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: \( \frac{1}{\alpha} \ln(\frac{1}{1 - \alpha}) \)
- Unsuccessful: \( \frac{1}{1 - \alpha} \)

**ReHashing:**
What happens when the array fills?

Better question:

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?

A Secret, Mystery Data Structure:

Algorithm:

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

ADT:
- insert
- remove
- isEmpty
Priority Queue Implementations

<table>
<thead>
<tr>
<th></th>
<th>insert</th>
<th>removeMin</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>O(n)</td>
<td>O(n)</td>
<td>O(n)</td>
<td>Unsorted Array</td>
</tr>
<tr>
<td>O(1)</td>
<td>O(n)</td>
<td>O(n)</td>
<td>Unsorted List</td>
</tr>
<tr>
<td>O(lg(n))</td>
<td>O(1)</td>
<td>O(lg(n))</td>
<td>Sorted Array</td>
</tr>
<tr>
<td>O(lg(n))</td>
<td>O(1)</td>
<td>O(lg(n))</td>
<td>Sorted List</td>
</tr>
</tbody>
</table>

...what errors exist in this table?

Which algorithm would we use?

A New Tree-like Structure:

CS 225 – Things To Be Doing:

1. Register for CS 225's Final Exam!
2. Exam #8 (programming, MP4-like and AVL) ongoing
3. MP5 due Monday, Nov. 6
4. lab_heaps due Sunday, Nov. 5
5. Daily POTDs