cdot \frac{1}{m^k} \leq \frac{1}{k!}.
$$

Using the union bound again, we have

$$
\Pr(\text{at least one bin receives at least } k \text{ balls}) \leq m \cdot \Pr(X_1 \geq k) \leq \frac{m}{k!}.
$$
Let $X_1$ be the number of balls in the first bin.

On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.

For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$.

So, the union bound gives

$$\Pr(X_1 \geq k) \leq \binom{m}{k} \cdot \frac{1}{m^k} \leq \frac{1}{k!}.$$

Using the union bound again, we have

$$\Pr(\text{at least one bin receives at least } k \text{ balls}) \leq m \cdot \Pr(X_1 \geq k) \leq \frac{m}{k!}.$$

Now we set $k = 3 \log m$ and observe that $\frac{m}{k!} \leq \frac{1}{m}$ for $m \geq 2$, and we are done.
Let $X_1$ be the number of balls in the first bin.

On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.

For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$.

So, the union bound gives

$$\Pr(X_1 \geq k) \leq \frac{m}{k!} \cdot \frac{1}{m^k} \leq \frac{1}{k!}.$$ 

Using the union bound again, we have

$$\Pr(\text{at least one bin receives at least } k \text{ balls}) \leq \frac{m}{k!}.$$ 

Why is $\frac{m}{k!} \leq \frac{1}{m}$? (when $k = 3 \log m$)

Now we set $k = 3 \log m$ and observe that $\frac{m}{k!} \leq \frac{1}{m}$ for $m \geq 2$, and we are done.
Let $X_1$ be the number of balls in the first bin. On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.

For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$. 

Now we set $k = 3 \log m$ and observe that 

\[
\frac{m}{k!} \leq \frac{1}{m}
\]

for $m \geq 2$, and we are done.
Let $X_1$ be the number of balls in the first bin.

On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.

For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$.

Now we set $k = 3 \log m$ and observe that $m^k \leq \frac{1}{m}$ for $m \geq 2$, and we are done.

Using the union bound again, we have

$$\Pr(\text{at least one bin receives at least } k \text{ balls}) \leq m \cdot \Pr(X_1 \geq k) \leq m^k \leq \frac{1}{k!}.$$
Let $X_1$ be the number of balls in the first bin.

On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.

For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$.

Now we set $k = 3 \log m$ and observe that $m^k! \leq \frac{1}{m}$ for $m \geq 2$.

\[ \Pr(\text{at least one bin receives at least } k \text{ balls}) \leq m \cdot \Pr(X_1 \geq k) \leq m^k \cdot \frac{1}{m^k} \leq \frac{1}{k!}. \]

Using the union bound again, we have

\[ \Pr(X_1 \geq k) \leq \left( m^k \right) \cdot \frac{1}{m^k} \leq \frac{1}{k!}. \]

Now we set $k = 3 \log m$ and observe that $m^k! \leq \frac{1}{m}$ for $m \geq 2$, and we are done.
Let $X_1$ be the number of balls in the first bin.

On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.

For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$.

Now we set $k = 3 \log m$ and observe that $m^k \leq \frac{1}{m}$ for $m \geq 2$, and we are done.

**Proof continued...**

**Why is $\frac{m}{k!} \leq \frac{1}{m}$? (when $k = 3 \log m$)**

$k! = k \times (k - 1) \times (k - 2) \times \ldots \times 2 \times 1$

$k! > 2 \times 2 \times 2 \times \ldots \times 2 \times 1 = 2^{k-1}$

Let $k = 3 \log m$ ...

$k! > 2^{(3 \log m - 1)} \geq 2^{2 \log m} = (2^{\log m})^2 = m^2$

Using the union bound, we have

$\Pr(\text{at least one bin receives at least } k \text{ balls}) \leq \frac{m}{k!}$.

Now we set $k = 3 \log m$ and observe that $\frac{m}{k!} \leq \frac{1}{m}$ for $m \geq 2$, and we are done.
Let $X_1$ be the number of balls in the first bin. On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.

For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$.

Now we set $k = 3 \log m$ and observe that $k! \leq 1 \cdot 2 \cdot 2 \cdot 2 \ldots \cdot 2 \cdot 1 = 2^{k-1}$.

Let $k = 3 \log m$ ...

$$k! > 2^{(3 \log m - 1)} \geq 2^{2 \log m} = (2^{\log m})^2 = m^2$$

So $\frac{m}{k!} \leq \frac{m}{m^2} = \frac{1}{m}$.

Now we set $k = 3 \log m$ and observe that $\frac{m}{k!} \leq \frac{1}{m}$ for $m \geq 2$, and we are done.
Proof continued...

- Let $X_1$ be the number of balls in the first bin.
- On the event $X_1 \geq k$, one can find a subset of size $k$ of the balls such that all these balls are in the first bin. We will choose $k$ shortly.
- For $k$ given balls, they go into the first bin with probability $\frac{1}{m^k}$.
- So, the union bound gives
  \[
  \Pr(X_1 \geq k) \leq \binom{m}{k} \cdot \frac{1}{m^k} \leq \frac{1}{k!}.
  \]
- Using the union bound again, we have
  \[
  \Pr(\text{at least one bin receives at least } k \text{ balls}) \leq m \cdot \Pr(X_1 \geq k) \leq \frac{m}{k!}.
  \]
- Now we set $k = 3 \log m$ and observe that $\frac{m}{k!} \leq \frac{1}{m}$ for $m \geq 2$, and we are done.
**Lemma**

If $h$ is selected uniformly at random from all functions $U \to [m]$ then, over $m$ fixed inputs,

$$\Pr (\text{any chain has length } \geq 3 \log m ) \leq \frac{1}{m}.$$ 

**Observe**

In this lemma we insert $m$ keys, i.e. $n = m$.

**Proof**

The problem is equivalent to showing that if we randomly throw $m$ balls into $m$ bins, the probability of having a bin with at least $3 \log m$ balls is at most $\frac{1}{m}$.