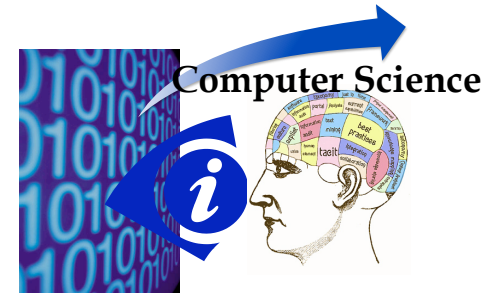


## COMPSCI 100, Fall 2009 Owen Astrachan

<http://www.cs.duke.edu/courses/cps100/fall09>  
<http://www.cs.duke.edu/~ola>

## Data into Information and Knowledge



## What is Computer Science?

What is it that distinguishes it from the separate subjects with which it is related? What is the linking thread which gathers these disparate branches into a single discipline? My answer to these questions is simple --- *it is the art of programming a computer*. It is the art of designing efficient and elegant methods of getting a computer to solve problems, theoretical or practical, small or large, simple or complex.

C.A.R. (Tony)Hoare

## Programming != Computer Science

- What is the nature of intelligence? How can one predict the performance of a complex system? What is the nature of human cognition? Does the natural world 'compute'?
- It is the interplay between such fundamental challenges and the human condition that makes computer science so interesting. The results from even the most esoteric computer science research programs often have widespread practical impact. Computer security depends upon the innovations in mathematics. Your Google search for a friend depends on state-of-the-art distributed computing systems, algorithms, and artificial intelligence.

<http://www.post-gazette.com/stories/co/04186/341012.stm>

## Efficient design, programs, code

Using the language: Java (or C++, or Python, or ...), its idioms, its idiosyncracies

Object-oriented design and patterns. Software design principles transcend language, but ...

Know data structures and algorithms. Trees, hashing, binary search, sorting, priority queues, greedy methods, graphs ...

Engineer, scientist: what toolkits do you bring to programming? Mathematics, design patterns, libraries --- standard and others...

## Course Overview

- **Active Lectures, Recitations, Quizzes, Programs**
  - Recitation based on current assignments and topics
    - Discuss answers, answer new questions, small quiz
    - More opportunities for questions to be answered.
  - Active Lectures based on readings, questions, programs
    - Online quizzes used to motivate/ensure reading
    - In-class questions used to ensure understanding
  - Programs
    - Theory and practice of data structures and (OO) programming
    - Fun, practical, tiring, ...
    - Weekly APT programs and longer programs
- **Exams/Tests**
  - Semester: open book
  - Final: open book

## What's in Compsci 100?

- **Understanding tradeoffs: reasoning, analyzing, describing...**
  - Algorithms
  - Data Structures
  - Programming
  - Design
- **Object oriented programming using Java**
  - Eclipse, JDK, Ambient, ...
  - Language, Design Patterns, Design Methodologies ...
  - Problem-solving
  - From design to code

## Questions

If you gotta ask, you'll never know  
Louis Armstrong: "What's Jazz?"



If you gotta ask, you ain't got it  
Fats Waller: "What's rhythm?"



What questions did you ask today?  
Arno Penzias

## Tradeoffs

Programming, design,  
algorithmic, data-  
structural

Simple, elegant, quick,  
efficient: what are our  
goals in programming?  
What does XP say about  
simplicity? Einstein?

Fast programs, small  
programs, run anywhere-  
at-all programs. Runtime,  
space-time, your time,  
CPU time...

How do we decide what  
tradeoffs are important?  
Tension between  
generality, simplicity,  
elegance, ...

## From Blog to Scientific Visualization

- Text Cloud aka Tag Cloud?
  - > Number of occurrences/emphasis indicated by size of word
  - > Great visual/statistic: <http://chir.ag/phernalia/preztags/>



- What is involved with generating tag clouds?
  - > Steps? Issues?
  - > See `SimpleTagMaker.java`

## Analysis of SimpleTagMaker

- Which classes used have static methods?
  - > Do methods in CSSMaker need to be non-static? Why?
  - >
- How would we create a tag cloud from a file of words?
  - > What steps are needed? Algorithmic? Java?
- Why is there an `IOException` that's thrown from main?
  - > What is an exception, how do we handle them?
- Any questions?

## Problem Solving and Programming

- How many words are in a file?
  - > What's a word?
  - > What's a file?
  - > How do we solve this: simply, quickly, ...?
    - What's the best we can do? Constraints?
- How many different/unique words are in a file?
  - > How is this related to previous task?
- How many words do two files have in common?
  - > Spell-checking, Google did you mean ..?

## Fast, cheap, out-of-control?

- This is valid and correct Java code, questions?

```
import java.util.*;
import java.io.*;
public class SimpleUnique {
    public static void main(String[] args)
        throws FileNotFoundException{
        Scanner s = new Scanner(new File("/data/kjv10.txt"));
        String[] words = s.useDelimiter("\\Z").next().split("\\s+");
        TreeSet<String> set = new TreeSet<String>();
        set.addAll(Arrays.asList(words));
        System.out.printf("total #: %d, unique #: %d\n",
            words.length, set.size());
    }
}
```

## How fast is fast? How cheap is cheap?

- How do we measure how fast the code/design is?
  - > Can we implement this design in C++?
  - > Can we implement this in Python?
- We want a measure that's independent of language?
  - > What are we measuring?
  - > How do we express answer?
  - > Units? Best case? Average? Worst?
- What is answer using recognized terminology?
  - >
  - >

## What is Computer Science?

- Computer science is no more about computers than astronomy is about telescopes.



Edsger Dijkstra

- Computer science is not as old as physics; it lags by a couple of hundred years. However, this does not mean that there is significantly less on the computer scientist's plate than on the physicist's: younger it may be, but it has had a far more intense upbringing!



Richard Feynman

<http://www.wordiq.com>

## Tracking different/unique words

- We want to know how many times 'the' occurs
  - > Do search engines do this? Does the number of occurrences of "basketball" on a page raise the priority of a webpage in some search engines?
    - Downside of this approach for search engines?
- Constraints on solving this problem
  - > We must read every word in the file (or web page)
  - > Search for the word? Avoid counting twice? Store?
  - > Are there fundamental limits on any of these operations?  
*Where should we look for data structure and algorithmic improvements?*

## Faster? Slower? How does this work?

```
public class SortingUniqueCounter {
    public static int uniqueCount(String[] list) {
        Arrays.sort(list);
        String last = list[0];
        int count = 1;
        // Invariant: count is number of unique words in list[0..k]
        for (int k = 1; k < list.length; k++) {
            if (!list[k].equals(last)) {
                last = list[k];
                count++;
            }
        }
        return count;
    }
}
```

## Some Java Vocabulary and Concepts

- **Java has a huge standard library**
  - > Organized in *packages*: java.lang, java.util, javax.swing, ...
  - > API browseable online, but Eclipse IDE helps a lot
- **Java methods have different kinds of access inter/intra class**
  - > Public methods ...
  - > Private methods ...
  - > Protected and Package methods ...
- **Primitive types (int, char, double, boolean) are not objects but everything else is literally an instance of class Object**
  - > `foo.callMe()` ;

## Basic data structures and algorithms

- **Arrays are typed and fixed in size when created**
  - > Don't have to fill the array, but cannot expand it
  - > Can store int, double, String, ...
- **ArrayList (and related interface List) grows**
  - > Stores objects, not primitives
    - Autoboxing in Java 5 facilitates int to/from Integer conversion
  - > Old code requires casts, ArrayLists typed as storing Objects
  - > ArrayList objects grow themselves intelligently
- **java.util package has lots of data structures and algorithms**
  - > Use rather than re-implement, but know how to do both

## Search: measuring performance

- **How fast is fast enough?**

```
/**
 * @return true if key in a, else return false
 */
boolean search(String[] a, String key){
    for(int k=0; k < a.length; k++)
        if (a[k].equals(key)) return true;
    return false;
}
```
- **Java details: parameters? Return values? ArrayLists?**
  - > See next page for alternate code
- **How do we measure performance of code? Of algorithm?**
  - > Does processor make a difference? Dual-core? Quad-core? 64-bit? Cell processor? ...

## Structuring Information: ideas & code

- Is an element in an array, Where is an element in an array?
  - > DIY: use a loop
  - > Use Collections, several options
  - > Tradeoffs?

```
public boolean contains(String[] list,
                        String target){
    for(String s : list){
        if (s.equals(target)) return true;
    }
    return false;
}
```

```
public boolean contains(String[] list, String target){
    return Arrays.asList(list).contains(target);
}
```

```
public boolean contains(String[] list, String target){
    return
        new HashSet<String>(Arrays.asList(list)).contains(target);
}
```

## Class/OO/Java tradeoffs

- If you search a list of elements once, you must “touch” every element, so why worry about doing anything else?
  - > Can you skip an element when searching? Why?
  - > What about a sorted list of elements?
- If you search repeatedly, it makes sense to organize data
  - > Leverage or amortize the cost of the organization over several searches
  - > Where do we store the organized data so that we can access it repeatedly?

## Who is Alan Perlis?

- It is easier to write an incorrect program than to understand a correct one
- Simplicity does not precede complexity, but follows it
- If you have a procedure with ten parameters you probably missed some
- If a listener nods his head when you're explaining your program, wake him up
- Programming is an unnatural act
- Won first Turing award



<http://www.cs.yale.edu/homes/perlis-alan/quotes.html>

## Computer Science in a Nutshell

Web Images Groups News Froogle more x  
Google Search | I'm Feeling Lucky



[Advertising Programs](#) - [Business Solutions](#) - [About Google](#)

©2004 Google - Searching 8,058,044,551 web pages