The mysterious hash

How can I find out if my delicious hash contains a specific “ingredient” in constant time?

What's in an object?

"Ask not what you can do to an Object, ask what an Object can do to itself"

JFK
plan for the day

- Review big-Oh, worst-case analysis
  - For arrays and ArrayLists, preview of Hashing
  - Why do we care about performance?
  - Why do we care about measuring?
  - Alternatives to worst-case analysis?

- How do you reason about trade-offs
  - Have to know what you're trading

Runtime (Big-oh)

<table>
<thead>
<tr>
<th></th>
<th>Array</th>
<th>(Array)List</th>
<th>(Hash)Set</th>
<th>(Hash)Map</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Get</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contains</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Add

- myArray[3] = 7;
- myArrayList.add(3, 7); //index, element

Runtime (Big-oh)

<table>
<thead>
<tr>
<th></th>
<th>Array</th>
<th>(ArrayList)</th>
<th>(HashSet)</th>
<th>(HashMap)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>O(1)</td>
<td>*O(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Get</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contains</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
preview: amortized analysis

• What happens when you call .add(…)?
  • Internally, what does an ArrayList do?
  • Allocates an internal array of size 16 (e.g.,)
  • When you add the 17th element, what happens?
    • Make new internal array of size 32, copy, use that
  • Big-Oh Complexity of adds 1,2,3,…,16?
  • Big-Oh Complexity of add 17?

• $1+2+4+8+\ldots+N$ where $N = 2^k$

preview: amortized analysis

• $1+2+4+8+\ldots+N$ where $N = 2^k$
  • What's $1+2+4+\ldots+32 = \ldots$?
  • What's $1+2+\ldots+2^k$
  • In terms of $N$, (where $N = 2^k$)

• Constants in big-Oh, what is $2N$, $4N$, $18N$ using $O$-notation?
Get

- `int var = myArray[3];`
- `int var = myArrayList.get(3);`

---

Runtime (Big-oh)

<table>
<thead>
<tr>
<th></th>
<th>Array</th>
<th>(Array)List</th>
<th>(Hash)Set</th>
<th>(Hash)Map</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>O(1)</td>
<td>*O(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Get</strong></td>
<td>O(1)</td>
<td>O(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Contains</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
for(int i = 0; i < myArray.length; i++){
    if(myArray[i] == 7)
        return true;
}

boolean inThere = myArrayList.contains(7);

---

<table>
<thead>
<tr>
<th></th>
<th>Array</th>
<th>(Array)List</th>
<th>(Hash)Set</th>
<th>(Hash)Map</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>O(1)</td>
<td>*O(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Get</td>
<td>O(1)</td>
<td>O(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contains</td>
<td>O(N)</td>
<td>O(N)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Remove

- boolean removed = myArrayList.remove(7);

```
7 2 4 1
```

### Runtime (Big-oh)

<table>
<thead>
<tr>
<th></th>
<th>Array</th>
<th>(Array)List</th>
<th>(Hash)Set</th>
<th>(Hash)Map</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>O(1)</td>
<td>*O(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove</td>
<td>O(N)</td>
<td>O(N)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Get</td>
<td>O(1)</td>
<td>O(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contains</td>
<td>O(N)</td>
<td>O(N)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Runtime (Big-oh)

<table>
<thead>
<tr>
<th></th>
<th>Array</th>
<th>(Array)List</th>
<th>(Hash)Set</th>
<th>(Hash)Map</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>O(1)</td>
<td>*O(1)</td>
<td>O(1)</td>
<td>O(1)</td>
</tr>
<tr>
<td>Remove</td>
<td>O(N)</td>
<td>O(N)</td>
<td>O(1)</td>
<td>O(1)</td>
</tr>
<tr>
<td>Get</td>
<td>O(1)</td>
<td>O(1)</td>
<td>O(1)</td>
<td>O(1)</td>
</tr>
<tr>
<td>Contains</td>
<td>O(N)</td>
<td>O(N)</td>
<td>O(1)</td>
<td>O(1)</td>
</tr>
</tbody>
</table>

### Understanding Objects
• Built in Java function for Object
• All objects inherit .hashCode()
  • You can Override .hashCode() with your own code!

```
Circle[] c = new Circle[numCircles];
for (Circle c; i++)
  c[i], i, colors[i%]
```

• Hash Yourself!

```
String name = "Tabitha";
System.out.println(name.hashCode());
```

111673433
Hashing

- **HashTable**
  - array of fixed size
  - with a key to each location
  - each key is mapped to an index in the table

- **HashCode**
  - Every object has a hashCode
    - integer value
  - In our made-up example
    - Object – joe
    - hashCode – 31
  - Could two different objects have the same hashCode?
Hashing

- Hash function
  - simple to compute
    - example
      - hashCode % (mod) 10
  - Use hash function to calculate key to hash table
Two equal objects should hash to the same place (have the same hash code and key)

if a.equals(b)
then
    a.hashCode() == b.hashCode()
Hashing

- Hash function
  - simple to compute
  - ensure two distinct keys get different cells

2/3/14
Hashing

- Separate Chaining
  - make your table into a list!

```java
if a.equals(b) then a.hashCode() == b.hashCode()

HOWEVER

if a.hashCode() == b.hashCode() then a.equals(b) || !a.equals(b)
```
Built in Java function for Object
All objects inherit .equals()

```java
circle[] c = new Circle[numCircles];
for (int i = 0; i < c.length; i++) {
    if (a.equals(b))
        System.out.println("equal");
    else
        System.out.println("not equal");
}
```
.equals()

- Built in Java function for Object
- All objects inherit .equals()
  - You can Override .equals() with your own code!

```java
public boolean equals(Object obj) {
    if (obj == this) {
        return true;
    }
    if (obj == null || obj.getClass() != this.getClass()) {
        return false;
    }
    YourObjectType temp = (YourObjectType) obj;
```