PTFW (and assignments)

- Review maps, toward analytical and empirical analysis of...
  - everything
- How are objects compared in Java?
  - When would you want to compare?
  - What can’t be compared?
- Empirical and Analytical Analysis
  - How do we analyze code?
  - How do we discuss code?
  - How do we analyze algorithms?
  - How do we discuss algorithms?

Topdown v Bottomup

- Programming is changing our world
  - Empowering, liberating, equalizing,…
- Everything is a bit: all 0’s and 1’s
  - From jpg to mp3 to …
- It’s about problems! It’s about details!
  - Should we think about problems to get to the details?
  - Should we master details before grand thinking?
- See Wikipedia on topdown v bottomup design

From data to information

- Data that’s organized can be processed
  - Is this a requirement?
  - What does “organized” means
- Purpose of map in Markov assignment?
  - Properties of keys?
  - Comparable v. Hashable
- TreeSet v. HashSet
  - Speed v. order
  - Memory considerations

Analysis: Algorithms and Data Structures

- We need a vocabulary to discuss performance
  - Reason about alternative algorithms and implementations
  - It’s faster! It’s more elegant! It’s safer! It’s less filling!
- Need empirical tests, analytical/mathematical tools
  - Given two methods, which is better? Run them to check.
    - 30 seconds vs. 3 seconds, easy. 5 hours vs. 2 minutes, harder
    - What if it takes two weeks to implement the methods?
  - Use mathematics to analyze the algorithm,
  - The implementation is another matter, cache, compiler optimizations, OS, memory,…
How fast does the code run?

- “As soon as an Analytical Engine exists, it will necessarily guide the future course of the science. Whenever any result is sought by its aid, the question will then arise — by what course of calculation can these results be arrived at by the machine in the shortest time?”
  - Babbage, Analytical Engine 1864
- Question: What’s the fastest way to sort a million 32-bit integers?
- Obama’s answer: I don’t think the bubble sort is the way to go

Quantitative Measurements of Code

- Typically measure running time (memory?)
  - Other things to measure?
  - What about wall-clock v CPU time? Java: wall-clock
- Typically change size of input/problem to validate runtime hypotheses
  - Not the data itself, but the number of data items
  - Size of string vs. number of strings in array?
- Doubling hypothesis: What effect does doubling input size have on running time?
  - Linear: time doubles, quadratic: factor of four, ...
  - Log analysis for power laws

Different measures of complexity

- Worst case
  - Gives a good upper-bound on behavior
  - Never get worse than this
  - Drawbacks?
- Average case
  - What does average mean?
  - Averaged over all inputs? Assuming uniformly distributed random data?
  - Drawbacks?
- Best case
  - Linear search, useful?

Anita Borg 1949-2003

- "Dr. Anita Borg tenaciously envisioned and set about to change the world for women and for technology. … she fought tirelessly for the development technology with positive social and human impact."
- “Anita Borg sought to revolutionize the world and the way we think about technology and its impact on our lives.”
  - http://www.youtube.com/watch?v=1yPxd5jqg_Q
**Frequencies and MapAnalyzer.java**

- **Code interlude**
  - Why do we look at code for examples?

- **Difference between HashMap and TreeMap?**
  - What goes in them. Contract with users. Details?
  - What comes out of the maps, keysets, values,....

- **How does hashing and comparing work?**
  - What does an object need?
  - What are runtime guarantees?

---

**From Comparable to Comparator**

- **When a class implements Comparable** then ...
  - Instances are comparable to each other
    - "apple" < "zebra", 6 > 2
  - Sorting Strings, Sorting other objects, ...
  - Method compareTo invoked when ...
  - Comparable<..> types the parameter to compareTo
  - Return < 0, == 0, > 0 according to results of comparison

- **Suppose we want to change how Strings compare**
  - Or change class Foo implements Comparable<Foo>
  - What if we need more than one way to compare Foo's?

---

**Ask not what you can do to an object...**

- **What can all objects do to themselves?**
  - Print themselves – call toString()
  - Hash themselves – call hashCode()
  - Clone themselves – call clone()
    - Can only call if Cloneable interface implemented

- **What kind of objects can sort themselves**
  - None, but some can be sorted!
  - How do you sort them? Make sure they’re comparable!
  - compareTo() and the Comparable<..> interface

---

**Running times @ 10^6 instructions/sec**

<table>
<thead>
<tr>
<th>N</th>
<th>O(log N)</th>
<th>O(N)</th>
<th>O(N log N)</th>
<th>O(N^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.000003</td>
<td>0.00001</td>
<td>0.000033</td>
<td>0.0001</td>
</tr>
<tr>
<td>100</td>
<td>0.000007</td>
<td>0.00010</td>
<td>0.000664</td>
<td>0.1000</td>
</tr>
<tr>
<td>1,000</td>
<td>0.00010</td>
<td>0.00100</td>
<td>0.010000</td>
<td>1.0</td>
</tr>
<tr>
<td>10,000</td>
<td>0.00013</td>
<td>0.01000</td>
<td>0.132900</td>
<td>1.7 min</td>
</tr>
<tr>
<td>100,000</td>
<td>0.00017</td>
<td>0.10000</td>
<td>1.661000</td>
<td>2.78 hr</td>
</tr>
<tr>
<td>1,000,000</td>
<td>0.00020</td>
<td>1.0</td>
<td>19.9</td>
<td>11.6 day</td>
</tr>
<tr>
<td>1,000,000,000</td>
<td>0.00030</td>
<td>16.7 min</td>
<td>18.3 hr</td>
<td>318 centuries</td>
</tr>
</tbody>
</table>