EECS 2001A : Introduction to the Theory of Computation

Suprakash Datta

Course page: `http://www.eecs.yorku.ca/course/2001`
Also on Moodle

Administrivia

- Lectures: Tue, Thu 4-5:30 pm online (link on moodle page)
- Tutorial: Fri 4-5:30 pm online
- Tests (50%): 2 tests, 25% each
- Final exam (40%)
- Homework/Assignments (10%)
- Office hours: Tue, Thu 5:30 - 6:30 pm or by appointment, online

Textbook: Michael Sipser. Introduction to the Theory of Computation, Third Edition. Cengage Learning, 2013.

Homework, Grades

- All submissions are using Moodle (or Crowdmark, through Moodle)
- All course information will be online – Moodle and on the public course webpage
- All homework MUST be typed. You will get a zero if you submit handwritten solutions. You may use Office, Google Docs, LaTeX, or other packages but all submissions must be in pdf format.
- We will use Crowdmark for grading. Follow instructions for submissions and re-appraisal requests.
- Grades will be on Moodle

More Administrivia

- Tutorials (1.5 hours/week) are **mandatory**. These will usually be led by me. Most tutorials will be on problem solving. We will shift the tutorial to Thursday and keep Fridays free except for tests.
- Missed tests cannot be made up. If you have a valid medical reason, the weight will be transferred to the final
- If you have serious non-medical reasons (having work is not one), talk to me. We will deal with those on an ad hoc basis
- Breaches of academic honesty: Will be dealt with very strictly. Read the detailed policies on the class webpage

Resources

- We will follow the textbook
- There are more resources than you can use, including books, lecture slides and notes, online texts, video lectures, assignments
- My slides are not a not a substitute for, or a comprehensive summary of, the textbook

Next

What is this course about?

The Big Picture

- What is Computation?
- Why do we need a Theory of Computation?

The Big Picture - 2

Previous courses (EECS 1012,1022, 2030 (?), 2011(?)):
Given a problem:

- Figure out an algorithm
- Code it, debug, test with "good" inputs
- Some idea of running time, asymptotic notation
- Implicit assumption: you can write a program to solve any problem. Is this true?

Some Problems of Interest

Which of these are solvable?

- Code reachability problem
- Program termination problem
- Program correctness problem
- Guaranteeing that a system is secure

Let us take a step back....

Reasoning about Computation

Computational problems may be

- Solvable, quickly
- Solvable in principle, but takes an unfeasible amount of time (e.g. thousands of years on the fastest computers available)
- (provably) not solvable

What does "not solvable" mean?

Reasoning about Computation - 2

- Need formal reasoning to make credible conclusions
- Mathematics is the language developed for formal reasoning
- As far as possible, we want our reasoning to be intuitive

Course Objectives

- Learning about different computation models
  - Finite Automata
  - Pushdown Automata
  - Turing Machines
- Reasoning about computation
  What these models can and cannot do
  - What does it mean to say "there does not exist an algorithm for this problem"?
  - Reason about the hardness of problems
- Reasoning about CLASSES of problems
  - There are different degrees of hardness of problems
  - Eventually, build up a hierarchy of problems based on their hardness.

Important point

- We are concerned with solvability, NOT efficiency.

- EECS 2011 (Data Structures), EECS 3101 (Design and Analysis of Algorithms) deal with efficiency issues.

Theory of Computation - parts

- Automata Theory (EECS 2001): Talks primarily about the power of computational models
- Complexity Theory (EECS 4115): Classification of problems by runtime or other performance measures
- Computability Theory (EECS 2001, 4101): Solvability of problems

My Objectives

- Help you understand the relevance and importance of Theory of Computation
- Instill in you the mindset of asking 'why' and 'how'
- Assist you in thinking at different levels of abstraction

My Expectations

- You will attend classes and tutorials regularly
- Want to build solid foundations
- Ask for help when needed
- Follow academic honesty regulations (see the class webpage for more details on policies)

To do well in this class

- This is an applied Mathematics course – practice instead of reading
- Try to get as much as possible from the lectures
- Keep the big picture in mind. ALWAYS.
- Follow along in class rather than take notes. Ask questions in class or outside class
- Keep up with the class
- Read the book, not just the slides