#26: Hashing: Collision Handling
October 26, 2018 · Wade Fagen-Ulmschneider

Every hash table contains three pieces:
1. A hash function, $f(k)$: keyspace $\rightarrow$ integer
2. An array.
3. A collision handling strategy.

Collision Handling Strategy #1: Separate Chaining
Example: $S = \{16, 8, 4, 13, 29, 11, 22\}$, $|S| = n$
\(h(k) = k \mod 7\), $|Array| = N$

Collision Handling Strategy #2: Probe-based Hashing
Example: $S = \{16, 8, 4, 13, 29, 11, 22\}$, $|S| = n$
\(h(k) = k \mod 7\), $|Array| = N$

Load Factor:

Running time of Separate Chaining:
<table>
<thead>
<tr>
<th>Worst Case</th>
<th>SUHA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert</td>
<td></td>
</tr>
<tr>
<td>Remove/Find</td>
<td></td>
</tr>
</tbody>
</table>

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...
...

What problem occurs?

Double Hashing:
Example: $S = \{16, 8, 4, 13, 29, 11, 22\}$, $|S| = n$
\(h_1(k) = k \mod 7, h_2(k) = 5 - (k \mod 5), |Array| = N\)

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: \( \frac{1}{2}(1 + \frac{1}{1 - \alpha}) \)
- Unsuccessful: \( \frac{1}{2}(1 + \frac{1}{1 - \alpha})^2 \)

Double Hashing:
- Successful: \( \frac{1}{\alpha} \times \ln(\frac{1}{1 - \alpha}) \)
- 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:

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. Programming Exam B is on-going
2. MP5 has been released; EC+7 deadline is Monday night
3. lab_btree due Sunday
4. Daily POTDs are ongoing!