Announcements/Follow-ups

• Final exam
  – This Friday August 2
  – Normal class period – 80 minutes

• P7(‘s) due Friday 11pm
  – Corrections: subMatrix, matrix mult.
  – Equals method and Round-off error
  – ArrayList copy constructor

• Wednesday’s topics

• Grades

• Reflectance
Wednesday’s Topics

• **Graphical User Interface** programming in Java and multi-threading
• Regular expressions and Finite automata
• Other programming languages (C, matlab, ruby, Ocaml)
• Combinatorics and probability
• Neural modeling
Grades

• Midterm 2 has been curved on grades server
• Overall percentage grades so far:
  – Average 81%
  – Standard deviation: 16%
• Remaining grades:
  – P6 style, P7 (5.5 %)
  – Quiz 5, Lab 10 (1.2%)
  – Final exam (24%)
  – Participation (5%)
Reflectance

• The ability of your source code to “hold up a mirror” and inspect itself.

• Allows things like:
  – Manipulating fields (including private ones)
  – Converting between source code and Strings
    • “BasicSoldier”
    • “ShinyCoin”

• Requires certain execution environments
  – Not available in Applets (used on the internet)
The Class class

• Represents a data-type
  – Reference types: objects, enums
  – Primitive types

• Is generic!

• All constructors are *private*. Methods are used for acquiring a Class object:
  – myObj.getClass();
  – myPrimitive.class;
  – Class.forName("...");
  – And more: Reflectance example
The Reflection API

- Acquiring Class objects is just the tip of the iceberg of a large API:
  - `myClass.newInstance();` //instantiates the class
  - `myClass.getMethods();` //enumerates the methods
  - There is a Method class
    - `myMethod.invoke(...);`
  - There is a Type class for generic type parameters
  - There is an Annotation class
    - `@Override`
    - `@Test`
More on Garbage collection

• Eligibility for garbage collection:
  – Objects are *eligible* for collection once no more reference variables contain their address
  – Circular references are also eligible if they are unreachable from external code
  – GarbageCollection example

• It may be a long time before an eligible object is actually collected
  – You can suggest a round of collection with `System.gc()`, but this does not guarantee that collection actually occurs
More on Garbage collection

• Whenever the collector frees an object, its finalize() method is called (inherited from Object)
  – used to make sure resources (e.g. file streams) get released

• The foregoing is true for most JVM implementations
  – Assume for the exam
  – The official JVM spec does not actually specify the underlying garbage collection algorithm!
Command line Java

• Your source-code filename.java can be written in any text editor

• Compiling source-code to byte-code on the command line:
  – Stand-alone source:
    javac filename.java
  – With libraries:
    javac -cp path1\lib1.jar;path2\lib2.jar filename.java
    • “cp” stands for “classpath”: directory paths where Java searches for files

• Running byte-code in the JVM:
  – Stand-alone:
    java filename
    • No extension
  – With libraries:
    java -cp .;path1\lib1.jar;path2\lib2.jar filename
    • ‘;’ in front

• Mac: ‘:’ instead of ‘;’ to separate paths
Command line Java

• Packaging your library into a Java AR chive:
  – jar cf output.jar inputdirectory
  – ‘cf’ for create file
  – A JAR is essentially a zip archive

• Extracting files from an existing JAR:
  – jar xf library.jar
Algorithmic Complexity

• The “complexity” of an algorithm refers to the time and space required.

• Complexity is typically a function of the input size.

• Given a list of size n:
  
  – how many steps are required by linear search, in terms of n?

  – How many steps are required by binary search, in terms of n?
int n = array.length;
for(int i = 0; i < n; i++) {
    if(array[i] == query) return i;
}
return -1;

20?
Complexity of Linear search

```java
int n = array.length;
for(int i = 0; i < n; i++) {
    if(array[i] == query) return i;
}
return -1;
```

< n steps?

20?

```plaintext
0  1  2  3  4  5  6  7  8  9  10
16 3 22 7 9 8 11 1 18 4
```
Complexity of Linear search

```java
int n = array.length;
for(int i = 0; i < n; i++) {
    if(array[i] == query) return i;
}
return -1;
```

< 4n steps?

20?
Complexity of Linear search

```java
int n = array.length;
for (int i = 0; i < n; i++) {
    if (array[i] == query) return i;
}
return -1;
```

< 4n+2 steps?

20?

0 1 2 3 4 5 6 7 8 9
16 3 22 7 9 8 11 1 18 4

❌
 Complexity of Linear search

int n = array.length;
for(int i = 0; i < n; i++) {
    if(array[i] == query) return i;
}
return -1;

< mn+b steps
Complexity of Binary search

\[
n/1 \quad \begin{array}{ccccccccccccccccc}
\end{array} \quad \text{hi}
\]
## Complexity of Binary Search

<table>
<thead>
<tr>
<th>( n/1 )</th>
<th>( \text{low} )</th>
<th>( \text{hi} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( n/2 )</th>
<th>( \text{low} )</th>
<th>( \text{hi} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Complexity of Binary search

\[
\begin{array}{l}
\frac{n}{4} & \text{low} & \text{hi} \\
\hline
\end{array}
\]
### Complexity of Binary search

<table>
<thead>
<tr>
<th>$n/1$</th>
<th>low</th>
<th>hi</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n/2$</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$n/4$</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$n/8$</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Complexity of Binary search

Steps: \( \frac{n}{2^T} = 1 \rightarrow n = 2^T \rightarrow \log_2 n = T \)
Linear vs Binary

![Graph showing linear vs binary growth patterns with logarithmic and linear scales.](image-url)
Asymptotic Complexity

• Solving $\log_2 x < mx + b$ numerically (with a computer)
  – BigO example

• Key ideas of asymptotic notation:
  – Only behavior for large input sizes is significant
    • Small input is processed in milliseconds, with either algorithm
  – Constant factors are not important
    • Small fixed values of $m = 0.1, 0.05, 0.025$, etc. don’t help much when input size is huge
  – Focus is often on the worst-case scenarios
    • When linear search goes to the end of the list
    • When binary search goes to a length 1 sub-list
“Big-O” notation

• Roughly speaking (when x gets big enough):
  – “log₂ x < x”
  – “1000x < x²”
  – “x¹⁰⁰⁰ < 2ˣ”
  – “1000ˣ < xˣ”

• Roughly speaking (relative to higher-degree polynomials, exponentials, logarithms, etc.):
  – “x³ + 3x² + 3x + 1 = x³”
“Big-O” notation

- Formal notation:
  - $\log_2 x = O(x)$
  - $1000x = O(x^2)$
  - $x^{1000} = O(2^x)$
  - $1000^x = O(x^x)$

- Formal notation:
  $$x^3 + 3x^2 + 3x + 1 = O(x^3)$$
\[ f(x) = O(g(x)) \]

**Definition:** \( \exists M, x_0 : x > x_0 \Rightarrow |f(x)| \leq M|g(x)| \)

\[ x_0 \approx 12 \]

\[ M = 2 \]
Induction and Recursion: Josephine’s Problem

• Queen Josephine rules a land of female logicians and their husbands
• At least one husband is unfaithful
• Wives know all husbands’ fidelities but their own
• Gunshots can be heard by all
• As soon as a woman deduces her own husbands fidelity, she must shoot him at midnight the same day
• After several nights, all women make the correct deductions. How?
Induction and Recursion

5. Friday? The day after Thursday.
4. Thursday? The day after Wednesday
3. Wednesday? The day after Tuesday.
2. Tuesday? The day after Monday.
1. Monday? The first day of the week.

BASE CASE
Induction

Sums of odds:
• $1 = 1$
• $1 + 3 = 4$
• $1 + 3 + 5 = 9$
• $1 + 3 + 5 + 7 = 16$
• $1 + 3 + 5 + 7 + 9 = 25$
• $\sum_{i=1}^{n} 2n - 1 = n^2$  ???
Induction

Fermat Numbers: \( 2^{2^n} + 1 \)

- \( 2^{2^1} + 1 = 5 \) is prime.
- \( 2^{2^2} + 1 = 17 \) is prime.
- \( 2^{2^3} + 1 = 257 \) is prime.
- \( 2^{2^4} + 1 = 65537 \) is prime.
- \( 2^{2^5} + 1 = 4294967297 \) is not prime! (Euler)
Induction

Sums of odds:
• $1 = 1 = 1^2$. (Base Case)
• Suppose $\sum_{i=1}^{n} 2n - 1 = n^2$ ... (Inductive Hypothesis)
• Does $\sum_{i=1}^{n+1} 2n - 1 = (n + 1)^2$ ??? (Inductive proof)

Sum of Naturals:

$1+2+3+4+\ldots+(n-2)+(n-1)+n = \frac{n(n+1)}{2}$ ???
Recursion: Induction in reverse

\[ n! = n \times (n - 1)! \quad \text{Friday} \]
\[ (n - 1)! = (n - 1) \times (n - 2)! \quad \text{Thursday} \]
\[ (n - 2)! = (n - 2) \times (n - 3)! \quad \text{Wednesday} \]
...
\[ 1! = 1 \quad \text{Monday (Base Case)} \]

Writing recursive methods:
1. Handle the base case!!!
2. Assume the method already works at every sub-step, and call it.
3. Combine the result of the recursive call with the current step.

Recursion Examples