TAFTW (Take Aways for the Week)

- Graded work this week:
  - APT Quiz, details and overview
  - Markov assignment, details and overview

- Concepts: Empirical and Analytical Analysis
  - Algorithms and Data Structures
  - Benchmarking and empirical analyses
  - Terminology, mathematics, analytical analyses

- Java idioms: Interfaces: general and Comparable
- Software Engineering: Unit Testing and JUnit
APT Quiz and Markov Overview

● APT Quiz – meant to demonstrate mastery of concepts. If you don't do this now, you'll have an opportunity to demonstrate mastery later
  ➢ Self check on where you are, help us too
  ➢ Validate your own work with APTs
  ➢ It's ok to do a green dance, partial dance ok too!

● Markov Assignment
  ➢ Basics of Java Objects, real/interesting scenario
  ➢ Do not leave this until the last two days
Comparing objects and trade-offs

- How are objects compared in Java?
  - When would you want to compare?
  - What can’t be compared?

- Empirical and Analytical Analysis
  - Why are some lists different?
  - Why is adding in the middle fast?
  - Why is adding in the middle slow?

- How do you measure performance?
From Comparable to … TreeMap/Sort

- When a class implements `Comparable` then ...
  - Instances are comparable to each other
    - "apple" < "zebra", 6 > 2
    - Sorting Strings, Sorting WordPairs, ...
    - Method `compareTo` invoked when ...

- `Comparable<...>` types the parameter to `compareTo`
  - Return < 0, == 0, > 0 according to results of comparison
Strings: simple Comparable

- Strings compare themselves lexicographically aka Dictionary order
  - "zebra" > "aardvark", but "Zebra" < "aardvark"
  - You can't use <, ==, > with Strings
- "zebra".compareTo(s) returns < 0 or == 0 or > 0
  - According to less than, equal to, greater than
- Helper: "zebra".compareToIgnoreCase(s)

- implements Comparable<String> means?
  - Requires a method, what about correctness?
Comparable?
Liberté, Egalité, Comparable

- Can we compare points?
  - [https://courses.cs.washington.edu/courses/cse331/11sp/lectures/slides/04a-compare.pdf](https://courses.cs.washington.edu/courses/cse331/11sp/lectures/slides/04a-compare.pdf)

- Key take-away: Comparable should be consistent with equals
  - If `a.equals(b)` then `a.compareTo(b) == 0`
  - Converse is also true, e.g., if and only if
How do we compare points?

- Naïve approach? First compare x, then y? Let's look at .equals(..) first
  - Why is parameter an Object?
  - Everything is an Object!

```java
public boolean equals(Object o) {
    if (o == null || ! (o instanceof Point)) {
        return false;
    }
    Point p = (Point) o;
    return p.x == x && p.y == y;
}
```
How do we compare points?

- Naïve approach? First compare x, then y? Let's look at `.compareTo(..)`
  - Why is parameter a Point?

```java
public int compareTo(Point p) {
    if (this.x < p.x) return -1;
    if (this.x > p.x) return 1;
    if (this.y < p.y) return -1;
    if (this.y > p.y) return 1
    return 0;
}
```
Useful math trick

- Use subtraction to help with return values
  
  [Link to Stack Overflow](http://stackoverflow.com/questions/2654839/rounding-a-double-to-turn-it-into-an-int-java)

```java
public int compareTo(Point p) {
    int deltaX = (int) Math.round(x - p.x);
    int deltaY = (int) Math.round(y - p.y);
    if (deltaX == 0) return deltaY;
    return deltaX;
}
```
Comparable and Interfaces


Some questions look at KWICModel.java, code we've previously examined in class. But now looking at interfaces
Empirical and Analytical Analysis

● We can run programs to look at "efficiency"
  ➢ Depends on machine, environment, programs

● We can analyze mathematically to look at efficiency from a different point of view
  ➢ Depends on being able to employ mathematics

● We will work on doing both, leading to a better understanding in many dimensions
What is a `java.util.List` in Java?

- **Collection of elements, operations?**
  - Add, remove, traverse, ...
  - What can a list do to itself?
  - What can we do to a list?

- **Why more than one kind of list: Array and Linked?**
  - Useful in different applications
  - How do we analyze differences?
  - How do we use them in code?
What’s the Difference Here?

- How does find-a-track work? Fast forward?
public double removeFirst(List<String> list) {
    double start = System.nanoTime();
    while (list.size() != 1){
        list.remove(0);
    }
    double end = System.nanoTime ();
    return (end-start)/1e9;
}
List<String> linked = new LinkedList<String>();
List<String> array = new ArrayList<String>();
double ltime = splicer.removeFirst(splicer.create(linked,100000));
double atime = splicer.removeFirst(splicer.create(array,100000));

● Time taken to remove the first element?
https://git.cs.duke.edu/201fall16/building-arrays/blob/master/src/ListSplicer.java
Remove First in 2011

<table>
<thead>
<tr>
<th>Size $10^3$</th>
<th>link</th>
<th>array</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.003</td>
<td>0.045</td>
</tr>
<tr>
<td>20</td>
<td>0.001</td>
<td>0.173</td>
</tr>
<tr>
<td>30</td>
<td>0.001</td>
<td>0.383</td>
</tr>
<tr>
<td>40</td>
<td>0.002</td>
<td>0.680</td>
</tr>
<tr>
<td>50</td>
<td>0.002</td>
<td>1.074</td>
</tr>
<tr>
<td>60</td>
<td>0.002</td>
<td>1.530</td>
</tr>
<tr>
<td>70</td>
<td>0.003</td>
<td>2.071</td>
</tr>
<tr>
<td>80</td>
<td>0.003</td>
<td>2.704</td>
</tr>
<tr>
<td>90</td>
<td>0.004</td>
<td>3.449</td>
</tr>
<tr>
<td>100</td>
<td>0.007</td>
<td>4.220</td>
</tr>
</tbody>
</table>
Remove First in 2016

- Why are timings good?
- Why are timings bad?

<table>
<thead>
<tr>
<th>Size $10^3$</th>
<th>link</th>
<th>array</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.0036</td>
<td>0.0102</td>
</tr>
<tr>
<td>20</td>
<td>0.0016</td>
<td>0.0375</td>
</tr>
<tr>
<td>30</td>
<td>0.0012</td>
<td>0.0756</td>
</tr>
<tr>
<td>40</td>
<td>0.0003</td>
<td>0.2228</td>
</tr>
<tr>
<td>50</td>
<td>0.0003</td>
<td>0.235</td>
</tr>
<tr>
<td>60</td>
<td>0.0004</td>
<td>0.2945</td>
</tr>
<tr>
<td>70</td>
<td>0.0005</td>
<td>0.3975</td>
</tr>
<tr>
<td>80</td>
<td>0.0007</td>
<td>0.5456</td>
</tr>
<tr>
<td>90</td>
<td>0.0006</td>
<td>0.7091</td>
</tr>
<tr>
<td>100</td>
<td>0.0007</td>
<td>0.8827</td>
</tr>
</tbody>
</table>
Analytical Analysis

- Since LinkedList is roughly linear
  - Time to remove first element is constant, but must be done N times
  - Vocabulary, time for one removal is O(1) — constant and doesn't depend on N
  - Vocabulary, time for all removals is O(N) — linear in N, but slope doesn't matter
- For ArrayList, removing first element entails ...
  - Shifting N-1 elements, so this is O(N)
- All: (N-1) + (N-2) + ... + 3 + 2 + 1 = O(N²)
  - Sum is (N-1)N/2
Interfaces

● What is an interface? What does Google say?
  ➢ Term overloaded even in English
  ➢ What is a Java Interface?

● Abstraction that defines a contract/construct
  ➢ Implementing requires certain methods exist
    • For example, Comparable interface?
  ➢ Programming to the interface is enabling
    • What does Collections.sort actually sort?

● IDE helps by putting in stubs as needed
  ➢ Let Eclipse be your friend
Why use Interfaces?

- **Implementation can vary without modifying code**
  - Code relies on interface, e.g., `addFront` or `removeMiddle`
  - Argument passed has a concrete type, but code uses the interface in compiling
- **Actual method called determined at runtime!**

- **Similar to API, e.g., using the Twitter API**
  - Calls return JSON, the format is specified, different languages used to interpret JSON
Markov Interlude: JUnit and Interfaces

- How do we design/code/test EfficientMarkov?
  - Note: it implements an Interface!
  - Note: MarkovTest can be used to test it!

- How do we design/code/test WordGram?
  - Can we use WordGram tester when first cloned?
  - Where is implementation of WordGram?
  - How do you make your own?
JUnit tests

- To run these must access JUnit library, jar file
  - Eclipse knows where this is, but ...
  - Must add to build-path aka class-path, Eclipse will do this for you if you let it

- Getting all green is the goal, but red is good
  - You have to have code that doesn't pass before you can pass
  - Similar to APTs, widely used in practice

- Testing is extremely important in engineering!
  - See also QA: quality assurance
JUnit Interlude

- Looking at PointExperiment classes:
  - [https://git.cs.duke.edu/201fall16/pointExperiment/tree/master/src](https://git.cs.duke.edu/201fall16/pointExperiment/tree/master/src)

- Create JUnit tests for some methods, see live run through and summary

- JUnit great for per-method testing in isolation from other methods
public double removeMiddleIndex(List<String> list) {
    double start = System.nanoTime();
    while (list.size() != 1){
        list.remove(list.size()/2);
    }
    double end = System.nanoTime();
    return (end-start)/1e9;
}

● What operations could be expensive here?
  ➢ Explicit: size, remove (only one is expensive)
  ➢ Implicit: find n\textsuperscript{th} element
Remove Middle 2011

<table>
<thead>
<tr>
<th>size</th>
<th>link</th>
<th>array</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.105</td>
<td>0.023</td>
</tr>
<tr>
<td>20</td>
<td>0.472</td>
<td>0.09</td>
</tr>
<tr>
<td>30</td>
<td>0.984</td>
<td>0.192</td>
</tr>
<tr>
<td>40</td>
<td>1.83</td>
<td>0.343</td>
</tr>
<tr>
<td>50</td>
<td>3.026</td>
<td>0.534</td>
</tr>
<tr>
<td>60</td>
<td>4.288</td>
<td>0.767</td>
</tr>
<tr>
<td>70</td>
<td>6.078</td>
<td>1.039</td>
</tr>
<tr>
<td>80</td>
<td>7.885</td>
<td>1.363</td>
</tr>
</tbody>
</table>

Middle Index Removal

\[
y = 0.1332x^2 - 0.0849x + 0.0679
\]

\[
y = 0.0212x^2 - 0.0002x + 0.0032
\]
Remove Middle 2016

<table>
<thead>
<tr>
<th>size</th>
<th>link</th>
<th>array</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.0635</td>
<td>0.0057</td>
</tr>
<tr>
<td>20</td>
<td>0.2644</td>
<td>0.0131</td>
</tr>
<tr>
<td>30</td>
<td>0.4808</td>
<td>0.0345</td>
</tr>
<tr>
<td>40</td>
<td>0.8524</td>
<td>0.0531</td>
</tr>
<tr>
<td>50</td>
<td>1.4025</td>
<td>0.0844</td>
</tr>
<tr>
<td>60</td>
<td>1.8418</td>
<td>0.1245</td>
</tr>
<tr>
<td>70</td>
<td>2.9064</td>
<td>0.1777</td>
</tr>
<tr>
<td>80</td>
<td>3.7237</td>
<td>0.2224</td>
</tr>
<tr>
<td>90</td>
<td>4.6833</td>
<td>0.3102</td>
</tr>
<tr>
<td>100</td>
<td>7.8717</td>
<td>0.3824</td>
</tr>
</tbody>
</table>

Remove Middle Index 2016

- Array list: $y = 0.11x^2 - 0.4587x + 0.6957$, $R^2 = 0.97136$
- Linked list: $y = 0.0045x^2 - 0.0082x + 0.0125$, $R^2 = 0.99827$
ArrayList and LinkedList as ADTs

- As an ADT (abstract data type) ArrayList supports
  - Constant-time or $O(1)$ access to the $k$-th element
  - Amortized linear or $O(n)$ storage/time with add
    - Total storage used in $n$-element vector is approx. $2n$, spread over all accesses/additions (why?)
    - Add/remove in middle is "expensive" $O(n)$, why?

- What's underneath here? How Implemented?
  - Concrete: array – contiguous memory, must be contiguous to support random access
  - Element 20 = beginning + 20 x size of a pointer
ArrayList and LinkedList as ADTs

- **LinkedList as ADT**
  - Constant-time or O(1) insertion/deletion anywhere, but...
  - Linear or O(n) time to find where, sequential search

- **Linked good for add/remove at front**
  - Splicing into middle, also for 'sparse' structures

- **What's underneath? How Implemented**
  - Low-level linked lists, self-referential structures
  - More memory intensive than array: two pointers
Inheritance and Interfaces

- Interfaces provide method names and parameters
  - The method signature we can expect and use!
  - What can we do to an ArrayList? To a LinkedList?
  - What can we do to a Map or Set or PriorityQueue?
  - java.util.Collection is an interface

- New in Java 8: Interfaces can have code!
Nancy Leveson: Software Safety

Founded the field

- Mathematical and engineering aspects
  - Air traffic control
  - Microsoft word

"C++ is not state-of-the-art, it's only state-of-the-practice, which in recent years has been going backwards"

- Software and steam engines once deadly dangerous?

- THERAC 25: Radiation machine killed many people
Big-Oh, O-notation: concepts & caveats

● Count how many times “simple” statements execute
  ➢ In the body of a loop, what matters? (e.g., another loop?)
  ➢ Assume statements take a second, cost a penny?
    • What's good, what’s bad about this assumption?

● If a loop is inside a loop:
  ➢ Tricky because the inner loop can depend on the outer, use math and reasoning

● In real life: cache behavior, memory behavior, swapping behavior, library gotchas, things we don’t understand,…
More on O-notation, big-Oh

- Big-Oh hides/obscures some empirical analysis, but is good for general description of algorithm
  - Allows us to compare algorithms in the limit
  - $20N$ hours vs $N^2$ microseconds: which is better?

- O-notation is an upper-bound, this means that $N$ is $O(N)$, but it is also $O(N^2)$; we try to provide tight bounds.
More on O-notation, big-Oh

- O-notation is an upper-bound, this means that \( N \) is \( O(N) \), but it is also \( O(N^2) \); we try to provide tight bounds. Formally:
  - A function \( g(N) \) is \( O(f(N)) \) if there exist constants \( c \) and \( n \) such that \( g(N) < cf(N) \) for all \( N > n \).
Notations for measuring complexity

● **O-notation/big-Oh:** \( O(n^2) \) is used in algorithmic analysis, e.g., Compsci 330 at Duke. Upper bound in the limit
  - Correct to say that linear algorithm is \( O(n^2) \), but useful?

● **Omega is lower bound:** \( \Omega(n \log n) \) is a lower bound for comparison based sorts
  - Can't do better than that, a little hard to prove
  - We can still engineer good sorts: TimSort!
Simple examples of array/loops: O?

for(int k=0; k < list.length; k += 1) {
    list[k] += 1;   // list.set(k, list.get(k)+1);
}
//-----

for(int k=0; k < list.length; k += 1)
    for(int j=k+1; j < list.length; j += 1)
        if (list[j].equals(list[k]))
            matches += 1;
//---

for(int k=0; k < list.length; k += 1)
    for(int j=k+1; j < list.length; j *= 2)
        value += 1;
Multiplying and adding big-Oh

- Suppose we do a linear search then do another one
  - What is the complexity? $O(n) + O(n)$
  - If we do 100 linear searches? $100 \times O(n)$
  - If we do $n$ searches on an array of size $n$? $n \times O(n)$

- Binary search followed by linear search?
  - What are big-Oh complexities? Sum?
  - What about 50 binary searches? What about $n$ searches?
What is big-Oh about?

- **Intuition:** avoid details when they don’t matter, and they don’t matter when input size \( N \) is big enough
  - Use only leading term, ignore coefficients
    
    \[
    \begin{align*}
    y &= 3x & y &= 6x-2 & y &= 15x + 44 \\
    y &= x^2 & y &= x^2-6x+9 & y &= 3x^2+4x
    \end{align*}
    \]

- **The first family is \( O(n) \), the second is \( O(n^2) \)
  - Intuition: family of curves, generally the same shape
  - Intuition: linear function: double input, double time, quadratic function: double input, quadruple the time
Some helpful mathematics

- \( 1 + 2 + 3 + 4 + \ldots + N \)
  - \( N \frac{(N+1)}{2} \), exactly = \( N^2 / 2 + N / 2 \) which is \( \mathcal{O}(N^2) \) why?

- \( N + N + N + \ldots + N \) (total of \( N \) times)
  - \( N * N = N^2 \) which is \( \mathcal{O}(N^2) \)

- \( N + N + N + \ldots + N + \ldots + N + \ldots + N \) (total of \( 3N \) times)
  - \( 3N * N = 3N^2 \) which is \( \mathcal{O}(N^2) \)

- \( 1 + 2 + 4 + \ldots + 2^N \)
  - \( 2^{N+1} - 1 = 2 \times 2^N - 1 \) which is \( \mathcal{O}(2^N) \) – in terms of last term, call it \( X \), this is \( \mathcal{O}(X) \)