Welcome!

- CS105, taught last semester as “Topics in Computer Science”, but is now officially “Great Insights in Computer Science”.
- Several innovations this semester:
  - Using Rutgers’ online course support
  - iClickers for in-class interaction
  - Programs for out-of-class interaction
Survey

- Do you have a clicker?
  A. no
  B. yes

About Me

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

- Cool facts, cool ideas. Ideas are “how to” facts. Shoot for one or 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?
  - That’s what I want to tell you...

**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 fun/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.
Clickers

- Available at several bookstores.
- Can keep them (most popular clicker at Rutgers) or sell them back.
- We’ll use them for attendance, reading comprehension quizzes, straw polls.
- Hope to eventually add interactive demos, but software not yet ready.

[iclicker]
http://www.iclicker.com/

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.

- machine learning, pattern recognition
- computer graphics
- AI search
- language games
- robotics
- genetic algorithms
- data compression

They Are Everywhere!

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

Survey

• How many computers are with you right now? Laptop, handheld game, cell phone, GPS device, PDA, mp3 player, ...

A. 0
B. 1
C. 2
D. 3
E. 4 or more

Previously Unthinkable

![Google Logo](image1.png)

![Lion](image2.png)

![GPS Device](image3.png)

![Robotic Hand](image4.png)
Life After Death?

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.
- **Computing:** application, high-level language, machine language, logic blocks, logic gates, physical.
- **Vision (Marr):** computational, algorithmic, implementation.
- **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...
Your Choice

- Babbage’s Difference Engine
  - Simple computation for sequences
- How do UPC barcodes work?
  - Hierarchy of codes
  A. Difference Engine
  B. UPC barcodes

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!

Difference Engine: Ideas

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,___?
Generating the Evens

2 → 4 → 6 → 8 → 10 → 12

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

Generating Squares

1 → 4 → 9 → 16 → 25 → 36

3 → 5 → 7 → 9 → 11

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.
Naming The Sequences

2 → 4 → 6 → 8 → 10 → 12

The Evens

Sequence: 1, 4, 9, 16, 25, 36, ...

1 → 4 → 9 → 16 → 25 → 36

Program: 3 → 5 → 7 → 9 → 11

1, 3, 2

The Squares

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?
• Download “Python” and play with it yourself!
  [link](http://www.cs.rutgers.edu/~mlittman/courses/cs442-06/python/differences.py)
• Can match any finite-length sequence. How long might the difference sequence need to be?
Bar Codes

- Many different styles of barcodes.
- Most common is UPC-A, in use in most North American retail stores.
- I will describe the UPC encoding.
- Many of the same ideas apply to other codes: checks, photostamps, IR remotes.

Universal Product Codes

- First scanned product, Wrigley’s gum (1974).
- Method of identifying products at point of sale by 11-digit numbers.
- Method of encoding digit sequences so they can be read quickly and easily by machine.
**Reduction Idea**

- Each level uses an **encoding** to translate to the next level.
  - Patterns of ink.
  - Sequence of 95 zeros and ones.
  - Sequence of 12 digits.
  - Sequence of 11 digits.
  - Name of a retail product.

**Product Name**

- Ponds Dry Skin Cream
  - 3.9 oz (110g)
  - Unilever Home and Personal Care USA
- Name Badge Labels (Size 2 3/16" x 3 3/8")
  - 100 Labels
  - Avery Dennison/Avery Division
11-Digit Number

- Digit = \{0,1,2,3,4,5,6,7,8,9\}
- Sequence of 11 digits
- How many different items can they encode?
  A. 10,000,000,000
  B. 100,000,000,000
  C. 9,999,999,999
  D. 19,999,999,999

Encode Name By 11 Digits

- First 6 digits: Manufacturer
  - First digit, product category:
    - 0, 1, 6, 7, 8, or 9: most products
    - 2: store’s use, for variable-weight items
    - 3: drugs by National Drug Code number
- Last 5 digits: Manufacturer-assigned ID
Examples

- Labels: 0-72782-051440
  - 0=general product
  - 72782=Avery
  - 051440=Avery’s code for this product

- Ponds: 3-05210-04300
  - 3=drug code
  - 05210=Unilever
  - 04300=National Drug Code for this product

12-Digit Number

- The UPC folks decided to include another digit for error checking. Example:
  - 01660000070  Roses Lime Juice (12 oz)
  - 04660000070  Eckrich Franks, Jumbo (16 oz)
  - 05660000070  Reese PB/Choc Egg (34 g)
  - 08660000070  Bumble Bee Salmon (14.75 OZ)

- Misread digit #2 and you turn sweet to sour.
Check Digit

1. Add the digits in the odd-numbered positions (first, third, fifth, etc.) together and multiply by three.

2. Add the digits in the even-numbered positions (second, fourth, sixth, etc.) to the result.

3. Subtract the result from the next-higher multiple of ten. The result is the check digit.

Code and Example

```python
def checkDigit(d1,d2,d3,d4,d5,d6,d7,d8,d9,d10,d11):
    step1 = (d1+d3+d5+d7+d9+d11)*3
    step2 = step1+d2+d4+d6+d8+d10
    step3 = (-step2)%10
    return step3
```

<table>
<thead>
<tr>
<th>Code</th>
<th>Check Digit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lime juice:</td>
<td>01660000070</td>
</tr>
<tr>
<td>Franks:</td>
<td>04660000070</td>
</tr>
<tr>
<td>Choc Egg:</td>
<td>05660000070</td>
</tr>
<tr>
<td>Salmon:</td>
<td>08660000070</td>
</tr>
</tbody>
</table>

odd-digit sum: 0+6+0+0+0+0=6
even-digit sum: 1+6+0+0+7=14
odd*3+even = 6*3+14=32
subtract from mult of 10=40-32=8

all are two digits different now
Bits

- We’ve gone from a product name to an 11-digit number to a 12-digit number. Next: bits.
- abcdefghijkl→101abcdef01010
  ghijkl101

Digits encoded as 7-bit patterns, chosen to be:
- as different as possible
- start with 0, end with 1
- switch from 0 to 1 twice
- no more than 4 of the same bit in a row

Last 6 digits have 0s and 1s reversed!

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

How Many Bits?

- From our 12-digit number, how many bits (zeros and ones) long is the code now?

A. 84
B. 95
C. 100
D. 23
E. 12
Finally, Ink!

- Given the long pattern of bits, we write a 1 as a bar and a zero as a space.
- Two 1s in a row become a double wide bar.
- Two 0s in a row become a double wide space.
- Never have more than 4 in a row.
- Starts and ends with bars.

Example

- Barcode for skin cream:
  - 3-05210-04300-8 (8 is the check digit)
    - start: 101; 3: 011101
    - 05210: 0001101-011001-0010011-0011001-0001101
    - middle: 01010
    - 04300: 1110010-1011100-1000010-1110010-1110010 (rev)
    - 8: 1001000 (rev); end: 101
  - The digits underneath are for our benefit.
Summary

- Product name turned to 11-digit code
- 11-digit code extended to 12 digits by adding a check digit
- 12 digits become a 95-bit sequence
- 95 bits are drawn in ink 1=black, 0=white

Reverse the process to get the product!
Next Time

- Get iClicker
- Read Hillis: Preface and Chapter 1, up to and including “Boolean Logic”.
- We’ll have a brief iClicker quiz on the material.