What is “computer science”?

Computer science is engineering
- math
- cognitive science
- linguistics
- business

The course website will have all the information you need: http://www.cs.uoregon.edu/classes/06S/cis170
- check the site often for announcements, schedule changes, etc

Computer Science is Engineering
- Engineers design and build artifacts
  - based on scientific knowledge of the area
  - evaluate cost/benefit tradeoffs
  - use professional standards
- Examples
  - structural engineering (buildings, bridges, ...)
  - electrical engineering (power grids, electronics, ...)
- Computer engineering: design and construction of computer systems
  - logic chips, power supplies, fans, packaging, ...
- Software engineering: design and implementation of computer applications
  - analyze requirements, specify functionality, write code, test
  - what science supports this field of engineering?
Computer Science is Math

- Mathematicians deal with formal systems
  - logic, deductive and inductive proof
  - algebraic formulas, transformations
  - abstractions that may have many applications
- Algorithms are formal descriptions of methods for solving problems
- Computer scientists often prove properties of algorithms
  - common methods for sorting \( n \) items can take \( n^{3/2} \) steps or \( n \times \log_2 n \) steps

<table>
<thead>
<tr>
<th></th>
<th>( n )</th>
<th>( n^{3/2} )</th>
<th>( \log_2 n )</th>
<th>( n \times \log_2 n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>poker hand</td>
<td>5</td>
<td>12</td>
<td>2.32</td>
<td>12</td>
</tr>
<tr>
<td>bridge hand</td>
<td>13</td>
<td>85</td>
<td>3.70</td>
<td>48</td>
</tr>
<tr>
<td>full deck</td>
<td>52</td>
<td>1352</td>
<td>5.70</td>
<td>296</td>
</tr>
</tbody>
</table>

Computer Science is Cognitive Science

- Cognitive psychologists study mental processes, e.g.
  - memory
  - perception
  - decision making
- Computer scientists use concepts from cognitive science when designing software
  - graphical user interfaces (organized so users find them easier to work with)
  - scientific visualization (summarizing, displaying large amounts of data)
- Artificial intelligence aims to build software to do many things humans do
  - machine learning (e.g. discriminate cancerous from normal cells)
  - planning (most effective allocation of scarce resources)
  - natural language processing (e.g. machine translation)

Computer Science is Linguistics

- The study of computer languages has been influenced by many ideas from linguistics
  - grammars and grammar rules
  - parsing (e.g. “diagramming” a sentence)
- Computer science applications include
  - regular expressions and pattern matching
  - analysis of DNA and protein sequences
  - natural language processing

Computer Science is Business

- Concepts from economics and business administration play a big role in computer science
- Software engineering:
  - understanding customers and their requirements
  - end-user support (training, documentation, ...)
  - group projects, co-operative design
  - OMSE: approximately 1/3 MBA style courses
- Custom vs commodity components
  - Linux “cluster” vs SMP multiprocessor
The Science of Computing

- The “science” in computer science includes
  - algorithms: what are the most efficient methods for solving problems?
  - languages: what are the best ways to express algorithms?
  - software engineering: how can we build useful and reliable programs?
  - computer engineering: how can we build cost-effective computer systems?
- Computer science helps people solve problems
  - science
  - engineering
  - medicine
- Computer science helps people be more effective or creative
  - architecture
  - communications
  - music and the arts

Computer Science Research

- Universities
  - UO, OSU, nearly every university in the US and abroad
- Government labs
  - Department of Energy (PNL, Sandia, Oak Ridge, ...)
  - National Institutes of Health (NLM, ...)
- Industry labs
  - IBM, Xerox, Bell Labs in the 1970s and 80s
  - Microsoft, Intel, IBM, Google, ...
  - Pharmaceutical companies, automobile industry, ...

CS in a Liberal Arts University

- At the University of Oregon the CIS Department is part of the College of Arts and Sciences
  - At many other universities CS is in Engineering (e.g. EECS, CSE, ...)
  - Many universities have also established a College of Computing, where CS is just one of many departments
- CIS majors take
  - 54 credits CIS
  - 32 credits Math
  - 12 credits science (physics, chemistry, biology, or psychology)
  - Technical writing
- Our philosophy: critical thinking and an interdisciplinary viewpoint are just as important as technical skills

Why This Course?

- The goals for CIS 170 are to give students an introduction to computer science
  - a brief introduction to key ideas
  - short projects on selected topics, e.g.
    - there will be lectures on gene sequence databases and methods used to search for patterns in data
    - a project will involve searching an internet database that stores descriptions of genes in the human genome
- Motivation:
  - fun, interesting science course for non-CS majors
  - broad overview of the field for pre-CS majors
**Textbook**

- The textbook this term is *The New Turing Omnibus*, by A. K. Dewdney
- Dewdney is a professor at Western Ontario University
  - teaches introduction to CS
  - interested in computational biology
- He is also a science writer
  - long-time columnist for *Scientific American*
- This book is a collection of essays on key ideas in computer science
- We will only have time for a few chapters
- There will be additional handouts and on-line references for most topics

No, that's not a typo: it's Turing, not Touring

**Textbook (cont’d)**

- I have mixed feelings about using this book as a textbook
  - Pro:
    - very interesting topics
    - written for general audience
    - challenging but not overwhelming
  - Con:
    - old, with very dated descriptions
    - most topics need additional explanations
- An important goal for my lecture slides:
  - augment the textbook
  - provide additional information you need for problem sets, projects, and exams

No, that's not a typo: it's Turing, not Touring

---

**Turing**

- Alan Turing (1912-1954) was a British mathematician
- Considered by many the “founding father” of modern computer science
- John’s Book Club recommendation: *Alan Turing: The Enigma*, by Andrew Hodges
  - One of Turing’s major accomplishments was the design of a machine to break the Enigma code used by Germany in WWII

Note tie clip... “Honorary Member” of CIS Faculty Office: South wall of Deschutes Hall

**On-Line Resources**

- In addition to the textbook, you will find a lot of course information on-line
  - the “resources” link on the class web page will take you to a page that has links to web sites with related information
  - there are also two electronic books you can read on-line
  - this page is still evolving -- send me suggestions if you find anything interesting
- We also have a “Wiki Web”
  - a wiki is an interactive and collaborative web site -- people who visit the site can add new pages, or edit existing pages
  - for our class: you can post questions, answer other students’ questions, ...
  - click on the “wiki” link on the class home page
Course Outline

- Topics from NTO we plan to cover include:
  - algorithms
  - automata
  - strings and algorithms on strings
    - exact match (finding a substring)
    - sorting
    - inexact match (finding similar strings)
  - grammars
  - machine learning
  - databases
  - computer networks

Problem Sets, Projects, and Exams

- There will be a set of weekly homework assignments
  - problem sets: paper-and-pencil assignments, similar to those in a math class
  - projects: short "labs" where you will use software to explore that week's topic
    - example: algorithms on strings
      - problem set will have questions about patterns used in string search
      - project will use software that searches for specified patterns in sets of DNA strings (some from a local file, some from the internet)
- There will be two exams
  - one covers topics from the first half of the course
  - the other (during finals week) covers topics from the second half
  - the best way to study for the exam: do the problem sets and projects!

Grading Policy

- Your grade will be based on
  - scores on exams
  - grades on problem sets and projects
  - class participation
    - answering questions, participating in discussions, etc
    - posting to the Wiki web is a good way to contribute
  - extra credit

- Policy for late work:
  - turn in whatever you have completed by the deadline
  - anything extra submitted after the deadline may be considered for extra credit
    (if we don’t discuss solutions in class)
  - no extra credit unless you submit something by the initial deadline

Why Bioinformatics?

- The plan this term is to choose computer science topics related to a common theme
- Bioinformatics refers to the use of computer and information technology in biology
  - molecular biology: study of DNA, RNA, proteins
  - developmental biology: early growth and differentiation of cells
  - evolutionary biology: speciation and diversification of groups of organisms
- We will learn general concepts of algorithms, languages, and computer systems, and use bioinformatics applications as motivating examples
Ruby

- CIS 170 does not require experience in computer programming
- This course is not an introduction to programming: there will be no programming assignments
- There are, however, lab projects using a programming language named Ruby
- Why do we say these are “labs” and not “programming”?
  - programming involves planning and preparation
  - a programming project typically involves design, coding, and testing
  - all but the simplest programming projects require two weeks or more
- CIS 170 labs will require between one and three hours
- Using a system like Ruby gives you a chance to experiment and explore the concepts

Ruby (cont’d)

- Ruby is a general-purpose programming language
- it can be used to write large applications
- In CIS 170 we will use it as a “workbench”
  - load some data into the system
  - type in statements that process the data, produce some results
  - write statements that implement algorithms, test them on the data
- You have a choice for running Ruby:
  - download and install it on your own computer -- it’s free
  - see instructions in the resources section of the class web page
  - use a machine in one of the UO computer center labs
    - Klamath B13
    - McKenzie 101

About This Course

- Spring 2006 is the second time this course has been offered
  - first time we’ve used Ruby
  - the first time I’ve taught it
  - the first time I’ve taught a 100-level class
- Expect a work in progress
  - late changes in projects or lecture topics
- I’ll try to post assignments a week in advance
  - two weeks before they are due
- I’ll put lecture notes on-line
  - the first 2 weeks are there now
  - future slides may not make it to the web page until after the lecture

About the Instructors

- John Conery
  - professor of computer and information science
  - 23 years at UO
  - research: programming languages, parallel processing, bioinformatics
  - Ruby experience: novice (read 2 books, written 3 programs)
- Tom Bulatewicz
  - graduate student in computer and information science
  - Ph.D. expected June 2006
  - just found out last week he needs to learn Ruby
A Note About Class Participation

- The class wiki is one way to participate
  - post questions, comments, references to information you found, ...
  - think of the wiki as a 10-week chat
  - don’t worry about ruining a page -- just let us know and we can back up and restore an old version

- Asking questions in class is another way to contribute
  - do the assigned reading before class
  - come ready to ask questions
  - don’t be shy about asking questions in class

- Laptops in class are OK
  - especially if you run Ruby -- try examples during class? -- or view slides
  - but be considerate of others -- no web browsing or non-class related activities

Getting Started

- Purchase the textbook
  - read chapter 1 (Algorithms) before Friday
  - read chapter 2 (Automata) before next Wednesday

- Figure out how you will run Ruby
  - don’t wait until the first project is due
  - options
    - download to your own computer, test the installation
    - find one of the computer center labs, make sure you can run Ruby there

Important: Assignment 0

- Send e-mail to conery@uoregon.edu
  - make sure “CIS 170” is in the subject line
  - use your official UO e-mail account
  - let me know what you want me to call you (first name, nickname, ...)
    - you can call me “John”
  - list your previous experience with computers
    - e-mail and web only
    - writing papers, using databases, etc
    - some programming experience

- I will respond before Friday

- When you send me e-mail:
  - I can start building my gradebook
  - I can make sure your mail isn’t thrown out by my spam filter