CS244: Theory of Computation

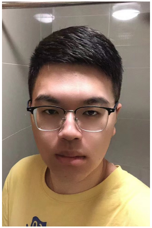
Fu Song
ShanghaiTech University

Fall 2022

Outline

Course Information

- Instructor: Fu Song
  Homepage: `faculty.sist.shanghaitech.edu.cn/faculty/songfu`
  Office: room 1A-504C, SIST Building
  Email: `songfu@shanghaitech.edu.cn`
- TAs: Cunhan You
- HOs: TBD



(a) Cunhan You

Course Information

- Textbook: Introduction to the Theory of Computation (3rd Ed.), Michael Sipser, MIT, 2012
- Discussion, Slides and Homework: PIAZZA (Access code: SISTCS244)
  `https://piazza.com/shanghaitech.edu.cn/fall2022/cs244`
- Preliminaries (optional): algorithms and discrete mathematics
- Grading: Quiz&Discussion 20%, HW (6 sets) 30%, Paper reading&presentation 10%, Midterm 15%, Final exam 25%
- Extra credit of final grades: from 1 point upto 100 points depending upon your technical report (e.g. proposing new useful models and studying decision problems thereof, solving some important or long-stand open problem, etc., 2-3 students per group)

Outline

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

Outline

Automata and Languages

- Introduces models of computation
  - We will study variants of automata and grammars
  - Each model determines what can be expressed, as we will see in Part I of this course
  - Will allow us to become familiar with simple models before we move on to more complex models like a Turing machine
  - Given a model, we can examine computability and complexity

Outline

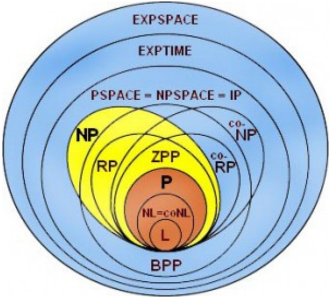
Computability Theory

- A major mathematical discovery in 1930s
  - Certain problems cannot be solved by computers
  - That is, they have no algorithmic solution
- We can ask what a model can and cannot do
  - As it turns out, a simple model of a computer, Turing machine, can do everything that a computer can do
  - So we can use a Turing machine to determine what a computer can and cannot do (i.e., compute)

Outline

Complexity Theory

- How hard is a problem?
- You might already know a lot about this
  - How to determine the time and/or space complexity of most simple algorithms, e.g., Big-O notation
- We take one step forward and study more complexity-classes, e.g., P, NP, PSPACE



Outline

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

About This Course

- 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 2-4 times
  - You should not give up after 10 minutes if you are stumped by a problem
  - at least 5-7 hours per week

Outline

Mathematical Preliminaries

- Mathematical Notations
  - Sets
  - Sequences and Tuples
  - Functions and Relations
  - Graphs
  - Finite and Infinite Words
  - Finite and Infinite Trees
  - Boolean Logic
- Proofs and Types of Proofs

Outline

Sets

- A set is a group of objects, order doesn't matter
  - The objects are called elements or members
  - Examples
    - Finite set: $\{1, 3, 5\}$
    - Infinite set: $\{1, 3, 5, \cdots\}$, or $\{x \mid x \in \mathbb{Z} \wedge x \pmod 2 \neq 0\}$
  - You should know these operators/concepts
    - Subset: $A \subseteq B$ or $A \subset B$
    - Cardinality: Number elements in set ($|A|$) (injective, surjective, bijections)
    - Intersection ($A \cap B$), Union ($A \cup B$), Difference ($A - B$) and Complement ($\overline{A}$)
    - DeMorgan's Laws: $\overline{A \cap B} \equiv \overline{A} \cup \overline{B}$, $\overline{A \cup B} \equiv \overline{A} \cap \overline{B}$
    - Emptyset: $\emptyset$
    - Venn Diagrams: can be used to visualize sets
  - Powersets: All possible subsets of a set
    - E.g. $S = \{a, b, c\}$,
      $2^S = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$
    - In general, what is the cardinality of 2^S? $|2^S| = 2^{|S|}$

Sequences and Tuples

- A sequence is a list of objects, order matters
  - Examples: $(1, 3, 5)$ and $(3, 1, 5)$
- In this course we will use term tuple instead
  - $(1, 3, 5)$ is a 3-tuple
  - a k-tuple has k elements
- Cartesian product (a.k.a. cross project) is an operation on sets but yields a set of tuples
  - Example: if $A = \{1, 2\}$ and $B = \{x, y, z\}$, then
  $$A \times B = \{(1, x), (1, y), (1, z), (2, x), (2, y), (2, z)\}$$
  - If we have k sets $A_1, A_2, \ldots, A_k$, we can take the Cartesian product $A_1 \times A_2 \cdots \times A_k$ which is the set of all k-tuples $(a_1, a_2, \cdots, a_k)$ where $a_i \in A_i$
  - We can take Cartesian product of a set with itself A^k represents
  $$\underbrace{A \times A \times A \cdots \times A}_{k}$$

Functions

- A function maps an input to a (single) output
  - $f(a) = b$, f maps a to b
- The set of possible inputs is the domain and the set of possible outputs is the range
  - $f : D \rightarrow R$
  - D is the domain of f and R is the range of f
- The function $f : D \rightarrow R$ is
  - a total function if $\forall a \in D$: $f(a)$ is defined, otherwise partial function
  - a bijective function if
    - is total
    - $\forall a, a' \in D$, $a \neq a' \rightarrow f(a) \neq f(a')$ (injective)
    - $\forall b \in R . \exists a \in D$ such that $f(a) = b$ (surjection)

Big-O Notation

- Given two total functions $f, g : \mathbb{N} \to \mathbb{N}$
  - $f(n) = O(g(n))$, if $\exists c, d \geq 1$ such that $\forall n \geq d$, $f(n) \leq c \cdot g(n)$
    $g(n)$ is an upper bound for $f(n)$
  - $f(n) = \Omega(g(n))$, if $\exists c, d \geq 1$ such that $\forall n \geq d$, $c \cdot f(n) \geq g(n)$
    $g(n)$ is a lower bound for $f(n)$
  - $f(n) = \Theta(g(n))$, if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$
- $f(n) = \Omega(g(n))$ iff $g(n) = O(f(n))$
- The big-O notation compares the rate of growth of functions rather than their values, so when $f(n) = \Theta(g(n))$, $f(n)$ and $g(n)$ have the same rates of growth, but can be very different in their values.

Relations

- A predicate is a function with range $\{True, False\}$
  - Example: even(4) = True
- A (k-ary) relation is a predicate whose domain is a set of k-tuples $A_1 \times A_2 \times A_3 \cdots \times A_k$
  - If $k = 2$, then binary relation (e.g., $=, <, \cdots$)
  - Can just list what is true (e.g., even(4))
- A (k-ary) relation R can be seen as a set of k-tuples, e.g., $R \subseteq A_1 \times A_2 \times A_3 \cdots \times A_k$,

$$(a_1, \cdots a_k) \in R \text{ iff } R(a_1, \cdots a_k) = True$$

Equivalence Relations

- An equivalence relation R is a binary relation over a domain D satisfying the following three properties:
  - Reflexive: $x\ R\ x$
  - Symmetric: $x\ R\ y$ iff $y\ R\ x$
  - Transitive: if $x\ R\ y$ and $y\ R\ z$, then $x\ R\ z$
  - Try $=, <$

Equivalence Classes

- For every element $x \in D$, an equivalence relation R over a domain D induces an equivalence class:

$$[\![x]\!]_R := \{x' \in D \mid x \; R \; x'\}$$

  - Suppose $R = \{(1,1),(2,2),(1,2),(2,1),(3,3),(4,4),(3,4),(4,3)\}$
  - $[\![1]\!]_R = \{1, 2\}$
  - $[\![3]\!]_R =? \; \{3, 4\}$

- $\forall x, y \in D$, either $[\![x]\!]_R = [\![y]\!]_R$ or $[\![x]\!]_R \cap [\![y]\!]_R = \emptyset$ (Why?)

- A binary relation R over D is a partial order if it is reflexive, transitive, and antisymmetric $(x \; R \; y \wedge y \; R \; x \Rightarrow x = y)$

- A binary relation R over D is a total order (a.k.a. linear order) if it is a partial order and $\forall x, y \in D$, either $x \; R \; y$ or $y \; R \; x$.

Graphs

- A directed graph G is a tuple (V, E), where
  - V is a set of vertices
  - $E \subseteq V \times V$ is a set of edges that are 2-tuples
- A undirected graph G is a tuple (V, E), where
  - V is a set of vertices
  - $E \subseteq \{\{v_1, v_2\} \mid v_1, v_2 \in V\}$ is a set of edges that are 2-sets
- Notations:
  - The degree of a vertex (for undirected graph) is the number of edges touching it, $\mathtt{degree}(v) := |\{v' \in V \mid \{v, v'\} \in E\}|$
  - The in-degree (resp. out-degree) of a vertex (for directed graph), $\mathtt{indegree}(v) := |\{v' \in V \mid (v', v) \in E\}|$ and $\mathtt{outdegree}(v) := |\{v' \in V \mid (v, v') \in E\}|$
  - A path is a sequence of nodes connected by edges
  - A simple path does not repeat nodes
  - A path is a cycle if it starts and ends at same node
  - A simple cycle repeats only first and last node
  - A graph is a unranked tree if it is connected and has no simple cycles

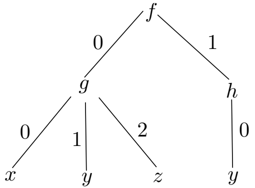
Finite and Infinite Words

- An alphabet Σ is any non-empty finite set
  - Members of the alphabet are (alphabet) symbols (or letters)
  - $\Sigma_1 = \{0, 1\}$
  - $\Sigma_2 = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$
- A finite word (string in textbook) over an alphabet is a finite sequence of symbols from the alphabet
  - 0100 is a string from Σ_1 and cat is a string from Σ_2
  - The length of a string w, $|w|$ is its number of symbols
    - If $|w| = n$, then w can be written as $w_0 w_1 \cdots w_{n-1}$, where $w_i \in \Sigma$
    - The empty string, ε, has length 0
  - Strings can be concatenated,
    - ww' is string w concatenated with string w'
    - A string w can be concatenated with itself k times, denoted by w^k
- A ω-word over an alphabet is an infinite sequence of symbols from the alphabet, e.g., $(01)^\omega$

Finite and Infinite Trees

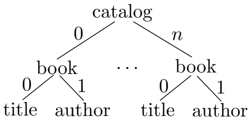
- Finite ranked trees
  - Ranked alphabet Σ: Rank function $rank : \Sigma \to \mathbb{N}$.
  - E.g., every node labeled by σ has $rank(\sigma)$ children
  - Tree domain: A nonempty finite subset D of $\mathbb{N}^*$ such that
    - if $xi \in D$ for some $i \in \mathbb{N}$, then $x \in D$, i.e., $x \in \mathbb{N}^*$
    - if $xi \in D$ for some $i \in \mathbb{N}$ and $x \in D$, then $xj \in D$ for any $j \leq i$.
  - Ranked trees: A Σ-tree is a mapping $t : D \to \Sigma$ such that

$$\forall x \in D, rank(t(x)) = |\max\{i \mid xi \in D\}|.$$

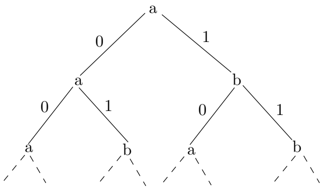


Finite and Infinite Trees: continued

- Finite unranked trees
  - Alphabet Σ is unranked
  - E.g., every node labeled by σ can have an arbitrary number of children
  - Unranked trees: A mapping $t : D \to \Sigma$ (no rank constraints).



- ω-trees (ranked or unranked): a mapping $t : \mathbb{N}^* \to \Sigma$.



$t(\varepsilon) = a$

$t(x0) = a$

$t(x1) = b$

Formal Languages and Closure Properties

- Formal languages

  A set of finite words, ω-words, finite trees, etc.

- Language-theoretical operations
  - Union: $L_1 \cup L_2$,
  - Intersection: $L_1 \cap L_2$,
  - Complementation: $\Sigma^* \setminus L$, $\Sigma^\omega \setminus L$, ...
  - Homomorphism: A mapping $h : \Sigma \to \Pi \cup \{\varepsilon\}$.

Boolean Logic

- Boolean logic is a mathematical system built around True and False or 0 and 1
- Below are the Boolean operators, which can be defined by a truth table

$\wedge$	and/conjunction	$1 \wedge 1 \equiv 1$; else 0
$\vee$	or/disjunctions	$0 \vee 0 \equiv 0$; else 1
$\neg$	not	$\neg 1 \equiv 0$; $\neg 0 \equiv 1$
$\rightarrow$	implication	$1 \rightarrow 0 \equiv 0$; else 1
$\leftrightarrow$	equality/biimplication	$1 \leftrightarrow 1 \equiv 1$; $0 \leftrightarrow 0 \equiv 1$; else 0

- Can prove equality using truth tables, e.g., DeMorgan's law and Distributive law

Outline

Proofs and Types of Proofs

- Proofs are a big part of this class
- A proof is a convincing logical argument
  - Proofs in this class need to be clear, but not very formal
  - The books proofs are often informal, using English, so it isn't just that we are being lazy
- Types of Proofs
  - $A \Leftrightarrow B$ means A if and only if (iff) B
    - Prove $A \Rightarrow B$ and prove $B \Rightarrow A$
  - Disproof by counterexample (prove false via an example)
  - Proof by construction (main proof technique we will use)
  - Proof by contradiction
  - Proof by induction

Proof Example 1

- For any two sets A and B, prove $\overline{A \cap B} \equiv \overline{A} \cup \overline{B}$
- The proof of $\overline{A \cup B} \equiv \overline{A} \cap \overline{B}$ refers to Theorem 0.20, page 20
- What proof technique to use? Any ideas
- Prove in each direction:
  - First prove forward direction, then backward directions, (e.g., show if element x is in one of the sets then it is in the other)
  - We will do in words, as formal as possible formal definitions of each operator

Proof Example 1: Proof ($\Rightarrow$)

- Assume $x \in \overline{A \cap B}$, we show that $x \in \overline{A} \cup \overline{B}$

1. $x \in \overline{A \cap B}$ [Assumption]
2. $\Rightarrow x \notin A \cap B$ [Def. of complement]
3. $\Rightarrow (x \notin A) \vee (x \notin B)$ [Def. of intersection]
4. $\Rightarrow (x \in \overline{A}) \vee (x \in \overline{B})$ [Def. of complement]
5. $\Rightarrow x \in \overline{A} \cup \overline{B}$ [Def. of union]

Proof Example 1: Proof ($\Leftarrow$)

- Assume $x \in \overline{A} \cup \overline{B}$, we show that $x \in \overline{A \cap B}$

1. $x \in \overline{A} \cup \overline{B}$ [Assumption]
2. $\Rightarrow (x \in \overline{A}) \vee (x \in \overline{B})$ [Def. of union]
3. $\Rightarrow (x \notin A) \vee (x \notin B)$ [Def. of complement]
4. $\Rightarrow x \notin A \cap B$ [Def. of intersection]
5. $\Rightarrow x \in \overline{A \cap B}$ [Def. of complement]

Proof Example 2

- Prove or disprove: All prime numbers are odd
- What proof technique to use? Any ideas
- Disproof by counterexample uses three steps:
  1. State false: Not all prime numbers are odd
  2. Give a counterexample: consider the number 2
  3. Explain why your counterexample is a counterexample
     - $2 = 2 \times 1$, so 2 is even
     - 2 has only two factors 2 and 1, so it is prime

Proof Example 3

- Prove for every even number $n > 2$, there is a 3-regular undirected graph with n vertices (Theorem 0.22, page 21)
  - A undirected graph is k-regular if every vertex has degree k
- What proof technique to use? Any ideas
- Proof by construction
  - Many theorems say that a specific type of object exists. One way to prove it exists is by constructing it.
  - May sound weird, but this is by far the most common proof technique we will use in this course
  - We may be asked to show that some property is true. We may need to construct a model which makes it clear that this property is true

Proof Example 3

- Can you construct such a graph for $n = 4, 6, 8$?
  - Try now (Hint: place the vertices into a circle)
  - Can you find a pattern?
  - Generalize the pattern and that is the proof
- Solution
  - Place the vertices in a circle and then connect each node to the ones next to it, which gives us a 2-regular graph
  - Then connect each node to the one opposite it and you are done
  - This is guaranteed to work because if the number of nodes is even, the opposite node will always get hit exactly once
  - Note: if it was odd, this would not work

Proof Example 4

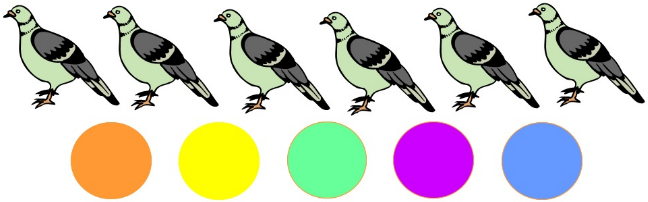
- Prove $\sqrt{2}$ is irrational
- What proof technique to use? Any ideas
- Proof by contradiction uses three steps:
  1. first assume that the statement P is false
  2. then show that leads to a contradiction
  3. therefore, statement P must be true

Proof Example 4

- Prove $\sqrt{2}$ is irrational
- Assume $\sqrt{2}$ is rational
    1. $\sqrt{2}$ is rational
    2. $\Rightarrow \sqrt{2} \equiv \frac{m}{n}$ for some integers m, n. Without loss of generality, we assume that $\frac{m}{n}$ is in lowest terms (i.e., reduced fraction)
    3. $\Rightarrow n \times \sqrt{2} \equiv m$
    4. $\Rightarrow 2 \times n^2 \equiv m^2$
    5. $\Rightarrow m$ is even, let $m = 2 \times k$
    6. $\Rightarrow n^2 \equiv 2 \times k^2$
    7. $\Rightarrow n$ is even, let $n = 2 \times h$
    8. $\Rightarrow \sqrt{2} \equiv \frac{m}{n} \equiv \frac{2 \times k}{2 \times h} \equiv \frac{k}{h}$
    9. $\Rightarrow \frac{m}{n}$ is not in lowest terms, resulting in a contradiction

Discussion

- Pigeonhole principle: prove for every integer n, if $n+1$ objects are put into n boxes, then at least one box must contain 2 or more objects



Proof Example 5

- Prove for every (undirected) graph $G = (V, E)$, the sum of degrees of all vertices is even, i.e., $\sum_{v \in V} \texttt{degree}(v)$ is even
- What proof technique to use? Any ideas
- Proof by induction uses three steps:
  1. Base case(s): one or more particular cases that represent the most basic case (e.g. $|E| = 0$, or $|E| = 0$ and $|E| = 1$)
  2. Induction hypothesis: assumption that we would like to be based on
     - Weak induction: assume the step that you are currently stepping on holds (e.g. let's assume that $|E| = n$ holds)
     - Strong induction: assume the steps that you have stepped on before including the current one holds (e.g. let's assume that $|E| = i$ holds for all $0 < i \leq n$)
  3. Inductive Step: prove that the next step based on the induction hypothesis holds (e.g. $|E| = n + 1$ holds)

Proof Example 5

- Prove for every (undirected) graph $G = (V, E)$, the sum of degrees of all vertices is even, i.e., $\sum_{v \in V} \texttt{degree}(v)$ is even
- Proof by induction uses three steps:
  1. Base case: $|E| = 0 \Rightarrow \sum_{v \in V} \texttt{degree}(v) = 0$, and 0 is even
  2. Induction hypothesis: assume the statement holds when $|E| = n$
  3. Inductive Step: $|E| = n + 1$. When adding an edge into E, it is by definition between two vertices (but can be the same), each vertex then has its degree increase by 1, or 2 overall. Hence, $\sum_{v \in V} \texttt{degree}(v)$ is even.

Outline

Recap

- Mathematical Notation
  - Sets
  - Sequences and Tuples
  - Functions and Relations
  - Graphs
  - Strings and Languages
  - Boolean Logic
- Proofs and Types of Proofs
  - $A \Leftrightarrow B$ means A iff B
    - Prove $A \Rightarrow B$ and prove $B \Rightarrow A$
  - Disproof by counterexample (prove false via an example)
  - Proof by construction (main proof technique we will use)
  - Proof by contradiction
  - Proof by induction