Compsci 201
Collections, Hashing, Objects

G is for …

- Git
  - Version control that’s ubiquitous
- Garbage Collection
  - Java recycles
- Google
  - How to find Stack Overflow

Announcements

- Assignment P1 due yesterday
  - You are in the grace period through midnight
- APT-3 due Tues, Feb 4
  - Can still turn in Friday til 11:59pm
- Discussion 4 on Feb 3
  - Prediscussion, do before, out today
- Reading on calendar
  - Slowing down ….. Nothing posted…

Breakfast 201 coming…

- Maybe Wed. Feb 5 9:30am…
- 30 minutes, discuss whatever with me
- Enjoy breakfast
- Up to 25 people
### Plan for the Day

- **Generic classes: ArrayList to HashSet**
  - From ArrayList to HashSet to Collections to …

- From `Object.equals` to `Object.hashCode`
  - Everything is an Object, what can an object do?

- Maps, Interfaces, Analysis
  - Next week and next assignment

### ArrayList Review

- **What is an ArrayList?**
  - A class that "wraps an array"
  - Part of `java.util.Collections` hierarchy
  - Almost an array: constant-time access to any element given an index (independent of N)

- How are elements added?
  - New array allocated, values copied, continue

### DIYAD ArrayList

- **Do It Yourself Algorithm and Datastructure**
  - SimpleStringArrayList: some methods
  - GrowableStringArrayList: more methods

- Differences between +100, +1000, and *2
  - Helper methods are private: `checkSize()`

### SimpleStringArrayList

- **DIYAD - I want to write an ArrayList class**
- State to define an array
- Methods to
  - Constructor - Create an array – fixed size
  - Add an element to an array
  - Get an element from an array
SimpleStringArrayList (part 1)

```java
public class SimpleStringArrayList {
    private String[] myStorage;
    private int mySize;
    private static int MAX_SIZE = 10000;

    public SimpleStringArrayList() {
        myStorage = new String[MAX_SIZE];
        mySize = 0;
    }

    public void add(String s) {
        if (mySize < myStorage.length) {
            myStorage[mySize] = s;
            mySize++;
        } else {
            throw new ArrayIndexOutOfBoundsException("out of storage at "+mySize);
        }
    }
}
```

SimpleStringArrayList (part 2)

```java
public int size() { return mySize; }

public String get(int index) {
    if (0 <= index && index < mySize) {
        return myStorage[index];
    }
    throw new ArrayIndexOutOfBoundsException("out of range with "+index);
}
```

GrowableStringArrayList

- DIYAD – write another ArrayList Class

DIYAD ArrayList

- Do It Yourself Algorithm and Datastructure
  - SimpleStringArrayList: some methods
  - GrowableStringArrayList: more methods

- Differences between these two classes?
  - Growable – grows as needed, not static
GrowableStringArrayList (part 1)

```java
public class GrowableStringArrayList {
    private String[] myStorage;
    private int mySize;
    private final static int MAX_SIZE = 500;

    public GrowableStringArrayList() {
        myStorage = new String[MAX_SIZE];
        mySize = 0;
    }

    public void add(String s) {
        checkSize();
        myStorage[mySize] = s;
        mySize++;
    }
}
```

GrowableStringArrayList (part 2)

```java
private void checkSize() {
    if (mySize >= myStorage.length) {
        String[] storage = new String[(int)(myStorage.length * 2)];
        System.arraycopy(myStorage, 0, storage, 0, myStorage.length);
        myStorage = storage;
    }
}

public void add(int index, String s) {
    if (index < 0 || index > mySize) {
        throw new IndexOutOfBoundsException("bad index in add "+index);
    }
    checkSize();
    System.arraycopy(myStorage, index, myStorage, index+1, mySize-index);
    myStorage[index] = s;
    mySize++;
}

public int size() { return mySize; }
```

Analysis via Pictures Again

- Growing array by doubling each time
  - Create/copy 1, 2, 4, 8, 16, ... \(2^N\)
- If \(X = 2^N\), we've created \(2 \times 2^{N-1}\), or \(2X - 1\)
  - Roughly \(X\), where "roughly" defined later
Analysis of Diyad ArrayLists

- **SimpleStringArrayList**
  - Add 10,000 strings? ok. Add one more? BAD

- **GrowableStringArrayList**
  - Add as many strings as memory allows, how?

- **ConformingArrayList**
  - Is-a java.util.List, also stores any Object type
  - Must implement List methods, interface

DIYAD Ideas

- Move from String to GrowableString to Generic
  - Lots of work to fit in with Collections hierarchy
  - For our own work? Easier! All of Java? Harder!

  - Differences between +10, +1000, *2 and *1.2
    - How do we measure empirically
    - How do we measure analytically
    - Private method `checkSize()`

Diyad ArrayList Growth

- When internal array full? Create new, copy, use
  - Efficient add, get, set when done repeatedly
  - Not efficient if resize with +1, +100, +1000
    - Is possible if resize with *2 or *1.25

```java
public void add(String s) {
    checkSize();
    myStorage[mySize] = s;
    mySize++;
}

private void checkSize() {
    if (mySize >= myStorage.length) {
        String[] storage = new String[myStorage.length + 10000];
        System.arraycopy(myStorage, srcPos:0, storage, destPos:0, myStorage.length);
        myStorage = storage;
    }
}
```

Analysis with Math+Pictures

- If we grow by adding 1 (or 100 or 1000)
  - Copy 1, then 2, then 3, then … then N
  - 1+2+ … + N = N(N+1)/2
    - Same as 100+200+300+…
    - Roughly N²
  - Divide by 2, multiply by 100
Analysis via Math+Pictures Again

- Growing array by doubling each time
  - Create/copy 1, 2, 4, 8, 16, ... $2^N$
  - Total is $1 + 2 + ... + 2^N = 2^{N+1} - 1$
- If $X = 2^N$, we've created $2 \times 2^{N-1}$, or $2X - 1$
  - Roughly $X$, where "roughly" defined later

Run times summarized

- Re-sizing geometrically and additively
  - Allocate new array, copy all pointers/references

<table>
<thead>
<tr>
<th>grow with x 2</th>
<th>size</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000000</td>
<td>0.028</td>
<td></td>
</tr>
<tr>
<td>2000000</td>
<td>0.037</td>
<td></td>
</tr>
<tr>
<td>3000000</td>
<td>0.053</td>
<td></td>
</tr>
<tr>
<td>4000000</td>
<td>0.066</td>
<td></td>
</tr>
<tr>
<td>5000000</td>
<td>0.117</td>
<td></td>
</tr>
<tr>
<td>6000000</td>
<td>0.121</td>
<td></td>
</tr>
<tr>
<td>7000000</td>
<td>0.143</td>
<td></td>
</tr>
<tr>
<td>8000000</td>
<td>0.211</td>
<td></td>
</tr>
<tr>
<td>9000000</td>
<td>0.270</td>
<td></td>
</tr>
<tr>
<td>10000000</td>
<td>0.260</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>grow with x 1.25</th>
<th>size</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000000</td>
<td>0.051</td>
<td></td>
</tr>
<tr>
<td>2000000</td>
<td>0.087</td>
<td></td>
</tr>
<tr>
<td>3000000</td>
<td>0.117</td>
<td></td>
</tr>
<tr>
<td>4000000</td>
<td>0.153</td>
<td></td>
</tr>
<tr>
<td>5000000</td>
<td>0.218</td>
<td></td>
</tr>
<tr>
<td>6000000</td>
<td>0.338</td>
<td></td>
</tr>
<tr>
<td>7000000</td>
<td>0.303</td>
<td></td>
</tr>
<tr>
<td>8000000</td>
<td>0.398</td>
<td></td>
</tr>
<tr>
<td>9000000</td>
<td>0.452</td>
<td></td>
</tr>
<tr>
<td>10000000</td>
<td>0.468</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>grow with +10000</th>
<th>size</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000000</td>
<td>1.507</td>
<td></td>
</tr>
<tr>
<td>2000000</td>
<td>1.585</td>
<td></td>
</tr>
<tr>
<td>3000000</td>
<td>2.740</td>
<td></td>
</tr>
<tr>
<td>4000000</td>
<td>5.146</td>
<td></td>
</tr>
<tr>
<td>5000000</td>
<td>7.304</td>
<td></td>
</tr>
<tr>
<td>6000000</td>
<td>8.315</td>
<td></td>
</tr>
<tr>
<td>7000000</td>
<td>10.428</td>
<td></td>
</tr>
<tr>
<td>8000000</td>
<td>14.233</td>
<td></td>
</tr>
<tr>
<td>9000000</td>
<td>21.434</td>
<td></td>
</tr>
<tr>
<td>10000000</td>
<td>21.927</td>
<td></td>
</tr>
</tbody>
</table>

Diyad ArrayList Summary

- If we grow additively: +1, or +100, or +1000
  - Performance is quadratic, for an array of N elements we expect $N^2$ time (allocate/copy)

- If we grow geometrically: *2, *1.2, *3
  - Performance is linear, for an array of N elements we expect N time (allocate copy)

- Ignore constants: $N^2 / 2$ or 100*N or 200N or ...
Coding is today's language of creativity. All our children deserve a chance to become creators instead of consumers of computer science.

I personally believe that the most important thing we have to do today is use technology to address societal problems, especially in developing regions.

Can E be anything? String, Point, …

- Method `.equals` that works as expected for E!
  - Internal array `myStorage` contains Objects
  - `ConformingArrayList<String>`
    - What `.equals` is called? Object or String?
    - Runtime decision, *not compile time* decision
    - What does elt reference point to? String!!

Why Diyad?

- Traditionally `use ArrayList<E> -- client code`
  - Understand methods via API
  - Problem solving in many contexts
  - Efficiency: `a.get(1)` as fast as `a.get(1000)`

- Why efficient? Understanding by analysis
  - From the internal array which is efficient
  - From doubling on resize rather than adding one
Toward Applications

- We can speak with a limited vocabulary
  - Learn vocabulary then speak, then read

- We can also write code similarly
  - Eventually debugging may require understanding how `equals` works

Massive Data sets

- How do we find what #hashtags are trending on Twitter in real-time?
  - 6,000 tweets/second, 350,000/minute, …
  - Do we weight by tweeter-importance?

- Must be able to look up very quickly, cannot skim through all hashtags/all data
  - Conveniently, we use hashing and hash tables!

Toward Understanding HashSet

- Adding objects to HashSet<..>, avoid duplicates
  - We'll see with Point class, doesn't work
  - We'll see with String class, does work
  - Just as we needed to add `equals()` …
    - We need to add `hashCode()`

- Need some knowledge of Object and internals of HashSet<..>, how does `set.add(X)` work?
  - Every object can convert itself to a number
  - Ask not what you can do to an object …

Making `.contains` efficient

- Why is ArrayList.contains(..) slow?
  - Search through entire list to find something
  - If list is sorted can we do better?
    - Think of a number between 1 and 1,024, I'll tell you high, low, correct: how many guesses needed?

- How do you search for a book in the stacks?
  - That's not what you do in the stacks?
  - What about in ancient times …
Simple Example Hashing
Want a mapping of Soc Sec Num to Names

- Duke's CS Student Union wants to be able to quickly find out info about its members. Also add, delete and update members. Doesn't need members sorted.
  - 267-89-5431  John Smith
  - 703-25-6141  Jack Adams
  - 319-86-2115  Betty Harris
  - 476-82-5120  Rose Black
- Hash Table size is 0 to 10
- Possible Hash Function: \( H(ssn) = \text{last 2 digits mod 11} \)

Have a list of size 11 from 0 to 10

- Insert these into the list
- Insert as (key, value) tuple
  - (267-89-5431, John Smith)
  - (703-25-6141, Jack Adams)
  - (319-86-2115, Betty Harris)
  - (476-82-5120, Rose Black)

H(267-89-5431) = 31 \% 11 = 9
  - John Smith
H(703-25-6141) = 41\%11 = 8
  - Jack Adams
H(319-86-2115) = 15 \% 11 = 4
  - Betty Harris
H(476-82-5120) = 20\%11 = 9
  - Rose Black

Have a list of size 11 from 0 to 10

- Insert these into the list
- Insert as (key, value) tuple
  - (267-89-5431, John Smith)
  - (703-25-6141, Jack Adams)
  - (319-86-2115, Betty Harris)
  - (476-82-5120, Rose Black)

H(267-89-5431) = 31 \% 11 = 9
  - John Smith
H(703-25-6141) = 41\%11 = 8
  - Jack Adams
H(319-86-2115) = 15 \% 11 = 4
  - Betty Harris
H(476-82-5120) = 20\%11 = 9
  - Rose Black
Another way: Put similar together in an ArrayList

- Insert these into the list
- Insert as (key, value) tuple
  (267-89-5431, John Smith)
  (in example, only showing name)

H(267-89-5431) = 31 %11 = 9
  John Smith
H(703-25-6141) = 41%11 = 8
  Jack Adams
H(319-86-2115) = 15%11 = 4
  Betty Harris
H(476-82-5120) = 20%11 = 9
  Rose Black

Finding an Object's number ..

- Every object has `.hashCode()` method
  - Returns int value, used as “locker number”
  - Could return 39, 2, 57, … even -321
  - Ideally uses properties of object to compute

- Cannot guarantee different for every Object!
  - Search items in same locker
  - Use `.equals` find in locker

Ideal world? Real world!

Hash Metaphor and Pseudocode

- Finite number of lockers, or buckets, table entries
  - Each locker stores ArrayList for hash collisions
    - In real world, might be another structure in locker

- Given object, find it's locker/bucket number
  - locker # == `o.hashCode() % table_size`

- Search through locker to see if target there
  - `for(Object o : locker) if o.equals(target)`
Point.hashCode

• Convert a Point to a number
  • Try to make every point a different number
  • That's not possible!!
    • For method below, what non-equal points have same .hashCode()?

```
@Override
public int hashCode() {
    return (int) (myX*1000 + myY);
}
```

Inefficient but Correct .hashCode

• Suppose .hashCode() simply returns 5
  • Every Point goes in the same locker
  • There are always collisions, but we try to minimize them. How are collisions resolved?

• Can we modify PointDriver.java to stress-test?
  • How many different points can be made?

The hashCode contract

• Every object has .hashCode() method
  • Inherited from Object, but typically overridden
  • Use @Override and read online

• Must respect .equals(): If a.equals(b) ?
   • a.hashCode() == b.hashCode()
   • Converse not true! There will be collisions

When Strings Collide

• Generate strings that will collide
  • Find such strings in the wild
    • [Link to String.java]

<table>
<thead>
<tr>
<th>String</th>
<th>hashCode</th>
<th>String</th>
<th>hashCode</th>
</tr>
</thead>
<tbody>
<tr>
<td>ayay</td>
<td>3009136</td>
<td>buzzards</td>
<td>-931102253</td>
</tr>
<tr>
<td>ayBZ</td>
<td>3009136</td>
<td>righto</td>
<td>-931102253</td>
</tr>
<tr>
<td>bZay</td>
<td>3009136</td>
<td>snitz</td>
<td>109586548</td>
</tr>
<tr>
<td>bZbZ</td>
<td>3009136</td>
<td>unprecludible</td>
<td>109586548</td>
</tr>
</tbody>
</table>
WOTO (correctness counts)


Work in 201

• How important are APTs?
  • How important are APT quizzes?

• How important are assignments?
  • Earlier assignments, later assignments?

• How important: reading and WOTO in-class
  • How important are reading quizzes?