Implementation of a Double Hash Table

December 9\textsuperscript{th}, 2003

Matt Nyerges
Todd Schneider
Why a Double Hash?

- Using a single hash often results in many collisions, due to the birthday paradox.
- Normally when a collision occurs, the new item is either chained onto the earlier item, or the item is placed into the next available bucket with open addressing.
- No matter how a collision is dealt with, the result is the same- increased time for insertion and searching!

The Solution: A Double Hash reduces collisions to nearly 0, dramatically improving performance.
Implementation of the Double Hash

- Very straightforward. The hash table is given two hash functions during construction. When an item is inserted, it is placed in the bucket in accordance with the first hash function and inside the bucket we use open addressing to handle collisions with the second hash function.

- Each bucket is resized and rehashed when it becomes > 2/3 full.
  - Especially useful when open addressing is used to ensure item is never far from its original specified location.
  - Improves efficiency by reducing out of place items resulting from collisions.
  - The only drawback is that it degrades the speed of insertion in the case of having to rehash.
To facilitate performance analysis, the Double Hash was compared to the hash implemented in Ford & Topp.

“knuth_words.txt” was loaded into each hash table.

War and Peace was then spell checked and output to file, and times were recorded.

The hash table sizes were varied to give most accurate performance numbers.
Results obtained on a P4 2.6GHz machine with 512 MB of DDR 2700 RAM.

Results Given in Seconds.

<table>
<thead>
<tr>
<th>Buckets</th>
<th>101</th>
<th></th>
<th>155</th>
<th></th>
<th>201</th>
<th></th>
<th>255</th>
<th></th>
<th>301</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Insert</td>
<td>Find</td>
<td>Insert</td>
<td>find</td>
<td>insert</td>
<td>find</td>
<td>insert</td>
<td>find</td>
<td>insert</td>
<td>find</td>
</tr>
<tr>
<td>Double Hash</td>
<td>2.16</td>
<td>1.41</td>
<td>1.28</td>
<td>1.21</td>
<td>1.52</td>
<td>1.06</td>
<td>.75</td>
<td>1.24</td>
<td>.77</td>
<td>1.09</td>
</tr>
<tr>
<td>FT Hash</td>
<td>13.20</td>
<td>107.3</td>
<td>9.19</td>
<td>69.8</td>
<td>7.21</td>
<td>53.45</td>
<td>5.49</td>
<td>42.45</td>
<td>4.66</td>
<td>36.06</td>
</tr>
</tbody>
</table>
The spell checker is implemented using the FOX Toolkit.

Users can load, save, edit, etc…

When users select Spell Check, the content of the text area is checked for errors. Items spelled incorrectly are returned with the tag <BAD> in front of them. e.g. <BAD>humphapoooh
Questions?

- You know you have some...