Chapter 6, Hash tables:

- General introduction; separate chaining (Today)
- Open addressing (next week Monday)
- Hash table performance (Monday after Thanksgiving)

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>Data Structure</th>
<th>Find</th>
<th>Insert</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsorted array</td>
<td>$\Theta(n)$</td>
<td>$\Theta(1)$</td>
<td>$\Theta(n)$</td>
</tr>
<tr>
<td>Sorted array</td>
<td>$\Theta(lg\ n)$</td>
<td>$\Theta(n)$</td>
<td>$\Theta(n)$</td>
</tr>
<tr>
<td>Linked list</td>
<td>$\Theta(n)$</td>
<td>$\Theta(1)$</td>
<td>$\Theta(1)$</td>
</tr>
<tr>
<td>Balanced BST</td>
<td>$\Theta(lg\ n)$</td>
<td>$\Theta(1)$</td>
<td>$\Theta(1)$</td>
</tr>
<tr>
<td>What we want</td>
<td>$\Theta(1)$</td>
<td>$\Theta(1)$</td>
<td>$\Theta(1)$</td>
</tr>
</tbody>
</table>
key

\[ h_1(k) \]

number in \([0, \infty)\)

\[ 0 \ldots \infty \]

\[ \text{mod } m \]

number in \([0, m)\)

\[ 0 \ldots m \]
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
rehash $\longrightarrow$

\[
\begin{align*}
O(1) & \quad c_0 \\
O(1) & \quad c_0 \\
O(1) & \quad c_0 \\
\vdots & \quad \vdots \\
O(1) & \quad c_0 \\
O(n) & \quad c_1 + c_2 n \\
O(1) & \quad c_0 \\
\vdots & \quad \vdots \\
O(1) & \quad c_0 \\
\end{align*}
\]

\[
T(n) = (n - 1)c_0 + c_1 + c_2 n = (c_0 + c_2)n + c_1 - c_0 = \Theta(n)
\]
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 Monday, Nov 21)

Due Fri, Nov 18 (end of day)
Read Sections 7.(1 & 2)
Take quiz

Due Mon, Nov 21 (end of day)
Do Project 7.1 (as practice problem)

Due Mon, Nov 28 (end of day) (recommended to be done before break)
Read Section 7.3
Do Exercises 7.(4,5,7,8)
Take quiz