Welcome!

- CS442, officially “Topics in Computer Science”, but aspiring to be “Great Insights in Computer Science”.
- Need permission number to register. Mail mlittman@cs.rutgers.edu.
- Should be 3 credits---need to fix it.
- Can’t be taken for CS major credit.
About Me

- 2000-2002: (Back) to NJ, AT&T Labs Research.
- 2002-: Rutgers: heading “Rutgers Laboratory for Real-Life Reinforcement Learning” (RL3).

My Goals

- Cool facts, cool ideas. Ideas are “how to” facts. Shoot for two per lecture.
- People (my parents, say), don’t understand how I have a PhD in CS and can’t help them when their Windows box crashes.
- If it’s not about XP, what else is there?
Introduction to CS101s

- Target audience: Undergrads, as a first (possibly only) computer science course.
- Seminar in Computers and Society: What impact have computers had on the world?
- Introduction to Computer Science: How do I learn to create my own software?
- Introduction to Computers and Their Application: What do I need to know about computing technology?

Course Goals: Questions

- What is Computer Science?
- Why is it interesting?
- How is it different from software engineering?
- What are the insights that make computer science its own academic discipline?
Textbook

- Not too daunting or detailed.
- Inspiring and informative.
- *Pattern on the Stone, the simple ideas that make computers work*, Danny Hillis, Basic Books, 1998.
- Enjoyable to read; not really a textbook at all.
- I will add meat to the wonderful skeleton he creates.

Syllabus: Top-level View

Organizing the material into three major sections:

I. How does a computer work?
II. What do we know about computation?
III. What cool things are computers doing?
I. How Computers Work

• Sequence of easy-to-understand layers.
• Top: high-level programming languages express ideas in a computer-friendly form.
• Bottom: bits and logic gates, computer’s work done by physical components.
• We’ll go bottom up, conceptually creating a working computer in the process.

II. Computation

• Computers compute. But, what is computation, and how powerful is it?
• How do computer scientists see the world in terms of computational problems?
• What’s an algorithm, why might one be preferred to another if they solve the same problem.
• What problems can be solved and how fast?
III. Useful Applications

We’ll survey how computing power is being harnessed.

- computational biology
- machine learning, pattern recognition
- computer graphics
- databases
- AI search
- security/cryptography
- networking
- robotics
- operating systems
- ...feedback?

They Are Everywhere!

- 1977: “There is no reason for any individual to have a computer in his home.”
Previously Unthinkable

One-Word Summary

- If I had to summarize the intellectual contribution of computer science in one word, it would be “reduction”.

- Computer scientists solve problems by reducing them to simpler problems.

- We’ll see this same idea played out over and over again in different settings...
Levels of Complexity

- **Networking** (OSI Layers): application, presentation, session, transport, network, data link, physical.
- **Vision** (Marr): computational, algorithmic, implementation.
- **Computing**: application, high-level language, machine language, logic blocks, logic gates, physical.
- **Storage hierarchy**: offline-storage, hard disk, RAM, cache, registers.

Today’s Idea

- I will start with the textbook next time, introducing bits and some simple gates.
- Please read the Preface and Chapter 1, Section 1.
- But, I want to give you something to chew on to get those gears turning...
Charles Babbage: Facts

- Charles Babbage lived in England in the 1800s. A polymath, he solved problems from Astronomy to Zoology.
- **Invented**: cowcatcher, flat-rate postage, Operations Research, standard RR gauge.
- Held Newton’s chair at Cambridge, but spent his last years railing against organ grinders.
- His design for the *analytical engine* presaged much of the design of modern digital computers.

The Difference Engine

- Originally a “computer” was a job description of a person who created numerical tables. Useful, if tedious.
- Babbage had a scheme to automate table creation using “finite differences”.
- 25,000 parts, 15 tons, 8 feet high, never built!
- His redesigned Difference Engine No. 2 was built in 1990 and actually worked!
Let’s play a game:

- 0,1,2,3,4,___?
- 3,5,7,9,11,___?
- 1,4,9,16,25,___?
- 1,3,6,10,15,___?

---

Instructions:
1. Start with 2.
2. To get the next one, add 2 to the previous one.

2 → 4 → 6 → 8 → 10 → 12
Generating Squares

Instructions:
1. Start with 1.
2. Start the increment with 3.
3. To get the next one, add the increment.
4. To get the next increment, add 2 to the previous one.

The Squares

1 → 4 → 9 → 16 → 25 → 36
3 → 5 → 7 → 9 → 11
2 2 2 2

Naming The Sequences

The Evens

2 → 4 → 6 → 8 → 10 → 12
2 2 2 2 2

The Squares

1 → 4 → 9 → 16 → 25 → 36
3 → 5 → 7 → 9 → 11
2 2 2 2
Programming the “DE”

- So, to produce the Evens, we enter “2 2” into the Difference Engine and turn the crank.
- The Squares are “1 3 2”.
- The Triangle numbers are “1 2 1”.
- Try:
  - Odds?
- Count by 5s?
- Cubes?
  - What do these do:
    - “4”? 
    - “10 -1”? 
    - “100 -19 2”? 
    - “0 1 2 1”? 

CS in a Microcosm?

- Get a feel for programming: need to figure out how to say what you want to say in a way the machine understands.
- Simple operation (repeated addition) used to build up more complex objects.
More Advanced Stuff

- Each difference sequence corresponds to a polynomial and vice versa. Why? What is the relationship between the length of the difference sequence and the degree of the polynomial?

- You can start a sequence at any point by starting with the corresponding column. How can a sequence be moved backward automatically?

- Can match any finite-length sequence. How long might the difference sequence need to be?

- Download “Python” and play with it yourself! http://www.cs.rutgers.edu/~mlittman/courses/cs442-06/python/differences.py

Next Time

- Think about the Difference Engine.

- Read Hillis: Preface and Chapter 1, Section 1.