6.045 Automata, Computability, and Complexity / Great Ideas in Theoretical Computer Science

MIT, Spring 2012
https://stellar.mit.edu/S/course/6/sp12/6.045/

Place and Time  
Tuesdays and Thursdays, 2:30-4PM in 32-124  
Recitations: Fridays 11AM in 34-301 (Rotem), Fridays 2PM in 34-302 (Leonid)

Instructor  
Scott Aaronson  
aaronson@csail.mit.edu, www.scottaaronson.com  
32-G638  
Office hours: Mondays 1:30-3PM or by appointment

TAs  
Leonid Grinberg  
leonidg@mit.edu  
Office hours: Tuesdays 6-7PM in 24-311 or by appointment

Rotem Oshman  
rotem@mit.edu  
Office hours: Wednesdays 2-3PM in 32-G630 or by appointment

Graders  
Ishaan Chugh, Liz Simon, Warut Suksompong, Gustavo Goretkin

This course provides a challenging introduction to some of the central ideas of theoretical computer science. It attempts to present a vision of “computer science beyond computers”: that is, CS as a set of mathematical tools for understanding complex systems such as universes and minds. Beginning in antiquity, the course will progress rapidly through finite automata, Turing machines and computability, circuits and other concrete computational models, efficient algorithms and reducibility, NP-completeness, the P versus NP problem, the power of randomness, modern cryptography, computational learning theory, and quantum computing and the physical limits of computation. Class participation is important.

Requirements. Students taking 6.045 will be graded on the following basis:

45% psets
20% midterm exam
35% final exam
+10% participation bonus (including: speaking up in class, attending recitation, participating in recitation, attending office hours, contributing questions and answers on the Piazza site)

Textbook. The “main” textbook for the course is Introduction to the Theory of Computation by Michael Sipser. This book covers most material from the first half of the course. There are also two optional or supplementary books:

2. *The Nature of Computation* by Cris Moore and Stephan Mertens. New book that also covers material from the second half of the course.

**Scribe Notes.** We’re very lucky to have detailed scribe notes, free on the web, from the first time Prof. Aaronson taught this course, as well as PowerPoint slides used by Prof. Nancy Lynch. The notes and slides are accessible at https://stellar.mit.edu/S/course/6/sp12/6.045/materials.html.

**Psets.** There will be 6 psets; schedule of due dates is now available on the Stellar site. Late psets will receive half credit, except in special circumstances such as illness or family emergency. Psets should be turned in during recitation, or into the pset bins outside Prof. Aaronson’s office. For those who prefer Stellar submission, we apologize: we used that in previous years, but it created various logistical nightmares for the graders. Pset problems will often ask you to prove theorems, but we’re looking for clarity of understanding rather than “rigor for rigor’s sake.” So for example, a brief verbal description of an algorithm is better than complicated spaghetti code; and a clear explanation of how you tried to solve a problem and failed will earn partial credit, while obvious nonsense won’t!

**Piazza Site.** Following an extremely successful trial last year, we’ll using Piazza, a Q&A site where you can submit questions about the course material or administrative issues, and get answers from other students, the TAs, or the professor. You can submit questions and answers either anonymously or using your name. If you do the latter, then participation on Piazza can count toward your participation grade. Just go to http://piazza.com/class#spring2012/6045 and sign up using your email address.

**Collaboration Policy.** You are welcome to collaborate on psets, provided that (1) you write up your solutions individually, and (2) you list the names of all collaborators.

**Prerequisites.** We assume that you have taken 6.042 Mathematics for Computer Science, or have equivalent mathematical preparation. In particular, we will assume you have basic “mathematical maturity”: i.e., that you are comfortable both reading and writing mathematical proofs.

**Syllabus (extremely approximate).**

**Logic, Math, and Machines**

- Ancient computational thinking (Euclid et al.)
- Finite automata
- Turing machines and the halting problem
- Oracles and computability
- Gödel’s completeness and incompleteness theorems
- Philosophical considerations (Penrose and “strong AI”)

**Computational Complexity**

- Decision trees and circuits
- Polynomial time and its justification
- Nontrivial examples of polynomial-time algorithms
- The concept of a reduction
- P, NP, and NP-completeness; the Cook-Levin Theorem
- The P versus NP problem and why it’s hard

**Randomness, Adversaries, and the Physical World**
The power of probabilistic algorithms
Private-key cryptography and one-way functions
Public-key cryptography and trapdoor functions
Pseudorandom number generators
Does randomness really help? The P versus BPP question
Zero-knowledge proofs
Computational learning theory
Quantum computing
The ultimate physical limits of computation