Chapter 6, Hash tables:

- General introduction; separate chaining (**Today**)
- (No class Monday—enjoy the eclipse)
- Open addressing (next week Wednesday)
- Hash functions (next week Friday)
- Perfect hashing (week-after Monday)
- Hash table performance (week-after Friday)

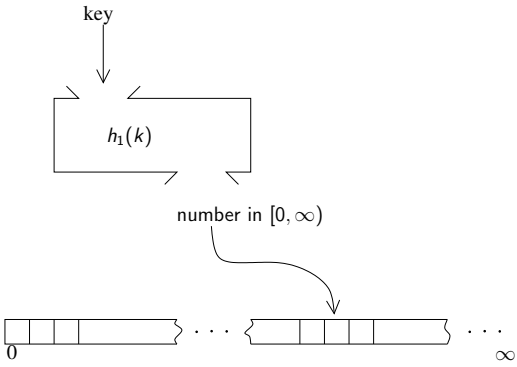
Today:

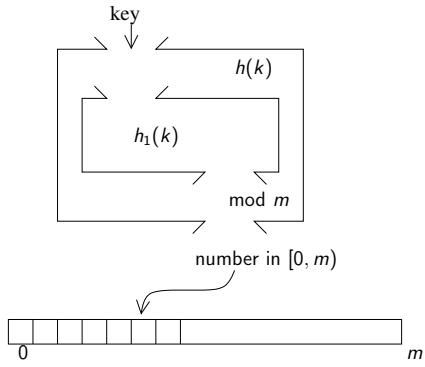
- The story of the Map ADT
- Goals and terminology of the unit
- Separate chaining implementation
- Variables and metrics of performance

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

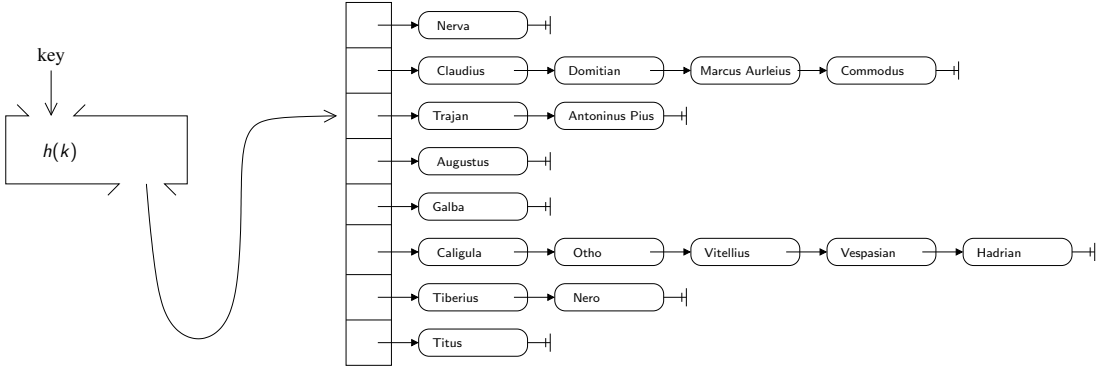
`containsKey()`	Find
`get()`	Find
`put()`	Find + insert
`remove()`	Find + delete

	Find	Insert	Delete
Unsorted array	$\Theta(n)$	$\Theta(1)$ $[\Theta(n)]$	$\Theta(n)$
Sorted array	$\Theta(\lg n)$	$\Theta(n)$	$\Theta(n)$
Linked list	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$
Balanced BST	$\Theta(\lg n)$	$\Theta(1)$ $[\Theta(\lg n)]$	$\Theta(1)$ $[\Theta(\lg n)]$
What we want	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$

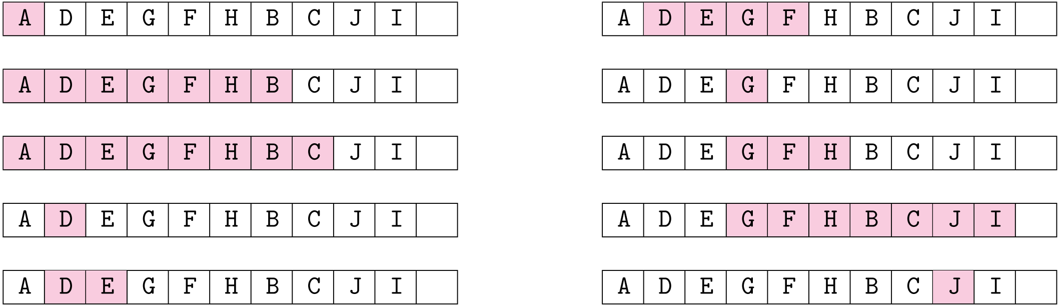

key
$h_1(k)$
number in $[0, \infty)$
0
∞


key
$h(k)$
$h_1(k)$
mod m
number in $[0, m)$
0
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 Wednesday)
- All about hash functions. (next week Friday)
- Solution 3: Perfect hashing. (week-after Monday)
- Looking carefully at performance. (week-after Wednesday)

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 (α): 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 α
- Rehash strategy
- Hash function

$$
\begin{array}{rll}
 & O(1) & c_0 \\
 & O(1) & c_0 \\
 & O(1) & c_0 \\
 & \vdots & \\
 & O(1) & c_0 \\
\text{rehash} \longrightarrow & O(n) & c_1 + c_2 n \\
 & O(1) & c_0 \\
 & \vdots & \\
 & O(1) & c_0
\end{array}
\Bigg\}
\begin{array}{rcl}
T(n) & = & (n-1)c_0 + c_1 + c_2 n \\
 & = & (c_0 + c_2)n + c_1 - c_0 \\
 & = & \Theta(n)
\end{array}
$$

Hash functions should distribute the keys *uniformly* and *independently*.

Uniformity:

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

Independence:

$$P(h(k_1) = i) \quad = \quad P(h(k_1) = i \mid h(k_2) = j)$$

Coming up:

Do **Optimal BST** *project (due Monday, April 8)*

Due **Mon, Apr 8** *(end of day)*
Read Sections 7.(1 & 2)
Take quiz (on Sections 7.(1 & 2), end of day)

Due **Thurs Apr 11**
Read Section 7.3
Do Exercises 7.(4,5,7,8)
Take quiz