Michael Forbes
mforbes@illinois.edu
↦ 2019-01-15.1
2019-01-15.2 ← cs 579

cs 579: Computational Complexity: Lecture 1   26 – 30

today: admin [survey form]
motivation and goals
background

admin

class: cs 579 Computational Complexity
TR 3:30-4:45 1109 Siebel
courses.engr.illinois.edu/cs579 ← piazza link

me: Prof. Michael A. Forbes
mforbes@illinois.edu
Siebel 3220
office hrs: T5, R1, or by appt

TA: Zander Kelley
awk2@illinois.edu
W 4-6 SC 3303

grades: - 75% 6 biweekly psets, starts Thurs
[get lighter in 2nd half due to project]
[late policy is online]
- 30% course project
groups of 2+ε [ε = rounding]
read paper ⟼ - 30 min presentation
- short report

refs: - "Introduction to the Theory of Computation", by Sipser
2nd or 3rd edition ok
for ~ first ½ of course
- "Computational Complexity" by Arora-Barak
for ~ second ½ of course
- course notes [~ illegible]

prereq: models of computation: 374 475, ... [will overlap]
[automata, TMs, etc]
- algorithms: 473
- discrete math: 173
- Mathematical maturity

⇒ Questions

Michael Forbes
miforbes@illinois.edu
2019-01-15.2 ← 2019-01-15.1
→ 2019-01-15.3
cs579

motivation & goals

cryptography [encryption is everywhere]

Alice ⟷ Bob
Eve

secret key k — secret key k

message m $\xrightarrow{Enc(m,k)}$

want: [decrypt] $Dec(Enc(m,k),k) = m$ [algo]

[security] Crack(Enc(m,k)) "reveals nothing" about m

if Crack is efficient algo ← requires correct modeling

[lack of algo]

punchline: crypto requires - easy problems] of a structured form
- hard problems

Q: are there hard problems?
which problems are hard?] this course
why are problems hard?

A: → is a hard question

Q: what is "convincing evidence" that a problem is hard?
someone on the internet said it was hard
we tried really hard and found no algorithm
no algo of specific/natural form can solve the problem
"similar" problems can be proven to be hard
unconditional mathematical proof of hardness
in a reasonable model
] also this course

goals: identify important computational problems
shortest paths in graphs
primality testing
satisfiability of boolean formula

identify important computational resources
time
space
ability to solve a given computational problem

Q: does using more of a resource give more computational power? [catterns]
how do different types of resources compare? [sleep vs catterns]
problems vs resources? [catterns vs theorems]

Michael Forbes
Mforbes@illinois.edu

this course: - structural complexity (3/4)
- theory of Turing machines, different resources
- few unconditional results
- concrete complexity (1/4)
- theory of finite computational models, eg circuits
- "more" unconditional results

⟹ questions

background

def: a <u>language</u> is a set $L \subseteq \underbrace{\Sigma}^*$

← alphabet, often $\{0,1\}$

Q: given $x \in \Sigma^*$, is $x \in L$? [prototypical computational question capturing most other variants]

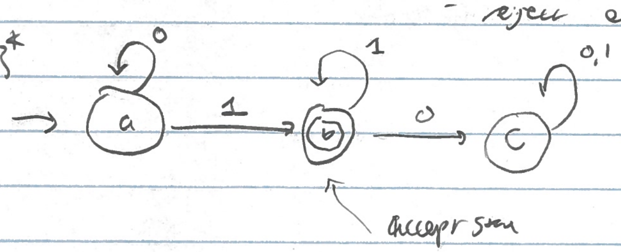
[need model of computation

def a <u>deterministic finite automaton</u> (DFA) is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where
- Q is finite set of <u>states</u>
- Σ is finite set called <u>alphabet</u>
- $\delta: Q \times \Sigma \to Q$ is <u>transition function</u>
- $q_0 \in Q$ <u>start state</u>
- $F \subseteq Q$ <u>accept state</u>

A DFA <u>accepts</u> $x \in \Sigma^*$ if
- start at q_0 and x_1
- at ith step transition $q \longmapsto q' = \delta(q, x_i)$
- at nth step $n = |x|$ [is a computer]
  - accept if $q \in F$
  - reject else

$L(M) = \{x \mid M \text{ acc } x\}$ ← <u>regular</u> language

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



$L(M) = \{0^* 1 1^*\}$ is regular

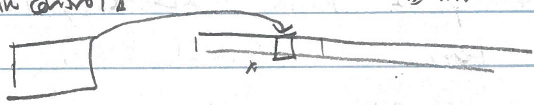
Fact: $\{0^n 1^n : n \in \mathbb{N}\}$ is <u>not</u> regular [this is amazing]

[DFA's well understood]

[need stronger model]

Michael Forbes
mforbes@illinois.edu
2019-01-15.4 ← 2019-01-15.3
→ 2019-01-17.1
cs579

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



formally: Q set of states $\{q_{start}, q_{rej}, q_{acc}\}$
Γ tape alphabet
δ transition function
δ(current symbol, state) → (head movement, new symbol, new state)

TM computes by: - tape initialized to $x \sqcup^*$ ← blank symbol $\sqcup \in \Gamma$
- head placed at start of tape
- iterate δ until reach q_{acc} or q_{rej}

language L of TM M: M on x reaches q_{acc} ⇒ $x \in L$
" q_{rej} $\notin$
never reaches ⇒ $x \notin L$ [doesn't halt]

Fact: - exists TM M w/ $L(M) = \{0^n 1^n\}$ [example]
- any language of any known programming language, is also the language of some TM
↳ possible

Church-Turing thesis:

Fact: $L = \{\langle A, x \rangle \mid A$ does not halt when running on $x\}$ (A: TM; x: input)
language of a TM
↳ is undecidable [even TM's are limited]

Q: what can TM's do **efficiently**?

next time: time complexity