Michael Forbes
miforbes@illinois.edu

2018-01-17.2 ← cs 579 ↦ 2018-01-17.1

cs 579 : Computational Complexity : Lecture 1

today : admin ‖ survey form
motivation
computational resources

admin

class: cs 579 Computational Complexity
MW 3:30 - 4:45 1105 Siebel
courses.engr.illinois.edu/cs579 ← piazza link

me: Prof. Michael A. Forbes
miforbes@illinois.edu
Siebel 3220
office hrs: M2, W1, or by appt

TA: Robert Andrews
rgandre2@illinois.edu
office hrs: F2-4, lounge between 3304, 3232

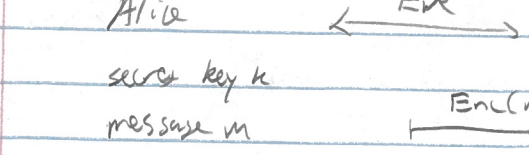
grades: 70%: 6 biweekly psets, starts monday ‖ easier in 2nd half ‖
30%: course project ‖ details online ‖ late policy online ‖
groups of 2+ε
read paper ↦ 30 min present
short report

ref: "Computational Complexity" by Arora-Barak ‖ $10 copy online ‖
course notes ‖ illegible handwriting ‖

prereq: algorithms
models of computation
discrete math ...
‖ mathematical maturity ‖

= Questions?

cryptography
‖ encryption is everywhere ‖



want: ‖ decrypt ‖ $Dec(Enc(m,k),k) = m$ ‖ algorithm ‖
‖ security ‖ $Crack(Enc(m,k))$ "reveals nothing" about m
if Crack efficient ‖ lack of algo ‖

Michael Forbes
miforbes@illinois.edu
2018-01-17.2 ← 2018-01-17.1
→ 2018-01-17.3
CS 579

punchline: Crypto requires - easy problems
- hard problems
} of a structured form

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

perspectives:

problem-centric view
1) identify important problems - matching in graphs
- breaking crypto
- halting problem
- satisfiability
2) establish exact complexity of these problems
ie "on modern computers it takes $\geq 2^{128}$ years to break AES"
[this is too hard]

resource-centric view
identify important computational resources
pro: hierarchy theorems: more resources more power
simulation " : convert one resource to another
completeness " : find problems which exemplify a resource
lower bounds " : find problems which require lots of resources
barrier " : prove that we can't prove

= questions

computational resources
time [most important]
P = { computational tasks where n-bit inputs solved in poly(n) steps } [same polynomial]
= efficient algo "in theory"
[will do TM formalism next time]
Q: what problems are in P? not in P?

space
L = { [will define later] problems which don't require lots of RAM } " $O(\log n)$ bits of memory]
[eg L+C]
thm: $L \subseteq P$ [simulate
Q: $L = P$?

Michael Forbes
MForbes@illinois.edu

2018-01-17.2 → 2018-01-17.3 ← 2018-01-17.4

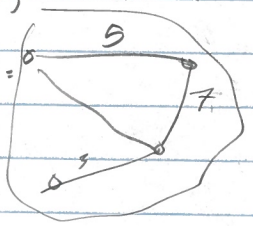
CS 579

nondeterminism

NP = { " verified in poly(n) steps }

eg traveling salesman problem (TSP)

graph $G=(V,E)$:



want: visit all nodes cheaply

[too many solutions to enumerate]

can verify given tour - visits all nodes - is cheap

Q: P = NP? [prototypical question of field, Millennium prize, lots of money, fame]

randomness

BPP = { " in poly(n) steps, using randomness }

eg: election polling

Q: P = BPP?

parallelism

NC = { " in poly(log(n)) wall-clock time and poly(n) work }

[double # processors, double time?]

Q P vs NC?

non uniformity:

P/poly = { " on poly(n) size computer }

[double length of computer code, go twice as fast?]

Q P vs P/poly?

interactions:

NP = TSP $G=(V,E)$ visit nodes cheaply?

Prover (all powerful) → path → Verifier (computationally limited)

easily verify cheap tour

PH: Angel ⟷ Merlin ⟷ Demon

wants "yes" ⟷ ⟷ wants "no"

O(1) # rounds

Q: P vs NP vs PH?

Michael Forbes
MiForbes@illinois.edu
2018-01-17.4 ← → 2018-01-17.3
cs579

Knowledge:
- (prover gives cheap NP ⇒ proof cheap for exists)
- Q: Prover convinces Verifier ↗ but "reveals nothing" about how? (zero knowledge)
- ∃ password example ∀

Quantum:
- ∀ quantum mechanics is very successful physical theory ∀ (∃ existing computer ≠ classical)
- Q: computationally exploit physics? P vs BQP? BQP vs NP?
- ∃ what it can't do ∀

punchline: these questions are all open. ∃ but we know some stuff ∀

next time: Turing machine recap
time complexity
time hierarchy theorem