6.892	Class 1	Feb. 6, 2019

6.892: Algorithmic Lower Bounds / Fun with Hardness Proofs

"Hardness made Easy"

Prof. Erik Demaine
TAs: Jeffrey Bosboom & Jayson Lynch
http://courses.csail.mit.edu/6.892/spring19/

What is this class?

- practical guide to proving computational problems are formally hard / intractable
- NOT a complexity course
  (but we will use/refer to needed results)
- (anti) algorithmic perspective

Why take this class?

- know your limits in algorithmic design
- master techniques for proving hardness — key problems, proof styles, gadgets
- cool connections between problems
- fun problems like Mario & Tetris (& serious problems)
- solve puzzles → publishable papers
- collaborative research / problem solving
  Fall 2014 has led to 12 papers so far!

Background: algorithms, asymptotics, combinatorics

- no complexity background needed
  (but also little overlap with a complexity class)

Topics: (on handout/webpage)

- NP-completeness (3SAT, 3-partition, Hamiltonicity, geometry)
- PSPACE, EXPTIME, ...
- Games, Puzzles, & Computation (Constraint Logic, Sudoku, Nintendo, Tetris, Rush Hour, Chess, Go, ...)
- 0, 1, 2 player/team games
- counting (#P) & uniqueness (ASP)
- undecidability & P-completeness (parallelism)
- inapproximability (PCP, APX, Set Cover, ind. set, UGC, ...)
- fixed-parameter intractability (W, clique, ...)
- optional: economic game theory (PPAD), 3SUM (toward n^2)

Recommended texts: Games, Puzzles, & Computation [Hearn & Demaine 2009]
Garey & Johnson [1979]

Typical complexity of puzzles & games:

	0 players	1 player	2 players	2 teams + hidden info.
bounded (poly.)	P	NP	PSPACE	NEXPTIME
unbounded (exp.)	PSPACE	PSPACE	EXPTIME	R / undecidable

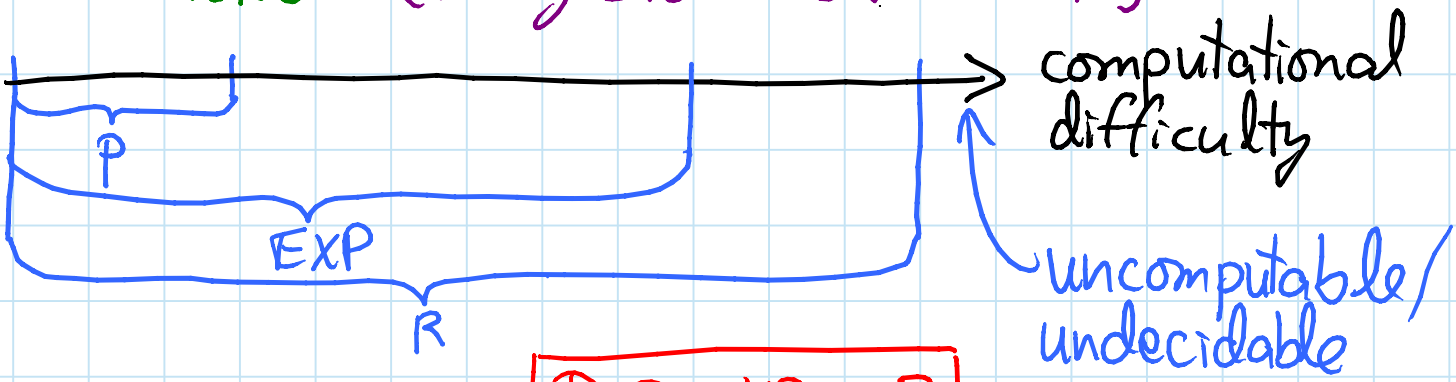
Intro to complexity: CRASH COURSE

$P = \{$problems solvable in polynomial time$\}$ ($\to n^c$)

$EXP = \{$problems solvable in exponential time$\}$ ($\to 2^{n^c}$)

$R = \{$problems solvable in finite time$\}$

↳ "recursive" [Turing 1936; Church 1941]



$$P \subsetneq EXP \subsetneq R$$

Examples:

- negative-weight cycle detection $\in P$
- $n \times n$ Chess $\in EXP$ but $\notin P$
  - ↳ who wins from given board config.?
- Tetris $\in EXP$ but don't know whether $\in P$
  - ↳ survive given pieces from given board
- halting problem $\notin R$
- "most" decision problems $\notin R$
  (# algorithms $\approx \mathbb{N}$; # dec. problems $\approx 2^{\mathbb{N}} = \mathbb{R}$)

NP = {decision problems solvable in poly. time via a "lucky" algorithm}

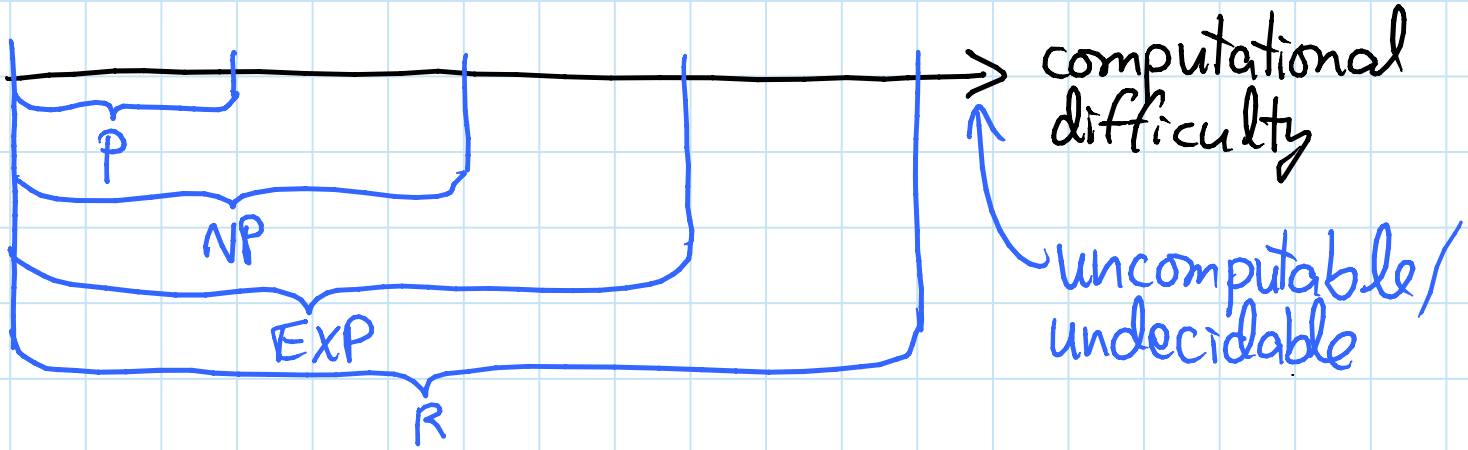
↗ answer ∈ {YES, NO}

↳ can make lucky guesses, always "right", without trying all options

- nondeterministic model: algorithm makes guesses & then says YES or NO
- guesses guaranteed to lead to YES outcome if possible (NO otherwise)

= {decision problems with solutions that can be "checked" in polynomial time}

- when answer = YES, can "prove" it & poly.-time algorithm can check proof



Example: Tetris ∈ NP

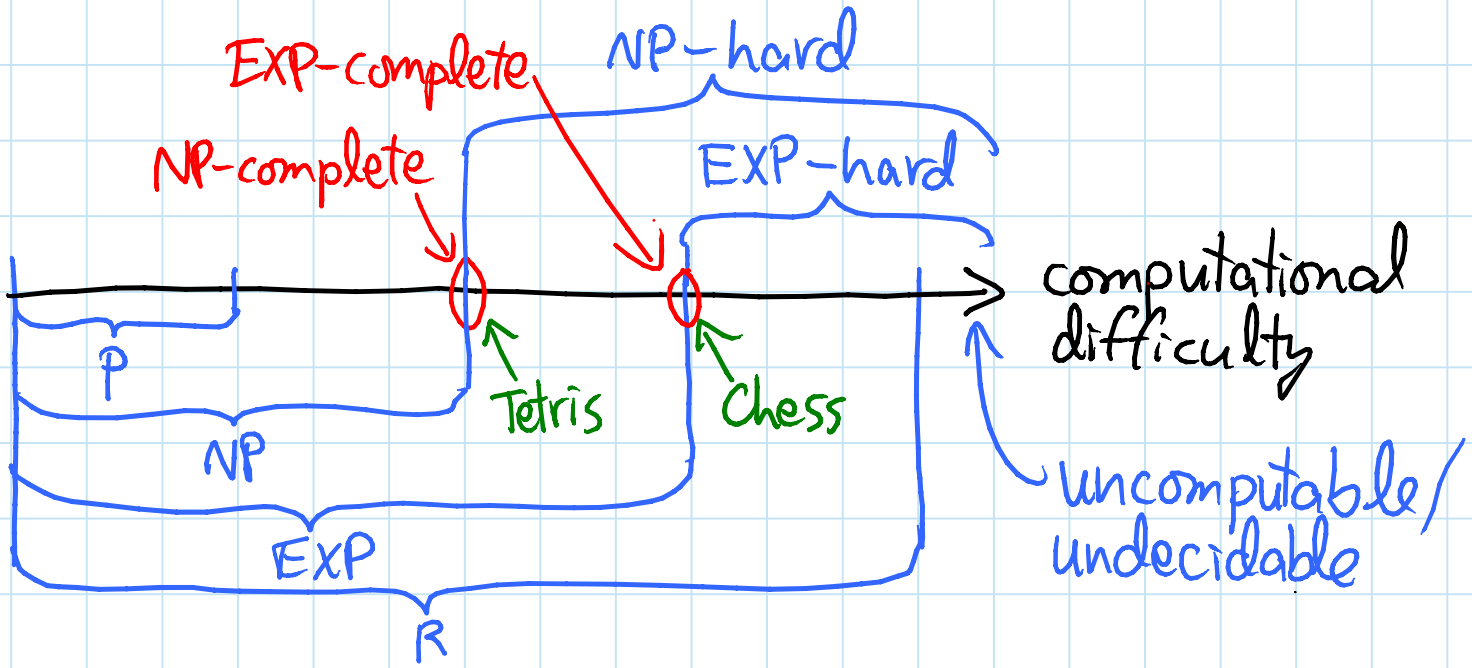
- nondeterministic alg: - guess each move
  - did I survive?
- proof of YES: list what moves to make (rules of Tetris are easy)

P≠NP: big conjecture (worth $1,000,000)
≈ can't engineer luck
≈ generating (proofs of) solutions can be harder than checking them

CoNP = negations (YES ↔ NO) of problems ∈ NP
= problems with good proofs of NO answer

X-hard = "as hard as" (defined later) every problem ∈ X (NP, EXP, etc.)
X-complete = X-hard ∩ X
sometimes "X-easy" = ∈X

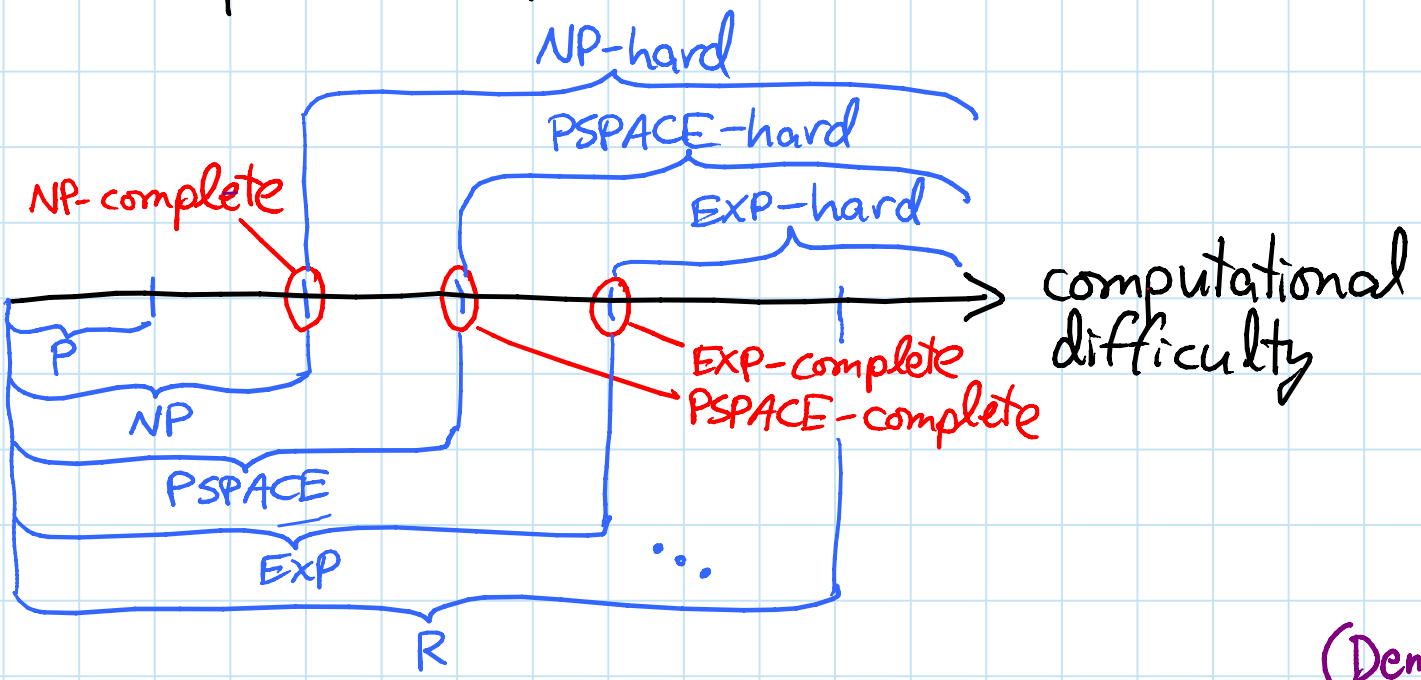
e.g. Tetris is NP-complete [L3]
[Breukelaar, Demaine, Hohenberger, Hoogeboom, Kosters, Liben-Nowell 2004]
⇒ if P ≠ NP, then Tetris ∈ NP \ P



e.g. Chess is EXP-complete ⇒ ∉ P
⇒ Chess ∈ EXP \ NP if NP ≠ EXP (also open)

PSPACE = {problems solvable in polynomial space}

- $\subseteq$ EXP: only exponentially many states
- $\supseteq$ NP: simulate all executions, take running OR
- open whether either is strict



e.g. Super Mario Bros. is PSPACE-complete {Demaine, Viglietta, Williams 2016}
$\Rightarrow \notin P$ if $P \neq NP$ or $NP \neq PSPACE$

Beyond exponential: (not too important)

$EXP(TIME) \subseteq EXPSPACE \subseteq 2EXP(TIME) \subseteq 2EXPSPACE \subseteq \cdots$

double exponential: $2^{2^{n^c}}$

Also $L = LOGSPACE \rightarrow O(\lg n)$ bits of space!

$EXP \subsetneq 2EXP \subsetneq \cdots$ ← time & space hierarchy theorems

$L \subsetneq PSPACE \subsetneq EXPSPACE \subsetneq 2EXPSPACE \subsetneq \cdots$ ↵

Nondeterministic:

- NPSPACE = PSPACE [Savitch 1970] (very useful!) ↗ in general, space bound squares
- NEXP, N2EXP, ...: analogs of NP

What does "as hard as" mean?

poly. size ("blowup")

Reduction from A to B = poly-time algorithm to convert: instance of A → instance of B (other constraints possible)

such that solution to A = solution to B

⇒ if can solve B then can solve A

$B \in P$, $B \in NP$, ⋮ ⇒ $A \in P$, $A \in NP$, ⋮

⇒ B is at least as hard as A

(A is a special case of B)

if A→B then
A is X-hard
⇓
B is X-hard

- this is a "one-call" reduction [Karp]
- "multi-call" reduction [Turing] also possible: solve A using an oracle that solves B
  - doesn't help much for problems we consider

Examples from algorithms:

- unweighted shortest paths → weighted ($w=1$)
- min-product path → min-sum path ($\lg$)
- longest path → shortest path (negate)
- min-weight k-step path → min-weight path
  (k copies of graph + links between adj. layers)

Almost all hardness proofs are by reduction from known hard problem to your problem

Exponential Time Hypothesis: (ETH) [Impagliazzo & Paturi - CCC 1999]

no $2^{o(n)}$-time algorithm for 3SAT

↳ formula size or # variables (see L20)

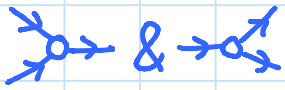
- concrete $P \neq NP$ conjecture relating to EXPTIME

- if reduction from 3SAT → problem B of size blowup n^c then ETH ⇒ no $2^{o(n^{1/c})}$-time alg. for B

Example source problem:

Hamiltonicity: (from L8)

- path/cycle is Hamiltonian if it visits every vertex exactly once

- NP-complete to decide whether there's a Hamiltonian path in following graph types:
  1. grid graph 
  2. 3-regular directed graphs 

- ETH ⇒ no $2^{o(|V|)}$-time algorithm for ②
  & no $2^{o(\sqrt{|V|})}$-time algorithm for ①
  & these are tight [L20]

Examples of hardness proofs:

100% speedrunning video games: [≈Forišek - FUN 2010] (see L8)

- collecting all n objects in a maze environment (possibly with specified start and/or finish) in the minimum possible time is NP-complete
- proof: reduction from Hamiltonicity (1)
  - grid graph → maze (edge → cooridor, vertex → branch)
  - target time = $n \cdot$ vertex time $+ (n-1) \cdot$ edge time
    (↳ need to be consistent)
- linear blowup assuming $O(1)$-size vertex/edge gadgets
  $\Rightarrow 2^{o(\sqrt{n})}$ time impossible assuming ETH

Applications: NP-hard to 100% speedrun:
- Mario (collecting all coins or red coins)

Applications: NP-hard to speedrun:
- Zelda 2 dungeons (collect all keys, doors @ end) & every Zelda game with "small keys"
- Metroidvania & most Zelda games & RPGs (collect all powerups, need all at end)

Applications: NP-hard to win
- Katamari Damacy (collecting all objects within the time limit)

Examples of hardness proofs:

1×n edge matching: [Bosboom, Demaine, Demaine, Hesterberg, Manurangsi, Yodpinyanee – JIP 2017]

6.890 2014

- reduction from Hamiltonicity ✓
- one tile per vertex & one tile per edge

s

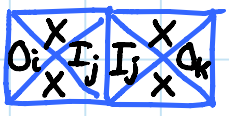
v_i

t

(v_i, v_j) unused

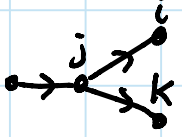
EXCEPT into s & out of t

- for signed edge matching (same colors, opposite signs), 180° rotation is impossible ⇒ easy to argue
- for unsigned edge matching (same colors)
  - s must be at one end (A unique, U_1 twice)
    - say left end
  - vertex tiles can rotate 0 or 180° (U_i twice)
  - right of t tile must be unused edges
  ⇒ left of t tile must have all vertices
  - vertex tiles can't be adjacent ($I_i \neq I_j$, $O_j \neq O_i$)
  ⇒ ≥1 edge tile in between
  - 2 edges in a row ⇒ miss a vertex



can't enter j

→ not contradiction for s but we omit edges → s



can't leave j

→ not contradiction for t but we omit edges t →

⇒ s, (edge, vertex)*, edge, t ← Hamiltonian path

Class format:

- video lectures + handwritten & scribe notes (from 2014)
  - DEMO: note sync & jump, playback speed
  - completion & scribe feedback form DEMO
    DUE BY NOON ON WEDNESDAY
  - questions on Coauthor DEMO
- class (W 7-9:30) for every ≈2 lectures REQUIRED ATTENDANCE
  - new/additional material
  - group puzzle solving "SOLVED"
    ↳ learn material, collaboration skills, warmup for:
  - attack open problems "OPEN"
    ↳ build research skills, thrill of unknown, fun & challenge of advancing frontiers of research
  - implement/visualize reductions "CODING"
- optional session (R 7-9) for open problems etc.
- Coauthor software to coordinate in/outside class
  MUST POST/BE @MENTIONED EACH WEEK DEMO STATS
- weekly psets DUE TUESDAYS AT NOON DEMO PS1
- scribe notes: revise/improve old notes including students' feedback
  (DUE TUESDAY NOON FOLLOWING CLASS)
- project & presentation
  - pose and/or try to solve open problem (e.g. from class)
  - implement reductions or exhaustive search(es)
  - survey a subfield (not well-covered in lectures)
  - Wikipedia (write/improve several articles)
  - art: visualization, sculpture, game design, ...