For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.
Reminders

Final exam
I will send out a list of what I think we covered this term
2 hours (not 3)
Comprehensive, but weighted towards end
Reviews scheduled

Underground Guide

Today’s lecture
What do computer scientists do?
  What does this computer scientist do
Overview of term
What Do Computer Scientists Do?

They think computationally

Computational thinking will be a fundamental skill used by everyone in the world by the middle of the 21st Century.

Just like the three r’s: reading, riting, and rithmetic.

Ubiquitous computing and computers will enable the spread of computational thinking.
Computational Thinking: the Process

Identify or invent useful abstractions

Formulate solution to a problem as a computational experiment

Design and construct a sufficiently efficient implementation of experiment

Validate experimental setup

Run experiment

Evaluate results of experiment

Repeat as needed
The Two A’s of Computational Thinking

Abstraction
Choosing the right abstractions
Operating in terms of multiple layers of abstraction simultaneously
Defining the relationships the between layers

Automation
Think in terms of mechanizing our abstractions
Mechanization is possible
Because we have precise and exacting notations and models
There is some “machine” below (human or computer, virtual or physical)

1Ideas adapted from Jeannette Wing
Examples of Computational Thinking

How difficult is this problem and how best can I solve it?

Theoretical computer science gives precise meaning to these and related questions and their answers

Thinking recursively

Reformulating a seemingly difficult problem into one which we know how to solve.

Reduction, embedding, transformation, simulation
Examples of Computational Thinking

Choosing an appropriate representation or modeling the relevant aspects of a problem to make it tractable

Prevention, detection, and recovery from worst-case scenarios through redundancy, damage containment, and error correction

Using the difficulty of solving hard problems to foil would be evil doers
What One Group Does, My Research Group

Goals
Help people live longer and better quality lives
In collaboration with clinicians
Have fun pushing the frontiers of
Computer Science
Electrical Engineering
Medicine

Technical areas
Machine learning, clustering, data mining
Algorithm design
Signal processing
Software systems
Specific Research Activities

Extracting clinically useful information from electrical signals
   Heart, brain, and connected anatomy
   Signal processing, algorithms, clustering, machine learning, …

Predicting adverse cardiac events
   Zeeshan Syed, Phil Sung, Jenna Wiens, Eugene Shih
   Collin Stultz, Ben Scirica

Detecting and responding to epileptic seizures
   Ali Shoeb, Al Kharbouch, Naveen Verma
   Steve Schachter, Trudy Pang, Syd Cash
Technical Skills Utilized

**Machine learning, clustering, data mining**
- Exploiting patient specificity
- Deriving new medical knowledge from large data sets

**Algorithm design**
- Computing novel functions
- Making things fast enough to actually use

**Signal processing**
- Extracting physiological relevant features from noisy signals

**Software systems**
- Reliability and predictability matters a lot
  - We have used a closed loop neuro-stimulator on humans

December 9, 2008 ©John Guttag
Example 1. Treating Epilepsy

Prevalence of ~1%; all ages
All countries

Characterized by recurrent seizures
Generated by abnormal electrical activity in brain

Seizures occur unpredictably
Often lead to injury and prolonged impairment

Multiplicity of manifestations

Acquired
Head Injury
Intracranial Hemorrhage
Infection
Stroke

Inherited
Ion Channelopathy
Defective Neural Organization
Seizure Onset Seems Unpredictable

**May result in injury**
Fractures, intracranial hematomas, burns, etc.

**May result in death**
Mortality rate 2-3 times that of general population
Accidents, aspiration, drowning, etc.
SUDEP (annual risk estimated to be 1 per 100 for patients with symptomatic seizures)
Early Detection of Seizure Onset

Two onset times
  Electrographic
  Clinical

Detecting electrographic onset
  Use scalp EEG

Therapeutic value
  Provide warning
  Summon help
  Fast acting drugs
  Neural stimulation
EEG varies greatly across patients
   Epileptics have abnormal baselines
   Generic detectors have not worked particularly well
Pretty consistent patterns for an individual
Use patient-specific detectors
Use machine learning to build patient-specific seizure onset detector. Highly successful retrospective studies
Turn on neural stimulator at start of seizure. Study in progress at BIDMC
Example 2: Predicting Death

Joint work with Collin Stultz and Zeeshan Syed

**Acute coronary syndrome (ACS) common:** ~1.25M/year in U.S.
15% - 20% of these people will suffer cardiac-related death within 4 years

**Stratifying risk key to choosing treatments**
- Who gets a defibrillator?
- Who should be treated aggressively with statins

**We think that we have a new and better way of doing this**
- Morphological variability (MV)
- Tested on ~8000*2*24*60*70 heart beats
Identifying High Risk Cases Vitally Important

Useful to find patients who should be:
  Monitored more closely
  Treated more aggressively

E.g., implanted defibrillators

Too many: Potentially risky, always expensive (~$50k)
  90% of recipients received < 0 medical benefit
Too few: 100’s of deaths/day potentially avoidable
Clinical characteristics
   E.g., gender or high blood pressure

Biomarkers
   E.g., cholesterol levels

Echocardiography
   Ultrasound to measure, e.g., left ejection fraction

Electrocardiography (ECG)
   Established methods, e.g., HRV and DC
   New method: Morphologic Variability (MV)
      Measures variability in shape of heart beats
Calculating Morphologic Distance

Euclidean Distance
*Sequences are aligned “one to one”.*

“Warped” Time Axis
*Nonlinear alignments are possible.*

Use a variant of Dynamic Time-Warping (DTW)

Similar to Smith-Waterman algorithm to align amino acid sequences

Construct distance matrix

Find minimum cost path through it using *dynamic programming*
One Evaluation

764 patients admitted to hospital with non-ST-elevation ACS MI or unstable angina
   But less immediately dangerous than with ST-elevation

Holter ECG recorded at 128 Hz for 2-4 days within 48 hours of event
   ~160M heart beats

Examined only one channel

90 day follow-up for cardiovascular death
Mortality Curves Using Quartile

- High Morphologic Variability (n = 191)
  - 4.19%
  - 5.73%
- Low Morphologic Variability (n = 573)
  - 0.17%
  - 0.70%

HR 8.46
P < 0.001
Wrapping Up the Term
Five Major Topics

Learning a language for expressing computations – Python

Learning about the process of writing and debugging a program

Learning about the process of moving from an ambiguous problem statement to a computational formulation of a method for solving the problem

Learning a basic set of recipes – algorithms

Learning how to use simulations to shed light on problems that don’t easily succumb to closed form solutions
Why Python?

**Relatively easy to learn and use**
- Simple syntax
- Interpretive, which makes debugging easier
- Don’t have to worry about managing memory

**Modern**
- Supports currently stylish mode of programming, object-oriented

**Increasingly popular**
- Used in an increasing number of subjects at MIT and elsewhere
- Increasing use in industry
- Large and ever growing set of libraries
Writing, Testing, and Debugging Programs

**Take it a step at time**
Understand problem
Think about overall structure and algorithms independently of expression in programming language
Break into small parts
Identify useful abstractions (data and functional)
Code and unit test a part at a time
First functionality, then efficiency
Start with pseudo code

**Be systematic**
When debugging, think scientific method
Ask yourself why program did what it did, not why it didn’t do what you wanted it to do.
From Problem Statement to Computation

Break the problem into a series of smaller problems

Try and relate problem to a problem you or somebody else have already solved
  E.g., can it be viewed as a knapsack problem

Think about what kind of output you might like to see, e.g., what plots

Formulate as an optimization problem
  Find the min (or max) values satisfying some set of constraints

Think about how to approximate solutions
  Solve a simpler problem
  Find a series of solutions that approaches (but may never reach) a perfect answer
Algorithms

Big O notation
Orders of growth
  Exponential, Polynomial, Linear, Log
Amortized analysis

Kinds of Algorithms
Exhaustive enumeration, Guess and check, Successive approximation, Greedy algorithms, Divide and conquer, Decision Trees, Dynamic programming

Specific algorithms
  E.g., Binary search, Merge sort

Optimization problems
  Knapsack problems
Simulation

An excuse (and framework) to learn a bit about probability and statistics

An excuse to build interesting programs

The power of random choice
  Much of the world is or appears to be stochastic
  Can be used to solve problems that are not inherently random

Assessing the quality of an answer
  Not always obvious

Building models of parts of the world
Pervasive Themes

Power of abstraction
Systematic problem solving
What Next

Many of you have worked very hard

Only you know your return on investment
  Take a look at early problem sets
  Think about what you’d be willing tackle now

Remember that you can write programs to get answers

There are other CS courses you could take
  6.01, 6.034, 6.005, 6.006