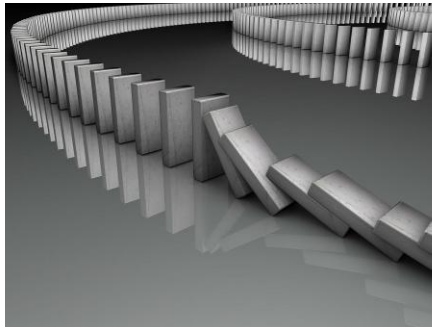
CMSC 28100

Introduction to Complexity Theory

Autumn 2025

Instructor: William Hoza

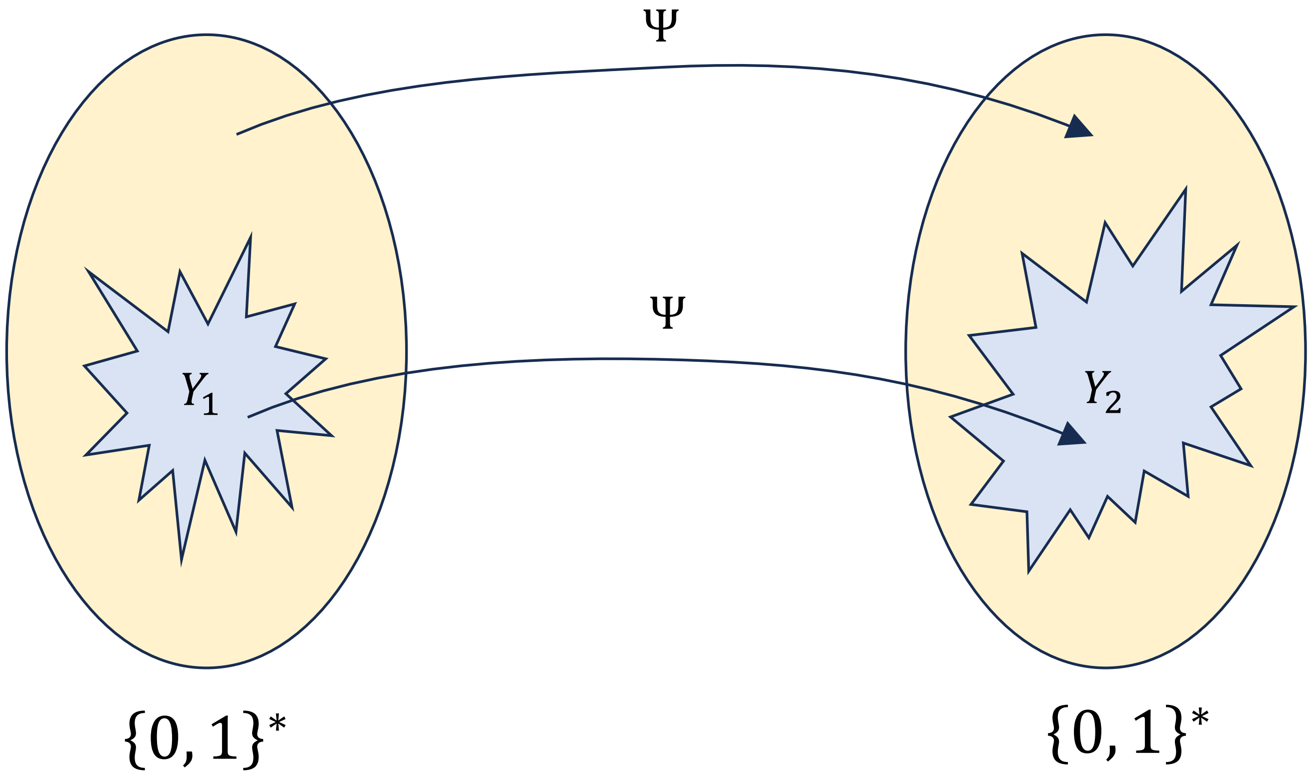
What about CLIQUE?



- $\mathrm{CLIQUE} = \{\langle G, k \rangle : G \text{ has a } k\text{-clique}\}$
- It seems likely that $\mathrm{CLIQUE} \notin \mathrm{P}$
- Can we prove it by doing a reduction from $\mathrm{BOUNDED\text{-}HALT}$?
- Answer: Probably not!
- To understand why, we need to go beyond "in P or not in P"

Mapping reductions

- $Y_1 \leq_{\mathrm{P}} Y_2$ means there is an efficient way to convert questions of the form "is $w \in Y_1$?" into questions of the form "is $w' \in Y_2$?"



EXP-hardness

- Let $Y \subseteq \{0,1\}^*$
- **Definition:** Y is "EXP-hard" if, for every $L \in \mathrm{EXP}$, we have $L \leq_{\mathrm{P}} Y$
- Interpretation:
  - Y is at least as hard as any language in EXP
  - Every problem in EXP is basically a special case of Y

Example: BOUNDED-HALT is EXP-hard

- $\text{BOUNDED-HALT} = \{\langle M, w, T\rangle : M \text{ halts on } w \text{ within } T \text{ steps}\}$
- **Claim:** BOUNDED-HALT is EXP-hard
- **Proof:** Let $Y \in \text{EXP}$. We will show $Y \leq_{\text{P}}$ BOUNDED-HALT
- There is a TM M that $\begin{cases} \text{accepts } w \text{ within } 2^{|w|^k} \text{ steps} & \text{if } w \in Y \\ \text{loops} & \text{if } w \notin Y \end{cases}$
- Mapping reduction: $\Psi(w) = \left\langle M, w, 2^{|w|^k} \right\rangle$

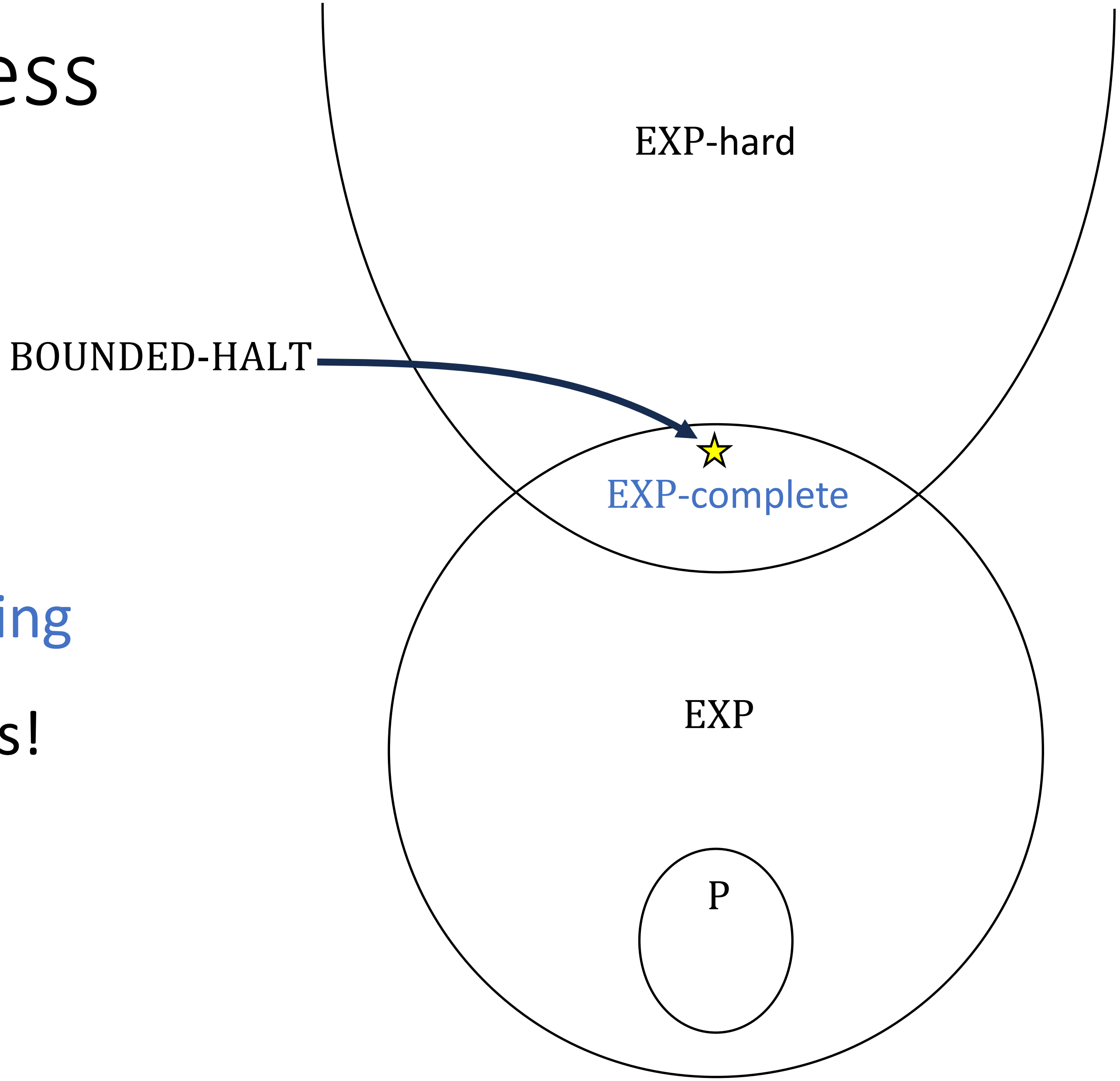
EXP-hard languages are intractable

- Let $Y \subseteq \{0, 1\}^*$
- **Claim:** If Y is EXP-hard, then $Y \notin \mathrm{P}$
- **Proof:** There exists $L \in \mathrm{EXP}$ such that $L \notin \mathrm{P}$
- Since Y is EXP-hard, we have $L \leq_{\mathrm{P}} Y$

EXP-completeness

- Let $Y \subseteq \{0, 1\}^*$
- **Definition:** We say Y is EXP-complete if Y is EXP-hard and $Y \in \mathrm{EXP}$
- The EXP-complete languages are the hardest languages in EXP
- If Y is EXP-complete, then the language Y can be said to “capture” / “express” the entire complexity class EXP

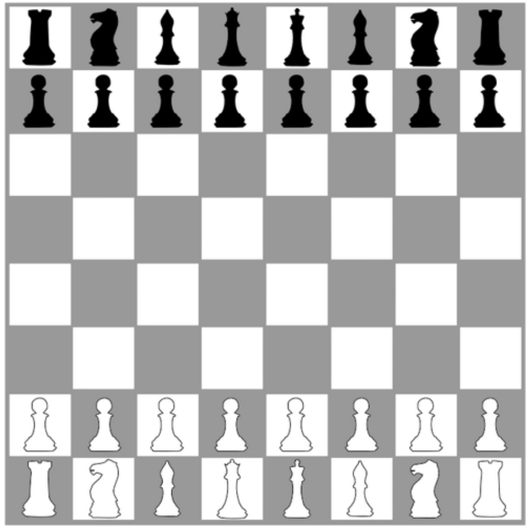
EXP-completeness



There are many interesting

EXP-complete languages!

Example: Chess



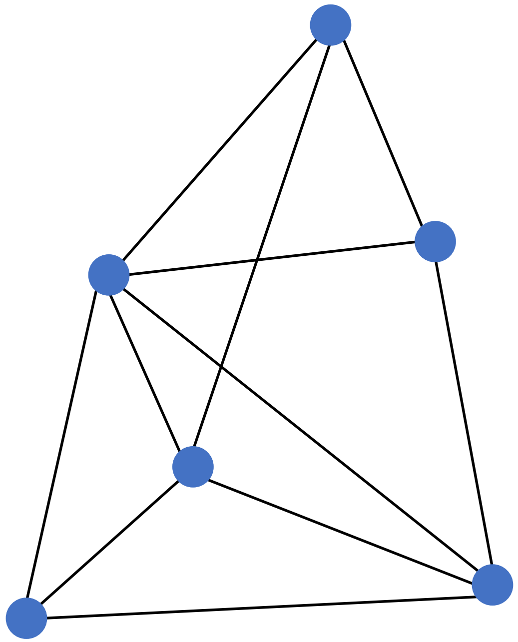
- Let $\text{GENERALIZED-CHESS} = \{\langle P \rangle : P \text{ is an arrangement of chess pieces on an } N \times N \text{ board from which player 1 can force a win}\}$

Theorem: GENERALIZED-CHESS is EXP-complete.

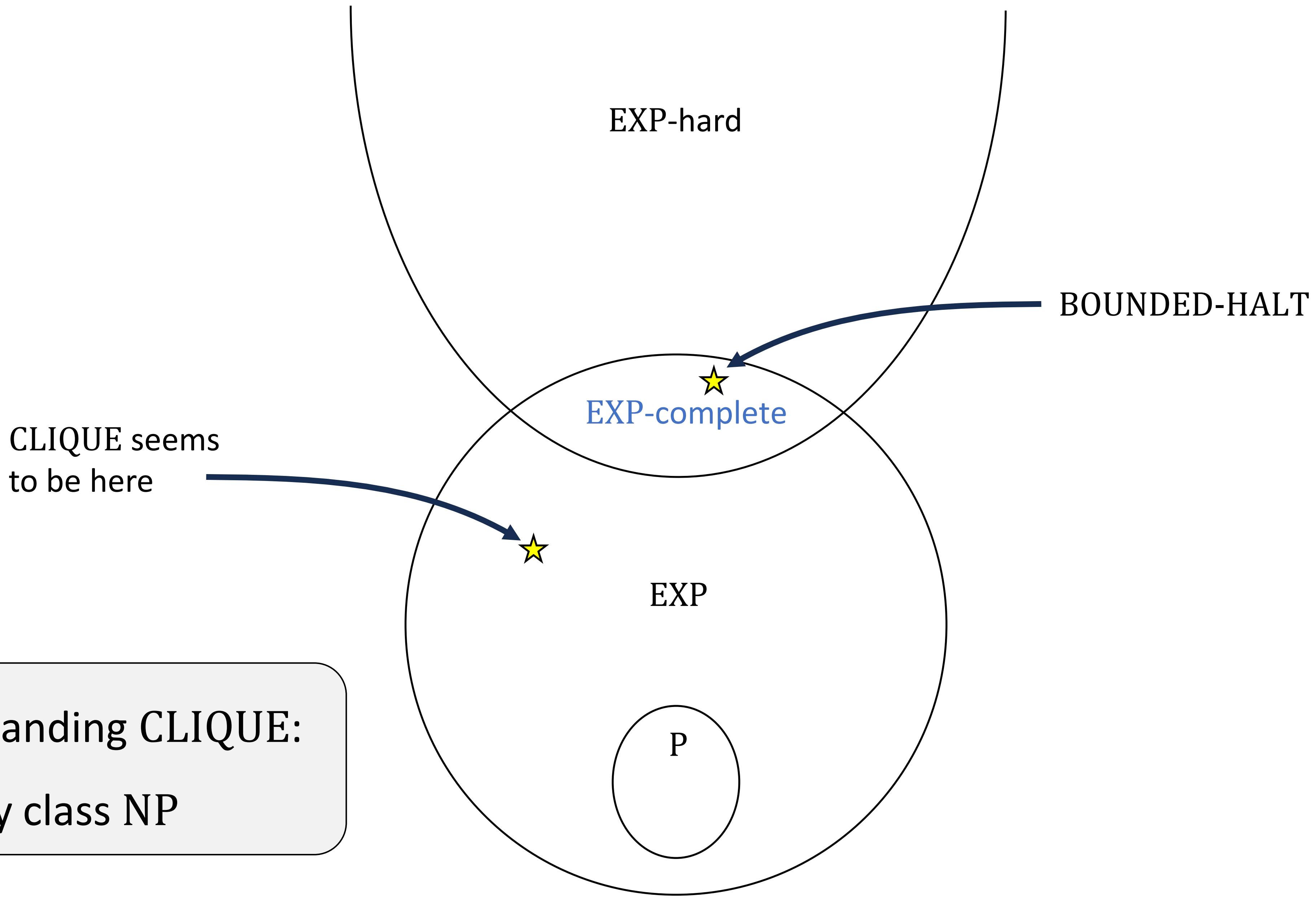
Consequently, GENERALIZED-CHESS $\notin$ P.

- (Proof omitted. This theorem will not be on exercises/exams)

Why reductions don't always work



- We would like to prove $\mathrm{CLIQUE} \notin \mathrm{P}$
- We could try proving $\mathrm{BOUNDED\text{-}HALT} \leq_{\mathrm{P}} \mathrm{CLIQUE}$
- But that would imply that CLIQUE is EXP-hard
- In reality, CLIQUE is probably not EXP-hard!



Key to understanding CLIQUE:

The complexity class NP

The complexity class NP

- Let $Y \subseteq \{0,1\}^*$
- **Definition:** $Y \in \mathrm{NP}$ if there exists a randomized polynomial-time Turing machine M such that for every $w \in \{0,1\}^*$:
  - If $w \in Y$, then $\Pr[M \text{ accepts } w] \neq 0$
  - If $w \notin Y$, then $\Pr[M \text{ accepts } w] = 0$

"Nondeterministic Turing machine"

- "Nondeterministic Polynomial-time"

Another example of a language in $\mathbf{NP}$

- $\text{FACTOR} = \{\langle K, M \rangle : K \text{ has a prime factor } p \leq M\}$
- **Claim:** $\text{FACTOR} \in \text{NP}$
- **Proof:**
  1. Pick $R \in \{2, 3, 4, \ldots, M\}$ uniformly at random
  2. Check whether K/R is an integer (long division)
  3. If it is, accept; if it isn't, reject

How to interpret NP

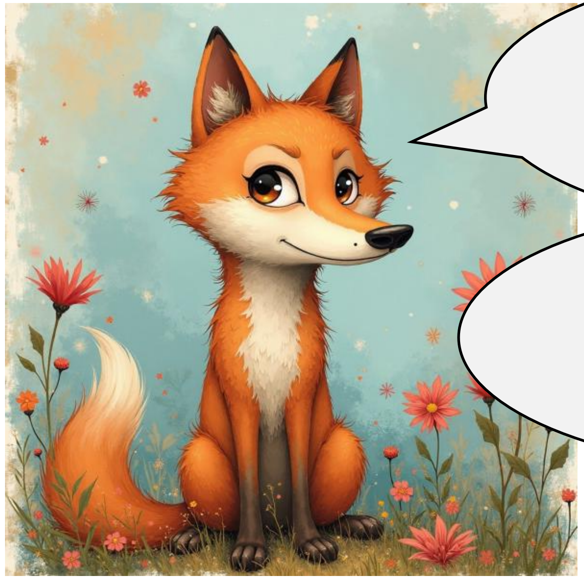
- NP is not intended to model the concept of tractability
- A nondeterministic polynomial-time algorithm is not a practical way to solve a problem
- Instead, NP is a conceptual tool for reasoning about computation

Another way of thinking about $\mathbf{NP}$

- Two equivalent ways of defining NP:

1. One person, computing with a coin
    - (Randomized Turing machine model)
2. Two people: A prover and a verifier
    - (No randomness)

Prover and verifier



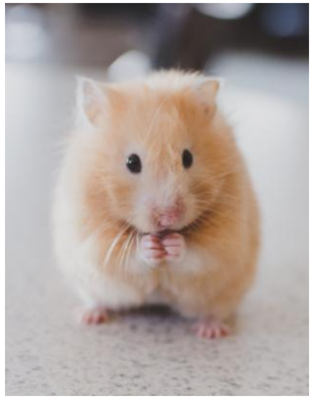
$w \in Y$, because
[blah blah blah]

Accept/Reject

Prover
(Computationally
Unbounded)

Verifier
(Polynomial Time)

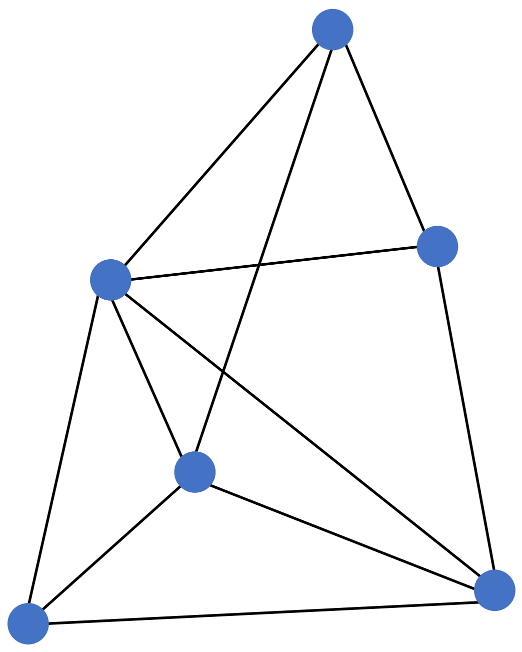
Prover and verifier



- Let $Y \subseteq \{0,1\}^*$
- **Definition:** A polynomial-time verifier for Y is a polynomial-time deterministic Turing machine V such that for some constant $k \in \mathbb{N}$, we have:
  - For every $w \in Y$, there exists $x \in \{0,1\}^*$ such that $|x| \leq |w|^k$ and V accepts $\langle w, x \rangle$
    - "Completeness"
  - For every $w \notin Y$, for every $x \in \{0,1\}^*$, V rejects $\langle w, x \rangle$
    - "Soundness"

"Proof" / "Certificate" / "Witness"

Example: CLIQUE



- **Claim:** There exists a polynomial-time verifier for CLIQUE
- **Verifier:** Given $\langle G, k, x \rangle$:
  - Check whether x encodes a k-clique in G
  - If yes, accept, if no, reject
- Polynomial time ✔ Completeness ✔ Soundness ✔

Equivalence of the two definitions

- Let $Y \subseteq \{0,1\}^*$
- **Claim:** $Y \in \mathrm{NP}$ if and only if there exists a polynomial-time verifier for Y
- **Proof:**
  - ($\Leftarrow$) Randomly pick a certificate x, then run the verifier
  - ($\Rightarrow$) Verifier runs randomized TM with certificate in place of random bits
- Get comfortable with both ways of thinking about NP

Comparing NP and P/poly

Prover (NP)	Advisor (P/poly)
Computationally unbounded	Computationally unbounded
Knows entire input	Only knows length of input
Untrustworthy	Trustworthy