Reminder: Complexity Classes

A decision problem corresponds to a language $L \subseteq \{0,1\}^*$ containing all yes-instances (w.r.t. some encoding)

Reminder: Complexity Classes

A decision problem corresponds to a language $L \subseteq \{0,1\}^*$ containing all yes-instances (w.r.t. some encoding)

- P **(polynomial-time):** The class of all languages that can be decided by a deterministic polynomial-time Turing machine

Reminder: Complexity Classes

A decision problem corresponds to a language $L \subseteq \{0,1\}^*$ containing all yes-instances (w.r.t. some encoding)

- P **(polynomial-time):** The class of all languages that can be decided by a deterministic polynomial-time Turing machine
- BPP **(bounded-error probabilistic polynomial-time):** The class of all languages L that can be decided in polynomial-time by a probabilistic Turing machine T with probability of error bounded by $\frac{1}{3}$

— If $x \in L$, then $\Pr[T(x)$ accepts$] \geq \frac{2}{3}$

— If $x \notin L$, then $\Pr[T(x)$ accepts$] \leq \frac{1}{3}$

Reminder: Complexity Classes

A decision problem corresponds to a language $L \subseteq \{0,1\}^*$ containing all yes-instances (w.r.t. some encoding)

- P **(polynomial-time):** The class of all languages that can be decided by a deterministic polynomial-time Turing machine
- BPP **(bounded-error probabilistic polynomial-time):** The class of all languages L that can be decided in polynomial-time by a probabilistic Turing machine T with probability of error bounded by $\frac{1}{3}$

— If $x \in L$, then $\Pr[T(x)$ accepts$] \geq \frac{2}{3}$

— If $x \notin L$, then $\Pr[T(x)$ accepts$] \leq \frac{1}{3}$

- NP **(non-deterministic polynomial-time):** The class of all languages that can be decided by a non-deterministic polynomial-time Turing machine T

— If $x \in L$, then at least one computation path of $T(x)$ accepts

— If $x \notin L$, then all computation paths of $T(x)$ reject

Reminder: Complexity Classes

A decision problem corresponds to a language $L \subseteq \{0,1\}^*$ containing all yes-instances (w.r.t. some encoding)

- P **(polynomial-time):** The class of all languages that can be decided by a deterministic polynomial-time Turing machine
- BPP **(bounded-error probabilistic polynomial-time):** The class of all languages L that can be decided in polynomial-time by a probabilistic Turing machine T with probability of error bounded by $\frac{1}{3}$

— If $x \in L$, then $\Pr[T(x) \text{ accepts}] \geq \frac{2}{3}$

— If $x \notin L$, then $\Pr[T(x) \text{ accepts}] \leq \frac{1}{3}$

- NP **(non-deterministic polynomial-time):** The class of all languages that can be decided by a non-deterministic polynomial-time Turing machine T

— If $x \in L$, then at least one computation path of $T(x)$ accepts

— If $x \notin L$, then all computation paths of $T(x)$ reject

- PSPACE **(polynomial-space):** The class of all languages that can be decided by a deterministic Turing machine that uses a polynomial amount of space (w.r.t. the size of the input)

Reminder: Complexity Classes

A decision problem corresponds to a language $L \subseteq \{0,1\}^*$ containing all yes-instances (w.r.t. some encoding)

- P **(polynomial-time):** The class of all languages that can be decided by a deterministic polynomial-time Turing machine
- BPP **(bounded-error probabilistic polynomial-time):** The class of all languages L that can be decided in polynomial-time by a probabilistic Turing machine T with probability of error bounded by $\frac{1}{3}$

— If $x \in L$, then $\Pr[T(x)$ accepts$] \geq \frac{2}{3}$

— If $x \notin L$, then $\Pr[T(x)$ accepts$] \leq \frac{1}{3}$

- NP **(non-deterministic polynomial-time):** The class of all languages that can be decided by a non-deterministic polynomial-time Turing machine T

— If $x \in L$, then at least one computation path of $T(x)$ accepts

— If $x \notin L$, then all computation paths of $T(x)$ reject

- PSPACE **(polynomial-space):** The class of all languages that can be decided by a deterministic Turing machine that uses a polynomial amount of space (w.r.t. the size of the input)

P $\subseteq$ BPP $\subseteq$ PSPACE   P $\subseteq$ NP $\subseteq$ PSPACE

Reminder: Complexity Classes

A decision problem corresponds to a language $L \subseteq \{0,1\}^*$ containing all yes-instances (w.r.t. some encoding)

- P **(polynomial-time):** The class of all languages that can be decided by a deterministic polynomial-time Turing machine
- BPP **(bounded-error probabilistic polynomial-time):** The class of all languages L that can be decided in polynomial-time by a probabilistic Turing machine T with probability of error bounded by $\frac{1}{3}$

— If $x \in L$, then $\Pr[T(x)$ accepts$] \geq \frac{2}{3}$

— If $x \notin L$, then $\Pr[T(x)$ accepts$] \leq \frac{1}{3}$

- NP **(non-deterministic polynomial-time):** The class of all languages that can be decided by a non-deterministic polynomial-time Turing machine T

— If $x \in L$, then at least one computation path of $T(x)$ accepts

— If $x \notin L$, then all computation paths of $T(x)$ reject

- PSPACE **(polynomial-space):** The class of all languages that can be decided by a deterministic Turing machine that uses a polynomial amount of space (w.r.t. the size of the input)

P $\subseteq$ BPP $\subseteq$ PSPACE   P $\subseteq$ NP $\subseteq$ PSPACE   The relation between BPP and NP is unknown

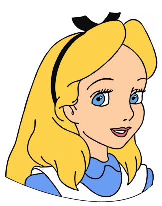
Zero Knowledge Proofs

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Zero Knowledge Proofs

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

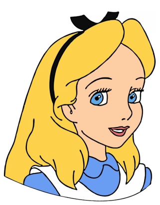


Zero Knowledge Proofs

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

I know a solution to this Sudoku instance

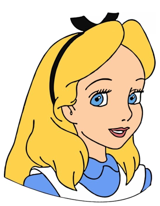


Zero Knowledge Proofs

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Really? Show it to me!

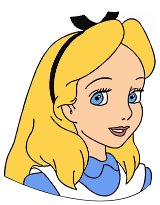


Zero Knowledge Proofs

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

I don't want to reveal it to you

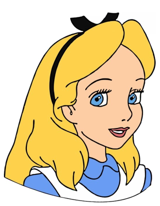


Zero Knowledge Proofs

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Then I don't believe you really have a solution

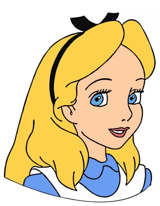


Zero Knowledge Proofs

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

I can prove to you I have a solution without revealing anything about it

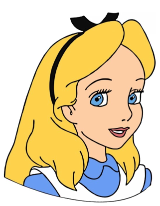


Zero Knowledge Proofs

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

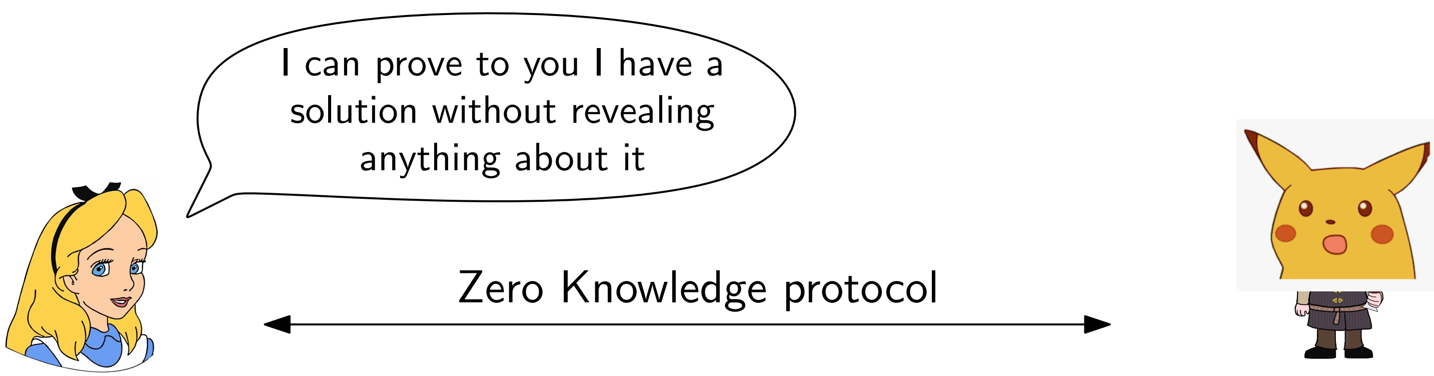
I can prove to you I have a solution without revealing anything about it



Zero Knowledge Proofs

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

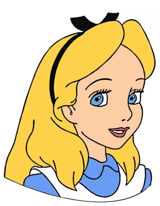


Zero Knowledge Proofs

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

I am now convinced
you have the solution



Zero Knowledge protocol

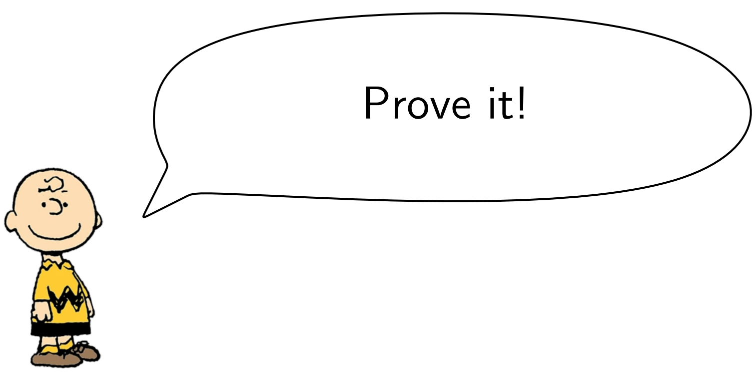
Zero Knowledge Proofs

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Hey, Charlie!
Alice has a solution to
this Sudoku instance

Zero Knowledge Proofs

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9



Zero Knowledge Proofs

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Even if I definitely know she has a solution, somehow I have no way of proving that

What are proofs anyway?

There is some **claim** x known to both Alice and Bob

A **prover** (Alice) wants to convince a **verifier** (Bob) that the claim is true

What are proofs anyway?

There is some **claim** x known to both Alice and Bob

A **prover** (Alice) wants to convince a **verifier** (Bob) that the claim is true

What is a proof?

- Classically, a proof is a string w that "convinces" the verifier

What are proofs anyway?

There is some **claim** x known to both Alice and Bob

A **prover** (Alice) wants to convince a **verifier** (Bob) that the claim is true

What is a proof?

- Classically, a proof is a string w that "convinces" the verifier

$x =$ "In a right triangle with legs of length a, b and hypotenuse length c, we have $a^2 + b^2 = c^2$"
$w =$ "Consider a right triangle with vertices a, b, c and let... QED."

What are proofs anyway?

There is some **claim** x known to both Alice and Bob

A **prover** (Alice) wants to convince a **verifier** (Bob) that the claim is true

What is a proof?

- Classically, a proof is a string w that "convinces" the verifier

$x =$ "In a right triangle with legs of length a, b and hypotenuse length c, we have $a^2 + b^2 = c^2$"

$w =$"Consider a right triangle with vertices a, b, c and let... QED."

$x =$ "The number N is the product of two primes"

$w = (p, q)$ where p, q are primes and $pq = N$

What are proofs anyway?

There is some **claim** x known to both Alice and Bob

A **prover** (Alice) wants to convince a **verifier** (Bob) that the claim is true

What is a proof?

- Classically, a proof is a string w that "convinces" the verifier

$x =$ "In a right triangle with legs of length a, b and hypotenuse length c, we have $a^2 + b^2 = c^2$"

$w =$"Consider a right triangle with vertices a, b, c and let... QED."

$x =$ "The number N is the product of two primes"

$w = (p, q)$ where p, q are primes and $pq = N$

Consider claims of the form "x belongs to some language L"

What are proofs anyway?

There is some **claim** x known to both Alice and Bob

A **prover** (Alice) wants to convince a **verifier** (Bob) that the claim is true

What is a proof?

- Classically, a proof is a string w that "convinces" the verifier

$x =$ "In a right triangle with legs of length a, b and hypotenuse length c, we have $a^2 + b^2 = c^2$"

$w =$"Consider a right triangle with vertices a, b, c and let... QED."

$x =$ "The number N is the product of two primes"

$w = (p, q)$ where p, q are primes and $pq = N$

Consider claims of the form "x belongs to some language L"

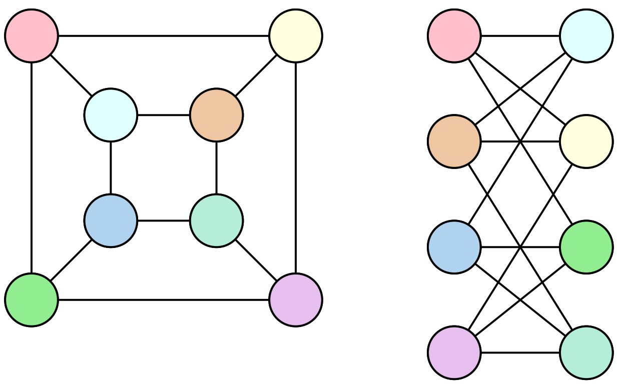
We can model the verifier as a polynomial-time algorithm $\mathcal{V}(x, w)$ that takes as input the claim x and the proof w, and outputs 1 if and only if it accepts w as a proof of x

Example: Graph Isomorphism

Graph isomorphism problem

G_1 is isomorphic to G_2 iff $\exists$ bijection $\pi : V_1 \to V_2$ s.t. $(u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$.

$G_1 = (V_1, E_1)$ $\quad$ $G_2 = (V_2, E_2)$

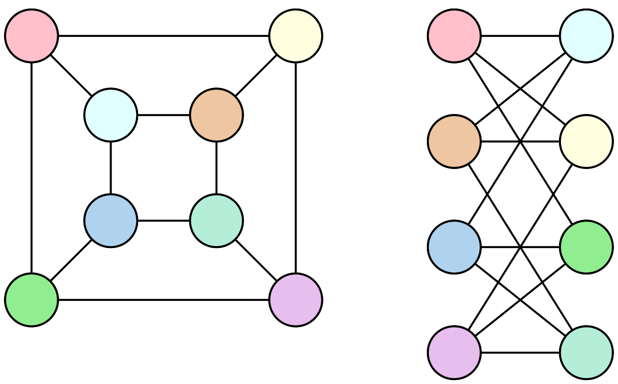


Example: Graph Isomorphism

Graph isomorphism problem

G_1 is isomorphic to G_2 iff $\exists$ bijection $\pi : V_1 \to V_2$ s.t. $(u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$.

$G_1 = (V_1, E_1)$ $G_2 = (V_2, E_2)$



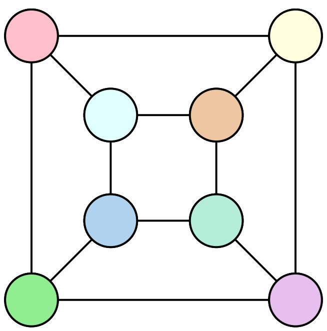
Claim: G_1 and G_2 are isomorphic.

Example: Graph Isomorphism

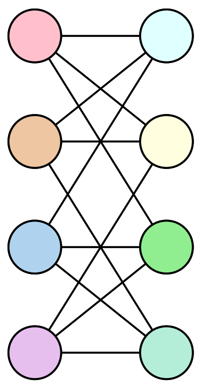
Graph isomorphism problem

G_1 is isomorphic to G_2 iff $\exists$ bijection $\pi : V_1 \to V_2$ s.t. $(u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$.

$G_1 = (V_1, E_1)$



$G_2 = (V_2, E_2)$



Claim: G_1 and G_2 are isomorphic.

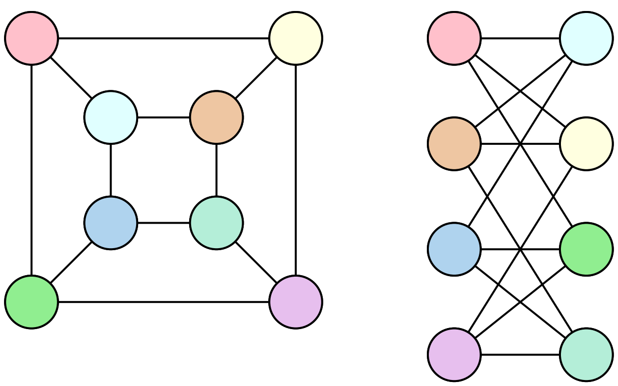
$$L = \{(G_1, G_2) \mid \exists \text{ bijection } \pi : V_1 \to V_2 \text{ s.t. } (u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2\}$$

Example: Graph Isomorphism

Graph isomorphism problem

G_1 is isomorphic to G_2 iff $\exists$ bijection $\pi : V_1 \rightarrow V_2$ s.t. $(u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$.

$G_1 = (V_1, E_1)$ $\quad$ $G_2 = (V_2, E_2)$



Claim: G_1 and G_2 are isomorphic.

$$L = \{(G_1, G_2) \mid \exists \text{ bijection } \pi : V_1 \rightarrow V_2 \text{ s.t. } (u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2\}$$

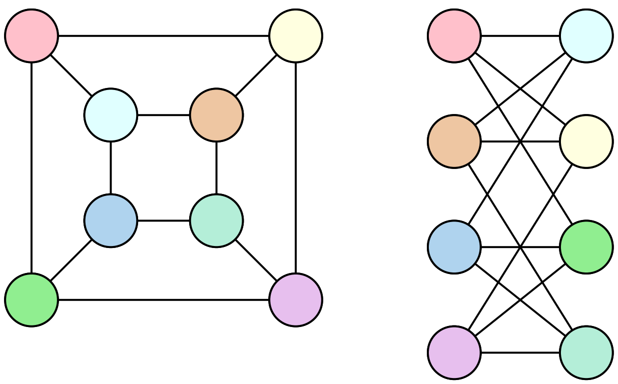
$$x = (G_1, G_2)$$

Example: Graph Isomorphism

Graph isomorphism problem

G_1 is isomorphic to G_2 iff $\exists$ bijection $\pi : V_1 \to V_2$ s.t. $(u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$.

$G_1 = (V_1, E_1)$ $\qquad$ $G_2 = (V_2, E_2)$



Claim: G_1 and G_2 are isomorphic.

$$L = \{(G_1, G_2) \mid \exists \text{ bijection } \pi : V_1 \to V_2 \text{ s.t. } (u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2\}$$

$x = (G_1, G_2)$

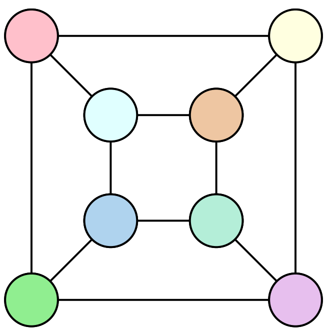
$w = \pi$

Example: Graph Isomorphism

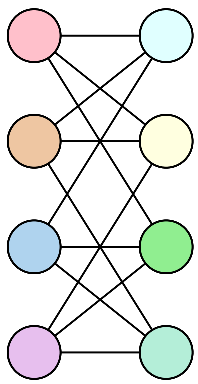
Graph isomorphism problem

G_1 is isomorphic to G_2 iff $\exists$ bijection $\pi : V_1 \to V_2$ s.t. $(u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$.

$G_1 = (V_1, E_1)$



$G_2 = (V_2, E_2)$



Claim: G_1 and G_2 are isomorphic.

$$L = \{(G_1, G_2) \mid \exists \text{ bijection } \pi : V_1 \to V_2 \text{ s.t. } (u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2\}$$

$x = (G_1, G_2)$

$w = \pi$

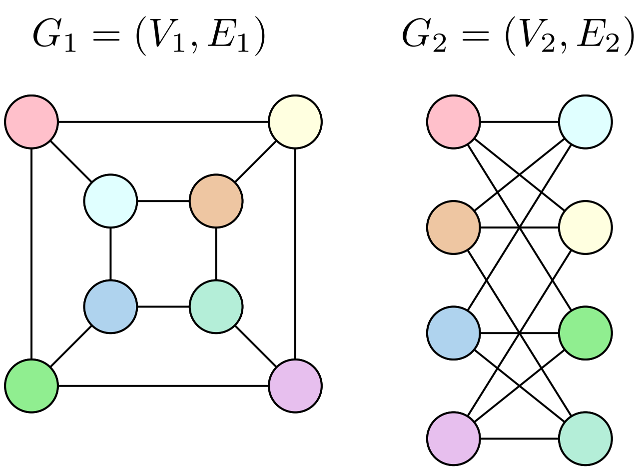
Algorithm $\mathcal{V}((G_1, G_2), \pi)$:

- For all $u, v \in V_1^2$:
  - If $((u, v) \in E_1 \wedge (\pi(u), \pi(v)) \notin E_2) \vee ((u, v) \notin E_1 \wedge (\pi(u), \pi(v)) \in E_2)$: Return 0
- Return 1

Example: Graph Isomorphism

Graph isomorphism problem

G_1 is isomorphic to G_2 iff $\exists$ bijection $\pi : V_1 \to V_2$ s.t. $(u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$.



Claim: G_1 and G_2 are isomorphic.

$$L = \{(G_1, G_2) \mid \exists \text{ bijection } \pi : V_1 \to V_2 \text{ s.t. } (u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2\}$$

$x = (G_1, G_2)$

$w = \pi$

The verifier learns the isomorphism π

Algorithm $\mathcal{V}((G_1, G_2), \pi)$:

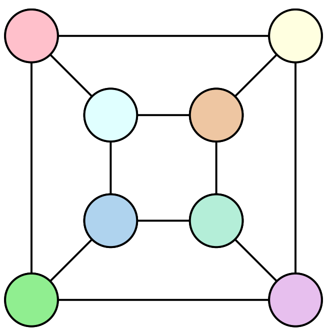
- For all $u, v \in V_1^2$:
  - If $((u, v) \in E_1 \land (\pi(u), \pi(v)) \notin E_2) \lor ((u, v) \notin E_1 \land (\pi(u), \pi(v)) \in E_2)$: Return 0
- Return 1

Example: Graph Isomorphism

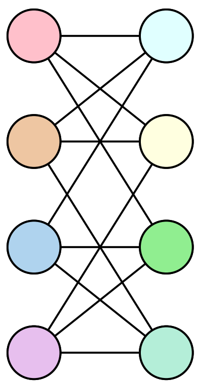
Graph isomorphism problem

G_1 is isomorphic to G_2 iff $\exists$ bijection $\pi : V_1 \to V_2$ s.t. $(u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2$.

$G_1 = (V_1, E_1)$



$G_2 = (V_2, E_2)$



Claim: G_1 and G_2 are isomorphic.

$$L = \{(G_1, G_2) \mid \exists \text{ bijection } \pi : V_1 \to V_2 \text{ s.t. } (u, v) \in E_1 \iff (\pi(u), \pi(v)) \in E_2\}$$

$x = (G_1, G_2)$

$w = \pi$

The verifier learns the isomorphism π

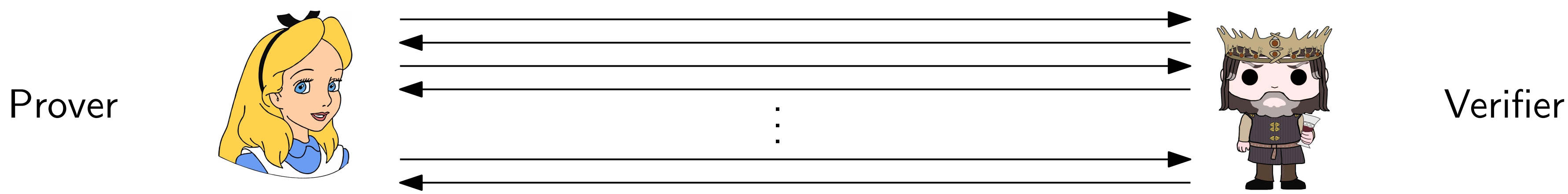
Is it necessary to reveal π?

Algorithm $\mathcal{V}((G_1, G_2), \pi)$:

- For all $u, v \in V_1^2$:
  - If $((u, v) \in E_1 \wedge (\pi(u), \pi(v)) \notin E_2) \vee ((u, v) \notin E_1 \wedge (\pi(u), \pi(v)) \in E_2)$: Return 0
- Return 1

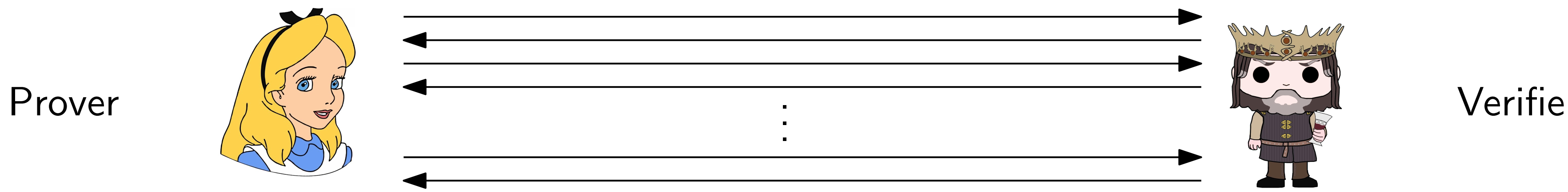
Interactive Proofs Systems

In an **interactive proof system**, the prover and the verifier exchange multiple messages following some **protocol**



Interactive Proofs Systems

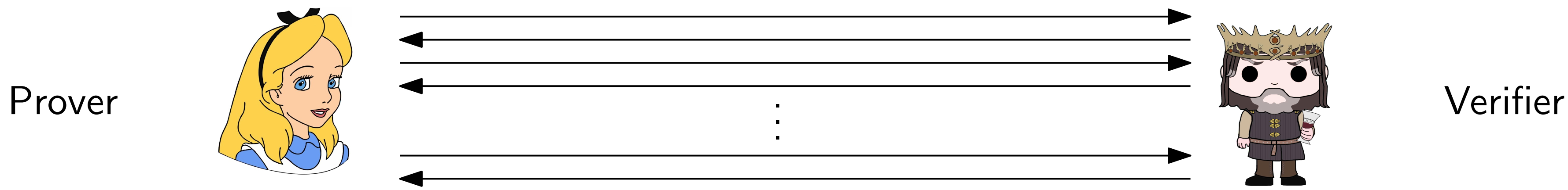
In an **interactive proof system**, the prover and the verifier exchange multiple messages following some **protocol**



The prover is a **computationally unbounded** algorithm

Interactive Proofs Systems

In an **interactive proof system**, the prover and the verifier exchange multiple messages following some **protocol**

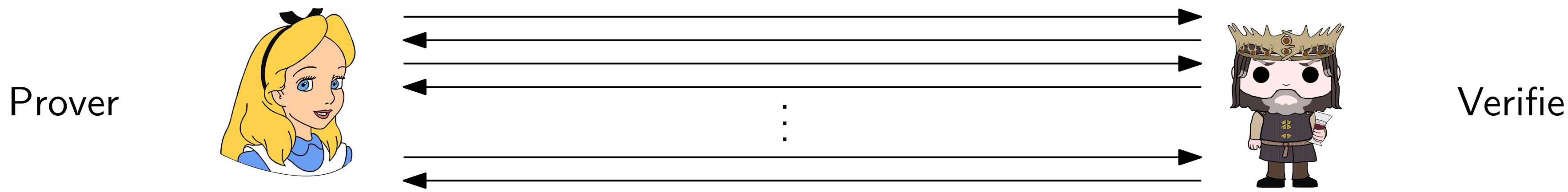


The prover is a **computationally unbounded** algorithm

The verifier will be a polynomial-time **randomized** algorithm

Interactive Proofs Systems

In an **interactive proof system**, the prover and the verifier exchange multiple messages following some **protocol**



The prover is a **computationally unbounded** algorithm

The verifier will be a polynomial-time **randomized** algorithm

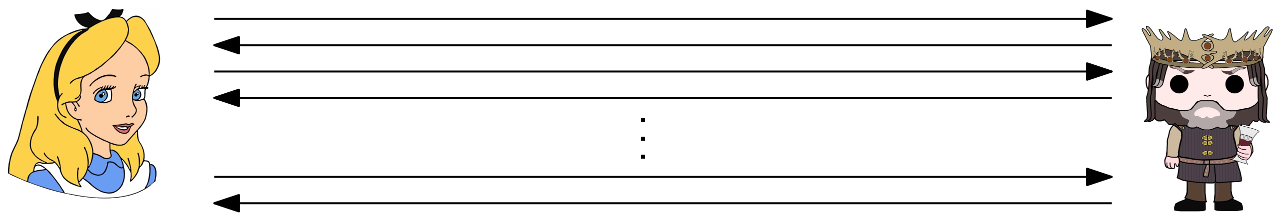
Ideally, at the end of the interaction:

- If $x \in L$, the prover knows a proof, and follows the protocol, then the verifier will be convinced that $x \in L$ (**completeness**)

Interactive Proofs Systems

In an **interactive proof system**, the prover and the verifier exchange multiple messages following some **protocol**

Prover



Verifier

The prover is a **computationally unbounded** algorithm

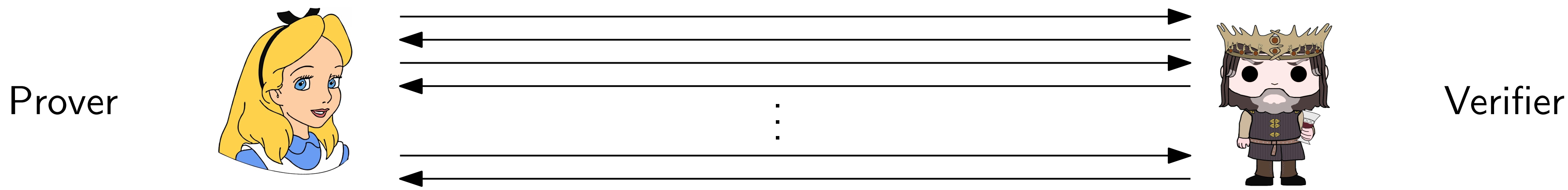
The verifier will be a polynomial-time **randomized** algorithm

Ideally, at the end of the interaction:

- If $x \in L$, the prover knows a proof, and follows the protocol, then the verifier will be convinced that $x \in L$ (**completeness**)
- If $x \notin L$, no prover (even a cheating prover that deviates from the protocol) manages to convince the verfier that $x \in L$ (**soundness**)

Interactive Proofs Systems

In an **interactive proof system**, the prover and the verifier exchange multiple messages following some **protocol**



The prover is a **computationally unbounded** algorithm

The verifier will be a polynomial-time **randomized** algorithm

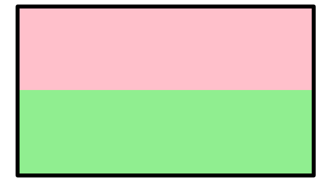
Ideally, at the end of the interaction:

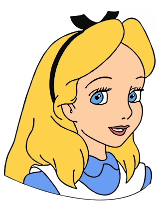
- If $x \in L$, the prover knows a proof, and follows the protocol, then the verifier will be convinced that $x \in L$ (**completeness**)
- If $x \notin L$, no prover (even a cheating prover that deviates from the protocol) manages to convince the verfier that $x \in L$ (**soundness**)

We relax the above requirements by allowing the verifier to commit errors (with a small probability).

A Physical Interactive Proof System

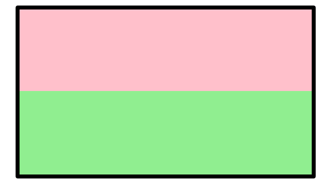
Suppose that the prover needs to convince a colorbrind verifier that this sheet of paper contains multiple colors:





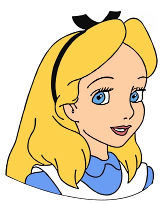
A Physical Interactive Proof System

Suppose that the prover needs to convince a colorbrind verifier that this sheet of paper contains multiple colors:



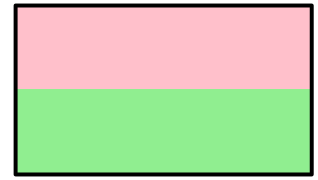
The verifier secretly chooses a random bit b

- If $b = 0$ the verifier keeps the sheet in the original orientation 
- If $b = 1$ the verifier flips the sheet upside down 



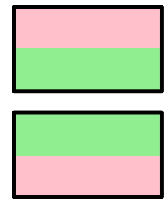
A Physical Interactive Proof System

Suppose that the prover needs to convince a colorbrind verifier that this sheet of paper contains multiple colors:

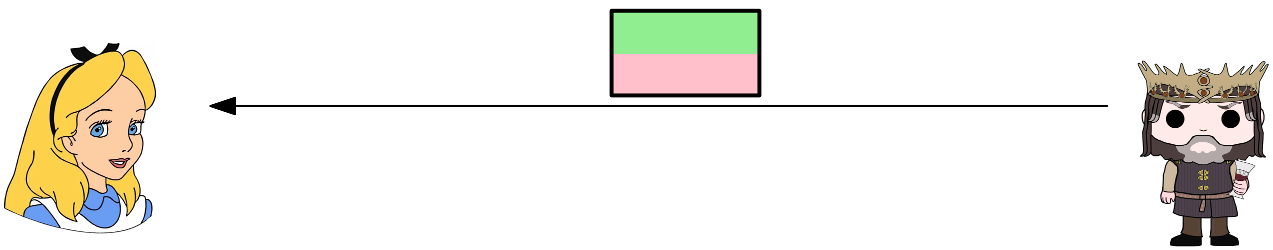


The verifier secretly chooses a random bit b

- If $b = 0$ the verifier keeps the sheet in the original orientation
- If $b = 1$ the verifier flips the sheet upside down

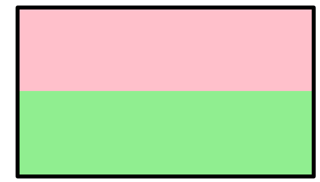


Then the verifier shows the sheet to the prover



A Physical Interactive Proof System

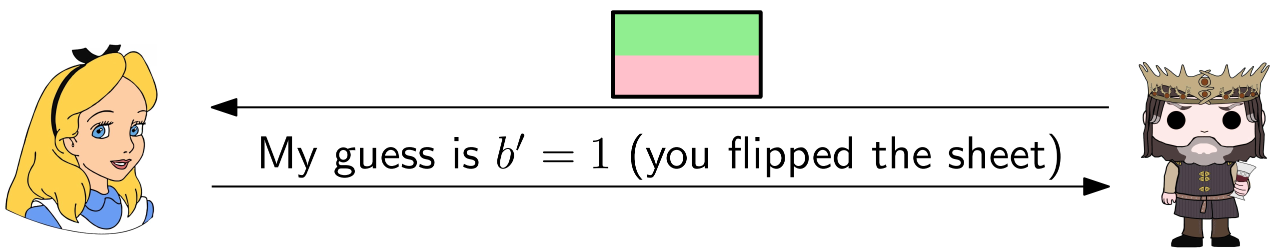
Suppose that the prover needs to convince a colorbrind verifier that this sheet of paper contains multiple colors:



The verifier secretly chooses a random bit b

- If $b = 0$ the verifier keeps the sheet in the original orientation 
- If $b = 1$ the verifier flips the sheet upside down 

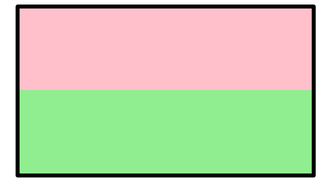
Then the verifier shows the sheet to the prover



The prover observes the sheet and replies with a guess b' about b

A Physical Interactive Proof System

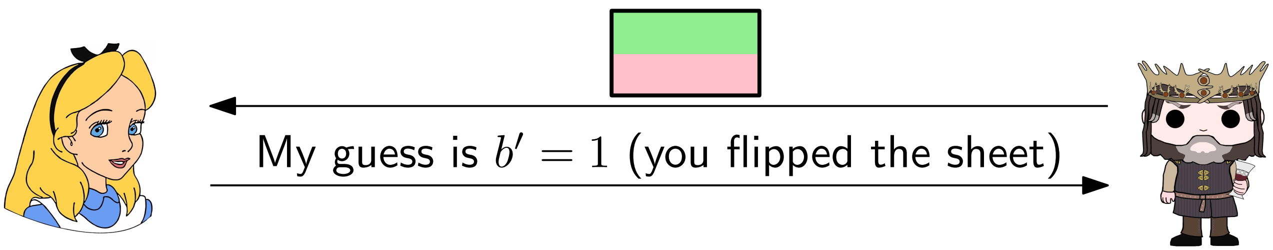
Suppose that the prover needs to convince a colorbrind verifier that this sheet of paper contains multiple colors:



The verifier secretly chooses a random bit b

- If $b = 0$ the verifier keeps the sheet in the original orientation 
- If $b = 1$ the verifier flips the sheet upside down 

Then the verifier shows the sheet to the prover

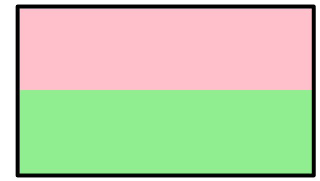


The prover observes the sheet and replies with a guess b' about b

The verifier accepts the proof if $b' = b$ and rejects it if $b' \neq b$

A Physical Interactive Proof System

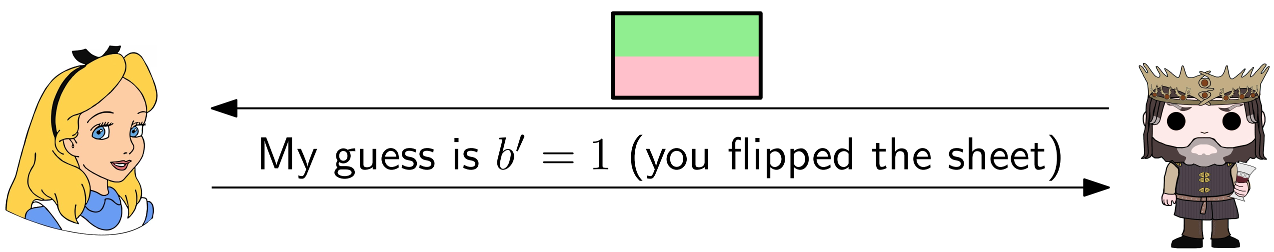
Suppose that the prover needs to convince a colorbrind verifier that this sheet of paper contains multiple colors:



The verifier secretly chooses a random bit b

- If $b = 0$ the verifier keeps the sheet in the original orientation 
- If $b = 1$ the verifier flips the sheet upside down 

Then the verifier shows the sheet to the prover



Honest prover (completeness):

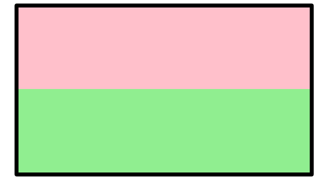
$\Pr[\text{the verifier accepts}] = 1$

The prover observes the sheet and replies with a guess b' about b

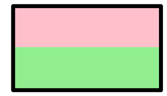
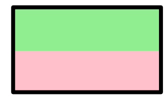
The verifier accepts the proof if $b' = b$ and rejects it if $b' \neq b$

A Physical Interactive Proof System

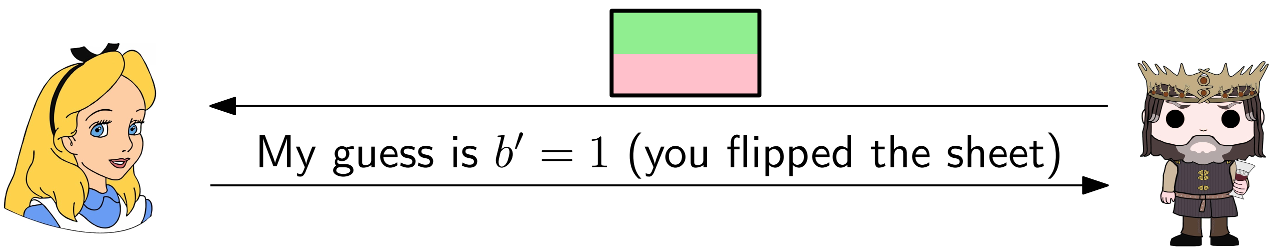
Suppose that the prover needs to convince a colorbrind verifier that this sheet of paper contains multiple colors:



The verifier secretly chooses a random bit b

- If $b = 0$ the verifier keeps the sheet in the original orientation 
- If $b = 1$ the verifier flips the sheet upside down 

Then the verifier shows the sheet to the prover



The prover observes the sheet and replies with a guess b' about b

The verifier accepts the proof if $b' = b$ and rejects it if $b' \neq b$

Honest prover (completeness):

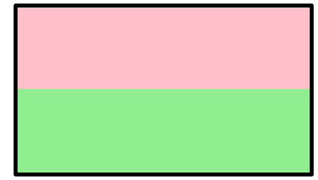
$\Pr[\text{the verifier accepts}] = 1$

Cheating prover (soundness):
(the sheet is monochromatic)

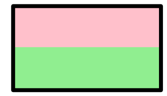
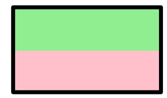
$\Pr[\text{the verifier accepts}] = \frac{1}{2}$

A Physical Interactive Proof System

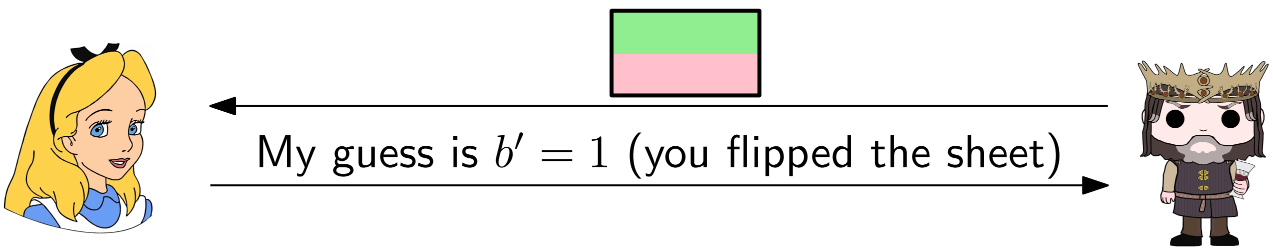
Suppose that the prover needs to convince a colorbrind verifier that this sheet of paper contains multiple colors:



The verifier secretly chooses a random bit b

- If $b = 0$ the verifier keeps the sheet in the original orientation 
- If $b = 1$ the verifier flips the sheet upside down 

Then the verifier shows the sheet to the prover



The prover observes the sheet and replies with a guess b' about b

The verifier accepts the proof if $b' = b$ and rejects it if $b' \neq b$

What if the verifier does not like those odds?

Honest prover (completeness):

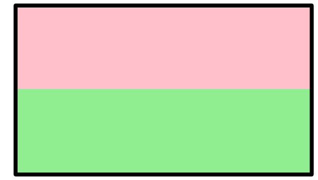
$\Pr[\text{the verifier accepts}] = 1$

Cheating prover (soundness):
(the sheet is monochromatic)

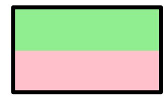
$\Pr[\text{the verifier accepts}] = \frac{1}{2}$

A Physical Interactive Proof System

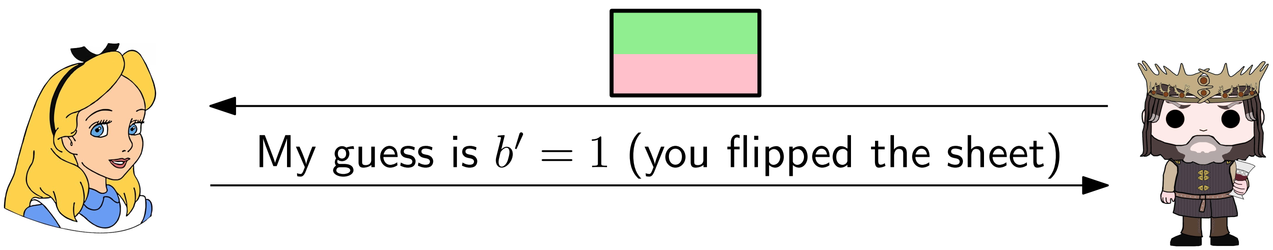
Suppose that the prover needs to convince a colorbrind verifier that this sheet of paper contains multiple colors:



The verifier secretly chooses a random bit b

- If $b = 0$ the verifier keeps the sheet in the original orientation 
- If $b = 1$ the verifier flips the sheet upside down 

Then the verifier shows the sheet to the prover



The prover observes the sheet and replies with a guess b' about b

The verifier accepts the proof if $b' = b$ and rejects it if $b' \neq b$

Honest prover (completeness):

$\Pr[\text{the verifier accepts}] = 1$

Cheating prover (soundness):
(the sheet is monochromatic)

$\Pr[\text{the verifier accepts}] =$ ~~$\frac{1}{2}$~~ $\frac{1}{2^k}$

What if the verifier does not like those odds?

Repeat the experiment k times. Accept iff all trials succeed.

Interactive Proof Systems (Formal)

Given a pair of interactive algorithms A, B and a common input x, we write $(A, B)(x)$ to denote the output of B at the end of the interaction

Interactive Proof Systems (Formal)

Given a pair of interactive algorithms A, B and a common input x, we write $(A, B)(x)$ to denote the output of B at the end of the interaction

Definition: A pair of interactive (randomized) algorithms (P, V) is called an **interactive proof system** for language L if V runs in polynomial time and the following two conditions hold:

- **Completeness:** For every $x \in L$, $\Pr[(P, V)(x) = 1] \geq \frac{2}{3}$
- **Soundness:** For every $x \notin L$ and every interactive algorithm A, $\Pr[(A, V)(x) = 1] \leq \frac{1}{3}$

Interactive Proof Systems (Formal)

Given a pair of interactive algorithms A, B and a common input x, we write $(A, B)(x)$ to denote the output of B at the end of the interaction

Definition: A pair of interactive (randomized) algorithms (P, V) is called an **interactive proof system** for language L if V runs in polynomial time and the following two conditions hold:

- **Completeness:** For every $x \in L$, $\Pr[(P, V)(x) = 1] \geq \frac{2}{3}$
- **Soundness:** For every $x \notin L$ and every interactive algorithm A, $\Pr[(A, V)(x) = 1] \leq \frac{1}{3}$

Notice that:

- Completeness only needs to hold for the prover P

Interactive Proof Systems (Formal)

Given a pair of interactive algorithms A, B and a common input x, we write $(A, B)(x)$ to denote the output of B at the end of the interaction

Definition: A pair of interactive (randomized) algorithms (P, V) is called an **interactive proof system** for language L if V runs in polynomial time and the following two conditions hold:

- **Completeness:** For every $x \in L$, $\Pr[(P, V)(x) = 1] \geq \frac{2}{3}$
- **Soundness:** For every $x \notin L$ and every interactive algorithm A, $\Pr[(A, V)(x) = 1] \leq \frac{1}{3}$

Notice that:

- Completeness only needs to hold for the prover P
- Soundness needs to hold regardless of the (cheating) prover

Interactive Proof Systems (Formal)

Given a pair of interactive algorithms A, B and a common input x, we write $(A, B)(x)$ to denote the output of B at the end of the interaction

Definition: A pair of interactive (randomized) algorithms (P, V) is called an **interactive proof system** for language L if V runs in polynomial time and the following two conditions hold:

- **Completeness:** For every $x \in L$, $\Pr[(P, V)(x) = 1] \geq \frac{2}{3}$
- **Soundness:** For every $x \notin L$ and every interactive algorithm A, $\Pr[(A, V)(x) = 1] \leq \frac{1}{3}$

Notice that:

- Completeness only needs to hold for the prover P
- Soundness needs to hold regardless of the (cheating) prover

IP is the class of all languages that admit an interactive proof system

Probability Amplification

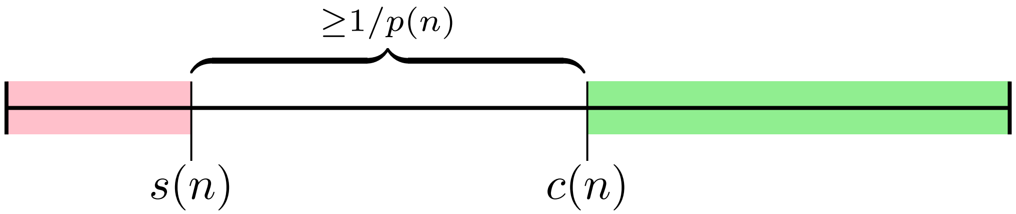
The constants $\frac{2}{3}$ and $\frac{1}{3}$ are arbitrary. Any pair of error probabilities that differ by at least $\frac{1}{p(n)}$, for some polynomial $p(n) > 0$, is enough.

Probability Amplification

The constants $\frac{2}{3}$ and $\frac{1}{3}$ are arbitrary. Any pair of error probabilities that differ by at least $\frac{1}{p(n)}$, for some polynomial $p(n) > 0$, is enough.

Let (P, V) be an interactive proof system such that:

- $\Pr[(P, V)(x) = 1] \geq c(n)$ for $x \in L$
- $\Pr[(P, V)(x) = 1] \leq s(n)$ for $x \notin L$
- $s(n) - c(n) \geq \frac{1}{p(n)}$

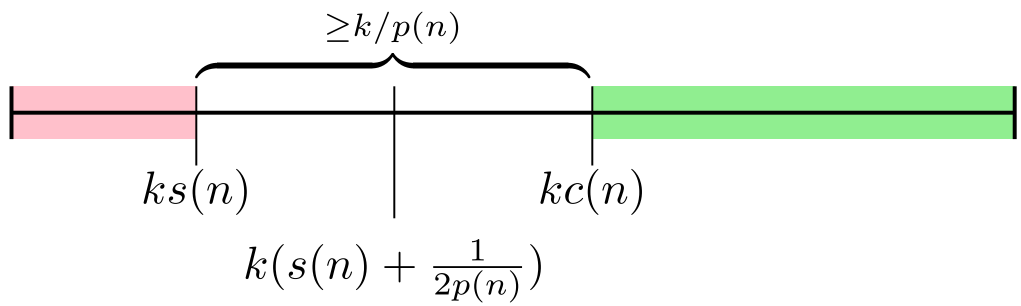


Probability Amplification

The constants $\frac{2}{3}$ and $\frac{1}{3}$ are arbitrary. Any pair of error probabilities that differ by at least $\frac{1}{p(n)}$, for some polynomial $p(n) > 0$, is enough.

Let (P, V) be an interactive proof system such that:

- $\Pr[(P, V)(x) = 1] \geq c(n)$ for $x \in L$
- $\Pr[(P, V)(x) = 1] \leq s(n)$ for $x \notin L$
- $s(n) - c(n) \geq \frac{1}{p(n)}$



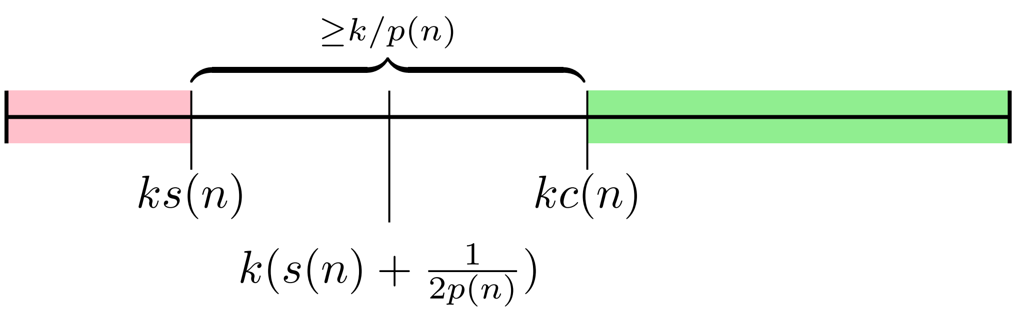
Strategy: Repeat the protocol k times and accept if the $(P, V)(x) = 1$ in at least $k(s(n) + \frac{1}{2p(n)})$ executions

Probability Amplification

The constants $\frac{2}{3}$ and $\frac{1}{3}$ are arbitrary. Any pair of error probabilities that differ by at least $\frac{1}{p(n)}$, for some polynomial $p(n) > 0$, is enough.

Let (P, V) be an interactive proof system such that:

- $\Pr[(P,V)(x) = 1] \geq c(n)$ for $x \in L$
- $\Pr[(P,V)(x) = 1] \leq s(n)$ for $x \notin L$
- $s(n) - c(n) \geq \frac{1}{p(n)}$



Strategy: Repeat the protocol k times and accept if the $(P,V)(x) = 1$ in at least $k(s(n) + \frac{1}{2p(n)})$ executions

Chernoff bound: Let $X = X_1 + X_2 + \ldots$ where the X_is are independent binary random variables and let $\mu = \mathbb{E}[X]$. Then, for any $\delta > 0$:

$$\Pr[X \leq (1-\delta)\mu] \leq e^{-\delta^2 \mu/2}$$

$O(p(n)^3)$ repetitions suffice

Efficiently verifiable proofs

The class NP can be thought of as the set of all claims that admit a short, efficiently verifiable proof

A language $L \subseteq \{0,1\}^*$ is in NP iff there exists a non-deterministic polynomial-time Turing machine T such that $x \in L$ iff (at least one computation path of) $T(x)$ accepts

Efficiently verifiable proofs

The class NP can be thought of as the set of all claims that admit a short, efficiently verifiable proof

A language $L \subseteq \{0,1\}^*$ is in NP iff there exists a non-deterministic polynomial-time Turing machine T such that $x \in L$ iff (at least one computation path of) $T(x)$ accepts

Equivalently: A language: $L \subseteq \{0,1\}^*$ is in NP iff there exists a polynomial-time algorithm $\mathcal{V}$ such that:

- For every $x \in L$, there exists a witness (i.e., a "proof") w with $|w| \in O(\text{poly}(|x|))$ s.t. $\mathcal{V}(x, w) = 1$.
- For every $x \notin L$, there is no witness w s.t. $\mathcal{V}(x, w) = 1$.

Efficiently verifiable proofs

The class NP can be thought of as the set of all claims that admit a short, efficiently verifiable proof

A language $L \subseteq \{0,1\}^*$ is in NP iff there exists a non-deterministic polynomial-time Turing machine T such that $x \in L$ iff (at least one computation path of) $T(x)$ accepts

Equivalently: A language: $L \subseteq \{0,1\}^*$ is in NP iff there exists a polynomial-time algorithm $\mathcal{V}$ such that:

- For every $x \in L$, there exists a witness (i.e., a "proof") w with $|w| \in O(\text{poly}(|x|))$ s.t. $\mathcal{V}(x, w) = 1$.
- For every $x \notin L$, there is no witness w s.t. $\mathcal{V}(x, w) = 1$.

Proof sketch:

$\implies$: A (poynomial-length) accepting computation path of $T(x)$ is a witness w
($\mathcal{V}$ simulates T but already know the "right" non-deterministic choices from w)

Efficiently verifiable proofs

The class NP can be thought of as the set of all claims that admit a short, efficiently verifiable proof

A language $L \subseteq \{0,1\}^*$ is in NP iff there exists a non-deterministic polynomial-time Turing machine T such that $x \in L$ iff (at least one computation path of) $T(x)$ accepts

Equivalently: A language: $L \subseteq \{0,1\}^*$ is in NP iff there exists a polynomial-time algorithm $\mathcal{V}$ such that:

- For every $x \in L$, there exists a witness (i.e., a "proof") w with $|w| \in O(\text{poly}(|x|))$ s.t. $\mathcal{V}(x, w) = 1$.
- For every $x \notin L$, there is no witness w s.t. $\mathcal{V}(x, w) = 1$.

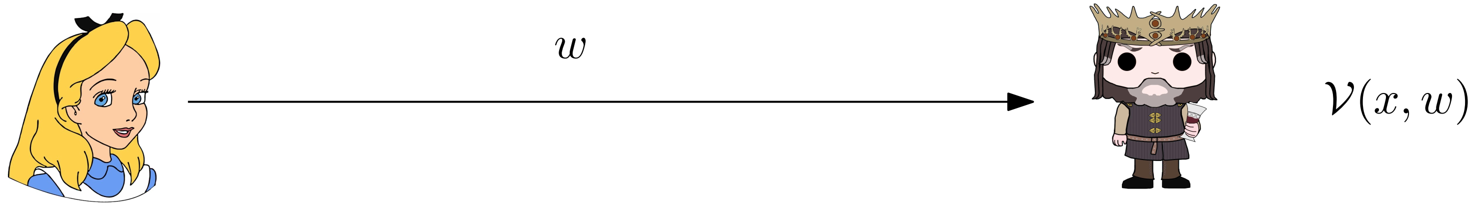
Proof sketch:

$\Longrightarrow$: A (poynomial-length) accepting computation path of $T(x)$ is a witness w
($\mathcal{V}$ simulates T but already know the "right" non-deterministic choices from w)

$\Longleftarrow$: A non-determistic Turing machine T can "guess" the witness w, and then check if $\mathcal{V}(x, w) = 1$

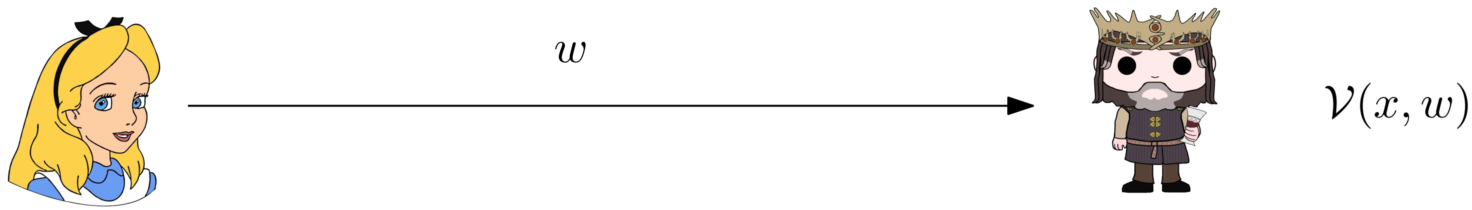
Interactive Proofs for NP and BPP

We immediately have an interactive proof for all languages $L \in \mathsf{NP}$



Interactive Proofs for NP and BPP

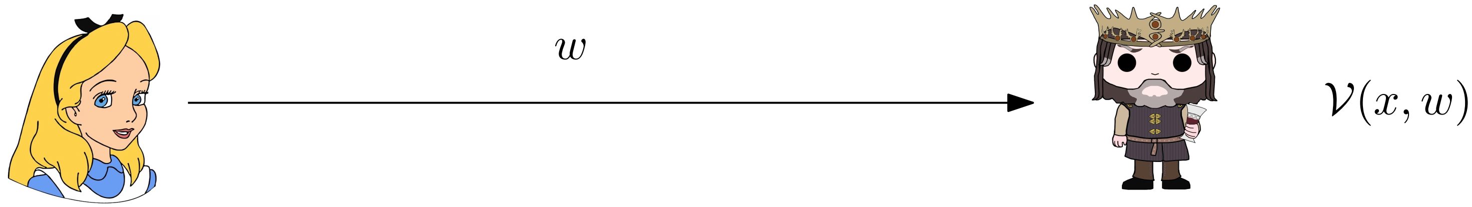
We immediately have an interactive proof for all languages $L \in$ NP



NP contains exactly all languages that admit an interactive proof with a deterministic verifier and in which at most one message is exchanged (from the prover to the verifier)

Interactive Proofs for NP and BPP

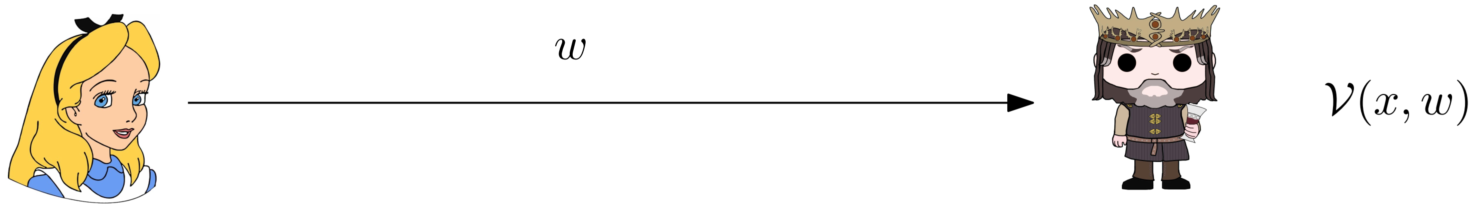
We immediately have an interactive proof for all languages $L \in$ NP   NP $\subseteq$ IP



NP contains exactly all languages that admit an interactive proof with a deterministic verifier and in which at most one message is exchanged (from the prover to the verifier)

Interactive Proofs for NP and BPP

We immediately have an interactive proof for all languages $L \in \mathsf{NP}$ $\mathsf{NP} \subseteq \mathsf{IP}$

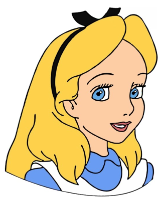


NP contains exactly all languages that admit an interactive proof with a deterministic verifier and in which at most one message is exchanged (from the prover to the verifier)

What if $L \in \mathsf{BPP}$?

Interactive Proofs for NP and BPP

We immediately have an interactive proof for all languages $L \in \mathsf{NP}$ $\qquad$ $\mathsf{NP} \subseteq \mathsf{IP}$



w →

$\mathcal{V}(x, w)$

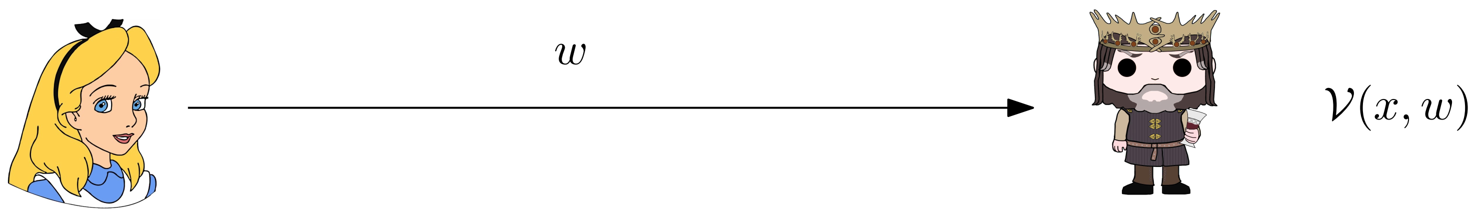
NP contains exactly all languages that admit an interactive proof with a deterministic verifier and in which at most one message is exchanged (from the prover to the verifier)

What if $L \in \mathsf{BPP}$?

- There is a (randomized) polynomial time algorithm $\mathcal{A}(x)$ that decides whether $x \in L$

Interactive Proofs for NP and BPP

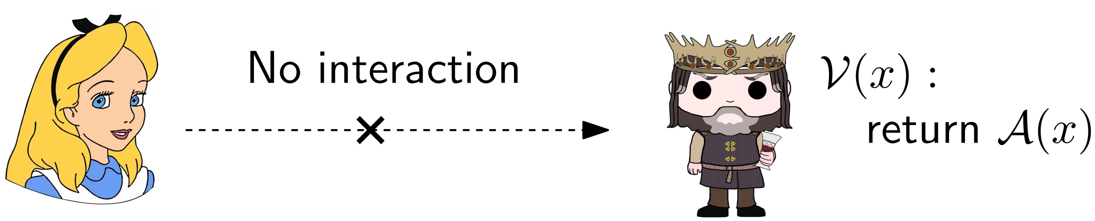
We immediately have an interactive proof for all languages $L \in \text{NP}$ NP $\subseteq$ IP



NP contains exactly all languages that admit an interactive proof with a deterministic verifier and in which at most one message is exchanged (from the prover to the verifier)

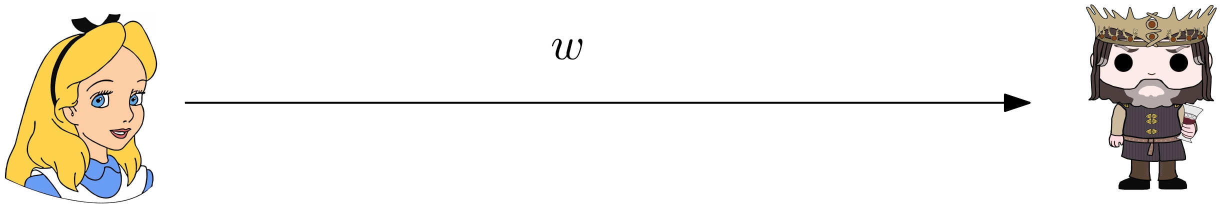
What if $L \in \text{BPP}$?

- There is a (randomized) polynomial time algorithm $\mathcal{A}(x)$ that decides whether $x \in L$
- There is no need for a witness! The verifier can convince itself that the claim is true!
- The verifier ignores the prover and runs $\mathcal{A}(x)$



Interactive Proofs for NP and BPP

We immediately have an interactive proof for all languages $L \in \mathsf{NP}$ — $\mathsf{NP} \subseteq \mathsf{IP}$

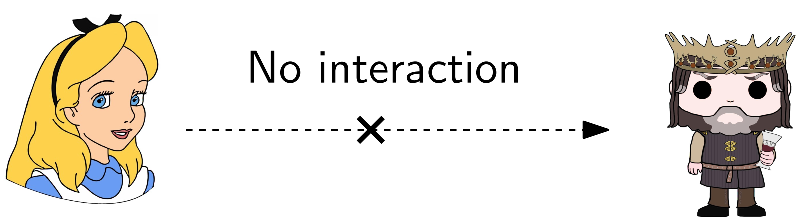


$\mathcal{V}(x, w)$

NP contains exactly all languages that admit an interactive proof with a deterministic verifier and in which at most one message is exchanged (from the prover to the verifier)

What if $L \in \mathsf{BPP}$? — $\mathsf{BPP} \subseteq \mathsf{IP}$

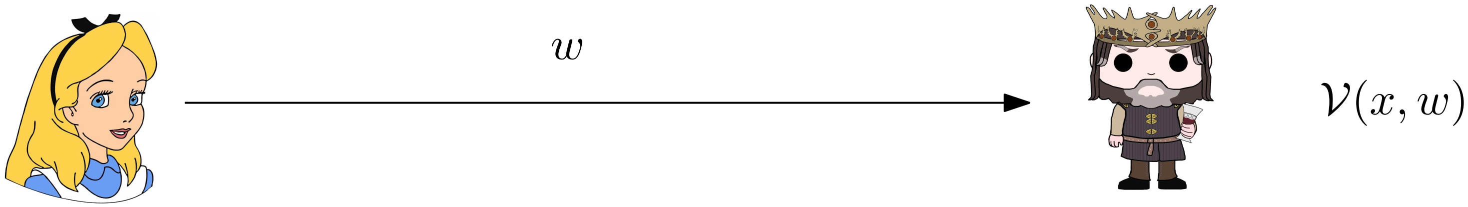
- There is a (randomized) polynomial time algorithm $\mathcal{A}(x)$ that decides whether $x \in L$
- There is no need for a witness! The verifier can convince itself that the claim is true!
- The verifier ignores the prover and runs $\mathcal{A}(x)$



$\mathcal{V}(x)$:
  return $\mathcal{A}(x)$

Interactive Proofs for NP and BPP

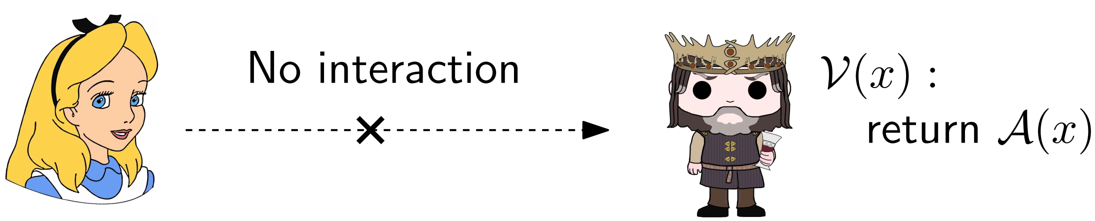
We immediately have an interactive proof for all languages $L \in$ NP NP $\subseteq$ IP



NP contains exactly all languages that admit an interactive proof with a deterministic verifier and in which at most one message is exchanged (from the prover to the verifier)

What if $L \in$ BPP? BPP $\subseteq$ IP

- There is a (randomized) polynomial time algorithm $\mathcal{A}(x)$ that decides whether $x \in L$
- There is no need for a witness! The verifier can convince itself that the claim is true!
- The verifier ignores the prover and runs $\mathcal{A}(x)$



An interactive proof in which the verifier never talks to the prover is **degenerate**

An Interactive Proof for Graph Non-Isomorphism

Let's look at a non-degenerate interactive proof for a problem that is not known to be in $\mathsf{NP} \cup \mathsf{BPP}$

The language L contains all pairs of graphs (G_1, G_2) such that G_1 and G_2 are **not** isomorphic

An Interactive Proof for Graph Non-Isomorphism

Let's look at a non-degenerate interactive proof for a problem that is not known to be in $\mathsf{NP} \cup \mathsf{BPP}$

The language L contains all pairs of graphs (G_1, G_2) such that G_1 and G_2 are **not** isomorphic

Notice: no obvious (short) witness!

An Interactive Proof for Graph Non-Isomorphism

Let's look at a non-degenerate interactive proof for a problem that is not known to be in NP $\cup$ BPP

The language L contains all pairs of graphs (G_1, G_2) such that G_1 and G_2 are **not** isomorphic

Notice: no obvious (short) witness!

If $G = (V, E)$ and π is a permutation on V, we denote by $\pi(G)$ the graph (V, F) where $F = \{(\pi(u), \pi(v)) \mid (u, v) \in E\}$

An Interactive Proof for Graph Non-Isomorphism

Let's look at a non-degenerate interactive proof for a problem that is not known to be in NP $\cup$ BPP

The language L contains all pairs of graphs (G_1, G_2) such that G_1 and G_2 are **not** isomorphic

Notice: no obvious (short) witness!

If $G = (V, E)$ and π is a permutation on V, we denote by $\pi(G)$ the graph (V, F) where $F = \{(\pi(u), \pi(v)) \mid (u, v) \in E\}$

Common input: $x = (G_1, G_2)$ where $G_1 = (V, E_1)$, $G_2 = (V, E_2)$, and $V = \{1, \ldots, n\}$

- The verifier chooses b u.a.r. in $\{1, 2\}$
- The verifier picks a random permutation $\pi : V \to V$ and sends the graph $G' = \pi(G_b)$ to the prover
- The prover checks whether G' is isomorphic to G_1. If so it replies with $b' = 1$, otherwise it replies with $b' = 2$.
- If $b' = b$, the verifier accepts. Otherwise it rejects

An Interactive Proof for Graph Non-Isomorphism

Completeness

If $(G_1, G_2) \in L$ then G_b will be isomorphic to exactly one of G_1 and G_2. The (computationally unbounded) prover always guesses correctly

An Interactive Proof for Graph Non-Isomorphism

Completeness

If $(G_1, G_2) \in L$ then G_b will be isomorphic to exactly one of G_1 and G_2. The (computationally unbounded) prover always guesses correctly

Soundness

Idea: If the input graphs are isomorphic (the prover is cheating), then a random isomorphic copy of one graph will be distributed identically to a random isomorphic copy of the other graph.

If G_1 and G_2, are isomorphic and π is a random permutation then $\Pr[\pi(G_1) = H] = \Pr[\pi(G_2) = H]$

An Interactive Proof for Graph Non-Isomorphism

Completeness

If $(G_1, G_2) \in L$ then G_b will be isomorphic to exactly one of G_1 and G_2. The (computationally unbounded) prover always guesses correctly

Soundness

Idea: If the input graphs are isomorphic (the prover is cheating), then a random isomorphic copy of one graph will be distributed identically to a random isomorphic copy of the other graph.

If G_1 and G_2, are isomorphic and π is a random permutation then $\Pr[\pi(G_1) = H] = \Pr[\pi(G_2) = H]$

For any graph G' isomorphic to G_1 (and G_2):

$\Pr[b = 1 \mid \pi(G_b) = G']$

$\Pr[b = 2 \mid \pi(G_b) = G']$

An Interactive Proof for Graph Non-Isomorphism

Completeness

If $(G_1, G_2) \in L$ then G_b will be isomorphic to exactly one of G_1 and G_2. The (computationally unbounded) prover always guesses correctly

Soundness

Idea: If the input graphs are isomorphic (the prover is cheating), then a random isomorphic copy of one graph will be distributed identically to a random isomorphic copy of the other graph.

If G_1 and G_2, are isomorphic and π is a random permutation then $\Pr[\pi(G_1) = H] = \Pr[\pi(G_2) = H]$

For any graph G' isomorphic to G_1 (and G_2):

$$\Pr[b = 1 \mid \pi(G_b) = G'] = \Pr[\pi(G_b) = G' \mid b = 1] \cdot \frac{\Pr[b=1]}{\Pr[\pi(G_b)=G']}$$

$$\Pr[b = 2 \mid \pi(G_b) = G'] = \Pr[\pi(G_b) = G' \mid b = 2] \cdot \frac{\Pr[b=2]}{\Pr[\pi(G_b)=G']}$$

An Interactive Proof for Graph Non-Isomorphism

Completeness

If $(G_1, G_2) \in L$ then G_b will be isomorphic to exactly one of G_1 and G_2. The (computationally unbounded) prover always guesses correctly

Soundness

Idea: If the input graphs are isomorphic (the prover is cheating), then a random isomorphic copy of one graph will be distributed identically to a random isomorphic copy of the other graph.

If G_1 and G_2, are isomorphic and π is a random permutation then $\Pr[\pi(G_1) = H] = \Pr[\pi(G_2) = H]$

For any graph G' isomorphic to G_1 (and G_2):

$$\Pr[b = 1 \mid \pi(G_b) = G'] = \Pr[\pi(G_b) = G' \mid b = 1] \cdot \frac{\Pr[b=1]}{\Pr[\pi(G_b)=G']} = \Pr[\pi(G_1) = G'] \cdot \frac{\Pr[b=1]}{\Pr[\pi(G_b)=G']}$$

$$\Pr[b = 2 \mid \pi(G_b) = G'] = \Pr[\pi(G_b) = G' \mid b = 2] \cdot \frac{\Pr[b=2]}{\Pr[\pi(G_b)=G']} = \Pr[\pi(G_2) = G'] \cdot \frac{\Pr[b=2]}{\Pr[\pi(G_b)=G']}$$

An Interactive Proof for Graph Non-Isomorphism

Completeness

If $(G_1, G_2) \in L$ then G_b will be isomorphic to exactly one of G_1 and G_2. The (computationally unbounded) prover always guesses correctly

Soundness

Idea: If the input graphs are isomorphic (the prover is cheating), then a random isomorphic copy of one graph will be distributed identically to a random isomorphic copy of the other graph.

> If G_1 and G_2, are isomorphic and π is a random permutation then $\Pr[\pi(G_1)=H]=\Pr[\pi(G_2)=H]$

For any graph G' isomorphic to G_1 (and G_2):

$$\Pr[b = 1 \mid \pi(G_b) = G'] \ = \Pr[\pi(G_b) = G' \mid b = 1] \cdot \frac{\Pr[b=1]}{\Pr[\pi(G_b)=G']} \ = \Pr[\pi(G_1) = G'] \cdot \frac{\Pr[b=1]}{\Pr[\pi(G_b)=G']}$$

$$\Pr[b = 2 \mid \pi(G_b) = G'] \ = \Pr[\pi(G_b) = G' \mid b = 2] \cdot \frac{\Pr[b=2]}{\Pr[\pi(G_b)=G']} \ = \Pr[\pi(G_2) = G'] \cdot \frac{\Pr[b=2]}{\Pr[\pi(G_b)=G']}$$

An Interactive Proof for Graph Non-Isomorphism

Completeness

If $(G_1, G_2) \in L$ then G_b will be isomorphic to exactly one of G_1 and G_2. The (computationally unbounded) prover always guesses correctly

Soundness

Idea: If the input graphs are isomorphic (the prover is cheating), then a random isomorphic copy of one graph will be distributed identically to a random isomorphic copy of the other graph.

> If G_1 and G_2, are isomorphic and π is a random permutation then $\Pr[\pi(G_1) = H] = \Pr[\pi(G_2) = H]$

For any graph G' isomorphic to G_1 (and G_2):

$\Pr[b = 1 \mid \pi(G_b) = G'] \; = \Pr[b = 2 \mid \pi(G_b) = G']$

An Interactive Proof for Graph Non-Isomorphism

Completeness

If $(G_1, G_2) \in L$ then G_b will be isomorphic to exactly one of G_1 and G_2. The (computationally unbounded) prover always guesses correctly

Soundness

Idea: If the input graphs are isomorphic (the prover is cheating), then a random isomorphic copy of one graph will be distributed identically to a random isomorphic copy of the other graph.

If G_1 and G_2, are isomorphic and π is a random permutation then $\Pr[\pi(G_1) = H] = \Pr[\pi(G_2) = H]$

For any graph G' isomorphic to G_1 (and G_2):

$$\Pr[b = 1 \mid \pi(G_b) = G'] \; = \Pr[b = 2 \mid \pi(G_b) = G'] \; = \frac{1}{2}$$

An Interactive Proof for Graph Non-Isomorphism

Completeness

If $(G_1, G_2) \in L$ then G_b will be isomorphic to exactly one of G_1 and G_2. The (computationally unbounded) prover always guesses correctly

Soundness

Idea: If the input graphs are isomorphic (the prover is cheating), then a random isomorphic copy of one graph will be distributed identically to a random isomorphic copy of the other graph.

If G_1 and G_2, are isomorphic and π is a random permutation then $\Pr[\pi(G_1) = H] = \Pr[\pi(G_2) = H]$

For any graph G' isomorphic to G_1 (and G_2):

$\Pr[b = 1 \mid \pi(G_b) = G'] \ = \Pr[b = 2 \mid \pi(G_b) = G'] \ = \frac{1}{2}$

Let R be the (possibily randomized) process used by the prover to compute the reply to the verifier

The verifier accepts with probability:

$\Pr[R(G_b) = b]$

An Interactive Proof for Graph Non-Isomorphism

Completeness

If $(G_1, G_2) \in L$ then G_b will be isomorphic to exactly one of G_1 and G_2. The (computationally unbounded) prover always guesses correctly

Soundness

Idea: If the input graphs are isomorphic (the prover is cheating), then a random isomorphic copy of one graph will be distributed identically to a random isomorphic copy of the other graph.

If G_1 and G_2, are isomorphic and π is a random permutation then $\Pr[\pi(G_1)=H]=\Pr[\pi(G_2)=H]$

For any graph G' isomorphic to G_1 (and G_2):

$$\Pr[b = 1 \mid \pi(G_b) = G'] \ = \Pr[b = 2 \mid \pi(G_b) = G'] \ = \tfrac{1}{2}$$

Let R be the (possibily randomized) process used by the prover to compute the reply to the verifier

The verifier accepts with probability:

$$\Pr[R(G_b) = b] \ = \sum_{G'} \Pr[R(G') = b \mid \pi(G_b) = G'] \cdot \Pr[\pi(G_b) = G']$$

An Interactive Proof for Graph Non-Isomorphism

Completeness

If $(G_1, G_2) \in L$ then G_b will be isomorphic to exactly one of G_1 and G_2. The (computationally unbounded) prover always guesses correctly

Soundness

Idea: If the input graphs are isomorphic (the prover is cheating), then a random isomorphic copy of one graph will be distributed identically to a random isomorphic copy of the other graph.

If G_1 and G_2, are isomorphic and π is a random permutation then $\Pr[\pi(G_1) = H] = \Pr[\pi(G_2) = H]$

For any graph G' isomorphic to G_1 (and G_2):

$\Pr[b = 1 \mid \pi(G_b) = G'] \ = \Pr[b = 2 \mid \pi(G_b) = G'] \ = \frac{1}{2}$

Let R be the (possibily randomized) process used by the prover to compute the reply to the verifier

The verifier accepts with probability:

$\Pr[R(G_b) = b] \ = \sum_{G'} \Pr[R(G') = b \mid \pi(G_b) = G'] \cdot \Pr[\pi(G_b) = G']$

$\Pr[R(G') = b \mid \pi(G_b) = G']$

An Interactive Proof for Graph Non-Isomorphism

Completeness

If $(G_1, G_2) \in L$ then G_b will be isomorphic to exactly one of G_1 and G_2. The (computationally unbounded) prover always guesses correctly

Soundness

Idea: If the input graphs are isomorphic (the prover is cheating), then a random isomorphic copy of one graph will be distributed identically to a random isomorphic copy of the other graph.

If G_1 and G_2, are isomorphic and π is a random permutation then $\Pr[\pi(G_1) = H] = \Pr[\pi(G_2) = H]$

For any graph G' isomorphic to G_1 (and G_2):

$$\Pr[b = 1 \mid \pi(G_b) = G'] \ = \Pr[b = 2 \mid \pi(G_b) = G'] \ = \frac{1}{2}$$

Let R be the (possibily randomized) process used by the prover to compute the reply to the verifier

The verifier accepts with probability:

$$\Pr[R(G_b) = b] \ = \sum_{G'} \Pr[R(G') = b \mid \pi(G_b) = G'] \cdot \Pr[\pi(G_b) = G']$$

$$\Pr[R(G') = b \mid \pi(G_b) = G'] \leq \sum_{b^* \in \{1,2\}} \Pr[R(G') = b^* \wedge b = b^* \mid \pi(G_b) = G']$$

An Interactive Proof for Graph Non-Isomorphism

Completeness

If $(G_1, G_2) \in L$ then G_b will be isomorphic to exactly one of G_1 and G_2. The (computationally unbounded) prover always guesses correctly

Soundness

Idea: If the input graphs are isomorphic (the prover is cheating), then a random isomorphic copy of one graph will be distributed identically to a random isomorphic copy of the other graph.

If G_1 and G_2, are isomorphic and π is a random permutation then $\Pr[\pi(G_1) = H] = \Pr[\pi(G_2) = H]$

For any graph G' isomorphic to G_1 (and G_2):

$$\Pr[b = 1 \mid \pi(G_b) = G'] \;= \Pr[b = 2 \mid \pi(G_b) = G'] \;= \tfrac{1}{2}$$

Let R be the (possibily randomized) process used by the prover to compute the reply to the verifier

The verifier accepts with probability:

$$\Pr[R(G_b) = b] \;= \sum_{G'} \Pr[R(G') = b \mid \pi(G_b) = G'] \cdot \Pr[\pi(G_b) = G']$$

$$\Pr[R(G') = b \mid \pi(G_b) = G'] \leq \sum_{b^* \in \{1,2\}} \Pr[R(G') = b^* \mid b = b^* \wedge \pi(G_b) = G'] \cdot \Pr[b = b^* \mid \pi(G_b) = G']$$

An Interactive Proof for Graph Non-Isomorphism

Completeness

If $(G_1, G_2) \in L$ then G_b will be isomorphic to exactly one of G_1 and G_2. The (computationally unbounded) prover always guesses correctly

Soundness

Idea: If the input graphs are isomorphic (the prover is cheating), then a random isomorphic copy of one graph will be distributed identically to a random isomorphic copy of the other graph.

If G_1 and G_2, are isomorphic and π is a random permutation then $\Pr[\pi(G_1) = H] = \Pr[\pi(G_2) = H]$

For any graph G' isomorphic to G_1 (and G_2):

$$\Pr[b = 1 \mid \pi(G_b) = G'] \ = \Pr[b = 2 \mid \pi(G_b) = G'] \ = \frac{1}{2}$$

Let R be the (possibily randomized) process used by the prover to compute the reply to the verifier

The verifier accepts with probability:

$$\Pr[R(G_b) = b] \ = \sum_{G'} \Pr[R(G') = b \mid \pi(G_b) = G'] \cdot \Pr[\pi(G_b) = G']$$

$$\Pr[R(G') = b \mid \pi(G_b) = G'] \leq \sum_{b^* \in \{1,2\}} \Pr[R(G') = b^*] \cdot \frac{1}{2}$$

An Interactive Proof for Graph Non-Isomorphism

Completeness

If $(G_1, G_2) \in L$ then G_b will be isomorphic to exactly one of G_1 and G_2. The (computationally unbounded) prover always guesses correctly

Soundness

Idea: If the input graphs are isomorphic (the prover is cheating), then a random isomorphic copy of one graph will be distributed identically to a random isomorphic copy of the other graph.

If G_1 and G_2, are isomorphic and π is a random permutation then $\Pr[\pi(G_1)=H]=\Pr[\pi(G_2)=H]$

For any graph G' isomorphic to G_1 (and G_2):

$$\Pr[b = 1 \mid \pi(G_b) = G'] \; = \Pr[b = 2 \mid \pi(G_b) = G'] \; = \tfrac{1}{2}$$

Let R be the (possibily randomized) process used by the prover to compute the reply to the verifier

The verifier accepts with probability:

$$\Pr[R(G_b) = b] \; = \sum_{G'} \Pr[R(G') = b \mid \pi(G_b) = G'] \cdot \Pr[\pi(G_b) = G']$$

$$\Pr[R(G') = b \mid \pi(G_b) = G'] \leq \tfrac{1}{2}$$

An Interactive Proof for Graph Non-Isomorphism

Completeness

If $(G_1, G_2) \in L$ then G_b will be isomorphic to exactly one of G_1 and G_2. The (computationally unbounded) prover always guesses correctly

Soundness

Idea: If the input graphs are isomorphic (the prover is cheating), then a random isomorphic copy of one graph will be distributed identically to a random isomorphic copy of the other graph.

> If G_1 and G_2, are isomorphic and π is a random permutation then $\Pr[\pi(G_1)=H]=\Pr[\pi(G_2)=H]$

For any graph G' isomorphic to G_1 (and G_2):

$\Pr[b = 1 \mid \pi(G_b) = G'] \;= \Pr[b = 2 \mid \pi(G_b) = G'] \;= \frac{1}{2}$

Let R be the (possibily randomized) process used by the prover to compute the reply to the verifier

The verifier accepts with probability:

$\Pr[R(G_b) = b] \;= \sum_{G'} \Pr[R(G') = b \mid \pi(G_b) = G'] \cdot \Pr[\pi(G_b) = G'] \;\leq \sum_{G'} \frac{1}{2} \cdot \Pr[\pi(G_b) = G']$

$\Pr[R(G') = b \mid \pi(G_b) = G'] \leq \frac{1}{2}$

An Interactive Proof for Graph Non-Isomorphism

Completeness

If $(G_1, G_2) \in L$ then G_b will be isomorphic to exactly one of G_1 and G_2. The (computationally unbounded) prover always guesses correctly

Soundness

Idea: If the input graphs are isomorphic (the prover is cheating), then a random isomorphic copy of one graph will be distributed identically to a random isomorphic copy of the other graph.

If G_1 and G_2, are isomorphic and π is a random permutation then $\Pr[\pi(G_1)=H]=\Pr[\pi(G_2)=H]$

For any graph G' isomorphic to G_1 (and G_2):

$$\Pr[b = 1 \mid \pi(G_b) = G'] \;= \Pr[b = 2 \mid \pi(G_b) = G'] \;= \tfrac{1}{2}$$

Let R be the (possibily randomized) process used by the prover to compute the reply to the verifier

The verifier accepts with probability:

$$\Pr[R(G_b) = b] \;= \sum_{G'} \Pr[R(G') = b \mid \pi(G_b) = G'] \cdot \Pr[\pi(G_b) = G'] \;\leq \sum_{G'} \tfrac{1}{2} \cdot \Pr[\pi(G_b) = G'] = \tfrac{1}{2}$$

$$\Pr[R(G') = b \mid \pi(G_b) = G'] \leq \tfrac{1}{2}$$

An Interactive Proof for Graph Non-Isomorphism

Completeness

If $(G_1, G_2) \in L$ then G_b will be isomorphic to exactly one of G_1 and G_2. The (computationally unbounded) prover always guesses correctly

Soundness

Idea: If the input graphs are isomorphic (the prover is cheating), then a random isomorphic copy of one graph will be distributed identically to a random isomorphic copy of the other graph.

If G_1 and G_2, are isomorphic and π is a random permutation then $\Pr[\pi(G_1)=H]=\Pr[\pi(G_2)=H]$

For any graph G' isomorphic to G_1 (and G_2):

$$\Pr[b = 1 \mid \pi(G_b) = G'] \; = \Pr[b = 2 \mid \pi(G_b) = G'] \; = \tfrac{1}{2}$$

Let R be the (possibily randomized) process used by the prover to compute the reply to the verifier

The verifier accepts with probability:

$$\Pr[R(G_b) = b] \; = \sum_{G'} \Pr[R(G') = b \mid \pi(G_b) = G'] \cdot \Pr[\pi(G_b) = G'] \; \leq \sum_{G'} \tfrac{1}{2} \cdot \Pr[\pi(G_b) = G'] = \tfrac{1}{2}$$

$\Pr[R(G') = b \mid \pi(G_b) = G'] \leq \frac{1}{2}$ **Use probability amplification** □