Open Addressing

Use first empty slot defined by probe sequence \( \langle S_0, \ldots, S_{m-1} \rangle \)

Let \( x = \text{element} . \text{hashCode}() \)

\[ S_0 = \text{hash}(x) \]

for \( i = 1, \ldots, m-1 \)

\[ S_i = (S_{i-1} + \text{stepHash}(x)) \mod m \]
## Slot

<table>
<thead>
<tr>
<th>0</th>
<th>Probe 1</th>
<th>S₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Probe 6</td>
<td>S₆</td>
</tr>
<tr>
<td>2</td>
<td>Probe 3</td>
<td>S₃</td>
</tr>
<tr>
<td>3</td>
<td>Probe 0</td>
<td>S₀</td>
</tr>
<tr>
<td>4</td>
<td>Probe 5</td>
<td>S₅</td>
</tr>
<tr>
<td>5</td>
<td>Probe 2</td>
<td>S₂</td>
</tr>
<tr>
<td>6</td>
<td>Probe 7</td>
<td>S₇</td>
</tr>
<tr>
<td>7</td>
<td>Probe 4</td>
<td>S₄</td>
</tr>
</tbody>
</table>

- \( m = 8 \) \( \uparrow \) hash table size
- \( \text{hash}(x) = 3 \)
- Step: \( \text{Hash}(x) = 5 \)
Locating an element - Go through hash table in order of the probe sequence until you either reach the desired element or an empty (unused) slot is reached.

Adding an element

Go through hash table in order given by probe sequence until the element is found or an empty slot $S$ is reached. If the element was not there and we will place it at slot $S$, $\text{table}[S] = e$.
Let's look at an example.

<table>
<thead>
<tr>
<th>element $e$</th>
<th>A</th>
<th>B</th>
<th>E</th>
<th>F</th>
<th>H</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{hash}(e.\text{hashCode}())$</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$\text{stepHash}(e.\text{hashCode}())$</td>
<td>7</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

Insert in order \{A, B, E, F, H, J\}

Table $m=8$

\[\text{B F H E A J}\]

EMPTY
Care must be selected in how \( m \) and \( \text{stepHash}(x) \) relate. Why?

\[
m = 8 \quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7
\]

\[
\text{hash} = i \\
\text{stepHash} = 4
\]

Avoid this bad behavior and guarantee probe sequence is a permutation of \( \langle 0, \ldots, m-1 \rangle \) by making \( m + \text{stepHash} \) relatively prime. Pick \( m \) to be a power of 2, make \( \text{stepHash} \) odd.
How can you remove an element?

Let's delete A.

Now search for F.

<table>
<thead>
<tr>
<th>element e</th>
<th>A</th>
<th>B</th>
<th>E</th>
<th>F</th>
<th>H</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>hash(e.hashCode())</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>stepHash(e.hashCode())</td>
<td>7</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

Replace reference to A, by a ref. to DELETED sentinel.

Search must continue until EMPTY is reached (or element is found).
Deleting an element

Problem: hash table can fill up with slots "marked as deleted" (ref to DELETED)

Partly address this problem by re-using a deleted slot along probe sequence when inserting a new element.

In a mternal locate method (to check if element is in set), remember the 1st DELETED slot found & return that (or first empty slot if there were no deleted slots).
Actual load versus target load

Load factor \( \alpha = \frac{n+d}{m} \)

- \# elements in Set
- \# slots marked as deleted
- \# slots deleted
- \# table size

During an unsuccessful search, \( \alpha \) is fraction of slots that will cause search to continue.

desired load factor \( \alpha^* \) (e.g., \( 1/2 \))

actual load is current value of \( \frac{n+d}{m} \)
Goal: keep $\lambda$ close to $\lambda^*$

Limit frequency of resizing $\leftarrow$ expensive

Double table size ($m$) when $\lambda$ reaches $\left\lfloor \frac{1 + \lambda^*}{2} \right\rfloor$ halfway between $\lambda^* + 1$

$\lambda^* = \frac{1}{2}$, resize when $\lambda = \frac{3}{4}$

Hash functions change & you must rebuild by re-inserting all elements (in order)

- go through slots, move to next if empty
- or deleted, re-insert elements in new table
The hash table could be oversized (and cluttered with deleted slots)

half table size when \( \frac{n}{m} \) drops down to \( \frac{2}{2} \)

reduce hash table size by a factor of 2.
Analysis

\[ E[\text{# probes in an unsuccessful search}] = 1 \cdot \text{prob. probe 0 occurs} + 1 \cdot \text{prob. probe occurs} \]

\[ \leq 1 + x + x^2 + x^3 + \ldots \]

\[ \Rightarrow \sum_{i=0}^{\infty} x^i = \frac{1}{1-x} \]

\[ \frac{n}{m} \cdot \frac{(n-1)}{(m-1)} \leq x^2 \]

\[ E[\text{# probes in a successful search}] = \frac{1}{2} \ln \frac{1}{1-x} \]
Tradeoff between space + search time

as we increase $\lambda$

use less space but higher expected cost per search

this is a pretty large value

\[
\frac{1}{1 - \frac{7}{8}} = 8
\]

As we decrease $\lambda$, use more space but have lower expected cost per search

\[
\lambda = \frac{1}{4}, \quad \frac{1}{1 - \frac{1}{4}} = \frac{4}{3}
\]