

CS 331, Fall 2024
Lecture 24 (12/2)

Today: - Recap/philosophy
- Oracles
- Between worlds
- Beyond P/NP

Recap/philosophy

Let $L \subseteq \{0,1\}^*$ be decision problem

Each $x \in \{0,1\}^n$ then is $\in L$ or not

e.g. $00010 \rightarrow \text{YES} \quad (\in L)$

$10101 \rightarrow \text{NO} \quad (\notin L)$

We view $x = \langle \text{input} \rangle$ as encoding

"human-readable" input by some std. map.

e.g. Independent Set: input = G , k
graph target size

For simplicity, if input invalid, $x \notin L$.

P : polynomial time.

$L \in P \iff \exists$ algo A s.t. $\forall x$,

$A(x)$ decides L in $\text{poly}(|x|)$ steps
(YES/NO)

NP : non-deterministic polynomial time.

$L \in NP \iff \exists$ algo A s.t. $\forall x$,

$x \in L \iff \exists h$ s.t. $A(x, h)$ returns YES
hint / proof / advice \nearrow
in $\text{poly}(|x|)$ steps
 $|h| = \text{poly}(|x|)$



psst... use h

"NP oracle": good at proving
? at refuting

We don't know if we live in

Algorithmica: $P = NP$

Pessiland*: $P \neq NP$

* Simplification
of Impagliazzo

Nonetheless, we can prove problems are

NP-hard: L is NP-hard iff

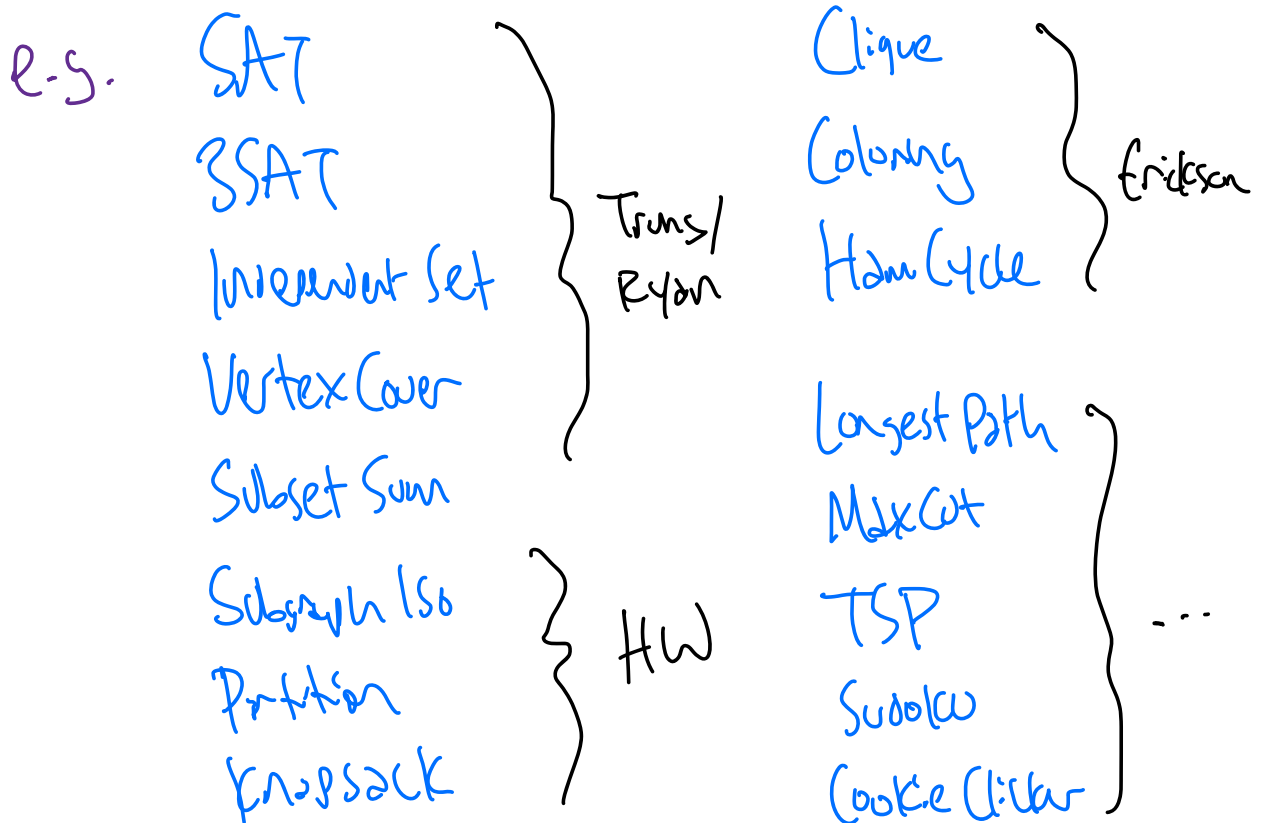
for all $L' \in NP$, if $L \in P$
then $L' \in P$

Unconditional statement. NP-complete: hardest $\in NP$

Why is this useful?

In 2019, 88-99% believe we live in **Pessimism**. Assuming this is true,

ANY NP-complete L can't be in P!



More fine-grained assumptions:

SAT takes c^n time for n literals (ETH)

View of reductions: $L \leq_P L'$

It's possible that $L \in P$ but $L' \notin P$

It's impossible that $L' \in P$ but $L \notin P$

So, L' is "harder".

What if we live in Algorithmica?

Also useful: reduction to SAT near-linear
in blowup of size.

Better SAT \Rightarrow better algos for all of NP!!!

Bad news: much of cryptography based on
assumption that we don't live in Algorithmica

Intuition: the key is the hint.

One more complexity class... \overline{NP}
"complement"

$L \in \overline{NP} \Leftrightarrow \exists$ algo A s.t. $\forall x$,
 $x \notin L \Leftrightarrow \exists h$ s.t. $A(x, h)$ returns **NO**
in $\text{poly}(|x|)$ steps
"refutation" \nearrow
 $|h| = \text{poly}(|x|)$

Claim: if we let \overline{L} = complement of L
(flip YES/NO)

then $L \in NP \Leftrightarrow \overline{L} \in \overline{NP}$

Proof: $x \notin \overline{L}$ is the same as $x \in L$

Use the same hint as for L .

Quick quiz: is PRIMES in NP/COMP?

$x \in \text{PRIMES}$ if $x = \langle n \rangle$ encodes prime n

Ans: COMP. Refutation: prime factorization

(more on this later...)

Oracles

Imagine you have an oracle who only decides L

Does G have
IS of size ≥ 10 ?

YES



What is on the
CS331 final exam?

IDK



Independent Set oracle

How impressed should you be?

Obviously, more impressed with harder problems

$L \leq_P L'$ if L' oracle can simulate L



can simulate



NP-hard: an oracle that can solve all **NP**

$A(x, \text{oracle})$

can decide all $L \in \text{NP}$

in $\text{poly}(|x|)$ time

+ $\text{poly}(|x|)$ calls to



Theorem:
(Cook-Levin)





is as good as

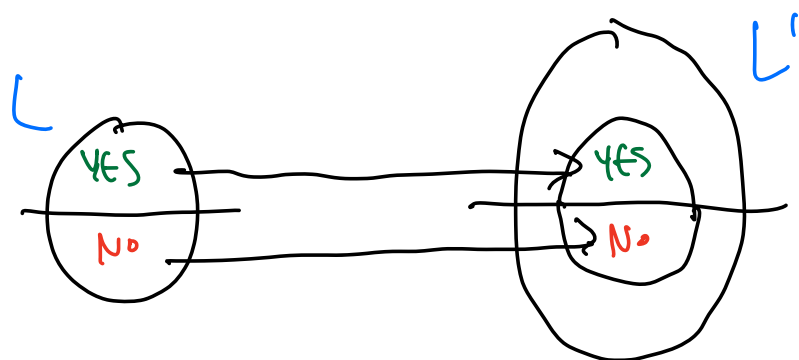


NP-oracle

Subtlety (optional for this class)

 is Turing oracle if can be called $poly(|x|)$ times in A .
(Trung's def)

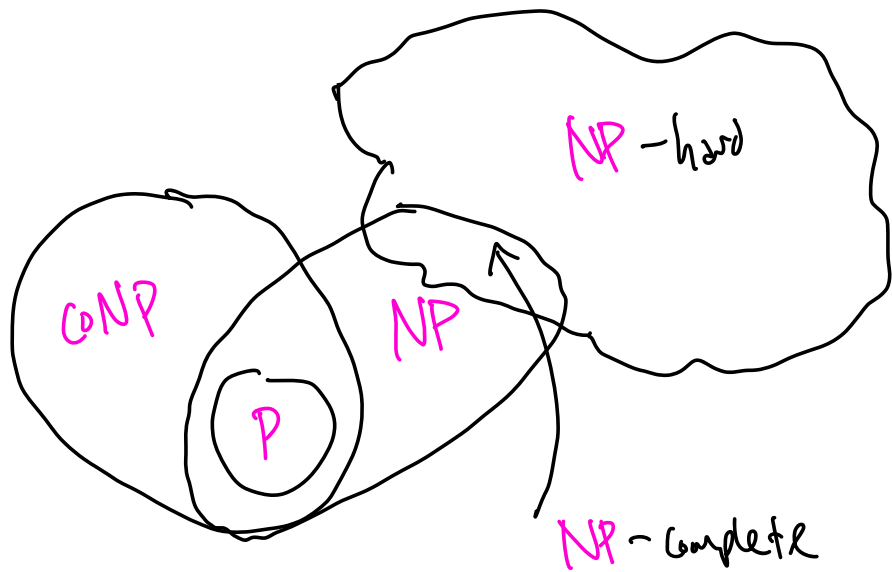
 is Karp oracle if can be called just once and you have to return its answer.
(Ryan's def)



In this class, either def is OK.

Impressively, SAT is NP-hard under Karp oracle.

Between worlds



Q: are there $L \in \text{NP-hard}$ but $\notin \text{NP}$?

A: yes. Famous example:

Halting: $x \in \text{Halting}$ if

$x = \langle A, \text{input} \rangle$ and

A does not run forever on input

Why is Halting NP-hard?

Let $L \in NP$. Want to decide $x \in L$?

Design A as follows:

Let A' be nondeterministic algo for L

Keep looping all configs h , stop if

$$A'(x, h) = \text{YES}$$

Ask  : does A halt on x ?

Why is Halting not in NP ?

In fact, Halting is not computable

Proof omitted, ask your TA
(key idea: "diagonalization")

Q: are there $L \in NP \cap coNP$ but $\notin P$?

A: We don't even know if $P = NP = coNP$

But... a few stories

PRIMES: in $coNP$ (obvious)

in NP (Pratt, 1975)

Proof sketch:

guess generator g

g^1, \dots, g^{n-1}

are all distinct. Then,
provide prime factors of $n-1$.

in BPP
(randomized P)

(Miller-Rabin, 1980)

in P

(Agrawal-Kayal-Saxena,
2002)

Graph Isomorphism : in NP (HW)

in NP-hard ???

in coNP ???

in quasi-P (Babai, 2016)

$n^{\text{poly}(\log(n))}$

Beyond P/NP

A few other cool classes.

PSPACE : decidable in $\text{poly}(n)$ space

L : decidable in $O(\log(n))$ space

$L \subseteq P \subseteq NP \subseteq PSPACE$

↑
unequal
↑

BPP: decidable w.p. $\geq \frac{2}{3}$ in $\text{poly}(n)$

Under slightly stronger than $P \neq NP$,

we get $P = BPP$.

Experts @ UT: David Zuckerman

Dan Moshkowitz

BQP: Same but with quantum

Unknown: relationship between BQP / NP

Expert @ UT: Scott Aaronson