CS 579: Computational Complexity: Lecture 1  26 - 30

today: admin & survey form

main theme and goals

backyard

admin

class: cs 579 computational complexity
TTh 3:30-4:45 1109 Siebel

courses: engr. illinois.edu/cs 579  → pizza link

as: Prof. Michael A. Forbes
mforbes@illinois.edu
Siebel 322D
office hrs: T5, R1, or by appr.

TA: Zander Kelley
awke22@illinois.edu
w4-6 SC 3303

grades: 70% 6 bivendly part: states Thurs
II due lighter 2nd half due to project II
II late policy is online II

- 30% course project
  groups of 2 or 4
  read paper → 30 min presentation

- short report

AS: Introduction to the Theory of Computation, by Sipser
2nd or 3rd ed. ok

- Computability & Complexity by Arora & Barak
  for a second Yr of CS

- course notes & well-sign up II

passing: models of computation: 374 475, II will accept II
I automate, Ms, etc II
- algorithms: 473
- discuss math: 173
- mathematical maturity.

= questions
Cryptography: encryption is everywhere.

Alice \rightarrow \text{ Eve} \rightarrow \text{ Bob}

- Secret key is secret.
- Message $m$ encrypted as $\text{Enc}(m, k)$.

- Decryption: $\text{Dec}(\text{Enc}(m, k), k) = m$

- Security: $\text{Crack}(\text{Enc}(m, k))$ "reveals nothing" about $m$

Crack is efficient and requires common modeling.

Puzzle: crypto requires easy problem? 

- hard problems? 

- which problems are hard? 

This course... why are problems hard?

A : is a hard question

- what is "convincing evidence" that a problem is hard?

- someone on the internet said it was hard

- never really hard and found no algorithm

- also, if specific/narrow form can solve the problem

- similar problems can be proven to be hard

- unconditional mathematical proof

Goals: identify computational problems

- subset problems in graphs
- primality testing
- satisfiability of boolean formulas
- finding important computational resources

Time, Space

ability to solve a given computational problem

Q : does using more of one resource give more computational power? pattern

- how do different types of resources compare? size vs. power?

- problem vs. resources? size comparison?
This course: - structural complexity (3/4)
  - theory of Turing machines, different resources
  - few unconditional results
  - conjectures (1/4)
  - theory of finite computational models, e.g., circuits
  - more unconditional results

Q: A language is a set \( L \subseteq \Sigma^* \) over an alphabet \( \Sigma \), often \( \{0, 1\} \).

A: Given \( x \in \Sigma^* \), is \( x \in L \)?

- Formal proof concept.
- Model capturing real-world phenomena.

Formal model of computation:

- **DFA** (Deterministic Finite Automaton): a 5-tuple \((Q, \Sigma, \delta, q_0, F)\) where
  - \( Q \) is a finite set of states
  - \( \Sigma \) is a finite set of input symbols
  - \( \delta : Q \times \Sigma \rightarrow Q \) is the transition function
  - \( q_0 \in Q \) is the start state
  - \( F \subseteq Q \) is the set of accepting states

A DFA accepts \( x \in \Sigma^* \) if \( x \) is of the form \( q_0 \xrightarrow{\epsilon} q_f \) for some \( q_f \in F \).

- Regular language: a language that can be accepted by a DFA.

- Regular languages are closed under union, intersection, complementation, and concatenation.

**Fact:** For any integer \( n \geq 0 \), \( 0^n1^n \) is not regular. This is amazing!}

- DFA's well understood
- Real (vague) models
def. a Turing Machine (TM) is \[ \begin{align*} & \text{finite control} \quad \text{I. infinite tape} \quad \text{II. infinite tape} \end{align*} \]

formally:

- \( \mathcal{Q} \): set of states \{ 0, \text{start}, \text{reject}, \text{accept} \}
- \( \Gamma \): tape alphabet
- \( \delta \): transition function
- \( \text{start} \in \mathcal{Q} \)

TM computes by:
- tape initialized to \( x \in \{ \text{blank} \}^k \)
- head placed at start of tape
- move \( \epsilon \) until reach 0, accept or reject

language \( L(M) \): \( M \) on input accepts \( \text{true} \) if \( x \in L \)
- reject if \( \text{false} \)

Fact: exists TM \( M \) s.t. \( L(M) = \{ \text{odd} \}^* \) — example II
- any language of any known programming language, is also the language of some TM

Church-Turing thesis:

Fact: \( L = \{ \text{true} | \exists A \in \text{TM} \text{ s.t. } \text{does not halt on input } x \} \)
- the language of a TM
- is undecidable if even TMs are bounded

Q: what can TMs do efficiently?

next time: time complexity