Northwestern University
EECS 101:
An Introduction to Computer Science for Everyone
Spring, 2011

Syllabus

(last updated 3/29/2011)

Course Home Page:

http://www.nucs101.org

What Roles Can This Class Can Play for You?

- For CS majors and minors, EECS 101 is a required core course in the Computer Science Curriculum.
- For Weinberg students, EECS 101 satisfies the Area III (Social and Behavioral Sciences) Distribution requirement.
- For everyone, this course explains the field of Computer Science.

Prerequisites:

- None.
- No prior knowledge of Computer Science is needed.
- All students at Northwestern are welcome.
- There is no programming in this course.

Faculty Instructors:

1. Ming-Yang Kao (coordinator; weeks 1, 2, and 10)
2. Jason Hartline (weeks 3 and 4)
3. Goce Trajcevski (weeks 5 and 6)
4. Bryan Pardo (week 7)
5. Robby Findler (week 8)
6. Jack Tumblin (weeks 9 and 10)

Teaching Assistants:

1. Elham Beheshtizavareh
2. Joseph Friedman

Undergraduate Assistants:

To be announced.

Times and Locations:

- Lectures: Tuesdays and Thursdays, 3:30-4:50 PM, Tech M345.
- Optional Recitation/Discussion Sessions: To be announced.

Who Should Take This Class?

1. All students at Northwestern who are interested in learning about what Computer Science is, what computer scientists do, and how both affect the world from the practical to the esoteric.
2. Freshmen and those who are thinking about pursuing the Computer Science majors/minors in either McCormick or Weinberg.
3. Computer Science students, for whom this course is required.
4. Computer Engineering students, who should become familiar with the breadth of the intellectual endeavor of Computer Science.
5. Weinberg students, for whom this course satisfies the area III distribution requirement.

What Is This Class about?

The primary goal of this course is to answer these simple questions:

- "What is Computer Science?"
- "What do computer scientists do?"
- "How does Computer Science interact with the rest of the world?"

The very ubiquity of the products of Computer Science has sadly not been coupled with a corresponding growth in the understanding of the intellectual content, structure, history, and aspirations of the field, or of what the people in the field actually think and do. Many people might answer the above three questions by saying that Computer Science is "programming" or "coding", that computer scientists are programmers or manage programming projects, and that the field interacts mostly by creating products. While the art of computer programming is important and can be joyful, Computer Science encompasses much more. Here are just a short selection of the kinds of questions and issues that Computer Science grapples with and that we will touch on in this class:

A. How can we make it possible for human beings to design and build ever more complex artifacts? Computer software is among the most complex creations of human beings and is at the bleeding edge of what we understand how to design and build. How do we manage its complexity?

B. How do we sustain the exponential growth of computing power, storage capacity, and communication capacity? How do we exploit it?

C. What are the mathematical and physical limits to computation and communication? What can't computers do? What is computation? What is communication?

D. What kinds of computational problems are there? How does the Universe limit how fast they can be solved?

E. What kinds of computers are there? Are some fundamentally more powerful than others?

F. What kinds of languages are there? Can we find one that is equally adept at getting things done on a computer and explaining how it's done to a person?

G. How can we translate from one language to another? How is this problem different for human languages and computer languages? Do answers for one have influence on the other?

H. How can computers and human beings best interact?

I. Is software creation an art, a science, or an engineering discipline?

J. Can we make a computer learn by itself instead of programming it?

K. Can we, and how can we, make a mind? How can we tell that we're successful?

L. What is intellectual property theft? What are the implications to society of the fact that a computer can process any information and making copies of information is free? What are the implications of the free (as in Freedom) software movement?

   http://en.wikipedia.org/wiki/Free_software_movement

M. What are the limits to computation-enabled security and privacy? What

implications do these fundamental limits have for politics and law?

N. Is computation based on physics or is physics based on computation?

O. Is the Universe a big computation?

Required Readings:

In addition to introducing questions and concepts, the readings below are also intended to present the personalities and history that are involved, and to provoke discussion and debate. Computer Science is a dynamic intellectual endeavor, and many core intellectual questions are still open. A separate reading list shows the sources from which this class draws.

You should purchase the following books. It's cheapest if you buy the paperback versions. Additional materials will be handed out in class or provided via the web. Note that we will read only selected portions of these books, but you are encouraged to dive deeper into these books and the other materials listed on the reading list.

A. David Harel, Computers Ltd: What They Really Can't Do, Oxford University Press, 2003.

B. Martin Davis, The Universal Computer: The Road from Leibniz to Turing, W.W. Norton and Company, 2000. This excellent book is out of print, but the NU Bookstore has a course-pack available.

C. Paul Graham, Hackers and Painters: Big Ideas from the Computer Age, O'Reilly Media, 2004.

D. Lawrence Lessig, Code: Version 2.0, Basic Books, 2006. This book is also available for free online at http://codev2.cc/ (Creative Commons license).

E. Richard Stallman, Free Software, Free Society: Selected Essays of Richard M. Stallman, Free Software Foundation, 2002. This book is also available for free online at http://www.gnu.org/ (GNU Public License).

F. Stephen Wolfram, A New Kind of Science, Wolfram Media, 2002. This book is also available for free online at http://www.wolframscience.com/.

G. Simon Singh, The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography, Anchor, 2000.

H. Ray Kurzweil, Are We Spiritual Machines? Ray Kurzweil vs. the Critics of

Strong AI, Discovery Institute, 2002. This book is also available for free online at http://www.kurzweilai.net.

I. Charles Petzold, Code: The Hidden Language of Computer Hardware and Software, Microsoft Press, 2000.

J. Eric Raymond, The Cathedral and the Bazaar: Musing on Linux and Open Source by an Accidental Revolutionary, O'Reilly, 2001. Also mostly available for free from http://www.catb.org/~esr/writings/cathedralbazaar/.

K. Scott Rosenberg, Dreaming in Code: Two Dozen Programmers, Three Years, 4732 Bugs, and One Quest for Transcendent Software, Crown,2007.

It's a lot more fun if students discuss and debate the readings! There is a Google discussion group that is accessible from the course web page, and students are also strongly encouraged to participate in the recitation/discussion sessions.

Essays:

During the course of the quarter, the students will pair up (i.e., work in teams of two people) to write two essays. The first essay will be on a personality in the history or present of Computer Science. The second essay will be on a computer science topic, such as one of the questions given above. Each essay will be 10 pages long, single-spaced.

The intent of these essays is threefold. (1) The students will get a view of Computer Science from two distinct cross-cutting points of view. (2) The students will have the opportunity to dive more deeply into particular aspects of the field. (3) The students will have the opportunity to do research and write as a team.

Exams:

There will be a midterm and a final. The final will not be cumulative.

Grading:

- 20% -- class participation informed by reading, including participation in the recitation/discussion sessions and the online discussion group.
- 40% -- two essays, 20% for each essay.
- 20% -- midterm.

- 20% -- final.

Lecture Schedule:

Week 1: Introduction

Dates: 3/29/2011, 3/31/2011.

Instructor: Ming-Yang Kao.

Outline:
- The long arc from the Antikythera mechanism to today.
- Historical roots of modern computer science in mathematics, electrical engineering, and psychology.
- A look at the big questions.
- Structure of the field (the core areas).
- What computer scientists do (how people are employed).

Readings:
1. Jeannette Wing, "Computational Thinking", Communications of the ACM, Volume 49, Number 1, January 2006, pp. 33-35.

   http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf

2. Bernard Chazelle, "Could Your iPod be Holding the Greatest Mystery in Modern Science?", Math Horizons, April, 2006.

   http://www.cs.princeton.edu/~chazelle/pubs/ipod.pdf

3. Vannevar Bush, As We May Think, The Atlantic Monthly, July, 1945.

   http://www.theatlantic.com/doc/194507/bush

4. Wikipedia, "The Antikythera Mechanism".

   http://en.wikipedia.org/wiki/Antikythera_mechanism

5. Chapter 2 from Paul Graham, Hackers and Painters, O'Reilly, 2004.

6. CNNMoney.com, Best Jobs in America, 2010.

   http://money.cnn.com/magazines/moneymag/bestjobs/2010/

7. Preamble from David Harel, Computers Limited: What They Really Can't

Do, Oxford University Press, 2000.

Week 2: Fundamental Theories – the core ideas and how they are evolving.

Dates: 4/5/2011, 4/7/2011.

Instructor: Ming-Yang Kao.

Outline:
- Discrete mathematics and logic (these areas of Mathematics, not calculus, form the mathematical basis of Computer Science).
- Theory of computation and its surprising facts.
- Theory of communication and its ubiquity.
- Classical computers.
- Simple "laws": Moore, Gilder, Murphy.

Readings:
1. Chapters 1-2, from David Harel, Computers Limited: What They Really Can't Do, Oxford University Press, 2000.

2. Chapters 1, 6, 7 from Martin Davis, The Universal Computer: The Road from Leibniz to Turing, W.W. Norton and Company, 2000.

Week 3: Algorithms, Tractability, and Intractability.

Dates: 4/12/2011, 4/14/2011.

Instructor: Jason Hartline.

Outline:
- Tractability of problems.
- Algorithms.
- Data structure concepts.
- Simulation and its impact.

Videos:
1. "Arthur Benjamin does Mathemagic", TED talk video.

   http://www.ted.com/talks/arthur_benjamin_does_mathemagic.html

Readings:
1. Scott Aaronson, "The Prime Facts: From Euclid to AKS", manuscript, 2003.

http://www.scottaaronson.com/writings/prime.pdf

2. Ian Stewart, "Million Dollar Minesweeper", Scientific American, October 2000. (Need to copy from library.)

3. Lance Fortnow, "The Status of the P Versus NP Problem", Communications of the ACM, 2009.

   http://cacm.acm.org/magazines/2009/9/38904-the-status-of-the-p-versus-np-problem/fulltext

4. Chapters 3-6, from David Harel, Computers Limited: What They Really Can't Do, Oxford University Press, 2000. (Optional.)

Week 4: Networks, Markets, and Crowds -- reasoning about a highly connected world.

Dates: 4/19/2011, 4/21/2011.

Instructor: Jason Hartline.

Outline:
- Structure of the web.
- Sponsored search.
- Small world phenomena.
- Social contagion.
- Information cascades.

Readings:
1. Duncan Watts, "Is Justing Timberlake a Product of Cumulative Advantage?", The New York Times, April 15, 2007.

   http://www.nytimes.com/2007/04/15/magazine/15wwlnidealab.t.html

2. Jon Kleinberg, "The Convergence of Social and Technological Networks", Communications of the ACM, November, 2008.

   http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.147.1260&rep=rep1&type=pdf

3. Steven Levy, "Secret of Googlenomics: Data-Fueled Recipe Brews Profitability", Wired Magazine, May 22, 2009.

http://www.wired.com/culture/culturereviews/magazine/17-06/nep_googlenomics?currentPage=all

4. Chris Anderson, "The Long Tail", October 2004.

   http://www.wired.com/wired/archive/12.10/tail.html

5. Eric Lofgren and Nina Fefferman, "The untapped potential of virtual game worlds to shed light on real world epidemics", The Lancet Infectious Diseases, September 2007.

   http://breakglass.files.wordpress.com/2007/10/lofgren_fefferman_lancet.pdf

Week 5: Security and a Bit of Law and Politics.

Dates: 4/26/2010, 4/28/2011.

Instructor: Goce Trajcevski.

Outline:
1. Cryptography.
2. Cracking the uncrackable.
3. Secure protocols and their magic.
4. Intrusion detection.

Readings:
1. Chapters 4,6,7 from Simon Singh, The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography, Anchor, 2000.

2. Chapter 6 from David Harel, Computer Limited: What They Really Can't Do, Oxford University Press, 2000.

3. Tadayoshi Kohno, Adam Stubblefield, Aviel Rubin, and Dan Wallach, Analysis of an Electronic Voting Machine, IEEE Symposium on Security and Privacy, 2004. (Available online. It is sufficient to skim this.)

4. Feldman, Halderman, and Felten, Security Analysis of the Dielbold AccuVote-TR Voting Machine. (It is sufficient to skim this.)

   http://itpolicy.princeton.edu/voting

Week 6: Computer Systems – raising the abstraction.

Dates: 5/3/2010, 5/5/2011.

Instructor: Goce Trajcevski.

Outline:
- Architecture.
- Operating systems.
- Databases.
- Networking.
- Middleware.

Videos:
1. ARPANET Video (origins of the Internet). (Watch on your own.)

   http://www.newmediamedicine.com/blog/2006/08/16/ arpanew-video/

2. Possible demos of assembler, Linux Trace Toolkit and Ethereal. (Watch in class.)

Readings:
1. Chapters 2, 3, 10, 11, 18, 22 from Charles Petzold, Code: The Hidden Language of Computer Hardware and Software, Microsoft Press, 2000.

2. Wikipedia article on operating systems.

   http://en.wikipedia.org/wiki/Operating_system

3. Wikipedia article on middleware.

   http://en.wikipedia.org/wiki/Middleware

4. Wikipedia article on database management systems. (Skim.)

   http://en.wikipedia.org/wiki/Database_management_system

5. Wikipedia article on Internet. (Skim.)

   http://en.wikipedia.org/wiki/Internet

Week 7: Artificial Intelligence – making minds and solving problems we don't know how to solve.

Dates: 5/10/2011, 5/12/2011.

Instructor: Bryan Pardo.

Outline:
- Turing test.
- Are people fancy computers?
- Logic-based approaches.
- Heuristic search.
- Vision and image understanding.
- Robotics.
- Machine learning – statistics as if computers existed.

Videos:
1. Koza Genetic Programming Example. (Watch in class.)

Readings:
1. Chapter 7 from David Harel, Computer Limited: What They Really Can't Do, Oxford University Press, 2000.

2. Chapter 9 from Martin Davis, The Universal Computer: The Road from Leibniz to Turing, W.W. Norton and Company, 2000.

3. John McCarthy, "What is Artificial Intelligence?".

   http://www.formal.stanford.edu/jmc/whatisai/

4. Chapters 2+5 (John Searle + Rebuttal) and Chapters 5+9 (Thomas Ray + Rebuttal): Selections from Ray Kurzweil, Are We Spiritual Machines? Ray Kurzweil vs. the Critics of Strong AI, Discovery Institute, 2002. Also available for free on Kurzweil's site http://www.kurzweilai.net/.

Week 8: More Systems, Languages, and Software Engineering.

Dates: 5/17/2011, 5/19/2011.

Instructor: Robby Findler.

Outline:
- Compilers.
- Software Engineering.
- Mathematical thinking about grammar and semantics.
- Computation as understanding languages.
- Formal language design.
- Programs and proofs.
- Language design and implementation in practice and its convergence

with theory.

Reading :

1. Chapter 24 from Charles Petzold, Code: The Hidden Language of Computer Hardware and Software, Microsoft Press, 2000.

2. History of Programming Languages (chart and content). (Skim.)

   http://www.levenez.com/lang

3. Wikipedia articles on programming languages. (Skim.)

   http://en.wikipedia.org/wiki/Programming_language
   http://en.wikipedia.org/wiki/History_of_programming_languages

4. Chapter 1, 2, 3, and 10 from Scott Rosenberg, Dreaming in Code: Two.

5. Dozen Programmers, Three Years, 4732 Bugs, and One Quest for Transcendent Software, Crown, 2007.

6. "Programming Languages Explained", Chapter 10 of Paul Graham, Hackers and Painters, O'Reilly, 2004.

7. Wikipedia article on compilers.

   http://en.wikipedia.org/wiki/Compiler

Week 9: Human Computer Interaction and Graphics.

Dates: 5/24/2011, 5/26/2011.

Instructor: Jack Tumblin.

Outline:

- Applied geometry on a computer.
- Issues in interface design.
- Psychology of using computers.
- Where your PC came from, and where it might go…

Videos:

1. SIGGRAPH Video Review, Issue 137, The Story of Computer Graphics, Video, 1999. (We may watch in class.)

2. Doug Engelbart Demo. (Watch on your own – OK to skim.)

http://sloan.stanford.edu/MouseSite/1968Demo.html

3. Alan Kay Video "Doing With Images Makes Symbols". (Watch on your own – great history.)

   http://video.google.com/videoplay?docid=-533537336174204822

Readings:
1. Chapter 6 from Scott Rosenberg, Dreaming in Code: Two Dozen Programmers, Three Years, 4732 Bugs, and One Quest for Transcendent Software, Crown, 2007.

2. "Taste For Makers", Chapter 9 of Paul Graham, Hackers and Painters, O'Reilly, 2004.

Week 10: The Bleeding Edge and Crazy Ideas and/or Slack Time.

Dates: 5/31/2011, 6/2/2011.

Instructors: Jack Tumblin and Ming-Yang Kao.

Outline:
- Quantum computing.
- Digital Physics.
- Biological computing with DNA.
- Smart Dust.
- More.

Videos:
1. Play Conway's Game of Life.

   http://www.bitstorm.org/gameoflife/

Readings:
1. Chapters 1 and 2 from Stephen Wolfram, A New Kind of Science, Wolfram Media, 2002. This book is also available for free online at http://www.wolframscience.com/.

2. Rule 110 (Wikipedia Article).

   http://en.wikipedia.org/wiki/Rule_110_cellular_automaton

3. Conway's Game of Life.

http://en.wikipedia.org/wiki/Conway's_Game_of_Life

4. Nick Bostrom, Are You Living in a Computer Simulation?, Philosophical Quarterly, Volume 53, Number 211, 2003. Available online at http://www.simulation-argument.com.

5. Chapter 8 from Simon Singh, The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography, Anchor, 2000.

6. Slack or more (to be determined depending on available time).