BU CS 332 – Theory of Computation

Please point a browser to
https://forms.gle/wzKrQznUm2eLG3yr8
for in-class polls

- Lecture 1:
  - Course information
  - Overview

Reading:
Sipser Ch 0

Mark Bun

January 22, 2025

Course Information

Course Staff

- <u>Me:</u> **Mark** Bun (he/him/his)
  - Tentative office hours: M 4:30-6, Tu 2-3:30
  - Research interests: Theory of computation (!)

    More specifically: Computational complexity, data privacy, cryptography, foundations of machine learning

- <u>Teaching Fellow:</u> Nadya Voronova
  - Tentative office hours: M 9:30-11, F 5-6
  - Research interests: Theory of computation

    Sublinear/space-bounded computation, quantum computation

- <u>Teaching Fellow:</u> Mandar Juvekar
  - Tentative office hours: Tu 3:30-5, Th 11:15-12:15
  - Research interests: Theory of computation

    Quantum computation, cryptography, meta-complexity

Course Webpage

https://cs-people.bu.edu/mbun/courses/332_S25/

Serves as the syllabus and schedule

Check back frequently for updates!

CS 332: Elements of the Theory of Computation, Spring 2025

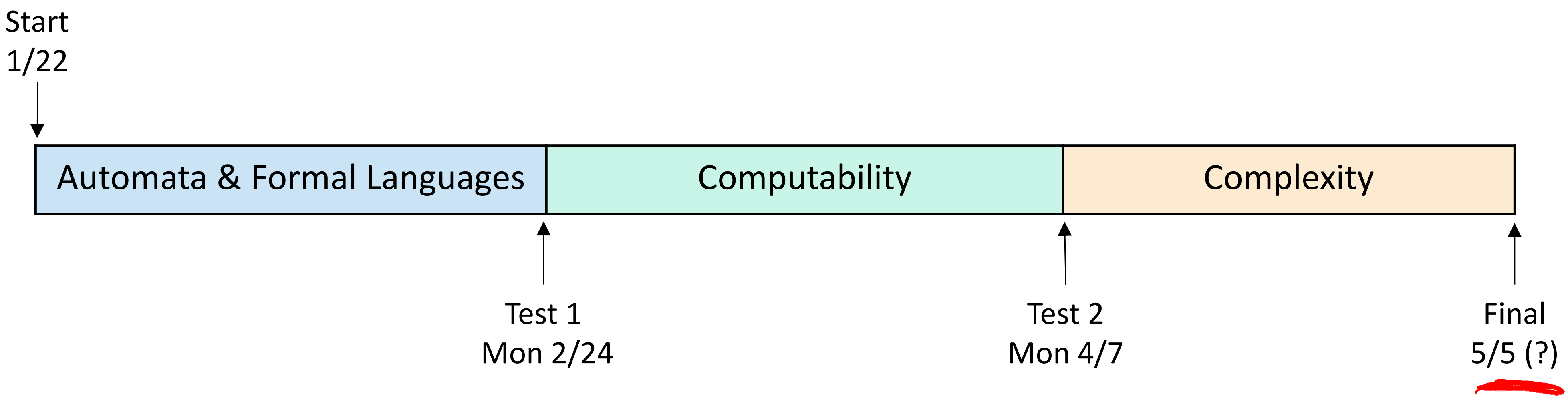
Course Overview

This course is an introduction to the theory of computation. This is the branch of computer science that aims to understand which problems can be solved using computational devices and how efficiently those problems can be solved. To be able to make precise statements and rigorous arguments, computational devices are modeled using abstract mathematical "models of computation." The learning objectives of the course are to:

- Foremost, understand how to rigorously reason about computation through the use of abstract, formal models.
- Learn the definitions of several specific models of computation including finite automata, context-free grammars, and Turing machines, learn tools for analyzing their power and limitations, and understand how they are used in other areas of computer science.
- Learn how fundamental philosophical questions about the nature of computation (Are there problems which cannot be solved by computers? Can every problem for which we can quickly verify a solution also be solved efficiently?) can be formalized as precise mathematical problems.
- Gain experience with creative mathematical problem solving and develop the ability to write correct, clear, and concise mathematical proofs.

Instructor:	Mark Bun, mbun [at] bu [dot] edu
Instr. Office Hours:	TBD
	TBD
Teaching Fellow:	Mandar Juvekar, mandarj [at] bu [dot] edu
Mandar's Office Hours:	TBD
	TBD
Teaching Fellow:	Nadya Voronova, voronova [at] bu [dot] edu
Nadya's Office Hours:	TBD
	TBD
Class Times:	Mon, Wed 2:30-3:45 PM (CAS B12)
Discussion Sections:	Fri 9:30-10:20 AM (CAS 203)
	Fri 11:15 AM-12:05 PM (MCS B37)
	Fri 12:30-1:20 PM (CAS 233)
	Fri 3:30-4:30 PM (CAS 233)

Course Structure



Grading

- Homework (33%): Roughly 11 of these
- In-class tests (55%):
  - Test 1 (16%)
  - Test 2 (16%)
  - Final (23%)
- Participation (12%): In-class polls, discussions, etc.

Homework Policies

- Weekly assignments due Tuesday @ 11:59PM
- No late days
- Lowest homework score will be dropped

- Homework to be submitted via Gradescope
  - Entry code: NY66PW

- You are encouraged to typeset your solutions in LaTeX (resources available on course webpage)

- HW1 will be out, due Tu 1/28 (some housekeeping, brief review)

Homework Policies: Collaboration

- You are encouraged to work with your classmates to discuss most of the homework problems
- HOWEVER:
  - You may collaborate with at most 3 other students
  - You must acknowledge your collaborators and write “Collaborators: none” if you worked alone
  - You must write your solutions by yourself
  - You may not share written solutions
  - You may not search for solutions on the web or ask Chat-GPT to produce solutions for you
  - You may not receive help from anyone outside the course (including students from previous years)
- Some problems will be marked “INDIVIDUAL”. **No collaboration** is allowed on these problems.

Collaboration Dilemma

If you worked alone on a homework assignment…

(a) You don't need to include a collaboration statement

(b) You should invent a fake collaborator and acknowledge them appropriately

(c) You should include the collaboration statement "Collaborators: none"

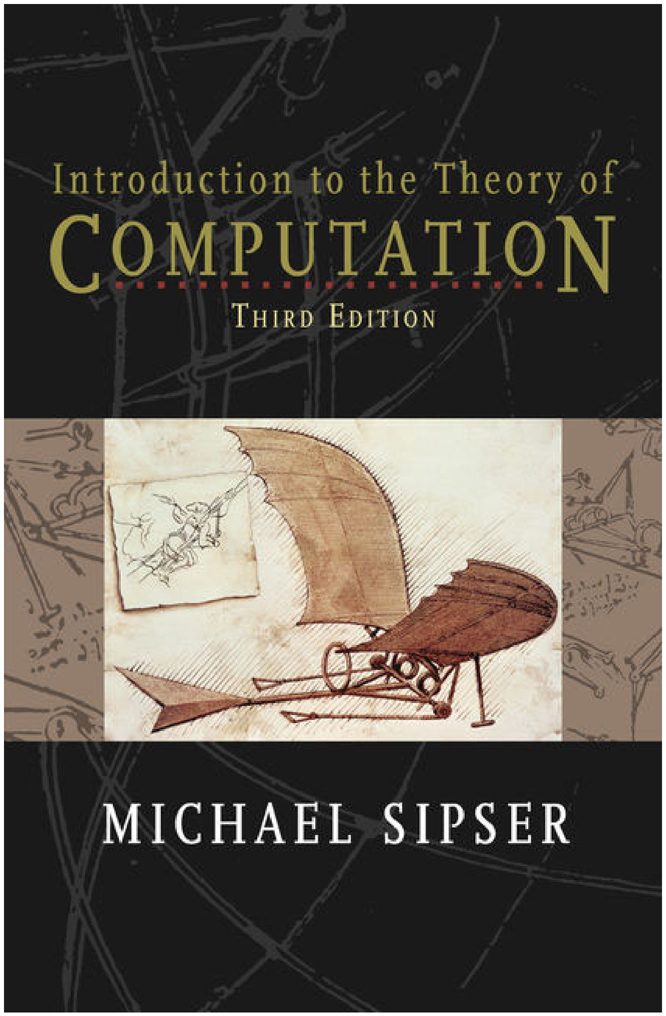
(d) You should hurriedly call up three of your friends in the class at 11:55PM, briefly discuss the problems, and acknowledge them

Homework Policies: Collaboration

Details of the collaboration policy may be found here:
https://cs-people.bu.edu/mbun/courses/332_F25/handouts/collaboration-policy.pdf

Important: Sign this document to affirm you understand it, and turn it in via Gradescope by 11:59PM, Tue 1/28

Textbook



Introduction to the Theory of Computation (Third Edition)
by Michael Sipser

- It’s fine if you want to use an older edition, but section numbers may not be the same
- Other resources available on course webpage

Participation

- Your class participation score (13% of the course grade) will be determined by
  - Answering in-class polls on Google Forms
  - Handing in completed worksheets at the end of each discussion section
  - Completing other activities, e.g., course feedback forms

- You can also increase your participation score by participating thoughtfully in lecture, discussion, office hours, and on Piazza

Piazza

- We will use Piazza for announcements and discussions
  - Ask questions here and help your classmates
  - Please use private messages / email sparingly

https://piazza.com/bu/spring2025/cs332

Disability and Access Services

- If you have a Faculty Accommodation Letter for Disability & Access Services, please send it to me by Piazza or email as soon as you can
- Class sessions will be privately audiotaped (for student personal use only)
- Let me know if you are interested in a (paid) position as an in-class note taker: https://www.bu.edu/disability/accommodations/procedures/specific/notetaker-service/

Expectations and Advice for Succeeding in CS 332

Learning Objectives

- <u>Understand how to rigorously reason about computation using mathematical models</u>
- Learn about several specific models of computation, how to analyze them, and how they are used throughout computer science
- Learn how to pose deep philosophical questions about the nature of computation as precise mathematical problems
- Gain experience with problem solving and develop proof-writing skills

Our (the Course Staff's) Responsibilities

- Guide you through difficult parts of the material in lecture
- Encourage active participation in lectures / section
- Assign practice problems and homework that will give you a deep understanding of the material
- Give detailed (formative) feedback on assignments
- Be available outside of class (office hours, Piazza)
- Regularly solicit feedback to improve the course

Your Responsibilities

- Concepts in this course take <u>time</u> to sink in. Keep at it and be careful not to fall behind.
- Do the assigned reading on each topic before the corresponding lecture
- Take advantage of office hours
- Participate actively in lectures/sections and on Piazza
- Allocate lots of time for the course: comparable to a project-based course, but spread more evenly

Prerequisites

This class is fast-paced and assumes experience with mathematical reasoning and algorithmic thinking

You must have passed CS 330 – Intro to Algorithms

This means you should be comfortable with:

- Set theory
- Functions and relations
- Graphs
- Pigeonhole principle
- Propositional logic

- Asymptotic notation
- Graph algorithms (BFS, DFS)
- Dynamic programming
- NP-completeness

Come talk to me if you have concerns about your preparation for the course

Advice on Homework

- Start working on homework early! You can get started as soon as it's assigned.
- Spread your homework time over multiple days.
- You may work in groups (of up to 4 people), but <u>think about each problem for at least 30 minutes before your group meeting</u>.

- To learn problem solving, you have to do it:
  - Try to think about how you would solve any presented problem before you read/hear the answer
  - Do exercises in the textbook in addition to assigned homework problems

JUST DO IT

Advice on Reading

- Not like reading a novel
- The goal is not just to find the answers, but to learn and understand the techniques
- Always try to predict what's coming next
- Always think about how you would approach a problem before reading the solution – maybe you'll discover a different way of solving it!
- This applies to things that are not explicitly labeled as exercises or problems

Academic Integrity

Extremely important: Read and understand the Collaboration and Honesty policy before you sign it

Violations of the collaboration policy…will result in an automatic failing grade and will be reported to the Academic Conduct Committee (ACC). The ACC often suspends or expels students deemed guilty of plagiarism or other forms of cheating.

If you find yourself in a desperate situation:

- Hand in as much of the assignment as you're able to complete
- Remember the lowest HW grade is dropped
- Talk to us! We want to help

…cheating is seriously not worth it

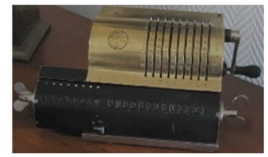
Course Overview

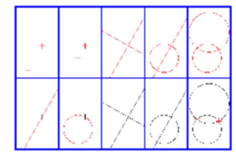
Objective

Build a *theory* out of the idea of *computation*

What is "computation"

- Examples:
  - Paper + pencil arithmetic
  - Abacus
  - Mechanical calculator
  - Ruler and compass geometry constructions
  - Java/C programs on a digital computer





$$\begin{array}{r} 332 \\ +729 \\ \hline 1061 \end{array}$$

- For us: Computation is the processing of information by the unlimited application of a finite set of operations or rules

Other examples of computation?

Playing Chess (or any game)

Baking (following a recipe)

Photosynthesis? (Biological processes) ⟶ DNA, protein folding

Following sheet music

Speaking language

Building Legos

Following road directions

Journalism (processing news into articles)

https://www.youtube.com/watch?v=0ltvGZUvpvw ⟵ Slime molds

Quantum computation

Computation in the brain?

Internet / distributed protocols

Equilibria in markets

Cryptography, security protocols

What do we want in a "theory"?

- General ideas that apply to many different systems
- Expressed simply, abstractly, and precisely

- Generality
  - Independence from Technology: Applies to the future as well as the present
  - Abstraction: Suppresses inessential details

- Precision: Can prove formal mathematical theorems
  - Positive results (what *can* be computed): correctness of algorithms and system designs
  - Negative results (what *cannot* be computed): proof that there is no algorithm to solve some problem in some setting (with certain cost)

Parts of a Theory of Computation

- Models for machines (computational devices)
- Models for the problems machines can be used to solve
- Theorems about what kinds of machines can solve what kinds of problems, and at what cost

This course: Sequential, single-processor computing

Not covered:

- Parallel machines
- Distributed systems
- Quantum computation
- Real-time systems
- Mobile computing
- Embedded systems

What is a (Computational) Problem?

A single question with infinitely many instances

Examples:

abba - YES abb - NO

- Parity: Given a string consisting of a's and b's, does it contain an even number of a's?
- Primality: Given a natural number x (represented in binary), is x prime?
- Halting Problem: Given a C program, can it ever get stuck in an infinite loop?

For us: Focus on *decision* problems (yes/no answers) on *discrete* inputs

What is a (Computational) Problem?

For us: A problem will be the task of determining whether a *string* is in a *language*

PARITY: Given a string x (consisting of a's & b's) does x have an even # of a's?

$\iff$ Given a string $x \in \{a,b\}^*$, is $x \in L$

where $L = \{ y \in \{a,b\}^* \mid y \text{ has an even \# of a's} \}$

What is a (Computational) Problem?

For us: A problem will be the task of determining whether a *string* is in a *language*

PARITY

- **Alphabet**: A finite set Σ

  Ex. $\Sigma = \{a, b, \ldots, z\}$

  $\Sigma_1 = \{a, b\}$

- **String:** A finite concatenation of alphabet symbols

  Ex. $bqr, ababb$

  abba aab baa

  ε denotes empty string, length 0

  Σ^* = set of all strings using symbols from Σ

  $\{a, b\}^*$

  $= \{y \mid y \text{ is a (finite) string of symbols from } \Sigma\}$

- **Language:** A set L of strings

  $L \subseteq \Sigma_1^*$

Examples of Languages

Parity: Given a string consisting of a's and b's, does it contain an even number of a's?

$\Sigma = \{a, b\}$ $\quad L = \{y \in \{a,b\}^* \mid y$ has an even # of a's$\}$

Primality: Given a natural number x (represented in binary), is x prime?

$\Sigma = \{0, 1\}$ $\quad L = \{y \in \{0,1\}^* \mid y$ represents a prime$\}$

Halting Problem: Given a C program, can it ever get stuck in an infinite loop?

$\Sigma = \{$extended ASCII$\}$ $\quad L = \{P \mid P$ is a C program that could enter an infinite loop$\}$

Primality language

Which best represents the language corresponding to the Primality problem? (I.e., strings over the alphabet $\{0, 1\}$ that are binary representations of prime numbers.)

Let's say the most significant bit is on the left, so "100" is the binary representation of 4.

$PRIMES = \{ x \in \{0,1\}^* \mid x$ represents a prime in binary $\}$

(a) $\{2, 3, 5, 7, \dots\}$

(b) $\{10, 11, 101, 111, \dots\}$

(c) $\{11, 111, 11111, 1111111, \dots\}$

(d) $\{11, 011, 101, 110, 111, 0111, \dots\}$