Every hash table contains three pieces:
1. A **hash function**, \( f(k) \). The hash function transforms a key from the keyspace into a small integer.
2. An **array**.
3. A mystery third element.

Collision Handling Strategy #2: Probe-based Hashing

Example: \( S = \{ 16, 8, 4, 13, 29, 11, 22 \} \), \( |S| = n \)

\[
h_1(k) = k \mod 7, \quad \text{|Array|} = N
\]

**Linear Probing:**
- Try \( h(k) = (k + 0) \mod 7 \), if full...
- Try \( h(k) = (k + 1) \mod 7 \), if full...
- Try \( h(k) = (k + 2) \mod 7 \), if full...

**Double Hashing:**
- Try \( h(k) = (k + 0 \cdot h_2(k)) \mod 7 \), if full...
- Try \( h(k) = (k + 1 \cdot h_2(k)) \mod 7 \), if full...
- Try \( h(k) = (k + 2 \cdot h_2(k)) \mod 7 \), if full...

...  

\[
h(k, i) = (h_1(k) + i \cdot h_2(k)) \mod 7
\]

**Running Time:**

**Linear Probing:**
- Successful: \( \frac{1}{2} \left( 1 + \frac{1}{(1-\alpha)} \right) \)
- Unsuccessful: \( \frac{1}{2} \left( 1 + \frac{1}{(1-\alpha)} \right)^2 \)

**Double Hashing:**
- Successful: \( \frac{1}{\alpha} \cdot \ln \left( \frac{1}{(1-\alpha)} \right) \)
- Unsuccessful: \( \frac{1}{(1-\alpha)} \)

**Separate Chaining:**
- Successful: \( 1 + \frac{\alpha}{2} \)
- Unsuccessful: \( 1 + \alpha \)

**Running Time Observations:**
1. As \( \alpha \) increases:
2. If \( \alpha \) is held constant:
Running Time Observations:

Linear Probing:
Successful: $\frac{1}{2}(1 + \frac{1}{1-\alpha})$
Unsuccessful: $\frac{1}{2}(1 + \frac{1}{1-\alpha})^2$

Double Hashing:
Successful: $\frac{1}{\alpha} \cdot \ln\left(\frac{1}{1-\alpha}\right)$
Unsuccessful: $\frac{1}{(1-\alpha)}$

ReHashing:
What happens when the array fills?

Better question:

Algorithm:

Which collision resolution strategy is better?

- Big Records:

- Structure Speed:

What structure do hash tables replace?

What constraint exists on hashing that doesn’t exist with BSTs?

Why talk about BSTs at all?

Analysis of Dictionary-based Data Structures

<table>
<thead>
<tr>
<th></th>
<th>Hash Table</th>
<th>AVL</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Amortized</td>
<td>Worst Case</td>
<td></td>
</tr>
<tr>
<td>Find</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage Space</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A Secret, Mystery Data Structure:

ADT:

- insert
- remove
- isEmpty

CS 225 – Things To Be Doing:

1. Theory Exam 3 starts next week (Tuesday, April 3rd)
2. MP5 has been released; EC+7 deadline is tonight
3. lab_btree due this Tuesday, March 27th at 11:59pm
4. Daily POTDs are ongoing!