BU CS 332 – Theory of Computation

- Lecture 1:
  - Course information
  - Overview

Reading:

Sipser Ch 0

Mark Bun

January 25, 2021

Course Information

Course Staff

- <u>Me:</u> **Mark** Bun (he/him)
  - At BU since Sept. 2019
  - Office hours: Wed 4-5PM, Th 9-10AM
  - Research interests: Theory of computation (!)

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

- <u>TF:</u> Nadya Voronova
  - Office hours: Tu 3-4PM, Wed 9-10AM

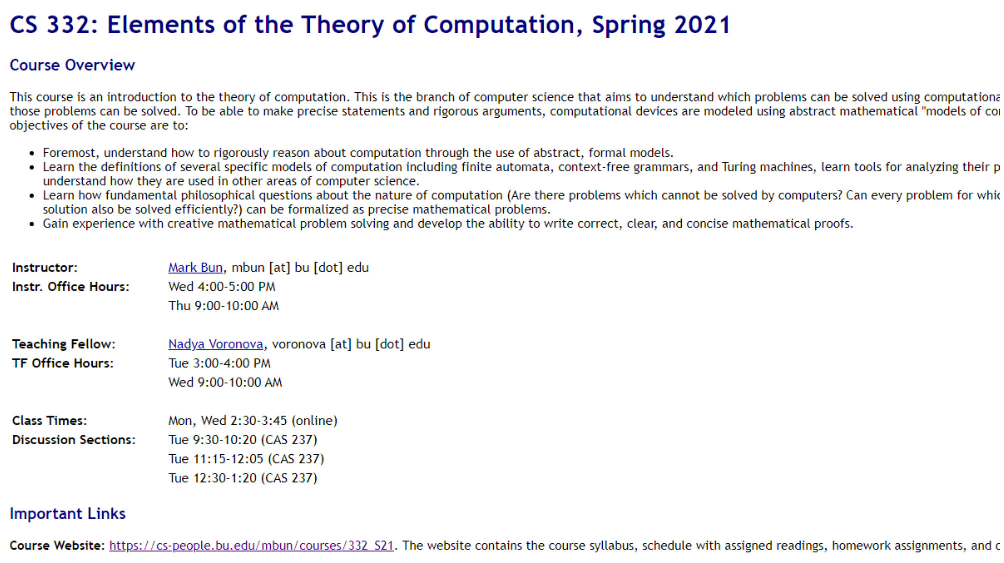
- …hopefully others

Course Webpage

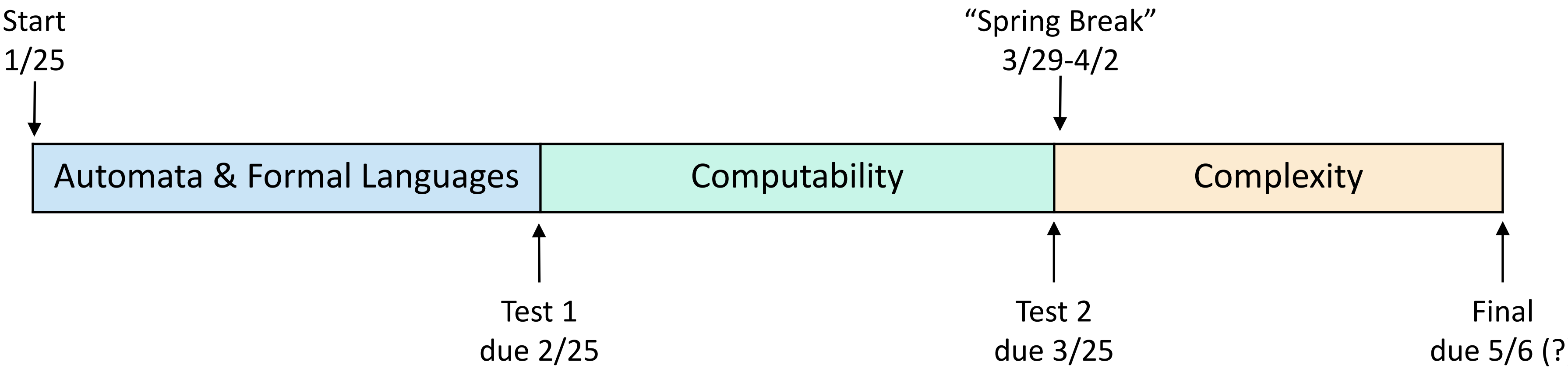
https://cs-people.bu.edu/mbun/courses/332_S21/

Serves as the syllabus and schedule

Check back frequently for updates!



Course Structure



Grading

- Homework (45%): Roughly 10 of these
- Take-home tests (40%):
  - Test 1 (10%)
  - Test 2 (10%)
  - Final (20%)
- Participation (15%): Gradescope check-ins, HW0, etc.

Homework Policies

- Weekly assignments due Thursday @ 11:59PM
- No late days, no extensions
- Lowest homework score will be dropped

- Homework to be submitted via Gradescope
  - Entry code: 2RB88B

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

- HW0 out, due Th 1/28 (just some housekeeping)
- HW1 to be released on Th 1/28, due Th 2/4

Homework Policies: Collaboration

- You are encouraged to work with your classmates to discuss homework problems

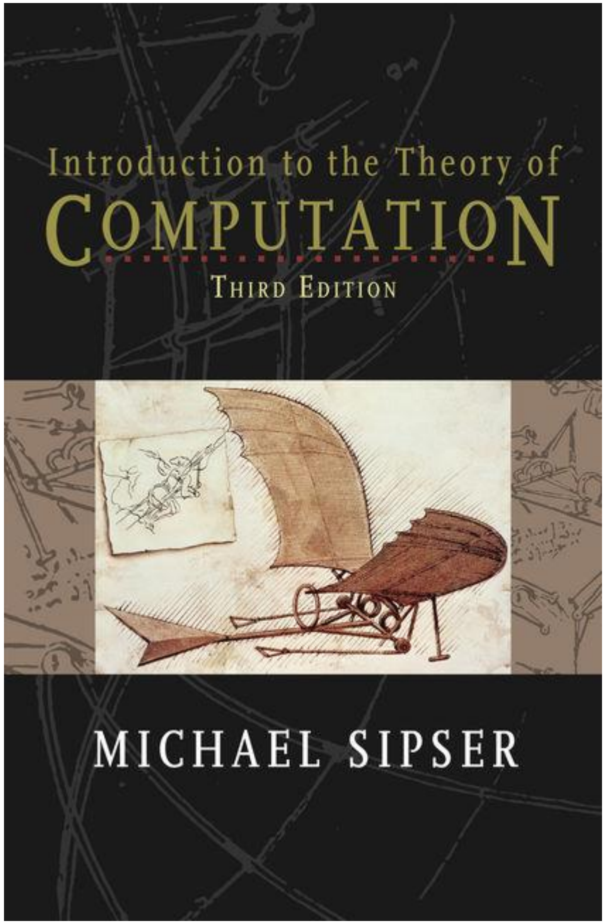
- HOWEVER:
  - You may collaborate with at most 3 other students
  - You must acknowledge your collaborators and write “Collaborators: none” if you worked alone
  - You must write your solutions by yourself
  - You **may not** share written solutions
  - You **may not** search for solutions using the web or other outside resources
  - You **may not** receive help from anyone outside the course (including students from previous years)

Homework Policies: Collaboration

Details of the collaboration policy may be found here:
https://cs-people.bu.edu/mbun/courses/332_S21/handouts/collaboration.pdf

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

Gradescope Check-ins

- Your class participation score (15% of the course grade) will be determined by your answers to short reflection questions after each lecture
- Questions will be based on our in-class polls and discussions
- You'll be graded 50% on participation and 50% on correctness

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/spring2021/cs332

You can earn bonus points toward your participation grade by participating thoughtfully on Piazza

Expectations and Advice for Succeeding in CS 332

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 some time 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 questions 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 think about each problem for at least 30 minutes before your group meeting.

- 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

Advice on Reading

- Not like reading a novel
- The goal is not to find out 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
- 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

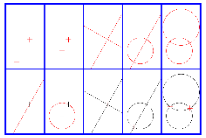
Course Overview

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



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

Other examples of computation?

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:

- 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 recognizing whether a *string* is in a *language*

What is a (Computational) Problem?

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

- **Alphabet**: A finite set Σ

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

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

  Ex. $bqr, ababb$

  ε denotes empty string, length 0

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

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

Examples of Languages

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

Σ = L =

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

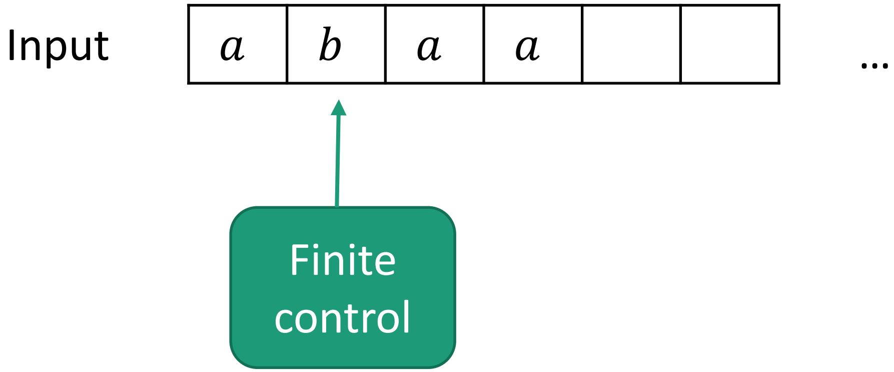
Σ = L =

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

Σ = L =

Machine Models

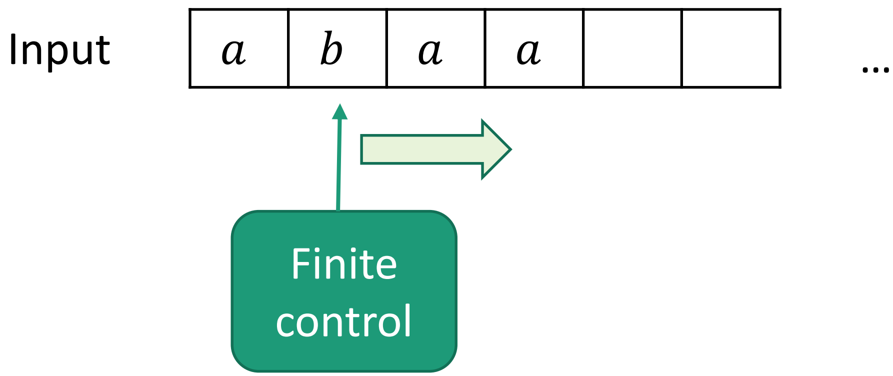
Computation is the processing of information by the **unlimited application** of a **finite set** of operations or rules



Abstraction: We don't care how the control is implemented. We just require it to have a finite number of states, and to transition between states using fixed rules.

Machine Models

- <u>Finite Automata (FAs):</u> Machine with a finite amount of unstructured memory

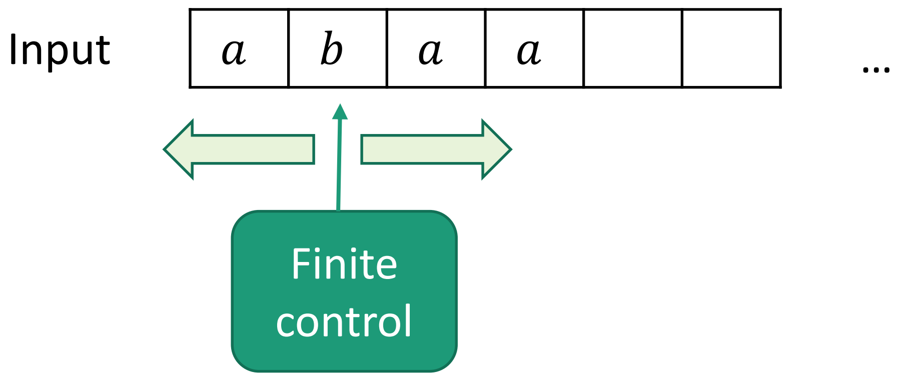


Control scans left-to-right
Can check simple patterns
Can't perform unlimited counting

Useful for modeling chips, simple control systems, choose-your-own adventure games…

Machine Models

- Turing Machines (TMs): Machine with unbounded, unstructured memory



Control can scan in both directions
Control can both read and write

Model for general sequential computation
Church-Turing Thesis: Everything we intuitively think of as "computable" is computable by a Turing Machine

What theorems would we like to prove?

We will define classes of languages based on which machines can recognize them

Inclusion: Every language recognizable by a FA is also recognizable by a TM

Non-inclusion: There exist languages recognizable by TMs which are not recognizable by FAs

Completeness: Identify a “hardest” language in a class

Robustness: Alternative definitions of the same class

Ex. Languages recognizable by FAs = regular expressions

Why study theory of computation?

- You’ll learn how to formally reason about computation
- You’ll learn the technology-independent foundations of CS

Philosophically interesting questions:

- Are there well-defined problems which cannot be solved by computers?
- Can we always find the solution to a puzzle faster than trying all possibilities?
- Can we say what it means for one problem to be “harder” than another?

Why study theory of computation?

- You'll learn how to formally reason about computation
- You'll learn the technology-independent foundations of CS

Connections to other parts of science:

- Finite automata arise in compilers, AI, coding, chemistry https://cstheory.stackexchange.com/a/14818
- Hard problems are essential to cryptography
- Computation occurs in cells/DNA, the brain, economic systems, physical systems, social networks, etc.

Why study theory of computation?

Practical knowledge for developers

"Boss, I can't find an efficient algorithm.
I guess I'm just too dumb."

"Boss, I can't find an efficient algorithm
because no such algorithm exists."

Will you be asked about this material on job interviews?

No promises, but a true story…