Separate chaining requires linked lists, pointers, nodes.

Alternative: when collision occurs, try other indices.

(2) Linear Probing (Open Addressing)

If hash(key) full, try:

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

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

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

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

\[\text{ex. } n = 10 \]

\[
\begin{array}{c|c|c}
0 & 19 & 9 \\
1 & 18 & 9 \\
2 & 29 & 9 \\
3 & & 9 \\
4 & & 9 \\
5 & & 9 \\
6 & & 9 \\
7 & 38 & 8 \\
8 & 49 & 8 \\
9 & & 8 \\
\end{array}
\]

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

\[= (\text{key mod 10} + 0) \mod 10 = 9 \mod 10 \]

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

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

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

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

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

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

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

\[\text{insert (29) at } 9, 0, 1, 2 \checkmark \]
Problem? Makes clusters (long blocks of filled slots).
If \( \lambda = m/n > 0.5 \) 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:

\[ (\text{hash(key)} + 1) \mod n, (\text{hash(key)} + 4) \mod n \]

In general:

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

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

```
Now: insert(19) \Rightarrow 9 full

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

\cdot insert(29) \Rightarrow 9 full < same as 19>

\cdot try: ((29 \mod 10) + 1) \mod 10 = 10 \mod 10 \geq 0 full

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

\n
\n
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 \( \text{find()} \).
- Remove? How to remove? Can we just remove the element? No.

ex: 

| 19 |
| 29 |
| 9  |

Suppose remove(49)

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

Instead for remove: mark as 'deleted'.

49 + deleted = true

Time: O(m) (Still requires find().)

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, ...)

(This material + double-hashing is not required for exams.)
With Doubling Hashing, apply another hash function.

**Original function:** \( \text{hash}_1(\text{key}) = \text{key} \mod n \)

**New:** \( \text{hash}_2(\text{key}) = n' - \text{key} \mod n' \)

(\( n' \): prime # smaller than \( n \))

**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:** \( n=11, \ n' = 7, \ \text{key}=19 \)

**Probe sequence for 19?**

\( h_0 = (\text{hash}_1(\text{key})) \mod n = (19 \mod 11) \mod 11 = 8 \)

\( h_1 = (\text{hash}_1(\text{key}) + \text{hash}_2(\text{key})) \mod 11 \)

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

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

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

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

Probe sequence for 19: 8, 10, 1, 3
Now, suppose \( key = 30 \)

\[
h_0(30) = 30 \mod 11 = \boxed{8} \quad \text{(same as 19)}
\]

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

\[
h_1(30) = (8 + (7 - 30 \mod 7)) \mod 11 = \boxed{2}
\]
\[
h_2(30) = (8 + 2(5)) \mod 11 = \boxed{7}
\]
\[
h_3(30) = (8 + 3(5)) \mod 11 = \boxed{11}
\]

Probe sequence for 30: 8, 2, 7, 1...
What should be the table size?

Previous examples had $n=10$ for easy modulus computation.

Better to have prime table size.

Prime table size reduces # of collisions in practice.
Problem with open addressing?
Table can get full

Re-hashing - elements are reinserted into a larger table

Recall: load = \( \lambda = \frac{m}{n} = \frac{\text{# elements}}{\text{table size}} \)

How to re-hash?
1. hash table has threshold \( s \)
2. rehash occurs when \( \lambda > s \)
   a. create new table of size first prime larger than \( 2n \)
   b. scan original table and rehash elements into new table

<table>
<thead>
<tr>
<th>( \lambda )</th>
<th>( n=7 )</th>
<th>( \beta = .5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>23</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td>6</td>
</tr>
</tbody>
</table>

Rehash when:
\[ \lambda > .5 \]
\[ m > \frac{\beta}{n} \]
\[ m > \frac{\beta}{7} \]
\[ m > 3.5 \Rightarrow m = 4 \]

\[ \frac{m}{n} > \frac{\beta}{7} \]

\[ \frac{m}{7} > \frac{\beta}{7} \]

\[ n = 17 \]