CIS 170: The Science of Computing
http://www.cs.uoregon.edu/classes/09W/cis170

What is “computer science”?
Course outline
lecture topics
projects
exams
Getting started

Course Information

Professor:
- John Conery
  conery@cs.uoregon.edu

GTF:
- Shad Stafford
  staffors@cs.uoregon.edu

All the information you need (announcements, syllabus, schedule, lecture notes) is online at the class web site:
http://www.cs.uoregon.edu/classes/08S/cis170

Use Blackboard to download labs, submit completed labs, and check grades:
https://blackboard.uoregon.edu

About the Instructors

John Conery
- professor of computer and information science
- 26th year at UO
- research: programming languages, parallel processing, bioinformatics
- teaching: mostly 400/500/600 level CIS classes on computer architecture and scientific applications

Shad Stafford
- Ph.D. student in computer and information science
- research interest in computer networks
- was GTF for CIS 170 in 2007

Today’s Topic

CIS 170 is a general introduction to computer science
- intended for non-majors and pre-CIS majors

Today’s lecture:
- what is computer science?
- what parts of computer science will we study this term?
- course organization
- projects, exams, grading policy
What is Computer Science?

- From the title of the field, one might think computer science is “the study of computers”
- A quote from a famous computer scientist: “Computer Science is no more about computers than astronomy is about telescopes” Edsger Dijkstra (1930 – 2002)
- If CS is not the study of computers, what is it?
- To answer this question, let’s first ask what computers are, and what they can do -- then come back to the main question of “what is computer science?”

Electronic Brains

- When computers first started being widely used in the 1950s the general public viewed them as “electronic brains”
  - they were huge, expensive, mysterious
  - found only in the largest corporate, government, or university labs
  - sci-fi movies exploited this “electronic brain” viewpoint
- Why a “brain” instead of some other analogy?
  - the calculations performed by computers were the sorts of things only highly trained people were able to do
  - example: compute the trajectory of a rocket

Appliances

- Today computers are essential tools in modern society
- Personal computers (desktops and laptops)
  - write papers, manage personal finances, ....
  - entertainment: games, video, audio, ....
- Business computers
  - day-to-day operations: payroll, billing, ...
  - customer service: web sites, customer data, ...
- Embedded computers
  - microchips used as controllers in cars, phones, buildings, ....
- Supercomputers
  - large “number crunchers” used in scientific research and other areas
  - parallel processing: from a few dozen to a few thousand CPU chips

What Computers Can’t Do

- One way to approach the question of “what is a computer?” is to look at some of the things we don’t expect a computer to be able to do

Problem 1: Send e-mail to a person you don’t know
- The problem: you need to have an e-mail address
- In some cases you can do some research and track down an address (e.g. admissions office at a university)
- In other cases (e.g. someone you met at a party) you might be stuck
- An important point: a person would have a hard time with this problem, too
  - unless you supply more information (e.g. name of the host of the party) a friend would also be stuck
What Computers Can’t Do (cont’d)

Problem 2: Which school should I go to?
- We normally don’t expect computers to solve important personal problems.
- To choose a university, you probably used factors such as
  - subject areas (does the school have programs that interest you?)
  - quality (some measures are objective, like SAT scores of entering class, but other factors are subjective)
  - location (objective: travel time from home; subjective: quality of life)
- People make these decisions after weighing all the factors.
- Would you trust a computer to solve this problem?
- Again the issue here is not the computer; it’s the nature of the problem.
  - you don’t expect another person to “solve” this problem for you, either
  - you get recommendations, but not “solutions”
  - the problem: how to quantify the subjective factors like quality of life.

What Computers Can’t Do (cont’d)

Problem 3: Invite friends to watch a DVD
- It would be nice if our cell phones were more like personal assistants.
  - pick up the phone, say “invite Erica and Katie over to watch One Tree Hill.”
  - your phone would negotiate with their phones to pick a time that works.
- As of January 2009 this is mostly a fanciful idea.
  - computers are getting better at understanding speech.
  - computers also manage schedules and find common openings.
  - but the sort of detailed negotiation human assistants do is not (yet) possible.
- Unlike the previous two problems, here we have an example of something humans can do that machines (so far) cannot.
  - will a future computer be able to solve this problem?

What Computers Can’t Do (cont’d)

Problem 4: The perfect game of chess
- You might think it would be easy for a computer to win a game of chess.
  - The rules of the game are simple, and it’s straightforward to write a program that would have a computer examine all possible moves.
  - The problem: there are too many moves to consider.
    - by most estimates there are around $10^{43}$ possible games.
    - even if a supercomputer could evaluate one trillion ($10^{12}$) different board positions each second, it would take $10^{31}$ seconds, or about $10^{21}$ years, to look at them all.
    - some perspective: physicists estimate the age of the universe to be $10^{13}$ years....
  - So here we have a new type of limitation: a practical limit.
  - As was the case for the first two problems, people are no better than machines at performing this task.
    - grand masters do not consider all possible moves....

What Computers Can’t Do (cont’d)

Problem 5: Non-computable functions
- A famous problem in computer science is known as “the halting problem.”
  - The goal: write a program that would determine if another program is stuck.
    - example: you’re writing a paper and the pointer changes from an arrow into the “busy” icon.
    - usually the icon changes back after a second or two, but after a minute you begin to worry.
    - it would be nice to have a “halt checker” program that you could run and ask it whether the word processor has crashed or if you just need to be patient.
  - This problem is related to paradoxes in logic (e.g. “this statement is false”)
    - later this term we will see a proof that it is impossible to write a halt check program.
  - This is a new type of limitation -- a mathematical barrier instead of a technological barrier -- but once again people are no better than machines at solving this problem.
Recap: Computational Limits

- An interesting theme showed up on the previous slides.
- Some tasks that computers cannot perform are also impossible for humans.
- The difficulty is in the nature of the problem, not the person or thing trying to solve it:
  - not well specified (e-mail to unknown person)
  - attributes not quantifiable (quality of life at a university)
  - impractical (chess)
  - impossible (halt checker)

What Computers Can Do

- From this discussion of problems computers cannot solve we can draw some conclusions about the nature of problems they can solve.
  1. The problem must have a well-specified starting point.
  2. There must also be a well-defined outcome.
  3. The solution is obtained by following a specific set of instructions.

Computation

- A computation is a sequence of well-defined operations that lead from an initial starting point to a desired final outcome.
  - note this definition does not include the word “computer”
  - a computation is a process that can be carried out by a person or a machine
  - the same computation might be carried out using any one of a number of different technologies

Example: Average Age

- As an example of a computation, suppose we want to know the average age of a group of students.
  - in this case “average” means “arithmetic mean”
  - method: compute the sum of the ages, then divide by the number of students
  - For a small group one could use paper and pencil or a hand calculator.
  - For larger groups (e.g., average age of entering freshman class) one would probably use a computer.

<table>
<thead>
<tr>
<th>Name</th>
<th>Date of Birth</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sanchez, Maria</td>
<td>Feb 14, 1988</td>
<td>20</td>
</tr>
<tr>
<td>Sanders, Eric</td>
<td>Mar 24, 1978</td>
<td>30</td>
</tr>
<tr>
<td>Sato, Noriko</td>
<td>Oct 14, 1989</td>
<td>18</td>
</tr>
<tr>
<td>Singer, Fred</td>
<td>Apr 30, 1983</td>
<td>25</td>
</tr>
<tr>
<td>Smith, John</td>
<td>Feb 26, 1990</td>
<td>18</td>
</tr>
</tbody>
</table>

Average (Mean) Age: \( \frac{20 + 30 + 18 + 25 + 18}{5} = 22.2 \)
An Old Idea

- The idea that problems can be solved by following a detailed set of instructions is very old
  - ancient Greek, Persian, and Chinese philosophers all developed computational approaches to solving arithmetic problems
  - example: Eratosthenes’ method for finding prime numbers

- Until the 1940s the word “computer” was a job title
  - human computers were people trained to carry out mathematical operations
  - they followed a set of precise instructions required to create mathematical tables
  - used for navigation, astronomy, mathematics

What is Computer Science?

- Computer science is the study of computation
  - investigating problems that can be solved computationally
  - programming languages used to describe computations
  - machines that carry out computations
  - theoretical limits of computation (what is or is not computable)
  - computational solutions to problems in math, science, medicine, business, education, journalism, ...

- Computers play a key role
  - but (getting back to Dijkstra) computer science is not “about computers”

Computer Science Research

- Research in computer science is found in the same organizations that do research in other areas of science
  - Universities
    - UO, OSU, PSU, nearly every university in the US and around the world
  - 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 in a separate college

- At UO 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 perspective are just as important as technical skills
Why This Course?

- Where does CIS 170 fit into the computer science curriculum at UO?
- The goals for CIS 170 are to give students an introduction to computer science
  - a brief introduction to key ideas
  - projects on selected topics
- Motivation:
  - fun, interesting science course for non-CS majors
  - broad overview of the field for pre-CS majors

Textbook

- I have used two different textbooks for this course in the last three years
- I couldn’t find one I really liked, so...
- I will post PDFs of chapters on the course web site
  - only 50% of topics are ready
  - I will find web sites or other free material for the remaining topics

A Laboratory for Computational Experiments

- The main idea behind this book is that the best way to learn computer science is to work on “lab projects”
- Analogy -- labs in intro chemistry classes
  - instructor prepares the materials and methods
  - students follow instructions, write up a report and submit it
  - an advantage of computer labs: no nasty smells or messy explosions...
- Hands-on experience with the subject is more effective than simply reading about it

CIS 170 Labs

- Lab projects in CIS 170 are based on a programming language named Ruby
- Ruby is a general-purpose language that can be used to write large applications
- In CIS 170 we will use Ruby as a “workbench”
  - we will give you the programs and data
  - you will run programs, modify them, see what they do, learn how they work
- Using a system like Ruby gives you a chance to experiment with computations

Ruby
A Programmer’s Best Friend
CIS 170 Labs (cont’d)

- CIS 170 does not require experience in computer programming
- This course is not an introduction to programming, and there will be no programming assignments
- Why do we say CIS 170 projects 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 weeks of concentrated effort  
  - CIS 170 labs will require from one to three hours (if you are prepared)
- The goal for CIS 170 is to use Ruby as a notation for describing computations  
  - you will eventually learn to follow the notation and understand the main concepts  
  - there will be suggestions for optional projects for those who want to try their hand at writing their own Ruby programs

You have a choice for running Ruby

1. Do the labs on your own computer  
   - Ruby is open source software  
   - already installed on Macs (with OS/X 10.4 and 10.5)  
   - easy to download and install on Windows and Linux systems  
     - you can find instructions on the class web page
2. Use a machine in one of the UO computer center labs  
   - Klamath B13  
   - McKenzie 101

Course Outline

- Some of the topics we plan to cover include:
  - data representation  
    - how are numbers, strings, songs, pictures and other things stored in a computer?
  - algorithms  
    - how do machines process the data stored in their memories?  
    - what are the limits to what a machine can do?
  - computer networks  
    - how is data transferred between systems?  
    - how does the internet work?  
    - security, privacy, and other issues
  - databases
  - artificial intelligence  
    - what are some of the issues in getting computers to understand language?

On-Line Resources

- In addition to the textbook, you will find a lot of course information on-line  
  - the “resources” section of the class web page has links to web sites with related information  
  - there is also on-line reference material for the software we will be using  
  - this web site is still evolving -- send me suggestions if you find anything interesting
- Blackboard has facilities for group discussions  
  - for our class: you can post questions, answer other students’ questions, ...
  - I’m still learning how to use it -- if you post a question please e-mail me to remind me to log in....
  - I may also start a blog where you can post comments and get a discussion going
Exams

- There will be two exams
- The midterm, given in class during the 5th week, will cover material from the first four weeks
  - 20% of your grade
- The final will be a comprehensive exam
  - 30% of your grade
  - most questions will be based on material from the last five weeks
- The best way to study for the exams: do the projects!
  - the “short answer” questions in labs will be very similar to the sorts of questions that will be on the exam

Grading Policy

- Your grade will be based on
  - scores on exams
  - grades on labs
  - class participation
    - answering questions, participating in discussions, etc
    - posting to the on-line discussion 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

A Note About Class Participation

- The on-line discussion groups are one way to participate
  - post questions, comments, references to information you found, ...
  - think of the discussion group as a 10-week chat
- 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
  - but be considerate of others -- no web browsing or non-class related activities

First Week

- Tue Jan 6
  - introduction

- Thu Jan 8
  - introduction to Ruby
  - Lab 0: “Hello World”
    - due Mon Jan 12
Getting Started

- Download Chapters 1 and 2 of the textbook
  - note: these PDFs are for your use only
  - please don’t pass them on to anyone outside the class without asking me first

- Figure out how you will run Ruby
  - don’t wait until the first project is due!
  - download and install on your system, or go to one of the labs and make sure you can start and run Ruby (see next slide)

Getting Started (cont’d)

- Log on to Blackboard (http://blackboard.uoregon.edu)
- Make sure you can connect to the CIS 170 pages
- You will be using Blackboard to
  - download lab projects
  - submit completed labs
  - check on your grades
- Download and run Lab 0: “Hello, World”
  - a trivial project to verify that Ruby is installed and working
  - do the lab and submit it (make sure the homework submission process is working)
  - this project is due Jan 12 (next Monday)

Next topic:

Introduction to Ruby