# COSC 39 Theory of Computation

Hsien-Chih Chang
HSIEN-CHIH.CHANG@DARTMOUTH.EDU

## Course Description

Computers, capable of performing universal computations and simulating other computers, is one of the greatest achievements of human civilization. One of the key components is the ability to abstract and formalize computation itself as a concrete mathematical object, called the *Turing machine*. Surprisingly, such definition is sufficiently robust that any attempts to formalize computation end up arriving at definitions provably equivalent to Turing machines, which encapsulate all the physical computing devices known to human beings.

While the whole discipline has been growing explosively for the past several decades, many important questions regarding the fundamental principle of computation remain unanswered; this includes the *P versus NP*, one of the Millennium Prize Problems that resists all attempts to be solved despite serious attempts from the most brilliant minds.

In this course we will study computation through the abstract lenses of Turing machines (including their variations), the benefit of knowing computations from a theoretical perspective, and how they relate to the rest of computer science.[1]

## Course Objectives

At the end of the course, the students are expected to be able to:

- *Conceptually*:
  - Treat programs as abstract objects
  - Understand the relation between machine models and languages classes
  - Have a concrete mental picture of nondeterminism
  - Explain and interpret the importance of universal computation
  - Explain formally and informally why the P versus NP problem is important/unimportant to the modern study of computation
  - Describe how reductions create relationship between languages
- *Skill-wise*:
  - Construct convincing and rigorous mathematical arguments
  - Formally define problems and algorithms
  - Explain and utilize the equivalence between three models of regular languages
  - Compare the power of different machine models using reductions
  - Describe and prove some basic properties about P and NP
  - Demonstrate NP-hardness by performing reductions correctly

[1] After taking the course, when your future supervisor assigns you a million-dollar project involving some difficult optimization tasks, you can explain your lack of progress by *formally demonstrating* that the assigned task is as hard as these other thousands of problems that nobody knows how to solve efficiently since the birth of computers. Sit back, and wait for your well-deserved promotion.

## Prerequisite

A proof-based course like *COSC 30: Discrete Mathematics* or equivalent is required as prerequisite. Alternatively, students with reasonable mathematical background and maturity, in particular the ability to follow arguments of a proof, are welcome to take the course with instructor's permission. We will hand out homework 0 in the first week to give the students an idea of the background knowledge and skills expected.

## Related Courses

- *COSC 40 Computational Complexity*: This is the *sequel* of COSC 39. The theory of computational complexity aims at studying and classifying computational problems using reductions based on their *difficulties* — the various resources needed to solve the problem.[2] Connections between problems help us to identify central problems of study, whose breakthrough often comes with development of new insights and techniques.
- *COSC 31 Algorithms*: This is the *flip side* of COSC 39. Algorithms are ubiquitous in computer science; this course focuses on the design and analysis of (mostly classical) algorithms, with similar emphasis on the development of rigorous and unambiguous ways of communicating ideas. Both courses will prepare you for a similar skill set that is necessary for any future career related to computer science. Unlike COSC 39 and 40, the goal of algorithms is to *actually* solve the problems (at least with (lots of) simplifying assumptions[3]).

## Textbook

There is no *required* textbook. We will use the following book as our main reference:

- *Introduction to the Theory of Computation* by Michael Sipser.
  Either the 2nd or 3rd edition is fine.

We will also use a mix of books and class notes listed below as references. Additional relevant reading materials will be linked on the course webpage each week.

- *Models of Computation*, lecture notes by Jeff Erickson.
- *Computational Complexity: A Modern Approach* by Sanjeev Arora and Boaz Barak.

## Teaching Methods and Expectations

*Due to COVID-19 all lectures, sessions, and office hours will be online. All meeting-related information will be on the course webpage.*

The ***lecture hours*** will be spent alternatingly between lectures and working sessions. We will utilize the X-hour (Tuesday 1:40–2:30p) and have the following weekly schedule:

- Monday and Wednesday: lectures
- Tuesday and Thursday: working sessions

[2] The goal of complexity theory is not trying to solve the problems themselves, but to find out who to blame when nobody can solve the problem.

[3] in a similar fashion to physicists assuming an elephant to be a three-dimensional round object floating in space.

I firmly believe that the real learning only happens when the students are actually engaging the materials through a set of well-designed problems; not through passive learning like sitting in lectures. Consequently, the students are *not* required to attend the lectures live (while all of you are still encouraged to). All the lectures will be recorded and uploaded after class.

However, the students are *strongly encouraged* to participate in ***working sessions***. The goal of these sessions is to make sure that the students are actively participating in learning the subject, and making sure that everyone is still following the class. During each working session we will hand out a few basic practice problems that can be solved using the content discussed in the previous lecture. The students will work in groups actively trying to solve the problem. The instructors will be around helping to answer any questions coming up. Outside the regular lecture hours the instructor will also provide ***office hours*** on a weekly basis; all students are encouraged to attend this discussion-based session.

There will also be weekly ***assignments***. Each week we will assign a few questions, generally harder than the practice problems. These problems are designed specifically at the right difficulty so that it is challenging (and even sometimes frustrating), but rewarding and aiming to expand our knowledge and skills after solving them (maybe with some help from your colleagues). Therefore it is okay if you find the problems hard; that is by design. The students are expected to spend a reasonable amount of time working on the assignments. In a sense, these are the *real* driver of the class. Discussions with other students and using online resources are encouraged. Some homework problems require additional readings; all the extra readings required for the homework will be made available.

Tentatively we will have two ***midterms*** and one ***final exam***. The formats of the exams are to be determined. We plan to have one of the midterms to be a *debate* between the students. Other exams are likely to be a mixture between the *written* and *oral* formats.

## Grading

The final grading will be based on the following factors (the tentative percentages are included):

- Practice problems (10%)
- Assignments (30%)
- Midterms (20% each)
- Final exam (20%)

About the grading rubric: Because we have a small-size class, I would prefer to talk to each of you and set up concrete goals after reviewing your submission to homework 0. At the middle of the term we will review the goals together and revise them if necessary. At the end of the term the grades will be assigned based on the predetermined goals.

## Tentative Course Schedule and Assignments

All materials are subject to change based on the actual class interaction and student feedback.

- Week 0: ***Introduction***
  - What is this course about? Knowledge, skill, philosophy
- Week 1: ***Languages and Machine Models***
  - Strings, languages, regular expressions

  - Finite automata: definitions, construction and design, examples
- Week 2: ***Nondeterminism***
  - NFAs: intuition, definitions, equivalence with DFAs
  - Closure properties, equivalence between NFAs and regular expressions
- Week 3: ***Beyond Regularity***
  - Non-regular languages, fooling sets
  - Myhill-Nerode Theorem, minimizing DFAs, pumping lemma
- Week 4: ***Turing Machines and Decidability***
  - Turing machines: definitions, examples, decidable languages
  - Universal Turing machines, code is data; Church-Turing thesis, undecidability
- Week 5: ***Resources and Verification***
  - Time complexity, verification, P vs NP
  - Verifiers, Nondeterministic TMs, equivalence
- Week 6: ***Reductions***
  - Mapping and oracle reductions, polynomial-time reductions, NP-hardness
  - More NP-hard problems through reductions, Cook-Levin theorem

**Advanced topics (tentatively)**

- Week 7: ***Randomness***
  - Random walks, Markov chains, sampling
  - Online models
- Week 8: ***Interaction***
  - Interactive proofs, games, zero-knowledge proofs
  - Distributed computing
- Week 9: ***Onward***
  - Space complexity, circuits and parallel computing, functional programming, grammars and compilers, cryptography, quantum computing, time travel, and more!