Resolving Collisions w/ Open Addressing assuming \( n \leq m \)

The point is to avoid auxiliary linked lists. Use that space for a table.

Instead, create a probe sequence as a function of key value. (and pointers)

\[ \text{permuation of slots to try.} \]

\[
\begin{array}{l}
36 \\
43 \\
78 \\
5 \\
103 \\
2014 \\
\end{array}
\]

\[
\text{ex: } h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6.
\]

\[
\begin{array}{l}
\text{Insert}(64): \\
\quad \text{Try } T[9]: \text{ full} \\
\text{ Try } T[2]: \text{ full} \\
\text{ Try } T[4]: \text{ full} \\
\text{ Try } T[8]: \text{ ok}
\end{array}
\]

\[
\begin{array}{l}
\text{Search}(64): \text{ follows same sequence.} \\
\text{ Would return "not found" after 4 attempts.}
\end{array}
\]
ex: $h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6$.

Really this is $h(k, i)$

$h(64, 1) = 9$ \quad h(64, 2) = 2 \quad h(64, 3) = 4$
ex: \( h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6. \)

Really this is \( h(k, i) \)

\( h(64, 1) = 9 \)
\( h(64, 2) = 2 \)
\( h(64, 3) = 4 \)

\( \text{Delete}(64) : \)

\( h(64, 1) = 9 \), occupied by 2014
\( h(64, 2) = 2 \), occupied by 43
\( h(64, 3) = 4 \), occupied by 78
\( h(64, 4) = 8 \), found 64, DELETE IT.

OK?
ex: \( h(64) \rightarrow 9, 2, 4, 8, 1, 3, 11, 7, 10, 5, 6. \)

Really this is \( h(k, i) \)  

\[
\begin{align*}
h(64, 1) &= 9 & h(64, 2) &= 2 & h(64, 3) &= 4 \quad \text{etc.}
\end{align*}
\]

Delete(64):  
\[
\begin{align*}
h(64, 1) &= 9, \text{ occupied by } 2014 \\
h(64, 2) &= 2, \text{ occupied by } 43 \\
h(64, 3) &= 4, \text{ occupied by } 78 \\
h(64, 4) &= 8, \text{ found 64, DELETE IT.}
\end{align*}
\]

What if \( h(103) \rightarrow 4, 8, 2, 7, 1, 10, 11, 3, 5, 9, 6 \)?

Search(103):  
\[
\begin{align*}
h(103, 1) &= 4, \text{ occupied by } 78 \\
h(103, 2) &= 8, \text{ empty: declare 103 not in T.}
\end{align*}
\]

Could use special "deleted" markers, but search time increases

(Consider deleting all but one element, and then searching for it)
Typical probing sequences

Linear probing: \( h(k,i) = (h(k,0) + i) \mod m \), \( \sim h(k) \) and wrap around.

...tends to generate clusters.

Diagram:

Probability of extending a cluster:

\[ \frac{|\text{cluster}|}{m} \]

slows down search
Typical probing sequences

Linear probing: $h(k, i) = (h(k, 0) + i) \mod m \sim h(k)$ and wrap around.

... tends to generate clusters.

Quadratic probing: $h(k, i) = (h(k, 0) + c \cdot i + d \cdot i^2) \mod m$

Less clustering, need to make sure sequence hits all slots

Both generate $m$ probe sequences in total

Double hashing: $h(k, i) = (h_1(k) + i \cdot h_2(k)) \mod m$

Each $k$ has "random" offset

Generates $O(m^2)$ probe sequences: better

Heuristic: choose $m = 2^r$ & $h_2(k)$: odd.
Analysis of open addressing

Assuming uniform hashing:

- Each key is equally likely to have any of the $m!$ permutations as probe sequence (independent of other keys)

Even though all we have so far is $O(m^2)$

Simple uniform hashing:

For a random $h$, every slot is equally likely
Analysis of Open Addressing

Assuming Uniform Hashing: each key is equally likely to have any of the \( m! \) permutations as probe sequence (independent of other keys)

Recall \( n < m \), so \( \alpha < 1 \).

Claim: \( E[\#\text{probes}] \leq \frac{1}{1-\alpha} \left( \frac{m}{m-n} \right) \)

(search)

If true, then for \( n \ll m \) \( E[\#\text{probes}] = O(1) \)

\( \Rightarrow n = \frac{1}{2} m \rightarrow 2 \) probes

\( \Rightarrow 90\% \text{ full table} \rightarrow 10 \) probes

Works well if you can afford a table \( \sim \) data \( \times 2 \)

but keep in mind: we’re using a very strong assumption
Claim: $E[\#\text{probes}] \leq \frac{1}{1-\alpha}$ Look at unsuccessful search

$P[1\text{st probe collides}] = \frac{n}{m} \quad \rightarrow \text{need 2nd probe}$

$P[2\text{nd probe collides}] = \frac{n-1}{m-1} \quad \rightarrow \text{need 3rd probe}$

$\vdots$

$\frac{n-i}{m-i} < \frac{n}{m} = \alpha$

$E[\#\text{probes}] = 1 + \frac{n}{m} \left( 1 + \frac{n-1}{m-1} \left( 1 + \frac{n-2}{m-2} \left( \cdots \left( 1 + \frac{0}{m-n} \right) \right) \right) \right)$

$\leq 1 + \alpha \left( 1 + \alpha \left( 1 + \alpha \left( \cdots \left( 1 + \alpha \right) \right) \right) \right) \quad \cdots n \text{ terms}$

$\leq 1 + \alpha + \alpha^2 + \alpha^3 + \cdots \quad \cdots \infty \text{ terms}$

$= \sum_{i=0}^{\infty} \alpha^i = \frac{1}{1-\alpha}$

see CLRS for alternate analysis incl. successful search

Remember, probe sequence is a permutation.

Never check one slot twice.
Suggested reading:

- perfect hashing
  Family of hash functions, pick one randomly. Beats adversaries.

- universal hashing
  Fixed input. Create $h()$ based on this.

- cuckoo hashing!
  Cuckoos Use Mafia Tactics, And They Work
  April 18, 2014 | by Stephen Luntz