What is this course about?

What is “Computation”?

• Can we define computation without referring to a modern computer?
• Can we define a computer mathematically?
  – Yes → Turing machines
• Is computation definable independent of present-day engineering limitations, understanding of physics, etc.?
• Can a computer solve any problem, given enough time and disk-space?
• Or are there fundamental limits to computability?
What is this course about?

This course is about the fundamental capabilities and limitations of computers/computation.

This course covers 3 areas, which make up the theory of computation:

– Automata and Languages
– Computability Theory
– Complexity Theory
Areas of Theory of Computation

- What can be computed?
- Can a computer solve any problem, given enough time and disk-space?
- How fast can we solve a problem?
- How little disk-space can we use to solve a problem?
- What problems can we solve given really very little (constant) space?
What problems can a computer solve?

- Not all problems!!!
- Eg. Given a C-program, we cannot check if it will not eventually crash
- Verification of correctness of programs is hence impossible!
What problems can a computer solve?

- Even checking whether a C-program will halt/terminate is not possible!

Program(n)
  Input integer n>1
  while (n != 1)
    if (n is even)
      return Program(n/2)
    else
      return Program(3n+1)

E.g., for input n=34: 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

Impossible to decide whether this program terminates for all possible input values.
What problems can a computer solve?

- Certain problems cannot be solved by computers
  - I.e., they have no algorithmic solution
- A Turing machine, a simple model for a computer, can do everything that a computer can do (Church-Turing thesis)
- We can use a Turing machine to determine what a computer can and can’t do (i.e., compute)
Areas of Theory of Computation

**Computability**
- How fast can we compute a function?
- How much memory space do we require?
  - Polynomial time computable
  - Non-det Poly Time (NP)
  - Approximation, Randomization

**Complexity**
Functions that cannot be computed fast:
- Applications to security
- Encrypt fast, decryption cannot be done fast
  - RSA cryptography
  - Web applications
What can we do with machine with finite memory space?

- Examples:
  - traffic signals,
  - vending machines
  - hardware circuits
  - ...

- Applications:
  - pattern matching,
  - modeling,
  - verification of hardware,
  - ...

Areas of Theory of Computation

What can we do with machine with finite memory space?

- Examples:
  - traffic signals,
  - vending machines
  - hardware circuits
  - ...

- Applications:
  - pattern matching,
  - modeling,
  - verification of hardware,
  - ...

COMPUTABILITY

COMPLEXITY

AUTOMATA
Models

- Automata:
  - DFA/ NFA
  - Foundations of computing
  - Mathematical methods of argument
  - Simple setting
Models

- Context-free languages
  - Grammars, parsing
  - Finite state machines with recursion (or stack)
  - Still a simple setting but infinite space
Models

• Turing machines (1940s):
  – The most general notion of computing
  – The Church-Turing thesis

• Allow to study limits of computability
  – Can we solve any mathematical question methodically?
  – Gödel’s Theorem: NO!!!
  – “Even the most powerful machines cannot solve some problems”
Models

• Alan Turing
  – “On Computable Numbers, with an Application to the Entscheidungs problem” (1936)
  – Proved uncomputable functions exist (“halting problem”)
  – Church-Turing thesis
  – Cryptanalysis work breaking Enigma in WW2

• Noam Chomsky
  – “Logical Structure of Linguistic Theory” (1957)
  – Introduced the notion of formal languages arguing generative grammars are at the base of natural languages
  – Hierarchy of formal languages that coincides with computation

• Rabin and Scott
  – “Finite automata and their decision problems” (1959)
  – Introduced non-deterministic automata and the formalism we still use today
About this Course

• Theory of Computation traditionally considered challenging
  – I expect (and hope) that you will find this to be true =)

• A very different kind of course
  – In many ways, a pure theory course
    • But very grounded (the models of computation are not abstract at all)
  – Proofs are an integral part of the course, although I and the text both rely on informal proofs
    • But the reasoning must still be clear

• The only way to learn this material is by doing problems
  – You should expect to spend several hours per week on homework
  – You should expect to read parts of the text 2-3 times
  – Do not give up after 5 minutes if you are stumped by a problem!
Instructor: Lorenzo

HTA: Kevin

Isidor

Koyena

Joseph

Jamison

Shamay
Textbook and course materials

- Textbook: Michael Sipser
  *Introduction to Theory of Computation*
  (2\textsuperscript{nd} ed. recommended; 1\textsuperscript{st} ed may be ok)

- Course website:
  
  https://cs.brown.edu/courses/csci1010/
  
  - Use the course Syllabus for reference
  - Lecture Slides available on the website
  - Homework and Lab assignments published on the website

- Piazza
Homework

• Homeworks every week: out on Thursday after class and due the on Thursday of the next week before start of class. Posting dates and hand-back dates posted on the website.

• Homeworks submitted using Gradescope.

• Be rigorous but not excessively verbose

• Work in a group, but think and try to solve each problem yourself!
  – List you collaborators on your submission
  – Always write your own solutions
  – Do not “distribute the problems within the group”
  – Collaboration is not allowed for take-home midterm

• Simple late-HW policy: Late submission will not be accepted.

• At the beginning of the course, all students are required to subscribed to such collaboration policy (http://cs.brown.edu/courses/csci1010/files/doc/collab.pdf), or the digital version.
Laboratories

At the beginning of the semester students will be assigned a lab meeting time. If you need to change lab meeting times for a given week, please notify the TAs at least one week before your lab meeting.

Every two weeks, students will be assigned two lab problems, and will be grouped with two other students who were assigned the same problems. They will be responsible for preparing a solution to the problems before their lab meeting time. Group members will present their problem and proposed solution to their lab group, working through any remaining difficulties during lab time. When done, the group will orally present their solutions to all lab problems to a TA. Each student will be given a lab grade of 0 if absent, 5 if present, and 10 if present and demonstrating a satisfactory understanding of their assigned lab problem.

You may not take written notes away from lab. Solutions for all lab problems must be written up individually and submitted as part of the homework assignment for that week.
How to do well

• You must understand the concepts well!
  – If you do not, there is almost no chance of success =(
• If you do understand the concepts, there is very little else to learn!
  – You can do really well =D
• You must do problems. There’s no replacement for this.
  – Discuss problems and solutions in group but always write your own solutions
• Attending lectures is highly advised!
• Don’t postpone learning; you will not be able to “make up” later.
  – Start the homework early on!
• Topics get hard quickly.
• Come to office hours!
• Participate in class!!!