1: Introduction

- Computers are everywhere creating new capabilities.
- Computer science is the study of "reduction": making complex out of simple.
- **Skill**: Reading barcodes.
2: Bits and Switches

- Bits: 0/1, simple but can represent anything
- **Skill**: Evaluate an expression using and/or/not.

3: Gates

- Bits can be made out of many things.
- Universal logic gates can simulate and/or/not.
- Gates can be defined in terms of simpler gates (reduction).
- Logical expressions, truth tables
4: Binary

- Some simple logic gates.
- **Skill**: Recognize some simple functions of logic gates.
- Representing numbers in binary.
- **Skill**: convert between binary and decimal (base 10).
- **Skill**: adding numbers in binary.

5: Machine Language

- Fundamental circuits:
  - CPU interprets machine instructions.
  - Memory holds data and programs and is addressed in binary.
  - Arithmetic circuits perform mathematical calculations.
  - Each clock cycle corresponds to a small instruction being executed.
- Machine-language program is made from bytes.
- Everything in the computer is made from these small instructions.
6: Subroutines

- **The hierarchy**: application programs -> software libraries -> high-level languages -> machine language -> logic blocks -> basic logic gates -> physical bits (transistors).
- Subroutines can package up repeated operations (like song choruses).
- **Skill**: Write a song as a series of subroutines.
- Parameters let you use the same subroutine for related tasks.
- The subroutine stack keeps track of the program’s place.

7: Algorithms

- Sock matching.
- Different approaches to a problem can be faster than others!
- **Skills**:
  - Convert a truth table to a logical expression.
  - Convert and add binary numbers.
  - Count the number of gates in a (multilevel) logic block.
8: Growth Rates

- Computer scientists analyze algorithms by their running time as a function of the size of the input.
- Can sing generalizations of songs with $n$ verses.
- Number of syllables is a pain to figure out as a function of $n$.
- Big-O notation (asymptotic growth rate) simplifies the process.

8: more

- Major classes of song growth rates:
  - constant-size verses: linear $O(n)$
  - verses grow because includes a number, which grows: linear-logarithmic ($O(n \lg n)$).
  - verses grow by a constant size: quadratic ($O(n^2)$).
  - verses grow by a constant size and include larger numbers: quadratic-logarithmic ($O(n^2 \lg n)$).
- **Skill**: Recognize growth rate of songs.
8: more

- **Skill**: Distinguish proper (growth rate) and improper ("big") uses of the word "exponential".
- The growth-rate classes are also very useful for analyzing algorithms like the sock sorter.

9: Compilers

- People write programs in languages that cannot be directly run on the computer.
- Compilers translate the programs into machine language that the computer can run.
- Parser captures the structure of the program, allows for optimizations and code generation.
- Interpreters don't translate the program, but just simulate it themselves.
- **Skill**: Interpreting expression trees.
10: Graph Algorithms

- Google builds a map of the web by visiting web pages it knows about, then looking on those pages for the address of other pages.
- This operation is called "graph search". Used to solve mazes, also.
- Graph defined by nodes, links. Other terms: source, sink, path, cycle, connected components, tour, directed, undirected, tree. Node \(x\) is "reachable" from \(y\) if there's a path from \(y\) to \(x\).

11: Sorting Algorithms

- Sorting speeds up the search for information in a list of \(n\).
- Selection Sort: Repeatedly find, remove the smallest. \(O(n^2)\).
- Binary search: Ask a question that splits the remaining set of options in half. \(O(\lg n)\).
- Quick Sort: Split into big/small elements relative to pivot. \(O(n \lg n)\).
11: more

- Skills:
  - Is the use of “exponential” right?
  - Given a generalized $n$-verse song, what is its syllable growth rate?
  - How many comparisons does a sorting algorithm make?

14: Computability

- If an assumption leads to self contradiction, the assumption is broken.
- Barber paradox, Godel’s theorem, Kantor’s diagonalization
- Assumption that we can detect whether a program loops forever leads to a program that loops forever and doesn’t loop forever.
- The “halting problem” is not solvable.
15: Randomness

• Some problems solved simply and efficiently using random numbers (survey, quicksort).
• Seemingly random numbers can be generated using complex numerical mixing-up functions.
• Random bits useful for sending secret messages.
• **Skill:** Will a given subroutine halt on all inputs?

17: Parallel Computation

• Moore’s Law: Roughly, computers double in speed every year and a half.
• Some problems can be sped up by letting more than one computer work on them at a time.
• Some can’t!
• Some others can, but only if you’re clever.
18: Compression

- The number of bits that it takes to represent a message varies with the encoding.
- Finding a shorter encoding is “compression”.
- Huffman encoding uses few bits for common characters.
- **Skill**: Given a string, build a Huffman code for it, encode and decode strings.

19: Neural Networks

- Classifiers map feature vectors to yes/no.
- Classifiers can be created automatically (learned) by analyzing a training set of examples.
- Perceptrons use a linear threshold function and can solve linearly separable problems.
- Decision trees can be constructed from examples and applied to new instances (demo).
21: Genetic Algorithms

- Search algorithm using a population to find good solutions.
- Uses populations of individuals (often bitstrings) that mate if their fitness is sufficiently high to produce new generations of improved individuals.
- **Skill**: Mate two bitstrings and find the result.
- Nice summary of earlier concepts!

22. Robotics

- “while True” is useful when constructing programs that must continue to do the right thing.
- Reinforcement learning defines task by specifying the reward function / goal and letting the program change its behavior to make things work.
- Robots are just computers.
Next Time

• Final! Wednesday, Dec. 20, 12-3pm
• This room.
• Open book.

With sincere apologies to Simon and Garfunkel...
A = True
B = True

def sorter1():
    x = getSock()
    y = getSock()
    if not match(x,y):
        replaceSock(x)
        replaceSock(y)

lightOn = True
### Level Examples Alternatives

<table>
<thead>
<tr>
<th>Level</th>
<th>Examples</th>
<th>Alternatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>software libraries</td>
<td>graphics, animation, robotics</td>
<td>networking, security, mathematics</td>
</tr>
<tr>
<td>high-level language</td>
<td>Python</td>
<td>C, Java, C++, Logo, LISP, Fortran, ML</td>
</tr>
<tr>
<td>machine language</td>
<td>ML&lt;sup&gt;3&lt;/sup&gt;</td>
<td>x86, CARDIAC, Z80</td>
</tr>
<tr>
<td>logic gates</td>
<td>equal, ifthenelse, add</td>
<td>memlookup, memwrite</td>
</tr>
<tr>
<td>basic logic gates</td>
<td>and, or, not</td>
<td>nor, nand, xor</td>
</tr>
<tr>
<td>physical bits</td>
<td>0,1 via high/low voltage</td>
<td>water pressure, kinetic energy</td>
</tr>
</tbody>
</table>

---

#### Programming an Adder

**Circuit level:**

**Instruction level:**

- \( H = A \text{ xor } B = (A \text{ and } \neg B) \text{ or } (\neg A \text{ and } B) \)
- \( C_1 = (A \text{ and } B) \text{ or } (C \text{ and } H) \)
- \( S = H \text{ xor } C = (H \text{ and } \neg C) \text{ or } (\neg H \text{ and } C) \)
Karl Sims’ Accomplishments

def OldMacdonald():
        print 'Old Macdonald had a farm, E-I-E-I-O'
        print 'And on his farm he had a cow, ... beer on the wall.
91 bottles of beer on the wall.
91 bottles of beer.
If one of those bottles should happen to fall.

96 bottles of beer on the wall.
96 bottles of beer.
If one of those bottles should happen to fall.
95 bottles of beer on the wall.
95 bottles of beer.
If one of those bottles should happen to fall.
94 bottles of beer on the wall.
94 bottles of beer.
If one of those bottles should happen to fall.
93 bottles of beer on the wall.
93 bottles of beer.
If one of those bottles should happen to fall.
92 bottles of beer on the wall.
92 bottles of beer.
If one of those bottles should happen to fall.
91 bottles of beer on the wall.
91 bottles of beer.
91 bottles of beer.
Four score and seven years ago our fathers brought forth on this continent, a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.

Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battle-field of that war. We have come to dedicate a portion of that field, as a final resting place for those who here gave their lives that that nation might live. It is altogether fitting and proper that we should do this.

But, in a larger sense, we can not dedicate -- we can not consecrate -- we can not hallow -- this ground. The brave men, living and dead, who struggled here, have consecrated it, far above our poor power to add or detract. The world will little note, nor long remember what we say here, but it can never forget what they did here. It is for us the living, rather, to be dedicated here to the unfinished work which they who fought here have thus far so nobly advanced. It is rather for us to be here dedicated to the great task remaining before us -- that from these honored dead we take increased devotion to that cause for which they gave the last full measure of devotion -- that we here highly resolve that these dead shall not have died in vain -- that this nation, under God, shall have a new birth of freedom -- and that government of the people, by the people, for the people, shall not perish from the earth.
But There Are Lots of Them

- iPod: 60Gb. 1Gb = one billion bytes, each of which is 8 bits.
- Format uses 128 Kbps (kilobits or 1000 bits per second of sound).
- So, 62,500 minutes of sound or 15,000 songs at 4 minutes per song.

- Screen: 320x240 pixels, each of which stores 1 byte each of R,G,B intensity (24 bits). That’s 76,800 pixels and 1.8M bits.
- At 30 frames per sec., that’s 55.3 million bits per second or 144 min. of (quiet) video.