Separate chaining requires linked lists, pointers, mod

Alternative: when collision occurs, try other indices

(2) Linear probing (Open Addressing)

If hash(key) full, try:

\[(\text{hash}(\text{key}) + i) \mod n \quad \text{for } i = 0, 1, 2, 3, \ldots \]

\[\text{add offset} \quad \text{always mod n to get value in } [0, n-1]\]

In general, try \((\text{hash}(\text{key}) + i) \mod n\) for \(i = 0, 1, 2, 3, \ldots\)

\[\text{Note: linear function}\]

\[n = 10\]

\[\text{insert}(49) \Rightarrow 9 \quad (\text{hash}(\text{key}) + 0) \mod n\]

\[= (\text{key mod 10} + 0) \mod 10\]

\[= 9 \mod 10\]

\[\text{insert}(38) \Rightarrow 8\]

\[\text{insert}(19) \Rightarrow 9\]: full so try:

\[(\text{hash}(\text{key}) + 1) \mod 10\]

\[= (9 + 1) \mod 10 = 0\]

\[\text{insert}(18) \Rightarrow 8\] (full)

\[(8 + 1) \mod 10 = 9\] (full)

\[(8 + 2) \mod 10 = 0\] (full)

\[(8 + 3) \mod 10 = 1\]

\[\text{insert}(29) \Rightarrow 9 x, 0 x, 1 x, 2 x\]
Go to RunTime

Problem: Makes clusters (long blocks of filled slots).
If $\frac{m}{n} > 0.5$ can be shown to require 2.5 probes on average.

Primary clustering: long blocks of filled slots.

Instead: Quadratic probing:
If hash(key) full, try:

$(\text{hash(key)} + 1) \mod n$ \hspace{1cm} $(\text{hash(key)} + 2) \mod n$

In general:
Try $h_i(key) = (\text{hash(key)} + i^2) \mod n$ for $i = 0, 1, 2$.

\[\begin{array}{c|c}
0 & 19 \\
1 & \\
2 & 29 \\
3 & \\
4 & \\
5 & \\
6 & \\
7 & \\
8 & 49 \\
9 & \\
\end{array}\]

Now: \(\text{insert}(19)\Rightarrow 9 \text{ full}\)

try: \(((19 \mod 10) + 1) \mod 10 = 10 \mod 10 = 0\)

\(\text{insert}(29)\Rightarrow 9 \text{ full \hspace{1cm} (same as 19)}\)

try: \(((29 \mod 10) + 1) \mod 10 = 10 \mod 10 = 0 \text{ full}\)

\((29 \mod 10) + 4 \mod 10 = 13 \mod 10 = 3\)

Time?

Worst-case:
- Find: $O(m)$ (NOT $O(n)$) (probe every element)
- Insert: duplicates OK (stored separately): $O(m)$ (find empty slot)

""" together: $O(m)$ requires find()"""
Remove? How to remove? Can we just remove the element? No.

Example:

<table>
<thead>
<tr>
<th>0</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td></td>
</tr>
</tbody>
</table>

Suppose remove(49)

Now find(19): 19 mod 0 = 9, empty, return null

Instead for remove: mark as 'deleted'. Slot available for future insert but value is removed.

| 9 | 49 | deleted = true |

Avg-Case: Find/Insert/Remove: O(1)

Elements will be fairly spread out in table, so constant # of probes.

Problem with both Linear & Quadratic Probing?

Secondary clustering - keys mapped to the same index (ex. 49, 19, 29) follow same probe sequence (9, 0, 3, ... )
Avoid with **Doubling Hashing** - apply another hash function:

- Original function: $h_1(key) = key \mod n$
- New function: $h_2(key) = n' - key \mod n'$

$n'$: prime # smaller than $n$.

*(will discuss later)*

**Try:**

$h_i(key) = (h_{i-1}(key) + i \cdot h_2(key)) \mod n$ for $i = 0, 1, 2, 3$.

**Why a prime # for $n'$?**

In previous examples used $n = 10$ for easy modulus, but using a prime number greatly reduces chances of collision.

**Example:** $n = 11$, $n' = 7$, $key = 19$

**Probe sequence for 19:**

$h_0 = h_{i-1}(key) \mod n = (19 \mod 11) \mod 11 = 8$

$h_1 = (h_{i-1}(key) + h_2(key)) \mod 11$

$\quad = (8 + (7 - 19 \mod 7)) \mod 11$

$\quad = (8 + 2) \mod 11 = 10$

$h_2 = (8 + 2(2)) \mod 11 = 1$

$h_3 = (8 + 3(2)) \mod 11 = 3$

**Probe sequence for 19:** 8, 10, 1, 3
Now, suppose key = 30

\[ h_0(30) = 30 \mod 11 = 8 \] (same as 19)

With linear + quadratic probing 30 would follow the same probe sequence as 19 since they both initially hashed to 8.

\[ h_1(30) = (8 + (7 - 30 \mod 7)) \mod 11 = 12 \]
\[ h_2(30) = (8 + 2(5)) \mod 11 = 7 \]
\[ h_3(30) = (8 + 3(5)) \mod 11 = 1 \]

Probe sequence for 30: 8, 2, 7, 1
Problem with open addressing?
Table can get full

Re-hashing: elements reinсerted into larger table.

Recall: \( \text{load} = \lambda = \frac{m}{n} \) \( \pm \) elements

Re-hashing:
(1) hash table has threshold 8
(2) rehash occurs when \( \lambda > 8 \)
(a) create new table of size first prime larger than 2n
(b) scan original table and rehash elements into new table

### Example

<table>
<thead>
<tr>
<th>( n )</th>
<th>( \frac{m}{n} )</th>
<th>( \frac{m}{7} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>15</td>
<td>m &gt; 3.5 ( \Rightarrow_m = 4 )</td>
<td>m &gt; 3.5 ( \Rightarrow_m = 4 )</td>
</tr>
</tbody>
</table>

\( N = 17 \)