CS 310: Hash Table Wrap-up

Chris Kauffman

Week 7-2
Logistics

HW 2
- Due tonight
- Questions for group discussion?

Goals Today
- Hash Tables Wrap-up
- Maps and Sets

Midterm Exam
- Next Monday Review
- Next Wednesday Exam
- Covers material through this week (Hash tables)
- Open resource
Removal in Open Addressing: Follow Chain

<table>
<thead>
<tr>
<th>Item</th>
<th>Code</th>
<th>Pos</th>
<th>Added</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>6</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>7</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>E</td>
<td>5</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>8</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>G</td>
<td>11</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>H</td>
<td>12</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>I</td>
<td>9</td>
<td>13</td>
<td>9</td>
</tr>
</tbody>
</table>

- Suppose `remove(X)` sets position to null
- What are the booleans assigned to?

```java
h.remove(A); boolean b1 = h.contains(C);
h.remove(D); boolean b2 = h.contains(F);
h.remove(E); boolean b3 = h.contains(I);
```
Avoid Breaking Chains in Removal

- Don’t set removed records to null
- Use place-holders, in Weiss it’s `HashSet.HashEntry`

```java
private static class HashEntry {
    public Object element;    // the element
    public boolean isActive;  // false if marked deleted
    public HashEntry( Object e ) {
        this( e, true );
    }
    public HashEntry( Object e, boolean i ){
        element = e;
        isActive = i;
    }
}
```

Explore `weiss/code/HashSet.java`

- `remove(x)` sets `isActive` to false
- `contains(x)` treats slot as filled
- `rehash()` ignores inactive entries
Quadratic Probing

Try the following sequence until an empty array element is found

pos, pos+1^2, pos+2^2, pos+3^2, \ldots pos+i^2

- Primary clustering fixed: not putting in adjacent cells
- add works up to load = 0.5
  - Weiss Theorem 20.4, pg 786
- Can be done efficiently (Weiss pg 787)
- Complexity Not fully understood
  - No known relation of load to average cells searched
  - Interesting open research problem
Probe Sequence Differences

> Math.abs("Marylee".hashCode()) % 11
5

Linear Probe

Quadratic Probe

> Math.abs("Barb".hashCode()) % 11
5 --> Where?
Rehashing

High load $\rightarrow$ make a bigger array, rehash, get small load
- Akin to expanding backing array in ArrayList
- Allocate a new larger array
- Copy over all active items to the new array
- Array should have prime number size
- $O(n)$ to rehash
Hash Tables in Java

java.util.HashMap Map built from hashing
java.util.HashSet Set built from hashing
java.util.Hashtable Map built from hashing, earlier class, synchronized for multithread apps
Hash Take-Home

- Provide $O(1)$ add/remove/contains
- Separate chaining is a pragmatic solution
  - Hash buckets have lists
- Open Address Hashing
  - Look in a sequence of buckets for an object
- Linear probing is one way to do open address hashing
  - Simple to implement: look in adjacent buckets
  - Performance suffers load approaches 1
  - Primary clustering hurts performance
- Quadratic probing is another way to do open address hashing
  - Prevents primary clustering
  - Must keep hash half-empty to guarantee successful add
  - Not fully understood mathematically
Hash Tables are another Container

Containers

- Like arrays, linked lists, trees, hash tables
- Have add(x), remove(x), contains(x) methods
  - add(x) put x in the DS
  - removeLast() get rid of "last" item
  - remove(x) take x out of DS
  - contains(x) is x in DS?

Speed Comparisons

- Speeds for array or ArrayList?
- Speeds for LinkedList?
- Speeds for hash table?
Operation Complexities (Speed)

- **add(x):** put x in the DS
- **removeLast():** get rid of "last" item
- **remove(x):** take x out of DS
- **contains(x):** is x in DS?

| | add(x) | removeLast() | remove(x) | contains(x) |
|--------------+----------+--------------+-----------+-------------|
| ArrayList | O(1) | O(1) | O(n) | O(n) |
| LinkedList | O(1) | O(1) | O(n) | O(n) |
| Hash Table | O(1) | X | O(1) | O(1) |

This table is slightly misleading

- Careful of semantics of each operation
- Presence/lack of sorting property
- Set/Map distinctions
- What about space complexity of each?
Hash Table Recap

hashCode()

- What is the return type of the `hashCode()` method?
- Which classes implement the `hashCode()` method?
- Describe the *hash contract* to which implementations of `hashCode()` should adhere.
- Should a `HashTable` class implement `hashCode()`?

Hash Table Lookups

Describe how a `HashTable` locates objects it might contain...

- using Separate Chaining
- using Open Addressing and Linear Probing
- using Open Addressing and Quadratic Probing

Describe how `remove(x)` works

- using Separate Chaining
- using Open Addressing
Hash Table Recap

Load

- What is the load of a hash table and how is it calculated?
- What is the range load can assume in Separate Chaining?
- Is the range different for Open Addressing?
- What is the relation of load to the runtime complexity of hash table methods in Separate Chaining?
- "..." in Open Addressing with Linear Probing?
- "..." in Open Addressing with Quadratic Probing?
- How can the load be reduced?
Hash Table Recap: Draw Some Pictures

Add Stuff
add(x) the objects to the right into a hash table
  ▶ Default array size 7
  ▶ Do not rehash/expand in any case

Do this 3 times
1. Using Separate Chaining
2. Using Open Addressing with Linear Probing
3. Using Open Addressing with Quadratic Probing