Chapter 6, Hash tables:

- General introduction; separate chaining (Today)
- Open addressing (next week Monday)
- Hash functions (next week Wednesday)
- Perfect hashing (Monday after next)
- Hash table performance (Wednesday after next)

Today:

- A few test 2 comments
- The story of the Map ADT
- Goals and terminology of the unit
- Separate chaining implementation
- Variables and metrics of performance
Find | Search the data structure for a given key

Insert | Add a new key to the data structure

Delete | Get rid of a key and fix up the data structure

containsKey() | Find

get() | Find

put() | Find + insert

remove() | Find + delete
<table>
<thead>
<tr>
<th></th>
<th>Find</th>
<th>Insert</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unsorted array</strong></td>
<td>$\Theta(n)$</td>
<td>$\Theta(1)$ [Θ($n$)]</td>
<td>$\Theta(n)$</td>
</tr>
<tr>
<td><strong>Sorted array</strong></td>
<td>$\Theta(lg\ n)$</td>
<td>$\Theta(n)$</td>
<td>$\Theta(n)$</td>
</tr>
<tr>
<td><strong>Linked list</strong></td>
<td>$\Theta(n)$</td>
<td>$\Theta(1)$</td>
<td>$\Theta(1)$</td>
</tr>
<tr>
<td><strong>Balanced BST</strong></td>
<td>$\Theta(lg\ n)$</td>
<td>$\Theta(1)$ [Θ(lg $n$)]</td>
<td>$\Theta(1)$ [Θ(lg $n$)]</td>
</tr>
<tr>
<td><strong>What we want</strong></td>
<td>$\Theta(1)$</td>
<td>$\Theta(1)$</td>
<td>$\Theta(1)$</td>
</tr>
</tbody>
</table>
$h_1(k)$

number in $[0, \infty)$

$0 \cdots \infty$

$h_1(k)$

mod $m$

number in $[0, m)$

$0 \cdots m$
Separate chaining: \( \frac{n}{m} < \alpha \) where \( \alpha > 1 \)
Open addressing: $\frac{n}{m} < \alpha$ where $\alpha < 1$
Unit agenda:

- Solution 1: Separate chaining (plus basic concepts and terminology). (Today)
- Solution 2: Open addressing. (Next week Monday)
- All about hash functions. (Next week Wednesday)
- Solution 3: Perfect hashing. (Monday after next)
- Looking carefully at performance. (Wednesday after next)
Hash table terminology:

- Hash table: A *data structure*, not an ADT.
- Bucket: A position in the (main) array, or, abstractly, an index in the range \([0, m)\).
- Hash function: A function from keys to buckets.
- Collision: When two keys are hashed to the same bucket.
- Chain: A sequence of keys that needs to be searched through to find a given key.
- Load factor \((\alpha)\): An upper bound on the ratio of keys to buckets.
Factors in best vs worst vs expected case:

- State of the table
- Length of the bucket
- Position of key in the bucket.

Parameters that can be adjusted for engineering a hash table:

- Load factor $\alpha$
- Rehash strategy
- Hash function
\[ \begin{align*}
& \text{rehash} \quad \rightarrow \quad T(n) = (n-1)c_0 + c_1 + c_2 n \\
& \quad = (c_0 + c_2)n + c_1 - c_0 \\
& \quad = \Theta(n)
\end{align*} \]
Hash functions should distribute the keys *uniformly* and *independently*.

Uniformity:

\[ P(h(k) = i) = \frac{1}{m} \]

Independence:

\[ P(h(k_1) = i) = P(h(k_1) = i \mid h(k_2) = j) \]
Coming up:

Do **Optimal BST** project (suggested by today, Friday, April 8)

Due **today, Fri, Apr 8** (end of day)
Take quiz (on Sections 7.(1 & 2), should have read before class)

Due **Tues, Apr 12**
Do practice problem, recreating separate chaining example
Read Section 7.3 Take quiz