We want to store $n$ elements from the universe, $U$ in a dictionary. Typically $n = |U|$ is much, much larger than $m$.

**THEOREM**

We write $[m]$ to denote the set $\{0, \ldots, m - 1\}$.\footnote{Consider any $n$ fixed inputs to the hash table (which has size $m$). Pick $h$ uniformly at random from the set of all functions $U \rightarrow [m]$. The expected run-time per operation is $O(1 + \frac{1}{m})$, or simply $O(1)$ if $m \geq n$.}

**Proof**

Let $x, y$ be two distinct keys from $U$. Let indicator r.v. $I_{x,y}$ be 1 if $h(x) = h(y)$, we have that, $Pr(h(x) = h(y)) = \frac{1}{m}$.

This is because $h(x)$ and $h(y)$ are chosen uniformly and independently from $[m]$. Therefore, $E(I_{x,y}) = Pr(I_{x,y} = 1) = Pr(h(x) = h(y)) = \frac{1}{m}$.

We have that, $E(I_{x,y}) = \frac{1}{m}$

---

### Time complexity

We cannot avoid collisions entirely since $u \gg m$; some keys from the universe are bound to be mapped to the same position. (remember $u$ is the size of the universe and $m$ is the size of the table).

By building a hash table with chaining, we get the following time complexities:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Worst case time</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>add$(x, v)$</td>
<td>$O(1)$</td>
<td>Simply add item to the list link if necessary.</td>
</tr>
<tr>
<td>lookup$(x)$</td>
<td>$O(\text{length of chain containing } x)$</td>
<td>We might have to search through the whole list containing $x$.</td>
</tr>
<tr>
<td>delete$(x)$</td>
<td>$O(\text{length of chain containing } x)$</td>
<td>Only $O(1)$ to perform the actual delete... but you have to find $x$ first</td>
</tr>
</tbody>
</table>

So how long are these chains?

---

### True randomness

Consider any $n$ fixed inputs to the hash table (which has size $m$), i.e. any sequence of $n$ add/lookup/delete operations.

Pick $h$ uniformly at random from the set of all functions $U \rightarrow [m]$. The expected run-time per operation is $O(1 + \frac{1}{m})$, or simply $O(1)$ if $m \geq n$.

**Proof**

Let $x, y$ be two distinct keys from $U$. Let indicator r.v. $I_{x,y}$ be 1 if $h(x) = h(y)$, we have that, $E(I_{x,y}) = \frac{1}{m}$.

Let $N_k$ be the number of keys stored in $T$ that are hashed to $h(x)$, so in the worst case it takes $N_k$ time to look up $x$ in $T$.

Observe that $N_k = \sum_{y \in T} I_{x,y}$, the keys in $T$.

Finally, we have that $E(N_k) = \sum_{y \in T} E(I_{x,y}) = \sum_{y \in T} E(I_{x,y}) = \frac{n}{m}$.

Linearity of expectation.
For both,

**Problem:** how do we specify an arbitrary (e.g., a truly random) hash function?

For each key in \( U \) we need to specify an arbitrary position in \( T \).

This is a number in \([m]\), so requires \( \log_2 m \) bits.

So in total we need \( u \log_2 m \) bits, which is a ridiculous amount of space!

\((\text{in particular, it’s much bigger than the table :s})\)

Why not pick the hash function as we go?

Could we not generate \( h(x) \) when we first see \( x \)?

Wouldn’t we only use \( n \log_2 m \) bits? (one per key we actually store)

The problem with this approach is recalling \( h(x) \) the next time we see \( x \)

Essentially we’d need to build a dictionary to solve the dictionary problem!

This has become rather cyclic... let’s try something else!

---

### Weakly universal hashing

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

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

where \( h \) is chosen uniformly at random from \( H \).

**Theorem**

Consider any \( n \) fixed inputs to the hash table (which has size \( m \)).

Let \( H \) be a weakly universal set of hash functions.

The expected run-time per operation is \( O(1) \) if \( m \geq n \).

**Proof**

The proof we used for true randomness works here too (which is nice)

---

### True randomness vs. weakly universal hashing

For both,

- **true randomness**
  - \( h \) is picked uniformly from the set of all possible hash functions
  - and **weakly universal hashing**
  - \( h \) is picked uniformly from a weakly universal set of hash functions

we have seen that when \( m \geq n \),

the expected lookup time in the hash table is \( O(1) \).

Since constructing a weakly universal set of hash functions seems much easier than obtaining true randomness, this is all good news!

Isn’t it?

What about the length of the longest chain? (the longest linked list)

If it is very long, some lookups could take a very long time...

---

### Constructing a weakly universal family of hash functions

- Suppose \( U = [n] \), i.e., the keys in the universe are integers \( 0 \) to \( n - 1 \).

- Let \( p \) be any prime bigger than \( n \).

- For \( a, b \in [p] \), let

\[
h_{a,b}(x) = (ax + b \mod p) \mod m,
\]

where \( a \neq 0 \) and \( m \) is a prime number.

\( H_{a,b} = \{h_{a,b} | a \in \{1, \ldots, p-1\}, b \in \{0, \ldots, p-1\}\} \)

**Theorem**

\( H_{a,b} \) is a weakly universal set of hash functions.

**Proof**


---

### Longest chain – true randomness

**Lemma**

If \( h \) is selected uniformly at random from all functions \( U \to [m] \),

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

**Observation**

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

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

This rubbish upper bound of $\frac{1}{m}$ does not necessarily rule out the possibility that the tightest upper bound is indeed very small. However, the upper bound of $\frac{1}{m}$ is in fact tight.
Proof:
- For any two keys $x, y$, let indicator r.v. $I_{x,y}$ be 1 if $h(x) = h(y)$.
- Let $r.v. C$ be the total number of collisions: $C = \sum_{x,y \in T} I_{x,y}$.
- Using linearity of expectation and $E(I_{x,y}) = \frac{1}{m}$ (h is weakly universal),
  $$E(C) = \sum_{x,y \in T} E(I_{x,y}) = \left( \frac{m}{2} \right) \cdot \frac{1}{m} \leq \frac{m}{2}.$$  
- by Markov’s inequality, $Pr(C \geq m) \leq \frac{E(C)}{m} \leq \frac{1}{2}$.
- Let r.v. $L$ be the length of the longest chain. Then $C \geq \left( \frac{L}{2} \right)$.

  This is because a chain of length $L$ causes $\left( \frac{L}{2} \right)$ collisions.

Conclusions

For both, true randomness ($h$ is picked uniformly from the set of all possible hash functions) and weakly universal hashing ($h$ is picked uniformly from a weakly universal set of hash functions)

we have seen that when $m > n$, the expected lookup time in a hash table with chaining is $O(1)$.

Lemma

If $h$ is selected uniformly at random from all functions $U \rightarrow [m]$ then,

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

Lemma

If $h$ is picked uniformly at random from a weakly universal set of hash functions,

$$Pr(\text{any chain has length} \geq 1 + \sqrt{2m}) \leq \frac{1}{2}.$$  

(both Lemmas hold for $m$ any fixed inputs)