Learning Objectives

Review fundamentals of hash tables

Introduce closed hashing approaches to hash collisions

Determine when and how to resize a hash table

Justify when to use different index approaches
A Hash Table based Dictionary

Client Code:

```c
1 Dictionary<KeyType, ValueType> d;
2 d[k] = v;
```

A **Hash Table** consists of three things:
1. A hash function
2. A data storage structure
3. A method of addressing *hash collisions*
Open vs Closed Hashing

Addressing hash collisions depends on your storage structure.

• **Open Hashing:** store \( k,v \) pairs externally

• **Closed Hashing:** store \( k,v \) pairs in the hash table
### Hash Table (Separate Chaining)

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>B+</td>
<td>2</td>
</tr>
<tr>
<td>Anna</td>
<td>A-</td>
<td>4</td>
</tr>
<tr>
<td>Alice</td>
<td>A+</td>
<td>4</td>
</tr>
<tr>
<td>Betty</td>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>Brett</td>
<td>A-</td>
<td>2</td>
</tr>
<tr>
<td>Greg</td>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>Sue</td>
<td>B</td>
<td>7</td>
</tr>
<tr>
<td>Ali</td>
<td>B+</td>
<td>4</td>
</tr>
<tr>
<td>Laura</td>
<td>A</td>
<td>7</td>
</tr>
<tr>
<td>Lily</td>
<td>B+</td>
<td>7</td>
</tr>
</tbody>
</table>

#### Diagram

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greg</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brett</td>
<td>A-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Betty</td>
<td></td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ali</td>
<td></td>
<td></td>
<td>B+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alice</td>
<td>A+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anna</td>
<td></td>
<td>A-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sue</td>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Laura</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lily</td>
<td></td>
<td></td>
<td>B+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Simple Uniform Hashing Assumption

Given table of size $m$, a simple uniform hash, $h$, implies

$$\forall k_1, k_2 \in U \text{ where } k_1 \neq k_2, \ Pr(h[k_1] = h[k_2]) = \frac{1}{m}$$

Uniform: keys are equally likely to hash to any position

Independent: key hash values are independent of other keys
Separate Chaining Under SUHA

Under SUHA, a hash table of size $m$ and $n$ elements:

Expected length of chain is ______________.

find runs in: ____________.

insert runs in: ____________.

remove runs in: ____________.
Collision Handling: Probe-based Hashing

\[ S = \{ 1, 8, 15 \} \]

\[ h(k) = k \% 7 \]

\[ |S| = n \]

\[ |\text{Array}| = m \]
Collision Handling: Linear Probing

\[ S = \{ 16, 8, 4, 13, 29, 11, 22 \} \]

\[ h(k) = k \mod 7 \]

\[ |S| = n \]

\[ |\text{Array}| = m \]

\[ h(k, i) = (k + i) \mod 7 \]

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...
Try ...
Collision Handling: Linear Probing

\[ S = \{ 16, 8, 4, 13, 29, 11, 22 \} \]

\[ |S| = n \]

\[ h(k, i) = (k + i) \mod 7 \]

\[ |Array| = m \]

_\text{find}(29)_
Collision Handling: Linear Probing

\( S = \{ 16, 8, 4, 13, 29, 11, 22 \} \)  \hspace{1cm} |S| = n

\( h(k, i) = (k + i) \mod 7 \)  \hspace{1cm} |Array| = m

_remove(16)
A Problem w/ Linear Probing

Primary clustering:

Description:

Remedy:
Collision Handling: Quadratic Probing

\[ S = \{ 16, 8, 4, 13, 29, 11, 22 \} \quad |S| = n \]
\[ h(k) = k \mod 7 \quad |\text{Array}| = m \]

\[ h(k, i) = (k + i^2) \mod 7 \]

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8</td>
<td>16</td>
<td>4</td>
<td>13</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Try \( h(k) = (k + 0) \mod 7 \), if full...
Try \( h(k) = (k + 1\times1) \mod 7 \), if full...
Try \( h(k) = (k + 2\times2) \mod 7 \), if full...
Try ...
A Problem w/ Quadratic Probing

Secondary clustering:

0
1
2
3
4
5
6
7
8
9

Description:

Remedy:
Collision Handling: Double Hashing

\[ S = \{ 16, 8, 4, 13, 29, 11, 22 \} \]
\[ |S| = n \]
\[ h_1(k) = k \mod 7 \]
\[ h_2(k) = 5 - (k \mod 5) \]

\[ h(k, i) = (h_1(k) + i \cdot h_2(k)) \mod 7 \]

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...
Try ...
Running Times
(Expectation under SUHA)

Open Hashing:

insert: ____________.

find/ remove: ____________.

Closed Hashing:

insert: ____________.

find/ remove: ____________.

(Don’t memorize these equations, no need.)
Running Times  *(Don’t memorize these equations, no need.)*

The expected number of probes for find(key) under SUHA

**Linear Probing:**
- Successful: \( \frac{1}{2} \left( 1 + \frac{1}{1-\alpha} \right) \)
- Unsuccessful: \( \frac{1}{2} \left( 1 + \frac{1}{1-\alpha} \right)^2 \)

**Double Hashing:**
- Successful: \( \frac{1}{\alpha} \times \ln \left( \frac{1}{1-\alpha} \right) \)
- Unsuccessful: \( \frac{1}{1-\alpha} \)

**Separate Chaining:**
- Successful: \( 1 + \frac{\alpha}{2} \)
- Unsuccessful: \( 1 + \alpha \)

Instead, observe:
- As \( \alpha \) increases:
- If \( \alpha \) is constant:
Running Times

The expected number of probes for find(key) under SUHA

**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: $\frac{1}{1-\alpha}$

When do we resize?
Resizing a hash table

How do we resize?
Which collision resolution strategy is better?
- Big Records:
- Structure Speed:

What structure do hash tables implement?

What constraint exists on hashing that doesn’t exist with BSTs?

Why talk about BSTs at all?
## Running Times

<table>
<thead>
<tr>
<th></th>
<th>Hash Table</th>
<th>AVL</th>
<th>Linked List</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Find</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Amortized:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Worst Case:</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Insert</strong></td>
<td>Amortized:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Worst Case:</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Storage Space</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
std data structures

std::map
std data structures

std::map
   ::operator[]
   ::insert
   ::erase

   ::lower_bound(key) ➔ Iterator to first element ≤ key
   ::upper_bound(key) ➔ Iterator to first element > key
std data structures

std::unordered_map
   ::operator[]
   ::insert
   ::erase

   ::lower_bound(key) ➔ Iterator to first element ≤ key
   ::upper_bound(key) ➔ Iterator to first element > key
std data structures

`std::unordered_map`
  `::operator[]`
  `::insert`
  `::erase`

  `::lower_bound(key)` ➔ Iterator to first element ≤ key
  `::upper_bound(key)` ➔ Iterator to first element > key

  `::load_factor()`
  `::max_load_factor(ml)` ➔ Sets the max load factor
Coming up next...
Bonus Slides
Hash Function (Division Method)

Hash of form: \( h(k) = k \mod m \)

Pro:

Con:
Hash Function (Multiplication Method)

Hash of form: $h(k) = \lfloor m(kA \% 1) \rfloor$, $0 \leq A \leq 1$

Pro:

Con:
Hash Function (Universal Hash Family)

Hash of form: $h_{ab}(k) = ((ak + b) \mod p) \mod m$, $a, b \in Z_p^*, Z_p$

$\forall k_1 \neq k_2$, $Pr_{a,b}(h_{ab}[k_1] = h_{ab}[k_2]) \leq \frac{1}{m}$

Pro:

Con: