Michael A. Forbes
miforbes@illinois.edu
2024-08-27 / cs579

# cs579 Computational Complexity : Lecture 1 (2024-08-27)

today : - logistics
- motivation and goals
- background
[[ move to front ]]

class : cs579 Computational Complexity
TR 15:30 - 16:45, Siebel 1302
courses.engr.illinois.edu/cs579/fa2024

instructor : Prof. Michael A. Forbes
miforbes@illinois.edu
[[ office hours ]] [[ after class ]]
Siebel 3220 · T/T, 10-6 by appt
[[ if tighter in 2nd half, to accommodate projects ]]

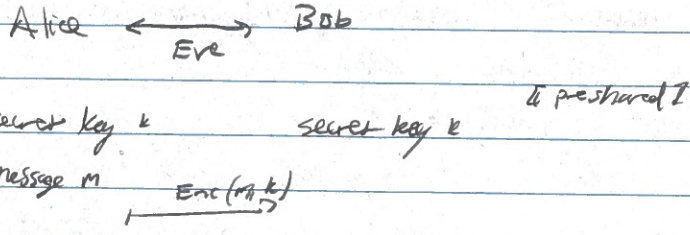
grades : - psets (70%) - biweekly [[ late policy online ]]
- project (30%) - groups of size 2 to 6 [[ ε-ranking ]]
- paper + you - 30 min presentation [[ end of semester ]]

ref : - "Introduction to the Theory of Computation", Sipser (2nd ed.) 3rd edition
- first half of course [[ very well written ]] [[ please read ]]
- "Computational Complexity: A Modern Approach", Arora, Barak
- second half of course
- course notes [[ scanned ]] [[ illegible ]] [[ please ask if not clear ]]

prereq : - discrete math cs173
- models of computation cs374, cs475 [[ overlap ]]
- algorithms cs473 [[ CIT ]]
- mathematical maturity

= [[ Q ]]

Q : why are we here? [[ I.e., why I am here ]]
A : cryptography [[ encryption is everywhere ]]



Michael A. Forbes
miforbes@illinois.edu
2024-08-27.2
CS579

want: decryption: $Dec(Enc(m,k),k)=m$ [efficient also]
encryption: $Create(Enc(m,k))$ "reveals nothing" [modeling] [observe]
about $m$ if $Create$ is efficient also [lack of efficient also]
=> crypto requires - easy problems
- hard problems
related
[all problems easy?]

Q: Do we have hard problems?
find a hard problem? [explicit vs existence]
A: are important problems easy or hard? [only some problems are relevant]
↳ are hard questions [hardness-start somewhere]

Q: what is "convincing evidence" a problem is hard?
[Q]
someone on the internet [Chat GPT] said it was hard
we thought for a long time, and found no efficient algo [Lovász]
no algo of specific/natural form can efficiently solve problem
relate to other "seemingly hard" problems
"similar" problems can be proven to be hard
unconditional mathematical part of hardness
[also this course: non-existence]

goals: identify important computational problems
- shortest paths in graphs
- primality testing
- satisfiability of boolean formula
identify important computational problems
- time
- space
- ability to solve a specific computational problem

Q: using more of a resource yields more computational power? [caffeine → sharp]
compare different resources? [sleep vs caffeine for theory] [turning into caffeine]
problems vs resources?

this course: - structural complexity (~3/4)
- theory of Turing machines, different resources
- few unconditional hardness results
- concrete complexity (~1/4)

· vs 2023
· vs 2023
· vs 2023

Michael A. Forbes
miforbes@illinois.edu
2024-08-27.3
cs579

- theory of finite computational models, eg circuits
- "more" unconditional hardness results

= [[ Q ]]
[[ background ]]

def: a language is set $L \subseteq \Sigma^*$ ← all finite strings over $\Sigma$
[[ finite ]]
↑ alphabet often $\{0,1\}$
[[ prototypical question, often captures variants ]]

Q: given $x \in \Sigma^*$, is $x \in L$?
[[ decision problem ]]

[[ model ]]
def: a deterministic finite automaton (DFA) is a 5-tuple
$M = (Q, \Sigma, \delta, q_0, F)$ where:
- $Q$ finite set of states
- $\Sigma$ alphabet
- $\delta: Q \times \Sigma \to Q$ transition function
[[ single ]] [[ multiple ]]
- $q_0 \in Q$ start state
- $F \subseteq Q$ accept states

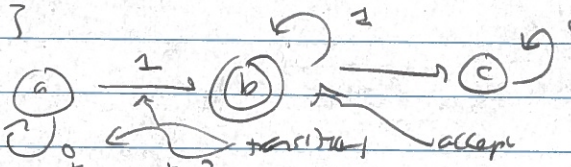
cs 456
vs 2023
$|F| = 1$ : a vsA

a DFA accepts $x \in \Sigma^*$ if - start on $q_0$
- at $i$th step, transition $q \mapsto q' = \delta(q, x_i)$ ← $i$th symbol
- after $n$th step, accept if $q \in F$ [[ $n = |x|$ conversion ]] reject else

denote $L(M) = \{x : M \text{ acc } x\}$, the language of $M$

def a language $L$ is regular if $L = L(M)$ for some DFA $M$.
vs 2023.

eg: $\Sigma = \{0,1\}$

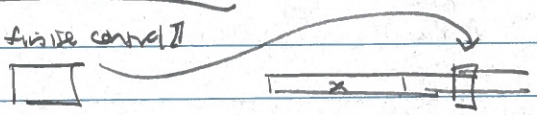



$L(M) = \{0^* 1 1^*\}$ is regular
↑ includes empty string

vs 2024.
face [cs 374] - $\{0^n 1^n : n \in \mathbb{N}\}$ is [not] regular
[[ amazing! ]]
[[ DFAs well understood ]]
[[ real storage model ]]

Michael A. Forbes
M.forbes@illinois.edu
2024-08-27-4
cs 579

def: a Turing Machine (TM) is a [DFA w/ tape for storage] [def by picture]




formally: - $Q$ set of states, $\{q_{start}, q_{acc}, q_{rej}\} \subseteq Q$
- $\Gamma$ tape alphabet, $\Gamma \supseteq \Sigma$
- $\delta : \Gamma \times Q \to \{L, S, R\} \times \Gamma \times Q$
  (current symbol, current state; tape head movement; write new symbol; new state)

· cs vs δ
· vs 2023

TM computes by - tape initialized as $x \sqcup \sqcup \cdots$ ($\sqcup \in \Gamma \setminus \Sigma$ [blank])
- head placed at start of tape
- iterate $\delta$ until reach $q_{acc}$ or $q_{rej}$ [halts]

· vs 2023

the language $L$ of TM $M$ is: $M$ on $x$ reached $q_{acc} \Rightarrow x \in L$
$q_{rej} \Rightarrow x \notin L$
[rely] halt $\Rightarrow x \notin L$

· repeat warning

[TM >> DFA]

fact: - exists TM $M$ w/ $L(M) = \{0^n 1^n : n \in \mathbb{N}\}$

· vs 2023
- any language of any [known] realistic computational process [TM ≥ CPU] is also the language of some TM

· vs 2023
Church-Turing thesis: [possible] [philosophy]

fact [cs 374]: $L = \{\langle M, x\rangle$ (encoding) : $M$ does not halt on $x\}$
($M$: TM, $x$: input)
is [not] the language of any TM that [always] halts.
$L$ is undecidable

Q: what can TMs do [efficiently]?

today: - logistics
- motivation and goals
- background

next lecture: time complexity