OH TODAY: 1:15pm - 2:15pm.

**Hashing.**

data structure that is used to support dynamic set operations: Insert, Delete, Search.

**Direct Addressing (Array).**

\[ U \]

\[ T[0..k] \]

**Insert \((T, x)\):**

\[ T[0..k] \leftarrow x \]

\(O(1) + m\)
Search \((T, k)\)

return \(T[k]\)

Delete \((T, x)\)

\(T[x, \text{key}] \leftarrow \text{NIL}\)

What if \(|U|\) is very large?

\(|U| \gg \# \text{keys stored}\)

Hash table (hashy)
Since \(|U| > |T|\), by PHP two keys will be mapped (hashed) to the same location in \(T\). Multiple keys being mapped to the same location is called collision.

Hashing with Chainiing.

![Diagram showing chaining]

- Insert \((T, x)\) at the head of \(T\)
- Insert \(x\) \(\langle \ldots \rangle\) into the linked list \(T[h(x, key)]\)
Search ($T, k$)

- Search obj with key $k$ in the
  linked list $T[h(k)]$.

Delete ($T, x$)

- we have handle to $x$.
- delete $x$ from the doubly linked list $T[h(x, key)]$.

Analysis:

$m : |T|.$

$m : \# \text{ elements stored in } T.$
$\alpha$: load factor given by $n/m$.

**Simple Uniform Hashing Assumption (SUHA):**

"Our hash fn is a good one": Each key is equally likely to be hashed (mapped) to any of the $m$ slots in $T$.

**Unsuccessful Search:**

**Lemma:** In Hashing w/ chaining, assuming SUHA, unsuccessful search takes $\Theta(1 + \alpha)$ time on expectation.

**Proof:** Suppose we are searching key $k$.

$$E[\text{time}] = \sum P(\text{k is mapped to } i) \cdot E[\text{time} | \text{k is mapped to } i]$$
\[
\sum_{i=0}^{m-1} \frac{1}{m} \cdot E[|T[i]|]
\]

\[
= \sum_{i=0}^{m-1} \frac{1}{m} \cdot \frac{n}{m} \left( \log n \cdot X_f = i, \text{ if element \(j\) is mapped to \(i\)} \right)
\]

\[
= m \cdot \frac{n}{m^2} = \frac{n}{m} = \alpha.
\]

Assume \( h(i_c) \) takes \( O(1) \) time,

\[E[\text{time}] = \Theta(1 + \alpha).\]

**Successful Search.**

\[\Theta(1 + \alpha)\]
we will assume that we are equally likely to search for any of the $n$ elements on the table.

Let $x_1, x_2, \ldots, x_n$ be the order in which elements are inserted into $T$.

Note that since elements are inserted at the head of the list, to find $x_i$, we have to pass through all elements from $x_{i+1}, x_{i+2}, \ldots, x_n$ that are mapped to the same slot as $x_i$. 
\[ \mathbb{E}[\text{time}] = \sum_{i=1}^{m} p(x_i \text{ is searched}) \cdot \mathbb{E}[\text{time} \mid x_i \text{ is searched}] \]

\[ = \sum_{i=1}^{m} \frac{1}{n} \cdot \left( 1 + \sum_{j=i+1}^{m} \mathbb{E}[X_{ij}] \right) \]

where \( X_{ij} = 1 \), if \( h(x_i \cdot \text{key}) = h(x_j \cdot \text{key}) \).

\[ = \sum_{i=1}^{m} \frac{1}{n} \left( 1 + \sum_{j=i+1}^{m} p(X_{ij} = 1) \right) \]

\[ = \sum_{i=1}^{m} \frac{1}{n} \left( 1 + \sum_{j=i+1}^{m} \frac{1}{m} \right) \]

\[ = \sum_{i=1}^{n} \left( \frac{1}{n} + \frac{1}{n} \cdot \frac{n-i}{m} \right) \]
\[
\begin{align*}
&= 1 + \frac{1}{n} \sum_{i=1}^{n-i} \frac{n-i}{m} \\
&= 1 + \frac{1}{n} \left( \frac{n^2}{m} - \frac{n(n+1)}{2m} \right) \\
&= 1 + \frac{n}{m} - \frac{n+1}{2m} \\
&\quad \left( + \frac{n}{2m} - \frac{n}{2m} \right) - \frac{1}{2m} \\
&= 1 + \frac{\chi}{2} - \frac{\chi}{2n} \\
&= O(1 + \alpha).
\end{align*}
\]
Suppose $n = O(m)$ then

$\mathbb{E}[\text{search time}] = O(1)$.

**Hashing.**

Suppose $k \in (0, 1)$ uniform distribution then the hash fn.

$$h(k) = \lfloor km \rfloor$$ fulfills the SUHA.

**Division Method.**

Suppose $n$.

$$h(k) = k \mod m.$$ 

$m$ should not be $2^p$ (power of 2).

$$968 \mod 100.$$
Generally \( m \) is chosen to be a prime that is not close to a power of 2.

**Multiplication method.**

Constant \( A \), \( 0 < A < 1 \).

\[
h(k) = \begin{cases} \frac{m \cdot (k \cdot A \mod 1)}{k \cdot A - \lfloor kA \rfloor} \\
\end{cases}
\]

Dependence on \( m \) is less.

\[
A = \frac{\sqrt{5} - 1}{2}
\]
Open Addressing.

All objects are stored in the table $T$.

If $h(k)$ is full, we need to systematically determine where to insert/search the item.

In open addressing $\alpha \leq 1$. 
Bestimmen: \( h : U \rightarrow \{0, 1, \ldots, m-1\} \)

\[ k \]

Now: \( h : U \times \{0, 1, \ldots, m-1\} \rightarrow \{0, 1, \ldots, m-1\} \)

\[ h(k, 0) = h(k, 0) + 1 \]
\[ h(k, i) = h(k, i) + 2 \]

\( \langle h(k, 0), h(k, 1), h(k, 2), \ldots, h(k, m-1) \rangle \)

We will assume that this seq. of probes is a permutation of \( \{0, 1, \ldots, m-1\} \) (we want all slots in the table to be searched)

Insert \((T, k)\)

\[ i \leftarrow 0 \quad // \text{probe } m. \]

\[
\text{repeat}
\]
\[ j \leftarrow h(k, i) \]

\[ \text{assume } \text{obj} = \text{key}. \]
if $T[j] = \text{NIL}$ then

$T[j] \leftarrow k$

return $j$

$i \leftarrow i + 1$

until $i = m$

print ("Table is full").

---

Search ($T$, $k$)

$i \leftarrow 0$ // probe $m$.

repeat

$j \leftarrow h(k, i)$

if $T[j] = k$ then

$T[j] \leftarrow k$

return $j$

$i \leftarrow i + 1$
\[
\text{until } i = m \text{ or } T[j] = \text{NIL}
\]
\[
\text{print("Not in the feblu").}
\]

**Uniform Hashing Assumption** : Each of the \(m!\) permutation of the probe seq. is equally likely.

**Linear Probing**

\[
h(k, i) = (h'(k) + i) \mod m.
\]

\[
h(k, 0) = h'(k)
\]

\[
h(k, 1) = (h'(k) + 1) \mod m
\]

Linear Probing creates big gap & big gap clusters if filled slots.
Pr of empty slot bus filled by the next elem = \frac{(i+1)}{m}.

**Quadratic Probing**

\[ h(k, i) = (h_1(k) + c_1i + c_2i^2) \mod m. \]

# probe sep. considered by linear probing:

\[ \langle 18, 26, 27, 19, 1, 2, \ldots \rangle \]

18, 19, \ldots

**Double Hashing.**
\[ h(k, i) = \left( h_1(k) + i \cdot h_2(k) \right) \mod m. \]

Start: \( h_1(k) \)  offset: \( h_2(k) \)

# probe seq: \( m^2 \).