6.5440	Class 1	Sept. 6, 2023

6.5440: Algorithmic Lower Bounds / (6.5954) Fun with Hardness Proofs — "Hardness made Easy"

Prof. Erik Demaine

TAs: Josh Brunner, Lily Chung, Jenny Diomidova

http://courses.csail.mit.edu/6.5440/fall23/

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
  past offerings have led to 33 papers & 8 theses so far!

Background: algorithms, asymptotics, combinatorics
- no complexity background needed
  (but also little overlap with a complexity class)

Topics:

- NP-completeness (3SAT, 3-partition, Hamiltonicity, geometry)
- PSPACE, EXPTIME, ...
- Games, Puzzles, & Computation (Constraint Logic, Sudoko, 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:

→ free: hardness.mit.edu

- Computational Intractability [Demaine, Gasarch, Hajiaghayi 2024]
- Games, Puzzles, and Computation [Hearn & Demaine 2009]
- Garey & Johnson [1979]

↳ ebook from MIT Libraries

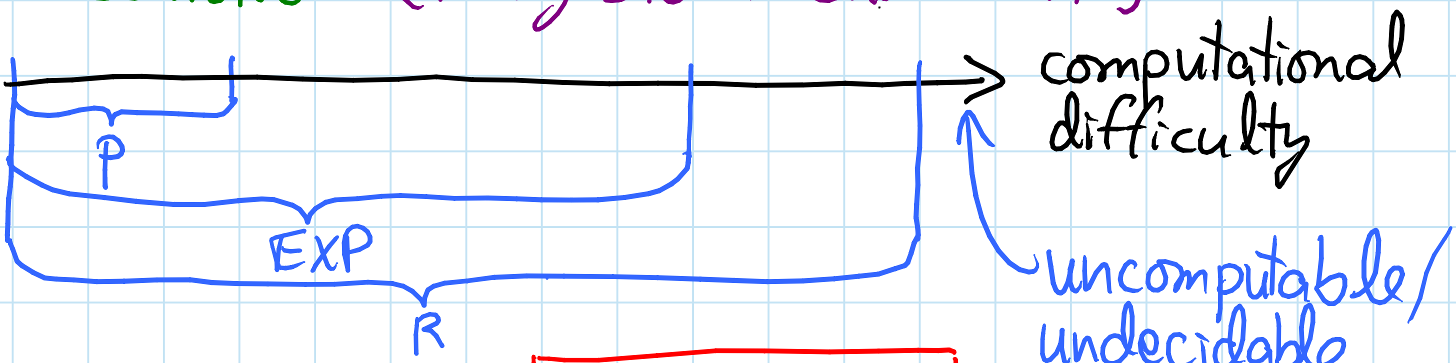
Typical complexity of puzzles & games:

	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]



$$\boxed{P \subsetneq EXP \subsetneq R}$$

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 $\notin R$
  ↳ does this code ever finish?
- "most" decision problems $\notin R$
  (# algorithms $\approx \mathbb{N}$; # dec. problems $\approx 2^{\mathbb{N}} = \mathbb{R}$)

answer ∈ {YES, NO}

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

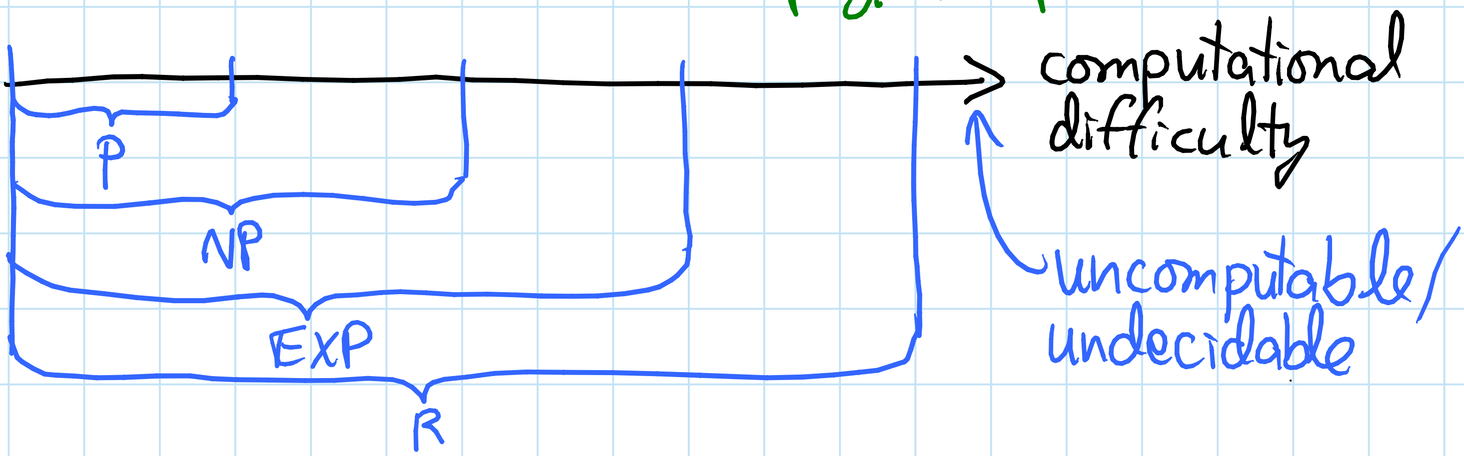
can make lucky guesses, always "right", without trying all options

- 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

⇒ poly.-size 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)

$P \neq NP$: big conjecture (worth \$1,000,000)

$\approx$ can't engineer luck

$\approx$ generating (proofs of) solutions can be harder than checking them

CoNP = negations (YES $\leftrightarrow$ NO) of problems $\in$ NP

= problems with good proofs of NO answer

NP, EXP, etc.

defined later

X-hard = "as hard as" every problem $\in$ X

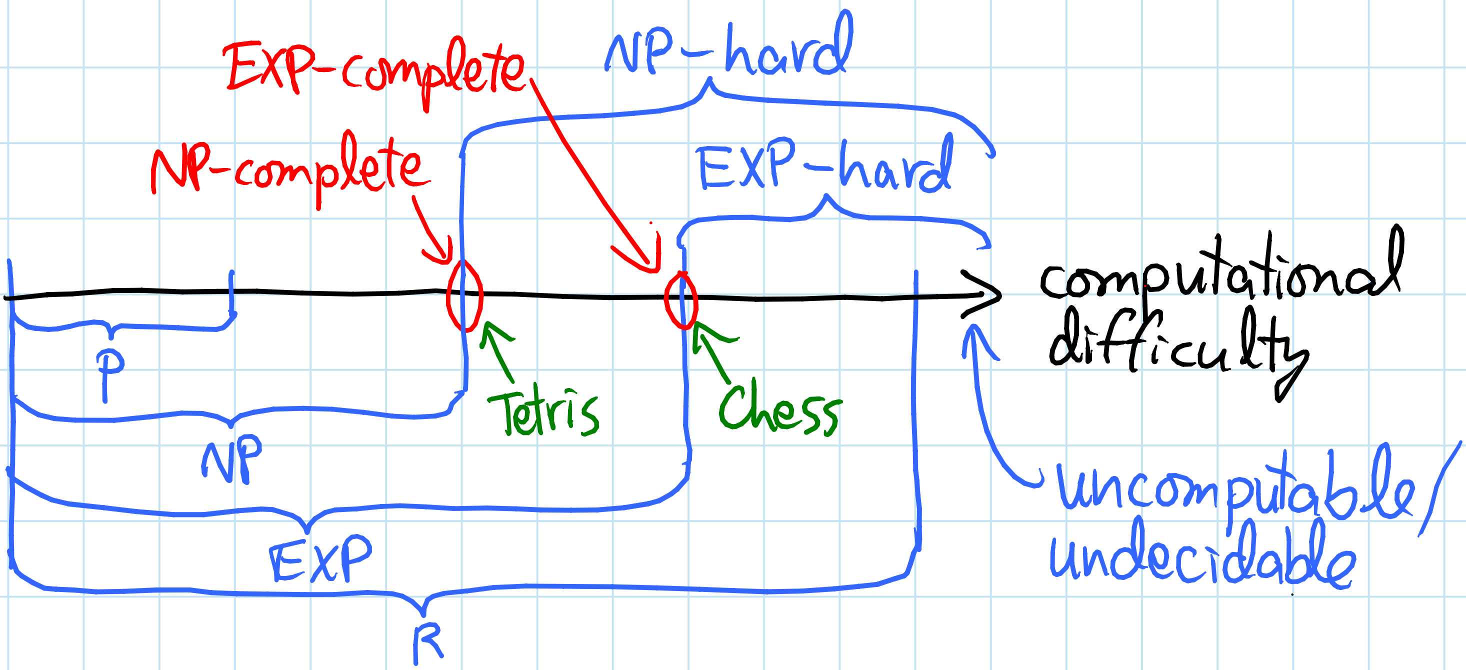
X-complete = X-hard $\cap$ X

sometimes "X-easy" = $\in$ X

e.g. Tetris is NP-complete [L3]

[Breukelaar, Demaine, Hohenberger, Hoogeboom, Kosters, Liben-Nowell 2004]

$\Rightarrow$ if $P \neq NP$, then Tetris $\in NP \setminus P$

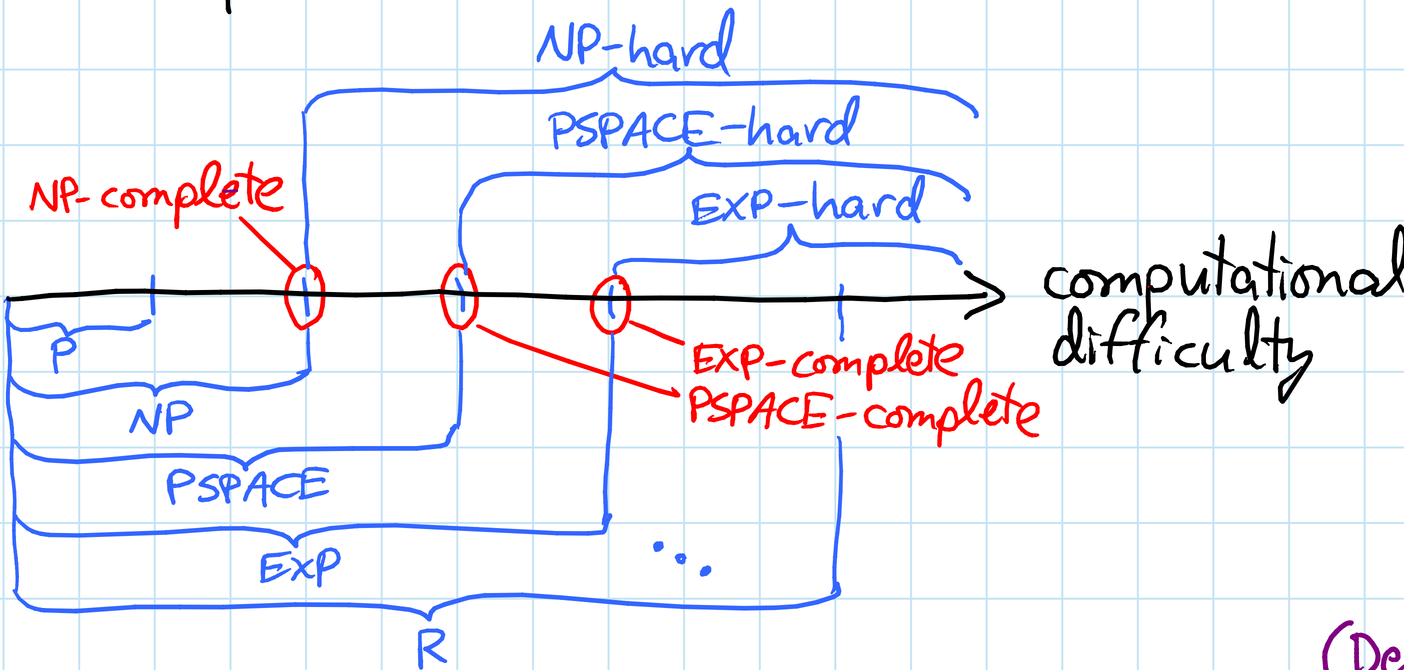


e.g. Chess is EXP-complete $\Rightarrow \notin P$

$\Rightarrow$ Chess $\in EXP \setminus NP$ if $NP \neq EXP$ (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 [L10]}

$\Rightarrow \notin$ 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 → $O(\lg n)$ 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] e.g. Mario ∈ NPSPACE = PSPACE
  in general, space bound squares
- NEXP, N2EXP, ...: analogs of NP

What does "as hard as" mean?

poly. size ("blowup")

(other constraints possible)

Reduction from A to B = poly-time algorithm to convert: instance of A → instance of B

such that solution to A = solution to B

⇒ if can solve B then can solve A

$B \in P$ $A \in P$
$B \in NP$ $A \in NP$

⇒ 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

$\hookrightarrow$ formula size or # variables (see L20)

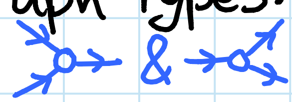
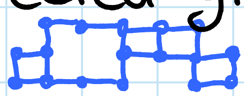
- concrete $P \neq NP$ conjecture relating to EXPTIME

- 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 L9)

- 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. 3-regular directed graphs 
  2. grid graph 
     - vertices $V \subseteq \mathbb{Z}^2$ (grid points)
     - edge $(u,v) \Leftrightarrow \text{distance}(u,v) = 1$

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

Examples of hardness proofs:

↗see L8

100% speedrunning video games: [≈Forišek – FUN 2010]

- 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 ②
  - grid graph → maze (edge → cooridor, vertex → branch)
  - target time = $n \cdot \underbrace{\text{vertex time}} + (n-1) \cdot \underbrace{\text{edge time}}$

    ↳ 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:

The Witness:

- squares of 2 colors + broken edges
  - ↳ ≤ 1 color per region of path
- k-triangles + broken edges
  - ↳ k incident edges in path
- edge hexagons (no broken edges)
  - ↳ must be visited by path
- 3-triangles (no broken edges)

Requirements: (taking for credit)

Measurement:

- watch all video lectures (2 × 1.5 hr / week / speed) before corresponding classes: — completion 👍 on Coauthor
  - 20% REQUIRED
- attend class (MW 3–4:30+): — attendance sheet
  - new material
  - progress reports
  - guest lectures
- participate in problem solving — coauthor ≥ 1 post on Coauthor by next SUNDAY
  - solved problems (pset puzzles)
  - open problems (research!)
  - coding problems (implement/visualize reductions)
  - collaborate!
  - 40% REQUIRED
- above all REQUIRED: must do MOST lectures & classes
- Problem sets, weekly — Gradescope
  - due MONDAYs @ NOON, including PS1 Sept. 11
  - 50% OPTIONAL
- project — code, write-up, present
  - based on in-class work on open/coding probs.
  - can be collaborative & done at any time
  - 50% OPTIONAL

↳ TOTAL: 160%

Grading scheme:

- lowest-scoring component's weight reduced until Σ weights = 100%

⇒ can skip project OR problem sets OR do both to make up for lost points

- this decides your letter grade (A, B, ...); we may use other schemes for +/−

Coauthor: master record of notes, ideas, progress
- anything worth saving should end up here.
  - e.g. post photos of useful whiteboards
- can use asynchronously too (between meetings)
  - Cocreate is a useful digital whiteboard
    https://cocreate.csail.mit.edu
- questions thread, including completion
- solved problems:
  - your posts are private (to avoid spoiling) to your coauthors (& us)
  - we may publish your answers to class
- open problems & research are SECRET: do not share outside this class!
- add whoever you're working with as coauthors

GitHub: for coding problems & paper/project writing
- feel free to create PRIVATE repos. (get permission before making public)
- link from relevant Coauthor thread

Office hours: this Friday @ 3pm
or by appointment