15-251
Great Theoretical Ideas in Computer Science

Lecture 28:
A Computational Lens on Proofs

Proof. Define f_{ij} as in (5). As f is symmetric, we only need to consider f_{12}.

$$\mathbf{E}\left[f_{12}^2\right] = \mathbf{E}_{x_3\ldots x_n}\left[\frac{1}{4}\cdot\left(f_{12}^2(00x_3\ldots x_n) + f_{12}^2(01x_3\ldots x_n) + f_{12}^2(10x_3\ldots x_n) + f_{12}^2(11x_3\ldots x_n)\right)\right]$$
$$= \frac{1}{4}\mathbf{E}_{x_3\ldots x_n}\left[(f(00x_3\ldots x_n) - f(11x_3\ldots x_n))^2 + (f(11x_3\ldots x_n) - f(00x_3\ldots x_n))^2\right]$$
$$\geq \frac{1}{2}\left(\binom{n-2}{r_0-1}\cdot 2^{-(n-2)}\cdot 4 + \binom{n-2}{n-r_1-1}\cdot 2^{-(n-2)}\cdot 4\right)$$
$$= 8\cdot\left(\frac{(n-r_0+1)(n-r_0)}{n(n-1)}\cdot\binom{n}{r_0-1} + \frac{(n-r_1+1)(n-r_1)}{n(n-1)}\cdot\binom{n}{r_1-1}\right)2^{-n}.$$

Inequality (6) follows by applying Lemma 2.2.

In order to establish inequality (7), we show a lower bound on the principal Fourier coefficient of f:

$$\widehat{f}(\emptyset) \geq 1 - 2\left(\sum_{s<r_0}\binom{n}{s} + \sum_{s>n-r_1}\binom{n}{s}\right)2^{-n},$$

which implies that

$$\widehat{f}(\emptyset)^2 \geq 1 - 4\cdot\left(\sum_{s<r_0}\binom{n}{s} + \sum_{s<r_1}\binom{n}{s}\right)2^{-n}.$$

□

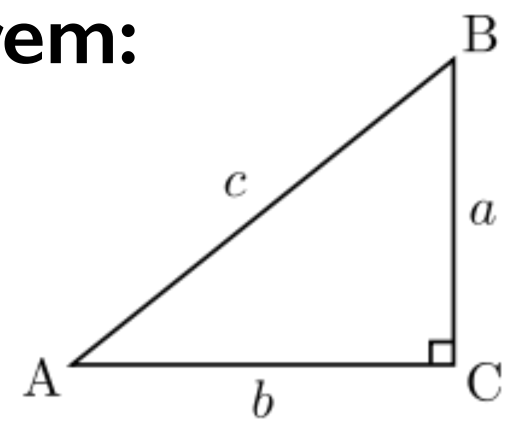
December 6th, 2016

Evolution of “proof”

First there was GORM

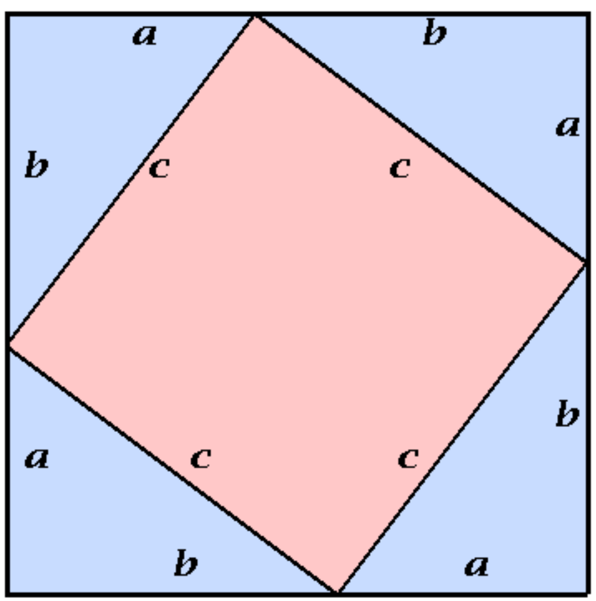
GORM = Good Old Regular Mathematics

Pythagoras's Theorem:



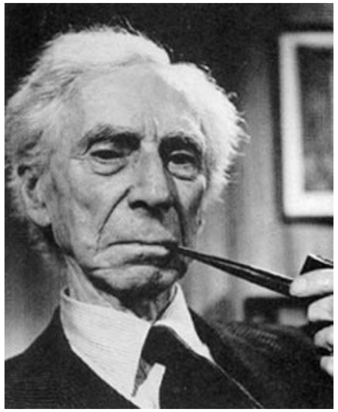
$$a^2 + b^2 = c^2$$

Proof:



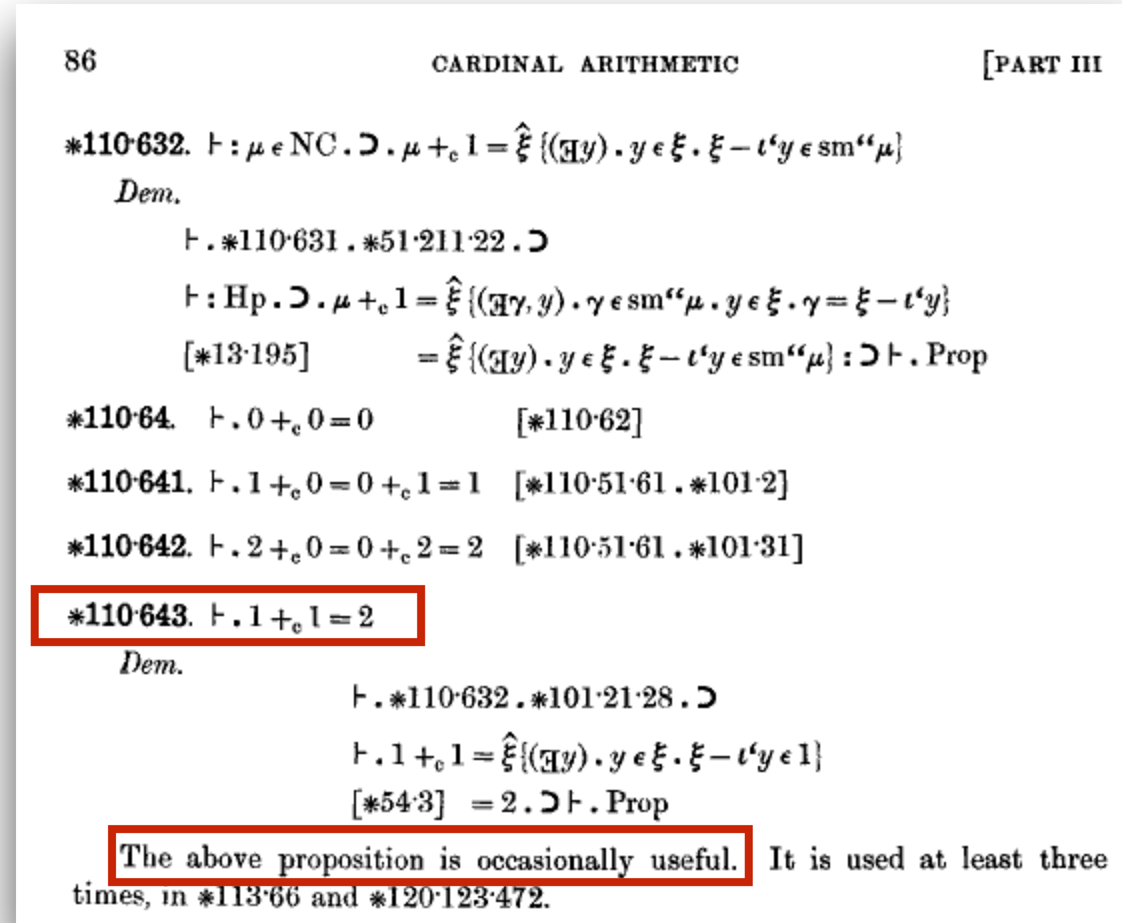
$$(a + b)^2 = a^2 + 2ab + b^2$$

Then there was Russell



Russell and others worked on formalizing GORM proofs.

Principia Mathematica Volume 2



86 CARDINAL ARITHMETIC [PART III

$*110\cdot632.\ \vdash : \mu \in \mathrm{NC} \,.\, \supset .\, \mu +_c 1 = \hat{\xi}\{(\exists y) \,.\, y \in \xi \,.\, \xi - \iota`y \in \mathrm{sm}``\mu\}$

Dem.

$\vdash .\, *110\cdot631 \,.\, *51\cdot211\cdot22 \,.\, \supset$

$\vdash : \mathrm{Hp} \,.\, \supset .\, \mu +_c 1 = \hat{\xi}\{(\exists \gamma, y) \,.\, \gamma \in \mathrm{sm}``\mu \,.\, y \in \xi \,.\, \gamma = \xi - \iota`y\}$

$[*13\cdot195] \quad = \hat{\xi}\{(\exists y) \,.\, y \in \xi \,.\, \xi - \iota`y \in \mathrm{sm}``\mu\} : \supset \vdash .\, \mathrm{Prop}$

$*110\cdot64.\ \vdash .\, 0 +_c 0 = 0 \quad [*110\cdot62]$

$*110\cdot641.\ \vdash .\, 1 +_c 0 = 0 +_c 1 = 1 \quad [*110\cdot51\cdot61 \,.\, *101\cdot2]$

$*110\cdot642.\ \vdash .\, 2 +_c 0 = 0 +_c 2 = 2 \quad [*110\cdot51\cdot61 \,.\, *101\cdot31]$

$*110\cdot643.\ \vdash .\, 1 +_c 1 = 2$

Dem.

$\vdash .\, *110\cdot632 \,.\, *101\cdot21\cdot28 \,.\, \supset$

$\vdash .\, 1 +_c 1 = \hat{\xi}\{(\exists y) \,.\, y \in \xi \,.\, \xi - \iota`y \in 1\}$

$[*54\cdot3] \quad = 2 \,.\, \supset \vdash .\, \mathrm{Prop}$

The above proposition is occasionally useful. It is used at least three times, in *113·66 and *120·123·472.

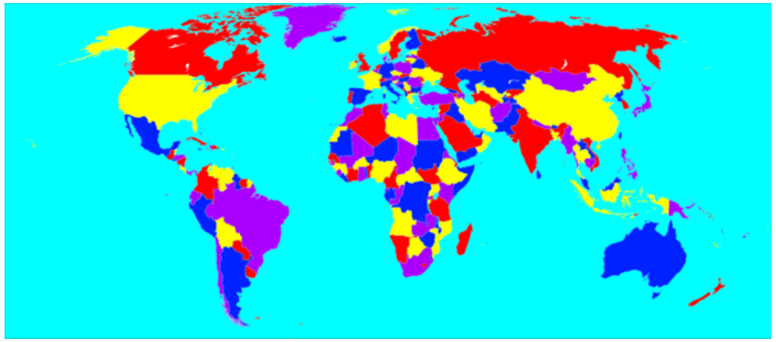
This meant proofs could be found mechanically.
And could be verified mechanically.

Then there were computers

All this played a key role in the birth of computer science.

Computers themselves can find proofs.
(automated theorem provers)

Computers can help us find proofs
(e.g. 4-Color Theorem)



Are these really proofs?

And now…

Thanks to computer science, a "proof" can be:

- randomized
- interactive
- zero-knowledge
- spot-checkable

Original goal of a "proof":

explain and understand a truth.

Now?

Review of NP

Definition:

A language A is in **NP** if

- there is a polynomial time TM V
- a polynomial p

such that for all x:

$$x \in A \iff \exists u \text{ with } |u| \leq p(|x|) \text{ s.t. } V(x,u) = 1$$

"$x \in A$ iff there is a poly-length proof u that is verifiable by a poly-time algorithm."

NP: A game between a Prover and a Verifier

Verifier

poly-time
skeptical

Prover

omniscient
untrustworthy

Given some input x (known both to **Verifier** and **Prover**)

Prover wants to convince **Verifier** that $x \in A$.

Prover cooks up a "proof" u and sends it to **Verifier**.

Verifier (in poly-time), should be able to tell if the proof is legit.

NP: A game between a Prover and a Verifier

Verifier

poly-time
skeptical

Prover

omniscient
untrustworthy

"Completeness"

If $x \in A$, there must be some poly-length proof u that convinces the **Verifier**.

"Soundness"

If $x \notin A$, no matter what "proof" **Prover** gives, **Verifier** should detect the lie.

NP: A game between a Prover and a Verifier

Verifier

poly-time
skeptical

Prover

omniscient
untrustworthy

If we have a protocl for A that is **complete** and **sound**:

$$A \in \textbf{NP}.$$

Limitations of **NP**

Many languages are in **NP**.

SAT, 3SAT, CLIQUE, MAX-CUT, VERTEX-COVER, SUDOKU, THEOREM-PROVING, 3COL, …

Anything not known to be in **NP** ?

Consider the complement of 3SAT:

Given an <u>unsatisfiable</u> 3SAT formula,
how can the Prover prove it is unsatisfiable???

i.e. is the complement of 3SAT in **NP**?

How can we generalize the **NP** setting?

NP setting seems too weak for this purpose.

Also, people use more general ways of convincing each other of the validity of statements.

- Make the protocol **interactive**.

  You can show interaction doesn't really change the model.

- Make the verifier **probabilistic**.

  We don't think randomization by itself adds more power.

But, magic happens when you combine the two.

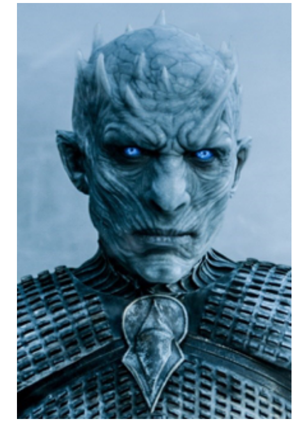
Power of Interaction + Randomization

Coke vs Pepsi Challenge

<u>Claim</u>: I can taste the difference between *Coke* and *Pepsi*.

How can I prove this to you?

Coke vs Pepsi



Choose *Coke* or *Pepsi* at random.

a challenge

Send it to me.

I taste it.

"*Coke*"

Give you an answer.

a response to the challenge

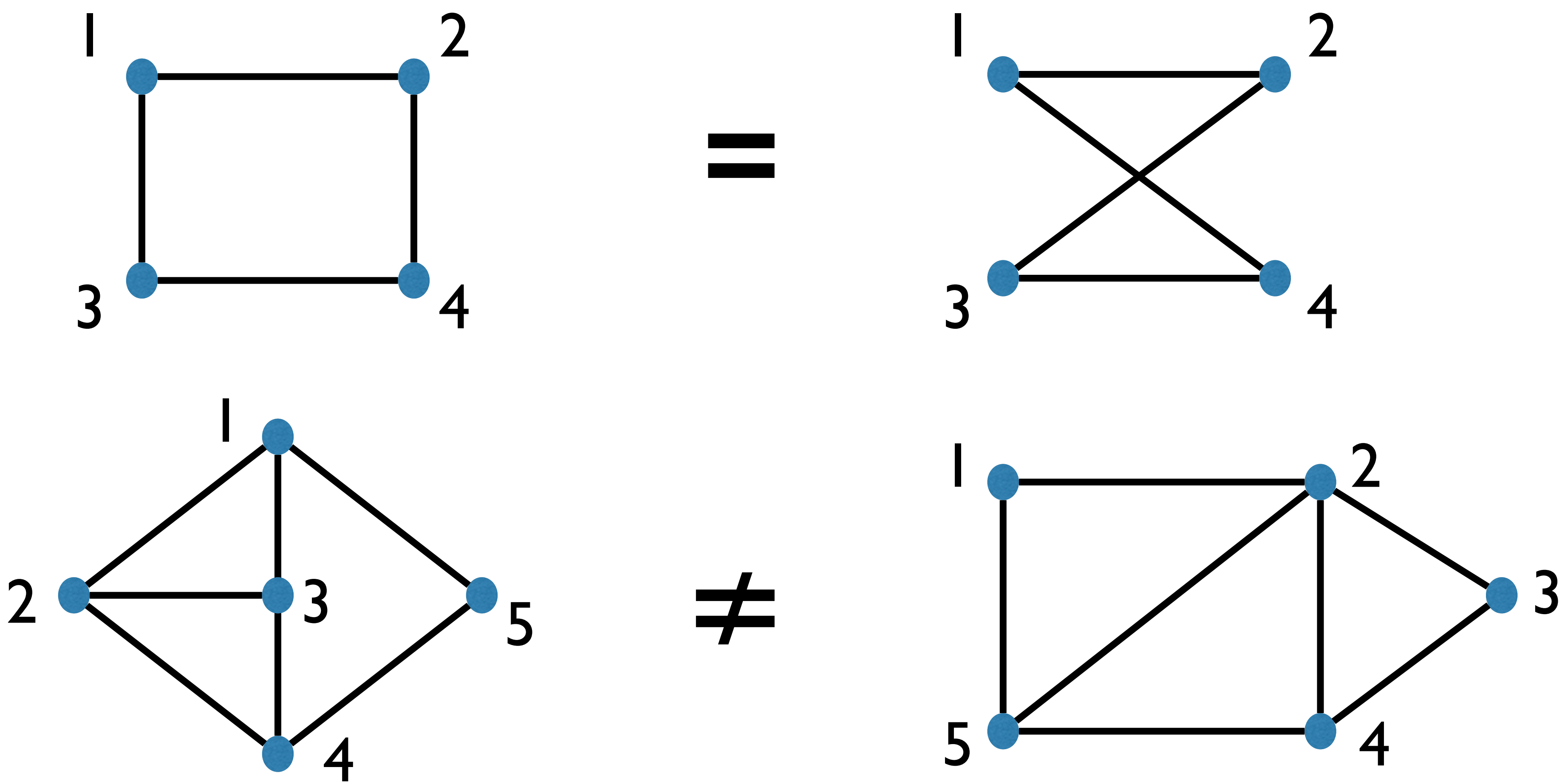
Repeat

Graph Isomorphism Problem

Given two graphs G_1, G_2, are they isomorphic?

i.e., is there a permutation π of the vertices such that

$$\pi(G_1) = G_2$$



Graph Isomorphism Problem

Is Graph Isomorphism in **NP**?

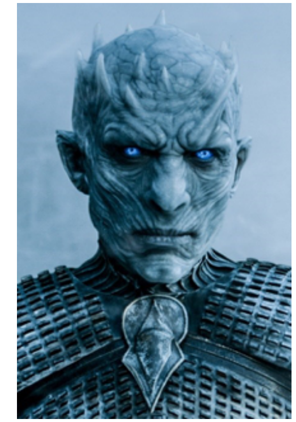
Sure! A good proof is the permutation of the vertices.

Is Graph Non-isomorphism in **NP**?

No one knows!

But there is a simple randomized interactive proof.

Interactive Proof for Graph Non-isomorphism



$\langle G_1, G_2 \rangle$

Pick at random $i \in \{1, 2\}$

Choose a permutation π of vertices at random.

$\pi(G_i)$ →

a challenge

← j

a response to the challenge

Accept if $i = j$

The complexity class **IP** (Interactive Proof)

We say that a language A is in **IP** if:

- there is a probabilistic poly-time **Verifier** 
- there is a computationally unbounded **Prover** 

challenges and responses

(poly rounds)

"Completeness"

If $x \in A$, **Verifier** accepts with prob. at least 2/3.

"Soundness"

If $x \notin A$, **Verifier** accepts with prob. at most 1/3.

How big is **IP** ?

Clearly **NP** $\subseteq$ **IP**.

Is $\overline{\mathrm{3SAT}}$ in **IP**?

Yes!

The complement of any language in **NP** is in **IP**:

$$\mathbf{coNP} \subseteq \mathbf{IP}$$

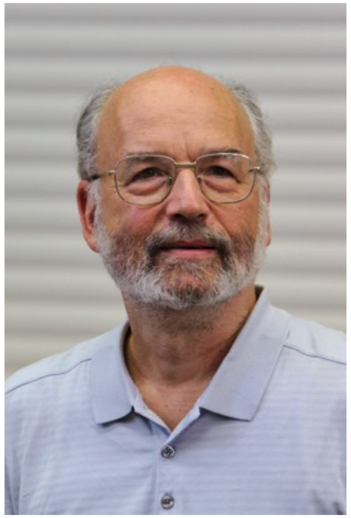
How big is **IP** ?

So how powerful are interactive proofs?

How big is **IP**?

Theorem:

IP = PSPACE



Adi Shamir

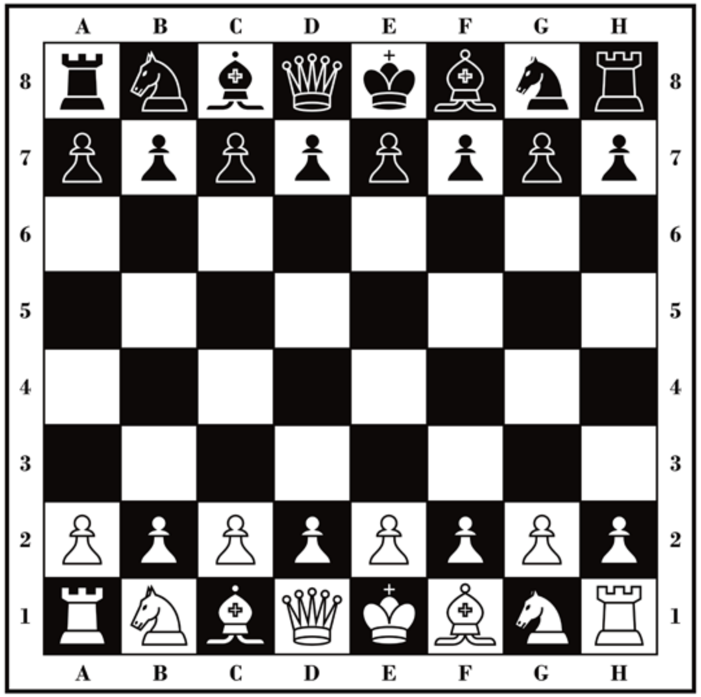
1990

(another application of polynomials)

Chess

An interesting corollary:

Suppose in chess, white can always win in $\leq$ 300 moves.



How can the wizard prove this to you?

SUMMARY SO FAR

NP = 1-round deterministic interaction between a **Prover** and a **Verifier**.

NP + multiple rounds = **NP**

NP + randomization = **NP** (*conjectured as such*)

NP + multiple rounds + randomization = **IP** = **PSPACE**

And now…

Thanks to computer science, a "proof" can be:

- randomized
- interactive
- zero-knowledge
- spot-checkable

Back to Graph Non-isomorphism

Does the verifier gain any insight about why the graphs are not isomorphic?

$\langle G_1, G_2 \rangle$

Pick at random $i \in \{1, 2\}$

Choose a permutation π of vertices at random.

$\pi(G_i)$ →

j ←

Accept if $i = j$

Zero-Knowledge Proofs

The **Verifier** is convinced,
but learns **nothing** about why the graphs are non-isomorphic!

The **Verifier** could have produced the communication transcript by himself, with no help from the **Prover**.

A proof with 0 explanatory content!

Is this useful?

Zero-Knowledge Proofs

Examples of scenarios it would be useful.

Which proofs can be turned into zero-knowledge proofs?

- Does every problem in **IP** have a zero-knowledge interactive proof?

- Does every problem in **NP** have a zero-knowledge interactive proof?

Zero-Knowledge Proofs for **NP**

Does every problem in **NP** have a zero-knowledge **IP**?

Yup! (under plausible cryptographic assumptions)


Goldreich


Micali


Wigderson

1986

Zero-Knowledge Proofs for **NP**

Does every problem in **NP** have a zero-knowledge **IP**?

Yup! (under plausible cryptographic assumptions)

It suffices to show this for your favorite **NP**-complete problem.

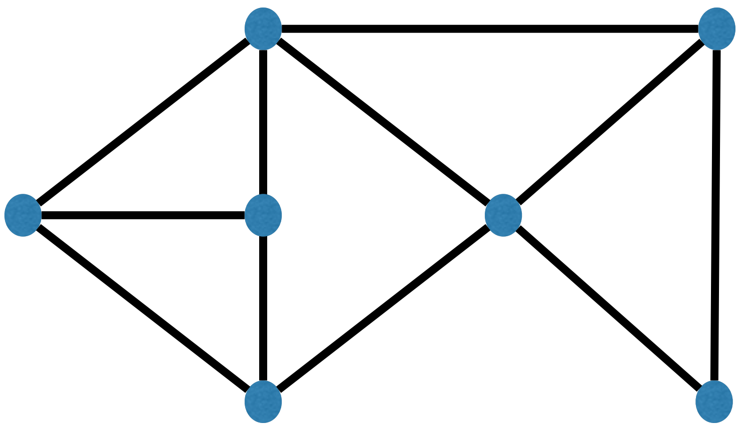
(every problem in **NP** reduces to an **NP**-complete prob.)

We'll pick the Hamiltonian cycle problem.

Zero-Knowledge Proofs for **NP**

Hamiltonian cycle problem

Given an undirected graph:

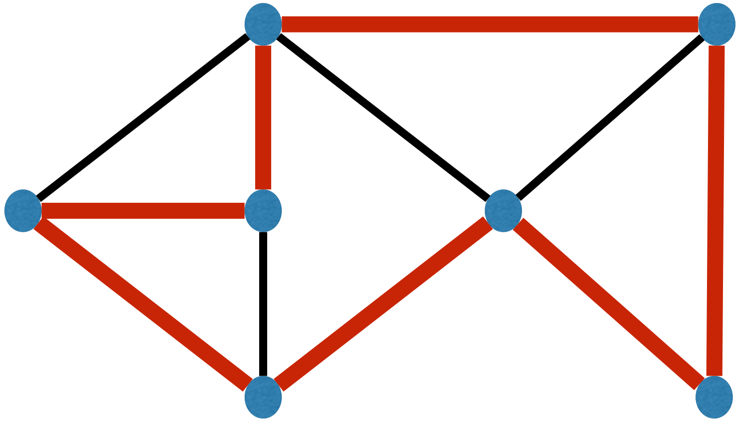


Does it have a cycle that visits every vertex exactly once?

Zero-Knowledge Proofs for **NP**

Hamiltonian cycle problem

Given an undirected graph:



Does it have a cycle that visits every vertex exactly once?

Zero-Knowledge Proofs for **NP**

The protocol

Given undirected graph G

Prover:

Picks randomly a permutation of the vertices π.

Sends $\pi(G)$ in a "locked" way:

- for each pair of vertices, there is a locked bit.
- the bit indicates whether the vertices are connected.

Verifier:

Flips a coin.

If heads, asks **Prover** to show him the Hamiltonian cycle.

If tails, asks **Prover** to unlock everything, and asks for π.

Zero-Knowledge Proofs for **NP**

Completeness ✓

Soundness ✓

Zero-knowledge ✓

All is good if:
the "locked" bits work the way they are meant to work.

- **Verifier** shouldn't be able to unlock them by himself.
- **Prover** shouldn't be able to change bit values.

Can be realized using **bit commitment schemes**.
(assuming Verifier is computationally bounded)

Zero-Knowledge for all?

Does every problem in **IP** = **PSPACE** have a zero-knowledge proof?

Yup!

Ben-Or

Goldreich

Goldwasser

Håstad

Kilian

Micali

Rogaway

1990

"Everything provable is provable in zero-knowledge"

Statistical vs Computational Zero-Knowledge

There is a difference between
- zero-knowledge proof for Graph Non-isomorphism
- zero-knowledge proof for Hamiltonian Cycle

Statistical zero-knowledge:

Verifier doesn't learn anything even if it was computationally unbounded.

Computational zero-knowledge:

Verifier doesn't learn anything assuming it cannot unlock the locks in polynomial time.

Statistical vs Computational Zero-Knowledge

SZK = set of all problems with
statistically zero-knowledge proofs

CZK = set of all problems with
computationally zero-knowledge proofs

IP = PSPACE = CZK

SZK is believed to be much smaller.

In fact, it is believed that it does not contain **NP**-complete problems.

And now…

Thanks to computer science, a "proof" can be:

- randomized
- interactive
- zero-knowledge
- spot-checkable

Spot-Checkable Proofs

Scenario:

I have a proof that $1+1 = 2$.

It is a few hundred pages long.

You have to verify its correctness.

Tiny mistake —> Super annoying to find!

Spot-Checkable Proofs

If only there was a way to "spot-check" the proof:

- check randomly a few bits
- w.h.p. correctly verify the proof

That's a dream that seems too good to be true.

Or is it?

Spot-Checkable Proofs

Question:

Given two graphs $G_0, G_1,$ is there a "spot-checkable" proof that they are non-isomorphic?

Exercise:

Find such a proof that is exponentially long.

Spot-Checkable Proofs

Probabilistically Checkable Proofs (PCP) Theorem:

Every problem in **NP** admits "spot-checkable" proofs of polynomial length.

The verifier can be convinced with high probability by looking only at a ***constant*** number of bits in the proof.

old proof (poly-length) → new proof (poly-length)

tiny local error → *error almost everywhere*

"New shortcut found for long math proofs!"

Spot-Checkable Proofs

Probabilistically Checkable Proofs (PCP) Theorem:

Every problem in **NP** admits "spot-checkable" proofs of polynomial length.

The verifier can be convinced with high probability by looking only at a ***constant*** number of bits in the proof.

1998

Arora-Lund-Motwani-Safra-Sudan-Szegedy

Spot-Checkable Proofs

This theorem is **equivalent** to:

PCP Theorem (version 2):

There is some constant ϵ such that if there is a polynomial-time ϵ-approximation algorithm for MAX-3SAT, then **P** = **NP**.

(It is **NP**-hard to approximate MAX-3SAT within an ϵ factor.)

This is called an "*hardness of approximation*" result.

They are hard to prove!

Spot-Checkable Proofs

PCP Theorem is one of the crowning achievements in CS theory!

Proof is a half a semester course.

Blends together:

P/NP
random walks
expander graphs
polynomials / finite fields
error-correcting codes
Fourier analysis

Summary

Computer science gives a whole new perspective on proofs:

- can be *probabilistic*
- can be *interactive*
- can be *zero-knowledge*
- can be *spot-checkable*
- can be *quantum mechanical*

Summary

old-fashioned proof + deterministic verifier

NP

randomization + interaction

PSPACE

PSPACE = Computationally Zero-Knowledge (CZK)

"Everything provable is provable in zero-knowledge"

(some special problems are in **SZK**)

Summary

PCP Theorem

Old-fashioned proofs can be turned into spot-checkable.
(you only need to check constant number of bits!)

Equivalent to an hardness of approximation result.

Opens the door to many other hardness of approximation results.